DELPHI

Советы программистов

Второе издание,
дополненное

под редакцией В. Озерова

Санкт-Петербург
2004

Delphi. Советы программистов

под редакцией В. Озерова

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Редактор	*В. Овчинников*
Корректура	*С. Журавина*
Верстка	*А. Дорошенко*
Художник	*С. Борин*

Озеров В.
Delphi. Советы программистов. – СПб: Символ-Плюс, 2004. – 976 с., ил.
ISBN 5-93286-037-5

Это издание представляет собой коллекцию ответов на нетрадиционные вопросы программирования на Delphi, нестандартных решений, интересных идей. Примеры рабочего кода, которые создавали многие программисты, собирались более двух лет и охватывают широкий круг вопросов: реализацию математических алгоритмов и работу с функциями Windows API, применение массивов, работу с графикой, а также управление рабочим столом, реестром, папками и файлами Windows, форматирование дискет, взаимодействие с аппаратным обеспечением. Значительное внимание уделено базам данных: таблицам dBASE и Paradox, настройке Delphi для работы с базами данных, подключению сервера Oracle или InterBase, особенностям использования SQL. Те, кто интересуется мультимедиа, найдут в книге советы по работе со звуком.

Рассмотрены создание компонентов с нужными свойствами, а также способы изменения или дополнения уже созданных. Этой теме посвящена самая большая глава сборника. Примеры, имеющие отношение к классам Delphi, помогут понять особенности взаимодействия в MDI- и SDI-приложениях, освоить создание новых форм и управление ими. Сборник также содержит советы по работе с Интернетом и применению механизма OLE для обмена данными в приложениях.

ISBN 5-93286-037-5

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, edit@symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.

Подписано в печать 6.01.2004. Формат 70x100 1/16 . Печать офсетная.
Объем 61 печ. л. Доп. тираж 2000 экз. Заказ N
Отпечатано с диапозитивов в Академической типографии «Наука» РАН
199034, Санкт-Петербург, 9 линия, 12.

Оглавление

Посвящается моей жене и дочке

Введение

Эта книга – не справочник и не учебник, и ее материал рассчитан на подготовленного читателя, хорошо освоившего приемы работы в среде Delphi. Если более точно, то книга рассчитана на программистов среднего или более высокого уровня.

Перед вами коллекция ответов на нетрадиционные вопросы программирования на Delphi, нестандартных решений, хитростей и интересных замыслов. Для практической пользы дела приведены конкретные примеры кода, позволяющие донести идею или полностью ответить на заданный вопрос.

Не удивляйтесь, обнаружив в книге код для Delphi 1 или даже для Turbo Pascal. Сам Pascal практически не изменился, а идеи, реализация и технология живы до сих пор. Для того чтобы найти описание какой-либо функции, можно заглянуть в электронный справочник Delphi, а «Советы» скорее предназначены для поиска общего решения.

Составитель не вносил изменений в листинги авторов и, по возможности, в сопровождающие их описания.

История проекта

Начало «Советам по Delphi» было положено в 1998 году, когда автор стал искать на просторах Интернета и копить многочисленные FAQ, посвященные Delphi. Но пользоваться ими было жутко неудобно, все они имели разный формат, содержали дублирующий текст и претендовали на полный охват всех аспектов программирования. Само собой, чтобы найти нужный ответ, нужно было заглянуть в каждый из файлов. О систематизации и поиске в ту пору задумывались очень немногие, поэтому автору удобнее было после удачного поиска и реализации кода «откладывать» совет в отдельную папочку с именем, совпадающим с тематикой совета.

На одном из этапов был внедрен формат справочной системы MS HTML Help, имеющий развитую систему поиска, закладок и древовидный каталог разделов. На систематизацию материала ушел месяц, но это значительно облегчило дальнейшее добавление новых советов.

Этот опыт завершился успехом, и в 1999 году первая версия «Советов» была выложена в Интернете.[1] Волею пиратов сборник попал на компакт-диск с

[1] Сначала на сервере XOOM по адресу *http://members.xoom.com/kuliba/*, потом на *http://www.webmachine.ru/delphi*.

новой версией Delphi и был растиражирован по всей России (и не только). Была организована система подписки, которая осуществлялась простой отправкой письма автору. Письма с просьбой о подписке идут автору до сих пор. Поскольку количество подписчиков резко возросло и перевалило за тысячу, было принято решение наладить полноценный оффлайн-сервис, в связи с чем примеры пополнялись и выкладывались с периодичностью в две недели, а советы, присланные читателями, обрабатывались и сразу включались в сборник. По всему было видно, что получился оффлайн-клуб программистов, нуждавшихся в заочном общении.

По истечении двух лет, когда количество советов приблизилось к двум тысячам, появилась идея обобщить этот труд, переведя его в иное качество. Для книги был отобраны лучшие примеры, и то, что вы держите в руках, по объему составляет примерно половину исходного материала. Надеюсь, что лучшую половину.

Структура книги

Книга состоит из 13 глав.

- Глава 1 «Алгоритмы». Здесь вы найдете материал и примеры, относящиеся к вопросам преобразования типов, реализации математических алгоритмов и работе с датами, а небольшая часть советов посвящена кодированию информации.
- Глава 2 «API». В главе приведен материал, касающийся работы с функциями Windows API. Не секрет, что многие функции API остались «за бортом» VCL и не имеют соответствующей инкапсуляции. Как изменить системное время? Как завершить работу Windows? Как получить список запущенных приложений? Круг рассматриваемых тем невелик, но в данной главе много примеров.
- Глава 3 «Pascal». Основное внимание в этой главе уделено работе с массивами. Показаны способы создания динамических массивов и использование адресной арифметики. Не оставлена без внимания работа с графической информацией. Как загрузить графический файл или нарисовать ломаную линию, вывести на экран надпись под углом или сделать ее прозрачной?
- Глава 4 «Базы данных». В данной главе собрана информация, необходимая для работы с базами данных. Большая часть материала посвящена работе с уже давно известными Paradox и dBASE. Но рассмотрена масса «тонких» мест, незаслуженно оставленных без внимания в других книгах: особенности доступа, запросы и восстановление информации. После ознакомления с материалом этой главы работа с базами данных станет для читателя обыденным делом.
- Глава 5 «BDE». Как настроить Delphi для работы с базами данных или подключить сервер Oracle или InterBase? Каковы особенности использования SQL? Ответы на эти и другие вопросы можно найти в этой главе.

- Глава 6 «Мультимедиа». Глава не так велика, но дает возможность познакомиться с принципами использования звукового сопровождения в приложениях.
- Глава 7 «Аппаратное обеспечение». Как из приложения определить конфигурацию компьютера, управлять клавиатурой или мышью, воспользоваться модемом? Эти и многие другие вопросы рассмотрены в данной главе.
- Глава 8 «Операционная система». Здесь описывается работа с Windows: управление рабочим столом, реестром, папками и файлами, форматирование дискет и т. д.
- Глава 9 «Компоненты». Это самая большая глава сборника. Здесь читатель узнает, как создавать компоненты с нужными свойствами или внести изменения или дополнения в уже существующие; как исправить ошибки в стандартном наборе или понять принцип и особенности использования компонентов.
- Глава 10 «Классы». Здесь приведен материал, имеющий отношение к классам Delphi. Эта глава поможет понять механизм взаимодействия в MDI- и SDI-приложениях, создание новых форм и управление ими. Обсуждается работа с буфером обмена Clipboard.
- Глава 11 «Интернет». Несколько советов для работы с Интернетом.
- Глава 12 «OLE». В примерах этой небольшой главы рассмотрен механизм обмена информацией в приложениях.
- Глава 13 «Часто задаваемые вопросы (FAQ)». Данные вопросы взяты из различных конференций, связанных с программированием в среде Delphi. Решения специально помещены в отдельную главу, т. к. почти не содержат поясняющего текста и могут быть трудными для начинающих программистов. Но не отбрасывайте их. Воспользуйтесь отладчиком и посмотрите, как они работают. И через некоторое время исчезнет еще одно темное пятно. Этот материал создан группой поддержки разработчиков Borland Developer Support и может быть получен с сайта *community.borland.com* (на английском языке). Печатается с разрешения фирмы Borland. Перевод выполнен Олегом Кулабуховым.

Предупреждение

В книге приведено большое количество исходного кода. Составитель и технические рецензенты проверили большинство решений и трюков и надеются, что у читателя не будет проблем с этими программами.

Тексты наиболее интересных примеров можно скачать по адресу *http://www.symbol.ru./library/delphi_secrets/*.

Составитель не отвечает за последствия применения приведенного кода. Используйте его на свой страх и риск. Если ваш компьютер взорвется из-за какого-нибудь совета, не обвиняйте составителя и не шлите гневные письма.

Составитель не претендует на авторство приведенных в книге решений. Автор или тот, кто прислал решение, указан после совета в квадратных скобках. В разделе «Перечень авторов» приведены их электронные адреса (возможно, на данный момент некоторые из них устарели).

Группа подготовки издания приложила максимум усилий к тому, чтобы установить авторство примеров, приведенных в книге. Мы сожалеем, что для некоторых советов автор не указан, и заранее приносим извинения тем, кто обнаружит в книге свои советы без подписи. Тот, кто прислал данный совет, забыл указать свое имя или источник информации, из которого это решение было почерпнуто. Сообщения о подобных случаях отправляйте составителю по адресу *webmaster@webmachine.ru* для внесения изменений в электронный вариант справочника и следующее издание книги. Многие советы взяты из новостных групп, в которых код передается «из рук в руки», переделывается и усовершенствуется. Для такого «народного творчества» авторство также не указано.

Соглашения, принятые в данной книге

Чтобы читателю легче было ориентироваться в материале, в листингах программ, а также для выделения в тексте программных элементов, имен файлов и папок используется моноширинный шрифт.

От составителя

Занимаясь программированием уже более десяти лет, я пришел к мысли создать книгу, которая реально передавала бы опыт, накопленный сообществом программистов, как новичкам, так и маститым его представителям. Фрагменты информации с примерами разбросаны по Интернету и новостным группам. Большая их часть не систематизирована и не переведена на русский язык, а поиск конкретной информации отнимает неоправданно много времени. Программисты, отчаявшись ответить на свои вопросы самостоятельно, обязательно задавали их через какой-то промежуток времени другим программистам. Данная книга представляет собой подборку наиболее удачных, по мнению составителя, ответов на чаще всего задаваемые вопросы и в этом смысле является результатом многолетнего труда программистского сообщества.

Перечень рассмотренных тем ограничен лишь стремлением автора как-то систематизировать информацию. Реально же никаких рамок не существует, и круг решаемых средствами Delphi задач в среде Windows безграничен. Существует множество изданий и справочников, описывающих технологии и функции системы. Материал же данной книги лежит в несколько иной плоскости. Составитель стремился дать программисту идею, путь решения задачи. Между тем, приведенный код вполне работоспособен и может подсказать решение описываемой проблемы. Другой целью было желание привести решения для наиболее типичных задач, с которыми сталкивается программист. Отправной точкой для такого круга задач служили вопросы, с

завидной периодичностью задаваемые в новостных группах. И, наконец, была предпринята попытка представить подборку наиболее интересных с точки зрения программирования материалов.

Удалась книга или нет, решать вам, читатель. Составитель с благодарностью примет в свой адрес любые замечания и сообщения об ошибках. Это позволит сделать следующее издание книги более полезным и качественным.

Благодарности

Хотя книги часто пишутся писателями в одиночку, эта книга не такая. Это коллективный труд многих программистов, благодаря энтузиазму которых и желанию помочь другим данный сборник появился на свет.

Я выражаю искреннюю благодарность Скорикову Владимиру, без которого этой книги просто не было бы. Из огромного количества материала он выбрал наиболее интересные и достойные публикации, сделал первую вычитку и проверку кода. Его энтузиазм и желание помочь сделали свое дело – книга перед вами. Код Владимира также приведен в книге.

Второй человек, без которого книга не увидела бы свет – Тугаев Олег с его искренним желанием мне помочь. Он потратил все лето только на то, чтобы проверить программный код, который приведен в книге. Его советы также включены в данную книгу. Благодарность за понимание и огромное терпение адресована и его семье.

Отдельное спасибо читателям, приславшим мне письма с указанием на ошибки и правильным кодом. Они были моими самыми дотошными рецензентами.

Я признателен всем, кто помог мне преодолеть трудности и сделать выход этой книги состоявшимся событием. Прежде всего это относится к читателям моих «Советов по Delphi» в электронном виде. Они задавали мне ритм, подбрасывали головоломки и находили новые темы. Это моя команда, мои читатели, и они же авторы большинства советов. Благодарю корпорацию Borland, выпустившую столь замечательный продукт – Delphi.

Благодарю издательство «Символ-Плюс», предложившее мне издать книгу и терпеливо ожидавшее от меня материал.

Своих соседей за долготерпение во время моих ночных работ под недостаточно тихую временами музыку.

Наконец, я хочу поблагодарить всех своих домашних. Мои жена и дочка старались соблюдать тишину и не жаловались, даже когда я работал над книгой в воскресные дни и праздники. Теперь, после выхода книги, я смогу сходить с ними в обещанные театр и зоопарк.

Спасибо моим родителям, переживавшим за мой первый опыт в создании книги.

1

Алгоритмы преобразования

Преобразование шестнадцатеричной строки в целое

Как шестнадцатеричное значение преобразовать в целое?

Строка вида "6A0F" преобразуется в число 27151.


```
function HexToLong(HexStr: string): Integer;
begin
  Result := 0;
  try
    Result := StrToInt('$' + HexStr);
  except
    on E: EConvertError do ShowMessage('Convert error');
    on E: EIntOverflow do ShowMessage('Out of range');
  end
end;
```

Преобразование целого в шестнадцатеричную строку

Как целое значение преобразовать в шестнадцатеричное?

Число 27151 преобразуется в строку вида "6A0F".

```
HexStr := IntToHex(Value, Digits);
```

Преобразование ASCII в шестнадцатеричное значение

Строка представляет собой массив ASCII-символов. Как организовать преобразование по аналогии с Delphi-функциями Ord и Chr? Процедура BytesToHexStr преобразует набор байт [0,1,1,0] в строку «30313130», функция BytesToHex выполняет аналогичные действия, HexStrToBytes выполняет обратное преобразование, а HexBytesToChar переводит шестнадцатеричную строку в строку символов.

```
unit HexStr;

interface

procedure BytesToHexStr(var hHexStr: String; PBA: PByteArray; InputLength: Word);
function BytesToHex(PBA: PByteArray; Size: Word): string;
procedure HexStrToBytes(s: string; PBA: PByteArray);
procedure HexBytesToChar(var Response: String; PBA: PByteArray; InputLength: Word);

implementation

procedure BytesToHexStr(var hHexStr: String; PBA: PByteArray; InputLength: Word);
const
  HexChars: array[0..15] of Char = '0123456789ABCDEF';
var
  i, j: Word;
begin
  SetLength(hHexStr, InputLength * 2);
  j := 1;
  for i:= 0 to InputLength - 1 do begin
    hHexStr[j] := HexChars[PBA[i] shr 4];
    inc(j);
    hHexStr[j] := HexChars[PBA[i] and 15];
    inc(j);
  end;
end;

function BytesToHex(PBA: PByteArray; Size: Word): string;
var
  i: Word;
begin
  Result := '';
  for i := 0 to Size - 1 do
    Result := Result + IntToHex(Ord(PBA[i]), 2);
end;
```

```
procedure HexStrToBytes(s: string; PBA: PByteArray);
var
  i, j: Word;
begin
  j := 1;
  for i := 0 to (Length(s) div 2) - 1 do begin
    PBA[i] := StrToInt('$' + Copy(s, j, 2));
    inc(j, 2);
  end;
end;

procedure HexBytesToChar(var Response: String; PBA: PByteArray; InputLength: Word);
var
  i: Word;
  c: byte;
begin
  SetLength(Response, InputLength);
  for i := 0 to (InputLength - 1) do begin
    c := PBA[i] and $f;
    if c > 9 then Inc(c, $37) else Inc(c, $30);
    Response[i+1] := Chr(c);
  end;
end;

end.
```

Преобразование двоичного числа в десятичное

Как преобразовать двоичное значение в десятичное?

Решение 1

```
function Base10(Base2: Integer): Integer; assembler;
asm
  cmp   eax, 100000000  // проверка максимального значения
  jb    @1              // значение в пределах допустимого
  mov   eax, -1         // флаг ошибки
  jmp   @exit           // выход, если -1
@1:
  push  ebx             // сохранение регистров
  push  esi
  xor   esi, esi        // результат = 0
  mov   ebx, 10         // вычисление десятичного логарифма
  mov   ecx, 8          // преобразование по формуле 10^8-1
@2:
  mov   edx, 0          // удаление разницы
  div   ebx             // eax - целочисленное деление на 10,
                        // edx - остаток от деления на 10
  add   esi, edx        // результат = результат + разность[I]
  ror   esi, 4          // перемещение разряда
  loop  @2              // цикл для всех 8 разрядов
```

```
    mov   eax, esi        // результат функции
    pop   esi             // восстановление регистров
    pop   ebx
  @exit:
  end;
```

Решение 2

Функция IntToBin выполнит преобразование десятичного числа в двоичное, а BinToInt – обратное преобразование.

```
function IntToBin(Value: Longint; Size: Integer): String;
var
  i: Integer;
begin
  Result := '';
  for i := Size downto 0 do begin
    if Value and (1 shl i) <> 0 then Result := Result + '1'
    else Result := Result + '0';
  end;
end;

function IntToBin(Value: Longint; Size: Integer): String;
var
  i: Integer;
begin
  Result := '';
  for i := Size downto 0 do begin
    if Value and (1 shl i) <> 0 then Result := Result + '1'
    else Result := Result + '0';
  end;
end;
```

Преобразование Comp в String

Были какие-то разговоры о том, что тип данных Comp *является каким-то ущербным, некорректным и что даже не существует подпрограмм, осуществляющих конвертацию* Comp *в* String *и обратно.*

Предлагаемый модуль включает в себя следующие функции: CompToStr, CompToHex, StrToComp, а также CMod и CDiv, которые являются вспомогательными и представляют собой реализацию функций MOD и DIV для типа Comp.

```
unit Compfunc;
interface

type
  CompAsTwoLongs = record
    LoL, HiL: LongInt;
  end;
```

```
const
  Two32TL: CompAsTwoLongs = (LoL:0; HiL:1);

var
  Two32: Comp ABSOLUTE Two32TL;
{ Некоторые операции могут окончиться неудачей, если значение
  находится вблизи границы диапазона Comp }

const
  MaxCompTL: CompAsTwoLongs = (LoL: $FFFFFFF0; HiL: $7FFFFFFF);

var
  MaxComp: Comp ABSOLUTE MaxCompTL;

function CMod(Divisor, Dividend: Comp): Comp;
function CDiv(Divisor: Comp; Dividend: LongInt): Comp;
function CompToStr(C: Comp): String;
function CompToHex(C: Comp; Len: Integer): String;
function StrToComp(const S: String): Comp;

implementation

uses
  SysUtils;

function CMod(Divisor, Dividend: Comp): Comp;
var
  Temp: Comp;
begin
{ Примечание: Оператор / для типа Comps ОКРУГЛЯЕТ результат,
  а не отбрасывает десятичные знаки }
  Temp := Divisor / Dividend;
  Temp := Temp * Dividend;
  Result := Divisor - Temp;
  if Result < 0 then Result := Result + Dividend;
end;

function CDiv(Divisor: Comp; Dividend: LongInt): Comp;
begin
  Result := Divisor / Dividend;
  if Result * Dividend > Divisor then Result := Result - 1;
end;

function CompToStr(C: Comp): String;
var
  Posn: Integer;
begin
  if C > MaxComp then
    Raise ERangeError.Create('Comp слишком велик для преобразования в string');
  if C < 0 then Result := '-' + CompToStr(-C)
```

```
  else begin
      Result := '';
      Posn := 0;
      while True do begin
        Result := Char(Round($30 + CMod(C, 10))) + Result;
        if C < 10 then break;
        C := CDiv(C, 10);
        Inc(Posn);
        if Posn mod 3 = 0 then Result := ',' + Result;
      end;
    end;
  end;

  function CompToHex(C: Comp; Len: Integer): String;
  begin
    if (CompAsTwoLongs(C).HiL = 0) and (Len <= 8) then
      Result := IntToHex(CompAsTwoLongs(C).LoL, Len)
    else
      Result := IntToHex(CompAsTwoLongs(C).HiL, Len - 8)
              + IntToHex(CompAsTwoLongs(C).LoL, 8)
  end;

  function StrToComp(const S: String): Comp;
  var
    Posn: Integer;
  begin
    if S[1] = '-' then Result := -StrToComp(Copy(S, 2, Length(S) - 1))
    else if S[1] = '$' then      { Шестнадцатеричная строка }
      try
        if Length(S) > 9 then begin
  { Если строка некорректна, исключение сгенерирует StrToInt }
          Result := StrToInt('$' + Copy(S, Length(S) - 7, 8));
          if Result > 10 then Result := Result + Two32;
  { Если строка некорректна, исключение сгенерирует StrToInt }
          CompAsTwoLongs(Result).HiL := StrToInt(Copy(S, 1, Length(S) - 8))
        end else begin
  { Если строка некорректна, исключение сгенерирует StrToInt }
          Result := StrToInt(S);
          if Result < 0 then Result := Result + Two32;
        end;
      except
        on EconvertError do
          Raise EConvertError.Create(S + ' некорректный Comp');
      end else begin
        Posn := 1;
        Result := 0;
        while Posn <= Length(S) do
          case S[Posn] of
              ',':  Inc(Posn);
           '0'..'9':  begin
                      Result := Result * 10 + Ord(S[Posn]) - $30;
```

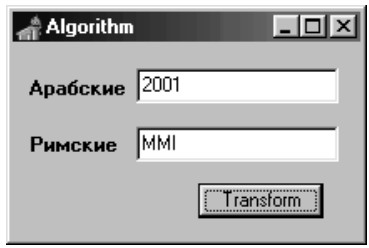
```
                Inc(Posn);
              end;
          else
            Raise EConvertError.Create(S +' некорректный Comp');
        end;
    end;
end;

end.
```

[News Group]

Преобразование арабских чисел в римские

Как преобразовать арабские числа в римские?

Функция получает в качестве параметра любую десятичную величину и возвращает результат в виде строки, содержащей римские цифры.




```
function TForm1.DecToRoman(Decimal: Integer): String;
const
 Romans: array[1..13] of String = ('I', 'IV', 'V', 'IX', 'X', 'XL', 'L',
                                   'XC', 'C', 'CD', 'D', 'CM', 'M');
 Arabics: array[1..13] of Integer = (1, 4, 5, 9, 10, 40, 50, 90, 100, 400, 500, 900, 1000);
var
  i: Integer;
begin
  result := '';
  for i := 13 downto 1 do
    while (Decimal >= Arabics[i]) do begin
      Decimal := Decimal - Arabics[i];
      result := result + Romans[i];
    end;
end;
```

Преобразование в EBCDIC

Как перекодировать строку?

Функция конвертирует любую строку. Можете доработать ее, для того чтобы она могла преобразовывать другие типы данных. Но если вам нужны

дополнительные преобразования и обработка данных, то стоит задуматься о приобретении специализированного программного обеспечения...

```
const
  a2e: array [0..255] of byte =
  (000, 001, 002, 003, 055, 045, 046, 047, 022, 005, 037, 011, 012, 013, 014, 159,
   016, 017, 018, 019, 182, 181, 050, 038, 024, 025, 063, 039, 028, 029, 030, 031,
   064, 090, 127, 123, 091, 108, 080, 125, 077, 093, 092, 078, 107, 096, 075, 097,
   240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 122, 094, 076, 126, 110, 111,
   124, 193, 194, 195, 196, 197, 198, 199, 200, 201, 209, 210, 211, 212, 213, 214,
   215, 216, 217, 226, 227, 228, 229, 230, 231, 232, 233, 173, 224, 189, 095, 109,
   121, 129, 130, 131, 132, 133, 134, 135, 136, 137, 145, 146, 147, 148, 149, 150,
   151, 152, 153, 162, 163, 164, 165, 166, 167, 168, 169, 192, 106, 208, 161, 007,
   104, 220, 081, 066, 067, 068, 071, 072, 082, 083, 084, 087, 086, 088, 099, 103,
   113, 156, 158, 203, 204, 205, 219, 221, 224, 236, 252, 176, 177, 178, 062, 180,
   069, 085, 206, 222, 073, 105, 154, 155, 171, 015, 186, 184, 183, 170, 138, 139,
   060, 061, 098, 079, 100, 101, 102, 032, 033, 034, 112, 035, 114, 115, 116, 190,
   118, 119, 120, 128, 036, 021, 140, 141, 142, 065, 006, 023, 040, 041, 157, 042,
   043, 044, 009, 010, 172, 074, 174, 175, 027, 048, 049, 250, 026, 051, 052, 053,
   054, 089, 008, 056, 188, 057, 160, 191, 202, 058, 254, 059, 004, 207, 218, 020,
   225, 143, 070, 117, 253, 235, 238, 237, 144, 239, 179, 251, 185, 234, 187, 255);

procedure StringA2E(var StringToConvert: String);
var
  Loop: integer;
begin
  for Loop := 1 to Length(StringToConvert) do
    StringToConvert[Loop] := Char(a2e[Ord(StringToConvert[Loop])]);
end;
```

[News Group]

Добавление лидирующих символов

Как в начало строки вставить символ? Количество вставляемых символов может быть различным.

Если необходимо в начало строки вставить определенный символ, например, преобразовать «1010» в «0001010», воспользуйтесь следующей функцией:

```
function PadL(s_InStr: string; i_Wide: integer; c_Chr: char): string;
begin
  while Length(s_InStr) < i_Wide do s_InStr := c_Chr + s_InStr;
  Result := s_InStr;
end;
```

[Good_mag]

Преобразование BMP в ICO

Как файл формата BMP преобразовать в формат пиктограмм – ICO?

Нужно создать две маски типа `Bitmap` – маску `AND` и маску `XOR`. Можно также передать дескрипторы масок `Handle` в функцию `CreateIconIndirect()` и использовать их в приложении:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  IconSizeX: integer;
  IconSizeY: integer;
  AndMask: TBitmap;
  XorMask: TBitmap;
  IconInfo: TIconInfo;
  Icon: TIcon;

begin
{ Получаем размеры пиктограммы }
  IconSizeX := GetSystemMetrics(SM_CXICON);
  IconSizeY := GetSystemMetrics(SM_CYICON);
{ создаем маску "And" }
  AndMask := TBitmap.Create;
  AndMask.Monochrome := true;
  AndMask.Width := IconSizeX;
  AndMask.Height := IconSizeY;
{ рисуем на маске "And" }
  AndMask.Canvas.Brush.Color := clWhite;
  AndMask.Canvas.FillRect(Rect(0, 0, IconSizeX, IconSizeY));
  AndMask.Canvas.Brush.Color := clBlack;
  AndMask.Canvas.Ellipse(4, 4, IconSizeX - 4, IconSizeY - 4);
{ Рисуем для проверки }
  Form1.Canvas.Draw(IconSizeX * 2, IconSizeY, AndMask);
{ Создаем маску "Xor" }
  XorMask := TBitmap.Create;
  XorMask.Width := IconSizeX;
  XorMask.Height := IconSizeY;
{ Рисуем на маске "Xor" }
  XorMask.Canvas.Brush.Color := ClBlack;
  XorMask.Canvas.FillRect(Rect(0, 0, IconSizeX, IconSizeY));
  XorMask.Canvas.Pen.Color := clRed;
  XorMask.Canvas.Brush.Color := clRed;
  XorMask.Canvas.Ellipse(4, 4, IconSizeX - 4, IconSizeY - 4);
{ Рисуем для проверки }
  Form1.Canvas.Draw(IconSizeX * 4, IconSizeY, XorMask);
{ Создаем пиктограмму }
  Icon := TIcon.Create;
  IconInfo.fIcon := true;
  IconInfo.xHotspot := 0;
  IconInfo.yHotspot := 0;
  IconInfo.hbmMask := AndMask.Handle;
```

```
  IconInfo.hbmColor := XorMask.Handle;
  Icon.Handle := CreateIconIndirect(IconInfo);
{ Удаляем временные bitmap }
  AndMask.Free;
  XorMask.Free;
{ Рисуем для проверки }
  Form1.Canvas.Draw(IconSizeX * 6, IconSizeY, Icon);
{ Присваиваем пиктограмму приложению }
  Application.Icon := Icon;
{ Заставляем перерисоваться }
  InvalidateRect(Application.Handle, nil, true);
{ Освобождаем пиктограмму }
  Icon.Free;
end;
```

[Vozny Alexander]

Преобразование ICO в BMP

Нужно преобразовать пиктограмму (ICO) в формат BMP. Как это выполнить с помощью Delphi?

Решение 1

Воспользуйтесь небольшой процедурой:

```
procedure ConvertIcoToBmp(Icon: TIcon; FilePath: string);
var
  Bitmap: TBitmap;

begin
  Bitmap := TBitmap.Create;
  Icon.LoadFromFile(FilePath + '.ICO');
  Bitmap.Width := Icon.Width;
  Bitmap.Height := Icon.Height;
  Bitmap.Canvas.Draw(0, 0, Icon);
  Bitmap.SaveToFile(FilePath + '.BMP');
  Bitmap.Free;
end;
```

Решение 2

Чтобы преобразовать `Icon` в `Bitmap`, воспользуйтесь классом `TImageList`. Для обратного преобразования замените метод `AddIcon` на `Add` и метод `GetBitmap` – на `GetIcon`.

```
function Icon2Bitmap(Icon: TIcon): TBitmap;
begin
  with TImageList.Create(nil) do begin
    AddIcon (Icon);
    Result := TBitmap.Create;
```

```
        GetBitmap (0, Result);
    Free;
  end;
end;
```

[Astafiev]

Примечание

Во втором решении необходимо определиться с размерами получаемого изображения.

Преобразование BMP в JPEG

Как преобразовать файл формата BMP в формат JPEG?

Image1 – компонент TImage, содержащий растровое изображение. Используйте следующий фрагмент кода для конвертации изображения в файл формата JPEG:

```
procedure ConvertBmpToJpeg(Image1: TImage; FilePath: string);
var
  MyJpeg: TJpegImage;
begin
  MyJpeg := TJpegImage.Create;
{ Чтение изображения из файла }
  Image1.Picture.LoadFromFile(FilePath + '.BMP');
{ Назначение изображения объекту }
  MyJpeg.Assign(Image1.Picture.Bitmap);

{ Сохранение изображения на диске }
  MyJpeg.SaveToFile(FilePath + '.JPG');
  MyJpeg.Free;
end;
```

Примечание

Не забудьте добавить в uses *модуль* jpeg*, в котором описан тип* TjpegImage.

Арифметика времени

Как правильно работать с переменными, использующими время?

Работа с временными величинами в Delphi очень проста, если воспользоваться встроенными функциями преобразования. Определите глобальные переменные Hour, Minute, Second и инициализируйте их следующим образом:

```
Hour := EncodeTime(1, 0, 0, 0);
Minute := EncodeTime(0, 1, 0, 0);
Second := EncodeTime(0, 0, 1, 0);
```

Если вы предпочитаете константы, определите их следующим образом:

```
Hour = 3600000 / MSecsPerDay;
Minute = 60000 / MSecsPerDay;
Second = 1000 / MSecsPerDay;
```

Теперь, для того чтобы добавить 240 минут к переменной TDateTime, просто напишите (не забывайте о типах переменных):

```
T := T + 240 * Minute;
```

Примечание

В первом случае переменные должны иметь тип TTime *(подключите модуль* SysUtils*).*

Арифметика дат

Кто-нибудь пробовал написать функцию, возвращающую для определенной даты день недели?

Delphi содержит в своем арсенале функцию DayOfWeek, возвращающую целочисленный результат в диапазоне от 1 до 7:

```
function DayOfWeekStr(S: TDateTime): string;
begin

  case DayOfWeek(S) of
    1: Result := 'Воскресенье';
    2: Result := 'Понедельник';
    3: Result := 'Вторник';
    4: Result := 'Среда';
    5: Result := 'Четверг';
    6: Result := 'Пятница';
    7: Result := 'Суббота';
  end;

end;
```

[Иванов Андрей]

Номер месяца по его имени

Как получить номер месяца, если известно его название?

Воспользуйтесь циклом обхода элементов глобального массива LongMonthNames:

```
function GetMonthNumber(Month: String): Integer;
begin
  for Result := 1 to 12 do
```

```
      if Month = LongMonthNames[Result] then Exit;
  Result := 0;
end;
```

Примечание

Функция `GetMonthNumber` *возвращает номера месяцев от 1 до 12. Название месяца задается в формате текущей локализации Windows, например «Август», «Мау». При неправильном названии месяца возвращается значение 0.*

Получение элемента даты

Как из даты выделить нужный элемент?

Используйте универсальную функцию возврата значения элемента даты (год, месяц, день, квартал):

```
function RetDate(inDate: TDateTime; inTip: integer): integer;
var
  xYear, xMonth, xDay: word;
begin
  Result := 0;
  DecodeDate(inDate, xYear, xMonth, xDay);

  case inTip of
    1:  Result := xYear;                  // год
    2:  Result := xMonth;                 // месяц
    3:  Result := xDay;                   // день
    4:  if xMonth < 4 then Result := 1    // квартал
        else if xMonth < 7 then Result := 2
        else if xMonth < 10 then Result := 3
        else Result := 4;
  end;
end;
```

[Галимарзанов Фанис]

Год четырьмя цифрами

Необходимо отображать год четырьмя цифрами. Как это можно сделать?

Данный код преобразует дату из короткого представления `DD/MM/YY` в формат длинной даты `DD/MM/YYYY`.

Если `DD/MM/YY` – входное поле, а `DD/MM/YYYY` – поле базы данных, то приведенный выше код можно использовать при задании другого формата даты, с его соответствующим переопределением в панели управления.

```
LongDate := FormatDateTime('dd/mm/yyyy', StrToDate(ShortDate));
```

Преобразование даты в количество секунд

Мне нужно сравнить даты с точностью до секунд. Как это можно сделать?

Функция EncodeDate возвращает объект TDateTime, который определен как double. Это позволяет работать с датами и временем как с обычными числами – складывать, вычитать и т. д. При этом дата хранится в целой части числа, а время в дробной. Берем две даты (TDateTime), находим разность, а полученный результат преобразуем с помощью DateTimeToStr или чего-то подобного. Если же отбросить целую часть разницы двух чисел, то получим разницу в пределах одних суток. Все зависит от вашей фантазии...

Вычисление даты Пасхи

Как определить дату Пасхи?

Решение 1

Воспользуйтесь функцией CalcEaster. Далее приводится два варианта ее применения.

```
function CalcEaster(Year: Word): String;
var
  B, D, E, Q: Integer;
  GF: String;
begin
  B := 225 - 11 * (Year Mod 19);
  D := ((B - 21) Mod 30) + 21;
  if D > 48 then Dec(D);
  E := (Year + (Year Div 4) + D + 1) Mod 7;
  Q := D + 7 - E;
  if Q < 32 then begin
    if ShortDateFormat[1] = 'd' then
      Result := IntToStr(Q) + '/3/' + IntToStr(Year)
    else
      Result := '3/' + IntToStr(Q) + '/' + IntToStr(Year);
  end
  else begin
    if ShortDateFormat[1] = 'd' then
      Result := IntToStr(Q - 31) + '/4/' + IntToStr(Year)
    else
      Result := '4/' + IntToStr(Q - 31) + '/' + IntToStr(Year);
  end;
{ вычисление страстной пятницы }
  if Q < 32 then begin
    if ShortDateFormat[1] = 'd' then
      GF := IntToStr(Q - 2) + '/3/' + IntToStr(Year)
    else
      GF := '3/' + IntToStr(Q - 2) + '/' + IntToStr(Year);
  end
```

```
  else begin
    if ShortDateFormat[1]= 'd' then
      GF := IntToStr(Q - 31 - 2) + '/4/' + IntToStr(Year)
    else
      GF := '4/' + IntToStr(Q - 31 - 2) + '/' + IntToStr(Year);
  end;
end;
```

Решение 2

```
function Easter(Year: Integer): TDateTime;
{ Вычисляет и возвращает день Пасхи определенного года.
  Скорректировано для предотвращения переполнения целых, если передан год >= 6554}
var
  nMonth, nDay, nMoon, nEpact, nSunday, nGold, nCent, nCorx, nCorz: Integer;
begin
{ Номер Золотого Года в 19-летнем метоническом цикле: }
  nGold := (Year mod 19) + 1;
{ Вычисляем столетие: }
  nCent := (Year div 100) + 1;
{ Количество лет, в течение которых отслеживаются високосные года...
  для синхронизации с движением солнца: }
  nCorx := (3 * nCent) div 4 - 12;
{ Специальная коррекция для синхронизации Пасхи с орбитой луны: }
  nCorz := (8 * nCent + 5) div 25 - 5;
{ Находим воскресенье: }
  nSunday := (Longint(5) * Year) div 4 - nCorx - 10;
{ Предохраняем переполнение года за отметку 6554.
  Устанавливаем Epact - определяем момент полной луны: }
  nEpact := (11 * nGold + 20 + nCorz - nCorx) mod 30;
  if nEpact < 0 then nEpact := nEpact + 30;
  if ((nEpact = 25) and (nGold > 11)) or (nEpact = 24) then nEpact := nEpact + 1;
{ Ищем полную луну: }
  nMoon := 44 - nEpact;
  if nMoon < 21 then nMoon := nMoon + 30;
{ Позиционируем на воскресенье: }
  nMoon := nMoon + 7 - ((nSunday + nMoon) mod 7);
  if nMoon > 31 then begin
    nMonth := 4;
    nDay := nMoon - 31;
  end
  else begin
    nMonth := 3;
    nDay := nMoon;
  end;
  Easter := EncodeDate(Year, nMonth, nDay);
end;
```

Использование DateTime в DBGrid

При отображении `TDateTimeField` *в* `DBGrid` *с форматированием hh:mm (для показа только времени), любая попытка изменения времени приводит (при передаче данных) к ошибке примерно такого содержания: «07:00 is not a valid DateTime». Как переслать данные в виде: «Trunc(oldDateTimevalue) + StrToTime(displaytext)»?*

Следующий обработчик событий – `TDateTimeField.OnSetText` – не слишком элегантен, но он работает.

Предположим, что имеется маска редактирования, допускающая формат `hh:mm` или `hh:mm:ss`. Тогда процедура будет иметь следующий вид:

```
procedure TForm1.Table1Date1SetText(Sender: TField; const Text: String);
var
  d: TDateTime;
  t: string;
begin
  t := Text;
  with Sender as TDateTimeField do begin
    if IsNull then d := SysUtils.Date
    else d := AsDateTime;
    AsDateTime := StrToDateTime(Copy(DateToStr(d), 1, 10) + ' ' + t);
  end;
end;
```

Примечание

Функция `Copy` *как раз и формирует постоянную дату (в формате* `dd/mm/yyyy`*), которая автоматически вводится в поле,* `t` *– вводимое время.*

[News Group]

Вычисление восхода и захода солнца и луны

Воспользуйтесь готовым проектом, состоящим из трех модулей.

Модуль `sunproject.dpr`:

```
program sunproject;

uses
  Forms,
  main in 'main.pas' {Sun};

{$R *.RES}
begin
  Application.Initialize;
  Application.Title := 'Sun';
  Application.CreateForm(TSun, Sun);
  Application.Run;
end.
```

Модуль main.dfm:

```
object Sun: TSun
  Left = 210
  Top = 106
  BorderIcons = [biSystemMenu, biMinimize]
  BorderStyle = bsSingle
  Caption = 'Sun'
  ClientHeight = 257
  ClientWidth = 299
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  OldCreateOrder = False
  Position = poDesktopCenter
  OnCreate = CreateForm
  PixelsPerInch = 96
  TextHeight = 13
  object GroupBoxInput: TGroupBox
    Left = 4
    Top = 4
    Width = 173
    Height = 93
    Caption = ' Ввод '
    TabOrder = 0
    object LabelLongitude: TLabel
      Left = 35
      Top = 44
      Width = 78
      Height = 13
      Alignment = taRightJustify
      Caption = 'Долгота (град):'
    end
    object LabelTimeZone: TLabel
      Left = 13
      Top = 68
      Width = 100
      Height = 13
      Alignment = taRightJustify
      Caption = 'Часовая зона (час):'
    end
    object LabelAtitude: TLabel
      Left = 40
      Top = 20
      Width = 73
      Height = 13
      Alignment = taRightJustify
      Caption = 'Широта (град):'
    end
```

```
    object EditB5: TEdit
      Tag = 1
      Left = 120
      Top = 16
      Width = 37
      Height = 21
      TabOrder = 0
      Text = '0'
    end
    object EditL5: TEdit
      Tag = 2
      Left = 120
      Top = 40
      Width = 37
      Height = 21
      TabOrder = 1
      Text = '0'
    end
    object EditH: TEdit
      Tag = 3
      Left = 120
      Top = 64
      Width = 37
      Height = 21
      TabOrder = 2
      Text = '0'
    end
  end
  object GroupBoxCalendar: TGroupBox
    Left = 184
    Top = 4
    Width = 109
      Height = 93
      Caption = ' Календарь '
      TabOrder = 1
      object LabelD: TLabel
        Left = 19
        Top = 20
        Width = 30
        Height = 13
        Alignment = taRightJustify
        Caption = 'День:'
      end
      object LabelM: TLabel
        Left = 13
        Top = 44
        Width = 36
        Height = 13
        Alignment = taRightJustify
        Caption = 'Месяц:'
      end
```

```
    object LabelY: TLabel
      Left = 28
      Top = 68
      Width = 21
      Height = 13
      Alignment = taRightJustify
      Caption = 'Год:'
    end
    object EditD: TEdit
      Tag = 1
      Left = 56
      Top = 16
      Width = 37
      Height = 21
      TabOrder = 0
      Text = '0'
    end
    object EditM: TEdit
      Tag = 2
      Left = 56
      Top = 40
      Width = 37
      Height = 21
      TabOrder = 1
      Text = '0'
    end
    object EditY: TEdit
      Tag = 3
      Left = 56
      Top = 64
      Width = 37
      Height = 21
      TabOrder = 2
      Text = '0'
    end
  end
  object ButtonCalc: TButton
    Left = 12
    Top = 227
    Width = 169
    Height = 25
    Caption = '&Вычислить'
    TabOrder = 2
    OnClick = ButtonCalcClick
  end
  object ListBox: TListBox
    Left = 4
    Top = 104
    Width = 289
    Height = 117
    ItemHeight = 13
```

```
      TabOrder = 3
    end
    object ButtonClear: TButton
      Left = 192
      Top = 227
      Width = 91
      Height = 25
      Caption = '&Очистить'
      TabOrder = 4
      OnClick = ButtonClearClick
    end
  end
end
```

Модуль main.pas:

```
unit main;

interface

uses
  Windows, Messages, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls;

type
  TSun = class(TForm)
    GroupBoxInput: TGroupBox;
    LabelLongitude: TLabel;
    EditB5: TEdit;
    EditL5: TEdit;
    LabelTimeZone: TLabel;
    EditH: TEdit;
    GroupBoxCalendar: TGroupBox;
    LabelD: TLabel;
    LabelM: TLabel;
    LabelY: TLabel;
    EditD: TEdit;
    EditM: TEdit;
    EditY: TEdit;
    ButtonCalc: TButton;
    ListBox: TListBox;
    ButtonClear: TButton;
    LabelAtitude: TLabel;
    procedure Calendar;                      // Календарь
    procedure GetTimeZone;                   // Получение часового пояса
    procedure PosOfSun;                      // Получаем положение солнца
    procedure OutInform;                     // Процедура вывода информации
    procedure PossibleEvents(Hour: integer); // Возможные события
                                             // на полученный час
    procedure GetDate;                       // Получить значения даты
    procedure GetInput;                      // Получить значения широты,...
    procedure ButtonCalcClick(Sender: TObject);
```

```
    procedure CreateForm(Sender: TObject);
    procedure ButtonClearClick(Sender: TObject);
  private
    function Sgn(Value: Double): integer;    // Сигнум
  public
end;

var
  Sun: TSun;
  st: string;
  aA, aD: array [1 .. 2] of double;
  B5, H, M, Y, J3, H3, M3, C0: integer;
  D, L5, Z, Z0, Z1, A5, D5, R5, T, T0, TT, T3, L0, L2: double;
  H0, H1, H2, H7, N7, D7, M8, W8, A, B, A0, D0, A2, D1, D2, DA, DD: double;
  E, F, J, S, C, P, L, G, V, U, W, V0, V1, V2, AZ: double;

const
  P2 = Pi * 2;                      // 2 * Pi
  DR = Pi / 180;                    // Радиан на градус
  K1 = 15 * DR * 1.0027379;

implementation

{$R *.DFM}

uses
  SysUtils;

function TSun.Sgn(Value: Double): integer;
begin
  if Value = 0 then } Result := 0;
  if Value > 0 then Result := 1;
  if Value < 0 then Result := -1;
end;

procedure TSun.Calendar;
begin
  G := 1;
  if Y < 1583 then G := 0;
  D1 := Trunc(D);
  F := D - D1 - 0.5;
  J := -Trunc(7 * (Trunc((M + 9) / 12) + Y) / 4);
  if G = 1 then begin
    S := Sgn(M - 9);
    A := Abs(M - 9);
    J3 := Trunc(Y + S * Trunc(A / 7));
    J3 := -Trunc((Trunc(J3 / 100) + 1) * 3 / 4);
  end;
  J := J + Trunc(275 * M / 9) + D1 + G * J3;
  J := J + 1721027 + 2 * G + 367 * Y;
  if F >= 0 then Exit;
```

```
  F := F + 1;
  J := J - 1;
end;

procedure TSun.GetTimeZone;
begin
  T0 := T / 36525;
  S := 24110.5 + 8640184.813 * T0;
  S := S + 86636.6 * Z0 + 86400 * L5;
  S := S / 86400;
  S := S - Trunc(S);
  T0 := S * 360 * DR;
end;

procedure TSun.PosOfSun;
begin
// Фундаментальные константы (Van Flandern & Pulkkinen, 1979)
  L := 0.779072 + 0.00273790931 * T;
  G := 0.993126 + 0.0027377785 * T;
  L := L - Trunc(L);
  G := G - Trunc(G);
  L := L * P2;
  G := G * P2;
  V := 0.39785 * Sin(L);
  V := V - 0.01000 * Sin(L - G);
  V := V + 0.00333 * Sin(L + G);
  V := V - 0.00021 * TT * Sin(L);
  U := 1 - 0.03349 * Cos(G);
  U := U - 0.00014 * Cos(2 * L);
  U := U + 0.00008 * Cos(L);
  W := -0.00010 - 0.04129 * Sin(2 * L);
  W := W + 0.03211 * Sin(G);
  W := W + 0.00104 * Sin(2 * L - G);
  W := W - 0.00035 * Sin(2 * L + G);
  W := W - 0.00008 * TT * Sin(G);
// Вычисление солнечных координат
  S := W / Sqrt(U - V * V);
  A5 := L + ArcTan(S / Sqrt(1 - S * S));
  S := V / Sqrt(U);
  D5 := ArcTan(S / Sqrt(1 - S * S));
  R5 := 1.00021 * Sqrt(U);
end;

procedure TSun.PossibleEvents(Hour: integer);
var
  num: string;
begin
  st := '';
  L0 := T0 + Hour * K1;
  L2 := L0 + K1;
  H0 := L0 - A0;
```

```
  H2 := L2 - A2;
  H1 := (H2 + H0) / 2;                        // Часовой угол,
  D1 := (D2 + D0) / 2;                        // наклон в получасе
  if Hour <= 0 then V0 := S * Sin(D0) + C * Cos(D0) * Cos(H0) - Z;
  V2 := S * Sin(D2) + C * Cos(D2) * Cos(H2) - Z;
  if Sgn(V0) = Sgn(V2) then Exit;
  V1 := S * Sin(D1) + C * Cos(D1) * Cos(H1) - Z;
  A:= 2 * V2 - 4 * V1 + 2 * V0;
  B := 4 * V1 - 3 * V0 - V2;
  D := B * B - 4 * A * V0;
  if D < 0 then Exit;
  D := Sqrt(D);
  if (V0 < 0) and (V2 > 0) then st := st + 'Восход солнца в ';
  if (V0 < 0) and (V2 > 0) then M8 := 1;
  if (V0 > 0) and (V2 < 0) then st := st + 'Заход солнца в ';
  if (V0 > 0) and (V2 < 0) then W8 := 1;
  E := (-B + D) / (2 * A);
  if (E > 1) or (E < 0) then E := (-B - D) / (2 * A);
  T3 := Hour + E + 1 / 120;                   // Округление
  H3 := Trunc(T3);
  M3 := Trunc((T3 - H3) * 60);
  Str(H3:2, num);
  st := st + num + ':';
  Str(M3:2, num);
  st := st + num;
  H7 := H0 + E * (H2 - H0);
  N7 := -Cos(D1) * Sin(H7);
  D7 := C * Sin(D1) - S * Cos(D1) * COS(H7);
  AZ := ArcTan(N7 / D7) / DR;
  if (D7 < 0) then AZ := AZ + 180;
  if (AZ < 0) then AZ := AZ + 360;
  if (AZ > 360) then AZ := AZ - 360;
  Str(AZ:4:1, num);
  st := st + ', азимут ' + num;
end;

procedure TSun.OutInform;
begin
  if (M8 = 0) and (W8 = 0) then begin
    if V2 < 0 then ListBox.Items.Add('Солнце заходит весь день ');
    if V2 > 0 then ListBox.Items.Add('Солнце восходит весь день ');
  end else begin
    if M8 = 0 then ListBox.Items.Add('В этот день солнце не восходит ');
    if W8 = 0 then ListBox.Items.Add('В этот день солнце не заходит ');
  end;
end;

procedure TSun.GetDate;
begin
  D := StrToInt(EditD.text);
  M := StrToInt(EditM.text);
```

```
  Y := StrToInt(EditY.text);
end;

procedure TSun.GetInput;
begin
  B5 := StrToInt(EditB5.Text);
  L5 := -StrToInt(EditL5.Text);
  H := StrToInt(EditH.Text);
end;

procedure TSun.ButtonCalcClick(Sender: TObject);
var
  C0: integer;
begin
  GetDate;
  GetInput;
  ListBox.Items.Add('Широта: ' + EditB5.Text + ' Долгота: '
        + EditL5.Text + ' Зона: ' + EditH.Text + ' Дата: '
        + EditD.Text + '/' + EditM.Text + '/' + EditY.Text);
  L5 := L5 / 360;
  Z0 := H / 24;
  Calendar;
  T := (J - 2451545) + F;
  TT := T / 36525 + 1;                        // TT - столетия, начиная с 1900.0
  GetTimeZone;                                // Получение часового пояса
  T := T + Z0;
  PosOfSun;                                   // Получаем положение солнца
  aA[1] := A5;
  aD[1] := D5;
  T := T + 1;
  PosOfSun;
  aA[2] := A5;
  aD[2] := D5;
  if aA[2] < aA[1] then aA[2] := aA[2] + P2;
  Z1 := DR * 90.833;                          // Вычисление зенита
  S := Sin(B5 * DR);
  C := Cos(B5 * DR);
  Z := Cos(Z1);
  M8 := 0;
  W8 := 0;
  A0 := aA[1];
  D0 := aD[1];
  DA := aA[2] - aA[1];
  DD := aD[2] - aD[1];
  for C0 := 0 to 23 do begin
    P := (C0 + 1) / 24;
    A2 := aA[1] + P * DA;
    D2 := aD[1] + P * DD;
    PossibleEvents(C0);
    if st <> '' then ListBox.Items.Add(st);
```

```
    A0 := A2;
    D0 := D2;
    V0 := V2;
  end;

  OutInform;
  ListBox.Items.Add('');                    // Разделяем данные
end;

procedure TSun.CreateForm(Sender: TObject);
begin
  EditD.Text := FormatDateTime('d', Date);
  EditM.Text := FormatDateTime('m', Date);
  EditY.Text := FormatDateTime('yyyy', Date);
end;

procedure TSun.ButtonClearClick(Sender: TObject);
begin
  ListBox.Clear;
end;

end.
```

[Ермолаев Александр]

Вычисление расстояния при известных широте и долготе

Как вычислять расстояния на географической карте?

Попробуйте использовать следующий код.

Входные данные:

- `StartLat` (начальная широта) – градусы и сотые доли
- `StartLong` (начальная долгота) – градусы и сотые доли
- `EndLat` (конечная широта) – градусы и сотые доли
- `EndLong` (конечная долгота) – градусы и сотые доли

Выходные данные:

- `Distance` (расстояние) – расстояние в метрах
- `Bearing` (смещение) – смещение в градусах

Не забудьте включить модуль `Math` в список используемых модулей.

```
const
// Константы, используемые для вычисления смещения и расстояния
  D2R: Double = 0.017453;          // Константа для преобразования градусов в радианы
  R2D: Double = 57.295781;         // Константа для преобразования радиан в градусы
  a: Double = 6378137.0;           // Основные полуоси
  b: Double = 6356752.314245;      // Не основные полуоси
```

```
  e2: Double = 0.006739496742337; // Квадрат эксцентричности эллипсоида
  f: Double = 0.003352810664747;  // Выравнивание эллипсоида
var
// Передаваемые широта/долгота в градусах и сотых долях
  StartLat: double;       // Начальная широта
  StartLong: double;      // Начальная долгота
  EndLat: double;         // Конечная широта
  EndLong: double;        // Конечная долгота
// Переменные, используемые для вычисления смещения и расстояния
  fPhimean: Double;       // Средняя широта
  fdLambda: Double;       // Разница между двумя значениями долготы
  fdPhi: Double;          // Разница между двумя значениями широты
  fAlpha: Double;         // Смещение
  fRho: Double;           // Меридианский радиус кривизны
  fNu: Double;            // Поперечный радиус кривизны
  fR: Double;             // Радиус сферы Земли
  fz: Double;             // Угловое расстояние от центра сфероида
  fTemp: Double;          // Временная переменная, используемая в вычислениях
  Distance: Double;       // Вычисленное расстояния в метрах
  Bearing: Double;        // Вычисленное от и до смещение
begin
// Вычисляем разницу между двумя долготами и широтами и получаем среднюю широту
  fdLambda := (StartLong - EndLong) * D2R;
  fdPhi := (StartLat - EndLat) * D2R;
  fPhimean := ((StartLat + EndLat) / 2.0) * D2R;
// Вычисляем меридианные и поперечные радиусы кривизны средней широты
  fTemp := 1 - e2 * (Power(Sin(fPhimean), 2));
  fRho := (a * (1 - e2)) / Power(fTemp, 1.5);
  fNu := a / (Sqrt(1 - e2 * (Sin(fPhimean) * Sin(fPhimean))));
// Вычисляем угловое расстояние
  fz := Sqrt(Power(Sin(fdPhi/2.0), 2) + Cos(EndLat * D2R) *
  Cos(StartLat * D2R) * Power(Sin(fdLambda/2.0), 2));
  fz := 2 * ArcSin(fz);
// Вычисляем смещение
  fAlpha := Cos(EndLat * D2R) * Sin(fdLambda) * 1 / Sin(fz);
  fAlpha := ArcSin(fAlpha);
// Вычисляем радиус Земли
  fR := (fRho * fNu) / ((fRho * Power(Sin(fAlpha), 2))
        + (fNu * Power(Cos(fAlpha), 2)));
// Получаем смещение и расстояние
  Distance := (fz * fR);
  if ((StartLat < EndLat) and (StartLong < EndLong)) then
    Bearing := Abs(fAlpha * R2D)
  else if ((StartLat < EndLat) and (StartLong > EndLong)) then
    Bearing := 360 - Abs(fAlpha * R2D)
  else if ((StartLat > EndLat) and (StartLong > EndLong)) then
    Bearing := 180 + Abs(fAlpha * R2D)
  else if ((StartLat > EndLat) and (StartLong < EndLong)) then
    Bearing := 180 - Abs(fAlpha * R2D);
end;
```

Рисование кривых Безье

Существует ли исходный код или какая-либо информация для рисования кривых Безье?

Решение 1

Для рисования кривой Безье разделяем интервал между P1 и P2 на несколько отрезков (их количество влияет на точность воспроизведения кривой, 3 – 4 точки вполне достаточно), затем в цикле создаем массив точек, используем описанную выше процедуру с параметром t от 0 до 1 и рисуем данный массив точек при помощи функции PolyLine().

```
type
  PBezierPoint = ^TBezierPoint;
  TBezierPoint = record
    X, Y: double;       // основной узел
    X1, Y1: double;     // левая контрольная точка
    Xr, Yr: double;     // правая контрольная точка
  end;

// P1 и P2 - две точки TBezierPoint, расположенные между 0 и 1:
// когда t = 0 X = P1.X, Y = P1.Y; когда t = 1 X = P2.X, Y = P2.Y;
procedure BezierValue(P1, P2: TBezierPoint; t: double; var X, Y: double);
var
  t_sq, t_cb, r1, r2, r3, r4: double;
begin
  t_sq := t * t;
  t_cb := t * t_sq;
  r1 := (1 - 3 * t + 3 * t_sq - t_cb) * P1.X;
  r2 := (3 * t - 6 * t_sq + 3 * t_cb) * P1.Xr;
  r3 := (3 * t_sq - 3 * t_cb) * P2.X1;
  r4 := (t_cb) * P2.X;
  X := r1 + r2 + r3 + r4;
  r1 := (1 - 3 * t + 3 * t_sq - t_cb) * P1.Y;
  r2 := (3 * t - 6 * t_sq + 3 * t_cb) * P1.Yr;
  r3 := (3 * t_sq - 3 * t_cb) * P2.Y1;
  r4 := (t_cb) * P2.Y;
  Y := r1 + r2 + r3 + r4;
end;
```

[Streblechenko Dmitry]

Решение 2

```
procedure Bezier(A: PlotArray; MaxContrPoints: integer; var B: PlotArray;
                 MaxIntPoints: integer);
const
  MaxControlPoints = 25;

type
  CombiArray = array[0..MaxControlPoints] of Float;
```

```
var
  N: integer; ContrPoint, IntPoint: integer;
  T, SumX, SumY, Prod, DeltaT, Quot: Float;
  Combi: CombiArray;
begin
  MaxContrPoints := MaxContrPoints - 1;
  DeltaT := 1.0 / (MaxIntPoints - 1);
  Combi[0] := 1; Combi[MaxContrPoints] := 1;
  for N := 0 to MaxContrPoints - 2 do
    Combi[N + 1] := Combi[N] * (MaxContrPoints - N) / (N + 1);
    for IntPoint := 1 to MaxIntPoints do begin
      T := (IntPoint - 1) * DeltaT;
      if T <= 0.5 then begin
        Prod := 1.0 - T; Quot := Prod;
        for N := 1 to MaxContrPoints - 1 do
          Prod := Prod * Quot;
        Quot := T / Quot;
        SumX := A[MaxContrPoints + 1, 1];
        SumY := A[MaxContrPoints + 1, 2];
        for N := MaxContrPoints downto 1 do begin
          SumX := Combi[N - 1] * A[N, 1] + Quot * SumX;
          SumY := Combi[N - 1] * A[N, 2] + Quot * SumY;
        end;
      end else begin
        Prod := T; Quot := Prod;
        for N := 1 to MaxContrPoints - 1 do
          Prod := Prod * Quot;
        Quot := (1 - T) / Quot;
        SumX := A[1, 1];
        SumY := A[1, 2];
        for N := 1 to MaxContrPoints do begin
          SumX := Combi[N] * A[N + 1, 1] + Quot * SumX;
          SumY := Combi[N] * A[N + 1, 2] + Quot * SumY;
         end;
      end;
      B[IntPoint, 1] := SumX * Prod;
      B[IntPoint, 2] := SumY * Prod;
    end;
end;
```

[News Group]

Управление битами

Как получить доступ к битам переменной и управлять их значением?

Решение 1

```
unit Bitwise;
interface
```

```
function IsBitSet(const val: longint; const TheBit: byte): boolean;
function BitOn(const val: longint; const TheBit: byte): LongInt;
function BitOff(const val: longint; const TheBit: byte): LongInt;
function BitToggle(const val: longint; const TheBit: byte): LongInt;

implementation

function IsBitSet(const val: longint; const TheBit: byte): boolean;
begin
  result := (val and (1 shl TheBit)) <> 0;
end;

function BitOn(const val: longint; const TheBit: byte): LongInt;
begin
  result := val or (1 shl TheBit);
end;

function BitOff(const val: longint; const TheBit: byte): LongInt;
begin
  result := val and ((1 shl TheBit) xor $FFFFFFFF);
end;

function BitToggle(const val: longint; const TheBit: byte): LongInt;
begin
  result := val xor (1 shl TheBit);
end;

end.
```

Решение 2

SetWord – слово, которое необходимо установить. BitNum – номер бита, который необходимо выставить согласно определениям в секции const (Bit0, Bit1 и т. д.). GetBitStat возвращает значение True, если бит установлен и, False – в противном случае.

```
const
  Bit0 = 1;
  Bit1 = 2;
  Bit2 = 4;
  Bit3 = 8;
  Bit4 = 16;
  Bit5 = 32;
  Bit6 = 64;
  Bit7 = 128;

  Bit8 = 256;
  Bit9 = 512;
  Bit10 = 1024;
  Bit11 = 2048;
  Bit12 = 4096;
  Bit13 = 8192;
```

```
  Bit14 = 16384;
  Bit15 = 32768;

procedure SetBit(SetWord, BitNum: Word);
begin
  SetWord := SetWord Or BitNum;       { Устанавливаем бит }
end;

procedure ClearBit(SetWord, BitNum: Word);
begin
  SetWord := SetWord Or BitNum;       { Устанавливаем бит }
  SetWord := SetWord Xor BitNum;      { Переключаем бит }
end;

procedure ToggleBit(SetWord, BitNum: Word);
begin
  SetWord := SetWord Xor BitNum;      { Переключаем бит }
end;

function GetBitStat(SetWord, BitNum: Word): Boolean;
begin
  GetBitStat := SetWord and BitNum = BitNum; { Если бит установлен }
end;
```

Гауссово размывание

Ядро Гауссовой функции `exp(-(x^2 + y^2))` есть разновидность формулы `f(x)*g(y)`, которая означает, что мы можем выполнить двумерную свертку, создавая последовательность одномерных сверток, – сначала мы свертываем каждую строчку изображения, затем – каждую колонку. Хороший повод для ускорения (`N^2` становится `N*2`). Любая свертка требует некоторого места для временного хранения результатов. Ниже в коде программа `BlurRow` как раз распределяет и освобождает память для каждой колонки. Вероятно, это должно ускорить обработку изображения, правда, не ясно насколько.

Поле `size` в записи `TKernel` ограничено значением 200. Если вам нужен еще больший радиус, воспользуйтесь значениями `radius = 3, 5` или другими. Для большого количества данных методы свертки на практике оказываются эффективнее преобразований Фурье (об этом свидетельствуют многочисленные опыты).

Еще один комментарий все же необходим: гауссово размывание имеет одно магическое свойство, а именно – вы можете сначала размыть каждую строчку (применить фильтр), затем каждую колонку. Такая процедура занимает значительно меньше времени, чем двумерная свертка.

Во всяком случае, можете сделать так:

```
unit GBlur2;

interface
```

```
uses
  Windows, Graphics;

type
  PRGBTriple = ^TRGBTriple;
  TRGBTriple = packed record
               b: byte;         // легче для использования, чем тип RGBTBlue...
               g: byte;
               r: byte;
  end;

PRow = ^TRow;
TRow = array[0..1000000] of TRGBTriple;

PPRows = ^TPRows;
TPRows = array[0..1000000] of PRow;

const MaxKernelSize = 100;

type
  TKernelSize = 1..MaxKernelSize;

TKernel = record
  Size: TKernelSize;
  Weights: array[-MaxKernelSize..MaxKernelSize] of single;
end;

// идея заключается в том, что при использовании TKernel мы игнорируем
// Weights (вес), за исключением Weights в диапазоне -Size..Size.
procedure GBlur(theBitmap: TBitmap; radius: double);

implementation

uses
  SysUtils;

// Создаем K (Гауссово зерно) со среднеквадратичным отклонением = radius.
// Для текущего приложения устанавливаем переменные MaxData = 255,
// DataGranularity = 1. Теперь в процедуре установим значение K.Size так,
// что при использовании K мы будем игнорировать Weights (вес)
// с наименее возможными значениями. (Малый размер нам на пользу,
// поскольку время выполнения напрямую зависит от значения K.Size.)
procedure MakeGaussianKernel(var K: TKernel; radius: double;
                             MaxData, DataGranularity: double);
var
  j: integer;
  temp, delta: double;
  KernelSize: TKernelSize;
begin
  for j := Low(K.Weights) to High(K.Weights) do begin
    temp := j/radius;
```

```
    K.Weights[j] := exp(-temp * temp / 2);
  end;

// делаем так, чтобы sum(Weights) = 1:
  temp := 0;
  for j := Low(K.Weights) to High(K.Weights) do
    temp:= temp + K.Weights[j];
  for j := Low(K.Weights) to High(K.Weights) do
    K.Weights[j] := K.Weights[j] / temp;

// теперь отбрасываем (или делаем отметку "игнорировать" для переменной Size)
// данные, имеющие относительно небольшое значение – это важно, в противном случае
// смазывание происходит с малым радиусом и в той области, которая "захватывается"
// большим радиусом...
  KernelSize := MaxKernelSize;
  delta := DataGranularity/(2 * MaxData);
  temp := 0;
  while (temp < delta) and (KernelSize > 1) do begin
    temp := temp + 2 * K.Weights[KernelSize];
    dec(KernelSize);
  end;
  K.Size := KernelSize;

// теперь для обеспечения корректности возвращаемого результата проводим ту же
// операцию с K.Size, так чтобы сумма всех данных была равна единице:
  temp := 0;
  for j := -K.Size to K.Size do
    temp := temp + K.Weights[j];
  for j := -K.Size to K.Size do
    K.Weights[j] := K.Weights[j] / temp;
end;

function TrimInt(Lower, Upper, theInteger: integer): integer;
begin
  if (theInteger <= Upper) and (theInteger >= Lower) then result := theInteger
  else if theInteger > Upper then result := Upper
  else result := Lower;
end;

function TrimReal(Lower, Upper: integer; x: double): integer;
begin
  if (x < upper) and (x >= lower) then result := trunc(x)
  else if x > Upper then result := Upper
  else result := Lower;
end;

procedure BlurRow(var theRow: array of TRGBTriple; K: TKernel; P: PRow);
var
  j, n, LocalRow: integer;
  tr, tg, tb: double;                // tempRed и др.
  w: double;
```

```
begin
  for j := 0 to High(theRow) do begin
    tb := 0;
    tg := 0;
    tr := 0;

    for n  := -K.Size to K.Size do begin
      w := K.Weights[n];
// TrimInt задает отступ от края строки...
      with theRow[TrimInt(0, High(theRow), j - n)] do begin
        tb := tb + w * b;
        tg := tg + w * g;
        tr := tr + w * r;
      end;
    end;
    with P[j] do begin
      b := TrimReal(0, 255, tb);
      g := TrimReal(0, 255, tg);
      r := TrimReal(0, 255, tr);
    end;
  end;
  Move(P[0], theRow[0], (High(theRow) + 1) * Sizeof(TRGBTriple));
end;

procedure GBlur(theBitmap: TBitmap; radius: double);
var
  Row, Col: integer;
  theRows: PPRows;
  K: TKernel;
  ACol: PRow;
  P: PRow;
begin
  if (theBitmap.HandleType <> bmDIB) or (theBitmap.PixelFormat <> pf24Bit)
  then raise exception.Create('GBlur может работать только
                               с 24-битными изображениями');
  MakeGaussianKernel(K, radius, 255, 1);
  GetMem(theRows, theBitmap.Height * SizeOf(PRow));
  GetMem(ACol, theBitmap.Height * SizeOf(TRGBTriple));

// запись позиции данных изображения:
  for Row := 0 to theBitmap.Height - 1 do
    theRows[Row] := theBitmap.Scanline[Row];

// размываем каждую строчку:
  P := AllocMem(theBitmap.Width * SizeOf(TRGBTriple));
  for Row := 0 to theBitmap.Height - 1 do
    BlurRow(Slice(theRows[Row]^, theBitmap.Width), K, P);

// теперь размываем каждую колонку
  ReAllocMem(P, theBitmap.Height * SizeOf(TRGBTriple));
  for Col := 0 to theBitmap.Width - 1 do begin
```

```
// считываем первую колонку в TRow:
    for Row := 0 to theBitmap.Height - 1 do ACol[Row] := theRows[Row][Col];
    BlurRow(Slice(ACol^, theBitmap.Height), K, P);

// теперь помещаем обработанный столбец на свое место в данные изображения:
    for Row := 0 to theBitmap.Height - 1 do theRows[Row][Col] := ACol[Row];
  end;
  FreeMem(theRows);
  FreeMem(ACol);
  ReAllocMem(P, 0);
end;

end.
```

Пример использования

```
procedure TForm1.Button1Click(Sender: TObject);
var
  b: TBitmap;
begin
  if not openDialog1.Execute then exit;
  b := TBitmap.Create;
  b.LoadFromFile(OpenDialog1.Filename);
  b.PixelFormat:= pf24Bit;
  Canvas.Draw(0, 0, b);
  GBlur(b, StrToFloat(Edit1.text));
  Canvas.Draw(b.Width, 0, b);
  b.Free;
end;
```

Имейте в виду, что 24-битные изображения при системной палитре из 256 цветов требуют некоторых дополнительных хитростей, т. к. иначе эти изображения не только теряют некоторый объем информации, но и серьезно нарушают работу фильтра.

Рисование фрактальных графов

Предлагаемое решение (исходный код для Turbo Pascal 7).

```
program Fractal;

uses
  graph, crt;

const
  GrafType = 1; {1..3}

type
  PointPtr = ^Point;
  Point = Record
```

```
    X, Y: Word;
    Angle: Real;
    Next: PointPtr
  end;

  GrfLine = array [0..5000] of Byte;
  ChangeType = array [1..30] of Record
    Mean: Char;
    NewString: String;
  end;

var
  K, T, Dx, Dy, StepLength, GrafLength: Word;
  grDriver, Xt: Integer;
  grMode, ErrCode: Integer;
  CurPosition: Point;
  Descript: GrfLine;
  StartLine: String Absolute Descript;
  ChangeNumber, Generation: Byte;
  Changes: ChangeType;
  AngleStep: Real;
  Mem: Pointer;

procedure Replace(var Stroka: GrfLine; OldChar: Char; Repl: String);
var
  I, J: Word;
begin
  if (GrafLength = 0) Or (Length(Repl) = 0) then Exit;
  I := 1;
  while I <= GrafLength do begin
    if Chr (Stroka[I]) = OldChar then begin
      for J := GrafLength downto I + 1 do
        Stroka[J + Length(Repl) - 1] := Stroka[J];
      for J := 1 to Length(Repl) do
        Stroka[I + J - 1] := Ord(Repl[J]);
      I := I + J;
      GrafLength := GrafLength + Length(Repl) - 1;
      continue;
    end;
    I := I + 1;
  end;
end;

procedure PushCoord(var Ptr: PointPtr; C: Point);
var
  P: PointPtr;
begin
  New(P);
  P^.X := C.X;
  P^.Y := C.Y;
```

```
  P^.Angle := C.Angle;
  P^.Next := Ptr;
  Ptr := P;
end;

procedure PopCoord(var Ptr: PointPtr; var Res: Point);
begin
  if Ptr <> Nil then begin
    Res.X := Ptr^.X;
    Res.Y := Ptr^.Y;
    Res.Angle := Ptr^.Angle;
    Ptr := Ptr^.Next;
  end;
end;

procedure FindGrafCoord(var Dx, Dy: Word; Angle: Real; StepLength: Word);
begin
  Dx := Round(Sin (Angle) * StepLength * GetMaxX / GetMaxY);
  Dy := Round( - Cos (Angle) * StepLength);
end;

procedure NewAngle(Way: ShortInt; var Angle: Real; AngleStep: Real);
begin
  if Way >= 0 then Angle := Angle + AngleStep
  else Angle := Angle - AngleStep;
  if Angle >= 4 * Pi then Angle := Angle - 4 * Pi;
  if Angle < 0 then Angle := 4 * Pi + Angle;
end;

procedure Rost(var Descr: GrfLine; Cn: Byte; Ch: ChangeType);
var
  I: Byte;
begin
  for I := 1 to Cn do Replace(Descr, Ch[I].Mean, Ch[I].NewString);
end;

procedure Init1;
begin
  AngleStep := Pi / 8;
  StepLength := 7;
  Generation := 4;
  ChangeNumber := 1;
  CurPosition.Next := Nil;
  StartLine := 'F';
  GrafLength := Length(StartLine);
  with Changes [1] do begin
    Mean := 'F';
    NewString := 'FF+[+F-F-F]-[-F+F+F]';
  end;
end;
```

```
procedure Init2;
begin
  AngleStep := Pi / 4;
  StepLength := 3;
  Generation := 5;
  ChangeNumber := 2;
  CurPosition.Next := Nil;
  StartLine := 'G';
  GrafLength := Length (StartLine);
  with Changes [1] do begin
    Mean := 'G';
    NewString := 'GFX[+G][-G]';
  end;
  with Changes [2] do begin
    Mean := 'X';
    NewString := 'X[-FFF][+FFF]FX';
  end;
end;

procedure Init3;
begin
  AngleStep := Pi / 10;
  StepLength := 9;
  Generation := 5;
  ChangeNumber := 5;
  CurPosition.Next := Nil;
  StartLine := 'SLFF';
  GrafLength := Length (StartLine);
  with Changes [1] do begin
    Mean := 'S';
    NewString := '[+++G][---G]TS';
  end;
  with Changes [2] do begin
    Mean := 'G';
    NewString := '+H[-G]L';
  end;
  with Changes [3] do begin
    Mean := 'H';
    NewString := '-G[+H]L';
  end;
  with Changes [4] do begin
    Mean := 'T';
    NewString := 'TL';
  end;
  with Changes [5] do begin
    Mean := 'L';
    NewString := '[-FFF][+FFF]F';
  end;
end;
begin
  case GrafType of
```

```
    1:  Init1;
    2:  Init2;
    3:  Init3;
  end;
  grDriver := detect;
  InitGraph(grDriver, grMode, '');
  ErrCode := GraphResult;
  if ErrCode <> grOk then begin
    WriteLn('Graphics error:', GraphErrorMsg(ErrCode));
    Halt(1)
  end;
  with CurPosition do begin
    X := GetMaxX Div 2;
    Y := GetMaxY;
    Angle := 0;
    MoveTo(X, Y)
  end;
  SetColor(white);
  for K := 1 To Generation do begin
    Rost(Descript, ChangeNumber, Changes);
    Mark(Mem);
    for T := 1 To GrafLength do begin
      case Chr(Descript[T]) of
          'F':  begin
                  FindGrafCoord (Dx, Dy, CurPosition.Angle, StepLength);
                  with CurPosition do begin
                    Xt := X + Dx;
                    if Xt < 0 then X := 0
                    else X := Xt;
                    if X > GetMaxX then X := GetMaxX;
                    Xt := Y + Dy;
                    if Xt < 0 then Y := 0
                    else Y := Xt;
                    if Y > GetMaxY then Y := GetMaxY;
                    LineTo(X, Y)
                  end;
                end;
          'f':  begin
                  FindGrafCoord (Dx, Dy, CurPosition.Angle, StepLength);
                  with CurPosition do begin
                    Xt := X + Dx;
                    if Xt < 0 then X := 0 else X := Xt;
                    if X > GetMaxX then X := GetMaxX;
                    Xt := Y + Dy;
                    if Xt < 0 then Y := 0 else Y := Xt;
                    if Y > GetMaxY then Y := GetMaxY;
                    MoveTo(X, Y)
                  end;
                end;
          '+':  NewAngle(1, CurPosition.Angle, AngleStep);
          '-':  NewAngle( - 1, CurPosition.Angle, AngleStep);
```

```
          'I':  NewAngle(1, CurPosition.Angle, 2 * Pi);
          '[':  PushCoord (CurPosition.Next, CurPosition);
          ']':  begin
                  PopCoord(CurPosition.Next, CurPosition);
                  with CurPosition do MoveTo(X, Y);
                end;
      end;
    end;
    Dispose(Mem);
    Delay(1000);
  end;
  Repeat
  Until KeyPressed;
  CloseGraph;
end.
```

[Марковский Михаил]

Вращение изображения

С помощью предлагаемого программного кода реализуется быстрый и примитивный способ вращения изображения. По крайней мере, это тоже выход из положения, поскольку Windows этого делать не умеет.

```
procedure RotateRight(BitMap: TImage);
var
  FirstC, LastC, c, r: integer;

  procedure FixPixels(c, r: integer);
  var
    SavePix, SavePix2: tColor;
    i, NewC, NewR: integer;
  begin
    SavePix := Bitmap.Canvas.Pixels[c, r];
    for i := 1 to 4 do begin
      Newc := BitMap.Height - r + 1;
      Newr := c;
      SavePix2 := BitMap.Canvas.Pixels[Newc, Newr];
      Bitmap.Canvas.Pixels[Newc, Newr] := SavePix;
      SavePix := SavePix2;
      c := NewC;
      r := NewR;
    end;
  end;

begin
  if BitMap.Width <> BitMap.Height then exit;
  BitMap.Visible := False;
  with Bitmap.Canvas do begin
    FirstC := 0;
    LastC := BitMap.Width;
```

```
    for r := 0 to BitMap.Height div 2 do begin
      for c := FirstC to LastC do FixPixels(c, r);
      Inc(FirstC);
      Dec(LastC);
    end;
  end;
  BitMap.Visible := True;
end;
```

[News Group]

Примечание

Вращение происходит на 90 градусов вправо за одно выполнение процедуры. Не забудьте добавить компонент `TImage` *на форму, загрузить изображение и передать* `TImage` *в качестве параметра в процедуру вращения.*

64-битное кодирование/декодирование

Реализация алгоритма декодирования base64.

```
const
  Base64Table =
    'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/';

function Base64Decode(cStr: string): string;
var
  ResStr: string;
  DecStr: string;
  RecodeLine: array [1..76] of byte;
  f1, f2: word;
  l: integer;
begin
  l := length(cStr);
  ResStr := '';
  for f1 := 1 to l do
    if cStr[f1]='=' then RecodeLine[f1] := 0
    else RecodeLine[f1] := pos(cStr[f1], Base64Table) - 1;
  f1 := 1;
  while f1 < length(cStr) do begin
    DecStr := chr(byte(RecodeLine[f1] shl 2) + RecodeLine[f1 + 1] shr 4)
        + chr(byte(RecodeLine[f1 + 1] shl 4) + RecodeLine[f1 + 2] shr 2)
        + chr(byte(RecodeLine[f1 + 2] shl 6) + RecodeLine[f1 + 3]);
    ResStr := ResStr + DecStr;
    inc(f1, 4);
  end;
  Base64Decode := ResStr;
end;
```

[Варавва Алексей]

Защита программ перекрытием кода

Не секрет, что совершенной защиты не существует. Тем не менее, хорошая защита должна обеспечить такой уровень, чтобы на ее вскрытие нужно было затратить усилия, сравнимые с самостоятельным написанием программы. Разумеется, она должна быть многоуровневой и перекрывающейся (уровни должны работать независимо). Не забывайте, что хорошие взломщики неплохо знают Ассемблер, и высокоуровневые ухищрения от них не спасают. Следовательно, для построения высококлассной защиты с применением Ассемблера необходимо владеть последним в совершенстве. Не думайте, что вам это не подходит, т. к. слишком сложно или уже не модно. Хороший программист не пренебрегает Ассемблером и высшей математикой.

Один из методов – это перекрывающийся код. Он может показаться немного сложным для большинства из нас, но, зная несколько HEX-кодов инструкций процессора, вы тоже сможете создать небольшой по размеру перекрывающийся код. Перекрывающийся код можно сделать сколь угодно многоуровневым, а здесь я покажу лишь, в каком направлении надо «копать».

```
    temp_string := 'Den is Com';
    asm
        mov ax, $05EB
@as:  jmp @as-2
    end;
    ShowMessage('Сообщение');
```

На первый взгляд, это может озадачить, но на самом деле все очень просто. Первая инструкция заносит значение в AX. Вторая выполняет переход на значение операнда команды MOV. Код '05EB' переводится как 'JMP $+5' (помните, что слова хранятся в обратном порядке). Этот переход минует JMP и передает выполнение дальше. Вероятно, этого не будет достаточно для защиты, но технику ее создания демонстрирует.

Присваивание temp_string := 'Den is Com' существенной роли не играет, но может применяться при отладке программы, т. к. хорошо просматривается при использовании дизассемблера и отладчика. Возможно, ваши первые попытки будут приводить к частому зависанию компьютера, но не отчаивайтесь – защита того стоит. Попробуйте разработать свой способ сравнения строк (чаще всего ловятся именно эти инструкции), замаскируйте инструкции зависания компьютера и т. д.

[Den is Com]

Генерация случайного пароля

Необходимо, чтобы приложение само создавало пароли.

Возможно, данный способ пригодится. Совместимость: Delphi 5 и выше.

Внимание

Длина пароля должна быть меньше длины таблицы!

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Randomize;  // запускаем генератор случайных чисел
end;

function RandomPwd(PWLen: integer): string;
// таблица символов, используемых в пароле
const
  StrTable: string = '!#$%&/()=?@<>|{[]}\*~+#;:.-' +
                     + '_ABCDEFGHIJKLMabcdefghijklm'+
                     + '0123456789ДЦЬдцьЯNOPQRSTUVWXYZnopqrstuvwxyz';
var
  N, K, X, Y: integer;
begin
// проверяем максимальную длину пароля
  if (PWlen > Length(StrTable)) then K := Length(StrTable)- 1
  else K := PWLen;
  SetLength(result, K);              // устанавливаем длину конечной строки
  Y := Length(StrTable);             // Длина Таблицы для внутреннего цикла
  N := 0;                            // начальное значение цикла
  while N < K do begin               // цикл для создания K символов
    X := Random(Y) + 1;              // берем следующий случайный символ
// проверяем присутствие этого символа в конечной строке
    if Pos(StrTable[X], result) = 0 then begin
      inc(N);                        // символ не найден
      Result[N] := StrTable[X];      // теперь его сохраняем
    end;
  end;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  cPwd: string;
begin
  // вызываем функцию генерации пароля из 30 символов
  cPwd := RandomPwd(30);
  // ...
end;
```

[Nikolaev Igor]

Как закодировать строку

Мне надо закодировать информацию. Как это можно сделать?

Приведенная программа демонстрирует методы кодирования и раскодирования строк:

Примечание

Мы не отвечаем за уникальность и секретность алгоритма данной функции.

```
program Crypt;

{$APPTYPE CONSOLE}

const
  C1 = 52845;
  C2 = 22719;

function Encrypt(const S: String; Key: Word): String;
var
  I: byte;
begin
  SetLength(Result, Length(S));
  for I := 1 to Length(S) do begin
    Result[I] := char(byte(S[I]) xor (Key shr 8));
    Key := (byte(Result[I]) + Key) * C1 + C2;
  end;
end;

function Decrypt(const S: String; Key: Word): String;
var
  I: byte;
begin
  SetLength(Result, Length(S));
  for I := 1 to Length(S) do begin
    Result[I] := char(byte(S[I]) xor (Key shr 8));
    Key := (byte(S[I]) + Key) * C1 + C2;
  end;
end;

var
  S: string;
begin
  Write('>');
  ReadLn(S);
  S := Encrypt(S, 12345);
  WriteLn(S);
  S := Decrypt(S, 12345);
  WriteLn(S);
  Readln;
end.
```

Как стереть самого себя

При работе происходит блокировка исполняемого файла программы на диске до момента завершения работы программы. Предлагаемый код позволяет программе стереть саму себя с диска. При этом, если программа уже загрузилась в память, то ее работа может продолжаться. Метод действия: создается временный BAT-файл во временной директории на диске, который удаляет и программу, и себя. Применяя этот код для защиты, желательно использовать хотя бы простейшую шифровку текстовых строк XOR, т. к. они

хорошо просматриваются дизассемблером, и, разумеется, хакеру не составит труда обнаружить защиту.

Все файлы проекта:

```
program Project1;

uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

Модуль Unit1.dfm:

```
object Form1: TForm1
  Left = 192
  Top = 107
  Width = 435
  Height = 300
  Caption = 'Form1'
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  OldCreateOrder = False
  PixelsPerInch = 96
  TextHeight = 13
  object Button1: TButton
    Left = 48
    Top = 200
    Width = 313
    Height = 49
    Caption = 'Del me !'
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clWindowText
    Font.Height = -32
    Font.Name = 'MS Sans Serif'
    Font.Style = []
    ParentFont = False
    TabOrder = 0
    OnClick = Button1Click
  end
  object Memo1: TMemo
```

```
    Left = 48
    Top = 16
    Width = 305
    Height = 169
    TabOrder = 1
  end
end
```

Модуль Unit1.pas:

```
unit Unit1;

interface

uses
  Windows, Messages, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls,
  SysUtils;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Memo1: TMemo;
    procedure Button1Click(Sender: TObject);
end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
var
  Filename: String;
  aTempBuf: array[0..MAX_PATH] of char;
  bat_file: String;
  f: TextFile;
  si: TStartUpInfo;
  pi: TProcessInformation;
begin
  FileName := Application.ExeName;
  if GetShortPathName(PChar(FileName), aTempBuf, SizeOf(aTempBuf) - 1) > 0
  then FileName := StrPas(aTempBuf);
  GetEnvironmentVariable('TEMP', aTempBuf, MAX_PATH);
  bat_file := StrPas(aTempBuf) + '\' + 'delself.bat';
  if GetShortPathName(PChar(bat_file), aTempBuf, SizeOf(aTempBuf) - 1) > 0
  then bat_file := StrPas(aTempBuf);
  AssignFile(f, bat_file);
  rewrite(f);
  writeln(f, '@echo off');
```

```
    writeln(f, ':try');
    writeln(f, 'del ' + FileName);
    write(f,'if exist ' + FileName);
    writeln(f, ' goto try');
    write(f, 'del ' + bat_file);
    CloseFile(f);
    Memo1.Lines.LoadFromFile(bat_file);
    ZeroMemory(@si, SizeOf(si));
    si.cb := SizeOf(si);
    si.wShowWindow := SW_HIDE;
    si.dwFlags := STARTF_USESHOWWINDOW;
    CreateProcess(nil, PChar(bat_file), nil, nil, False, IDLE_PRIORITY_CLASS
                  or DETACHED_PROCESS, nil, nil, si, pi);
end;

end.
```

[Denis Com]

Пример защиты типа SHAREWARE

В качестве примера приведен небольшой участок программного кода, позволяющий быстро создать защиту для программ SHAREWARE, которая не влияет на функциональность самой программы, но настоятельно «просит» ее зарегистрировать и закрывает при каждом повторном запуске.

Технология данного метода заключается в том, что пользователь может запустить программу только один раз за текущий сеанс Windows.

Используйте обработчик события FormShow:

```
procedure TForm1.FormShow(Sender : TObject);
var
  atom: integer;
  CRLF: string;
begin
  if GlobalFindAtom('THIS_IS_SOME_OBSCUREE_TEXT') = 0 then
    atom := GlobalAddAtom('THIS_IS_SOME_OBSCUREE_TEXT')
  else begin
    CRLF := #10 + #13;
    ShowMessage('Данная версия предусматривает только один запуск'
              + 'в текущем сеансе Windows.' + CRLF
              + 'Для повторного запуска необходимо перезапустить Windows, или,'
              + CRLF + 'что лучше, - ' + CRLF + 'ЗАРЕГИСТРИРУЙТЕСЬ !');
    Close;
  end;
end;
```

Преимущество данного метода в том, что пользователю доступны все возможности программы, но только до момента ее закрытия или перезапуска системы. Вся хитрость заключается в сохранении некоторой строки в сис-

темных глобальных переменных («атомах») и последующей проверке ее в таблице «атомов» системы.

Перекодировка текста из DOS в Windows и наоборот

Как с помощью Delphi перекодировать текстовый файл из DOS в Windows и наоборот?

Решение 1

Str – текст для перекодировки.

```
procedure TForm1.WinToDos;
var
  Str: PChar;
begin
  Str := Memo1.Lines.GetText;            // Берем текст из TMemo как PChar
  CharToOem(Str, Str);                   // Перекодировка текста
  Memo2.Lines.SetText(Str);              // Передаем перекодированный текст
end;

procedure TForm1.DosToWin;               // С точностью до вызова функции API
var                                      // повторяем код
  Str: PChar;
begin
  Str := Memo1.Lines.GetText;
  OemToChar(Str, Str);
  Memo2.Lines.SetText(Str);
end;
```

Рекомендация: применяя функции CharToOem и OemToChar в качестве источника текста и приемника перекодированного текста, следует использовать одну и ту же переменную, иначе при перекодировке больших текстов может возникнуть ошибка типа "Access violation".

[Виталий Еремеев]

Решение 2

Используйте CharToOEM, OEMToChar, CharToOEMBuff, OEMToCharBuff.

[Nomadic]

Чтение и запись файлов UNIX

Мне нужно обрабатывать информацию, созданную в UNIX. Как эту информацию перекодировать для работы в Windows?

Данный модуль позволяет читать и записывать файлы формата UNIX.

```
unit StreamFile;

interface
```

```
uses
  SysUtils;

procedure AssignStreamFile(var F: Text; Filename: String);

implementation

const
  BufferSize = 128;

type
  TStreamBuffer = Array [1..High (Integer)] of Char;

  TStreamBufferPointer = ^TStreamBuffer;

  TStreamFileRecord = record
    Case Integer of
      1: (Filehandle: Integer;
         Buffer: TStreamBufferPointer;
         BufferOffset: Integer;
         ReadCount: Integer;);
      2: (Dummy: Array [1..32] of Char)
  end;

function StreamFileOpen(var F: TTextRec): Integer;
var
  Status: Integer;
begin
  with TStreamFileRecord(F.UserData) do begin
    GetMem (Buffer, BufferSize);
    case F.Mode of
      fmInput:
          FileHandle := FileOpen(StrPas(F.Name), fmShareDenyNone);
      fmOutput:
          FileHandle := FileCreate(StrPas(F.Name));
      fmInOut:  begin
          FileHandle := FileOpen(StrPas(F.Name),
                  fmShareDenyNone or fmOpenWrite or fmOpenRead);
          if FileHandle <> -1 then
          { Перемещаемся в конец файла. }
            status := FileSeek (FileHandle, 0, 2);
          F.Mode := fmOutput;
        end;
    end;
    BufferOffset := 0;
    ReadCount := 0;
    F.BufEnd := 0;
    if FileHandle = -1 then Result := -1
    else Result := 0;
  end;
end;
```

```
function StreamFileInOut(var F: TTextRec): Integer;
  procedure Read(var Data: TStreamFileRecord);

    procedure CopyData;
    begin
      while (F.BufEnd < Sizeof(F.Buffer) - 2)
        and (Data.BufferOffset <= Data.ReadCount)
        and (Data.Buffer[Data.BufferOffset] <> #10) do begin
          F.Buffer[F.BufEnd] := Data.Buffer^[Data.BufferOffset];
          Inc(Data.BufferOffset);
          Inc(F.BufEnd);
      end;
      if Data.Buffer[Data.BufferOffset] = #10 then begin
        F.Buffer[F.BufEnd] := #13;
        Inc(F.BufEnd);
        F.Buffer[F.BufEnd] := #10;
        Inc(F.BufEnd);
        Inc(Data.BufferOffset);
      end;
    end;

  begin
    F.BufEnd := 0;
    F.BufPos := 0;
    F.Buffer := '';
    repeat
      if (Data.ReadCount = 0) or (Data.BufferOffset > Data.ReadCount)
      then begin
          Data.BufferOffset := 1;
          Data.ReadCount := FileRead(Data.FileHandle, Data.Buffer^, BufferSize);
      end;
      CopyData;
    until (Data.ReadCount = 0) or (F.BufEnd >= Sizeof(F.Buffer) - 2);
    Result := 0;
  end;
  procedure Write(var Data: TStreamFileRecord);
  var
    Status: Integer;
    Destination: Integer;
    II: Integer;
  begin
    with TStreamFileRecord(F.UserData) do begin
      Destination := 0;
      for II := 0 to F.BufPos - 1 do begin
        if F.Buffer[II] <> #13 then begin
          Inc(Destination);
          Buffer^[Destination] := F.Buffer[II];
        end;
      end;
      Status := FileWrite(FileHandle, Buffer^, Destination);
```

```
      F.BufPos := 0;
      Result := 0;
    end;
  end;

begin
  case F.Mode of
    fmInput:   read(TStreamFileRecord(F.UserData));
    fmOutput:  write(TStreamFileRecord(F.UserData));
  end;
end;

function StreamFileFlush(var F: TTextRec): Integer;
begin
  Result := 0;
end;

function StreamFileClose(var F: TTextRec): Integer;
begin
  with TStreamFileRecord (F.UserData) do begin
    FreeMem(Buffer);
    FileClose(FileHandle);
  end;
  Result := 0;
end;

procedure AssignStreamFile(var F: Text; Filename: String);
begin
  with TTextRec(F) do begin
    Mode := fmClosed;
    BufPtr := @Buffer;
    BufSize := SizeOf(Buffer);
    OpenFunc := @StreamFileOpen;
    InOutFunc := @StreamFileInOut;
    FlushFunc := @StreamFileFlush;
    CloseFunc := @StreamFileClose;
    StrPLCopy(Name, FileName, Sizeof(Name) - 1);
  end;
end;

end.
```

Перенос русского текста по слогам

Как выполнить перенос текста по слогам?

Решение

```
unit Hyper;

interface
```

```
uses
  Windows, Classes, SysUtils;

function SetHyph(pc: PChar; MaxSize: Integer): PChar;
function SetHyphString(s: String): String;
function MayBeHyph(p: PChar; pos: Integer): Boolean;

implementation

type
  TSymbol = (st_Empty, st_NoDefined, st_Glas, st_Sogl, st_Spec);
  TSymbAr = array [0..1000] of TSymbol;
  PSymbAr = ^TSymbAr;

const
  HypSymb = #$1F;
  Spaces = [' ', ',', ';', ':', '.', '?', '!', '/', #10, #13];
  GlasChar = ['Й', 'й', 'У', 'у', 'Е', 'е','Ю', 'ю',
          'А', 'а', 'О', 'о', 'Э', 'э', 'Я', 'я', 'И', 'и',
    { english }
          'e', 'E', 'u', 'U', 'i', 'I', 'o', 'O', 'a', 'A', 'j', 'J'];
  SoglChar = ['Г', 'г', 'Ц', 'ц', 'К', 'к', 'Н', 'н',
          'Ш', 'ш', 'Щ', 'Щ', 'З', 'з', 'Х', 'х',
          'Ф', 'ф', 'В', 'в', 'П', 'п', 'Р', 'р', 'Л', 'л', 'Д', 'д',
          'Ж', 'ж', 'Ч', 'ч', 'С', 'с', 'М', 'м', 'т', 'Т', 'б', 'Б'
    { english }
          'q', 'Q', 'w', 'W', 'r', 'R', 't', 'T', 'y', 'Y',
          'p', 'P', 's', 'S', 'd', 'D', 'f', 'F', 'g', 'G',
          'h', 'H', 'k', 'K', 'l', 'L', 'z', 'Z', 'x', 'X',
          'c', 'C', 'v', 'V', 'b', 'B', 'n', 'N', 'm', 'M'];
  SpecSign = ['Ы', 'ы', 'Ь', 'ь', 'Ъ', 'ъ'];

function isSogl(c: Char): Boolean;
begin
  Result := c in SoglChar;
end;

function isGlas(c: Char): Boolean;
begin
  Result := c in GlasChar;
end;

function isSpecSign(c: Char): Boolean;
begin
  Result := c in SpecSign;
end;

function GetSymbType(c: Char): TSymbol;
begin
  if isSogl(c) then begin
```

```
    Result := st_Sogl;
    exit;
  end;
  if isGlas(c) then begin
    Result := st_Glas;
    exit;
  end;
  if isSpecSign(c) then begin
    Result := st_Spec;
    exit;
  end;
  Result := st_NoDefined;
end;

function isSlogMore(c: pSymbAr; start, len: Integer): Boolean;
var
  i: Integer;
begin
  for I := Start to Len-1 do begin
    if c^[i] = st_NoDefined then begin
      Result := false;
      exit;
    end;
    if (c^[i] = st_Glas)
      and ((c^[i+1] <> st_Nodefined) or (I <> Start)) then begin
        Result := True;
        exit;
    end;
  end;
  Result := false;
end;

{ расставляем переносы }
function SetHyph(pc: PChar; MaxSize: Integer): PChar;
var
  HypBuff: Pointer;
  h: PSymbAr;
  i: Integer;
  len: Integer;
  Cur: Integer;                    { текущая позиция в результирующем массиве}
  cw: Integer;                     { Номер буквы в слове }
  Lock: Integer;                   { счетчик блокировок }
begin
  Cur := 0;
  len := StrLen(pc);
  if (MaxSize=0) or (Len=0) then begin
    Result := nil;
    Exit;
  end;
  GetMem(HypBuff, MaxSize);
  GetMem(h, len + 1);
```

```
{ заполнение массива типов символов }
  for I := 0 to len - 1 do h^[i] := GetSymbType(pc[i]);
{ собственно расстановка переносов }
  cw := 0;
  Lock := 0;
  for I := 0 to Len-1 do begin
    PChar(HypBuff)[cur] := PChar(pc)[i]; Inc(Cur);
    if I >= Len - 2 then Continue;
    if h^[i] = st_NoDefined then begin
      cw := 0;
      Continue;
    end else
      Inc(cw);
    if Lock <> 0 then begin
      Dec(Lock);
      Continue;
    end;
    if cw <= 1 then Continue;
    if not (isSlogMore(h, i + 1, len)) then Continue;
    if (h^[i] = st_Sogl) and (h^[i - 1] = st_Glas)
      and (h^[i + 1] = st_Sogl)
      and (h^[i + 2] <> st_Spec) then begin
        PChar(HypBuff)[cur] := HypSymb;
        Inc(Cur);
        Lock := 1;
    end;
    if (h^[i] = st_Glas) and (h^[i - 1] = st_Sogl)
      and (h^[i + 1] = st_Sogl)
      and (h^[i + 2] = st_Glas) then begin
        PChar(HypBuff)[cur] := HypSymb;
        Inc(Cur);
        Lock := 1;
    end;
    if (h^[i] = st_Glas) and (h^[i - 1] = st_Sogl)
      and (h^[i + 1] = st_Glas)
      and (h^[i + 2] = st_Sogl) then begin
        PChar(HypBuff)[cur] := HypSymb;
        Inc(Cur);
        Lock := 1;
    end;
    if (h^[i] = st_Spec) then begin
      PChar(HypBuff)[cur] := HypSymb;
      Inc(Cur);
      Lock:=1;
    end;

  end;
  FreeMem(h, len + 1);
  PChar(HypBuff)[cur] := #0;
  Result := HypBuff;
end;
```

```
function Red_GlasMore(p: PChar; pos: Integer): Boolean;
begin
  while p[pos] <> #0 do begin
    if p[pos] in Spaces then begin
      Result := False;
      Exit;
    end;
    if isGlas(p[pos]) then begin
      Result := True;
      Exit;
    end;
    Inc(pos);
  end;
  Result := False;
end;

function Red_SlogMore(p: PChar; pos: Integer): Boolean;
var
  BeSogl, BeGlas: Boolean;
begin
  BeSogl := False; BeGlas := False;
  while p[pos] <> #0 do begin
    if p[pos] in Spaces then Break;
    if not BeGlas then BeGlas := isGlas(p[pos]);
    if not BeSogl then BeSogl := isSogl(p[pos]);
    Inc(pos);
  end;
  Result := BeGlas and BeSogl;
end;

function MayBeHyph(p: PChar; pos: Integer): Boolean;
var
  i: Integer;
  len: Integer;
begin
  I := pos;
  Len := StrLen(p);
  Result := (Len > 3) and (i > 2) and (i < Len - 2)
    and (not (p[i] in Spaces))
    and (not (p[i + 1] in Spaces)) and (not (p[i-1] in Spaces))
    and ((isSogl(p[i]) and isGlas(p[i - 1]) and isSogl(p[i + 1])
    and Red_SlogMore(p, i + 1)) or ((isGlas(p[i]))
    and (isSogl(p[i - 1])) and (isSogl(p[i + 1]))
    and (isGlas(p[i + 2]))) or ((isGlas(p[i]))
    and (isSogl(p[i - 1])) and (isGlas(p[i + 1]))
    and Red_SlogMore(p, i + 1)) or ((isSpecSign(p[i])));
end;

function SetHyphString(s: String): String;
var
  Res: PChar;
```

```
begin
  Res := SetHyph(PChar(s), Length(s)*2);
  Result := Res;
  FreeMem(Res, Length(s)*2);
end;

end.
```

[Nomadic]

Примечание

Функция `SetHyphString` *выполняет непосредственный перенос текста по слогам, функция* `MayBeHyph` *– индицирует возможность переноса текста в указанной позиции,* `SetHyph` *– необходима для работы* `SetHyphString` *(отдельно не вызывается).*

Сумма прописью

Очень часто в финансовых приложениях сумму нужно писать прописью. Как сумму, представленную цифрой, преобразовать в строку прописью?

Решение 1

```
function TextSum(S: double): string;

  function Conv999(M: longint; fm: integer): string;
  const
    c1to9m: array [1..9] of string[6] =
      ('один', 'два', 'три', 'четыре', 'пять', 'шесть', 'семь', 'восемь','девять');
    c1to9f: array [1..9] of string[6] =
     ('одна', 'две', 'три', 'четыре', 'пять', 'шесть', 'семь', 'восемь', 'девять');
    c11to19: array [1..9] of string[12] =
       ('одиннадцать', 'двенадцать', 'тринадцать', 'четырнадцать', 'пятнадцать',
        'шестнадцать', 'семнадцать', 'восемнадцать', 'девятнадцать');
    c10to90: array [1..9] of string[11] =
       ('десять', 'двадцать', 'тридцать', 'сорок', 'пятьдесят', 'шестьдесят',
        'семьдесят', 'восемьдесят', 'девяносто');
    c100to900: array [1..9] of string[9] =
       ('сто', 'двести', 'триста', 'четыреста', 'пятьсот',
        'шестьсот', 'семьсот', 'восемьсот', 'девятьсот');

var
  s: String;
  i: Longint;
begin
  s := '';
  i := M div 100;
  if I <> 0 then s := c100to900[i] + ' ';
  M := M mod 100;
  i := M div 10;
  if (M > 10) and (M < 20) then
```

```
    s := s + c11to19[M - 10] + ' '
  else begin
    if I <> 0 then s := s + c10to90[i] + ' ';
    M := M mod 10;
    if M <> 0 then
      if fm = 0 then s := s + c1to9f[M] + ' '
      else s := s + c1to9m[M] + ' ';
  end;
  Conv999 := s;
end;

var
  i: Longint;
  j: Longint;
  r: Real;
  t: String;
begin
  t := '';
  j := Trunc(S / 1000000000.0);
  r := j;
  r := S - r*1000000000.0;
  i := Trunc(r);

  if j <> 0 then begin
    t := t + Conv999(j, 1) + 'миллиард';
    j := j mod 100;
    if (j > 10) and (j < 20) then t := t + 'ов '
    else
      case j mod 10 of
         0:  t := t + 'ов ';
         1:  t := t + ' ';
      2..4:  t := t + 'а ';
      5..9:  t := t + 'ов ';
      end;
  end;
  j := i div 1000000;
  if j <> 0 then begin
    t := t + Conv999(j, 1) + 'миллион';
    j := j mod 100;
    if (j > 10) and (j < 20) then t := t + 'ов '
    else
      case j mod 10 of
         0:  t := t + 'ов ';
         1:  t := t + ' ';
      2..4:  t := t + 'а ';
      5..9:  t := t + 'ов ';
      end;
  end;
  i := i mod 1000000;
  j := i div 1000;
  if j <> 0 then begin
```

```
    t := t + Conv999(j, 0) + 'тысяч';
    j := j mod 100;
    if (j > 10) and (j < 20) then t := t + ' '
    else
      case j mod 10 of
           0:  t := t + ' ';
           1:  t := t + 'а ';
        2..4:  t := t + 'и ';
        5..9:  t := t + ' ';
      end;
  end;
  i := i mod 1000;
  j := i;
  if j <> 0 then t := t + Conv999(j, 1);
  t := t + 'руб. ';
  i := Round(Frac(S)*100.0);
  t := t + IntToStr(i) + ' коп.';
  TextSum := t;
end;
```

[Александр]

Решение 2

```
unit Numinwrd;

interface
  function sMoneyInWords(Nin: currency): string; export;
  function szMoneyInWords(Nin: currency): PChar; export;

implementation

uses
  SysUtils, Dialogs, Math;

type
  tri = string[4];
  mood = 1..2;
  gender = (m, f);
  uns = array[0..9] of string[7];
  tns = array[0..9] of string[13];
  decs = array[0..9] of string[12];
  huns = array[0..9] of string[10];
  nums = array[0..4] of string[8];
  money = array[1..2] of string[5];
  endings = array[gender,mood,1..3] of tri;       // окончания числительных и денег

const
  units: uns = ('', 'один ', 'два ', 'три ', 'четыре ', 'пять ',
                'шесть ', 'семь ', 'восемь ', 'девять ');
  unitsf: uns = ('', 'одна ', 'две ', 'три ', 'четыре ',
                 'пять ', 'шесть ','семь ','восемь ','девять ');
```

```
  teens: tns = ('десять ', 'одиннадцать ', 'двенадцать ', 'тринадцать ',
                'четырнадцать ', 'пятнадцать ', 'шестнадцать ',
                'семнадцать ', 'восемнадцать ', 'девятнадцать ');
  decades: decs = ('', 'десять ', 'двадцать ', 'тридцать ', 'сорок ',
                   'пятьдесят ', 'шестьдесят ', 'семьдесят ',
                   'восемьдесят ', 'девяносто ');
  hundreds: huns=('', 'сто ', 'двести ', 'триста ', 'четыреста ', 'пятьсот ',
                  'шестьсот ', 'семьсот ', 'восемьсот ', 'девятьсот ');
  numericals: nums = ('', 'тысяч', 'миллион', 'миллиард', 'триллион');
  RusMon: money = ('рубл', 'копе');
  ends: endings = ((('', 'а', 'ов'), ('ь', 'я', 'ей')),
                   (('а', 'и', ''), ('йка', 'йки', 'ек')));

threadvar
  str: string;

function EndingIndex(Arg: integer): integer;
begin
  if ((Arg div 10) mod 10) <> 1 then
    case (Arg mod 10) of
       1:  Result := 1;
    2..4:  Result := 2;
    else   Result := 3;
    end
  else Result := 3;
end;

{ Число Nin прописью, как функция }
function sMoneyInWords(Nin: currency): string;

var
  g: gender;                  // род
  Nr: comp;                   // целая часть числа
  Fr: integer;                // дробная часть числа
  i, iTri, Order: longint;    // триада

  procedure Triad;
  var
    iTri2: integer;
    un, de, ce: byte;         // единицы, десятки, сотни

    function GetDigit: byte;
    begin
      Result := iTri2 mod 10;
      iTri2 := iTri2 div 10;
    end;

  begin
    iTri := trunc(Nr/IntPower(1000, i));
    Nr := Nr - int(iTri * IntPower(1000, i));
    iTri2:= iTri;
```

```
    if iTri > 0 then begin
      un := GetDigit;
      de := GetDigit;
      ce := GetDigit;
      if i = 1 then g := f

      else g := m;           // женского рода только тысяча
      str := TrimRight(str) + ' ' + Hundreds[ce];
      if de = 1 then
        str := TrimRight(str) + ' ' + Teens[un]

      else begin
        str := TrimRight(str) + ' ' + Decades[de];
        case g of
          m:  str := TrimRight(str) + ' ' + Units[un];
          f:  str := TrimRight(str) + ' ' + UnitsF[un];
        end;

      end;
      if length(numericals[i]) > 1 then begin
        str := TrimRight(str) + ' ' + numericals[i];
        str := TrimRight(str) + ends[g, 1, EndingIndex(iTri)];
      end;
    end;                  // триада 0 ?
    if I = 0 then Exit;
    Dec(i);
    Triad;
  end;

begin
  str := '';
  Nr := int(Nin);
  Fr := round(Nin*100 + 0.00000001) mod 100;
  if Nr > 0 then
    Order := trunc(Log10(Nr) / 3)
  else begin
    str := 'ноль';
    Order := 0
  end;
  if Order > High(numericals) then
    raise Exception.Create('Слишком большое число для суммы прописью');
  i := Order;
  Triad;
  str := Format('%s %s%s %.2d %s%s', [Trim(str), RusMon[1],
      ends[m, 2, EndingIndex(iTri)], Fr, RusMon[2],
      ends[f, 2, EndingIndex(Fr)]]);
  str[1] := (ANSIUpperCase(copy(str, 1, 1)))[1];

  Result := str + #0;
end;
```

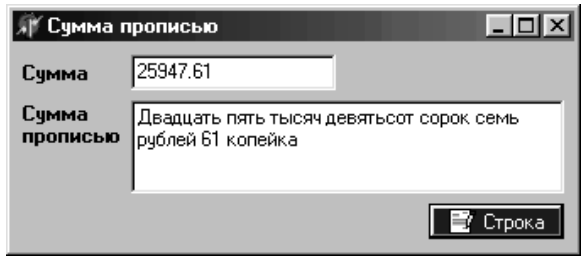
```
function szMoneyInWords(Nin: currency): PChar;
begin
  sMoneyInWords(Nin);
  Result := @(str[1]);
end;

end.
```

[Клюкач Олег]

Примечание

Функция sMoneyInWords *выдает результат типа* string*, а функция* szMoneyInWords *– типа* PChar*.*

Решение 3




```
unit valtostr;

interface

uses
  SysUtils, Math;

function SumStr(Sum: double):
string;

implementation

function SumStr(Sum: double): string;

type
  TSex = (Male, Female);

const
  MaxSum = 1000000000000 - 1;
  Ind: array [1..3, 1..3] of string = (('тысяча', 'тысячи', 'тысяч'),
                                       ('миллион', 'миллиона', 'миллионов'),
                                       ('миллиард', 'миллиарда', 'миллиардов'));

  Curr: array [1..2, 1..3] of string = (('рубль', 'рубля', 'рублей'),
                                        ('копейка', 'копейки', 'копеек'));

function ValToStr(Sum: word; Sex: TSex): string;

const
  f1to9m: array [1..9] of string = ('один', 'два', 'три', 'четыре', 'пять',
                                    'шесть', 'семь', 'восемь', 'девять');

  f1to9f: array [1..9] of string = ('одна', 'две', 'три', 'четыре', 'пять',
                                    'шесть', 'семь', 'восемь', 'девять');
```

```
  f11to19: array [1..9] of string = ('одиннадцать', 'двенадцать', 'тринадцать',
                                     'четырнадцать', 'пятнадцать', 'шестнадцать',
                                     'семнадцать', 'восемнадцать', 'девятнадцать');

  f10To90: array [1..9] of string = ('десять', 'двадцать', 'тридцать', 'сорок',
                                     'пятьдесят', 'шестьдесят', 'семьдесят',
                                     'восемьдесят', 'девяносто');

  f100to900: array [1..9] of string = ('сто', 'двести', 'триста', 'четыреста',
                                       'пятьсот', 'шестьсот', 'семьсот',
                                       'восемьсот', 'девятьсот');
  var
    Val: integer;
  begin
    result := '';
    Val := Sum div 100;
    if Val <> 0 then result := f100to900[Val] + ' ';
    Sum := Sum mod 100;
    Val := Sum div 10;
    if ((Sum > 10) and (Sum < 20)) then
      result := result + f11to19[Sum-10] + ' '
    else begin
      if Val <> 0 then result := result + f10to90[Val] + ' ';
      Sum := Sum mod 10;
      if Sum <> 0 then
        if Sex = Male then result := result + f1to9m[Sum] + ' '
        else result := result + f1to9f[Sum] + ' ';
    end;
  end;

var
  Sym: string;
  Val: double;
  Cnt, LastDidg: byte;
  Rub, Divisor: int64;
  Kop, Dividend: word;

begin
  result := '';
  if Sum < 0 then Exit;
  if Sum > MaxSum then
    raise Exception.Create('Превышение максимально допустимого значения');
  Val := Sum - Trunc(Sum);
  Rub := Trunc(Sum - Val);
  Kop := Round((Sum - Trunc(Sum))*100);
  if Rub = 0 then
    result := 'Ноль рублей '
  else
    for Cnt := 3 downto 0 do begin
      Divisor := Trunc(Power(1000, Cnt));
```

```
      Dividend := Rub div Divisor;
      Rub := Rub - Dividend*Divisor;
      if Dividend <> 0 then begin
        if Cnt = 1 then result := result + ValToStr(Dividend, Female)
        else result := result + ValToStr(Dividend, Male);
        if Cnt > 0 then begin
          LastDidg := Dividend mod 10;
          case LastDidg of
             0:  result := result + Ind[Cnt, 3] + ' ';
             1:  result := result + Ind[Cnt, 1] + ' ';
          2..4:  result := result + Ind[Cnt, 2] + ' ';
          5..9:  result := result + Ind[Cnt, 3] + ' ';
          end;
        end else begin
          LastDidg := Dividend mod 100;
          if ((LastDidg > 4) and (LastDidg < 21)) then
            result := result + Curr[1, 3] + ' '
          else begin
            LastDidg := Dividend mod 10;
            case LastDidg of
               0:  result := result + Curr[1, 3] + ' ';
               1:  result := result + Curr[1, 1] + ' ';
            2..4:  result := result + Curr[1, 2] + ' ';
            5..9:  result := result + Curr[1, 3] + ' ';
            end;
          end;
        end;
      end;
    end;
  if Kop = 0 then
    result := result + '00 копеек'
  else begin
    if Kop < 10 then result := result + '0' + IntToStr(Kop) + ' '
    else result := result + IntToStr(Kop) + ' ';
    if ((Kop > 4) and (Kop < 21)) then
      result := result + Curr[2, 3]
    else begin
      LastDidg := Kop mod 10;
      case LastDidg of
         0:  result := result + Curr[2, 3];
         1:  result := result + Curr[2, 1];
      2..4:  result := result + Curr[2, 2];
      5..9:  result := result + Curr[2, 3];
      end;
    end;
  end;
  Sym := AnsiUpperCase(result[1]);
  result[1] := Sym[1];
end;
end.
```

[VS]

Решение 4

Для англоязычного варианта

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Label1: TLabel;
    num: TEdit;
    spell: TEdit;
    procedure Button1Click(Sender: TObject);
  private
    function trans9(num: integer): string;
    function trans19(num: integer): string;
    function trans99(num: integer): string;
    function IntToSpell(num: integer): string;
  end;

var
  Form1: TForm1;

implementation
{$R *.DFM}

function TForm1.IntToSpell(num: integer): string;
var
  spell: string;
  hspell: string;
  hundred: string;
  thousand: string;
  tthousand: string;
  hthousand: string;
  million: string;
begin
  if num < 10 then spell := trans9(num);
  if (num < 20) and (num > 10) then spell := trans19(num);
  if (((num < 100) and (num > 19)) or (num = 10)) then begin
    hspell := copy(IntToStr(num), 1, 1) + '0';
    spell := trans99(StrToInt(hspell));
    hspell := copy(IntToStr(num), 2, 1);
    spell := spell + ' ' + IntToSpell(StrToInt(hspell));
  end;
  if (num < 1000) and (num > 100) then begin
    hspell := copy(IntToStr(num), 1, 1);
```

```
    hundred := IntToSpell(StrToInt(hspell));
    hspell := copy(IntToStr(num), 2, 2);
    hundred := hundred + ' hundred and ' + IntToSpell(StrToInt(hspell));
    spell := hundred;
  end;
  if (num < 10000) and (num > 1000) then begin
    hspell := copy(IntToStr(num), 1, 1);
    thousand := IntToSpell(StrToInt(hspell));
    hspell := copy(IntToStr(num), 2, 3);
    thousand := thousand + ' thousand ' + IntToSpell(StrToInt(hspell));
    spell := thousand;
  end;
  if (num < 100000) and (num > 10000) then begin
    hspell := copy(IntToStr(num), 1, 2);
    tthousand := IntToSpell(StrToInt(hspell));
    hspell := copy(IntToStr(num), 3, 3);
    tthousand := tthousand + ' thousand ' + IntToSpell(StrToInt(hspell));
    spell := tthousand;
  end;
  if (num < 1000000) and (num > 100000) then begin
    hspell := copy(IntToStr(num), 1, 3);
    hthousand := IntToSpell(StrToInt(hspell));
    hspell := copy(IntToStr(num), 4, 3);
    hthousand := hthousand + ' thousand and ' + IntToSpell(StrToInt(hspell));
    spell := hthousand;
  end;
  if (num < 10000000) and (num > 1000000) then begin
    hspell := copy(IntToStr(num), 1, 1);
    million := IntToSpell(StrToInt(hspell));
    hspell := copy(IntToStr(num), 2, 6);
    million := million + ' million and ' + IntToSpell(StrToInt(hspell));
    spell := million;
  end;
  IntToSpell := spell;
end;

function TForm1.trans99(num: integer): string;
var
  spell: string;
begin
  case num of
    10:  spell := 'ten';
    20:  spell := 'twenty';
    30:  spell := 'thirty';
    40:  spell := 'forty';
    50:  spell := 'fifty';
    60:  spell := 'sixty';
    70:  spell := 'seventy';
    80:  spell := 'eighty';
    90:  spell := 'ninety';
  end;
```

```
  trans99 := spell;
end;

function TForm1.trans19(num: integer): string;
var
  spell: string;
begin
  case num of
    11:  spell := 'eleven';
    12:  spell := 'twelve';
    13:  spell := 'thirteen';
    14:  spell := 'fourteen';
    15:  spell := 'fifteen';
    16:  spell := 'sixteen';
    17:  spell := 'seventeen';
    18:  spell := 'eighteen';
    19:  spell := 'nineteen';
  end;
  trans19 := spell;
end;

function TForm1.trans9(num: integer): string;
var
  spell: string;
begin
  case num of
    1:  spell := 'one';
    2:  spell := 'two';
    3:  spell := 'three';
    4:  spell := 'four';
    5:  spell := 'five';
    6:  spell := 'six';
    7:  spell := 'seven';
    8:  spell := 'eight';
    9:  spell := 'nine';
  end;
  trans9 := spell;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  numb: integer;
begin
  spell.text := IntToSpell(StrToInt(num.text));
end;
end.
```

Примечание

В данном варианте программы можно работать только с целыми числами, отсутствует написание денежной единицы.

Проверка кредитной карты

Данный модуль создан на основе алгоритма ccard Питера Миллера (Peter Miller). Автор не против бесплатного использования, но резервирует все права на данный алгоритм.

Примечание

Внесите данный модуль в список uses *любого модуля, которому необходим доступ к функции проверки кредитной карты.*

```
IsValidCreditCardNumber(CardNumber, ReturnMessage) returns Boolean
```

Таким образом можно, например, сообщить пользователю о недействительности карты.

CardNumber – строка, содержащая номер карты, которую необходимо проверить. ReturnMessage – строка, с помощью которой функция может возвратить любое сообщение (при этом старое содержимое строки стирается). Функция возвращает значение True, если номер карточки верен, и False – в противном случае.

Во входных параметрах функции допускаются тире и пробелы, если же встретятся другие символы, их можно удалить. Функция RemoveChar довольно легко реализует данную операцию. Для этого передайте ей входную строку и символ, который необходимо удалить.

Разрешаются любые изменения кода модуля для собственных целей, но в случае его распространения необходимо сообщить другим пользователям обо всех внесенных изменениях.

Вы можете воспользоваться предлагаемым модулем на свой страх и риск, не забывая при этом, что ответственность за какой-либо ущерб, причиненный данным модулем, лежит только на его пользователе.

На момент написания модуля он устойчиво работал под Delphi версий 1 и 2. Для Turbo Pascal необходимо внести некоторые несложные исправления (главным образом из-за различия реализации функций в модуле SysUtils).

```
unit Creditc;

interface
uses
  SysUtils;

function IsValidCreditCardNumber(CardNumber: String; var MessageText: String): Boolean;

implementation

const
  CardPrefixes: array[1..19] of string =
   ('2014', '2149', '300', '301', '302', '303', '304', '305', '34',
    '36', '37', '38', '4', '51', '52', '53', '54', '55', '6011');
```

```
  CardTypes: array[1..19] of String =
   ('enRoute',
    'enRoute',
    'Diner Club/Carte Blanche',
    'Diner Club/Carte Blanche',
    'Diner Club/Carte Blanche',
    'Diner Club/Carte Blanche',
    'Diner Club/Carte Blanche',
    'Diner Club/Carte Blanche',
    'American Express',
    'Diner Club/Carte Blanche',
    'American Express',
    'Diner Club/Carte Blanche',
    'Visa',
    'MasterCard',
    'MasterCard',
    'MasterCard',
    'MasterCard',
    'MasterCard',
    'Discover');

function RemoveChar(const Input: String; DeletedChar: Char): String;
{ Данная функция удаляет все вхождения указанного символа из переданной ей строки }
var
  Index: Word;
begin
  Result := Input;
  for Index := Length(Result) downto 1 do
    if Result[Index] = DeletedChar then Delete(Result, Index, 1);
end;

function ShiftMask(Input: Integer): Integer;
{ Простая оболочка для функции сдвига битов числа }
begin
  result := (1 shl (Input - 12));
end;

{ Это, вероятно, самый запутанный код, который вы когда-либо видели.
  Основное, что делает данная функция, - извлекает каждую цифру из номера карты
  для использования в формуле проверки контрольной суммы, устанавливаемой
  компаниями. Алгоритм производит выборку, начиная от последней цифры
  и заканчивая первой. }
function ConfirmChecksum(CardNumber: String): Boolean;
var
  CheckSum: Integer;     // Содержит значение операции
  Flag: Boolean;         // флаг готовности
  Counter: Integer;      // индекс счетчика
  PartNumber: String;    // используется для извлечения каждой цифры числа
  Number: Integer;       // используется для преобразования каждой цифры в число
```

```
begin
{ получаем стартовое значение счетчика }
  Counter := Length(CardNumber);
  CheckSum := 0;
  PartNumber := '';
  Number := 0;
  Flag := false;
  while (Counter >= 1) do begin

{ получаем текущую цифру }
    PartNumber := Copy(CardNumber, Counter, 1);
    Number := StrToInt(PartNumber);
    if (Flag) then begin      { только каждую вторую цифру }
      Number := Number * 2;
      if (Number >= 10) then Number := Number - 9;
    end;
    CheckSum := CheckSum + Number;
    Flag := not(Flag);
    Counter := Counter - 1;
  end;
  result := ((CheckSum mod 10) = 0);
end;

function GetMask(CardName: String): Integer;
begin
  result := 0;                 // значение по умолчанию
  if (CardName = 'MasterCard') then result := ShiftMask(16);
  if (CardName = 'Visa') then result := (ShiftMask(13) or ShiftMask(16));
  if (CardName = 'American Express') then result := ShiftMask(15);
  if (CardName = 'Diner Club/Carte Blanche') then result := ShiftMask(14);
  if (CardName = 'Discover') then result := ShiftMask(16);
end;

function IsValidCreditCardNumber(CardNumber: String; var MessageText: String): Boolean;
var
  StrippedNumber: String;   // используется для хранения числа без
                            // дополнительных символов
  Index: Integer;           // универсальный счетчик для циклов и т.п.
  TheMask: Integer;         // число, которое мы будем использовать для маски
  FoundIt: Boolean;         // используется для индикации, когда что-либо найдено
  CardName: String;         // хранит имя типа карты
  PerformChecksum: Boolean; // тип enRoute карты, если контрольная сумма не сошлась
begin
{ сначала избавимся от пробелов и тире }
  StrippedNumber := RemoveChar(CardNumber, ' ');
  StrippedNumber := RemoveChar(StrippedNumber, '-');
{ если строка была нулевой длины, то тоже OK }
  if (StrippedNumber = '') then begin
    result := true;
    exit;
  end;
```

```
{ инициализация возвращаемых переменных }
  MessageText := '';
  result := true;
{ устанавливаем нашу переменную-флаг }
  FoundIt := false;
{ проверка правильности введенных символов в номере карты }
  for Index := 1 to Length(StrippedNumber) do begin
    case StrippedNumber[Index] of
      '0'..'9': FoundIt := FoundIt;  { другими словами не ор }
       else
         MessageText := 'Неверно введенный символ';
         result := false;
         exit;
    end;
  end;
{ теперь давайте определим тип используемой карты }
  for Index := 1 to 19 do
    if (Pos(CardPrefixes[Index], StrippedNumber) = 1) then begin
      { мы обнаружили правильный тип }
      FoundIt := true;
      CardName := CardTypes[Index];
      TheMask := GetMask(CardName);
    end;
{ если тип карты не определен, указываем на это }
  if not FoundIt then begin
    CardName := 'Unknown Card Type';
    TheMask := 0;
    MessageText := 'Неизвестный тип карты ';
    result := false;
    exit;
  end;
  if ((Length(StrippedNumber) > 28) and result) then begin   { проверка длины }
    MessageText := 'Номер слишком большой ';
    result := false;
    exit;
  end;
{ проверка длины }
  if ((Length(StrippedNumber) < 12) or ((shiftmask(length(strippednumber))
    and themask) = 0)) then begin
      messagetext := 'неверная длина числа';
      result := false;
      exit;
  end;
{ проверяем вычисление контрольной суммы }
  if (cardname = 'enroute') then performchecksum := false
  else performchecksum := true;
  if (performchecksum and (not confirmchecksum(strippednumber))) then begin
      messagetext := 'неверная контрольная сумма';
      result := false;
      exit;
  end;
```

```
{ если результат равен true, тогда все ok }
  if (result) then messagetext := 'номер верен: тип карты: ' + cardname;
{ если строка была нулевой длины, то тоже OK }
  if (strippednumber = '') then result := true;
end;

end.
```

[News Group]

Проверка ISBN

ISBN (или International Standard Book Numbers, стандартные международные номера книг) – мистические кодовые числа, однозначно идентифицирующие книги. Цель этого совета заключается в том, чтобы убрать покров таинственности, окружающий структуру ISBN, и в качестве примера разработать приложение, проверяющее правильность создания кода-кандидата на ISBN.

ISBN имеет длину тринадцать символов, которые могут быть цифрами от 0 до 9, дефисом или буквой «X». Этот код состоит из четырех частей (между которыми стоит дефис): идентификатор группы, идентификатор издателя, идентификатор книги для издателя и контрольная цифра. Первая часть (идентификатор группы) используется для обозначения страны, географического региона, языка и пр. Вторая часть (идентификатор издателя) однозначно идентифицирует издателя. Третья часть (идентификатор книги) однозначно идентифицирует данную книгу среди коллекции книг, выпущенных данным издателем. Четвертая, заключительная часть (контрольная цифра), используется в коде алгоритма другими цифрами для получения поддающегося проверке ISBN. Количество цифр, содержащихся в первых трех частях, может быть различным, но контрольная цифра всегда содержит один символ (расположенный между «0» и «9» включительно, или «X» для величины 10). Таким образом, ISBN имеет длину тринадцать символов (десять чисел плюс три дефиса, разделяющих три части ISBN).

ISBN 3-88053-002-5 можно разложить на части следующим образом:

Группа:	3
Издатель:	88053
Книга:	002
Контрольная цифра:	5

ISBN можно проверить на правильность кода с помощью простого математического алгоритма. Суть его в следующем: нужно взять каждую из девяти цифр первых трех частей ISBN (пропуская дефисы), умножить каждую отдельную цифру на ее позицию в коде ISBN (считая справа от контрольной цифры), сложить эти произведения и прибавить контрольную цифру, после

чего разделить получившееся число на одиннадцать. Если после процедуры деления остатка нет (т. е. число по модулю 11 делится без остатка), кандидат на ISBN является верным кодом ISBN. Например, используем предыдущий образец ISBN 3-88053-002-5:

ISBN:	3 8 8 0 5 3 0 0 2 5
Множитель:	10 9 8 7 6 5 4 3 2 1
Продукт:	30+72+64+00+30+15+00+00+04+05 = 220

Поскольку 220 на одиннадцать делится без остатка, рассмотренный нами кандидат на ISBN является верным кодом ISBN.

Вот пример этой методики, изложенной на Delphi:

```
function IsISBN(ISBN: String): Boolean;
var
  Number, CheckDigit: String;
  CheckValue, CheckSum, Err: Integer;
  i, Cnt: Word;
begin
{ Получаем контрольную цифру }
  CheckDigit := Copy(ISBN, Length(ISBN), 1);
{ Получаем остальную часть, ISBN минус контрольная цифра и дефис }
  Number := Copy(ISBN, 1, Length(ISBN) - 2);
{ Длина разницы ISBN должны быть 11 и контрольная цифра между 0 и 9, или X }
  if (Length(Number) = 11) and (Pos(CheckDigit, '0123456789X') > 0) then begin
    { Получаем числовое значение контрольной цифры }
    if (CheckDigit = 'X') then CheckSum := 10
    else Val(CheckDigit, CheckSum, Err);
{ Извлекаем в цикле все цифры из кода ISBN, применяя алгоритм декодирования }
    Cnt := 1;
    for i := 1 to 11 do begin
{ Действуем, если текущий символ находится между "0" и "9", исключая дефисы }
      if (Pos(Number[i], '0123456789') > 0) then begin
        Val(Number[i], CheckValue, Err);
{ Алгоритм для каждого символа кода ISBN, Cnt - n-й обрабатываемый символ }
        CheckSum := CheckSum + CheckValue * (11 - Cnt);
        Inc(Cnt);
      end;
    end;
{ Проверяем делимость без остатка полученного значения на 11 }
    if (CheckSum mod 11 = 0) then IsISBN := True
    else IsISBN := False;
  end else IsISBN := False;
end;
```

[News Group]

Генерация еженедельных списков задач

Необходима программа, которая генерировала бы еженедельные списки задач. Программа должна просто показывать количество недель в списке задач и организовывать мероприятия, не совпадающие по времени. В предлагаемом текущем планировщике имеются 12 групп и планы на 11 недель.

Решение

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm1 = class(TForm)
    ListBox1: TListBox;
    Edit1: TEdit;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

const
  maxTeams = 100;

var
  Teams: array[1..maxTeams] of integer;
  nTeams, ix, week, savix: integer;

function WriteBox(week: integer): string;
var
  str: string;
  ix: integer;
begin
  Result := Format('Неделя=%d ', [week]);
  for ix := 1 to nTeams do begin
    if odd(ix) then Result := Result + ' '
    else Result := Result + 'v';
    Result := Result + IntToStr(Teams[ix]);
  end;
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  nTeams := StrToInt(Edit1.Text);
  if Odd(nTeams) then inc(nTeams);   // должны иметь номера каждой группы
  ListBox1.Clear;
  for ix := 1 to nTeams do
    Teams[ix] := ix;
  ListBox1.Items.Add(WriteBox(1));
  for week := 2 to nTeams - 1 do begin
    Teams[1] := Teams[nTeams - 1];   // используем Teams[1] в качестве
                                     // временного хранилища
    for ix := nTeams downto 2 do
      if not Odd(ix) then begin
        savix := Teams[ix];
        Teams[ix] := Teams[1];
        Teams[1] := savix;
      end;
    for ix := 3 to nTeams - 1 do
      if Odd(ix) then begin
        savix := Teams[ix];
        Teams[ix] := Teams[1];
        Teams[1] := savix;
      end;
    Teams[1] := 1;                   // восстанавливаем известное значение
    ListBox1.Items.Add(WriteBox(week));
  end;
end;

end.
```

[News Group]

Правильное округление дробных чисел

Как научить Delphi правильно округлять дробные числа?

Решение 1

```
function RoundEx(X: Double; Precision : Integer): Double;
{ Precision : 1 - до целых, 10 - до десятых, 100 - до сотых... }
var
  ScaledFractPart, Temp: Double;
begin
  ScaledFractPart := Frac(X) * Precision;
  Temp := Frac(ScaledFractPart);
  ScaledFractPart := Int(ScaledFractPart);
  if Temp >= 0.5 then ScaledFractPart := ScaledFractPart + 1;
  if Temp <= -0.5 then ScaledFractPart := ScaledFractPart - 1;
  RoundEx := Int(X) + ScaledFractPart / Precision;
end;
```

[Nomadic]

Решение 2

`Conv(2.005, 2)` возвращает `2.01`; `Conv(2.5, 0)` возвращает `3`.

```
function Conv(cs: double; numb: integer): double;
var
  db, db1, db2: double;
  i: int64;
  ii, ink, i1: integer;
  st: string;
begin
  db := cs - int(cs);
  ink := 1;
  for ii := 1 to numb do ink := ink * 10;
  db1 := db * ink;
  db2 := cs * ink * 100;
  i := trunc(int(db2) / 100);
  i1 := trunc(db2 - i * 100);
  if i1 > 49 then inc(i);
  result := i / ink;
end;
```

[Kordyum Alexandr]

Примечание

Автор этой программы, очевидно, не знал, что кроме положительных чисел есть и отрицательные. Читателям предлагается самим (ради тренировки) ликвидировать эту оплошность – программа правильно округляет только положительные числа.

Хочу подсказать еще один способ округления:

```
NumStr := FloatToStrF(Temp, ffFixed, 16, 2);
```

`NumStr` – строка приемник, `Temp` – число, `ffFixed` – формат, `16` – общая длина полученного числа, `2` – количество знаков после запятой.

Этот способ примечателен тем, что существует пять стандартных различных форматов: `ffGeneral`, `ffExponent`, `ffFixed`, `ffNumber`, `ffCurrency` – это позволяет найти себе нужный, на свой вкус. Если же вам необходимо получить результат в числовой форме, то можно написать:

```
Temp := StrToFloat(FloatToStrF(Temp, ffFixed, 16, 2));
```

Решение 3

```
function RoundFloat(R: Extended; Decimals: Integer): Extended;

var
  Factor: Extended;

begin
  Factor := Int(Exp(Decimals * Ln(10)));
```

```
  Result := Round(Factor * R) / Factor;
end;
```

[News Group]

Почему возникает ошибка при использовании функции StrToFloat

В моей программе функция `StrToFloat('32.34')` *вызывает исключение «'32.34' is not valid float». Если число записывается без десятичной точки, такой проблемы не возникает. Почему?*

В установках русской редакции Windows по умолчанию считается, что разделитель дроби – запятая. Изменить данный параметр можно либо с помощью Панели управления, либо программно:

```
DecimalSeparator := '.';
```

Или вызвать функцию так:

```
StrToFloat('32,24');
```

[Nomadic]

Эквивалент Trim$(), Mid$() и другие

Удаление конечных/начальных пробелов и левых/правых частей строк может осуществляться с помощью следующих решений:

Решение 1

```
unit TrimStr;
{$B-}

{ Автор: Bob Swart [100434, 2072]
  Доработка под Delphi: Тугаев Олег, knsprog@krintel.ru

  LTrim() - Удаляет все пробелы в левой части строки
  RTrim() - Удаляет все пробелы в правой части строки
  Trim() - Удаляет все пробелы по краям строки
  RightStr() - Возвращает правую часть строки заданной длины
  LeftStr() - Возвращает левую часть строки заданной длины
  MidStr() - Возвращает центральную часть строки заданной длины }

interface

const
  Space = #$20;

function LTrim(const Str: String): String;
function RTrim(Str: String): String;
function Trim(Str: String): String;
```

```
function RightStr(const Str: String; Size: Word): String;
function LeftStr(const Str: String; Size: Word): String;
function MidStr(const Str: String; Size: Word): String;

implementation

function LTrim(const Str: String): String;
var
  len, i: Word;
begin
  i := 1;
  len := Length(Str);
  while (i <= len) and (Str[i] = Space) do Inc(i);
  LTrim := Copy(Str, i, len)
end;

function RTrim(Str: String): String;
var
  len: Word;
begin
  len := Length(Str);
  while (len > 0) and (Str[len] = Space) do Dec(len);
  RTrim := Copy(Str, 1, len);
end;

function Trim(Str: String): String;
begin
  Trim := LTrim(RTrim(Str))
end;

function RightStr(Const Str: String; Size: Word): String;
var
  len: Word;
begin
  len := Length(Str);
  if Size >= len then Size := len
  else RightStr := Copy(Str, len - Size + 1, Size)
end;

function LeftStr(Const Str: String; Size: Word): String;
begin
  LeftStr := Copy(Str, 1, Size)
end;

function MidStr(Const Str: String; Size: Word): String;
var
  len: Word;
begin
  len := Length(Str);
  if Size >= len then Size := len
```

```
else MidStr := Copy(Str, ((len - Size) div 2) + 1, Size)
end;

end.
```

[News Group]

Решение 2

Для Mid$ используйте функцию

```
Copy(S: string; index, count: integer): string;
```

С помощью команды Copy можно осуществить операции Right$ и Left$, выполнив:

Copy(S, 1, Length) для Left$ и Copy(S, Start, 255) для Right$.

Примечание

Index *и* Length *– байтовые позиции вашего «отправного пункта», их можно получить с помощью* Pos().

Некоторые дополнительные функции:

```
const
  BlackSpace = [#33..#126];

{ Возвращает строку со всеми пробельными символами и с удаленными повторяющимися
  апострофами. }
function squish(const Search: string): string;
var
  Index: byte;
  InString: boolean;
begin
  InString := False;
  Result := '';
  for Index := 1 to Length(Search) do begin
    if InString or (Search[Index] in BlackSpace) then
      AppendStr(Result, Search[Index]);
    InString := ((Search[Index] = '''') and (Search[Index - 1] <> '\'))
      xor InString;
  end;
end;

{ Возвращает часть строки, находящейся перед первой найденной подстрокой Find
  в строке Search. Если Find не найдена, функция возвращает параметр Search. }
function before(const Search, Find: string): string;
var
  index: byte;
begin
  index := Pos(Find, Search);
```

```
  if index = 0 then Result := Search
  else Result := Copy(Search, 1, index - 1);
end;

{ Возвращает часть строки, находящейся после первой найденной подстроки Find
  в строке Search. Если Find не найдена, функция возвращает NULL. }

function after(const Search, Find: string): string;
var
  index: byte;
begin
  index := Pos(Find, Search);
  if index = 0 then Result := ''
  else Result := Copy(Search, index + Length(Find), 255);
end;

{ Возвращает первый символ последней найденной подстроки Find в строке Search. Если
  Find не найдена, функция возвращает 0. Подобна реверсированной Pos() }

function RPos(const Find, Search: string): byte;
var
  FindPtr, SearchPtr, TempPtr: PChar;
begin
  FindPtr := StrAlloc(Length(Find) + 1);
  SearchPtr := StrAlloc(Length(Search) + 1);
  StrPCopy(FindPtr, Find);
  StrPCopy(SearchPtr, Search);
  Result := 0;
  repeat
    TempPtr := StrRScan(SearchPtr, FindPtr^);
    if TempPtr <> nil then
      if (StrLComp(TempPtr, FindPtr, Length(Find)) = 0) then begin
        Result := TempPtr - SearchPtr + 1;
        TempPtr := nil;
      end else
        TempPtr := #0;
  until TempPtr = nil;
end;

{ Возвращает подстроку, вложенную между парой подстрок Front ... Back. }
function inside(const Search, Front, Back: string): string;
var
  Index, Len: byte;
begin
  Index := RPos(Front, before(Search, Back));
  Len := Pos(Back, Search);
  if (Index > 0) and (Len > 0) then
    Result := Copy(Search, Index + 1, Len - (Index + 1))
  else
    Result := '';
end;
```

```
{ Возвращает левую часть "остатка" inside() или Search. }
function leftside(const Search, Front, Back: string): string;
begin
  Result := before(Search, Front + inside(Search, Front, Back) + Back);
end;

{ Возвращает правую часть "остатка" inside() или Null. }
function rightside(const Search, Front, Back: string): string;
begin
  Result := after(Search, Front + inside(Search, Front, Back) + Back);
end;

{ Возвращает строку со всеми удаленными по краям белыми пробелами. }
function trim(const Search: string): string;
var
  Index: byte;
begin
  Index := 1;
  while (Index <= Length(Search)) and not (Search[Index] in BlackSpace) do
    Index:=Index + 1;
  Result := Copy(Search, Index, 255);
  Index := Length(Result);
  while (Index > 0) and not (Result[Index] in BlackSpace) do Index:=Index - 1;
    Result := Copy(Result, 1, Index);
end;
```

[News Group]

Примечание

Будьте внимательны – символы в строках нумеруются, начиная с 1.

Разбивка строки на слова

Вашему вниманию предлагается пара простых функций, позволяющих работать с отдельными словами в строке. Они могут использоваться для разбивки текстовых полей на отдельные слова (`for i := 1 to NumToken do ...`) с последующим их сохранением в базе данных.

Решение 1

```
function GetToken(aString: String; SepChar: Char; TokenNum: Word): String;
{ параметры:
  aString: полная строка
  SepChar: символ, служащий разделителем между словами
  TokenNum: номер требуемого слова (подстроки)
  result: искомое слово или пустая строка, если количество слов меньше значения
         'TokenNum' }
var
  Token: String;
  StrLen: Byte;
  TNum: Byte;
```

```
  TEnd: Byte;
begin
  StrLen := Length(aString);
  TNum := 1;
  TEnd := StrLen;
  while ((TNum <= TokenNum) and (TEnd <> 0)) do begin
    TEnd := Pos(SepChar, aString);
    if TEnd <> 0 then begin
      Token := Copy(aString, 1, TEnd - 1);
      Delete(aString, 1, TEnd);
      Inc(TNum);
    end
    else Token := aString;
  end;
  if TNum >= TokenNum then GetToken := Token
  else GetToken := '';
end;

function NumToken(aString: String; SepChar: Char): Word;
{ параметры:
  aString: полная строка
  SepChar: символ, служащий разделителем между словами (подстроками)
  result: количество найденных слов (подстрок) }
var
  RChar: Char;
  StrLen: Byte;
  TNum: Byte;
  TEnd: Byte;
begin
  if SepChar = '#' then RChar := '*'
  else RChar := '#';
  StrLen := Length(aString);
  Tnum := 0;
  Tend := StrLen;
  while TEnd <> 0 do begin
    Inc(TNum);
    TEnd := Pos(SepChar, aString);
    if TEnd <> 0 then aString[TEnd] := RChar;
  end;
  Result := TNum -1;
end;
```

Решение 2

```
function CopyColumn(const sstring: string; cfence: char; iindex: word): string;
var
  i, ileft: integer;
begin
  result := EmptyStr;
  if iindex = 0 then exit;
  ileft := 0;
  for i := 1 to Length(sstring) do
```

```
    if sstring[i] = cfence then begin
      Dec(iindex);
      if iindex = 0 then begin
        result := Copy(sstring, ileft + 1, i - ileft - 1);
        exit;
      end
      else ileft := i;
    end;
    Dec(iindex);
    if iindex = 0 then result := Copy(sstring, ileft + 1, Length(sstring));
end;
```

Замена подстрок

Кто-нибудь знает быстрый алгоритм поиска и замены всех найденных подстрок sub1 на sub2 в строке str?

Решение

```
function ReplaceSub(str, sub1, sub2: String): String;
var
  aPos: Integer;
  rslt: String;
begin
  aPos := Pos(sub1, str);
  rslt := '';
  while (aPos <> 0) do begin
    rslt := rslt + Copy(str, 1, aPos - 1) + sub2;
    Delete(str, 1, aPos + Length(sub1) - 1);
    aPos := Pos(sub1, str);
  end;
  Result := rslt + str;
end;
```

[Щаматис Сергей]

Первая буква каждого слова в строке прописью

Как сделать, чтобы первая буква каждого слова в строке была в верхнем регистре?

Решение

```
function proper(s: string): string;
var
  t: string;
  i: integer;
  newWord: boolean;
begin
  if s = '' then exit;
  s := lowercase(s);
  t := uppercase(s);
  newWord := true;
  for i:=1 to length(s) do begin
    if newWord and (s[i] in ['a'..'z']) then begin
```

```
      s[i] := t[i];
      newWord := false;
      continue;
    end;
    if s[i] in ['a'..'z', ''''] then continue;
    newWord := true;
  end;
  result := s;
end;
```

[News Group]

Примечание

Данный код не работает с русскими символами.

Удаление ненужных подстрок из строки

Решение 1

```
function RemoveInvalid(what, where: string): string;
// what – удаляемая подстрока, where – обрабатываемая строка
var
  tstr: string;
  n: Word;
begin
  tstr := where;
  while Pos(what, tstr) > 0 do begin
    n := Pos(what, tstr);
    tstr := Copy(tstr, 1, n - 1) +
            Copy(tstr, n + length(what), length(tstr) - length(what) - n + 1);
  end;
  Result := tstr;
end;
```

Решение 2

Используйте стандартную функцию Pascal `Delete`. Например:

```
function RemoveSubStr(what, where: string): string;
var
  tstr: string;
  n: Word;
begin
  tstr := where;
  while Pos(what, tstr) > 0 do begin
    n := Pos(what, tstr);
    Delete(tstr, n, Length(what));
  end;
  result := tstr;
end;
```

Паскалевский эквивалент StrTok

Решение 1

```
function NextToken(P: PChar; Divider: PChar): PChar;
const
  next: PChar = nil ;
begin
  if P = nil then P := next;
  if P <> nil then begin
    next := StrPos(P, Divider);
    if next <> nil then begin
      next^ := #0;
      next := @next[StrLen(Divider)];
    end;
  end;
  NextToken := P;
end;
```

[News Group]

Решение 2

```
function StrTok(Phrase: PChar; Delimeter: PChar): PChar;
const
  tokenPtr: PChar = nil;
  workPtr: PChar = nil;
var
  delimPtr: PChar;
begin
  if (Phrase <> nil) then workPtr := Phrase
  else workPtr := tokenPtr;
  if workPtr = nil then begin
    Result := nil;
    Exit;
  end;
  delimPtr := StrPos(workPtr, Delimeter);
  if (delimPtr <> nil) then begin
    delimPtr^ := Chr(0);
    tokenPtr := delimPtr + 1
  end

  else
    tokenPtr := nil;
  Result := workPtr;
end;
```

[News Group]

Корректное сравнение и арифметические действия с DWORD

Существуют ли корректные методы сравнения и выполнения арифметических действий с четырехбайтными беззнаковыми целыми числами (DWORD)?

Ничего лучшего, чем `PChar(a) < PChar(b)`, пока не придумали.

[Nomadic]

2

API

Переменные окружения DOS

Как получить доступ к переменным среды DOS?

Попробуйте компонент, код которого приведен ниже:

```
unit TDosEnv;

interface

uses
  Windows, Messages, Classes, Graphics, Controls, Forms, Dialogs;

type
  TDosEnvironment = class(TComponent)
  private
    FDosEnvList: TStringList;
    procedure DoNothing(Const Value: TStringList);
  protected
    dummy: Word;
    function GetDosEnvCount: Word;
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    function GetDosEnvStr(Const Name: String): String;
  published
    property DosEnvCount: Word read GetDosEnvCount write dummy;
    property DosEnvList: TStringList read FdosEnvList write DoNothing;
end;
```

```
procedure Register;

implementation

uses
  SysUtils;
constructor TDosEnvironment.Create(AOwner: TComponent);
var
  P: PChar;
  i: Integer;
begin
  inherited Create(AOwner);
  FDosEnvList := TStringList.Create;
  P := GetEnvironmentStrings;          // Win API
  i := 0;
  while P^ <> #0 do begin
    Inc(i);
    FDosEnvList.Add(StrPas(P));
    Inc(P, StrLen(P) + 1)              // Быстрый переход к следующей переменной
  end
end;

destructor TDosEnvironment.Destroy;
begin
  FDosEnvList.Free;
  FDosEnvList := nil;
  inherited Destroy
end;

procedure TDosEnvironment.DoNothing(Const Value: TStringList);
begin
  MessageDlg('TDosEnvironment.DosEnvList только для чтения!',
              mtInformation, [mbOk], 0)
end;

{ Возвращает количество переменных окружения. }
function TDosEnvironment.GetDosEnvCount: Word;
begin
  if Assigned(FDosEnvList) then {!!} Result := FDosEnvList.Count
  else Result := 0;
end;

{ Данная функция является измененной версией функции GetEnvVar, присутствующей в
  поставляемом с Delphi модуле WinDos. Она использует паскалевские строки вместо
  строк, заканчивающихся нулем. }
function TDosEnvironment.GetDosEnvStr(Const Name: String): String;
var
  i: Integer;
  Tmp: String;
  Len: Byte absolute Name;
```

```
begin
  i := 0;
  Result := '';
  if Assigned(FDosEnvList) then
{!!}  while i < FDosEnvList.Count do begin
      Tmp := FDosEnvList[i];
      Inc(i);
      if Pos(Name,Tmp) = 1 then begin
        Delete(Tmp, 1, Len);
        if Tmp[1] = '=' then begin
          Delete(Tmp,1,1);
          Result := Tmp;
          i := FDosEnvList.Count;
        end;
      end;
    end;
end;

procedure Register;
begin
  RegisterComponents('Dr.Bob', [TDosEnvironment]);
end;

end.
```

[News Group]

Изменение системного времени

Как сменить системное время Windows из программы, написанной на Delphi?

Примечание

В примере используются функции API GetSystemTime, SetSystemTime. *Эти функции работают со структурой* TSystemTime. *Для примера переведем часы на 3 часа назад и проконтролируем системное время. Обратите внимание – некоторые получаемые значения будут отличаться от ожидаемых данных. Это происходит потому, что функции работают с системными значениями даты и времени Windows. Для получения «правильных» результатов необходимы функции* GetLocalTime, SetLocalTime, *которые и локализуют данные.*

При необходимости можно конвертировать системные дату и время в «родную» структуру TdateTime, *используя функцию* SystemTimeToDateTime *и наоборот –* DateTimeToSystemTime.

Решение

```
procedure TForm1.Button1Click(Sender: TObject);
var
  t: TSystemTime;
```

```
begin
  GetSystemTime(t);
  t.wHour := t.wHour - 3;
  SetSystemTime(t);
  ShowMessage(TimeToStr(time));
end;
```

[Тугаев Олег]

Раскрытие строк с подстановкой вида '%SystemRoot%\IOSUBSYS\'

Как раскрыть строки с подстановками вида '%SystemRoot%\IOSUBSYS\'?

Используйте вызов API:

```
ExpandEnvironmentStrings(LPCTSTR lpSrc, LPTSTR lpDst, DWORD nSize);
```

[Nomadic]

Получение имени модуля

Решение 1

```
procedure TForm1.Button1Click(Sender: TObject);
var
  szFileName: array[0..49] of char;
  szModuleName: array[0..19] of char;
  iSize: integer;
begin
  StrPCopy(szModuleName, 'NameOfModule');
  iSize := GetModuleFileName(GetModuleHandle(szModuleName), szFileName,
                             SizeOf(szFileName));
  if iSize > 0 then ShowMessage('Имя модуля с полным путем: ' + StrPas(szFileName))
  else ShowMessage('Имя модуля не встречено');
end;
```

Решение 2

```
procedure TForm1.Button1Click(Sender: TObject);
var
  AppDirectory, AppPathName: string;
begin
  AppPathName := Application.ExeName;
  AppDirectory := ExtractFilePath(AppPathName);
  MessageDlg('Имя программы ' + AppPathName + #13 + 'Имя каталога программы '
    + AppDirectory, mtInformation, [mbOK], 0);
end;
```

[Степанов Павел]

Решение 3

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  ShowMessage(ParamStr(0));
end;
```

[Иванов Андрей]

Управление монитором

Выключить монитор:

```
SendMessage(HWND_BROADCAST, WM_SYSCOMMAND, SC_MONITORPOWER, 0);
```

Включить монитор:

```
SendMessage(HWND_BROADCAST, WM_SYSCOMMAND, SC_MONITORPOWER, -1);
```

[Sadovnikoff]

Изменение пиктограммы приложения

Как изменить пиктограмму приложения?

Присвойте свойству `Application.Icon` другую пиктограмму и вызовите функцию для немедленной перерисовки.

```
InvalidateRect(Application.Handle, NIL, True);

procedure TForm1.Button1Click(Sender: TObject);
var
  i: TIcon;
begin
  i := TIcon.Create;
  i.LoadFromFile('C:\Program Files\Common Files\Borland shared\Images\Icons\
                  chip.ico');
  Application.Icon := i;
  InvalidateRect(Application.Handle, nil, True);
end;
```

Как получить указатели всех процессов, запущенных в системе

Под Windows (Win32) это возможно с использованием вспомогательных информационных функций:

- Вызывается функция:

  ```
  hSnapshot := CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
  ```

- `Process32First()` – получение информации о первом процессе в списке;
- Далее в цикле `Process32Next()` – получение информации о следующем процессе в списке.

```
unit KernlUtl;

interface
uses
  TlHelp32, Windows, Classes, SysUtils;
procedure GetProcessList(List: TStrings);
procedure GetModuleList(List: TStrings);
function GetProcessHandle(ProcessID: DWORD): THandle;
procedure GetParentProcessInfo(var ID: DWORD; var Path: String);

const
  PROCESS_TERMINATE = $0001;
  PROCESS_CREATE_THREAD = $0002;
  PROCESS_VM_OPERATION = $0008;
  PROCESS_VM_READ = $0010;
  PROCESS_VM_WRITE = $0020;
  PROCESS_DUP_HANDLE = $0040;
  PROCESS_CREATE_PROCESS = $0080;
  PROCESS_SET_QUOTA = $0100;
  PROCESS_SET_INFORMATION = $0200;
  PROCESS_QUERY_INFORMATION = $0400;
  PROCESS_ALL_ACCESS = STANDARD_RIGHTS_REQUIRED or SYNCHRONIZE or $0FFF;

implementation

procedure GetProcessList(List: TStrings);
var
  I: Integer;
  hSnapshoot: THandle;
  pe32: TProcessEntry32;
begin
  List.Clear;
  hSnapshoot := CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
  if (hSnapshoot = -1) then Exit;
  pe32.dwSize := SizeOf(TProcessEntry32);
  if (Process32First(hSnapshoot, pe32)) then repeat
    I := List.Add(Format('%x, %x: %s', [pe32.th32ProcessID,
          pe32.th32ParentProcessID, pe32.szExeFile]));
    List.Objects[I] := Pointer(pe32.th32ProcessID);
  until not Process32Next(hSnapshoot, pe32);
  CloseHandle (hSnapshoot);
end;

procedure GetModuleList(List: TStrings);
var
  I: Integer;
```

```
  hSnapshoot: THandle;
  me32: TModuleEntry32;
begin
  List.Clear;
  hSnapshoot := CreateToolhelp32Snapshot(TH32CS_SNAPMODULE, 0);
  if (hSnapshoot = -1) then Exit;
  me32.dwSize := SizeOf(TModuleEntry32);
  if (Module32First(hSnapshoot, me32)) then repeat
    I := List.Add(me32.szModule);
    List.Objects[I] := Pointer(me32.th32ModuleID);
  until not Module32Next(hSnapshoot, me32);
  CloseHandle (hSnapshoot);
end;

procedure GetParentProcessInfo(var ID: DWORD; var Path: String);
var
  ProcessID: DWORD;
  hSnapshoot: THandle;
  pe32: TProcessEntry32;
begin
  ProcessID := GetCurrentProcessId;
  ID := 0;
  Path := '';
  hSnapshoot := CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
  if (hSnapshoot = -1) then Exit;
  pe32.dwSize := SizeOf(TProcessEntry32);
  if (Process32First(hSnapshoot, pe32)) then
    repeat
      if pe32.th32ProcessID = ProcessID then begin
        ID := pe32.th32ParentProcessID;
        Break;
      end;
    until not Process32Next(hSnapshoot, pe32);
  if ID <> -1 then
    if (Process32First(hSnapshoot, pe32)) then repeat
      if pe32.th32ProcessID = ID then begin
        Path := pe32.szExeFile;
        Break;
      end;
    until not Process32Next(hSnapshoot, pe32);
  CloseHandle (hSnapshoot);
end;

function GetProcessHandle(ProcessID: DWORD): THandle;
begin
  Result := OpenProcess(PROCESS_ALL_ACCESS, True, ProcessID);
end;
end.
```

[Nomadic]

Список запущенных приложений

Решение

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Wnd: hWnd;
  buff: array [0..127] of char;
begin
  ListBox1.Clear; Wnd := GetWindow(Handle, gw_HWndFirst);
  while Wnd <> 0 do begin                    // Не показываем:
    if (Wnd <> Application.Handle)           // Собственное окно
      and IsWindowVisible(Wnd)               // Невидимые окна
      and (GetWindow(Wnd, gw_Owner) = 0)     // Дочерние окна
      and (GetWindowText(Wnd, buff, sizeof(buff)) <> 0) then begin
        GetWindowText(Wnd, buff, sizeof(buff)); ListBox1.Items.Add(StrPas(buff));
    end;
    Wnd := GetWindow(Wnd, gw_hWndNext);
  end;
  ListBox1.ItemIndex := 0;
end;
```

[Nikolaev Igor]

Как запустить другую программу

Воспользуйтесь функцией `ExecuteFile()` из модуля `FMXUTILS.PAS`.

Пример вызова:

```
ExecuteFile('Notepad.exe', '', 'c:\windows', SW_SHOWNORMAL);
```

Примечание

Чтобы этот код заработал, необходимо добавить в `uses` *модуль* `ShellAPI`.

Работа с другим приложением без Hook и DLL на примере GetFocus

На стандартной форме (`Form1`) размещаем объекты:

- Таймер (`Timer1`) с периодом (`Interval`) 1000 мс или меньше;
- Заметки (`Label1`, `Label2`, `Label3`);
- Назначаем обработчик события таймера `OnTimer`;
- Назначаем `Form1.FormStyle = fsStayOnTop` поверх остальных окон.

```
procedure TForm1.Timer1Timer(Sender: TObject);
var
  dwTargetOwner: DWORD;          // указатель на подключаемый процесс
  dwThreadID: DWORD;             // указатель на текущий процесс
  Result: LongBool;
```

```
begin
{ В первой метке отображается Handle активного окна }
// Указатель на подключаемое приложение
  Label1.Caption := IntToStr(GetForegroundWindow);
// Подключение потока другого окна
  dwTargetOwner := GetWindowThreadProcessId(GetForegroundWindow, nil);
  dwThreadID := GetCurrentThreadId();           // указатель на текущий процесс
  if (dwTargetOwner <> dwThreadID) then         // если не один и тот же процесс
    Result := AttachThreadInput(dwThreadID, dwTargetOwner, True);  // подключение
{ Во второй метке отображается Handle объекта 'в фокусе' в активном окне }
  Label2.Caption := IntToStr(GetFocus);         // фокус в другом приложении
  if (Result) then
    AttachThreadInput(dwThreadID, dwTargetOwner, False);  // отключение
{ В третей метке отображается Handle объекта 'в фокусе' в активном окне,
  но если это окно другого приложения, то Handle будет равен нулю,
  т.к. попытка получения Handle происходит после отключения потока }
  Label3.Caption := IntToStr(GetFocus);         // проверка после отключения
{ Работу можно наблюдать, если запустить полученное приложение и сделать активным
  другое приложение. Ясно, что полученный Handle объекта можно использовать по
  своему разумению. Например, считать из объекта текст и т. п. }
end;
```

[SottNick]

Открытие файла из работающего приложения

При программировании MDI-приложений возникает следующая задача: если пользователь щелкнул по пиктограмме файла, тип которого поддерживается создаваемым приложением, то при запущенном приложении не нужно запускать его новую копию, а необходимо лишь открыть выбранный файл в уже работающем приложении.

В файле проекта:

```
var
  i: integer;
  hMainForm: HWND;
  CopyDataStruct: TCopyDataStruct;
  ParamString: string;
  WParam, LParam: integer;
begin
// ищем главное окно приложения, вместо Caption - nil,
// поскольку к заголовку главного окна может добавиться заголовок
// MDIChild (нужно позаботиться об уникальности имени класса главной формы)
  hMainForm := FindWindow('TMainForm', nil);
  if hMainForm = 0 then begin
    Application.Initialize;
    Application.CreateForm(TMainForm, frmMain);
    for i := 1 to ParamCount do
      TMainForm(Application.MainForm).OpenFile(ParamStr(i));
    Application.Run;
  end else begin
```

```
    ParamString := '';
// пересылаем все параметры в одну строку с разделителем #13
    for i := 1 to ParamCount do
      ParamString := ParamString + ParamStr(i) + #13;
// создаем запись типа TCopyDataStruct
    CopyDataStruct.lpData := PChar(ParamString);
    CopyDataStruct.cbData := Length(ParamString);
    CopyDataStruct.dwData := 0;
    Wparam := Application.Handle;
    Lparam := Integer(@CopyDataStruct);
// отсылаем сообщение WM_COPYDATA главному окну открытого приложения
    SendMessage(hMainForm, WM_CopyData, WParam, LParam);
    Application.Terminate;
  end;
end;

// Обработчик сообщения WM_COPYDATA
procedure TMainForm.CopyData(var Msg: TWMCopyData);
var
  ParamStr: string;
  CopyDataStructure: TCopyDataStruct;
  i: integer;
  len: integer;
begin
  CopyDataStructure := Msg.CopyDataStruct^;
  ParamStr := '';
  Len := CopyDataStructure.cbData;
  for i := 0 to len - 1 do
    ParamStr := ParamStr + (PChar(CopyDataStructure.lpData) + i)^;
  i := 0;
  while not(Length(ParamStr) = 0) do begin
    if isDelimiter(#13, ParamStr, i) then begin
      OpenFile(Copy(ParamStr, 0, i - 1));
      ParamStr := Copy(ParamStr, i + 1, Length(ParamStr) - i - 1);
    end;
    inc(i);
  end;
  inherited;
end;
```

[Пангин Дмитрий]

Примечание

Для работоспособности данного примера необходимо объявить переменную `frmMain` *и создать процедуру* `OpenFile`*, которая и будет загружать необходимые файлы.*

Обработка WM_SysCommand

Системное меню в приложениях Delphi ведет двойную жизнь. Когда основная форма активна, работает системное меню главной формы, а если приложение минимизировано – работает системное меню объекта ATpplictaion. В данном случае можно воспользоваться следующим программным кодом:

```
  ...
const
  SC_UDF = $EFF0;    { должен быть < $F000 и делиться на 16 }

procedure TForm1.FormCreate(Sender: TObject);
begin
  AppendMenu(GetSystemMenu(Handle, False), MF_STRING, SC_UDF, 'Всегда наверху');
  AppendMenu(GetSystemMenu(Application.Handle, False), MF_STRING,
          SC_UDF, 'Всегда наверху');
  Application.OnMessage := AppOnMessage;
end;

procedure TForm1.AppOnMessage(VAR Msg: TMsg; VAR Handled: Boolean);
begin
  if Msg.Message <> WM_SYSCOMMAND then Exit;
  if Msg.wParam AND $FFF0 <> SC_UDF then Exit;
//... здесь вы можете включить код для обработки системного сообщения ...
end;
```

[News Group]

Проблема синтаксиса DrawCaption

Электронная справка Microsoft не соответствует их же определениям в Windows.h функции DrawCaption. Программисты Borland использовали определение из Windows.h. Вот эти четыре параметра:

- Дескриптор окна (The window handle of the window);
- Контекст устройства для окна (A device context for the window);
- Прямоугольник (TRect);
- Целочисленное поле, содержащее флаги, указанные в электронной справке.

[News Group]

FlashWindow для пиктограмм

Приложения Delphi имеют невидимую форму уровня приложения, представляемую объектом TApplication. Когда главная форма имеет нормальное или максимальное состояние, функция FlashWindow работает правильно. Когда приложение минимизировано, пиктограмма реально являет-

ся невидимой формой приложения. Следовательно, FlashWindow нужно передавать Application.Icon.

Извлечение пиктограммы из файлов EXE и DLL

Каким образом извлечь пиктограмму из EXE- или DLL-файлов и отобразить ее на компоненте TImage *или небольшой области на форме?*

Решение 1

```
uses
  ShellAPI;

procedure TForm1.IconExtractor(FileName: PChar);
var
  MyIcon: TIcon;
begin
  MyIcon := TIcon.Create;
  try
    MyIcon.Handle := ExtractIcon(hInstance, FileName, 0);
    { Здесь можно что-нибудь сделать с пиктограммой, например: }
    Image1.Picture.Icon := MyIcon;
  finally
    MyIcon.Free;
  end;
end;
```

Решение 2

```
uses ShellAPI;

procedure TForm1.Button1Click(Sender: TObject);
var
  IconIndex: word;
  h: hIcon;
begin
  IconIndex := 0;
  h := ExtractAssociatedIcon(hInstance, 'C:\Windows\NOTEPAD.EXE', IconIndex);
  DrawIcon(Form1.Canvas.Handle, 10, 10, h);
end;
```

Как предотвратить запуск копии приложения

Решение 1

Предлагаемый способ окажется полезным для тех, у кого нет острой необходимости активировать существующую копию программы, а достаточно сообщить пользователю, что она уже запущена.

```
...
uses
  syncobjs;
```

```
...
var
  CheckEvent: TEvent;
...
procedure TForm1.FormCreate(Sender: TObject);
begin
  CheckEvent := TEvent.Create(nil, false, true, 'MYPROGRAM_CHECKEXIST');
  if CheckEvent.WaitFor(10) <> wrSignaled then begin
    // Сюда попадаем, если одна копия уже запущена.
    // Можно, например, сообщить об этом пользователю:
    ShowMessage('Эта программа уже запущена!');
    // Здесь можно завершить программу или сделать еще что-нибудь...
    Self.Close;
  end;
end;
```

[Волосенков Владимир]

Решение 2

Второе решение приведено в книге Д. Тейлора, Дж. Мишеля и др. «Delphi 3: библиотека программиста».[1]

```
program Project1;

uses
  Windows,
  Forms,
  Unit1 in 'Unit1.pas' {Form1};
const
  MemFileSize = 127;
  MemFileName = 'one_example';

var
  MemHnd: HWND;
{$R *.RES}

begin
  MemHnd := CreateFileMapping(HWND($FFFFFFFF), nil, PAGE_READWRITE, 0,
                              MemFileSize, MemFileName);
  if GetLastError <> ERROR_ALREADY_EXISTS then begin
    Application.Initialize;
    with TForm1.Create(nil) do
      try
        Show;
        Update;
        Application.CreateForm(TForm1, Form1);
      finally
        Free;
      end;
```

[1] Издательство «Питер», 1998 г.

```
    Application.Run;
  end else
    Application.MessageBox('Приложение уже запущено', 'Ошибка', MB_OK);
  CloseHandle(MemHnd);
end.
```

[Васильев Николай]

Решение 3

Избежать запуска второй копии программы можно, если в DPR-файле использовать функцию из библиотеки RxLib:

```
ActivatePrevInstance('TForm1', 'Значение Caption') ;
```

[Тарасов Николай]

Решение 4

Предлагаемый модуль только определяет, запущена программа или нет. Он не усложнялся автоматическим изменением имени семафора на случай, если две программы «захотят» использовать этот модуль одновременно. Целесообразно придумать для семафора собственное уникальное имя и переписать его в previnst.pas.

Для реализации предлагаемого решения в модуле program в части uses необходимо добавить previnst, и вы получите переменную ммм: Boolean, которая равна значению True, если копия программы уже запущена.

```
program Project1;

uses
  previnst, Windows, Forms,
  Unit1 in 'Unit1.pas' {Form1};
{$R *.RES}
begin
  if mmm then begin
    ShowWindow(FindWindow('TForm1', 'Имя окна, которое надо активизировать'),
               SW_restore);
    SetForegroundWindow(FindWindow('TForm1', 'Имя окна, которое
                                    надо активизировать'));
    Halt;            // завершить программу, не создавая ничего.
  end;
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

содержание модуля previnst.pas

```
unit Previnst;

interface
```

```
uses
  Windows;
var
  mmm: boolean;                // если true, то программа уже запущена

implementation

var
  hMutex: integer;
begin
  mmm := false;
  hMutex := CreateMutex(nil, TRUE, 'AbraShvabra');  // Создаем семафор
  if GetLastError <> 0 then mmm := true;            // Ошибка семафор уже создан
  ReleaseMutex(hMutex);
end.
```

[News Group]

Решение 5

Можно использовать переменную Atom, полная информация о которой приведена в справочном руководстве Delphi.

```
program Project1;

uses
  Windows, Forms,
  Unit1 in 'Unit1.pas' {Form1};

{$R *.RES}
const
  AtStr = 'MyProgram';

function CheckThis: boolean;
var
  Atom: THandle;
begin
  Atom := GlobalFindAtom(AtStr);
  Result := Atom <> 0;
  if not Result then GlobalAddAtom(AtStr);
end;

begin
  if not CheckThis then begin    // Запуск программмы
    Application.Initialize;
    Application.CreateForm(TForm1, Form1);
    Application.Run;
    GlobalDeleteAtom(GlobalFindAtom(AtStr));
  end else MessageBox(0, 'Нельзя запустить две копии программы', 'Error', 0);
end.
```

[Шубин В.]

Приоритет приложения

Как изменить приоритет приложения?

Решение

```
procedure TForm1.Button1Click(Sender: TObject);
var
  ProcessID: dword;
  ProcessHandle: THandle;
  ThreadHandle: THandle;
begin
  ProcessID := GetCurrentProcessID;
  ProcessHandle := OpenProcess(PROCESS_SET_INFORMATION, false, ProcessID);
  SetPriorityClass(ProcessHandle, REALTIME_PRIORITY_CLASS);
  ThreadHandle := GetCurrentThread;
  SetThreadPriority(ThreadHandle, THREAD_PRIORITY_TIME_CRITICAL);
end;
```

Улучшение работы LockWindowUpdate

Чтобы избежать излишнего мерцания экрана, можно воспользоваться таким эффективным способом, как посылка сообщения WM_SETREDRAW. Это позволит блокировать/разблокировать форму, не затрагивая при этом остальные окна.

Чтобы временно запретить перерисовку формы, необходим следующий код:

```
Perform(WM_SETREDRAW, 0, 0);
```

чтобы возвратиться к нормальному состоянию:

```
Perform(WM_SETREDRAW, 1, 0);
Refresh;
```

[News Group]

Использование WSAAsyncSelect при отсутствии формы

Что необходимо передать `WSAAsyncSelect` *в качестве параметра* `Handle`*, если тот запускается и используется в DLL, но при этом никакой формы в DLL не создается?*

Если WSAAsyncSelect применяется в DLL и нет формы, то рекомендуется следующее решение:

```
const
  WM_ASYNCSELECT = WM_USER + 0;

type
  TNetConnectionsManager = class(TObject)
  protected
    FWndHandle: HWND;
```

```
    procedure WndProc(var MsgRec: TMessage);
    ...
  end;

constructor TNetConnectionsManager.Create;
begin
  inherited Create;
  FWndHandle := AllocateHWnd(WndProc);
  ...
end;

destructor TNetConnectionsManager.Destroy;
begin
  ...
  if FwndHandle <> 0 then DeallocateHWnd(FWndHandle);
  inherited Destroy;
end;

procedure TNetConnectionsManeger.WndProc(var MsgRec: TMessage);
begin
  with MsgRec do
    if Msg = WM_ASYNCSELECT then WMAsyncSelect(MsgRec)
    else DefWindowProc(FWndHandle, Msg, wParam, lParam);
end;
```

Но рекомендуется посмотреть WinSock2

```
WSAEventSelect(FSocket, FEventHandle, FD_READ or FD_CLOSE);
WSAWaitForMultipleEvents(...);
WSAEnumNetworkEvents(FSocket, FEventHandle, lpNetWorkEvents);
```

То есть обойтись без окон и без очереди сообщений Windows, а заодно иметь возможность работать и с IPX/SPX, и с NetBios.

[Nomadic]

Контроль завершения приложения

Решение

```
function WinExecAndWait32(FileName: String; Visibility: integer): DWORD;
var
  zAppName: array[0..512] of char;
  zCurDir: array[0..255] of char;
  WorkDir: String;
  StartupInfo: TStartupInfo;
  ProcessInfo: TProcessInformation;
  Pd: pointer;
begin
  StrPCopy(zAppName, FileName);
  GetDir(0, WorkDir);
```

```
    StrPCopy(zCurDir, WorkDir);
    FillChar(StartupInfo, Sizeof(StartupInfo), #0);
    StartupInfo.cb := Sizeof(StartupInfo);
    StartupInfo.dwFlags := STARTF_USESHOWWINDOW;
    StartupInfo.wShowWindow := Visibility;
    if not CreateProcess(nil, zAppName, nil, nil, false, CREATE_NEW_CONSOLE
      or NORMAL_PRIORITY_CLASS, nil, nil, StartupInfo, ProcessInfo) then Result := 0
    else begin
      WaitforSingleObject(ProcessInfo.hProcess, INFINITE);
      GetExitCodeProcess(ProcessInfo.hProcess, Result);
    end;
  end;
```

В качестве дополнения внесем резонное исправление – вместо:

```
WaitforSingleObject(ProcessInfo.hProcess, INFINITE);
```

лучше написать:

```
while WaitforSingleObject(ProcessInfo.hProcess, 200) = WAIT_TIMEOUT do
  TForm1.Repaint;
```

Смысл замены: в первом варианте главное окно ждет завершения вызванного сообщения, не обрабатывая при этом никаких событий. Вследствие этого, главное окно не перерисовывается, что выглядит далеко не лучшим образом. Последний вариант исправляет этот недостаток.

[Trubachev Pavel]

Определение завершения работы Windows

Существует ли возможность завершения Windows для нормального закрытия работающего приложения Delphi?

Самым простым решением является создание обработчика события главной формы `OnCloseQuery`. Данное событие возникает как результат сообщения `WM_QUERYENDSESSION`, которое посылается всем работающим приложениям Windows в момент инициализации процесса окончания работы Windows. Логическая переменная `CanClose`, передаваемая обработчику как `var`-параметр, может позволить программе (и Windows) завершить свою работу, если параметру присвоено значение `True` (присвоение значения `False` не позволит программе завершить свою работу).

Следующий код демонстрирует, как можно воспользоваться описанным обработчиком события.

```
procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
  { Спрашиваем пользователя, если инициировано завершение работы. }
  if MessageDlg('Вы уверены?', mtConfirmation, mbYesNoCancel, 0) = mrYes then
    CanClose := true     { Разрешаем завершение работы. }
```

```
    else
      CanClose := false;    { Не разрешаем завершение работы. }
  end;
```

Перехват выгрузки операционной системы

Что делать, если в моей программе отсутствует диалоговое окно, а мне нужно «перехватить» сообщение о выгрузке операционной системы?

Воспользуйтесь функцией GetMessage(), в качестве HWND окна передайте Null (на Паскале – 0). Если в очереди сообщений следующее – WM_QUIT, то эта функция возвращает значение False. Создавая программу для Win32, используйте отдельный поток, организующий выход из программы.

[Nomadic]

Завершение работы Windows

Каким образом завершить работу операционной системы (функция ExitWindows*) из кода самой программы? Нужно ли для этого перезапускать операционную систему без перезагрузки компьютера.*

Решение 1

Рассмотрим две функции для перезапуска операционной системы:

```
procedure TForm1.RebootWindowsBtnClick(Sender: TObject);
begin
  if not ExitWindows(EW_RebootSystem, 0) then
    ShowMessage('Приложение не может завершить работу');
end;

procedure TForm1.RestartWindowsBtnClick(Sender: TObject);
begin
  if not ExitWindows(EW_RestartWindows, 0) then
    ShowMessage('Приложение не может завершить работу');
end;
```

Функция ExitWindows не была правильно документирована Microsoft, и поэтому не содержит описания возвращаемого значения. Более того, информация об этой функции практически отсутствует в других источниках. Вот правильное определение этой функции:

```
function ExitWindows (dwReturnCode: Longint; Reserved: Word): Bool;
```

Решение 2

Рассмотрим некоторые уточнения, которые касаются вопроса о завершении или перезагрузке Windows. Указанный код не даст результата для Windows NT (и, возможно, даже для Windows 2000). Ниже приводится корректно работающий код для Windows 9x и Windows NT.

```
procedure RebootSystem;
var
  handle, ph: THandle;
  pid: DWORD;
  luid: TLargeInteger;
  dummy, priv: TOKEN_PRIVILEGES;
  ver: TOSVERSIONINFO;
begin
  ver.dwOSVersionInfoSize := Sizeof(ver); GetVersionEx(ver);
  if ver.dwPlatformId = VER_PLATFORM_WIN32_NT then begin
    pid := GetCurrentProcessId;
    ph := OpenProcess(PROCESS_ALL_ACCESS, false, pid);
    if OpenProcessToken(ph, TOKEN_ADJUST_PRIVILEGES, handle) then
      if LookupPrivilegeValue(nil, 'SeShutdownPrivilege', luid) then begin
        priv.PrivilegeCount := 1;
        priv.Privileges[0].Attributes := SE_PRIVILEGE_ENABLED;
        priv.Privileges[0].Luid := luid;
        AdjustTokenPrivileges(handle, false, priv, 0, dummy, pid);
      end;
  end;
  ExitWindowsEx(EWX_REBOOT, 0);
end;
```

[Слабко Дмитрий]

Решение 3

К сожалению, 32-битная функция ExitWindowsEx не позволяет выполнить простой перезапуск Windows 9x. Тем не менее, есть 16-битная функция ExitWindows, которая позволяет выполнить перезапуск, будучи вызванной с параметром EW_RESTARTWindows.

Даже если вы работаете с правами Администратора, программа должна запросить дополнительные привилегии.

```
procedure Shutdown(Name: String;  // Имя машины (\\SERVER)
      Message: String;            // Сообщение
      Delay: Integer;             // Задержка перед рестартом
      Restart, CloseAll: Boolean);
var
  ph: THandle;
  tp, prevst: TTokenPrivileges;
  rl: DWORD;
begin
  OpenProcessToken(GetCurrentProcess, TOKEN_ADJUST_PRIVILEGES or TOKEN_QUERY, ph);
  LookupPrivilegeValue(Nil, 'SeShutdownPrivilege', tp.Privileges[0].Luid);
  tp.PrivilegeCount := 1; tp.Privileges[0].Attributes := 2;
  AdjustTokenPrivileges(ph, FALSE, tp, SizeOf(prevst), prevst, rl);
  InitiateSystemShutdown(PChar(name), PChar(Message), Delay, Restart, CloseAll);
  ShowMessage(SysErrorMessage(GetLastError));         // Результат
end;
```

[Nomadic]

Управление завершением работы Windows

Как в одном компоненте реализовать выключение компьютера, его перезагрузку, завершение сеанса работы пользователя, функцию выгрузки CD, выключение питания монитора и т. д.?

Предлагаем рассмотреть следующий пример:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  PowerControl1.Action := actCDEject;   // Или... actLogOFF, actShutDown...
  PowerControl1.Execute;
end;

unit PowerControl;

interface

uses
  Windows, Messages, SysUtils, Classes, Controls, Forms, Graphics, MMSystem;

type
  TAction = (actLogOFF, actShutDown, actReBoot, actForce,
             actPowerOFF, actForceIfHung, actMonitorOFF,
             actMonitorON, actCDEject, actCDUnEject);
  TPowerControl = class(TComponent)
  private
    FAction: TAction;
    Procedure SetAction(Value: TAction);
  public
    function Execute: Boolean;
  published
    property Action: TAction read FAction write SetAction;
end;

procedure Register;
implementation

procedure Register;
begin
  RegisterComponents('K2', [TPowerControl]);
end;

procedure TPowerControl.SetAction(Value: TAction);
begin
  FAction := Value;
end;

function TPowerControl.Execute: Boolean;
begin
  with (Owner as TForm) do
```

```
    case FAction of
       actLogOff:  ExitWindowsEx(EWX_LOGOFF,1);
     actShutDown:  ExitWindowsEx(EWX_SHUTDOWN,1);
       actReBoot:  ExitWindowsEx(EWX_REBOOT,1);
        actForce:  ExitWindowsEx(EWX_FORCE,1);
     actPowerOff:  ExitWindowsEx(EWX_POWEROFF,1);
  actForceIfHung:  ExitWindowsEx(EWX_FORCEIFHUNG,1);
   actMonitorOFF:  SendMessage(Application.Handle, WM_SYSCOMMAND,
                               SC_MONITORPOWER, 0);
    actMonitorON:  SendMessage(Application.Handle, WM_SYSCOMMAND,
                               SC_MONITORPOWER, -1);
      actCDEject:  mciSendstring('SET CDAUDIO DOOR OPEN WAIT', nil, 0, Handle);
    actCDUnEject:  mciSendstring('SET CDAUDIO DOOR CLOSED WAIT', nil, 0, Handle);
    end; {Case}
  Result := True;
end;

end.
```

[Nikolaev Igor]

Как консольное приложение может узнать, что Windows завершает работу?

Все процессы получают сигналы CTRL_CLOSE_EVENT, CTRL_LOGOFF_EVENT и CTRL_SHUTDOWN_EVENT. Делается это так:

```
function Ctrl_Handler(Ctrl: Longint): LongBool;
begin
  if Ctrl in [CTRL_SHUTDOWN_EVENT, CTRL_LOGOFF_EVENT] then begin
    // Bay, вау
  end else begin
    // Am I creator?
  end;
  Result := true;
end;
```

В программе:

```
SetConsoleCtrlHandler(Ctrl_Handler, TRUE);
```

[Nomadic]

Создание консольных приложений

В консольных приложениях в Delphi можно использовать, в принципе, весь дельфийский арсенал. Правда, и работать они будут лишь под Windows. (Кстати, этот способ можно применить для модернизации программ на Паскале под Windows. Этот код был использован для вывода результатов работы программы проверки (неважно чего), чтобы не просматривать файл с результатами. Главная проблема была в том, что консоль (если запуск осущест-

влялся из Windows) оставалась висеть позади формы до ее закрытия. Вреда конечно, никакого, но неприятно. Если же запуск выполняется из Нортона или чего-то подобного, то все идет нормально.

```
Program MyProgram;
{$APPTYPE CONSOLE}

uses
  Windows, Forms, Dialogs, SysUtils, StdCtrls, Controls; // и (или) т. п.

...

var
  ...
  SH,SW: integer;
  MainForm: TForm; // если нужна форма
  Memo: TMemo;
  // могут быть также любые другие визуальные компоненты
  ...
// здесь могут быть процедуры и функции, т. е. все, как в обычном Паскале

Begin

  ... // здесь какой-то код

{ а здесь, перед выводом формы, есть два пути:}
{ так}
  FreeConsole; // Отцепиться от консоли, т. е. она просто исчезнет
               // (в случае запуска из Windows) и останется только форма
{ или так}
//  Handle:= GetForegroundWindow; // Получить Handle консоли
//  ShowWindow(Handle, SW_HIDE);  // Спрятать консоль
// а в конце, перед завершением
//  ShowWindow(Handle, SW_SHOW); // Показать консоль

  { для помещения формы в центр экрана}
  SH:= Screen.Height;
  SW:= Screen.Width;
  MainForm:= TForm.Create(nil);
  with MainForm do
  try
    BorderStyle:= bsSizeable;
    Height:= 390;
    Width:= 390;
    Left:= (SW - Width) div 2;
    Top:= (SH - Height) div 2;
    Caption:= 'Моя программа';
      // здесь могут быть и другие компоненты
      Memo:= TMemo.Create(MainForm);
        with Memo do begin
          Parent:= MainForm;
```

```
          Align:= alClient;
          BorderStyle:= bsNone;
          Font.Name:= 'Courier New Cyr';
          Font.Size:= 9;
          ScrollBars:= ssVertical;
          Lines.LoadFromFile('MyProgram.txt');
        end;
    ShowModal;
  finally
    Free;
  end;

{ или можно вывести сообщение, например в случае неудачи (или наоборот)}
  with CreateMessageDialog('Текст сообщения',mtInformation,[mbOk]) do
  try
    Caption := 'Заголовок';
    ShowModal;
  finally
    Free;
  end;

...

//  это для второго пути, иначе она так и останется висеть свернутой
//  ShowWindow(Handle, SW_SHOW); // Показать консоль

End.
```

[Чумак Михаил]

3

Pascal (интегрированная среда)

Описание типов файлов для Delphi

Формат CAB

Это формат файлов, который Delphi предлагает теперь своим пользователям для размещения в Интернете. Формат Cabinet, являясь эффективным средством для упаковки нескольких файлов, имеет две основные характеристики: в отдельном «кабинете» (.CAB-файл) могут храниться несколько файлов, и сжатие данных выполняется в зависимости от типа файлов, что значительно увеличивает коэффициент сжатия. Создание Cabinet-файла зависит также от количества упаковываемых файлов и предполагаемого типа доступа к ним (последовательный, произвольный, одновременный ко всем файлам или доступ к нескольким файлам в одно и тоже время). Delphi не пользуется преимуществами сжатия файлов в зависимости от их типа.

Формат LIC

В действительности как такового формата .LIC-файла не существует. Обычно это такие же текстовые файлы, содержащие одну или две ключевые строки.

Формат INF

Все файлы в формате INF состоят из секций и пунктов. Каждая именованная секция содержит соответствующие пункты. Все .INF-файлы начинаются с заголовочной секции. После заголовка включенные секции могут располагаться в любом порядке. Каждый заголовок представляет собой строку с именем заголовка, заключенным в квадратные скобки. Далее следуют пункты: `ItemA = ItemDetail`. Для получения дополнительной информации обратитесь к документу «Device Information File Reference».

Формат DPR

Файл в формате DPR является центральным файлом проекта Delphi. Для программы он является опорным (в том смысле, что он создается при первом сохранении любого приложения и его удаление приводит к краху всего проекта). Файл .DPR содержит ссылки на другие файлы проекта и связывает формы с соответствующими модулями. Данный файл нужно редактировать с предельной осторожностью, т. к. неумелые действия могут привести к тому, что загрузить проект будет невозможно. Наличие этого файла является критическим при загрузке и перемещении (копировании) проекта.

Формат PAS

Это стандартный текстовый файл, который можно редактировать в текстовом редакторе. Однако это требует некоторой осторожности, поскольку редактирование может закончиться частичной потерей преимуществ двух других инструментов. Например, добавление кода для кнопки с декларацией типа никак не отразится на соответствующем .DFM-файле формы. Все .pas-файлы являются критическими при пересборке проекта.

Формат DFM

Данный файл содержит описание объектов, расположенных на форме. Содержимое файла можно отобразить в виде текста, вызвав правой кнопкой мыши контекстное меню и выбрав пункт View as text, или же с помощью конвертора CONVERT.EXE (расположенного в каталоге \BIN), также позволяющего перевести файл в текстовый вид и обратно. Данный файл следует редактировать очень внимательно, поскольку это может закончиться тем, что IDE не сможет загрузить форму. Наличие этого файла, как и файла DPR, становится опасным при перемещении и пересборке проекта.

Формат DOF

Данный текстовый файл содержит текущие установки для опций проекта, например, настройки компилятора и компоновщика, каталоги, условные директивы и параметры командной строки. Данные установки могут быть изменены пользователем путем изменений настроек проекта.

Формат DSK

Этот текстовый файл хранит информацию о состоянии проекта, например, об открытом окне и его координатах. Подобно .DOF-файлу, файл DSK создается на основе текущей обстановки в проекте.

Формат DPK

Файл в формате DPK содержит исходный код пакета (аналогично .DPR-файлу стандартного проекта Delphi). Подобно файлу .DPR, такой файл также представляет собой простой текстовый файл, который можно редактировать (см. предупреждение выше) в стандартном редакторе. Одна из причин, по которой вы можете это сделать, – необходимость вызова компилятора командной строки.

Формат DCP

Данный бинарный Image-файл фактически состоит из скомпилированного пакета. Информация о символах и дополнительных заголовках, требуемых IDE, содержится в .DCP-файле в полном объеме. Чтобы собрать (build) проект, IDE должен иметь доступ к этому файлу.

Формат DPL

Файл с расширением .DPL в действительности представляет собой выполняемый runtime-пакет. Такой файл является динамической библиотекой Windows со специфическими интегрированными характеристиками Delphi. Данный файл необходим в случае активизации приложения, использующего пакеты.

Формат DCI

Данный формат содержит как стандартные, так и определенные пользователем шаблоны кода, действующие в IDE. Файл в формате DCI может редактироваться стандартным текстовым редактором, или в самой IDE. Как и любой текстовый файл данных, используемый Delphi, редактировать его самостоятельно не рекомендуется.

Формат DCT

Это «частный» бинарный файл, содержащий информацию об определенных пользователем шаблонах компонентов. Такой файл не может быть отредактирован никакими способами через IDE. Поскольку данный файл является «личным» файлом IDE, то совместимость с последующими версиями Delphi не гарантируется.

Формат TLB

Файл .TLB является «частным» двоичным файлом библиотеки типов. Обеспечивает информацию для идентификации типов объектов и интерфейсов, доступных в сервере ActiveX. Подобно модулю или заголовочному файлу, .TLB служит в качестве хранилища необходимой символьной информации приложения. Поскольку данный файл является «личным», то совместимость с последующими версиями Delphi также не гарантируется.

Формат DRO

Данный текстовый файл содержит информацию об объектном хранилище. Каждый пункт данного файла содержит специфическую информацию о каждом доступном элементе в хранилище объектов. Хотя этот файл и является простым текстовым файлом, все же не рекомендуется править его вручную. Хранилище может редактироваться только с помощью меню Tools|Repository в самой IDE.

Формат RES

Это стандартный двоичный файл ресурсов Windows-формата, включающий в себя информацию о приложении. По умолчанию Delphi создает новый .RES-файл при каждой компиляции проекта в исполняемое приложение.

Формат DB

Файлы с таким расширением являются стандартными файлами СУБД Paradox.

Формат DBF

Расширение .DBF свойственно стандартным dBASE-файлам.

Формат GDB

Файлы с таким расширением относятся к стандартным файлам Interbase.

Формат DMT

Этот «частный» бинарный файл содержит встроенные и определенные пользователем шаблоны меню. Данный файл не может быть отредактирован никакими способами через IDE. Поскольку данный файл является «личным», то совместимость с последующими версиями Delphi не гарантируется.

Формат DBI

Данный текстовый файл содержит информацию, необходимую для инициализации Database Explorer. Такой файл не может быть отредактирован никакими способами через Database Explorer.

Формат DEM

Данный текстовый файл содержит некоторые стандартные, характерные для страны локализации, форматы компонента `TMaskEdit`. Как и любой текстовый файл данных, используемый Delphi, редактировать его самостоятельно не рекомендуется.

Формат OCX

Файл с расширением .OCX представляет собой специализированную DLL, содержащую все или несколько функций, связанные с элементом управления ActiveX. Файл OCX задумывался как «обертка», которая содержала бы сам объект и средства для связи с другими объектами и серверами.

Директивы компилятора, способные увеличить скорость

Скорость выполнения программы может упасть из-за применения динамических массивов. Поэтому целесообразно обратить внимание на ключи компилятора. После отладки кода установите эти три наиболее важных ключа:

```
{$R-} {Range checking off - проверка диапазона}
{$S-} {Stack checking off - проверка стека}
{$A+} {Word align data - "выравнивание слов"}
```

Сохранение пользовательских настроек

Я хотел бы записывать новые пользовательские цвета на место старых в файл ресурса, чтобы при следующем открытии формы она автоматически отображала новую цветовую схему.

Это нетрудно сделать, передавая форму при закрытии в поток для записи ее в отдельный файл. Затем, когда это будет выполнено, проверяйте наличие файла и организуйте его чтение. Данные действия реализуются с помощью следующего программного кода:

```
const
  FileName = 'Form1.stm';
constructor TForm1.Create(AOwner: TComponent);
begin
  if FileExists(FileName) then begin
    CreateNew(AOwner);
    ReadComponentResFile(FileName, Self);
  end else
    inherited Create(AOwner);
{ поместите здесь код в стиле 'OnCreate' }
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  WriteComponentResFile(FileName, Self);
end;
```

Опубликованное свойство в Инспекторе объектов

Как убрать опубликованное свойство из списка свойств в Инспекторе объектов?

Решение

```
interface

type
  TMyComp = class(TWinControl)
    ...
  end;

procedure Register;
implementation
  ...

procedure Register;
begin
  RegisterComponents('MyPage', [TMyComp]);
  RegisterPropertyEditor(TypeInfo(String), TMyComp, 'Hint', nil);
end;
```

[Nomadic]

Создание редактора свойств

Если вы присвоили свойству имя TableName, то полный цикл создания редактора свойств включает следующие шаги:

- Опишите класс редактора свойств:

```
type
  TTableNameProperty = class(TStringProperty)
    function GetAttributes: TPropertyAttributes; override;
    procedure GetValues(Proc: TGetStrProc); override;
  end;

implementation

{ TTableNameProperty }

function TTableNameProperty.GetAttributes: TPropertyAttributes;
begin
  Result := [paValueList];
end;

procedure TTableNameProperty.GetValues(Proc: TGetStrProc);
var
  TableName: String;
  I: Integer;
begin
{ здесь вы должны добавить свой код, чтобы с помощью цикла обойти имена всех
  таблиц, включенных в список }
  for I := 0 to ???? do begin
    TableName := ????[I];
    Proc(TableName);
  end;

end;
```

- Затем зарегистрируйте данный редактор свойств следующим образом:

```
RegisterPropertyEditor(TypeInfo(string), TcsNotebook, 'TableName',
                       TTableNameProperty);
```

[News Group]

Особенности вызова редактора свойств

Я пишу редактор для свойства TStrings. *В зависимости от значений других свойств хотел бы показывать или свой редактор свойства, или редактор* TStringListProperty, *заданный по умолчанию, но не знаю, как передавать управление* TStringListProperty*?*

Для решения поставленного вопроса необходимо назначить редактор свойства наследником TStringListProperty и, согласно обстоятельствам, вызывать метод предка Edit:

```
unit MyEditor;

interface

type
  TMyStringListProperty = class(TStringListProperty)
  procedure Edit; override;
end;

implementation

procedure TMyStringListProperty.Edit;
begin
  if { какие-то условия } then
    { что-то делаем }
  else
    inherited Edit;
end;

end.
```

[News Group]

Код определения свойств

На просторах Интернета можно найти компонент с именем TdemoProps. Данный компонент отлично иллюстрирует механизм сохранения и чтения значений свойств. Установите этот компонент в палитру Delphi, перенесите экземпляр на форму, просмотрите форму как текст. Вы увидите дополнительные свойства StringThing и Thing. Первое – свойство строки, второе – бинарное свойство, фактически запись. Если у вас есть HexEdit (шестнадцатеричный редактор) или что-то аналогичное, откройте в нем .DFM-файл, чтобы просмотреть теги ваших новых свойств вместе с их именами.

Если TReader/TWriter имеет специфические методы чтения/записи свойств, и вы хотите добавить, например, строку, целое, символ или что-то еще (проверьте описание соответствующих методов TReader в файлах помощи), то в этом случае используйте DefineProperty. В случае сложного объекта выберите метод DefineBinaryProperty, и ваши методы чтения и записи получат TStream вместо TReader/TWriter.

```
unit PropDemo;

interface

uses
  Windows, Messages, Classes, Graphics, Controls, Forms, Dialogs;

type
  TDemoProps = class(TComponent)
```

```
  private
    FStringThing: string;
    FThing: record
              i, j, k: integer;
              x, y: real;
              ch: char;
    end;
    procedure ReadStringThing(Reader: TReader);
    procedure WriteStringThing(Writer: TWriter);
    procedure ReadThing(Stream: TStream);
    procedure WriteThing(Stream: TStream);
  protected
    procedure DefineProperties(Filer: TFiler); override;
  public
    constructor Create(AOwner: TComponent); override;
  end;

procedure Register;

implementation

constructor TDemoProps.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  { создайте любые данные, чтобы было что передать в поток }
  FStringThing := 'Всем привет!';
  with FThing do begin
    i := 1;
    j := 2;
    k := 3;
    x := PI;
    y := 180 / PI;
    ch := '?';
  end;
end;

procedure TDemoProps.ReadStringThing(Reader: TReader);
begin
  FStringThing := Reader.ReadString;
end;

procedure TDemoProps.WriteStringThing(Writer: TWriter);
begin
  Writer.WriteString(FStringThing);
end;

procedure TDemoProps.ReadThing(Stream: TStream);
begin
  Stream.ReadBuffer(FThing, SizeOf( FThing));
end;
```

```
procedure TDemoProps.WriteThing(Stream: TStream);
begin
  Stream.WriteBuffer(FThing, SizeOf(FThing));
end;

procedure TDemoProps.DefineProperties(Filer: TFiler);
begin
  inherited DefineProperties(Filer);
  Filer.DefineProperty('StringThing', ReadStringThing,
                       WriteStringThing, FStringThing <> '');
  Filer.DefineBinaryProperty('Thing', ReadThing, WriteThing, true);
end;

procedure Register;
begin
  RegisterComponents('Samples', [TDemoProps]);
end;

end.
```

[News Group]

Отображение свойств во время выполнения программы

Воспользуйтесь предлагаемым кодом.

Примечание

Код был написан еще для Delphi 1, поэтому использован устаревший компонент TOutLine. *Данный материал хорошо описан в книге С. Орлика «Секреты Delphi на примерах», издательство БИНОМ, 1996 г.*

Вызовите функцию DisplayProperties так, как показано далее:

```
DisplayProperties(Form1, Outline1.Lines, 0);
```

```
uses TypInfo;
function GetIndentSpace(iIndentLevel: Integer): String; var iCnt: Integer;
begin
  Result := '';
  for iCnt := 0 to iIndentLevel - 1 do Result := Result + #9;
end;

procedure DisplayProperties(AObj: TObject; AList: TStrings; iIndentLevel: Integer);
var
  Indent: String;
  ATypeInfo: PTypeInfo;
  ATypeData: PTypeData;
  APropTypeData: PTypeData;
  APropInfo: PPropInfo;
  APropList: PPropList;
  iProp: Integer;
  iCnt: Integer;
```

```
  iCntProperties: SmallInt;
  ASecondObj: TObject;

  procedure AddLine(sLine: String);
  begin
    AList.Add(Indent + #160 + IntToStr(iProp) + ': ' + APropInfo^.Name
         + ' (' + APropInfo^.PropType^.Name + ')' + sLine);
  end;

begin
  try
    Indent := GetIndentSpace(iIndentLevel);
    ATypeInfo := AObj.ClassInfo;
    ATypeData := GetTypeData(ATypeInfo);
    iCntProperties := ATypeData^.PropCount;
    GetMem(APropList, SizeOf(TPropInfo) * iCntProperties);
    GetPropInfos(ATypeInfo, APropList);
    for iProp := 0 to ATypeData^.PropCount - 1 do begin
      APropInfo := APropList^[iProp];
      case APropInfo^.PropType^.Kind of
        tkInteger:
              AddLine(' := ' + IntToStr(GetOrdProp(AObj, APropInfo)));
         tkChar:
              AddLine(' := ' + chr(GetOrdProp(AObj, APropInfo)));
        tkFloat:
              AddLine(' := ' + FloatToStr(GetFloatProp(AObj, APropInfo)));
        tkString:
              AddLine(' := "' + GetStrProp(AObj, APropInfo) + '"');
          tkSet:
              AddLine(' := ' + IntToStr(GetOrdProp(AObj, APropInfo)));
        tkClass:
              begin
                ASecondObj := TObject(GetOrdProp(AObj, APropInfo));
                if ASecondObj = NIL then AddLine(' := NIL')
                else begin
                 AddLine('');
                 DisplayProperties(ASecondObj, AList, IIndentLevel + 1);
                end;
              end;
         tkMethod:
              AddLine('');
      else
        AddLine(' := >>НЕИЗВЕСТНО<<');
      end;
    end;
  except        { Выводим исключение и продолжаем дальше }
    on e: Exception do ShowMessage(e.Message);
  end;
  FreeMem(APropList, SizeOf(TPropInfo) * iCntProperties);
end;
```

[News Group]

Имя свойства в течение выполнения приложения

Если тип объекта не известен, то программный код может быть следующим:

```
if (Sender is TLabel) then ...
```

а если известен, то:

```
TLabel(Sender).Caption ...
```

Чтобы получить доступ к имени, воспользуйтесь одним из приведенных ниже примеров:

```
FormName := (MyForm as TObject).Name;
```

или

```
FormName := (MyForm as TControl).Name;
```

Редактор свойств для точки

Редактор TPoint не располагает информацией о типе данных. Следовательно, нельзя зарегистрировать для него редактор свойств. Можно применять редактор свойств только для строк, реальных и порядковых чисел, или указателей на объекты. Дело в том, что редактор свойств использует только следующие методы, чтобы иметь доступ к свойствам через RTTI:

```
GetValue/SetValue              для строк (strings)
GetFloatValue/SetFloatValue    для натуральных чисел (floats)
GetOrdValue/SetOrdValue        для порядковых (и указателей)
```

Решением может быть создание класса TPersistentPoint, являющегося наследником TPersistent и имеющего те же свойства, что и TPoint. Можете просто «обернуть» TPoint для хранения значений или создать явные поля. Непосредственная работа с TPoint сделает программирование с применением метода Assign легким и быстрым. Для процедур чтения и записи можно использовать поля записи, как показано ниже:

```
  ...
type
  TPersistentPoint = class(TPersistent)
  private
    FPoint: TPoint;
  published
    property X: integer read FPoint.X write FPoint.X;
    property Y: integer read FPoint.Y write FPoint.Y;
  end;
```

[News Group]

Свойство только для чтения во время выполнения приложения

Если значение свойства используется в процедуре, то сама процедура может содержать следующий код:

```
if csDesigning in ComponentState then begin
  //... код, устанавливающий значение свойства ...
end;
```

Это позволяет в режиме выполнения приложения присвоить свойству значение «только для чтения».

Свойство TStringList

Следующий пример демонстрирует подход, необходимый для создания свойства, имеющего тип TStringList:

```
  ...
  private
    FList: TStrings;
  protected
    procedure SetList(Value: TStrings);
  published
    property List: TStrings read FList write SetList;
  ...

constructor Txxxxx.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  FList := TStringList.Create;
end;

destructor Txxxxx.Destroy;
begin
  FList.Free;
  inherited Destroy;
end;

procedure Txxxxx.SetList(Value: TStrings);
begin
  FList.Assign(Value);
end;
```

[News Group]

Конфликт имен параметров

Некоторые обработчики событий (типа TStringGrid.OnDrawCell) получают параметры с именами ACol и ARow.

Возникает проблема при использовании следующего кода:

```
with Sender as TStringGrid do ...
```

где передаваемые параметры ACol и ARow теперь «прячутся» за пределами "with".

Чтобы обойти эту проблему, объявите:

```
var
  CellCol: integer absolute ACol;
  CellRow: integer absolute ARow;
```

Это позволяет обойтись без локальных переменных и работать с одноименными параметрами.

Вызов процедуры, имя которой содержится в переменной

Как вызвать процедуру, имя которой хранится в таблице, списке и т. п.?

Решение 1

```
unit ProcD;

interface

type
  MyProc = procedure(s: String);
  procedure RegisterProc(procName: String; proc: MyProc);
  procedure ExecuteProc(procName: String; arg: String);

implementation

uses
  Classes;

var
  ProcDict: TStringList;

procedure RegisterProc(procName: String; proc: MyProc);
begin
  ProcDict.AddObject(procName, TObject(@proc));
end;

procedure ExecuteProc(procName: String; arg: String);
var
  index: Integer;
begin
  index := ProcDict.IndexOf(ProcName);
  if index >= 0 then MyProc(ProcDict.Objects[index])(arg);
  // Можно вставить обработку исключительной ситуации - сообщение об ошибке
end;

initialization
  ProcDict := TStringList.Create;
  ProcDict.Sorted := true;
```

```
finalization
  ProcDict.Free;
end.
```

Решение 2

Можно создать переменную типа StringList, как показано далее:

```
StringList.Create;
StringList.AddObject('Proc1', @Proc1);
StringList.AddObject('Proc2', @Proc2);
```

Затем реализовать это в программе:

```
var
  myFunc: procedure;
begin
  if Stringlist.IndexOf(S) = -1 then
    MessageDlg('Не понял процедуру ' + S, mtError, [mbOk], 0)
  else begin
    @myFunc := Stringlist.Objects[Stringlist.IndexOf(S)];
    myFunc;
  end;
end;
```

[News Group]

Выполнение процедуры по ее адресу

Ключом здесь является использование оператора @, расположенного в левой части процедурной переменной.

```
var
  P: procedure(x, y: double);
procedure TForm1.Button1Click(Sender: TObject);
begin
  // получаем handle hDLL
  @P := GetProcAddress(hDLL, 'SOMEPROC');
  P(3, 4);
end;
```

[News Group]

Передача функции как параметра

В этом случае лучшим решением будет использование процедурного типа. Допустим, что DllFunction() на входе хочет получить определенную функцию. Поясним это на примере:

```
type
  TMyFuncType = function: integer;
```

```
var
  MyFunc: TMyFuncType;

function foo: integer;
begin
  result := 1;
end;

begin
  MyFunc := foo;
  DllFunction(longint(MyFunc));
```

Можно это сделать и так:

```
DllFunction(longint(@foo));
```

Для корректной работы необходимо объявить `foo` с директивой `far`, т. е. экспортировать ее в модуле.

Также, в зависимости от того, как написана `DllFunction()`, можно в вызове подразумевать приведение типа:

```
function DllFunction(p: TMyFuncType): Integer; far; external 'mydll';
```

В этом случае не нужна переменная `MyFunc` или оператор `@`.

В Delphi/Pascal можно передавать функции как параметры. Но для того чтобы этим воспользоваться, необходимо установить для компилятора тип.

Проверьте следующий код:

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}
```

```
type
  IntFunc = function: integer;

function DllFunction(iFunc: IntFunc): integer; far;
begin
 // Обратите внимание на то, что это вызов функции
  DllFunction := iFunc;
end;

function iFoo: integer; far;
begin
  iFoo := 1;
end;

procedure TestIFunc;
var
  i: integer;

begin
  i := DllFunction(iFoo);
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  TestIFunc;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  Close;
end;

end.
```

Можно воспользоваться двумя способами. Первый заключается в использовании следующего кода:

```
i := longint(@foo);
```

Такой способ подойдет, если вы хотите применить для передачи `longint`. Другой вариант, которым можно воспользоваться, – исключить работу с `longint` и вызывать функцию DLL следующим образом:

```
DLLfunction(@foo);
```

Имейте в виду, что если собираетесь вызывать `foo` из DLL, то необходимо предусмотреть вопросы совместимости для получения дополнительной информации почитайте описание функции `MakeProcInstance`.

Переменная в качестве имени процедуры

Каким образом можно использовать переменную типа `String` *в качестве имени процедуры?*

Если все процедуры, которые вы собираетесь вызывать, имеют список с одними и теми же параметрами, (или все без параметров), то это не трудно. Для этого необходимы:

- процедурный тип, соответствующий вашей процедуре, например:

```
type
  TMacroProc = procedure(param: Integer);
```

- массив, сопоставляющий имена процедур их адресам во время выполнения приложения:

```
type
  TMacroName = string[32];
  TMacroLink = record
    name: TMacroName;
    proc: TMacroProc;
  end;
  TMacroList = array [1..MaxMacroIndex] of TMacroLink;

const
  Macros: TMacroList = (
       (name: 'Proc1'; proc: Proc1),
       (name: 'Proc2'; proc: Proc2),

         ...
     );
```

- интерпретатор функций, типа:

```
procedure CallMacro(name: String; param: Integer);
var
  i: Integer;
begin
  for i := 1 to MaxMacroIndex do
    if CompareText(name, Macros[i].name) = 0 then begin
      Macros[i].proc(param);
      break;
    end;
end;
```

Макропроцедуры необходимо объявить в секции `Interface` модуля либо с ключевым словом `Far`, например:

```
procedure Proc1(n: Integer); far;
begin
  ...

end;
```

```
procedure Proc2(n: Integer); far;
begin
  ...
end;
```

[News Group]

Переменное количество параметров любого типа

Список параметров процедуры можно определить как 'x: array of const' и использовать почти любой тип параметра.

Пример

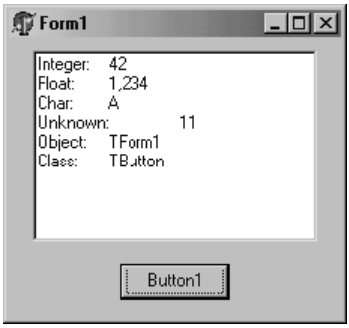
Разместите на форме компоненты Memo и Button и добавьте строку 'procedure Display(X: array of const);' в определения класса формы после комментария {Private Declarations}. Создайте функцию, подобную следующей:

```
procedure TForm1.Display(X: array of const);
var
  I: Integer;
begin
  Memo1.Clear;
  for I := 0 to High(X) do with TVarRec(X[I]) do
    with Memo1.Lines do
      case VType of
        vtInteger:  Add('Integer:'#9 + IntToStr(VInteger));
        vtBoolean:  if VBoolean then Add('Boolean:'#9'True')
                    else Add('Boolean:'#9'False');
           vtChar:  Add('Char:'#9 + VChar);
       vtExtended:  Add('Float:'#9 + FloatToStr(VExtended^));
         vtString:  Add('String:'#9 + VString^);
        vtPointer:  Add('Pointer:'#9 + Format('%p', [VPointer]));
          vtPChar:  Add('PChar:'#9 + StrPas(VPChar));
         vtObject:  Add('Object:'#9 + VObject.ClassName);
          vtClass:  Add('Class:'#9 + VClass.ClassName);
       else
         Add('Unknown:'#9 + IntToStr(VType));
       end;
end;
```

Теперь в обработчике события кнопки OnClick вызываем процедуру Display и передаем ей любой тип: числа, строки, PChar, объекты.

```
Display([42, 1.234, 'A', 'Васек Трубачев', Form1, TButton]);
```

Практически это программа с переменным числом параметров. На самом деле параметр один, но он является массивом, содержащим переменное количество параметров различного типа.



[News Group]

Проблема передачи записи

Определите базовый класс `Allrecs`:

```
tAllrecs = class
  function getVal(field: integer): string; virtual;
end;
```

Затем создайте классы для каждой записи:

```
recA = class(tAllrecs)
  this: Integer;
  that: String;
  the_other: Integer;
  function getVal(field: integer): string; virtual;
end;
```

Для каждой функции класса определите возвращаемый результат:

```
function recA.getVal(field: integer); string;
begin
  case field of
    1:  getVal := intToStr (this);
    2:  getVal := that;
    3:  getVal := intToStr (the_other);
  end;
end;
```

Затем определите:

```
function myFunc(rec: tAllrecs; field: integer);
begin
  label2.caption := allrecs.getVal(field);
end;
```

Теперь можно вызвать myFunc с любым классом, производным от tAllrecs, например:

```
myFunc(recA, 2);
myFunc(recB, 29);
```

[News Group]

Работа метода Assign

В общем случае утверждение 'Destination := Source' не идентично утверждению 'Destination.Assign(Source) '. Присваивание 'Destination := Source' принуждает Destination ссылаться на тот же объект, что и Source, а 'Destination.Assign(Source) ' копирует содержание объектных ссылок Source в объектные ссылки Destination. Если Destination является свойством некоторого объекта (тем не менее, и свойство не является ссылкой на другой объект, как, например, свойство формы ActiveControl или свойство DataSource элементов управления для работы с базами данных), то утверждение 'Destination := Source' идентично утверждению 'Destination.Assign(Source) '. Это объясняет, почему LB.Items := MemStr работает, в то время как MemStr := LB.Items – нет.

Создание объектных переменных

Материал, изложенный ниже, будет полезен всем, кто хоть раз сталкивался с ошибками указателей, исключительными ситуациями и GPF.

Назначаем переменную типа «некоторый класс»:

```
var
  MyVar: TMyClass;
```

Всех интересует: что делает в этом случае компилятор и достаточно ли выделено памяти для хранения указателя на экземпляр данного класса в куче памяти. Назначив такую переменную, вы распределили память не для данного класса, а только для указателя. Компилятор всегда инициализирует этот указатель в $FFFFFFFF, а это значит, что распределенный блок памяти пуст. Во всяком случае, этого достаточно, чтобы утверждать, что указатель не обеспечивает верную «позицию» памяти, и класс не содержит никаких данных.

Delphi распределяет и заполняет память автоматически, но и вы должны немного потрудиться. При использовании одного из классов Delphi или своего собственного, необходимо создать его экземпляр. Это значит, что вы должны распределить память и установить указатель на распределенный блок памяти. В Delphi это выглядит следующим образом:

```
MyVar := TMyClass.Create;
```

Это действительно просто, поскольку метод конструктора Create класса TMyClass является классовым методом – он работает в классе, а не в отдельном объекте. Когда вызывается конструктор, Delphi распределяет память и воз-

вращает значение указателя. Присмотритесь: не похоже ли это на вызов функции? Очень хорошо, если вы раньше не знали, что возвращалось при вызове, то теперь знаете. Вызов TMyClass.Create возвращает указатель на объект типа TMyClass.

Все, что действительно нужно помнить, – это:

- Объявление объектной переменной некоторого типа
- Создание объекта вызовом метода конструктора класса
- Использование объекта по назначению
- Освобождение объекта

```
procedure Example;
var
  MyObj: TMyClass;      // класс, который вы создаете
  MyList: TList;        // встроенный класс
begin
  MyObj := TMyClass.Create;
  // теперь MyObj содержит адрес блока памяти, распределенной для
  // экземпляра вашего класса
  MyList := TList.Create;
  // Код для работы с MyList
  ...
  // здесь мы что-то делаем с объектом
  ...
  MyList.Free;          // Ресурсы MyList удаляются из кучи
  MyObj.Free;           // то же самое для MyObj
end;
```

Создание объектов любого типа

Файл CLASSES.PAS определяет функцию с именем FindClass, возвращающую классовую ссылку на этот класс (такую же, как и при регистрации класса). Данная функция динамически создает компоненты на основе имени класса, введенного в поле редактирования. При этом необходимо вызвать RegisterClasses и перечислить все возможные классы, которые планируется создать.

```
unit Unit1;

interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, StdCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    Edit1: TEdit;
    procedure Button1Click(Sender: TObject);
  public
    NextTop: integer;
  end;
```

```
var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
var
  NewObj : TControl;
  NewClass: TPersistentClass;
begin
  NewClass := FindClass(Edit1.Text);
  NewObj := TControl(TComponentClass(NewClass).Create(Self));
  NewObj.Parent := Self;
  NewObj.Name := NewObj.ClassName + IntToStr(NextTop);
  NewObj.Top := NextTop;
  NextTop := NextTop + 20;
end;
initialization
  RegisterClasses([TButton, TEdit, TLabel]);
end.
```

Особенность использования StrAlloc и GetMem

Функция `StrAlloc` непосредственно использует процедуру `GetMem`, поэтому их работа невозможна по отдельности – обе «живут» за счет кучи. `StrAlloc` имеет преимущество, т. к. распределяет на четыре байта памяти больше, чем вам требуется, и хранит в них размер размещенного блока. Поэтому `StrDispose` знает, сколько памяти освободить. Вы можете этого не помнить (в отличие от `GetMem/FreeMem`).

Не смешивайте при работе эти две функциональные пары, иначе вы будете потрясены количеством возникающих ошибок.

Быстрое сравнение памяти

Предлагаем две функции, существенно повышающие производительность в приложениях, активно работающих с данными. Нужно всего лишь обеспечить контроль типов и границ допустимого диапазона, все остальное они сделают сами.

```
function Keys_are_Equal(OldRec, NewRec: OpenString;
                        KeyLn: integer): boolean; assembler;
asm
  push  ds
  mov   al, 01
  cld
  les   di, NewRec
  lds   si, OldRec
```

```
    mov   cx, KeyLn
    cli
    repe  cmpsb
    sti
    jz    @1
    xor   al, al
  @1:
    pop   ds
  end;

  function First_Key_is_Less(NewRec, OldRec: OpenString;
                             Keyln: integer): boolean; assembler;
  asm
    push  ds
    mov   al, 01
    cld
    les   di, NewRec
    lds   si, OldRec
    mov   cx, Keyln
    cli
    repe  cmpsb
    sti
    jz    @5
    jge   @6
  @5:
    xor  al, al
  @6:
    pop  ds
  end;
```

[News Group]

Примечание

Функции написаны уже давно – когда еще была директива `assembler` *и Microsoft «не догадывалась», что можно написать Win32. Нашим читателям придется приложить некоторые усилия для воплощения этой идеи, но она того стоит (хотя бы для тренировки).*

Арифметика указателей

Что можно рассказать об арифметике указателей в Delphi?

Для начала дадим короткое определение понятия арифметики указателей. При работе с динамической памятью все, чем вы располагаете, – это указатели на начала блоков памяти. Предположим, вам нужно просмотреть или обработать какие-то блоки памяти. Это возможно осуществить путем изменения указателя, который показывает на нужный фрагмент данных в памяти. Это называется арифметикой указателя.

Основополагающая идея при занятиях арифметикой указателей – указатель должен быть увеличен на значение корректного приращения. Корректное

приращение определяется размером объекта, на который показывает указатель. Например: char = 1 байт; integer = 4 байта; double = 8 байт и т. д. Функции Inc() и Dec() изменяют значение корректного приращения. Компилятор знает правильный размер объекта.

Для динамического распределения памяти можно воспользоваться предлагаемым модулем:

```
var
  MyArray: array[0..30] of char;
  b: ^char;
  i: integer;

begin
{ помещаем что-то в память (блок данных) }
  StrCopy(MyArray, 'Дельфи - это здорово!');
{ назначаем указатель на текущую позицию памяти }
  b := @MyArray;
  for i := StrLen(MyArray) downto 0 do begin
    if(b^) ...          { работаем с символом в текущей позиции указателя }
    inc(b);           { перемещаем указатель на следующий байт памяти }
  end;
```

Нижеследующий код демонстрирует работу функций Inc() и Dec(), увеличивающих или уменьшающих указатель на размер соответствующего типа:

```
var
  P1, P2: ^LongInt;
  L: LongInt;
begin
  P1 := @L;          { назначаем оба указателя на одно и то же место }
  P2 := @L;
  Inc(P2);           { Увеличиваем один }
{ Здесь мы получаем разницу между смещениями двух указателей. Поскольку
  первоначально они указывали на одно и то же место памяти, то результатом данного
  вызова будет разница между двумя указателями после вызова Inc(). }
  L := Ofs(P2^) - Ofs(P1^);      { L = 4; т.е. SizeOf(LongInt) }
end;
```

Можно изменить тип объекта, на который указывает P1 и P2, на какой-то другой и убедиться, что (SizeOf(P1^)) всегда возвращает величину корректного приращения.

Динамическое распределение памяти

Как уменьшить количество занимаемой памяти в сегменте данных?

Структура данных может быть подобна следующей:

```
type
  TMyStructure = record
    Name: String[40];
```

```
    Data: array[0..4095] of Integer;
end;
```

Она слишком большая для глобального распределения, так что вместо объявления глобальной переменной

```
var
  MyData: TMyStructure;
```

вы объявляете указатель на нужный тип и переменную этого типа:

```
type
  PMyStructure = ^TMyStructure;

var
  MyDataPtr: PMyStructure;
```

Такой указатель занимает всего лишь четыре байта сегмента данных.

Прежде чем использовать структуру данных, необходимо распределить ее в куче и получить к ней доступ через глобальные данные любым удобным для вас способом. Единственное отличие от традиционного способа заключается в необходимости использования символа «^» для обозначения указателя на данные:

```
New(MyDataPtr);
MyDataPtr^.Name := 'Советы по Delphi';
MyDataPtr^.Data[0] := 12345;
```

Завершив работу с памятью, освободите ее:

```
Dispose(MyDataPtr);
```

Массив объектов изображений

Как создать массив объектов изображений?

Сделать это напрямую и «визуально» не получится, но посмотрите на приведенный код:

```
type
  TForm1 = class(TForm)
  ...
  public
    images: array [1..10] of TImage;
  ...
end;

procedure TForm1.FormCreate(...);
var
  i: integer;
```

```
begin
  ...
  for i := 1 to 10 do begin
    images[i] := TImage.Create(self);
    with images[i] do begin
      parent := self;
      tag := i;    // это облегчит идентификацию изображения
                   //... установите другие необходимые свойства, например:
      OnClick := MyClickEventHndlr;
    end;
  end;
  ...
end;
```

Для того чтобы убедиться, что все модули в секции uses установлены правильно, перенесите на форму один такой динамический компонент и затем удалите его или установите его видимость в False. Более сложный способ заключается в разработке собственного компонента, делающего описанное выше.

Сохранение массива с изображениями

Как сохранить массив с изображениями?

Идея заключается в загрузке каждого изображения TBitmap во временный TMemoryStream. Свойство TMemoryStream.Size информирует нас о размере данных, которые нужно сохранить на диске. Затем запишите размер и сопроводите его данными. Эту манипуляцию проделайте для каждого TBitmap в массиве.

Для процедуры чтения сначала необходимо считать из потока размер данных TBitmap. Затем распределить область для типа TMemoryStream полученного размера и считать данные. Затем перепишите информацию из TFileStream в TMemoryStream. И, наконец, считайте из TMemoryStream сам объект TBitmap. Эту манипуляцию проделайте для каждого TBitmap в массиве.

```
procedure TForm1.SaveBoard;
var
  MemoryStream: TMemoryStream;
  FileStream: TFileStream;
  Writer: TWriter;
  Buffer: Pointer;
  Size: Longint;
  Column: Integer;
  Row: Integer;
begin
  MemoryStream := TMemoryStream.Create;
  FileStream := TFileStream.Create(SaveFilename, fmCreate);
  Writer := TWriter.Create(FileStream, $1000);
  try
    for Column := 0 to 4 do
      for Row := 0 to 4 do begin
```

```
        MemoryStream.Clear;
        Bitmaps[Column, Row].SaveToStream(MemoryStream);
        Buffer := MemoryStream.Memory;
        Size := MemoryStream.Size;
        Writer.WriteInteger(Size);
        Writer.Write(Buffer^, Size);
      end;
  finally
    Writer.Free;
    FileStream.Free;
    MemoryStream.Free;
  end;
end;

procedure TForm1.Open1Click(Sender: TObject);
var
  MemoryStream: TMemoryStream;
  FileStream: TFileStream;
  Buffer: Pointer;
  Reader: TReader;
  Column: Integer;
  Row: Integer;
  Size: Longint;
begin
  OpenDialog1.Filename := SaveFilename;
  if not OpenDialog1.Execute then Exit;
  MemoryStream := TMemoryStream.Create;
  FileStream := TFileStream.Create (OpenDialog1.Filename, fmOpenRead);
  Reader := TReader.Create(FileStream, $1000);
  try
    for Column := 0 to 4 do
      for Row := 0 to 4 do begin
        Size := Reader.ReadInteger;
        MemoryStream.SetSize(Size);
        Buffer := MemoryStream.Memory;
        Reader.Read(Buffer^, Size);
        Bitmaps[Column, Row].LoadFromStream(MemoryStream);
      end;
  finally
    Reader.Free;
    FileStream.Free;
    MemoryStream.Free;
  end;
  DrawGrid1.Repaint;
  SaveFilename := OpenDialog1.Filename;
  Caption := 'Bingo-создатель - ' + ExtractFilename (SaveFilename);
end;
```

[News Group]

Динамические массивы

Возможно ли создание динамически изменяющихся массивов в Delphi?

Решение 1

Для начала надо создать тип массива, зарезервировав для него самый большой размер, который когда-либо может понадобиться. Затем необходимо создать переменную, являющуюся указателем на этот тип. Это заставит компилятор распределить лишь четыре байта, необходимые для размещения указателя.

Но прежде чем воспользоваться массивом, для него необходимо распределить память. Функция `AllocMem` позволяет точно управлять выделяемым размером памяти. Для того чтобы определить количество байт, которые нужно распределить, умножьте размер массива на размер отдельного элемента массива. Имейте в виду, что самый большой блок, который можно распределить в любой момент в 16-битной среде, равен 64 Кбайт. Самый большой блок, который можно распределить в 32-битной среде, равен 4 Гбайт. Для определения максимального числа элементов, которые можно иметь в конкретном массиве (в 16-битной среде), разделите 65 520 на размер отдельного элемента. Например: `65520 div SizeOf(LongInt)`.

Пример задания типа массива и указателя:

```
type
  ElementType = LongInt;

const
  MaxArraySize = (65520 div SizeOf(ElementType));      // в 16-битной среде

type
  MyArrayType = array[1..MaxArraySize] of ElementType;

var
  P: ^MyArrayType;

const
  ArraySizeIWant: Integer = 1500;
```

Затем для распределения памяти под массив можно использовать следующую процедуру:

```
procedure AllocateArray;
begin
  if ArraySizeIWant <= MaxArraySize then
    P := AllocMem(ArraySizeIWant * SizeOf(LongInt));
end;
```

Не забывайте о том, что величина `ArraySizeIWant` должна быть меньше или равна `MaxArraySize`. Процедура, которая с помощью цикла устанавливает величину каждого члена:

```
procedure AssignValues;
var
  I: Integer;
begin
  for I := 1 to ArraySizeIWant do P^[I] := I;
end;
```

Имейте в виду, что необходимо самому организовать контроль допустимого диапазона. Если память распределена для массива с пятью элементами и вы попытаетесь назначить какое-либо значение шестому, то вызовете ошибку и, возможно, порчу памяти.

Помните также и о том, что после использования массива всю распределенную память необходимо освободить. Пример того, как избавиться от этого массива:

```
procedure DeallocateArray;
begin
  FreeMem(P, ArraySizeIWant * SizeOf(LongInt));
end;
```

Ниже приведен пример динамического массива:

```
unit Unit1;

interface

uses
  SysUtils, Windows, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  ElementType = Integer;

const
  MaxArraySize = (65520 div SizeOf(ElementType));      //в 16-битной среде

type
{ Создаем тип массива. Убедитесь в том, что вы установили максимальный диапазон,
  который вам может понадобиться. }
  TDynamicArray = array[1..MaxArraySize] of ElementType;
  TForm1 = class(TForm)
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
end;

var
  Form1: TForm1;
  P: ^TDynamicArray; { Создаем указатель на ваш тип массива }
```

```
const
{ Это типизированные константы. В действительности они являются статическими
  переменными, инициализируемыми во время выполнения указанными в исходном коде
  значениями. Это означает, что вы можете использовать типизированные константы
  точно так же, как и любые другие переменные. Удобство заключается в автоматически
  инициализируемой величине. }
  DynamicArraySizeNeeded: Integer = 10;

implementation
{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
{ Распределяем память для нашего массива. Будьте внимательны и распределяйте
  размер, в точности необходимый для размещения нового массива. Если вы попытаетесь
  записать элемент, выходящий за допустимый диапазон, компилятор не возразит, но
  объект исключения вам обеспечен }
  DynamicArraySizeNeeded := 500;
  P := AllocMem(DynamicArraySizeNeeded * SizeOf(Integer));
{ Как присвоить значение пятому элементу массива ... }
  P^[5] := 68;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
{ Вывод данных. }
  Button1.Caption := IntToStr(P^[5]);
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
{ Освобождаем распределенную для массива память. }
  FreeMem(P, DynamicArraySizeNeeded * SizeOf(Integer));
end;

end.
```

Решение 2

Для того чтобы работать с динамическими массивами, можно использовать два вида объявления типов:

```
type
  DAr = array of real;
var
  A: DAr;
```

или

```
var
  A: array of real;
```

Таким образом, получена ссылка на область памяти. Для того чтобы указать размер, обратитесь к процедуре SetLength. Ее также можно использовать в любом месте для определения размера массива, который необходим в данную минуту.

```
SetLength(A,7)
```

Так создается массив, состоящий из 7 элементов начиная с 0.

Важно

Первый элемент в динамическом массиве всегда нулевой.

Для определения верхней границы используем функцию High.

```
I := High(A);
```

I – верхняя граница. Для определения длины – Length(A), для определения нижней границы – Low(A). При нулевой длине массива High возвращает -1.

Пример

```
var
  a, b: array of integer;
begin
  SetLength(a, 2);
  SetLength(b, 2);
  a[0] := 2;
  b[0] := 3;
  a := b;
  b[0] := 4;
end;
```

После этих манипуляций a[0] равно 4. Дело в том, что присвоение a := b не вызывает копирования, т. к. a и b – это всего лишь указатели на область памяти. Для копирования необходимо использовать функцию Copy.

[Виталий Еремеев]

Решение 3

Создание одно- и двумерных динамических массивов:

```
unit DynArray;
{ Модуль для создания двух очень простых классов обработки динамических массивов
  TDynaArray - одномерный массив, TDynaMatrix - двумерный динамический массив }

interface

uses SysUtils;

type
  TDynArrayBaseType = double;
```

```
const
  vMaxElements =(High(Cardinal) - $f) div SizeOf(TDynArrayBaseType);

type
  TDynArrayNDX = 1..vMaxElements;

{ самый большой массив TDynArrayBaseType, который мы можем объявить }
  TArrayElements = array[TDynArrayNDX] of TDynArrayBaseType;
  PArrayElements = ^TArrayElements;
  EdynArrayRangeError = CLASS(ERangeError);
  TDynArray = class

  private
    fDimension: TDynArrayNDX;
    fMemAllocated: word;
    function GetElement(N: TDynArrayNDX): TDynArrayBaseType;
    procedure SetElement(N: TDynArrayNDX; const NewValue: TDynArrayBaseType);
  protected
    Elements: PArrayElements;
  public
    constructor create(NumElements: TDynArrayNDX);
    destructor Destroy; override;
    procedure Resize(NewDimension: TDynArrayNDX); virtual;
    property dimension: TDynArrayNDX read fDimension;
    property Element[N: TDynArrayNDX]: TdynArrayBaseType
                  read GetElement write SetElement; default;
end;

const
  vMaxMatrixColumns = 65520 div SizeOf(TDynArray);
{ построение матрицы класса с использованием массива объектов TDynArray }

type
  TMatrixNDX = 1..vMaxMatrixColumns;

{ каждая колонка матрицы будет динамическим массивом }
  TmatrixElements = array[TMatrixNDX] of TDynArray;
  PmatrixElements = ^TMatrixElements;
  TDynaMatrix = class

  private
    fRows: TDynArrayNDX;
    fColumns: TMatrixNDX;
    fMemAllocated: longint;
    function GetElement(row: TDynArrayNDX; column: TMatrixNDX): TDynArrayBaseType;
    procedure SetElement(row: TDynArrayNDX; column: TMatrixNDX;
                         const NewValue: TDynArrayBaseType);
  protected
    mtxElements: PMatrixElements;
  public
    constructor Create(NumRows: TDynArrayNDX; NumColumns: TMatrixNDX);
```

```
    destructor Destroy; override;
    property rows: TDynArrayNDX read fRows;
    property columns: TMatrixNDX read fColumns;
    property Element[row: TDynArrayNDX; column: TMatrixNDX]: TdynArrayBaseType
                    read GetElement write SetElement; default;
end;

implementation

constructor TDynArray.Create(NumElements: TDynArrayNDX);
begin
  inherited Create;
  fDimension := NumElements;
  GetMem(Elements, fDimension * SizeOf(TDynArrayBaseType));
  fMemAllocated := fDimension * SizeOf(TDynArrayBaseType);
  FillChar(Elements^, fMemAllocated, 0);
end;

destructor TDynArray.Destroy;
begin
  FreeMem(Elements, fMemAllocated);
  inherited Destroy;
end;

procedure TDynArray.Resize(NewDimension: TDynArrayNDX);
begin
  if (NewDimension < 1) then
    raise EDynArrayRangeError.CreateFMT('Индекс вышел за границы
                                        диапазона: %d', [NewDimension]);
  Elements := ReAllocMem(Elements, fMemAllocated,
                         NewDimension * sizeof(TDynArrayBaseType));
  fDimension := NewDimension;
  fMemAllocated := fDimension * SizeOf(TDynArrayBaseType);
end;

function TDynArray.GetElement(N: TDynArrayNDX): TDynArrayBaseType;
begin
  if (N < 1) OR (N > fDimension) then
    raise EDynArrayRangeError.CreateFMT('Индекс вышел за границы
                                        диапазона: %d', [N]);
  result := Elements^[N];
end;

procedure TDynArray.SetElement(N: TDynArrayNDX; const NewValue: TDynArrayBaseType);
begin
  if (N < 1) or (N > fDimension) then
    raise EDynArrayRangeError.CreateFMT('Индекс вышел за границы
                                        диапазона: %d', [N]);
  Elements^[N] := NewValue;
end;
```

```
constructor TDynaMatrix.Create(NumRows: TDynArrayNDX; NumColumns: TMatrixNDX);
var
  col: TMatrixNDX;
begin
  inherited Create;
  fRows := NumRows;
  fColumns := NumColumns;
{ выделение памяти для массива указателей (т. е. для массива TDynArrays) }
  GetMem(mtxElements, fColumns * sizeof(TDynArray));
  fMemAllocated := fColumns * sizeof(TDynArray);
{ теперь выделяем память для каждого столбца матрицы }
  for col := 1 to fColumns do begin
    mtxElements^[col] := TDynArray.Create(fRows);
    inc(fMemAllocated, mtxElements^[col].fMemAllocated);
  end;
end;

destructor TDynaMatrix.Destroy;

var
  col: TMatrixNDX;
begin
  for col := fColumns downto 1 do begin
    dec(fMemAllocated, mtxElements^[col].fMemAllocated);
    mtxElements^[col].Free;
  end;
  FreeMem(mtxElements, fMemAllocated);
  inherited Destroy;
end;

function TDynaMatrix.GetElement(row: TDynArrayNDX;
                                column: TMatrixNDX): TDynArrayBaseType;
begin
  if (row < 1) or (row > fRows) then
    raise EDynArrayRangeError.CreateFMT('Индекс строки вышел за границы
                                        диапазона: %d', [row]);
  if (column < 1) or (column > fColumns) then
    raise EDynArrayRangeError.CreateFMT('Индекс столбца вышел за границы
                                        диапазона: %d', [column]);
  result := mtxElements^[column].Elements^[row];
end;

procedure TDynaMatrix.SetElement(row: TDynArrayNDX; column: TMatrixNDX;
                                 const NewValue: TDynArrayBaseType);
begin
  if (row < 1) or (row > fRows) then
    raise EDynArrayRangeError.CreateFMT('Индекс строки вышел за границы
                                        диапазона: %d', [row]);
  if (column < 1) or (column > fColumns) then
    raise EDynArrayRangeError.CreateFMT('Индекс столбца вышел за границы
                                        диапазона: %d', [column]);
```

```
    mtxElements^[column].Elements^[row] := NewValue;
  end;
  end.
```

Примечание

Проблема состоит в том, что Delphi не позволяет создать массив `TArrayElements` *(его размер больше 2 Гбайт), поэтому необходимо уменьшить его размерность до приемлемых размеров.*

Решение 4

Иногда разработчик, имея дело с массивами, не знает, какого размера массив ему нужен. Тогда приходится использовать динамические массивы.

```
var
  intArray: array of integer;
```

В такой записи размер массива не указывается. Чтобы использовать его дальше, необходимо определить его размер (обратите внимание, что размер динамического массива можно устанавливать в программе):

```
begin
  intArray := New(IntArray, 100);
end;
```

[Nikolaev Igor]

Решение 5

Предварительно опишите тип:

```
type
  TBigArray = array [0..30000] of Integer;
var
  a, b: ^TBigArray;
```

Распределите память из кучи с помощью `GetMem`.

```
GetMem(a, SizeOf(TBigArray));
GetMem(b, SizeOf(TBigArray));
```

Обращение к элементу массива будет иметь вид:

```
a^[0] := xxx;
```

Заполнение массива случайными значениями

Как заполнить массив случайными значениями?

Решение

```
procedure FillArray(var A: array of Integer);
var
  I, S, R: Integer;
```

```
begin
  Randomize;
  for I := 0 to High(A) do A[I] := I;
  for I := High(A) downto 0 do begin
    R := Random(I);
    S := A[R];
    A[R] := A[I];
    A[I] := S;
  end;
end;
```

[Иваненко Федор]

Массив констант

Как правильно использовать массив "array of const"?

`Array of const` – это массив переменных, декларированных как константы. Непосредственно они представлены структурой `TVarRec`. Круглые скобки просто ограничивают массив. Массив констант дает возможность передавать процедуре переменное количество параметров type-safe (безопасным) способом. Например:

```
type
  TVarRec = record
    Data: record case Integer of
      0: (L: LongInt);
      1: (B: Boolean);
      2: (C: Char);
      3: (E: ^Extended);
      4: (S: ^String);
      5: (P: Pointer);
      6: (X: PChar);
      7: (O: TObject);
    end;
    Tag: Byte;
    Stuff: array[0..2] of Byte;
  end;

function PtrToStr(P: Pointer): String;
const
  HexChar: array[0..15] of Char = '0123456789ABCDEF';

  function HexByte(B: Byte): String;
  begin
    Result := HexChar[B shr 4] + HexChar[B and 15];
  end;

  function HexWord(W: Word): String;
  begin
    Result := HexByte(Hi(W)) + HexByte(Lo(W));
  end;
```

```
begin
  Result := HexWord(HiWord(LongInt(P))) + ':' + HexWord(LoWord(LongInt(P)));
end;

procedure Display(X: array of const);
var
  I: Integer;
begin
  for I := 0 to High(X) do
    with TVarRec(X[I]), Data do begin
      case Tag of
        0: ShowMessage('Integer: ' + IntToStr(L));
        1: if B then ShowMessage('Boolean: True')
           else ShowMessage('Boolean: False');
        2: ShowMessage('Char: ' + C);
        3: ShowMessage('Float: ' + FloatToStr(E^));
        4: ShowMessage('String: ' + S^);
        5: ShowMessage('Pointer: ' + PtrToStr(P));
        6: ShowMessage('PChar: ' + StrPas(X));
        7: ShowMessage('Object: ' + O.ClassName);
      end;
    end;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  P: array[0..6] of Char;
begin
  P := 'Привет'#0;
  Display([-12345678, True, 'A', 1.2345, 'ABC', @P,  P, Form1]);
end;
```

[News Group]

Примечание

В данном случае проблемы будут с передачей строки 'ABC' как параметра, т. к. ее тип будет на самом деле не `String`*, а* `AnsiString` *(см. определение типа* `TVarRec`*).*

Массив без ограничения типа и размера

Как работать с неограниченными по размеру и типу массивами?

Такая возможность существует, начиная с Delphi 4.

```
type                  // опишем свой тип
  MyType = record
    zap1: longword;
    zap2: char;
    zap3: string[10];
  end;
```

```
// опишем НЕОГРАНИЧЕННЫЙ массив типа MyType
var
  m: array of MyType;
  ...

procedure TForm1.Button1Click(Sender: TObject);
var
  i: byte;
begin
  for i:=0 to 9 do begin           // нумерация элементов начинается с нуля!
    SetLength(m, Length(m) + 1);   // увеличение длины массива на 1
    m[i].zap1 := i;                // присвоение
    m[i].zap2 := chr(i);           // полям
    m[i].zap3 := inttostr(i);      // значений
  end;
end;
  ...
  SetLength(m, 0);                 // освобождение памяти
end.
```

[Дьяченко Сергей]

Массивы размером более 64K

Не существует способа непосредственного доступа к массиву размером свыше 65520 элементов. Или вы пользуетесь для распределения памяти GlobalAlloc или TMemoryStream и создаете специализированный класс для доступа к элементам массива, или вы делаете это непосредственно вручную. Добраться до следующих сегментов GlobalAlloc объекта можно, строя указатели с помощью SelectorInc. Самый простой способ заключается в применении TMemoryStream.

```
type
  Tmyarr = class
    buffer: TMemoryStream;
    elsize: LongInt;
    constructor Create(esize, number: Word);
    destructor Free;
    procedure SetElement(index: Word; p: Pointer);
    procedure GetElement(index: Word; p: Pointer);
end;

implementation

constructor Tmyarr.Create(esize, number: Word);
var
  size: LongInt;
begin
  Inherited Create;
  buffer := TMemoryStream.Create;
  elsize := esize;
```

```
  size := esize * number;
  buffer.SetSize(size);
end;

destructor Tmyarr.Free;
begin
  if Self <> Nil then begin
    buffer.Free;
    Destroy;
  end;
end;

procedure Tmyarr.GetElement(index: Word; p: Pointer);
begin
  buffer.Seek(elsize * index, 0);
  buffer.Read(p^, elsize);
end;
procedure Tmyarr.SetElement(index: Word; p: Pointer);
begin
  buffer.Seek(elsize * index, 0);
  buffer.Write(p^, elsize);
end;
```

[News Goup]

Шаблон массива переменной длины

Как динамически создать массив записей и получить доступ к отдельным элементам?

Необходимо определить тип массива, который может содержать максимальное количество записей, затем задать тип, являющийся указателем на массив. Идея заключается в том, чтобы не создавать экземпляр самого большого массива; а распределить память для необходимого вам количества записей, воспользовавшись указательным типом и `GetMem`.

```
unit %s;
{ Вы можете использовать этот шаблон для создания массива переменной длины любого
  типа данных. Превратите шаблон в модуль, выполнив во всем файле
  операцию поиска/замены для замены знака процента на свой тип данных. }

interface

const
  %MaxCapacity = High(Cardinal) div SizeOf(%);

type
  T%Index = 0..%MaxCapacity - 1;
  T%s = array[T%Index] of %;
  P%s = ^T%s;

function %sSize(Capacity: T%Index): Cardinal;
function Get%s(Capacity: T%Index): P%s;
```

```
function Resize%s(var P: P%s; OldCapacity, NewCapacity: T%Index): P%s;
procedure Free%s(var P: P%s; Capacity: T%Index);

implementation

uses
  SysUtils;

function %sSize(Capacity: T%Index): Cardinal;
begin
  Result := Capacity * SizeOf(%);
end;

function Get%s(Capacity: T%Index): P%s;
begin
  GetMem(Result, %sSize(Capacity));
end;
function Resize%s(var P: P%s; OldCapacity, NewCapacity: T%Index): P%s;
begin
  ReAllocMem(P, %sSize(OldCapacity), %sSize(NewCapacity));
end;

procedure Free%s(var P: P%s; Capacity: T%Index);
begin
  FreeMem(P, %sSize(Capacity));
  P := nil;
end;

end.
```

Приведенный выше модуль определяет тип массива и указатель на массив, далее приведены четыре полезные подпрограммы для работы с ним.

```
unit MyRecs;

interface
type
  MyRecord = record
    AnInt: Integer;
    AString: string[10];
  end;

const MyRecordMaxCapacity = High(Cardinal) div SizeOf(MyRecord);

type
  TMyRecordIndex = 0..MyRecordMaxCapacity - 1;
  TMyRecords = array[TMyRecordIndex] of MyRecord;
  PMyRecords = ^TMyRecords;

function MyRecordsSize(Capacity: TMyRecordIndex): Cardinal;
function GetMyRecords(Capacity: TMyRecordIndex): PMyRecords;
```

```
function ResizeMyRecords(var P: PMyRecords; OldCapacity,
                         NewCapacity: TMyRecordIndex): PMyRecords;
procedure FreeMyRecords(var P: PMyRecords; Capacity: TMyRecordIndex);
implementation
uses  SysUtils;
function MyRecordsSize(Capacity: TMyRecordIndex): Cardinal;
begin
  Result := Capacity * SizeOf(MyRecord);
end;

function GetMyRecords(Capacity: TMyRecordIndex): PMyRecords;
begin
  GetMem(Result, MyRecordsSize(Capacity));
end;
function ResizeMyRecords(var P: PMyRecords; OldCapacity,
                         NewCapacity: TMyRecordIndex): PMyRecords;
begin
  ReAllocMem(P, MyRecordsSize(OldCapacity), MyRecordsSize(NewCapacity));
end;

procedure FreeMyRecords(var P: PMyRecords; Capacity: TMyRecordIndex);
begin
  FreeMem(P, MyRecordsSize(Capacity));
  P := nil;
end;
end.
```

Пример работы с массивом переменной длины. Помните, что указатель должен использоваться с символом «^»:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  P: PMyRecords;
begin
  P := GetMyRecords(10);
  try
    P^[0].AnInt := 2001;
    P^[0].AString := 'Космическая одиссея';
  finally
    FreeMyRecords(P, 10);
  end;
end;
```

[News Group]

Запись массива в поток

Как выполнить запись массива в поток?

Структура сохраняемых данных выглядит следующим образом:

```
type
  TMyRec = record
```

```
    SomeField: Integer;
    SomeOtherField: Double;
    TheRest: array[0..99] of Single;
  end;
```

Массив MyBlobField имеет тип TblobField, а MyRec – TMyRec. Для копирования содержимого MyRec в MyBlobField необходимо произвести следующую запись:

```
var
  Stream: TBlobStream;

begin
  Stream := TBlobStream.Create(MyBlobField, bmWrite);
  Stream.Write(MyRec, SizeOf(MyRec));
  Stream.Free;
end;
```

[News Group]

Проблема циклических ссылок

Имеется объект A и объект B, и им обоим необходимо вызывать методы друг друга...

Создайте абстрактный базовый класс, определяющий интерфейс класса, чтобы другие классы могли видеть базовый. Используйте абстрактные виртуальные методы и свойства. Затем объявите другие классы подклассами базового класса (при необходимости). Данный метод существенно поможет в структурировании приложения.

Получение ссылки на экземпляр класса

Необходимо в подпрограмме получить ссылку на дочернее окно MDI без сообщения подпрограмме. С каким конкретно классом MDI необходимо работать?

Такая подпрограмма работает с дочерним окном, которое может иметь только один экземпляр. Если оно не открыто, подпрограмма создаст его. Если оно открыто, – переместит его на передний план.

```
procedure FormLoader(FormClassType: TFormClass; var FormName);
begin
  if TForm(FormName) = nil then begin
    Application.CreateForm(FormClassType, FormName);
  end

  else begin
    TForm(FormName).BringToFront;
    TForm(FormName).WindowState := wsNormal;
  end;
end;
```

Такой модуль вызывается следующим образом:

```
procedure TfrmTest.sbOpenClick(Sender: TObject);
begin
  FormLoader(TfrmTest, frmTest);
end;
```

Функция, возвращающая тип

Мне нужно, чтобы функция возвращала тип. Как это сделать?

Решение

```
// функция Chameleon, возвращающая тип сгенерированного исключения
unit Unit1;

interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  MyBoolean = class
  public
    Value: boolean;
  end;

  MyInteger = class
  public
    Value: integer;
  end;

  MyClass = class
  public
    Value: TStrings;
  end;

  TForm1 = class(TForm)
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  public
    procedure MyProc;
    function Chameleon: boolean;
  end;

var
  Form1: TForm1;
implementation
{$R *.DFM}

function TForm1.Chameleon: boolean;
var
  b: MyBoolean;
```

```
    i: MyInteger;
    c: MyClass;
    r: integer;
  begin
    Randomize;
    r := Random(3);
    case r of

      0: begin
          b := MyBoolean.Create;
          raise b;
        end;

      1: begin
          i := MyInteger.Create;
          raise i;
        end;

      2: begin
          c := MyClass.Create;
          raise c;
        end;
    end;
  end;

  procedure TForm1.MyProc;
  begin
    try
      Chameleon;
    except
      on MyBoolean do ShowMessage('Функция возвратила класс MyBoolean');
      on MyInteger do ShowMessage('Функция возвратила класс MyInteger');
      on MyClass do ShowMessage('Функция возвратила класс MyClass');
    end;
  end;

  procedure TForm1.Button1Click(Sender: TObject);
  begin
    MyProc;
  end;

  end.
```

Проблема с типизированными файлами

При переходе на Delphi появляются проблемы с типизированными файлами. Если целый тип занимает 4 байта, определения всех записей должны быть изменены с расчетом на то, что вместо целого типа придется работать с типом тип `SmallInt`. Тем не менее, даже после такого изменения размер записей остается прежним...

Для решения этой проблемы необходимо использовать модификатор packed:

```
type
  TMyRecType = packed record
    ...
  end;
```

Использование перечислимых констант

Необходимо получить в Инспекторе объектов раскрывающийся список для перечислимого типа. Как это сделать?

Для этого необходимо создать собственный редактор свойства. Ниже приведен его простой пример:

```
type
  TBaudRateProperty = class(TStringProperty)
public
  function GetAttributes: TPropertyAttributes; override;
  procedure GetValues(Proc: TGetStrProc); override;
  function GetValue: string; override;
  procedure SetValue(const Value: string); override;
end;

  ...

type
  TBaudRate = (br110, br300, br600, br1200, br2400, br4800, br9600,
               br14400, br19200, br38400, br56000, br128000, br256000);
const
  BaudList: Array[TBaudRate] of string[7] =
                ('110', '300', '600', '1200', '2400', '4800', '9600',
                 '14400', '19200', '38400', '56000', '128000', '256000');

{ TbaudRateProperty }
function TBaudRateProperty.GetAttributes: TPropertyAttributes;
begin
  Result := [paValueList];
end;

procedure TBaudRateProperty.GetValues(Proc: TGetStrProc);
var
  i: TBaudRate;
begin
  for I := Low(TBaudRate) to High(TBaudRate) do Proc(BaudList[i]);
end;

function TBaudRateProperty.GetValue: string;
begin
  Result := BaudList[TBaudRate(GetOrdValue)];
end;
```

```
procedure TBaudRateProperty.SetValue(const Value: string);
var
  i: TBaudRate;
begin
  for I := Low(TBaudRate) to High(TBaudRate) do
    if BaudList[i]= Value then begin
      SetOrdValue(integer(i));
      exit;
    end;
  inherited SetValue(Value);
end;
```

[News Group]

Константа из другого модуля дает неверное значение

Константа из другого модуля дает неверное значение. Как устранить ошибку?

Запись, содержащая признак ошибки:

```
unit Main;
interface
uses VData;

const
  Wko = 0.9;
  ...
unit VData;
  ...

implementation

uses Main;

procedure ...;
begin
  { здесь Wko =...E+230 - наверное, бесконечность }
end;
```

Похоже, это действительно ошибка, причем **особо опасная**, т. к. может исказить результаты расчетов, не вызвав заметных нарушений работы программы.

Эксперимент показал, что любая вещественная константа, определенная в интерфейсе модуля, может быть неверно (или не совсем верно – например, вместо 0.7 может появиться 0.115) прочитана в другом модуле.

Ошибка особенно опасна тем, что она неустойчива и может пропадать и возникать без видимых причин. Например, возникнуть, если предыдущая компиляция была неудачной, и исчезнуть после использования константы в модуле, где она определена.

Устраняется такая ошибка указанием типа данных:

```
const
  Wko: double = 0.9;
```

Правда, теперь это уже не совсем константа...

[Nomadic]

Заголовок файла TGA

Описание заголовка файла для изображения формата Targa		
Смещение	**Длина (в байтах)**	**Описание**
0	1	Длина ID-поля (ID Field Length)
1	1	Тип цветовой карты (Color-map Type)
2	1	Тип изображения (Image Type)
Информация о специфике цветовой карты (Color-map-specific Info)		
3	2	Первое включение цветовой карты (First Color-map Entry)
5	2	Длина цветовой карты (Color-map Length)
7	1	Размер цветовой карты (Color-map Entry Size)
Информация о специфике изображения (Image-specific Info)		
8	2	Горизонтальная координата начала изображения (Image X Origin)
10	2	Вертикальная координата начала изображения (Image Y Origin)
12	2	Ширина изображения (Image Width)
14	2	Высота изображения (Image Height)
16	1	Бит на пиксел (Bits-Per-Pixel)
17	1	Биты дескриптора изображения (Image-Descriptor Bits)

Для изображений с разрешением True Color значение типа цветовой карты должно равняться нулю, в остальных случаях – единице. В случае, если цветовая карта присутствует, ее размер должен соответствовать значению 15, 16, 24 или 32. Для 15 и 16 каждая цветовая карта при загрузке использует 2 байта в формате:

```
Верхний байт      Нижний байт
A RRRRR GG          GGG BBBBB
```

где бит A устанавливается в значение 0 для 15-битных цветовых величин. 24-битный размер карты хранится как три байта в следующем порядке: (B)lue (синий), (G)reen (зеленый), и (R)ed (красный). 32-битный размер цветовой карты сответствует четырем байтам в следующем порядке: (B)lue (синий), (G)reen (зеленый), (R)ed (красный) и значение атрибута – (A)ttribute.

Наконец, код, хранящий тип изображения (Image Type), должен содержать одно из следующих значений:

```
Код                Описание
 0          Изображение отсутствует
 1          Цветовая карта, без компрессии
 2          True-color, без компрессии
 3          Черно-белое, без компрессии
 9          Цветовая карта, RLE-компрессия
10          True-color, RLE-компрессия
11          Черно-белое, RLE-компрессия
```

Горизонтальная и вертикальная координаты начала изображения (Image X & Y Origins), а также размеры изображения (Image Width & Height) разъяснений не требуют. Бит на пиксел (Bits-Per-Pixel) обозначает количество бит, содержащихся в точке изображения, и может быть равен значениям 8, 16, 24 и 32.

Дескриптор изображения (The Image Descriptor bytes) включает несколько битовых полей, которые содержат следующую информацию:

```
Биты                Описание
 0-3        Биты атрибутов (описаны ниже)
  4         Ориентация Слева-на-Право 0=Л/П 1=П/Л
  5         Ориентация Верх/Низ 0=Н/В 1=В/Н
 6-7        Чередование линий 00Н=Нет, 40Н=2 линии, 80Н=4 линии
```

Биты атрибутов служат для определения атрибутов цветов в цветовой карте или true-color пикселах:

```
Биты                Описание
  0         alpha-данные (alpha-канал) отсутствуют
  1         игнорирование, или не определено
```

```
Биты            Описание
  2       не определено, но должно быть сохранено
  3       наличие alpha-данных
  4       информация о пикселе уже была умножена на alpha-величину.
```

Файлы версии Targa 2.0 также имеют файловый колонтитул, который может содержать дополнительное изображение или комментарии. Эти файлы всегда заканчиваются строкой-терминатором 'TRUEVISION-TARGA.'. Так, если Targa-изображение заканчивается значением 'TRUEVISION-TARGA.' + 00H, то можно извлечь восемь байт до строки, чтобы найти начало расширенной области и месторасположение каталога сборки данного файла. Обычно файловый колонтитул версии 2.0 имеет следующий формат:

Байт	Длина	Описание
0	4	32-битное смещение расширенной области
4	4	32-битное смещение каталога сборки
8	17	TRUEVISION-TARGA
25	1	Двоичный ноль ($0)

Приведено описание «почтовой марки», которая может содержаться в формате Targa V2.0. Данная «марка»-пиктограмма должна иметь размеры 64×64 пиксела, представляет собой уменьшенный образ изображения, может включаться в файл по желанию компоновщика и не является обязательной. Область расширения:

Смещение	Длина	Описание
0	2	Размер области расширения (должна быть 495)
2	41	Имя автора
43	81	Авторские комментарии
124	81	Авторские комментарии
205	81	Авторские комментарии
286	81	Авторские комментарии
367	2	Месяц создания
369	2	День создания
371	2	Год создания

продолжение

Смещение	Длина	Описание
…	…	…
482	4	Смещение в файле таблицы цветовой коррекции
486	4	Смещение в файле изображения «почтовой марки»
490	4	Смещение в файле таблицы чередования линий
494	1	Байты атрибутов

Данная «почтовая марка», при ее наличии, может использоваться непосредственно. Она хранится в несжатом виде в том же цветовом формате (цветовой карте или True Color), что и исходное изображение.

Создание палитры

Решение

```
  ...
var
  Form1: TForm1;
  BlueVal: Byte;
  BluePalette: HPalette;

implementation
{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
var
  LogicalPalette: PLogPalette;
  ColorIndex: LongInt;
begin
  GetMem(LogicalPalette, (SizeOf(TLogPalette) + SizeOf(TPaletteEntry) * 256));
  GetSystemPaletteEntries(Canvas.Handle, 0, 256, LogicalPalette^.palPalEntry[0]);
  with LogicalPalette^ do begin
    palVersion := $300;
    palNumEntries := 256;
{$R-}
    for ColorIndex := 10 to 245 do
      with palPalEntry[ColorIndex] do begin
        peRed := 0;
        peGreen := 0;
        peBlue := 255 - (ColorIndex - 10);
        peFlags := PC_NOCOLLAPSE;
      end;
  end;
```

```
{$R+}
  BluePalette := CreatePalette(LogicalPalette^);
  FreeMem(LogicalPalette, (SizeOf(TLogPalette) + SizeOf(TPaletteEntry) * 256));
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  DeleteObject(BluePalette);
end;

procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
                               Shift: TShiftState; X, Y: Integer);
var
  OldPal: HPALETTE;
begin
  OldPal := SelectPalette(Canvas.Handle, BluePalette, False);
  RealizePalette(Canvas.Handle);
  Canvas.Pen.Color := $02000000 or (BlueVal * $00100000);
  Canvas.Pen.Width := 10;
  Canvas.MoveTo(0, 0);
  Canvas.LineTo(X,Y);
  SelectPalette(Canvas.Handle, OldPal, False);
  Inc(BlueVal);
  if BlueVal > 255 then BlueVal := 0;
end;
```

Изменение цветовой палитры изображения

Мне необходимо изменить цветовую палитру изображения с помощью `SetBitmapBits`. *Как это сделать?*

Использование `SetBitmapBits` – не самый подходящий вариант, поскольку данная процедура имеет дело с `HBitmap`, в котором формат пиксела не определен. Несомненно, это более безопасная операция, но никаких гарантий по ее выполнению дать невозможно.

Более целесообразно использовать функции DIB API. Далее представлен программный код, позволяющий изменить таблицу цветов. Создайте метод с такими же параметрами, как у `TFiddleProc`, и измените `ColorTable`, передаваемую как параметр. Затем вызовите процедуру `FiddleBitmap`, передающую `TBitmap` и ваш метод `fiddle`, например так:

```
FiddleBitmap(MyBitmap, Fiddler);
```

Предлагаемый код:

```
type
  TFiddleProc = procedure(var ColorTable: TColorTable) of object;
const
  LogPaletteSize = SizeOf(TLogPalette) + SizeOf(TPaletteEntry) * 255;

 ...
```

```
function PaletteFromDIB(BitmapInfo: PBitmapInfo): HPalette;
var
  LogPalette: PLogPalette;
  i: integer;
  Temp: byte;
begin
  with BitmapInfo^, bmiHeader do begin
    GetMem(LogPalette, LogPaletteSize);
    try
      with LogPalette^ do begin
        palVersion := $300;
        palNumEntries := 256;
        Move(bmiColors, palPalEntry, sizeof(TRGBQuad) * 256);
        for i := 0 to 255 do
          with palPalEntry[i] do begin
            Temp := peBlue;
            peBlue := peRed;
            peRed := Temp;
            peFlags := PC_NOCOLLAPSE;
          end;
{ создаем палитру }
        Result := CreatePalette(LogPalette^);
      end;
    finally
      FreeMem(LogPalette, LogPaletteSize);
    end;
  end;
end;

{ Следующая процедура на основе изображения создает DIB, изменяет ее таблицу
  цветов, создавая тем самым новую палитру, после чего передает ее обратно
  изображению. При этом используется метод косвенного вызова, с помощью которого
  изменяется палитра цветов - ей передается array[ 0..255 ] of TRGBQuad. }
procedure FiddleBitmap(Bitmap: TBitmap; FiddleProc: TFiddleProc);
const
  BitmapInfoSize = SizeOf(TBitmapInfo) + SizeOf(TRGBQuad) * 255;
var
  BitmapInfo: PBitmapInfo;
  Pixels: pointer;
  InfoSize: integer;
  ADC: HDC;
  OldPalette: HPalette;
begin
{ получаем DIB }
  GetMem(BitmapInfo, BitmapInfoSize);
  try
{ меняем таблицу цветов - ПРИМЕЧАНИЕ: она использует 256 цветов DIB }
    FillChar(BitmapInfo^, BitmapInfoSize, 0);
    with BitmapInfo^.bmiHeader do begin
      biSize := SizeOf(TBitmapInfoHeader);
      biWidth := Bitmap.Width;
```

```
      biHeight := Bitmap.Height;
      biPlanes := 1;
      biBitCount := 8;
      biCompression := BI_RGB;
      biClrUsed := 256;
      biClrImportant := 256;
      GetDIBSizes(Bitmap.Handle, InfoSize, biSizeImage);
{ распределяем место для пикселов }
      Pixels := GlobalAllocPtr(GMEM_MOVEABLE, biSizeImage);
      try
{ получаем пикселы DIB }
        ADC := GetDC(0);
        try
          OldPalette := SelectPalette(ADC, Bitmap.Palette, False);
          try
            RealizePalette(ADC);
            GetDIBits(ADC, Bitmap.Handle, 0, biHeight, Pixels, BitmapInfo^,
                      DIB_RGB_COLORS);
          finally
            SelectPalette(ADC, OldPalette, True);
          end;
        finally
          ReleaseDC(0, ADC);
        end;
{ теперь изменяем таблицу цветов }
        FiddleProc(PColorTable(@BitmapInfo^.bmiColors)^);
{ создаем палитру на основе новой таблицы цветов }
        Bitmap.Palette := PaletteFromDIB(BitmapInfo);
        OldPalette := SelectPalette(Bitmap.Canvas.Handle, Bitmap.Palette, False);
        try
          RealizePalette(Bitmap.Canvas.Handle);
          StretchDIBits(Bitmap.Canvas.Handle, 0, 0, biWidth, biHeight,
                        0, 0, biWidth, biHeight, Pixels, BitmapInfo^,
                        DIB_RGB_COLORS, SRCCOPY);
        finally
          SelectPalette(Bitmap.Canvas.Handle, OldPalette, True);
        end;
      finally
        GlobalFreePtr(Pixels);
      end;
    end;
  finally
    FreeMem(BitmapInfo, BitmapInfoSize);
  end;
end;
{ Пример метода "fiddle" }
procedure TForm1.Fiddler(var ColorTable: TColorTable);
var
  i: integer;
begin
  for i := 0 to 255 do
```

```
    with ColorTable[i] do begin
      rgbRed := rgbRed * 9 div 10;
      rgbGreen := rgbGreen * 9 div 10;
      rgbBlue := rgbBlue * 9 div 10;
    end;
end;
```

[News Group]

Функция для работы с палитрами RGB

Каким образом осуществляются операции в Delphi над палитрой? Предположим, имеется 4 открытых формы, которые должны использовать цвета, не входящие в стандартный набор из 20 именованных цветов. К ячейкам таблицы необходимо применить эти нестандартные цвета. Существует ли способ обновления системной палитры таким образом, чтобы все формы использовали одни и те же цвета?

При работе с палитрами рекомендуется применять функцию RGB. Если вы с ее помощью изменяете свойство Color, Windows довольно хорошо справляется с задачей подбора цветов для низкого разрешения, а в системах с высоким разрешением вы получите точный цвет RGB. Это могло бы послужить решением поставленного вопроса. Вот пример формы, которая содержит все оттенки от красного до синего:

```
procedure TForm1.FormClick(Sender: TObject);
var
  blue: Byte;
begin
  for blue := 0 to 255 do begin
    Color := RGB(255 - blue, 0, blue);
    update;
  end;
end;
```

Создание и использование 256-цветной палитры

Пример создания и применения палитры для 256-цветных изображений. Вам нужны API-функции SelectPalette или RealizePalette, в зависимости от того, как вы предполагаете использовать изображение.

```
procedure TForm1.MakePalette(forBitMap: TBitMap);
var
  pNewPal: PLogPalette;
  lSize: LongInt;
  nCntr: Byte;

begin
  lSize := SizeOf(TLogPalette) + SizeOf(TPaletteEntry) * 256;
  try
    GetMem(pNewPal, lSize);
    pNewPal^.palNumEntries := 256;
    pNewPal^.palVersion := $300;
```

```
{$R-}{ выключаем контроль допустимого диапазона }
{ создаем данные палитры... }
    for nCntr := 0 to 254 do begin
      pNewPal^.palPalEntry[nCntr].peRed := nCntr + 20;
      pNewPal^.palPalEntry[nCntr].peGreen := nCntr + 20;
      pNewPal^.palPalEntry[nCntr].peBlue := nCntr + 20;
      pNewPal^.palPalEntry[nCntr].peFlags := pc_nocollapse;
    end;
{$R+}{ включаем контроль допустимого диапазона }
{ удаляем старый hPal; предохраняемся от утечки памяти }
    DeleteObject(hPal);
{ создаем новую палитру на основе новых значений }
    hPal := CreatePalette(pNewPal^);
{ назначаем новую палитру }
    forBitMap.Palette := hPal;
  finally
    FreeMem(pNewPal, lSize);
  end;
end;
```

[News Group]

Загрузка 256-цветного Bitmap

Каким способом создать в памяти изображение, чтобы `TBitmap.LoadFromStream` *мог распознать его?*

Предлагаемый метод загружает и размещает «сырой» ресурс изображения, используя информационный заголовок файла изображения. Вот потомок `TBitmap`, инкапсулирующий сказанное:

```
type
  TMyBitmap = class(TBitmap)
  public
    procedure Load256ColorBitmap(Instance: THandle; BitmapName: PChar);
end;

procedure TMyBitmap.Load256ColorBitmap(Instance: THandle; BitmapName: PChar);

var
  HDib: THandle;
  Size: LongInt;
  Info: PBitmapInfo;
  FileHeader: TBitmapFileHeader;
  S: TMemoryStream;

begin
  HDib := LoadResource(Instance, FindResource(Instance, BitmapName, RT_BITMAP));
  if HDib <> 0 then begin
    Info := LockResource(HDib);
    Size := GetSelectorLimit(Seg(Info^)) + SizeOf(TBitmapFileHeader);
    with FileHeader do begin
```

```
        bfType := $4D42;
        bfSize := Size;
        bfReserved1 := 0;
        bfReserved2 := 0;
        bfOffBits := SizeOf(TBitmapFileHeader) + SizeOf(TBitmapInfoHeader);
        case Info^.bmiHeader.biBitCount of
            1:  bfOffBits := bfOffBits + 2 * 4;
            4:  bfOffBits := bfOffBits + 16 * 4;
            8:  bfOffBits := bfOffBits + 256 * 4;
        end;
      end;
      S := TMemoryStream.Create;
      try
        S.SetSize(Size);
        S.Write(FileHeader, SizeOf(TBitmapFileHeader));
        S.Write(Info^, Size - SizeOf(TBitmapFileHeader));
        S.Position := 0;
        LoadFromStream(S);
      finally
        S.Free;
        FreeResource(HDib);
      end;
    end else
      raise EResNotFound.Create(Format('Не могу найти ресурс изображения %s',
                                [BitmapName]));
end;
```

А вот как это можно использовать:

```
Image1.Picture.Bitmap := TMyBitmap.Create;
TMyBitmap(Image1.Picture.Bitmap).Load256ColorBitmap(hInstance, 'BITMAP_1');
```

[News Group]

Примечание

Код создавался для Delphi версии 1, поэтому читателям необходимо будет адаптировать его под конкретное применение в Win32 путем замены соответствующих функций.

Захват изображений

Рассмотрим пример кода, позволяющего с помощью TBitmap захватить часть изображения и сохранить его в файле. В нем применяется функция копирования палитры, необходимой при работе в режиме 256 цветов.

```
function CopyPalette(Palette: HPalette): HPalette;
var
  nEntries: integer;
  LogPalSize: integer;
  LogPalette: PLogPalette;
begin
  Result := 0;
```

```
  if Palette = 0 then Exit;
  GetObject(Palette, SizeOf(nEntries), @nEntries);
  if nEntries < 1 then Exit;
  LogPalSize := SizeOf(TLogPalette) + SizeOf(TPaletteEntry) * (nEntries - 1);
  GetMem(LogPalette, LogPalSize);
  with LogPalette^ do
    try
      palVersion := $300;
      palNumEntries := nEntries;
      GetPaletteEntries(Palette, 0, nEntries, palPalEntry[0]);
      Result := CreatePalette(LogPalette^);
    finally
      FreeMem(LogPalette, LogPalSize);
    end;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  Bitmap: TBitmap;
begin
  Bitmap := TBitmap.Create;
  try
    Bitmap.Width := 50;
    Bitmap.Height := 40;
    Bitmap.Palette := CopyPalette(Image1.Picture.Bitmap.Palette);
    Bitmap.Canvas.CopyRect(Rect(0, 0, 50, 40), Image1.Picture.Bitmap.Canvas,
                           Bounds(20, 10, 50, 40));
    Bitmap.SaveToFile('c:\windows\temp\junk.bmp');

  finally
    Bitmap.Free;
  end;
end;
```

[News Group]

Примечание

Поместите на форму компонент `Image` *(*`TImage1`*), загрузите в него рисунок. Результат программы будет лежать в каталоге* c:\windows\temp *в файле с именем* junk.bmp.

Bitmap без формы

Каким образом загрузить изображение (BMP) и отобразить его на рабочем столе без использования формы? (Отображать необходимо из DLL).

Существует один способ решения поставленной задачи: создать холст `TCanvas`, получить контекст устройства для рабочего стола и назначить его дескриптору холста. После рисования на холсте ваше творение будет отображено на рабочем столе. Например:

```
  ...

var
  DesktopCanvas: TCanvas;

begin
  DesktopCanvas := TCanvas.Create;
  try
    DesktopCanvas.Handle := GetDC(0);
    try
      DesktopCanvas.MoveTo(0, 0);
      DesktopCanvas.LineTo(Screen.Width, Screen.Height);
    finally
      ReleaseDC(0, DesktopCanvas.Handle);
      DesktopCanvas.Handle := 0;
    end;
  finally
    DesktopCanvas.Free;
  end;
end;
```

Можно создать TBitmap и загрузить в него файл .BMP. Единственная неприятность может произойти, если используется изображение с 256-цветной палитрой при работе в режиме с 256 цветами. Обойти это препятствие можно так: создать форму без границ и заголовка, установить ее высоту и ширину в ноль, поместить на нее компонент TImage и загрузить в него необходимое изображение. VCL создаст для вас нужную палитру.

[News Group]

Рисование без мерцания

Почему изображение мерцает, если я вызываю метод Repaint *или* Refresh*, а не* OnPaint *напрямую? Или это просто «вариация на тему»?*

Имеются две фазы обновления окна. В первой фазе, при выводе окна, Windows посылает ему сообщение WM_ERASEBKGND, оповещающее о необходимости стирания фона перед процедурой рисования. Затем посылается сообщение WM_PAINT, служащее сигналом для закрашивания «переднего плана».

Тем не менее, вы можете пропустить первую фазу, которая вызывает мерцание, одним из двух способов. Первый заключается в том, что вы форсируете обновление сами, с помощью вызова функции Windows API InvalidateRect. На входе он получает дескриптор окна, указатель на закрашиваемую область (передаем nil, если надо отрисовать всю область окна) и третий параметр, сообщающий о необходимости очистки фона. Вот как раз последний параметр и должен содержать значение False, если вы сами будете в методе Paint полностью отрисовывать всю область:

```
InvalidateRect(Handle, Nil, False);
```

Handle должен быть дескриптором формы или элемента управления.

Второй способ избежать мерцания заключается в использовании функций VCL. Можно указать VCL не стирать фон, добавляя [csOpaque] к значению свойства ControlStyle, как показано ниже:

```
ControlStyle := ControlStyle + [csOpaque];
```

Это ограничивает заполнение заднего фона, но вы все еще можете видеть процесс «наполнения» области изображением, т. е. сам процесс рисования. В этом случае от эффекта «мигания» можно избавиться, рисуя на TBitmap и выводя его затем на экран командой CopyRect.

[News Group]

Растягивание пиктограммы

Для вывода объекта TIcon *я вызываю функцию* StretchDraw*, но не могу изменить оригинальный размер, вне зависимости от заданной области* TRect.

StretchDraw не работает с пиктограммами. В данной ситуации лучше нарисовать пиктограмму в TImage и затем назначить изображение другому, большему TImage. Пример программного кода:

```
procedure TForm1.StringGrid1Click(Sender: TObject);
begin
  Image1.Canvas.FillRect(Image1.Canvas.ClipRect);
  Image1.Canvas.Draw(0, 0, TIcon(StringGrid1.Objects[StringGrid1.Col,
                     StringGrid1.Row]));
  Form2.Image1.Picture := Image1.Picture;
end;
```

Тень в заданной области

Как быстро наложить тень на заданную область?

Решение

```
procedure TForm1.DrawShadows(WDepth, HDepth: Integer);
var
  Dst, RgnBox: TRect;
  hOldDC: HDC;
  OffScreen, Pattern: TBitmap;
  Bits: array[0..7] of Word;
begin
  Bits[0] := $0055;
  Bits[1] := $00aa;
  Bits[2] := $0055;
  Bits[3] := $00aa;
  Bits[4] := $0055;
  Bits[5] := $00aa;
  Bits[6] := $0055;
  Bits[7] := $00aa;
  hOldDC := Canvas.Handle;
```

```
    Canvas.Handle := GetWindowDC(Form1.Handle);
    OffsetRgn(ShadeRgn, WDepth, HDepth);
    GetRgnBox(ShadeRgn, RgnBox);
    Pattern := TBitmap.Create;
    Pattern.ReleaseHandle;
    Pattern.Handle := CreateBitmap(8, 8, 1, 1, @(Bits[0]));
    Canvas.Brush.Bitmap := Pattern;
    OffScreen := TBitmap.Create;
    OffScreen.Width := RgnBox.Right-RgnBox.Left;
    OffScreen.Height := RgnBox.Bottom-RgnBox.Top;
    Dst := Rect(0, 0, OffScreen.Width, OffScreen.Height);
    OffsetRgn(ShadeRgn, 0, -RgnBox.Top);
    FillRgn(OffScreen.Canvas.Handle, ShadeRgn, Canvas.Brush.Handle);
    OffsetRgn(ShadeRgn, 0, RgnBox.Top);
  // BitBlt работает быстрее CopyRect
    BitBlt(OffScreen.Canvas.Handle, 0, 0, OffScreen.Width, OffScreen.Height,
           Canvas.Handle, RgnBox.Left, RgnBox.Top, SRCAND);
    Canvas.Brush.Color := clBlack;
    FillRgn(Canvas.Handle, ShadeRgn, Canvas.Brush.Handle);
    BitBlt(Canvas.Handle, RgnBox.Left, RgnBox.Top, OffScreen.Width,
           OffScreen.Height, OffScreen.Canvas.Handle, 0, 0, SRCPAINT);
    OffScreen.Free;
    Pattern.Free;
    OffsetRgn(ShadeRgn, -WDepth, -HDepth);
    ReleaseDC(Form1.Handle, Canvas.Handle);
    Canvas.Handle := hOldDC;
  end;
```

Функция рисует сложную тень на форме `Form1`. Для определения формы тени используется регион `ShadeRgn`, который был создан раньше.

Создание тени у метки

Эффект основан на коде модуля `gblur2` из предыдущего примера.

Данный метод позволяет создавать тень у текстовых меток `TLabel`. Создание тени присходит в фоновом режиме, во время «простоя» процессора.

Решение

```
ShowFade(CaptionLabel);
```

или

```
ShowFadeWithParam(CaptionLabel, 3, 3, 2, clGray);
```

Модуль BLUR.PAS

```
unit blur;

interface

uses
  Classes, Graphics, StdCtrls, gblur2;
```

```
const
  add_width = 4;
  add_height = 5;

type
  TBlurThread = class(TThread)
  private
    text_position: Integer;
    FadeLabel: TLabel;
    Temp_Bitmap: TBitmap;
    procedure ShowBlur;
    procedure SetSize;

  protected
    F_width: Integer;
    F_X: Integer;
    F_Y: Integer;
    F_color: TColor;
    procedure Execute; override;
  public
    constructor Create(Sender: TLabel; Fade_width: integer; Fade_X: Integer;
                       Fade_Y: Integer; Fade_color: TColor);
    destructor Destroy;
  end;

procedure ShowFade(Sender:TLabel);
procedure ShowFadeWithParam(Sender: TLabel; Fade_width: integer; Fade_X: Integer;
                            Fade_Y: Integer; Fade_color: TColor);

implementation

procedure ShowFadeWithParam(Sender: TLabel; Fade_width: integer; Fade_X: Integer;
                            Fade_Y: Integer; Fade_color: TColor);
var
  SlowThread: TBlurThread;
begin
  SlowThread := TBlurThread.Create(Sender, Fade_width, Fade_X, Fade_Y, Fade_color);
  SlowThread.Priority := tpIdle;
  SlowThread.Resume;
end;

procedure ShowFade;
var
  SlowThread: TBlurThread;
begin
  SlowThread := TBlurThread.Create(Sender, 3, 3, 3, clBlack);
  SlowThread.Priority := tpIdle;
//  SlowThread.Priority := tpLowest;
//  SlowThread.Priority := tpTimeCritical;
  SlowThread.Resume;
end;
```

```
constructor TBlurThread.Create(Sender: TLabel; Fade_width: integer;
                               Fade_X: Integer; Fade_Y: Integer; Fade_color: TColor);
begin
  Temp_Bitmap := TBitmap.Create;
  Temp_Bitmap.Canvas.Font := Sender.Font;
  FadeLabel := Sender;
  F_width := Fade_width;
  F_X := Fade_X;
  F_Y := Fade_Y;
  F_color := Fade_color;
  inherited Create(True);
end;

destructor TBlurThread.Destroy;
begin
  Temp_Bitmap.Free;
  inherited Destroy;
end;

procedure TBlurThread.ShowBlur;
begin
  FadeLabel.Canvas.Draw(text_position + F_X, F_Y, Temp_Bitmap);
  FadeLabel.Canvas.TextOut(text_position, 0, FadeLabel.Caption);
end;

procedure TBlurThread.SetSize;
begin
  if FadeLabel.Width < (Temp_Bitmap.Canvas.TextWidth(FadeLabel.Caption) + F_width
      + F_X {add_width}) then begin
    FadeLabel.Width := Temp_Bitmap.Canvas.TextWidth(FadeLabel.Caption) + F_width
        + F_X {add_width};
    FadeLabel.Tag := 2;
  end else FadeLabel.Tag := 0;
  if FadeLabel.Height < (Temp_Bitmap.Canvas.TextHeight(FadeLabel.Caption) + F_width
      + F_Y {add_height}) then begin
    FadeLabel.Height := Temp_Bitmap.Canvas.TextHeight(FadeLabel.Caption) + F_width
        + F_Y {add_height};
    FadeLabel.Tag := 1;
  end else if FadeLabel.Tag <> 2 then FadeLabel.Tag := 0;
end;
procedure TBlurThread.Execute;
begin
  Synchronize(SetSize);
  if FadeLabel.Tag = 0 then begin
    Temp_Bitmap.Width := FadeLabel.Width;
    Temp_Bitmap.Height := FadeLabel.Height;
    Temp_Bitmap.Canvas.Brush.Color := FadeLabel.Color;
    Temp_Bitmap.Canvas.FillRect(FadeLabel.ClientRect);
    Temp_Bitmap.Canvas.Font.Color := F_color;                  // clBlack
    if FadeLabel.Alignment = taRightJustify then
      text_position := FadeLabel.Width
```

```
                    - Temp_Bitmap.Canvas.TextWidth(FadeLabel.Caption) - F_width
                    - F_X {add_width}
    else if FadeLabel.Alignment = taCenter then
      text_position := (FadeLabel.Width -
                        Temp_Bitmap.Canvas.TextWidth(FadeLabel.Caption) - F_width -
                        F_X{add_width}) div 2
    else text_position := 0;
    Temp_Bitmap.Canvas.TextOut(0, 0, FadeLabel.Caption);
    Temp_Bitmap.PixelFormat := pf24Bit;
    GBlur(Temp_Bitmap, F_width);
//  Temp_Bitmap.SaveToFile('a.bmp');
    Synchronize(ShowBlur);
  end;
end;

end.
```

[Den is Com]

Компонент для отрисовки линий

В предлагаемом модуле используется компонент, инкапсулирующий функции рисования линий. С его помощью можно рисовать горизонтальные, вертикальные и диагональные линии. Он также позволяет добавить необходимые события в секцию published.

```
unit Lines;

interface

uses
  Windows, Messages, Classes, Graphics, Controls, Forms;

type
  TLineOrigin = (loTopLeft, loTopRight);
  TLine = class(TGraphicControl)
  private
    fOrigin: TLineOrigin;
    fPen: TPen;
    procedure SetOrigin(Value: TLineOrigin);
    procedure SetPen(Value: TPen);
  protected
    procedure Paint; override;
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
  published
    property Origin: TLineOrigin read fOrigin write SetOrigin default loTopLeft;
    property Pen: TPen read fPen write SetPen;
    property Height default 33;
    property Width default 33;
```

```
    procedure StyleChanged(Sender: TObject);
  end;

procedure Register;
implementation

procedure Register;
begin
  RegisterComponents('Samples', [TLine]);
end;

procedure TLine.SetOrigin(Value: TLineOrigin);
begin
  if fOrigin <> Value then begin
    fOrigin := Value;
    Invalidate;
  end;
end;

procedure TLine.SetPen(Value: TPen);
begin
  fPen.Assign(Value);
end;

procedure TLine.StyleChanged(Sender: TObject);
begin
  Invalidate;
end;

constructor TLine.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  Height := 33;
  Width := 33;
  fPen := TPen.Create;
  fPen.OnChange := StyleChanged;
  if csOpaque in ControlStyle then ControlStyle := ControlStyle - [csOpaque];
end;
procedure TLine.Paint;
begin
  with Canvas do begin
    Pen := fPen;
    if (Width <= Pen.Width) or (Height <= Pen.Width) then begin
      if Width <= Pen.Width then begin
        MoveTo(0, 0);
        LineTo(0, Height);
      end;
      if Height <= Pen.Width then begin
        MoveTo(0, 0);
        LineTo(Width, 0);
      end;
```

```
    end else
      case fOrigin of
        loTopLeft:  begin
                      MoveTo(0, 0);
                      LineTo(Width, Height);
                    end;
       loTopRight:  begin
                      MoveTo(Width, 0);
                      LineTo(0, Height);
                    end;
    end;
  end;
end;

destructor TLine.Destroy;
begin
  fPen.Free;
  inherited Destroy;
end;
end.
```

[News Group]

Примечание

Для получения вертикальных или горизонтальных линий установите свойство `Width` *или* `Height` *(соответственно) равным* `1`.

Отображение ломаной линии

Каким образом вывести ломаную линию на холсте, если размер массива (количество точек) не известен до момента запуска программы?

Решение заключается в хитрой комбинации функций `PolyLine` и `Polygon` с ассемблерным кодом.

```
type
  PPointArr = ^TPointArr;
  TPointArr = array[0..16380] of TPoint;

var
  I1: integer;
  Elements: word;
  PointArr: PPointArr;

begin
  GetMem(PointArr, (Elements + 1) * SizeOf(TPoint));
  try
    for I1 := 0 to Elements do begin
      PointArr^[I1].X := ReadNextXValue;
      PointArr^[I1].Y := ReadNextYValue;
    end;
```

```
{ Вызов Polygon(PointArr^), но только с Elements+1 элементами в открытом массиве }
    asm
      les   di, PointArr     { Помещаем указатель на PointArr }
      push  es
      push  di
      push  Elements         { Помещаем High(PointArr^) }
      les   di, self         { Помещаем указатель self }
      push  es
      push  di
      les   di, es:[di]      { Вызов self.Polygon }
      call  Polygon
    end;
  finally
    FreeMem(PointArr, (Elements + 1) * SizeOf(TPoint));
  end;
end;
```

Рисование на инструменте управления

Как рисовать на инструменте управления, например, на `TPanel`*?*

У всех компонентов, порожденных от `TCustomControl`, имеется свойство `Canvas` типа `TCanvas`.

Если свойство `Canvas` недоступно, то им можно воспользоваться, создав потомка и осуществив перенос этого свойства в раздел `Public`.

```
type
  TcPanel = class(TPanel)
  public
    property Canvas;
end;
```

Вывод текста на родительском элементе управления

Свойство `Canvas` в `TCustomControl` существует, но оно защищено. Поскольку свойство `Canvas` инкапсулирует Windows HDC (`Canvas.Handle`), можно создавать объект `TCanvas` и назначать через свойство `Handle` контекст устройства элементу управления, на котором вы хотите рисовать.

```
procedure AControl.DrawLabel(ACaption: TCaption);
var
  ACanvas: TCanvas; DC: HDC;
begin
  ACanvas := TCanvas.Create;
  try
    WindowHandle := Parent.Handle;
    DC := GetDeviceContext(WindowHandle);
    ACanvas.Handle := DC;
    with ACanvas do begin
  ...
    end;
```

```
    ACanvas.Handle := 0;
    ReleaseDC(WindowHandle, DC);
  finally
    ACanvas.Free;
  end;
end;
```

Надпись под углом

Как вывести на Canvas *надпись, расположенную под углом?*

Решение 1

```
function CreateRotatedFont(F: TFont; Angle: Integer): hFont;
var
  LF: TLogFont;
begin
  FillChar(LF, SizeOf(LF), #0);
  with LF do begin
    lfHeight := F.Height;
    lfWidth := 0;
    lfEscapement := Angle * 10;
    lfOrientation := 0;
    if fsBold in F.Style then lfWeight := FW_BOLD
    else lfWeight := FW_NORMAL;
    lfItalic := Byte(fsItalic in F.Style);
    lfUnderline := Byte(fsUnderline in F.Style);
    lfStrikeOut := Byte(fsStrikeOut in F.Style);
    lfCharSet := DEFAULT_CHARSET;
    StrPCopy(lfFaceName, F.Name);
    lfQuality := DEFAULT_QUALITY;
    lfOutPrecision := OUT_DEFAULT_PRECIS;
    lfClipPrecision := CLIP_DEFAULT_PRECIS;
    case F.Pitch of
      fpVariable:  lfPitchAndFamily := VARIABLE_PITCH;
         fpFixed:  lfPitchAndFamily := FIXED_PITCH;
    else
                   lfPitchAndFamily := DEFAULT_PITCH;
    end;
  end;
  Result := CreateFontIndirect(LF);
end;

 ...
if FontAngle < 0 then Canvas.Font.Handle := CreateRotatedFont(Font, FontAngle);
 ...
```

Данный код обеспечивает вращение только векторных шрифтов.

Решение 2

```
procedure MyRotateText(CV: TCanvas; sText:
String; X, Y, Angle: Integer);
var
  LogFont: TLogFont;
```

```
begin
  GetObject(CV.Font.Handle, SizeOf(TLogFont), @LogFont);
  LogFont.lfEscapement := Angle * 10;
  CV.Font.Handle := CreateFontIndirect(LogFont);
  CV.TextOut(X, Y, sText);
end;
```

Сохранение и восстановление шрифта

После назначения свойству `Handle` *нового шрифта старый уничтожается. Для восстановления оригинальных установок необходимо восстановить предыдущий шрифт. Есть какое-либо простое решение для сохранения исходного шрифта?*

Решение

```
var
  SaveFont: TFont;
 ...
  SaveFont := TFont.Create;
  SaveFont.Assign (CV.Font);
  CV.Font := CreateFontIndirect(…)
(...)
  CV.Font.Assign(SaveFont);
  SaveFont.Free;
```

[Fyodorov Oleg]

«Прозрачный» текст

Предлагаемый модуль реализует алгоритм «затухания» текста на холсте и обратного эффекта.

```
function TFadeEffect.FadeInText(Target: TCanvas; X, Y: integer;
                                FText: String): TRect;
var
  Pic: TBitmap;
  W, H: integer;
  PicRect, TarRect: TRect;
begin
  Pic := TBitmap.Create;
  Pic.Canvas.Font := Target.Font;
  W := Pic.Canvas.TextWidth(FText);
  H := Pic.Canvas.TextHeight(FText);
  Pic.Width := W;
  Pic.Height := H;
  PicRect := Rect(0, 0, W, H);
  TarRect := Rect(X, Y, X + W, Y + H);
  Pic.Canvas.CopyRect(PicRect, Target, TarRect);
  SetBkMode(Pic.Canvas.Handle, Transparent);
  Pic.Canvas.TextOut(0, 0, FText);
  FadeInto(Target, X, Y, Pic);
  Pic.Free;
```

```
  FadeInText := TarRect;
end;

procedure TFadeEffect.FadeOutText(Target: TCanvas; TarRect: TRect; Orig: TBitmap);
var
  Pic: TBitmap;
  PicRect: TRect;
begin
  Pic := TBitmap.Create;
  Pic.Width := TarRect.Right - TarRect.Left;
  Pic.Height := TarRect.Bottom - TarRect.Top;
  PicRect := Rect(0, 0, Pic.Width, Pic.Height);
  Pic.Canvas.CopyRect(PicRect, Orig.Canvas, TarRect);
  FadeInto(Target, TarRect.Left, TarRect.Top, Pic);
  Pic.Free;
end;
```

Вывод текста на экран с обрезанием по длине

Как вывести на экран текст с «красивым» усечением по длине (если текст не помещается на экране)?

Используйте вызов DrawTextEx, установив в параметре dwDTFormat значение DT_PATH_ELLIPSIS.

Создание DIB из BMP

Файл хранится в формате BMP. *Как преобразовать его в* DIB *и отобразить?*

Это не тривиально, но можно прибегнуть к функциям GetDIBSizes и GetDIB из модуля GRAPHICS.PAS. Рассмотрим две процедуры: одну для создания DIB из Tbitmap, а вторую для освобождения памяти:

```
procedure BitmapToDIB(Bitmap: TBitmap; var BitmapInfo: PBitmapInfo;
                      var InfoSize: DWORD; var Bits: pointer; var BitsSize: DWORD);
begin
  BitmapInfo := NIL;
  InfoSize := 0;
  Bits := NIL;
  BitsSize := 0;
  if not Bitmap.Empty then
    try
      GetDIBSizes(Bitmap.Handle, InfoSize, BitsSize);
      GetMem(BitmapInfo, InfoSize);
      Bits := GlobalAllocPtr(GMEM_MOVEABLE, BitsSize);
      if Bits = NIL then
        Raise EOutOfMemory.Create('Не хватает памяти для пикселов изображения');
      if not GetDIB(Bitmap.Handle, Bitmap.Palette, BitmapInfo^, Bits^) then
        Raise Exception.Create('Не могу создать DIB');
    except
      if BitmapInfo <> NIL then FreeMem(BitmapInfo, InfoSize);
      if Bits <> NIL then GlobalFreePtr(Bits);
```

```
      BitmapInfo := NIL;
      Bits := NIL;
      Raise;
    end;
end;

{ используйте FreeDIB для освобождения информации об изображении
  и битовых указателей }

procedure FreeDIB(BitmapInfo: PBitmapInfo; InfoSize: integer; Bits: pointer;
                  BitsSize: longint);
begin
  if BitmapInfo <> NIL then FreeMem(BitmapInfo, InfoSize);
  if Bits <> NIL then GlobalFreePtr(Bits);
end;
```

Создаем форму с Image и загружаем в него 256-цветное изображение, затем рядом размещаем TPaintBox. Добавляем следующие строчки к объявлениям private вашей формы:

```
  BitmapInfo: PBitmapInfo;
  InfoSize: DWORD;
  Bits: pointer;
  BitsSize: DWORD;
```

Создаем нижеприведенные обработчики событий, которые демонстрируют процесс отрисовки DIB:

```
procedure TForm1.FormCreate(Sender: TObject);

begin
  BitmapToDIB(Image1.Picture.Bitmap, BitmapInfo, InfoSize, Bits, BitsSize);
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  FreeDIB(BitmapInfo, InfoSize, Bits, BitsSize);
end;

procedure TForm1.PaintBox1Paint(Sender: TObject);
var
  OldPalette: HPalette ;
begin
  if Assigned(BitmapInfo) and Assigned(Bits) then
    with BitmapInfo^.bmiHeader, PaintBox1.Canvas do begin
      OldPalette := SelectPalette(Handle, Image1.Picture.Bitmap.Palette, False);
      try
        RealizePalette(Handle);
        StretchDIBits(Handle, 0, 0, PaintBox1.Width, PaintBox1.Height,
                      0, 0, biWidth, biHeight, Bits, BitmapInfo^, DIB_RGB_COLORS,
                      SRCCOPY);
```

```
        finally
          SelectPalette(Handle, OldPalette, True);
        end;
      end;
end;
```

Это поможет вам сделать первый шаг. С помощью этих способов вы можете создать собственный HPalette на основе DIB вместо использования TBitmap и своей палитры. Функция с именем PaletteFromW3DIB из GRAPHICS.PAS как раз этим и занимается, но она не объявлена как экспортируемая, поэтому для ее применения необходимо скопировать и вставить ее исходный код в свой модуль.

[News Group]

Двоичный файл с набором изображений

Как сохранить множество изображений в единственном бинарном файле?

В данном примере ваша запись помещается в объект. Хотя такая процедура и не является неотъемлемой частью алгоритма, но рано или поздно это придется сделать. В качестве средства для чтения и записи он использует потоки. Для тех, кто уже работал с потоками, данная технология не будет открытием. Одно из преимуществ потока в том, что для работы с графическими объектами – Bitmap, Icon, MetaFile – можно использовать методы SaveToStream и LoadFromStream.

С LoadFromStream связана одна проблема. При вызове Graphic.LoadFromStream, графика «оставляла» позицию потока не в конце записи, а в конце самого потока. Другими словами, если графический объект первый раз записывал себя в поток, данные заканчивались в позиции 247. Но когда графический объект «читал себя», он не останавливался в позиции 247, а читал себя из всего потока. Поэтому мог быть прочитан только один объект.

Решение проблемы заключается в установке позиции, на которой действительно заканчивается запись. Затем, после того как графический объект «прочтет себя» из потока, снова перемещаем позицию, готовя тем самым поток для чтения следующего объекта. Эти детали делают реализацию объекта чуть сложнее.

Дополнительно включена возможность обрабатывать: пиктограммы, метафайлы, а также простые изображения.

```
unit BIN;
interface

uses
  Graphics, Classes;

type
  TAlbumRec = class
```

```
  private
    FGraphic: TGraphic;
    FDescription: string;             { пример поля }
    FItemType: ShortInt;              { пример поля }
    procedure SetGraphic(AGraphic: TGraphic);

  public
    constructor Create;
    destructor Destroy; override;
    procedure LoadFromStream(Stream: TStream);
    procedure SaveToStream(Stream: TStream);
    property Graphic: TGraphic read FGraphic write SetGraphic;
    property Description: string read FDescription write FDescription;
    property ItemType: ShortInt read FItemType write FItemType;
end;

implementation

constructor TAlbumRec.Create;
begin
  inherited Create;
end;

destructor TAlbumRec.Destroy;
begin
  FGraphic.Free;
  inherited Destroy;
end;

procedure TAlbumRec.LoadFromStream(Stream: TStream);

var
  GraphicTypeCode: Char;
  EndPosition: LongInt;

begin
{ Считываем в потоке позицию, в которой заканчивается запись... }
  Stream.Read(EndPosition, SizeOf(EndPosition));
{ Считываем строку... }
  Stream.Read(FDescription[0], SizeOf(Byte));
  Stream.Read(FDescription[1], Byte(FDescription[0]));
{ Читаем целое... }
  Stream.Read(FItemType, SizeOf(FItemType));
{ Считываем код, сообщающий тип графического объекта, который необходимо
  создать ... }
  Stream.Read(GraphicTypeCode, SizeOf(GraphicTypeCode));
{ Освобождаем текущий графический объект и пересоздаем его.. }
  FGraphic.Free;
  FGraphic := nil;
  case GraphicTypeCode of
      'B':  FGraphic := TBitmap.Create;
```

```
      'I':  FGraphic := TIcon.Create;
      'M':  FGraphic := TMetafile.Create;
  end;

{ Загружаем из потока графику... }
  if FGraphic <> nil then FGraphic.LoadFromStream(Stream);
{ Ищем в потоке конечную позицию для данной записи. }
  Stream.Seek(EndPosition, 0);
end;

procedure TAlbumRec.SaveToStream(Stream: TStream);
var
  GraphicTypeCode: Char;
  StartPosition, EndPosition: LongInt;
begin
{ Запоминаем позицию потока для дальнейшей записи наших объектов... }
  StartPosition := Stream.Position;
{ Здесь мы собираемся записать позицию, где заканчиваются данные записи. Мы пока не
  знаем, как это позиционируется, поэтому записываем ноль, чтобы сохранить
  место }
  EndPosition := 0;
  Stream.Write(EndPosition, SizeOf(EndPosition));
{ Записываем строку Delphi }
  Stream.Write(FDescription[0], SizeOf(Byte));
  Stream.Write(FDescription[1], Byte(FDescription[0]));
{ Записываем целое... }
  Stream.Write(FItemType, SizeOf(FItemType));
{ Записываем код, сообщающий тип графического объекта, который мы собираемся
  писать }
  if (FGraphic = nil) or (FGraphic.Empty) then GraphicTypeCode := 'Z'
  else if FGraphic is TBitmap then GraphicTypeCode := 'B'
  else if FGraphic is TIcon then GraphicTypeCode := 'I'
  else if FGraphic is TMetaFile then GraphicTypeCode := 'M';
  Stream.Write(GraphicTypeCode, SizeOf(GraphicTypeCode));
{ Записываем графику ... }
  if (GraphicTypeCode <> 'Z') then FGraphic.SaveToStream(Stream);
{ Возвращаемся к месту, откуда мы начинали, и записываем конечную позицию, которую
  мы сохранили ... }
  EndPosition := Stream.Position;
  Stream.Seek(StartPosition, 0);
  Stream.Write(EndPosition, SizeOf(EndPosition));
{ Возвращаем конечную позицию, после этого поток готов для следующей записи ... }
  Stream.Seek(EndPosition, 0);
end;

procedure TAlbumRec.SetGraphic(AGraphic: TGraphic);
begin
  FGraphic.Free;
  FGraphic := nil;
  if AGraphic <> nil then begin
    FGraphic := TGraphic(AGraphic.ClassType.Create);
```

```
      FGraphic.Assign(AGraphic);
    end;
  end;

  end.
```

[News Group]

Примечание

Поскольку данный пример был ориентирован на пользователей Delphi 1, в 32-разрядных версиях Delphi (для Win32) будут выдаваться ошибки при трансляции следующих попыток работы со строками:

```
Stream.Read(FDescription[0], SizeOf(Byte))
```

Это вызвано тем, что 32-битная среда не ограничивает длину строки 255 символами и полагает, что ее размер не может храниться в одном (нулевом) байте. Поэтому вместо выражения `длина_строки := строка[0]` *нужно использовать* `длина_строки := Length(строка);` *а задавать длину строки надо при помощи процедуры* `SetLenght(имя_строки, размер_строки)`*, но не выражения* `строка[0] := длина_строки`.

Преобразование 16-битного DCR в 32-битный

Возможно ли преобразование 16-битного файла ресурсов, содержащего палитру значков компонентов в 32-битный файл?

Сначала запустите Resource Workshop и загрузите 16-битный файл ресурсов. Сохраните его как файл с расширением .RC (исходный файл ресурсов) и выйдите из RW.

Снова запустите RW. Укажите в настройках 32-битный режим работы с ресурсами. Откройте сохраненный на предыдущем этапе файл .RC. Снова сохраните его как .RES-файл (на этом этапе имя можно изменить, чтобы указать на 32-битный формат файла) и выйдите из RW.

[News Group]

Эксперт ресурсов

Что такое Resource Expert?

Resource Expert представляет собой встраиваемый (add-on) эксперт Delphi и составляет неотъемлемую часть Delphi Rad Pack. Эксперт помогает программисту портировать ресурсы в существующие проекты Delphi, преобразовывать диалоги и сценарии ресурсов меню для применения в традиционных приложениях Windows. Диалоговые ресурсы и их содержание преобразовываются в формы Delphi с аналогичными элементами управления, также преобразуемые в компоненты Delphi.

Resource Expert устанавливается с помощью процедуры установки Delphi Rad Pack Resource Workshop 4.5. После установки он включается в библиотеку компонентов Delphi, доступен как элемент меню Help и на вкладке Experts в диалоговом окне Forms Gallery (Галерея форм). Установка файлов Resource Expert может производиться как из среды Windows, так и с помощью командной строки Windows 9x и Windows NT.

Для установки файлов Resource Expert в среде Windows:

- Запустите процедуру установки Borland Resource Workshop;
- В третьем диалоговом окне, названном Resource Workshop – Resource Expert Options, убедитесь, что установлен флажок Install Resource Expert;
- В поле Install to: укажите каталог расположения файлов Resource Expert, по умолчанию это C:\DELPHI\RCEXPERT. Измените его по своему усмотрению;
- Остальная часть процесса установки Resource Workshop остается без изменений.

Для установки файлов Resource Expert из командной строки используйте следующие команды:

```
MD C:\DELPHI\RCEXPERT
CD C:\DELPHI\RCEXPERT
E:\INSTALL\RW\UNPAQ -X E:\INSTALL\RW\RESEXP.PAK
```

Примечание

В последней команде из приведенного списка предполагается, что диск E: является приводом CD-ROM, содержащим установочный компакт Rad Pack.

После установки файлов Resource Expert библиотека компонентов Delphi должна быть пересобрана. Для этого:

- Загрузите Delphi;
- Выберите пункты меню Options|Install Components;
- Нажмите кнопку Add...;
- В диалоговом окне Add Module введите полный путь к файлу rcexpert.pas или найдите его, нажав кнопку Browse...;
- В диалоговом окне Install Components Dialog нажмите кнопку OK.

Как использовать Resource Expert?

Для преобразования сценария ресурса (Resource scripts) и компиляции ресурса, в нормальной ситуации необходимо наличие всех исходных файлов. Они должны включать .RC, .MNU или .DLG файл(ы) и любые .H или .PAS файлы, на которые они ссылаются. Сценарии ресурса обычно используют WINDOWS.H и BWCC.H. Данные файлы обычно располагаются в каталогах типа \BC4\INCLUDE или \BP7\UNITS. Resource Expert поддерживает языковые расширения RC, определяемые в Resource Workshop.

Кроме того, Resource Expert может быть вызван через пункт меню Help|Resource Expert или с вкладки Experts диалогового окна Forms Gallery. Вызов последнего возможен, если установлен флажок Use on new form на вкладке Preferences диалогового окна Environment Options.

После запуска Resource Expert нажмите кнопку Next, таким образом вы пропустите окно презентации эксперта. Во втором окне эксперта пользователь может выбрать для преобразования сценарий ресурса. Могут быть выбраны несколько сценариев, при условии что все они находятся в одном каталоге. Конкретный тип сценария (.RC, .DLG или .MNU) может быть выбран в списке List Files of Type. Выбрав сценарий для преобразования, нажмите снова кнопку Next. Третье окно содержит единственное поле редактирования Include Path. Введите список каталогов, содержащих .H, .INC или включенные .PAS-файлы, используемые сценариями ресурса (если таковые имеются). Каждое имя каталога должно разделяться точкой с запятой. Для продолжения нажмите кнопку Next. В четвертом, последнем окне эксперта станет доступной кнопка Convert. Ее нажатие фактически запустит процесс преобразования. Если сценарий ресурса содержит много диалогов, сбросьте флажок Show all forms, это поможет увеличить скорость преобразования и уменьшит потребности в системных ресурсах.

Если в процессе преобразования обнаружится синтаксическая ошибка, ошибочный блок будет пропущен, а процесс преобразования продолжится со следующего блока. Возникшие ошибки сохраняются в файле журнала ERRLOG.TXT и выводятся в окне редактора Delphi. По окончании процесса преобразования для каждого ресурса диалога будет создана отдельная форма. Для ресурсов меню будет создана простая форма, содержащая компонент TMainMenu с теми же пунктами, что и преобразовываемый ресурс. Если перед началом преобразования какой-либо проект был активен, к нему будут добавлены все преобразованные формы. Каждая форма теперь может быть отредактирована и использована так, как будто бы это была любая форма Delphi.

Загрузка изображения/курсора из RES-файла

Изображения и курсоры могут храниться в файлах ресурсов (RES) и прилинковываться (связываться) к исполняемому файлу вашего приложения. RES-файлы могут создаваться с помощью утилит Image Editor и Borland Resource Workshop, входящие в поставку Delphi RAD Pack. Изображения и курсоры, хранимые в RES-файлах (после преобразования их в EXE или DLL) могут быть извлечены с помощью API-функций LoadBitmap и LoadCursor, соответственно.

Загрузка изображений

Функция API LoadBitmap определена следующим образом:

```
function LoadBitmap(Instance: THandle; BitmapName: PChar): HBitmap;
```

Первый параметр должен содержать дескриптор модуля (EXE или DLL), содержащего файл RES, из которого вы хотите получить ресурс. Delphi хранит

дескриптор запущенного EXE-файла в глобальной переменной с именем HInstance. В приведенном ниже примере мы предполагаем, что модуль, из которого мы пытаемся загрузить изображение, – ваше приложение. Тем не менее модуль мог бы быть другим файлом EXE или DLL. Следующий пример загружает изображение с именем BITMAP_1 из RES-файла, связанного с EXE-файлом приложения:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Bmp: TBitmap;
begin
  Bmp := TBitmap.Create;
  Bmp.Handle := LoadBitmap(HInstance,'BITMAP_1');
  Canvas.Draw(0, 0, Bmp);
  Bmp.Free;
end;
```

Имеется один недостаток применения API-вызова LoadBitmap: все-таки это API-вызов Windows 3.0, и он грузит изображение только как DDB (Device Dependent Bitmaps). Это может вызвать проблемы с цветовой палитрой при загрузке DIB (Device Independent Bitmaps) из RES-файла. Приведенный ниже код может использоваться для извлечения DIB из RES-файлов. Данный код загружает изображение как общий ресурс, передает его в поток, после чего выполняет Delphi-вызов LoadFromStream, реализующий палитру автоматически.

```
procedure TForm1.Button1Click(Sender: TObject);
const
{ Идентификатор типа изображения }
  BM = $4D42;
var
  Bmp: TBitmap;
  BMF: TBitmapFileHeader;
  HResInfo: THandle;
  MemHandle: THandle;
  Stream: TMemoryStream;
  ResPtr: PByte;
  ResSize: Longint;
begin
  BMF.bfType := BM;
{ Ищем, загружаем и блокируем ресурс, содержащий BITMAP_1 }
  HResInfo := FindResource(HInstance, 'BITMAP_1', RT_Bitmap);
  MemHandle := LoadResource(HInstance, HResInfo);
  ResPtr := LockResource(MemHandle);
{ Создаем Memory-поток, устанавливаем его размер, записываем туда заголовок
  изображения и, наконец, само изображение }
  Stream := TMemoryStream.Create;
  ResSize := SizeofResource(HInstance, HResInfo);
  Stream.SetSize(ResSize + SizeOf(BMF));
  Stream.Write(BMF, SizeOf(BMF));
  Stream.Write(ResPtr^, ResSize);
```

```
{ Освобождаем поток и сбрасываем его позицию в 0 }
  FreeResource(MemHandle);
  Stream.Seek(0, 0);
{ Создаем TBitmap и загружаем изображение из MemoryStream }
  Bmp := TBitmap.Create;
  Bmp.LoadFromStream(Stream);
  Canvas.Draw(0, 0, Bmp);
  Bmp.Free;
 Stream.Free;
end;
```

Загрузка курсоров

Функция API `LoadCursor` определена следующим образом:

```
function LoadCursor(Instance: THandle; CursorName: PChar): HCursor;
```

Первый параметр `Instance` должен содержать дескриптор модуля, содержащего файл RES. Как и в примере, приведенном выше, здесь предполагается, что модуль, из которого мы пытаемся загрузить курсор, – ваше приложение. Второй параметр – имя курсора. В секцию `interface` добавьте следующее объявление:

```
const
  crMyCursor = 5;       { Другие модули могут также использовать эту константу }
```

Затем допишите следующие две строчки к обработчику события формы `OnCreate`:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Screen.Cursors[crMyCursor] := LoadCursor(HInstance, 'CURSOR_1');
  Cursor := crMyCursor;
end;
```

Также можно изменить один из стандартных курсоров Delphi, как показано ниже (константы `Cursors` описаны в электронной справке):

```
procedure TForm1.FormCreate(Sender: TObject);
begin
{ Данный пример изменяет курсор SQL Hourglass }
  Screen.Cursors[crSQLWait] := LoadCursor(HInstance, 'CURSOR_1');
end;
```

Примечание

Обычно необходимо удалить любые ресурсы курсоров с помощью `DeleteCursor`*, тем не менее, в этом нет необходимости, поскольку Delphi сама удаляет все курсоры из массива* `Cursors`.

256-цветное изображение из .RES-файла

Функция, правильно читающая 256-цветные изображения из файла ресурсов.

```
function LoadBitmap256(hInstance: HWND; lpBitmapName: PChar): HBITMAP;
var
  hPal, hRes, hResInfo: THandle;
  pBitmap: PBitmapInfo;
  nColorData: Integer;
  pPalette: PLogPalette;
  X: Integer;
  hPalette: THandle;
begin
  hResInfo := FindResource(hInstance, lpBitmapName, RT_BITMAP);
  hRes := LoadResource(hInstance, hResInfo);
  pBitmap := Lockresource(hRes);
  nColorData := pBitmap^.bmiHeader.biClrUsed;
  hPal := GlobalAlloc(GMEM_MOVEABLE, (16 * nColorData));
{ hPal := GlobalAlloc(GMEM_MOVEABLE, (SizeOf(LOGPALETTE)
                  + (nColorData * SizeOf(PALETTEENTRY)));}
  pPalette := GlobalLock(hPal);
  pPalette^.palVersion := $300;
  pPalette^.palNumEntries := nColorData;
  for x := 0 to nColorData do begin
    pPalette^.palPalentry[X].peRed := pBitmap^.bmiColors[X].rgbRed;
    pPalette^.palPalentry[X].peGreen := pBitmap^.bmiColors[X].rgbGreen;
    pPalette^.palPalentry[X].peBlue := pBitmap^.bmiColors[X].rgbBlue;
  end;
  hPalette := CreatePalette(pPalette^);
  GlobalUnlock(hRes);
  GlobalUnlock(hPal);
  GlobalFree(hPal);
end;
end.
```

[News Group]

Несколько пиктограмм в Delphi EXE

Как с помощью Delphi присвоить выполнимому файлу сразу несколько пиктограмм? Необходимо сделать это так, чтобы, задав ассоциации типа файла и просматривая свое скомпилированное приложение, можно было видеть несколько доступных значков. К сожалению, опция Project|Options|Application|Icon *позволяет установить только одну пиктограмму.*

Создайте файл ресурса (.RES) в Image Editor и сохраните в нем свои пиктограммы. Затем свяжите («подлинкуйте») ресурс директивой компилятора `$R`, и ваше приложение будет иметь столько пиктограмм, сколько вы создадите.

Включение JPEG в EXE-файл

Я начинающий Delphi-программист и только что приступил к изучению этой замечательной среды разработки. Сейчас передо мной стоит задача распространения моей самой первой программы. Начиная с третьей версии, Delphi содержит модуль JPEG*, позволяющий работать с этим форматом изображений, и мне потребовалось включить графику* JPEG *в мой исполнимый файл для последующего использования в программе. Как это осуществить?*

Решение

Создайте файл сценария ресурса (*.RC) в обычном текстовом редакторе типа Notepad и добавьте следующую строку:

```
1  RCDATA  "MyPic.jpg"
```

Первый элемент является просто индексом ресурса. Второй элемент указывает на определенный пользователем ресурс. Третий, он же последний элемент, является именем JPEG-файла.

Для компиляции ресурса в .RES-файл используйте Borland Resource Compiler, BRCC32.EXE. В командной строке MS-DOS введите:

```
BRCC32 MyPic.RC
```

Это создаст файл ресурса с именем MyPic.RES.

Добавьте директиву компилятора к исходному коду программы. Она должна располагаться непосредственно за директивой формы, как показано ниже:

```
{$R *.DFM}
{$R MyPic.RES}
```

Добавьте следующий код к проекту (для этого создайте процедуру):

```
procedure LoadJPEGfromEXE;
var
  MyJPG: TJPEGImage;                // Объект - JPEG
  ResStream: TResourceStream;       // Объект - поток ресурсов
begin
  try
    MyJPG := TJPEGImage.Create;
    ResStream := TResourceStream.CreateFromID(HInstance, 1, RT_RCDATA);
    MyJPG.LoadFromStream(ResStream);
    Canvas.Draw(12, 12, MyJPG);
  finally
    MyJPG.Free;
    ResStream.Free;
  end;
end;
```

Обратили внимание на второй параметр процедуры CreateFromID объекта TResourceStream? Это просто индекс ресурса. Вы можете включить более одного

изображения JPEG в исполняемый модуль приложения, просто добавляя в .RC-файл строчку с другим индексом для каждого включаемого изображения.

Хранение данных в EXE-файле

Можно включить любой тип данных как RCDATA или пользовательский тип ресурса. Это очень просто. Данный совет поясняет общую технику создания такого ресурса.

```
type
  TStrItem = String[39];           { 39 символов + байт длины -> 40 байт }
  TDataArray = array [0..7, 0..24] of TStrItem;
const
  Data: TDataArray = (
                     (' ', ' ', ... ' '),            { 25 строк }
                     { всего 8 таких строк }
                     (' ', ' ', ... ' ');            { 25 строк }
```

Данные размещаются в вашем сегменте данных и занимают в нем 8 Кбайт. Если этого слишком много для вашего приложения – поместите реальные данные в ресурс RCDATA. Следующие шаги демонстрируют данный подход. Создайте небольшую безоконную программку, объявляющую типизированную константу как показано выше, и запишите результат в файл на локальный диск:

```
program MakeData;

type
  TStrItem = String[39];           { 39 символов + байт длины -> 40 байтов }
  TDataArray = array [0..7, 0..24] of TStrItem;

const
  Data: TDataArray = (
                     (' ', ' ', ... ' '),             { 25 строк }
                     { всего 8 таких строк }
                     (' ', ' ', ... ' '));            { 25 строк }
var
  F: File of TDataArray;

begin
  Assign(F, 'data.dat');
  Rewrite(F);
  Write(F, Data);
  Close(F);
end.
```

Теперь подготовьте файл ресурса и назовите его DATA.RC. Он должен содержать только следующую строчку:

```
DATAARRAY RCDATA "data.dat"
```

Сохраните файл, откройте сессию DOS, перейдите в каталог, где вы сохранили data.rc (там же, где и data.dat) и выполните следующую команду:

```
brcc data.rc   (brcc32 для Delphi 2.0 и выше)
```

Теперь у файл DATA.RES вы можете подключить к своему Delphi-проекту. Во время выполнения приложения вы можете генерировать указатель на данные этого ресурса и получить к ним доступ, что и требовалось.

```
type
  TStrItem = String[39];           { 39 символов + байт длины -> 40 байт }
  TDataArray = array [0..7, 0..24] of TStrItem;
  PDataArray = ^TDataArray;

const
  pData: PDataArray = Nil;          { в Delphi 2.0 используем Var }

implementation

{$R DATA.RES}

Procedure LoadDataResource;
var
  dHandle: THandle;
begin
{ pData := Nil; если pData - Var }
  dHandle := FindResource(hInstance, 'DATAARRAY', RT_RCDATA);
  if dHandle <> 0 then begin
    dHandle := LoadResource(hInstance, dHandle);
    if dHandle <> 0 then
      pData := LockResource(dHandle);
  end;
  if pData = Nil then

{ неудача, получаем сообщение об ошибке с помощью WinProcs.MessageBox, без помощи
  VCL, поскольку здесь код выполняется как часть инициализации программы и VCL,
  возможно, еще не инициализирована! }
end;

initialization
  LoadDataResource;
end.
```

[News Group]

Оглавление файлов помощи

Используйте HELP_FINDER, если «текущая вкладка» не является вкладкой Index или Find. HELP_FINDER открывает окно Help Topics, но не меняет вкладку с оглавлением (Contents), если текущая закладка – 'Index' или 'Find'.

Попробуйте след ующий код:

```
function L1InvokeHelpMacro(const i_strMacro: String;
                           const i_bForceFile: Boolean): Boolean;
begin
  if i_bForceFile then Application.HelpCommand(HELP_FORCEFILE, 0);
// Приведение типа PChar здесь необязательно.
  Result := Application.HelpCommand(HELP_COMMAND, Longint(PChar(i_strMacro)));
end;
```

Ищем ассоциированный файл помощи, открываем его (если он не открыт) и переходим на вкладку Index:

```
L1InvokeHelpMacro('Search()', True);
```

Ищем ассоциированный файл помощи, открываем его (если не он открыт) и переходим на вкладку Contents:

```
L1InvokeHelpMacro('Contents()', True);
```

Ищем ассоциированный файл помощи, открываем его (если он не открыт) и переходим на вкладку Find (только для WinHelp 4):

```
L1InvokeHelpMacro('Find()', True);
```

Отображение диалога Help Search

Решение 1

```
Application.HelpCommand(HELP_PARTIALKEY, 0);
```

Если данная команда не находит идентификатор #0 файла помощи (естественно, мы его и задаем), то выводится диалог Help Search.

Решение 2

Код демонстрирует способ вывода диалога WinHelp Search для электронной справки вашего приложения. Для этого следует послать системе электронной справки Windows (WinHelp) команду Help_PartialKey, что можно сделать с помощью метода объекта TApplication HelpCommand. Параметр для этой команды должен иметь тип PChar (можно привести к longint) и содержать строку, которую вам необходимо найти. Пример ниже использует для вызова диалога Search пустую строку, которую освобождает после его закрытия.

```
procedure TForm1.SearchHelp;
var
  P: PChar;
begin
  Application.HelpFile := 'c:\delphi\bin\delphi.hlp';
  P := StrNew('');
  Application.HelpCommand(Help_PartialKey, longint(P));
  StrDispose(P);
end;
```

Использование файла помощи

Код для трех стандартных пунктов меню Help:

```
procedure TForm1.Contents1Click(Sender: TObject);
begin
  Application.HelpCommand(HELP_CONTENTS, 0);
end;

procedure TForm1.SearchforHelpOn1Click(Sender: TObject);
begin
  Application.HelpCommand(HELP_PARTIALKEY, 0);
end;

procedure TForm1.HowtoUseHelp1Click(Sender: TObject);
begin
  Application.HelpCommand(HELP_HELPONHELP, 0);
end;
```

[News Group]

Невозможно открыть файл справки

Я создал файл справки для моего приложения и назвал его KidsHelp.hlp.

При запуске в системе, в которой файл был создан, программа находит его без проблем. На данной машине с процессором Pentium 120 установлена Windows 95. При запуске программы на второй системе, под Windows 3.1, при выборе пункта меню Using Help программа *не может открыть файл. Я создал файл справки с помощью программы* HC31.exe. *В самом проекте я не указывал полный путь к файлу справки, а указал только его имя.*

Для решения этой проблемы делают две вещи:

- всегда располагают файл справки в том же каталоге, что и приложение;
- назначают файл справки в обработчике события главной формы OnCreate таким образом:

```
Application.HelpFile := ChangeFileExt(Application.ExeName, '.HLP');
```

[News Group]

Закрытие файла справки

При закрытии моего приложения окно справки (если оно в это время открыто) не закрывается автоматически! Что надо сделать, чтобы оно закрывалось?

Решение

```
Application.HelpCommand(HELP_QUIT, 0);
```

[News Group]

Как мне создать интернет-ссылку в диалоге About?

Недавно в одной программе я увидел интересный эффект. Как вы знаете, почти каждая программа имеет диалоговое окно About *(*О программе*), так вот, в этом диалоге был простой WWW-адрес типа* http://www.somewhere.com. *Когда я перемещал мышь над этим адресом, он становился синим, ну прямо как на HTML-страничке! При щелчке на нем автоматически открывался броузер, установленный по умолчанию в системе, и осуществлялся переход по этому адресу. Кто-нибудь расскажет, как мне сделать это в моем собственном приложении?*

Расположите на форме компонент Label, заголовок которого будет демонстрировать ваш URL (я назвал его URLLabel). Я также назначаю этому компоненту другой цвет, чтобы он отличался от остального текста, расположенного на форме. Затем создайте у нашего компонента обработчик события OnClick следующего вида:

```
ShellExecute(Application.Handle, 'open', 'http://www.somewhere.com', nil, nil, 0);
```

или

```
ShellExecute(Application.Handle, 'open', 'mailto:towho@mysite.com', nil, nil, 0);
```

Не забудьте указать в списке используемых модулей модуль ShellAPI.

Для создания видимости ссылки присвойте свойству URLLabel.Cursor значение crAppStart.

Дополнение

```
uses
  ..., ShellAPI, ...

procedure TAboutBox.FormCreate(Sender: TObject);
begin
  Label1.Cursor := crHandPoint;
  Label2.Cursor := crHandPoint;
end;

procedure TAboutBox.Label1Click(Sender: TObject);
begin
  ShellExecute(Application.Handle, nil, 'http://www.xxx.com/xxx', nil, nil,
               SW_SHOWNOACTIVATE);
end;

procedure TAboutBox2.Label2Click(Sender: TObject);
begin
  ShellExecute(Application.Handle, nil, 'mailto:xxx@xxx.com', nil, nil,
               SW_SHOWNOACTIVATE);
end;

procedure TAboutBox.LabelMouseEnter(Sender: TObject);
begin
  with Sender as TLabel do begin
```

```
      if MouseInControl then Font.Color := clLime
      else Font.Color := clRed;
    end;
  end;
```

[Kravchenko Denis]

Таблицы строк

Ресурсы в виде таблиц строк (Stringtable) являются очень полезным подспорьем, если приложение должно хранить большое количество строк для их вывода во время выполнения приложения. Необходимо побороть искушение непосредственной вставки строк в программу, поскольку применение таблиц строк имеет два неоспоримых преимущества:

- Строки, хранимые в ресурсах, не занимают память до тех пор, пока они не будут загружены приложением.
- Stringtables легко редактировать, создавая таким образом локализованные (переведенные) версии приложения.

Таблицы строк компилируются в .RES-файл, включаемый в EXE-файл приложения во время сборки. Даже после того как вы распространите свое приложение, таблицы строк, содержащиеся в нем, могут редактироваться редактором ресурсов. Одним из редакторов ресурсов является Borland Resource Workshop, поставляемый в комплекте с Delphi. Он позволяет в режиме WYSIWYG редактировать как 16-, так и 32-битные ресурсы, как автономные, так и имплантированные в файлы .EXE или .DLL.

Тем более это удобно, если учесть, что вместе со всеми версиями Delphi поставляется компилятор ресурсов командной строки (Borland Resource Command Line Compiler) (BRCC.EXE и BRCC32.EXE), расположенный в Delphi-директории \Bin.

В нашем примере мы создадим интернациональное приложение, отображающее всего две кнопки. Кнопки будут иметь заголовки «Да» и «Нет», имеющие представления для английского, испанского и шведского языков.

Если вы вознамерились создавать многоязычные приложения с помощью Delphi, вам просто необходимо взглянуть на другие продукты фирмы Borland – Delphi Translation Suite и Language Pack software. Данные продукты позволяют изменять язык интерфейса приложения одним щелчком!

Пример

Для начала в каталоге с исходным кодом приложения мы должны создать текстовый файл, содержащий строковые ресурсы. Можно создать файл с любым именем (главное, чтобы он имел расширение .RC), но так, чтобы его имя не совпадало с именами файлов проекта. Это очень важно, поскольку Delphi автоматически создает множество файлов с ресурсами для вашего приложения с теми же именами и переписывает их, не заботясь о наличии таких же файлов, но созданных вашими руками.

Содержание .RC-файла для нашего примера. Файл содержит слова «Да» и «Нет» на английском, испанском и шведском языках:

```
STRINGTABLE {
 1, "&Yes"
 2, "&No"
 17, "&Si"
 18, "&No"
 33, "&Ja"
 34, "&Nej"
}
```

Файл начинается с ключевого слова STRINGTABLE, обозначая, что следом располагается таблица строк. Сами строки находятся внутри скобок, таким образом, таблица должна быть обрамлена двумя скобками – открывающей и закрывающей. Каждая строка должна содержать идентификатор, сопровождаемый строкой, заключенной в кавычки. Строка может содержать до 255 символов. Для того чтобы вставить нестандартный символ, напишите его восьмеричный код и предварите его обратной косой чертой. Если же требуется вставить саму обратную черту, то надо указать два таких символа. Вот примеры:

```
1, "A two\012line string"
2, "c:\\Borland\\Delphi"
```

Используемый номер индекса абсолютно не важен для компилятора. Вы должны иметь в виду, что таблицы строк располагаются в памяти в 16-битных сегментах (Win 3.xx).

Для компиляции .RC-файла в .RES-файл, который можно скомпоновать с приложением, необходимо набрать в командной строке полный путь к компилятору ресурсов и полный путь к компилируемому .RC-файлу.

Пример

```
c:\Delphi\Bin\brcc32.exe c:\Delphi\strtbl32.rc
```

По окончании процесса компиляции в указанном каталоге появляется файл с тем же именем, что и у .RC-файла, но имеющий расширение .RES. Для включения ресурсов в свое приложение необходимо в коде программы добавить следующую директиву компилятора, указывающую на файл с ресурсами:

```
{$R ResFileName.RES}
```

После того как .RES файл будет скомпонован с приложением, вы сможете воспользоваться связанными ресурсами из любого модуля вашего проекта, даже если директива $R определена в секции реализации (implementation) другого модуля.

Пример вызова функции LoadString() из Windows API для загрузки в массив символов третьей строки из таблицы строк:

```
if LoadString(hInstance, 3, @a, sizeof(a)) <> 0 then ...
```

В этом примере функция LoadString() передает дескриптор (hInstance) модуля, содержащего ресурс, индекс требуемой строки, адрес массива символов, куда будет передана строка, и размер самого массива. Функция LoadString возвращает количество реально переданных символов без учета терминатора. Будьте внимательны: при использовании UNICODE количество загружаемых байт будет другим.

Ниже приведен исчерпывающий пример создания многоязыкового приложения с помощью Delphi. Приложение совместимо как с 16- так и с 32-битными версиями Delphi.

Для этого придется создать два идентичных .RC-файла, один для 16-битной версии, второй для 32-битной, т. к. используемые ресурсы для каждой платформы свои. В данном примере мы создадим один файл с именем STRTBL16.RC, а другой с именем STRTBL32.RC. Скомпилируйте файл STRTBL16.RC с помощью 16-битного компилятора BRCC.EXE (расположен в каталоге \BIN Delphi 1) и файл STRTBL32.RC с помощью BRCC32.EXE (расположен в той же директории 32-битной версии Delphi).

Во время работы приложения мы выясняем язык операционной системы, установленный по умолчанию. Метод получения такой информации отличается для 16- и 32-битной версий Windows. Чтобы сделать код более читаемым, мы позаимствовали «языковые» константы из файла Windows.pas, применяемого в 32-битной версии Delphi.

```
{$IFDEF WIN32}
{$R STRTBL32.RES}
{$ELSE}
{$R STRTBL16.RES}
  const LANG_ENGLISH = $09;
  const LANG_SPANISH = $0a;
  const LANG_SWEDISH = $1d;
{$ENDIF}

function GetLanguage: word;
{$IFDEF WIN32}
{$ELSE}
var
  s: string;
  i: integer;
{$ENDIF}
begin
{$IFDEF WIN32}
  GetLanguage := GetUserDefaultLangID and $3ff;
{$ELSE}
  s[0] := Char(GetProfileString('intl', 'sLanguage', 'none', @s[1], SizeOf(s)-2));
  for i := 1 to length(s) do
    s[i] := UpCase(s[i]);
  if s = 'ENU' then GetLanguage := LANG_ENGLISH else
    if s = 'ESN' then GetLanguage := LANG_SPANISH else
      if s = 'SVE' then GetLanguage := LANG_SWEDISH else
```

```
        GetLanguage := LANG_ENGLISH;
{$ENDIF}
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  a: array[0..255] of char;
  StrTblOfs: integer;
begin
{ Получаем текущий язык системы и начало соответствующих строк в таблице }
  case GetLanguage of
    LANG_ENGLISH: StrTblOfs := 0;
    LANG_SPANISH: StrTblOfs := 16;
    LANG_SWEDISH: StrTblOfs := 32;
  else
    StrTblOfs := 0;
  end;
{ Загружаем и устанавливаем заголовок кнопки "Yes" в соответствии с языком }
  if LoadString(hInstance, StrTblOfs + 1, @a, SizeOf(a)) <> 0 then
    Button1.Caption := StrPas(a);
{ Загружаем и устанавливаем заголовок кнопки "No" в соответствии с языком }
  if LoadString(hInstance, StrTblOfs + 2, @a, SizeOf(a)) <> 0 then
    Button2.Caption := StrPas(a);
end;
```

Примечание

Те, кого не привлекают командная строка и ручное создание файлов ресурсов, могут воспользоваться любой имеющейся под рукой программой-редактором ресурсов.

Регулярные выражения

Я не понимаю, как работают REGULAR EXPRESSIONS *(регулярные выражения) в Delphi-диалоге* Replace dialog.

Регулярные выражения представляют собой специальные символы, употребляемые при поиске или переводе строки, позволяющие создавать шаблоны поиска вместо точного побуквенного ввода искомого текста.

Регулярные выражения подобны шаблонам DOS, но являются более мощным инструментом.

Выражение	Описание
?	Любой символ, кроме конца строки
*	Ноль или более символов (за исключением символа конца строки)
\t	Символ табулятора

продолжение

Выражение	Описание
\n	Символ конца строки
\c	Позиция курсора после сравнения
\\	Символ обратной косой черты
Символ	**Описание**
< или %	Начало строки
> или $	Конец строки
@	Ноль или более последних выражений
+	Один или более последних выражений
	Или последнее или следующее выражение
{}	Определение группы выражений
[ ]	Любой из символов в **[ ]**
[~]	Любой символ за исключением указанного в **[~]**

В тексте замены допускаются \t, \n, \c, а также:

\<n> – замещающий текст сравнивается с <n>-й группой (0 <= n <= 9)

Выражение	Результат
the	Поиск следующего включения «the»
{him} {her}	Поиск следующего включения «him» или «her»
<alone> или %alone$	Поиск следующего включения «alone» в данной строке
stuff*between	Поиск следующего включения «stuff», сопровождающегося «between» на той же строке
th[eo]se	Поиск следующего включения «these» или «those»
[A-Z][a-z]@;	Поиск следующего слова, начинающегося с большой буквы и заканчивающегося точкой с запятой
[0-9]+	Поиск одной или более последовательных цифр
[~ \t\n]	Поиск любого символа за исключением пробела, символа табулятора или новой строки

Имейте в виду, что выражения, содержащие символы *, @ и +, всегда являются сочетанием нескольких выражений, имеющих цель найти по возможности запрос, максимально близкий к искомому.

Скобки {} определяют группу в диалоге поиска, необходимую для поиска и (или) замены. Можно создать несколько групп и обращаться к ним по их номеру, закрепляемому за группой с помощью обратной косой черты и номером, стоящим за ней (счет начинается с 0, поиск ведется по группам слева направо).

Сложные примеры использования регулярных выражений при поиске и замене:

```
Поиск   FString\[{*}\]
Замена  Ord(FString[\0])
```

или

```
Поиск   {FString\[*\]}
Замена  Ord(\0)
```

Применение Tools Interface

Нужно работать с инструментальным интерфейсом (Tools Interface). Ну и как этим чудом воспользоваться?

Приведенный ниже код может использоваться для включения заголовка исходного кода, представляющего собой шапку с информацией об авторских правах, авторе, версии и т. д. при добавлении нового модуля или формы к проекту. TIAddInNotifier — класс, реализованный в ToolIntf и позволяющий «захватывать» такие события, как открытие файлов, их закрытие, открытие и закрытие проекта и т. д. Перекройте процедуру FileNotification для захвата событий AddedToProject и RemovedFromProject. В обработчике события AddedToProject можно получить доступ к новому модулю проекта, особенно это касается процедуры InsertHeader. Создайте наследника класса TIEditorInterface, расположенного в файле EditIntf.pas, и собственную процедуру InsertHeader.

VCSNotifier создается в другом модуле и здесь не показан. Приведенный ниже код является частью программы, осуществляющей контроль версий DLL. При создании код «живет» до тех пор, пока работает Delphi. При получении кода AddedToProject проверяйте наличие файла (должен быть новым), и что он является .Pas-файлом. Затем создайте VCSEditorInterface, унаследованный интерфейс, и используйте процедуру InsertHeader.

В самой процедуре InsertHeader создайте экземпляр TIEditReader для чтения нового модуля и TIEditWriter для его изменения.

```
unit VCSNtfy;

interface
```

```
uses  SysUtils, Dialogs, Controls, ToolIntf, EditIntf;

type
  TIVCSNotifier = class(TIAddInNotifier)
  public
    procedure FileNotification(NotifyCode: TFileNotification;
                           const FileName: string; var Cancel: Boolean); override;
  end;

  TIVCSEditorInterface = class(TIEditorInterface)
  public
    procedure InsertHeader;
  end;

var
  VCSNotifier: TIVCSNotifier;
  VCSModuleInterface: TIModuleInterface;
  VCSEditorInterface: TIVCSEditorInterface;

implementation

uses  FITIntf, FITStr, Classes;

procedure TIVCSNotifier.FileNotification(NotifyCode: TFileNotification;
                        const FileName: string; var Cancel: Boolean);
var
  TmpFileName: string;

begin
  case NotifyCode of
    fnRemovedFromProject: VCSProject.Remove(LowerCase(ExtractFileName(FileName)));
        fnAddedToProject: begin
                            if (not FileExists(FileName))
                                and (ExtractFileExt(FileName) = '.pas') then begin

{ новый файл с исходным кодом }
                              VCSModuleInterface :=
                                        ToolServices.GetModuleInterface(FileName);
                              if VCSModuleInterface <> nil then begin
                                VCSEditorInterface :=
                      TIVCSEditorInterface(VCSModuleInterface.GetEditorInterface);
                                VCSEditorInterface.InsertHeader;
                                VCSEditorInterface.Free;
                              end;
                              VCSModuleInterface.Free;
                            end;
                            TmpFileName := LowerCase(ExtractFileName(FileName));
                            if VCSProject.RecycleExists(TmpFileName) then begin
                              if MessageDlg('Вы хотите извлечь текущие '
                                          + ' записи из таблицы Recycle'
                                          + #13 + #10 + '          '
```

```
                                          + VCSProject.ProjectName + '/'
                                          + TmpFileName + '?', mtConfirmation,
                                            [mbYes,mbNo], 0 ) = mrYes then begin
                                VCSProject.Recycle(TmpFileName);
                              end;
                            end;
                          end;
  end;
end;

{ Начало TIVCSEditorInterface }
procedure TIVCSEditorInterface.InsertHeader;
var
  Module, TmpFileName, UnitName, InsertText, Tmp: string;
  Reader: TIEditReader;
  Writer: TIEditWriter;
  APos: Integer;
  F: TextFile;
begin
  TmpFileName := ExtractFileName(FileName);
  UnitName := SwapStr(TmpFileName, '.pas', '');
  SetLength(Module, 255);
  Reader := CreateReader;

  try
    Reader.GetText(0, PChar(Module), Length(Module));

  finally
    Reader.Free;
  end;
  APos := Pos('unit ' + UnitName, Module);
  if APos > 0 then begin
    try
      InsertText := '';
      AssignFile(F, VCSConfig.HeaderFileLocation);
      Reset(F);
      while not EOF(F) do begin
        Readln(F, Tmp);
        InsertText := InsertText + #13 + #10 + Tmp;
      end;
      CloseFile(F);
      InsertText := InsertText + #13 + #10;
      Writer := CreateWriter;
      try
        Writer.CopyTo(APos - 1);
        Writer.Insert(PChar(InsertText));

      finally
        Writer.Free;
      end;
    except
```

```
      On E: EStreamError do
        MessageDlg('Не могу создать шапку', mtInformation, [mbOK], 0);
    end;
  end;
end;

end.
```

[News Group]

Назначение события во время выполнения программы

Процедура, назначающая событие компонента обработчику события другого (или того же самого) компонента, где во время выполнения программы само событие и его обработчик заданы в виде строки. В случае неверных имен события или его обработчика процедура возбуждает исключительную ситуацию. Вы можете «очищать» событие, передавая компоненту Nil с обработчиком или нулевое имя самого обработчика.

Для демонстрации того, как это можно использовать, я включил в пример пару маленьких событий для кнопок. Во время работы вы могли бы протестировать это с парой областей редактирования, имеющих другой тип и другие имена обработчиков.

Почувствуйте мощь RTTI.

```
procedure SetEvent(ComponentWithEvent: TComponent; const Event: string;
                   ComponentWithHandler: TComponent; const Handler: string);
var
  PropInfo: PPropInfo; Method: TMethod;
begin
  PropInfo := GetPropInfo(ComponentWithEvent.ClassInfo, Event);
  if PropInfo = Nil then
    Raise Exception.CreateFmt('Событие %s не найдено в классе %s',
                              [Event, ComponentWithEvent.ClassName]);
  Method.Code := Nil;
  if Assigned(ComponentWithHandler) and (Handler <> '') then begin
    Method.Code := ComponentWithHandler.MethodAddress(Handler);
    if Method.Code = nil then
      Raise Exception.CreateFmt('Класс %s не имеет метода с именем %s',
                                [ComponentWithHandler.ClassName, Handler]);
  end;
  Method.Data := ComponentWithHandler;
  SetMethodProp(ComponentWithEvent, PropInfo, Method);
end;

{ примеры, показывающие, как использовать SetEvent }
procedure TForm1.SetBtnClick(Sender: TObject);
begin
  SetEvent(MenuItem, 'OnClick', Self, 'Test1Click');
end;
```

```
procedure TForm1.ClearBtnClick(Sender: TObject);
begin
  SetEvent(MenuItem, 'OnClick', Nil, '');
end;
```

[News Group]

Примечание

Этот код использует RTTI Delphi 1. В современных версиях этой среды программирования некоторые вызовы и соглашения видоизменились. Этот пример необходимо доработать для других версий Delphi.

Делегирование события

Как осуществляется делегирование события?

Покажем это на примере создаваемого компонента:

- Определите тип процедуры, выступающей в качестве обработчика события. Допустим, ваш обработчик `OnCalculate` имеет один параметр типа `Integer` (присутствующий лишь для демонстрации идеи).

```
type
  TCalculateEvent = procedure(I: Integer) of object;
```

- Теперь объявите ваш класс:

```
type
  TSomeClass = class(TObject)
  private
    FOnCalculate: TCalculateEvent;
    procedure DoCalculate(I: Integer);
  public
    property OnCalculate: TCalculateEvent read FonCalculate
                                          write FOnCalculate;
end;
```

- Метод `DoCalculate` совсем простой:

```
procedure TSomeClass.DoCalculate(I: Integer);
begin
  if Assigned(FOnCalculate) then FOnCalculate(I);
end;
```

Теперь вы можете присваивать значение объекту `TSomeClass` и назначать любую процедуру событию `OnCalculate` (естественно, при условии, что ее объявление соответствует объявлению `TCalculateEvent`).

[News Group]

Получение имени обработчика события

Имеется возможность получить значение указателя на обработчик события, который вы можете сравнить по адресу с другими методами, чтобы вычислить тот, которому передается данное событие:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  P: record
     case Integer of
       1: (E: TNotifyEvent);
       2: (P: Pointer);
     end;
begin
  P.E := Button1.OnClick;
  Label1.Caption := 'Обработчик события = ' + MethodName(P.P);
  ShowMessage(Format('%p', [P.P]));
end;
```

Синтаксис ссылки на событие

Я хочу проверить значение события, чтобы убедиться, что оно указывает в настоящий момент на определенную функцию.

По всей видимости, вам придется использовать две временные переменные. Попытайтесь сделать следующее:

```
  ...
var
  ClickMethod: TNotifyEvent;
  B1Click: TNotifyEvent;
  ...
  ClickMethod := Button1.OnClick;
  B1Click := Button1Click;
  if @ClickMethod = @B1Click then MessageBeep(0);
```

[News Group]

Сообщение для всех форм

Как послать сообщение всем формам?

Решение

```
  ...
var
  I: Integer;
  M: TMessage;

  ...
```

```
  with M do begin
    Message := ...
  ...
  end;
  for I := 0 to Pred(Screen.FormCount) do begin
    PostMessage(Forms[I].Handle, ...);
    // Если надо и всем
    Forms[I].Broadcast(M);
  end;
  ...
```

[News Group]

Имитация события MouseOff

Как не упустить момент, когда мышь покидает область визуального компонента? Есть какой-нибудь эквивалент событию `MouseMoveOff`*?*

В обработчике события `TComponent.MouseMove` установите логический флаг, а в обработчике контейнера компонента `MouseMove` проверяйте этот флаг; если он равен `True`, «виртуальное» событие произошло и требует обработки.

[News Group]

Обработка исключительных ситуаций

Как обработать ошибку, прежде чем программа уведомит о ней пользователя?

Для фильтрации, например, `EConvertError`, можно создать собственный обработчик исключений.

Объявите следующую процедуру в объекте главной формы:

```
procedure MyException(Sender: TObject; E: Exception);
```

Затем воспользуйтесь следующим кодом:

```
procedure TMyForm.MyException(Sender: TObject; E: Exception);
begin
  if (E.ClassType.ClassName = 'EConvertError') then begin
{ как-то общаемся с пользователем по поводу ошибки }
  end else
    Application.ShowException(E); { позволяем Delphi показать ошибку }
end;
```

Разрешите приложению воспользоваться вашим новым обработчиком исключений:

```
procedure TMyForm.FormCreate(Sender: TObject);
begin
  Application.OnException := MyException;
end;
```

Использование исключений в базе данных

Решение

```
try
  Tabl.Post;
except
  begin
    On EDatabaseError do ShowMessage('Не могу отправить данные (выполнить Post)');
    (Sender as TDBEdit).SetFocus;
  end;
end;
```

Выполните синтаксический разбор Error и генерируйте исключение вновь. Передайте по иерархии следующему обработчику объектов исключительных ситуаций, если больше нет нужды в обработке. Посредством:

```
On E: EDatabaseError do ...,
```

можно получить значение E.Error. Реально имя свойства с текстом ошибки должно быть похоже на что-то типа E.Message (уточните в электронной справке).

[News Group]

Определение версии Delphi

Как во время компиляции модуля определить, под какой версией Delphi она происходит?

Используйте следующий код:

```
{$IFDEF VERXXX}
  ...
{$ELSE}
  ...
{$ENDIF}
```

Ниже приведена таблица соответствия констант программным продуктам фирмы Borland:

```
* VER80 - Delphi 1
* VER90 - Delphi 2
* VER93 - C++Builder 1
* VER100 - Delphi 3
* VER110 - C++Builder 3
* VER120 - Delphi 4
* VER130 - Delphi 5
```

4

Базы данных

Проблемы с кириллицей в Database Desktop

Database Desktop отображает содержимое таблиц шрифтом без русских букв. В чем проблема?

Для DBD 5.0 в файле C:\Windows\PDOXWIN.INI вставить в секцию:

```
[Properties]
SystemFont=Arial Cyr
```

Для DBD 7.0 нужно исправить значение ключа реестра

```
HKEY_CURRENT_USER\Software\Borland\DBD\7.0\Preferences\Properties\
SystemFont = "Fixedsys"
```

Если такой ключ не существует, его следует создать. В Windows NT нужно изменить:

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Nls\CodePage]
  "1252"="c_1251.nls"
```

Примечание

Помогает также следующий метод. В файле win.ini *в конце раздела* [FontSubstitutes] *пишем:* Arial=Arial Cyr. *Перегружаем компьютер, т. к. изменения вступят в силу только после обработки этого файла на этапе загрузки.*

Информация о псевдониме BDE

Как получить информацию о псевдониме BDE?

Решение 1

```
procedure TForm1.Button1Click(Sender: TObject);
{ Получаем из BDE путь MyAlias }
const
  AliasName = 'MyAlias';
var
  MyAliasPath: string;
  ParamsList: TStringList;
begin
  ParamsList := TStringList.Create;
  try
    with Session do begin
      Session.GetAliasNames(ParamsList);
      Session.GetAliasParams(AliasName,ParamsList);
      MyAliasPath := Copy(ParamsList[0], 6, 50) + '\';
    end;
  finally
    ParamsList.Free;
  end;
  Label1.Caption := MyAliasPath;
end;
```

Примечание

Для работы данного примера поместите на форму компоненты `Label`, `Button`. *В обработчике события* `OnClick` *кнопки наберите код. Вместо* `MyAlias` *напишите любой реальный псевдоним (alias).*

Решение 2

```
uses
  DbiProcs, DBiTypes;

{ Возвращает каталог базы данных для псевдонима (без обратного слэша) }
function GetDataBaseDir(const Alias: string): String;
var
  sp: PChar;
  Res: pDBDesc;
begin
  try
    New(Res);
    sp := StrAlloc(length(Alias) + 1);
    StrPCopy(sp, Alias);
    if DbiGetDatabaseDesc(sp, Res) = 0 then Result := StrPas(Res^.szPhyName)
    else Result := '';
  finally
    StrDispose(sp);
```

```
    Dispose(Res);
  end;
end;
```

Примечание

Для работоспособности этого кода надо иметь активную таблицу в приложении с тем же псевдонимом, что и Alias. *Этот код несколько избыточен, поэтому предлагается вариант из справки:*

```
uses
  DBiTypes;

function GetDataBaseDir(const Alias: string): String;
var
  R: DBDesc;
begin
  Check(DbiGetDatabaseDesc(PChar(Alias), @R));
  Result := R.szPhyName;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
  Label1.Caption := GetDataBaseDir('Alias');
end;
```

База данных в кодировке CP1251

Как создать базу данных в кодировке CP1251?

Такая конструкция проходит на DB2 2.1.2/NT и UDB5/NT.

```
CREATE DATABASE Efes2
  USING CODESET 1251 TERRITORY RU
  COLLATE USING IDENTITY;
```

[Nomadic]

ASCII-драйвер для файлов CSV

Как использовать драйвер ASCII для файлов с разделительной запятой

Delphi (и BDE) способны сохранять таблицы в ASCII-файлах. Драйвер ASCII имеет возможность транслировать значения данных ASCII-поля фиксированной длины или файла с разделительной запятой в поля и величины, которые могут отображаться компонентом TTable. Трансляция ASCII-файла целиком зависит от сопровождающего файла схемы (Schema File). Файл схемы для файла данных ASCII определяет различные атрибуты, необходимые для преобразования данных ASCII-файла в значения отдельных полей. Определение полей для файла с ASCII-полями фиксированной длины – достаточно простая задача: необходимо знать позиции всех полей, для всех строк

они одинаковы. Для файлов с разделительной запятой данный процесс чуть более усложнен из-за того, что не все данные в таком файле во всех строках имеют одинаковую длину. Данный совет как раз и концентрируется на описании этой трудной темы, связанной с чтением данных из файлов с разделительной запятой, имеющих варьируемую длину поля.

Файл схемы

Файл схемы для файла данных ASCII содержит информацию, которая определяет оба типа файла (версии с разделительной запятой и полем с фиксированной длиной), а также определяет поля, которые представлены значениями данных в каждой строке файла данных ASCII. (Все поля файла схемы нечувствительны к регистру, поэтому написание `"ascii"` равнозначно написанию `"ASCII"`.) Для того чтобы файл схемы был признан в качестве такового, он должен иметь то же имя, что и файл данных ASCII, для которого он содержит схему, но иметь расширение `.SCH` (SCHema – схема). Атрибуты описания файла:

- `File name`: заключается в квадратные скобки, данный атрибут определяет имя файла данных ASCII (с расширением имени файла, которое должно быть равно `.TXT`).
- `Filetype`: определяет, имеет ли файл данных ASCII структуру файла с полями фиксированной длины (используется атрибут `FIXED`) или представляет собой файл с разделительной запятой (со значениями данных, которые потенциально могут изменять длину (используется атрибут `VARYING`).
- `Delimiter`: определяет символ, которым ограничиваются значения данных типа `String` (обычно двойные кавычки, десятичный ASCII-код 34).
- `Separator`: определяет символ, который используется для разделения отдельных значений данных (обычно запятая). Данный символ должен быть видимым символом, т. е. не может быть пробелом (десятичный код ASCII равен 32).
- `CharSet`: определяет драйвер языка (используется атрибут `ASCII`).

Перечисленные ниже атрибуты файла являются определениями поля, задающими правила для каждой строки файла данных ASCII. Данные определения служат для Delphi и BDE источником информации, первоначально необходимой для создания виртуального поля в памяти, в свою очередь служащего для хранения значений данных; тип данных виртуального поля определяется после чтения и трансляции данных из ASCII-файла, определения размера и применения атрибутов. Различные атрибуты, определяющие поле файла данных ASCII:

- `Field`: имя виртуального поля (всегда `"Field"`), сопровождаемое целым числом, определяющим порядковый номер поля относительно других полей в файле данных ASCII. Например, первое поле – `Field1`, второе – `Field2` и т. д.

- `Field name`: определяет выводимое имя поля, отображаемое в виде заголовка колонки в `TDBGrid`. Соглашения имен для таблиц ASCII такие же, как и для таблиц Paradox.
- `Field type`: определяет, какой тип данных BDE должен использоваться при трансляции значений данных каждого поля, и сообщает Delphi тип виртуального поля, которое необходимо создать. (Фактически формат для значений данных даты и времени будет определяться текущими настройками конфигурации BDE, окно с вкладкой `Date`.)

Используйте определение	Для значений типа
`CHAR`	Символ
`FLOAT`	64-битное число с плавающей точкой
`NUMBER`	16-битное целое
`BOOL`	`Boolean` (`T` или `F`)
`LONGINT`	32-битное длинное целое
`DATE`	Поле `Date`
`TIME`	Поле `Time`
`TIMESTAMP`	Поле `Date + Time`

- `Data value length`: максимальная длина значения данных соответствующего поля. Данный атрибут определяет длину виртуального поля, создаваемого Delphi для получения считываемых значений из ASCII-файла.
- `Number of decimals`: приложение к полю типа `FLOAT`; определяет количество цифр справа от десятичной точки; необходимо для включения в определение виртуального поля.
- `Offset`: отступ от начала строки, позиция начала данных описываемого поля; задается для всех строк файла.

Например, приведенное ниже определение поля относится к первому полю таблицы ASCII. Данная строка определяет значения данных типа `String` с именем `"Text"`. Максимальная длина значения данных составляет три символа (и в Delphi-компонентах для работы с базами данных типа `TDBGrid` поле будет отображаться только тремя символами): десятичный порядок (значение данных типа `String` никогда не сможет иметь десятичные значения, тем более после запятой) и смещение относительно нулевой позиции (поскольку описываемая область первая, то она сама начинается с нулевой позиции, перед ней не находится ни одно поле).

```
Field1 = Text, Char, 3, 00, 00
```

Пример файла схемы с тремя полями, первое поле имеет тип String, второе и третье – тип Date. Данный файл схемы должен содержаться в файле с именем DATES.SCH и обеспечивать определения полей для файла данных ASCII с именем DATES.TXT.

```
[DATES]
Filetype = VARYING
Delimiter = "
Separator = ,
CharSet = ascii
Field1 = Text, Char, 3, 00, 00
Field2 = First Contact, Date, 10, 00, 03
Field3 = Second, Date, 10, 00, 13
```

Данная схема определяет поле с разделительной запятой, где все данные могут быть отнесены к типу String, значения полей ограничены двойными кавычками. Отдельные значения полей разделены запятой (за исключением любых запятых, которые могут находиться между разделительными запятыми, внутри отдельных значений полей типа String). Первое поле типа character имеет длину три символа, без определения десятичного порядка и с нулевым отступом от начала строки. Второе поле данных имеет длину 10, без определения десятичного порядка и отступ, равный трем. Третье поле данных имеет длину 10 и отступ, равный 13; десятичный порядок не определен.

В случае чтения файлов ASCII с разделительной запятой параметры длины и отступа для определения поля относятся не к значениям данных в файлах ASCII (что явно противоположно для файлов с полями фиксированной длиной), а к виртуальным полям, определяемым в приложении, в котором будут размещены считываемые данные. Параметр длины должен отражать максимальную длину значений данных для каждого поля, не считая ограничительных кавычек и разделительной запятой. Это наиболее трудно при оценке значений данных типа String, поскольку фактическая длина значений данных может существенно различаться для каждой строки в файле данных ASCII. Параметр отступа для каждого поля не будет являться позицией значений данных в файле ASCII (что относится к файлам с полями фиксированной длины), а представляет собой совокупную длину всех предыдущих полей (и это относиться к определению полей в памяти, а не к значениям данных в файле ASCII).

Файл данных с именем DATES.TXT, который соответствует описанному выше файлу схемы:

```
"A",08/01/1995,08/11/1995
"BB",08/02/1995,08/12/1995
"CCC",08/03/1995,08/13/1995
```

Максимальная длина фактических значений данных в первом поле составляет три символа ("CCC"). Поскольку это первое поле и предшествующих полей не существует, отступ для данного поля равен нулю. Длина первого поля (3) интерпретируется как отступ для второго поля. Длина второго поля

(значение date) равна 10 и отражает максимальную длину значения данных этого поля. Совокупная длина первого и второго полей используется в качестве значения отступа для третьего поля (3 + 10 = 13).

Только когда соответствующая длина значения данных ASCII-файла или длина каждого поля добавляется к длине предыдущих полей, вычисляется значение отступа и получается позиция очередного поля, только тогда данный процесс правильно считает данные. Если из-за неправильных установочных параметров в файле схемы данные транслируются неверно, то в большинстве типов полей могут возникнуть неблагоприятные эффекты типа усечения строк или интерпретирования цифр как нулей. Обычно в таком случае данные выводятся, но ошибки не возникает. Тем не менее, данные определенного формата в процессе трансляции в подходящий тип могут вызвать ошибку, если считываемые символы не соответствуют символам, например, в типе date. В контексте вышесказанного, ошибка с типом date может возникнуть из-за того, что при неправильном определении в значения данных могут попасть данные другого, соседнего поля. При таком стечении обстоятельств трансляция данных прерывается. В файле схемы требуется установка правильной длины поля и его отступа.

ASCII–файл, содержащий разметку полей

Как правильно работать с текстовым файлом, имеющим разметку полей?

В том случае, если вы собираетесь работать с текстовым файлом так, как будто он имеет поля, необходим файл схемы, содержащий описание формата текстового файла, и который необходим для осуществления вызовов при работе с полями (Fields/FieldByName/Post/ и др.). Ниже приводится код, который можно использовать при создании своей программы:

```
{Подразумеваем, что Table1 - файл, который мы хотим скопировать в ASCII-файл.
 Используем TBatchMove, поскольку он быстро работает. Также это автоматически
 создаст файл схемы }

procedure TForm1.Button1Click(Sender: TObject);
var
  oDest: TTable;
  oBMove: TBatchMove;
begin
  try
    oDest := nil;
    oBMove := nil;
    Table1.Close;
    oDest := TTable.Create(nil);
    with oDest do begin
      DatabaseName := 'c:\delphi\files';
      TableName := 'Test.Txt';
      TableType := ttASCII;
    end;
```

```
{ Обратите внимание на то, что нет необходимости вызывать CreateTable }
    oBMove := TBatchMove.Create(nil);
    with oBMove do begin
      Source := Table1;
      Destination := oDest;
      Mode := batCopy;
      Execute;
    end;
  finally
    if Assigned(oDest) then oDest.Free;
    if Assigned(oBMove) then oBMove.Free;
  end;
end;

{ Теперь, допустим, файл схемы существует; сам текстовый файл может как быть,
  так его может и не быть. С помощью файла схемы мы уже можем работать с полями }

procedure TForm1.Button2Click(Sender: TObject);
var
  oTxt: TTable;
  i: Integer;
  f: System.Text;
begin
  try
    oTxt := nil;
    if not FileExists('c:\delphi\files\Test.Txt') then begin
      AssignFile(f, 'c:\delphi\files\Test.Txt');
      Rewrite(f);
      CloseFile(f);
    end;
    oTxt := TTable.Create(nil);
    with oTxt do begin
      DatabaseName := 'c:\delphi\files';
      TableName := 'Test.Txt';
      TableType := ttASCII;
      Open;
    end;
    with Table1 do begin
      DisableControls;
      if not Active then Open;
      First;
      while not EOF do begin
        oTxt.Insert;

{ В данном случае файл схемы описывает формат текстового файла;
  фактически, один к одному воспроизводятся поля таблицы в логическое определение
  полей в .sch-файле }

        for i := 0 to FieldCount - 1 do
          oTxt.Fields[i].AsString := Fields[i].AsString;
        oTxt.Post;
```

```
        Next;
      end;
    end;
  finally
    Table1.EnableControls;
    if Assigned(oTxt) then oTxt.Free;
  end;
end;
```

Получение физического пути к таблице

Как получить физический путь к таблице?

Если ссылка на таблицу получена через псевдоним, получить физический путь к ней не так просто. Для получения этого пути необходимо использовать функцию BDE DbiGetDatabaseDesc. Данной функции в качестве параметров передаются имя псевдонима и указатель на структуру DBDesc. Структура DBDesc будет заполнена информацией, относящейся к этому псевдониму. Определение структуры:

```
pDBDesc = ^DBDesc;
DBDesc = packed record    { Описание данной базы данных }
  szName: DBINAME;        { Логическое имя (или псевдоним) }
  szText: DBINAME;        { Описательный текст }
  szPhyName: DBIPATH;     { Физическое имя/путь }
  szDbType: DBINAME;      { Тип базы данных }
end;
```

Физическое имя/путь будет содержаться в поле szPhyName структуры DBDesc.

Возможные значения, возвращаемые функцией DBIGetDatbaseDesc:

- DBIERR_NONE — описание базы данных для pszName было успешно извлечено.
- DBIERR_OBJNOTFOUND — база данных, указанная в pszName, не была обнаружена.

Приведенный ниже пример кода показывает, как можно получить физический путь для компонента TTable, использующего псевдоним DBDemos:

```
var
  vDBDesc: DBDesc;
  DirTable: String;
begin
  Check(DbiGetDatabaseDesc(PChar(Table1.DatabaseName), @vDBDesc));
  DirTable := Format('%s\%s', [vDBDesc.szPhyName, Table1.TableName]);
  ShowMessage(DirTable);
end;
```

Получение информации о таблице

Как получить информацию о таблице?

Нужно воспользоваться свойством FieldDefs. В следующем примере список полей и их размеры передаются компоненту TMemo (расположенному на форме) с именем Memo1:

```
procedure TForm1.ShowFields;
var
  i: Word;
begin
  Memo1.Lines.Clear;
{ должно быть вызвано, если Table1 не активна }
  Table1.FieldDefs.Update;
  for i := 0 to Table1.FieldDefs.Count - 1 do
    with Table1.FieldDefs.Items[i] do
      Memo1.Lines.Add(Name + ' - ' + IntToStr(Size));
end;
```

Если просто нужны имена полей (FieldNames), то обратитесь к методам TTable — GetFieldNames, GetIndexNames для получения имен индексов:

```
var
  FldNames, IdxNames: TStringList

begin
  FldNames := TStringList.Create;
  IdxNames := TStringList.Create;
  if Table1.State = dsInactive then Table1.Open;
  Table1.GetFieldNames(FldNames);
  Table1.GetIndexNames(IdxNames);
  FldNames.Free;      {освобождаем stringlist}
  IdxNames.Free;
  end;
```

Для получения информации об определенном поле нужно использовать FieldDef.

Структура таблицы

Существует ли средство для вывода определения структуры таблицы?

Для этого существует утилита DB2LOOK. Она находится в каталоге \SQLLIB\MISC.

Пример использования:

```
CONNECT TO SAMPLE USER xxx USING yyy
DB2LOOK -d SAMPLE -u xxx -e -t employee
```

Вывод может быть перенаправлен в файл. Полный синтаксис выдаNomadiстся по команде:

```
DB2LOOK ?
```

[Из архивов fido7.ru.delphi]

Создание DBF-файла во время работы приложения

Как создать таблицу из работающего приложения?

Простейший способ – использовать запрос SQL. Таблицы можно создавать с индексом и без индекса.

Небольшой пример:

```
...
const
  CreateTab = 'CREATE TABLE ';
  IDXTab = 'PRIMARY KEY ';
  MyTabStruct = 'IDX_TAB DECIMAL(6,0), '
    + 'DATE_ DATE, '
    + 'FLD_1 CHARACTER(20), '
    + 'FLD_2 DECIMAL(7, 2), '
    + 'FLD_3 BOOLEAN, '
    + 'FLD_4 BLOB(1, 1), '
    + 'FLD_5 BLOB(1, 2), '
    + 'FLD_6 BLOB(1, 3), '
    + 'FLD_7 BLOB(1, 4), '
    + 'FLD_8 BLOB(1, 5) ';
  ...

function TForm1.CreateTable(TabName, TabStruct, TabIDX: string): boolean;
var
  qyTable: TQuery;
begin
  result := true;
  qyTable := TQuery.Create(Self);
  with qyTable do
    try
      try
        SQL.Clear;
        SQL.Add(CreateTab + TabName + '(' + TabStruct + TabIDX + ')');
        Prepare;
        ExecSQL;    // ExecSQL, а не Open. Иначе ...
      except

        // Обработка ошибок открытия таблицы
        Exception.Create('Ошибка открытия таблицы');
        result := false;
      end;
    finally
```

```
      Close;
    end;
end;

// создание таблицы без индекса
procedure TForm1.Button1Click(Sender: TObject);
begin
  if CreateTable('"MYTAB1.DBF"', MyTabStruct, '') then
    ...    // выполняем дальнейшие операции
  else
    ...
end;

// создание таблицы с индексом
procedure TForm1.Button2Click(Sender: TObject);
begin
  if CreateTable('"MYTAB2.DBF"', MyTabStruct, IDXTab + ' (IDX_TAB)') then
    ...    // выполняем дальнейшие операции
  else
    ...
end;
```

[VS]

Упаковка таблиц dBASE

Как упаковать таблицу dBASE?

Для упаковки таблицы dBASE убедитесь в том, что таблица открыта в монопольном (exclusive) режиме, и вызывайте `DbiPackTable`.

Решение

```
Table1.Close;
Table1.Exclusive := TRUE;
Table1.Open;
DbiPackTable(Table1.DBHandle, Table1.Handle, nil, nil, TRUE);
```

Убедитесь, что `DBITYPES`, `DBIPROCS`, `DBIERRS` включены в секцию `USES` вашего модуля и что в случае запуска из-под IDE в режиме проектирования таблица не активна.

Динамическое создание полей

Как во время работы приложения динамически создавать поля в наборе данных?

Решение 1

```
procedure TForm1.Button1Click(Sender: TObject);
var
  I: Integer;
  Field: TField;
```

```
begin
{ Поля можно добавлять только к неактивному набору данных. }
  Table1.Active := False;
{ Распределяем назначенные поля, если набор данных еще не был активным. }
  Table1.FieldDefs.Update;
{ Создаем все поля из определений и добавляем к набору данных. }
  for I := 0 to Table1.FieldDefs.Count - 1 do begin
{ Вот где мы действительно сообщаем набору данных о необходимости создания поля.
  Поле "назначается", но нам нужно не это, нам нужна просто ссылка на новое поле. }
    Field := Table1.FieldDefs[I].CreateField(Table1);
  end;
{ Вот пример того, как вы можете добавить дополнительные, вычисленные поля }
  Field := TStringField.Create(Table1);
  Field.FieldName := 'Total';
  Field.Calculated := True;
  Field.DataSet := Table1;
{ Теперь мы можем увидеть наши поля. }
  Table1.Active := True;
end;
```

Решение 2

Код создает полный набор «default» TField для TTable и добавит затем вычисляемое поле:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  f: TField;
  i: integer;
begin
  Table1.Close;
  for i := 0 to Table1.FieldDefs.Count - 1 do
    Table1.FieldDefs.Items[i].CreateField(Table1);
  f := TStringField.Create(Table1);
  f.Name := 'Table1CalcField';
  f.FieldName := 'CalcField';
  f.DisplayLabel := 'CalcField';
  f.Calculated := True;
  f.DataSet := Table1;
  Table1.Open;
end;
```

Следующий пример создаст два новых TField в TTable, «базируясь» на TField, определенных в режиме редактирования. Одно из новых полей вычиляемое, другое – нет:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  f1, f2: TField;
begin
  Table1.Close;
  f1 := TStringField.Create(Table1);
```

```
    f1.Name := 'Table1CalcField';
    f1.FieldName := 'CalcField';
    f1.DisplayLabel := 'CalcField';
    f1.Calculated := True;
    f1.DataSet := Table1;
    f2 := TFloatField.Create(Table1);
    f2.Name := 'Table1Population';
    f2.FieldName := 'Population';
    f2.DisplayLabel := 'Population';
    f2.DataSet := Table1;
    Table1.Open;
  end;
```

Создание индексного файла из приложения

Как создать индексный файл из приложения?

В таблицах dBASE или Paradox для создания нового индекса воспользуйтесь методом `AddIndex`. Для примера:

```
Table1.AddIndex('Articles', 'Title', []);
```

создаст индексный файл с именем «`Articles`», при этом поле `TITLE` выступает в качестве индексного ключа. При создании можно задавать различные индексные опции (например, указать, что файл уникальный, необслуживаемый и т. д.); для получения дополнительной информации обратитесь к электронной справке по Delphi.

Примечание

Таблица должна быть открыта исключительно для того, чтобы воспользоваться методом `AddIndex`.

Поддержка и обновление индексного файла, если только при создании не установлен флаг «необслуживаемый», происходит автоматически.

Создание таблицы с автоинкрементальным полем

Как создать таблицу с автоинкриментальным полем?

Допустим, имеется форма с кнопкой. Щелчок по кнопке должен посредством `DbiCreateTable` создать таблицу Paradox с автоинкрементальным (приращиваемым) полем.

```
unit Autoinc;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, DBTables, Db, Grids, DBGrids, DBITypes;
```

```
const
  szTblName = 'CR8PXTBL';       { Имя создаваемой таблицы. }
  szTblType = szParadox;        { Используемый тип таблицы. }

{ При создании таблицы используется полное описание поля }
const
  fldDes: array[0..1] of FLDDesc = (
  ({ Поле 1 - AUTOINC}
    iFldNum: 1;                 { Номер поля }
    szName: 'AUTOINC';          { Имя поля }
    iFldType: fldINT32;         { Тип поля }
    iSubType: fldstAUTOINC;     { Подтип поля }
    iUnits1: 0;                 { Размер поля }
    iUnits2: 0;                 { Десятичный порядок следования (0) }
    iOffset: 0;                 { Смещение в записи (0) }
    iLen: 0;                    { Длина в байтах (0) }
    iNullOffset: 0;             { Для Null-битов (0) }
    efldvVchk: fldvNOCHECKS;    { Проверка корректности (0) }
    efldrRights: fldrREADWRITE { Права }
  ),
  ({ Поле 2 - ALPHA}
    iFldNum: 2;
    szName: 'ALPHA';
    iFldType: fldZSTRING;
    iSubType: fldUNKNOWN;
    iUnits1: 10;
    iUnits2: 0;
    iOffset: 0;
    iLen: 0;
    iNullOffset: 0;
    efldvVchk: fldvNOCHECKS;
    efldrRights: fldrREADWRITE
  ));

type
  TForm1 = class(TForm)
    Button1: TButton;
    Database1: TDatabase;
    procedure Button1Click(Sender: TObject);
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
var
  TblDesc: CRTblDesc;
```

```
    uNumFields: Integer;
    Rslt: DbiResult;
    ErrorString: Array[0..dbiMaxMsgLen] of Char;
  begin
    FillChar(TblDesc, SizeOf(CRTblDesc), #0);
    lStrCpy(TblDesc.szTblName, szTblName);
    lStrCpy(TblDesc.szTblType, szTblType);
    uNumFields := Trunc(SizeOf(fldDes) / SizeOf (fldDes[0]));
    TblDesc.iFldCount := uNumFields;
    TblDesc.pfldDesc := @fldDes;
    Rslt := DbiCreateTable(Database1.Handle, TRUE, TblDesc);
    if Rslt <> dbiErr_None then begin
      DbiGetErrorString(Rslt, ErrorString);
      MessageDlg(StrPas(ErrorString), mtWarning, [mbOk], 0);
    end;
  end;
  end.
```

Примечание

Необходимо «бросить» на форму компонент Database. *Его свойству* AliasName *назначаем действующий псевдоним, активируем* Database – Connected = True. *В результате получаем таблицу* CR8PXTBL.DB *там, куда указывает псевдоним.*

Создание и удаление полей во время выполнения программы

Компоненты TField (точнее, потомки компонента TField с соответствующим типом поля) могут создаваться во время проектирования программы с помощью Fields Editor. Fields Editor вызывается двойным щелчком на пиктограмме компонента TTable или TQuery. Но потомки TField могут быть созданы и удалены и в режиме выполнения программы.

Потомки компонента TField (такие как, например, TStringField, TIntegerField и др.) создаются методом Create для того типа потомка TField, который подходит к соответствующему полю набора данных. Другими словами, для поля строкового типа текущего набора данных необходимо вызвать метод Create класса TStringField, являющегося потомком TField. Методу Create необходим один параметр – владелец потомка TField, расположенный на TForm. После создания компонента наследника TField, для того чтобы новый экземпляр объекта мог установить связь с необходимым полем набора данных, необходимо установить несколько ключевых свойств. Вот их список:

- FieldName: имя поля в таблице
- Name: уникальный идентификатор компонента-потомка TField
- Index: позиция компонента-потомка TField в массиве TFields (свойство Fields компонента TTable или TQuery, с которым будет связан TField)
- DataSet: компонент TTable или TQuery, с которым будет связан TField

Приведенный ниже код демонстрирует способ создания поля TStringField. Форма названа Form1 (здесь ссылка на переменную Self), активному набору данных TQuery присвоено имя Query1, а поле, для которого создается компонент TStringField, расположено в таблице dBASE с именем CO_NAME. Новый потомок TField будет вторым TField в свойстве-массиве Fields компонента Query1. Имейте в виду, что набор данных, связанный с новым потомком TField (в нашем случае Query1), перед добавлением TField должен быть закрыт, а после добавления вновь открыт.

```
procedure TForm1.Button1Click(Sender: TOObject);
var
  T: TStringField;
begin
  Query1.Close;
  T := TStringField.Create(Self);
  T.FieldName := 'CO_NAME';
  T.Name := Query1.Name + T.FieldName;
  T.Index := Query1.FieldCount;
  T.DataSet := Query1;
  Query1.FieldDefs.UpDate;
  Query1.Open;
end;
```

Вышеприведенный пример создает новый TStringField с именем Query1CO_NAME.

Для удаления существующего потомка TField достаточно вызова метода Free данного компонента. В примере, приведенном ниже, метод TForm FindComponent используется для получения указателя на компонент TStringField с именем Query1CO_NAME. Возвращаемое функцией FindComponent значение в случае успешного завершения будет иметь тип TComponent или NIL в противном случае. По возвращаемому значению можно определить, действительно ли существует компонент до того, как будет применен метод Free.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  TC: TComponent;
begin
  TC := FindComponent('Query1CO_NAME');
  if not (TC = nil) then begin
    Query1.Close;
    TC.Free;
    Query1.Open;
  end;
end;
```

Как и при создании TField, набор данных, связанный с потомком TField и активный в настоящий момент, перед вызовом данного метода должен быть закрыт и впоследствии вновь активирован.

Восстановление записи dBASE

Возможно ли восстановить запись из таблицы dBASE после ее удаления? Могли бы вы дать пример использования функции?

Предположим, на форме имеется кнопка (с именем «butRecall»), восстанавливающая текущую отображаемую (или позиционируемую курсором) запись. Данный код, будучи расположенным в обработчике события кнопки OnClick, это демонстрирует:

```
function GetTableCursor(oTable: TTable): hDBICur;

var
  szTable: Array [0..78] of Char;
begin
  StrPCopy(szTable, oTable.TableName);
  DbiGetCursorForTable(oTable.DBHandle, szTable, nil, Result);
end;

function dbRecall(oTable: TTable): DBIResult;
begin
  Result := DbiUndeleteRecord(GetTableCursor(oTable));
end;

procedure TForm1.butRecallClick(Sender: TObject);
begin
  if dbRecall(Table1) <> DBIERR_NONE then
    ShowMessage('Не могу восстановить запись!');
end;
```

Обработка исключения Index not found

Как мне открыть таблицу dBASE без требуемого MDX-файла? Я постоянно получаю исключение «Index not found...» (индекс не найден).

Во время создания таблицы dBASE с production-индексом (MDX) в заголовке DBF-файла устанавливается специальный байт. При последующем открытии таблицы dBASE драйвер читает этот специальный байт и, если он установлен, то пытается также открыть файл MDX. Если попытка открыть файл MDX заканчивается неудачей, возникает исключительная ситуация.

Для решения этой проблемы необходимо обнулить 28 байт в файле DBF, устраняющий зависимость таблицы от MDX-файла.

Нижеприведенный модуль является простым примером того, как можно обработать исключение при открытии таблицы, обнулив этот байт в DBF-файле и вновь открыв таблицу.

```
unit Fixit;

interface
```

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Db, DBTables;

type
  TForm1 = class(TForm)
    Table1: TTable;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

const
  TheTableDir = 'c:\temp\';
  TheTableName = 'animals.dbf';

{ В качестве параметра имя файла DBF }
procedure RemoveMDXByte(dbFile: String);
const
  Value: Byte = 0;
var
  F: File of byte;
begin
  AssignFile(F, dbFile);
  Reset(F);
  Seek(F, 28);
  Write(F, Value);
  CloseFile(F);
end;

{ Если файл .MDX не найден, DBF-файл исправляется и управление вновь передается
  данной процедуре для повторного открытия таблицы, но уже без MDX }
procedure TForm1.Button1Click(Sender: TObject);
begin
  try
{ устанавливаем каталог таблицы }
    Table1.DatabaseName := TheTableDir;
{ устанавливаем имя таблицы }
    Table1.TableName := TheTableName;
{ пытаемся открыть таблицу }
    Table1.Open;
  except
    on E: EDBEngineError do
{ Нижеследующее сообщение указывает на то, что файл MDX не найден: }
      if Pos('Index does not exist. File', E.Message) > 0 then begin
```

```
{ Сообщаем пользователю о наличии проблемы. }
        MessageDlg('Файл MDX не найден. Попытка открытия без индекса.',
                   mtWarning, [mbOk], 0);
{ удаляем байт MDX из заголовка таблицы }
        RemoveMDXByte(TheTableDir + TheTableName);
{ Посылаем кнопке сообщение для эмуляции ее нажатия. Этот трюк заставит данную
  процедуру выполниться повторно, и таблица будет открыта без файла MDX }
        PostMessage(Button1.Handle, cn_Command, bn_Clicked, 0);
      end;
  end;
end;
end.
```

Создание кросс-таблиц

Мне нужна помощь по реализации запроса кросс-таблицы в Delphi. Как это сделать?

Использовать pivot-таблицу должен все тот же общий механизм (относительно любой базы данных SQL).

Предположим, что есть данные продаж в таблице с полями Store, Product, Month, Sales, и необходимо отображать данные по продуктам за каждый месяц. (Примем, что поле Month для простоты имеет значения 1..12.)

Оригинальные данные примера:

Store	Product	Month	Sales
#1	Toys	1	100
#2	Toys	1	68
#1	Toys	2	150
#1	Books	1	75
...	...	...	...

Отчет должен быть похож на этот:

Product	January	February	March	...
Toys	168	150	...	...
Books	75	...	...	...

Установите pivot таблицу с именем tblPivot и 12 строками:

pvtMonth	pvtJan	pvtFeb	pvtMar	pvtApr	...
1	1	0	0	0	...
2	0	1	0	0	...
3	0	0	1	0	...
4	0	0	0	1	...
...	...	...	...	...	...

Теперь запрос, выполненный в виде:

```
select Product,
  January = sum(Sales * pvtJan),
  February = sum(Sales * pvtFeb),
  March = sum(Sales * pvtMar),
  April = sum(Sales * pvtApr),
  ...
where Month = pvtMonth
group by Product
```

даст вам информацию, опубликованную выше.

Поскольку pivot-таблица имеет только 12 строк, большинство SQL-серверов сохранят результат в кэш-памяти, так что скорость выполнения запроса будет весьма высокой.

Создание уникального ID для новой записи

Как создать уникальный индекс для поля?

Существует несколько способов задавать в таблице уникальный ID. Здесь ограничимся двумя.

1. Можно использовать автоинкрементальное поле. Этот метод не очень надежен. Если таблица каким-то образом испортится и понадобится ее пересобрать, автоинкрементные поля будут перенумерованы. Хотя это легкий способ для ситуации, когда нет ссылки на ID в других таблицах, но это не очень мудрое решение в других случаях.

2. Можно использовать ID-таблицу. Если имеется приложение, где нескольким таблицам необходимы уникальные ID, создайте ID-таблицу (IDTable) с двумя полями: Name (первичный ключ) и Last_Id. В методе BeforePost таблицы, которой необходим уникальный ID, сделайте так:

```
procedure TForm1.Table1BeforePost(DataSet: TDataSet);
var
  Id: Integer;
```

```
begin
{ проверяем, существует ли ID для этой записи }
  if Table1.Fields[0].AsInteger = 0  then begin
{ ищем имя таблицы в ID-Таблице }
    IDTable.FindKey([Name]);
{ извлекаем последний ID - подразумеваем блокировку записи }
    Id := IDTable.FieldByName('Last Id').AsInteger;
    Inc(Id);
{ записываем новый ID в ID-таблицу - подразумеваем разблокировку таблицы }
    IDTable.FieldByName('Last Id').AsInteger := Id;
    IDTable.Post;
{ записываем извлеченный ID в вашу таблицу }
    Table1.Fields[0].AsInteger := Id;
  end;
end;
```

Поместив этот код в обработчик события BeforePost, можно убедиться, что все ID будут последовательными. Недостаток: если пользователь во время попытки добавления новой записи вдруг передумает, то получится запись только с заполненным полем ID.

Для того чтобы воспользоваться данным способом (последовательные ID), поместите приведенный выше код в обработчик события таблицы OnNewRecord.

3. Можете использовать ID-файл.

Руководствуйтесь теми же принципами, что и в предыдущем способе, но вместо ID-таблицы создается ID-файл. Это дает преимущество за счет более высокой скорости работы, но в многопользовательской среде нужно заботиться о блокировке записей.

Таблицы в оперативной памяти

Как создать таблицу в оперативной памяти?

Таблицы InMemory являются характеристикой Borland Database Engine (BDE). Они создаются в RAM и удаляются при их закрытии. Работают они значительно быстрее и очень полезны в случае, если нужны быстрые операции в небольших таблицах. Данный пример использует вызов функции BDE DbiCreateInMemoryTable. Данный объект должен работать наподобие простой регулярной таблицы, за исключением того, что таблицы InMemory не поддерживают некоторые функции (типа проверки целостности, вторичных индексов и BLOB-полей), и в настоящее время данный код не содержит механизма обработки ошибок. Вероятно, при попытке создания Memo-поля будет получена ошибка.

```
unit Inmem;
interface
uses
  DBTables, WinTypes, WinProcs, DBITypes, DBIProcs, DB, SysUtils;
```

```
type
  TInMemoryTable = class(TTable)
  private
    hCursor: hDBICur;
    procedure EncodeFieldDesc(var FieldDesc: FLDDesc;
    const Name: string; DataType: TFieldType; Size: Word);
    function CreateHandle: HDBICur; override;
  public
    procedure CreateTable;
  end;
implementation
{ Эта функция виртуальная. Поскольку мы уже имеем дескриптор таблицы, то мы просто
  возвращаем его }
function TInMemoryTable.CreateHandle;
begin
  Result := hCursor;
end;
procedure TInMemoryTable.EncodeFieldDesc(var FieldDesc: FLDDesc;
            const Name: string; DataType: TFieldType; Size: Word);
const
  TypeMap: array[TFieldType] of Byte = (
    fldUNKNOWN, fldZSTRING, fldINT16, fldINT32, fldUINT16, fldBOOL,
    fldFLOAT, fldFLOAT, fldBCD, fldDATE, fldTIME, fldTIMESTAMP, fldBYTES,
    fldVARBYTES, fldBLOB, fldBLOB, fldBLOB
  );
begin
  with FieldDesc do begin
    AnsiToNative(Locale, Name, szName, SizeOf(szName) - 1);
    iFldType := TypeMap[DataType];
    case DataType of
      ftString,
       ftBytes,
     ftVarBytes,
        ftBlob,
        ftMemo,
      ftGraphic:
            iUnits1 := Size;
        ftBCD:
            begin
              iUnits1 := 32;
              iUnits2 := Size;
            end;
    end;
    case DataType of
      ftCurrency:  iSubType := fldstMONEY;
         ftBlob:  iSubType := fldstBINARY;
         ftMemo:  iSubType := fldstMEMO;
       ftGraphic:  iSubType := fldstGRAPHIC;
      end;
    end;
  end;
```

```
{ Эта функция ориентирована на использование DbiCreateInMemoryTable вместо
  DbiCreateTable. }

procedure TInMemoryTable.CreateTable;
var
  I: Integer;
  pFieldDesc: pFLDDesc;
  szTblName: DBITBLNAME;
  iFields: Word;
  Dogs: pfldDesc;
begin
  CheckInactive;
  if FieldDefs.Count = 0 then
    for I := 0 to FieldCount - 1 do
      with Fields[I] do
        if not Calculated then
          FieldDefs.Add(FieldName, DataType, Size, Required);
  pFieldDesc := nil;
  SetDBFlag(dbfTable, True);

  try
    AnsiToNative(Locale, TableName, szTblName, SizeOf(szTblName) - 1);
    iFields := FieldDefs.Count;
    pFieldDesc := AllocMem(iFields * SizeOf(FLDDesc));
    for I := 0 to FieldDefs.Count - 1 do
      with FieldDefs[I] do
        EncodeFieldDesc(PFieldDescList(pFieldDesc)^[I], Name, DataType, Size);

  { тип драйвера nil, т.к. поля логические }
      Check(DbiTranslateRecordStructure(nil, iFields, pFieldDesc, nil,
                                        nil, pFieldDesc));

  { здесь hCursor получает свое значение }
     Check(DbiCreateInMemTable(DBHandle, szTblName, iFields, pFieldDesc, hCursor));
    finally
      if pFieldDesc <> nil then FreeMem(pFieldDesc, iFields * SizeOf(FLDDesc));
      SetDBFlag(dbfTable, False);
    end;
  end;
end.
```

[News Group]

Примечание

Пример создан для работы в Delphi1. При адаптации к 32-битным версиям Delphi необходимо учесть, что определение `TFieldType` *изменилось: в него добавлены новые типы полей; перестали существовать или изменены некоторые функции и т. д.*

Проблема медленного доступа к таблице

У меня очень медленный доступ к таблице при первом обращении. Как решить эту проблему?

Данная проблема возникает из-за того, что BDE вначале запрашивает базу данных для получения информации о таблице, прежде чем он начнет с ней работать. Как только появляется информация о таблице, она кэшируется и обращения к таблице во время всего сеанса (пока `TDatabase.Connection` имеет значение `True`) происходят практически мгновенно. Для того чтобы использовать кэшируемую информацию и при последующем запуске приложения, в конфигурации BDE найдите необходимый псевдоним и установите `SHEMA CACHE = TRUE` и `SHEMA CACHE DIR = 'C:\TEMP'` или любой другой удобный каталог.

Примечание

При любом изменении структуры таблицы придется удалять кэш вручную. Имя файла, в котором хранится кэш, можно узнать, посмотрев в любом текстовом редакторе файл SCache.INI.

Есть еще параметр `SHEMA CACHE TIME`*, значение которого устанавливает периодичность обновления информации о структуре БД.*

Проблема загрузки DBCLIENT.DLL

Я включил DBCLIENT.DLL *в секцию* ADDITIONAL FILES *опций распространения по WEB, но этот файл никогда не загружается на клиента. Как это исправить?*

INF-файл должен включать в себя примерно такие строки:

```
[Add.Code]
dbclient.dll=dbclient.dll

[dbclient.dll]
file=http://yoursite.com/dbclient.cab
clsid={9E8D2F81-591C-11D0-BF52-0020AF32BD64}

RegisterServer=yes

FileVersion=4,0,0,36
```

Замените `yoursite` вашим http-адресом, по которому находится CAB-файл. `FileVersion` — это версия файла в вашем архиве CAB (проверьте информацию о версии библиотеки `dbclient`, чтобы быть уверенным в соответствии). Убедитесь, что `FileVersion` относится к версии вашей библиотеки DBCLIENT.DLL. Последнюю можно поместить в CAB-файл с помощью утилиты CABARC, расположенной в папке \Delphi\BIN. Команда вызова CABARC может выглядеть примерно так:

```
CABARC N DBCLIENT.CAB
```

Хитрости многопользовательского доступа к базам данных

Некоторые хитрости, знание которых может быть полезным в разработке баз многопользовательского доступа:

В модуле DBIPROCS Delphi 1.0 и в BDE.INT Delphi 2.0 существует функция с именем DBISetLockRetry(n).

Синтаксис – DBISetLockRetry(n), где n – продолжительность ожидания перед повторной попыткой вставки (в секундах), редактирования или другой операцией с таблицей. DBISetLockRetry(-1) будет бесконечно пытаться получить доступ к вашей таблице.

Хорошее место для вызова функции – обработчик события TableAfterOpen. В этом случае все, что нужно сделать, это:

```
DBISetLockRetry(x);
```

Работая с Delphi 1.0, не забудьте включить в вашу программу DBIProcs. В Delphi 2.0 включите BDE.

Эти требования обязательны при разработке многопользовательских приложений Delphi, работающих с файлами dBASE или Paradox.

Дубликат записи Paradox или dBASE

Существует ли какое-либо простое решение для приложения Delphi, позволяющее прочесть запись из первого поля таблицы Paradox (первичный ключ) и потом обратно добавить ее в таблицу в виде новой записи?

Это работает на том «основании», что как Table1, так и Table2 ссылаются на один и тот же табличный файл. Вы могли бы осуществить это и с единственным TTable, если бы сохранили содержание в активном буфере.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  p: CURProps;
begin
  Table2.Insert;
  DbiGetCursorProps(Table1.Handle, p);
  Move(Table1.ActiveBuffer^, Table2.ActiveBuffer^, p.iRecBufSize);
  Table2.FieldByName('ID').AsInteger := Table1.FieldByName('ID').AsInteger + 1;
  Table2.Post;
end;
```

Имя пользователя базы данных Paradox

Как можно получить имя пользователя базы данных Paradox?

Можно выполнить эту задачу, непосредственно обращаясь к BDE. Включите следующие модули в секцию Uses вашего модуля: Db, DBTables, BDE.

Ниже приведена функция с именем ID, возвращающая сетевое имя входа:

```
function ID: String;
var
  rslt: DBIResult;
  szErrMsg: DBIMSG;
  pszUserName: PChar;
begin
  try
    Result := '';
    pszUserName := nil;
    GetMem(pszUserName, SizeOf(Char) * DBIMAXXBUSERNAMELEN);
    rslt := DbiGetNetUserName(pszUserName);
    if rslt = DBIERR_NONE then Result := StrPas(pszUserName)
    else begin
      DbiGetErrorString(rslt, szErrMsg);
      raise Exception.Create(StrPas(szErrMsg));
    end;
    FreeMem(pszUserName, SizeOf(Char) * DBIMAXXBUSERNAMELEN);
    pszUserName := nil;
  except
    on E: EOutOfMemory do ShowMessage('Ошибка. ' + E.Message);
    on E: Exception do ShowMessage(E.Message);
  end;
  if pszUserName <> nil then
    FreeMem(pszUserName, SizeOf(Char) * DBIMAXXBUSERNAMELEN);
end;
```

Создание таблицы Paradox

Как создать таблицу Paradox из приложения?

Решение

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  with TTable.Create(Self) do begin
    DatabaseName := 'c:\windows\temp';
    TableName := 'FOO';
    TableType := ttParadox;
    with FieldDefs do begin
      Add('Age', ftInteger, 0, True);
      Add('Name', ftString, 25, False);
      Add('Weight', ftFloat, 0, False);
    end;
    IndexDefs.Add('MainIndex', 'Age', [ixPrimary, ixUnique]);
    CreateTable;
  end;

end;
```

Печать структуры таблицы Paradox

Как распечатать структуру таблицы Paradox?

Предполагается, что компоненту Table присвоено имя Table1.

```
procedure TForm1.Button1Click(Sender: TObject);
const
  FieldTypes: array[0..16] of String[10] = (
    'Unknown', 'String', 'Smallint', 'Integer', 'Word', 'Boolean', 'Float',
    'Currency', 'BCD', 'Date', 'Time', 'DateTime', 'Bytes', 'VarBytes', 'Blob',
    'Memo','Graphic');
var
  i, nX, nY, nHeight, nWidth: Integer;
  rtxtMetric: TTextMetric;
  s: array[0..3] of String[10];
begin
  with Table1.FieldDefs, Printer do begin
    Update;
    PrinterIndex := -1;
    Title := 'Структура ' + Table1.TableName;
    BeginDoc;
    nX := 0;
    nY := 0;
    GetTextMetrics(Canvas.Handle, rtxtMetric);
    nHeight := rtxtMetric.tmHeight;
    nWidth := rtxtMetric.tmAveCharWidth;
    for i := 0 to Count - 1 do begin
      s[0] := IntToStr(Items[i].FieldNo) + #9;
      s[1] := Items[i].Name + #9;
      s[2] := FieldTypes[Ord(Items[i].DataType)] + #9;
      s[3] := IntToStr(Items[i].Size);
      Canvas.TextOut(nX, nY, s[0]);
      Inc(nX, Length(s[0]) * nWidth);
      Canvas.TextOut(nX, nY, s[1]);
      Inc(nX, Length(s[1]) * nWidth);
      Canvas.TextOut(nX, nY, s[2]);
      Inc(nX, Length(s[2]) * nWidth);
      Canvas.TextOut(nX, nY, s[3]);
      nX := 0;
      nY := i * nHeight;
    end;
    EndDoc;
  end;
end;
```

Примечание

Для работы с принтером необходимо добавить в секцию uses *модуль* Printers.

Ускорение открытия таблицы Paradox

Таблица Paradox открывается очень медленно. В чем причина?

Попробуйте заблокировать файл перед попыткой открытия таблицы. Данная манипуляция перед открытием таблицы создаст файл PDOXUSER.LCK. После этого открытие таблиц будет более быстрым, особенно если их будут открывать, закрывать и снова открывать. После окончания удалите блокировку файла.

Пароли Paradox

Как при соединении с таблицей Paradox устранить/установить окошко с требованием ввести пароль, защищающий таблицу?

Свойство компонента Table.Active должно быть установлено в False (если она активна прежде, чем введен пароль, то получите это окно). Затем поместите следующий код в обработчике события формы OnCreate:

```
Session.AddPassword('Мой секретный пароль');
Table1.Active := true;
```

Замена пароля для таблицы Paradox из приложения

Как сменить пароль (master password) для таблицы Paradox?

Решение

```
var
  db: TDatabase;
  Desc: CRTblDesc;
begin
  db := Table1.OpenDatabase;
  FillChar(Desc, SizeOf(Desc), #0);
  StrCopy(Desc.szTblName, PChar(Table1.TableName));
  StrCopy(Desc.szTblType, szParadox);
  StrCopy(Desc.szPassword, 'password');
  Desc.bProtected := TRUE;
  Check(DbiDoRestructure(db.Handle, 1, @Desc, nil, nil, nil, FALSE));
end;
```

Особенность первичного индекса Paradox

Почему при создании таблицы Paradox с первичным нечувствительным к регистру индексом появляется ошибка?

В Paradox первичный индекс всегда CaseSensitive.

[Nomadic]

Создание поля autoincrement в таблицах Paradox

Согласно электронной документации по DBD, «автоприращиваемое» (Autoincrement) поле таблиц Paradox должно содержать значение Valcheck Minimum value.

Это отлично работает с новой таблицей, но вы не сможете добавить Autoincrement поле к существующей таблице, т. к. все значения Valchecks неактивны.

Решение – измените тип поля с «+» на «N», установите минимальное значение и восстановите тип поля на «+».

Доступ к файлам Paradox через BDE в сети Lantastic Network

Имею BDE, сеть Lantastic и работающие с ними приложения Delphi. На одной машине проблем не было, зато другие с BDE работать не смогли. В чем проблема?

Возможные варианты:

Решение 1

Первая машина имела BDE от фактической Database Engine, а не из Delphi.

В секции [386Enh] файла SYSTEM.INI была строка

```
network = lantasti.386
```

вместо правильной

```
network = *vnetbios,lantasti.386
```

Решение 2

Должно быть, проблема в файле LPICALLW.DLL. Если его удалить, то функциональность сети восстанавливается. Чтобы найти файл, загрузитесь без AUTOEXEC.BAT. Файл можно найти в директории \lantasti. Если к нему не установили путь типа PATH=C:\DOS, то при загрузке LPICALLW.DLL найден не будет. После этого система выведет диалоговое окно с сообщением о неудачной попытке найти файл LPICALLW.DLL. Нажмите кнопку OK, и сможете, используя Borland Delphi, Paradox или dBASE, получать доступ к файлам Paradox.

Решение 3

При выключении 32-битного доступа к файлам все заработало.

Изменение месторасположения NET-файла во время работы

Как изменить месторасположение файла PDOXUSRS.NET *во время выполнения программы?*

Для получения дескриптора сеанса, если используете сессию по умолчанию, необходимо вызвать DbiGetCurrSession.

```
DbiSetProp(hSessionHandle, sesNetFile, PChar('c:\newdir'));
```

Использование TClientDataSet в локальном приложении с таблицами Paradox

Как можно использовать TClientDataSet *в локальном приложении с таблицами Paradox, без применения компонентов* TProvider *и* TRemoteServer*?*

Вы не сможете отказаться от компонента TProvider, но можете использовать TClientDataSet в одноточечном (single-tier) приложении. Для того чтобы открыть Client DataSet, необходимо назначить провайдера данных вручную.

```
{ CDS = TClientDataSet }
{ Table1 = TTable }
CDS.Provider := Table1.Provider;
CDS.Open;
```

Также необходимо включить модуль DBClient в предложение Uses.

Чтение OLE из BLOB-поля Paradox

Говорят, что выполнить чтение OLE из BLOB-поля Paradox невозможно. Как поступать в данном случае?

Примечание

Предварительно на форму помещаем OleContainer *со страницы* SYSTEM *палитры компонентов.*

Решение

```
procedure TForm1.Button1Click(Sender: TObject);
var
  b: TBlobStream;
begin
  try
    b := TBlobStream.Create((Table1.FieldByName('OLE') as TBlobField), bmRead);
    OLEContainer1.LoadFromStream(b);
  finally
    b.free;
  end;
end;

procedure TForm1.Button2Click(Sender: TObject);
var
  b: TBlobStream;
begin
  try
    Table1.Insert;
    b := TBlobstream.Create((Table1.FieldByName('OLE') as TBlobField),
                            bmReadWrite);
    OLEContainer1.SaveToStream(b);
    Table1.Post;
```

```
  finally
    b.free;
  end;
end;
```

Проблемы работы с Paradox в одноранговой сети

Что нужно сделать для нормальной работы в одноранговой сети с базами Paradox?

В BDE Administrator с помощью команды Configuration|System|Init перейдите на вкладку Configuration и установите параметр LOCAL SHARE в True.

[Nomadic]

Проблемы работы с Paradox в сети

Я получаю ошибку приложения «... not initialized for accessing network files» (не инициализировано для доступа к сетевым файлам).

Программа Borland BDE Install не включает в себя автоматически драйвер для работы в сети для таблиц Paradox, если целевой компьютер подключен к сети. Пользователь получит сообщение об ошибке, если путь никем не установлен. Программы третьих фирм, устанавливающие BDE, поступают точно так же. Настройка сетевого каталога возможна программным путем из самой программы или с помощью пользователя и утилиты BDEconfig (BDE Administrator).

При запуске приложения разверните предусмотренную Borland библиотеку NETDIR.DLL (58 Кбайт), загруженную из форума PdoxWin, получите доступ к IDAPI.CFG и считайте значение сетевого каталога. Следующий код проверяет, был ли установлен сетевой каталог, и, если не был, то он временно устанавливается для текущего сеанса пользователя.

```
{ объявляем DLL-функцию }
function getCFGNetDir: pChar; far; external 'netdir' index 4;

{ проверяем и при необходимости восстанавливаем сетевой каталог }
procedure TForm1.FormCreate(Sender: TObject);
var
  theNetDir: PChar;
  theChar: Char;
begin
  theChar := ':';
  theNetDir := getCFGNetDir;
  if (StrScan(theNetDir, theChar) = nil) then Session.NetFileDir := 'C:\';
end;
```

Поля Byte в Paradox

Что за магия при записи в поле Paradox Byte*? По этому поводу в документации ничего не сказано.*

Есть два пути получить доступ к данным в `TBytesField`.

- Просто вызовите метод `GetData`, передавая ему указатель на буфер, где сам буфер должен иметь размер, достаточный для хранения данных. Для записи значений следует использовать `SetData`.

```
procedure SetCheckBoxStates;
var
  CBStates: array[1..13] of Byte;
begin
  CBStateField.GetData(CBStates);
  { Здесь обрабатываем данные... }
end;
```

- Используйте свойство `Value`, возвращающее вариантный массив байт (variant array of bytes).

```
procedure SetCheckBoxStates;
var
  CBStates: Variant;
begin
  CBStates := CBStateField.Value;
  { Здесь обрабатываем данные... }
end;
```

Первый метод, вероятно, будет легче, поскольку в этом случае можно воспользоваться уровнем байт. Запись данных получится сложнее, поскольку нужно будет работать с variant-методами типа `VarArrayCreate`.

Каскадное удаление с проверкой целостности

Как правильно выполнить удаление в связанных таблицах Paradox?

Таблицы Paradox имеют характеристику проверки целостности (`Referential Integrity`). Данная характеристика предотвращает добавление записей в дочернюю таблицу, для которых нет соответствующих записей в родительской таблице. Она также изменяет ключевое поле в дочерней таблице при изменениях в соответствующем ключевом поле родительской таблицы (обычно это называют каскадным обновлением). Эти события происходят автоматически и не требуют никакого вмешательства со стороны Delphi-приложений, использующих эти таблицы. Тем не менее, функция проверки целостности таблиц Paradox не работает с каскадным удалением. То есть, Delphi не позволит удалять записи в родительской таблице при наличии существующих записей в дочерней таблице. Это могут сделать только дочерние записи «без родителей», обходя проверку целостности. При попытке удаления такой родительской записи Delphi генерирует исключительную ситуацию.

Для того чтобы каскадное удаление дало эффект, требуется программное удаление соответствующих дочерних записей прежде, чем будет удалена родительская запись. В приложениях Delphi это делается посредством прерывания удаления записи в родительской таблице, удаления соответствующих записей в дочерней таблице (если таковая имеется), и затем окончательного удаления родительской записи.

Удаление записи таблицы осуществляется вызовом метода `Delete` компонента `TTable`, который удаляет текущую запись в связанной с компонентом таблице. Прерывание процесса удаления для выполнения других операций связано с созданием обработчика события `BeforeDelete` компонента `TTable`. Любые действия в обработчике события `BeforeDelete` произойдут прежде, чем приложением будет послана команда Borland Database Engine (BDE) на физическое удаление записи из табличного файла.

Для того чтобы обработать удаление одной или более дочерних записей, в обработчике события `BeforeDelete` необходимо организовать цикл, осуществляющий вызов метода `Delete` компонента `TTable` для всех записей дочерней таблицы. Цикл основан на условии, что указатель на запись в таблице позиционируется на следующую дочернюю запись `TTable` или на конец набора данных. Это также предполагает, что удаляются все дочерние записи, соответствующие родительским записям: если нет соответствующих записей, указатель на запись устанавливается на конец набора данных, условие выполнения цикла равно `False`, и метод `Delete` в теле цикла никогда не выполняется.

```
procedure TForm1.Table1BeforeDelete(DataSet: TDataset);
begin
  with Table2 do begin
    DisableControls;
    First;
    while not EoF do Delete;
    EnableControls;
  end;
end;
```

В вышеуказанном примере родительская таблица представлена компонентом `TTable` с именем `Table1` и дочерней таблицей с именем `Table2`. Методы `DisableControls` и `EnableControls` использованы в «косметических» целях, чтобы «заморозить» любые компоненты для работы с базами данных, которые могли бы отображать данные из таблицы `Table2` во время удаления записей. Эти два метода делают процесс визуально «гладким» и не являются обязательными. Метод `Next` в теле данного цикла вызываться не должен. Дело в том, что цикл начинается с первой записи и, поскольку каждая запись удаляется, запись, предшествующая удаленной, перемещается в наборе данных вверх, становясь одновременно первой и текущей записью.

Данный пример предполагает, что родительская и дочерняя таблицы связаны отношением Master-Detail, типичным для таблиц, для которых работает проверка целостности. В результате, в связанных таблицах становятся дос-

тупны только те записи дочерней таблицы, которые соответствуют текущей записи родительской таблицы. Все другие записи в дочерней таблице делаются недоступными через фильтрацию Master-Detail. Если таблицы не связаны отношениями Master-Detail, то есть два дополнительных замечания, которые необходимо принимать во внимание при удалении записи дочерней таблицы.

Первое: вызов метода First не переместит указатель записи в запись, соответствующей текущей записи родительской таблицы. Для этого необходимо воспользоваться методом FindKey, вручную перемещающий указатель на сопоставимую запись.

Второе замечание относится к условию цикла. Поскольку будут доступны другие записи, не относящиеся к записям родительской таблицы, то перед удалением записи необходимо осуществлять проверку на ее сопоставимость. Эта проверка должна проводиться дополнительно к методу EoF. Поскольку записи будут сортироваться по ключевому полю (первичного или вторичного индекса), все сопоставления (Master-Detail) будут непрерывными. Это будет истиной до достижения первой несопоставимой записи, после чего можно считать, что все сопоставимые записи были удалены.

Таким образом, предыдущий пример необходимо изменить следующим образом:

```
procedure TForm1.Table1BeforeDelete(DataSet: TDataset);
begin
  with Table2 do begin
    DisableControls;
    FindKey([Table1.Fields[0].AsString])
    while (Fields[0].AsStrring = Table1.Fields[0].AsString)
         and (not EoF) do Delete;
    EnableControls;
  end;
end;
```

В приведенном выше примере Table1 – первое поле родительской таблицы, на которой базируется проверка целостности, а Table2 – первое поле дочерней таблицы (Table2), с которым производится сопоставление.

Проблема транзакций

Если в транзакции изменена какая-то таблица, то для другого пользователя блокируется вся таблица до окончания транзакции. Как решить проблему?

По умолчанию оператор UPDATE в MS SQL Server пытается поставить эксклюзивную табличную блокировку. Это можно обойти при помощи ключевого слова FROM в сочетании с опцией PAGLOCK для использования MS SQL Server страничных блокировок вместо эксклюзивной табличной блокировки:

```
UPDATE orders SET customer_id=NULL FROM orders(PAGLOCK) WHERE customer_id=32;
```

Блокировка на всю таблицу при UPDATE ставится только в том случае, если по предикату нет индекса. Так можно просто проиндексировать таблицу ORDERS по полю CUSTOMER_ID и не забывать выполнять UPDATE STATISTIC, хотя это будет работать и с PAGLOCK. Просто не факт, что UPDATE всегда устанавливает табличную блокировку.

[Nomadic]

Пакование таблиц Paradox

Возвожно ли перестраивать и паковать таблицы Paradox из программ, написанных на Delphi?

Проверьте работу приведенной ниже функции, она пакует таблицы Paradox и dBase (требуется компонент TDatabase, указывающий на ту же директорию, где хранятся таблицы):

```
uses
  BDE;

function PackTable(tbl:TTable; db:TDatabase): DBIResult;
var
  crtd: CRTblDesc;
begin
  Result := DBIERR_NA;
  with tbl do if Active then Active := False;
  with db do if not Connected then Connected := True;
  FillChar(crtd,SizeOf(CRTblDesc), 0);
  StrPCopy(crtd.szTblName, tbl.TableName);
  crtd.bPack := True;
  Result := DbiDoRestructure(db.Handle, 1, @crtd, nil, nil, nil, FALSE);
end;
```

Пример использования:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if PackTable(Table1, DataBase1) = DBIERR_NONE then ...
  else MessageBeep(0);
end;
```

Вызов TUTILITY

Покажите пример вызова TUTILITY.DLL *из Delphi.*

Решение

```
...
var
  Session: hTUses;
  i: integer;
```

```
    ErrorCode: word;
    ResultCode: word;

    procedure BdeError(ResultCode: Word);
    begin
      if ResultCode <> 0 then
        Raise Exception.CreateFmt('BDE ошибка %x', [ResultCode]);
    end;

  begin
    try
      BdeError(DbiInit(nil));
      BdeError(TUInit(@Session));
      for i := 1 to High(TableNames) do begin
        WriteLn('Проверка ' + TableNames[i]);
        ResultCode := TUVerifyTable(Session, @TableNames[i, 1], szParadox,
                                    'TABLERRS.DB', nil, TU_Append_Errors, ErrorCode);
        BdeError(ResultCode);
        if ErrorCode = 0 then WriteLn('Успешно')
        else WriteLn('ОШИБКА! -- Для информации смотри TABLERRS.DB!');
        WriteLn('');
      end;

    finally
      BdeError(TUExit(Session));
      BdeError(DbiExit);
    end;

  end;
```

Исключение показа поля

Как исключить показ поля P_RECNO?

Можно отредактировать TTable для исключения P_RECNO или установить:

```
TableX.FieldbyName('P_RECNO').Visible := False;
```

Это можно сделать также и с помощью редактора полей (Fields Editor), который связан напрямую с компонентом Table. Для вызова редактора щелкните правой кнопкой мыши на компоненте Table и выберите самый верхний пункт контекстного меню (Fields Editor...). Добавьте все поля в список полей и выделите то поле, которое вы не хотите показывать в DBGrid. Найдите в Инспекторе объектов свойство Visible и установите его в False. Если имеется компонент TTable, дважды щелкните на пиктограмме компонента (расположенной на форме). В диалоге получите список полей, имеющих отношение к соответствующей таблице. Щелкните на одном из полей и проверьте в Инспекторе объектов свойство Visible, оно должно быть установлено в False.

Поля DBGrid и Memo

Как из Memo-поля выбрать данные для DBGrid?

В обработчик события OnGetText компонента TMemoField поместите следующую строку:

```
Text := GrabMemoAsString(TMemoField(Sender));
```

и поместите следующую функцию так, чтобы к ней можно было свободно обратиться:

```
function GrabMemoAsString(TheField: TMemoField): String;
begin
  if TheField.IsNull then Result := ''
  else with TBlobStream.Create(TheField, bmRead) do begin
    if Size >= 255 then begin
      SetLength(Result, 255);
      Read(Result, 255);
    end else begin
      SetLength(Result, Size);
      Read(Result, Size);
    end;

    Free;
    while Pos(#10, Result) > 0 do Result[Pos(#10, Result)] := ' ';
    while Pos(#13, Result) > 0 do Result[Pos(#13, Result)] := ' ';
  end;
end;
```

Информация из одной таблицы на двух формах

Как информацию из одной таблицы отобразить в двух формах?

1. Добавьте на вторую форму (Form2) компонент TTable.
2. В режиме разработки присвойте этой таблице такие же значения, как и у таблицы, расположенной на Form1.
3. В секции implementation Form2 создайте следующий фрагмент кода:

```
uses Form1;
```

4. Подключите процедуру к событию OnCreate в Form2 (через Инспектор объектов).
5. Добавьте к этой процедуре следующую строку:

```
Table1 := Form1.Table1;
```

6. В режиме разработки свяжите все компоненты с Table1, расположенным на Form1.

Остается только решить проблему синхронизации. Сделайте следующее:

На Form1:

- разместите Table1;
- разместите DataSource1;
- установите DataSource1.DataSet := Table1;
- разместите DataGrid;
- установите DataSource := DataSource1.

На Form2:

- разместите DataSource1 (#1 для этой формы);
- разместите любые другие необходимые вам БД компоненты;
- укажите у них в качестве источника данных DataSource1;
- в обработчике события OnCreate для этой формы (например, FormCreate), поместите следующий код:

```
with Form1 do Form2.DataSource1.DataSet := Table1;
```

Данный код подключает Table1 на Form1 к DataSource от Form2.

После таких действий данные будут отображены на Form2 и будут «синхронизированы» с данными, отображаемыми на Form1 (поскольку в действительности используется одна таблица).

Единственное здесь предостережение – если используется TDatabase. Компонент TDatabase не обязателен для получения доступа к базам данных, но, тем не менее, он обеспечивает дополнительный контроль в приложениях клиент/сервер.

Таким образом, если приложение не работает в среде клиент/сервер, нет необходимости использовать TDatabase. Все, что нужно, – TDataSource, TTable и компоненты для работы с базами данных.

Копирование и удаление таблиц из приложения

Необходимые для работы модули: DB, DBTables, BDE.

Необходимо указать каталог расположения, исходное имя таблицы, каталог назначения и имя таблицы, куда будет скопирована исходная таблица, и BDE скопирует ее целиком со всеми индексами.

Процедура удаления в качестве входных параметров использует каталог расположения и имя таблицы, при этом BDE удаляет как саму таблицу, так и все файлы, связанные с ней (индексы и т. п.).

```
procedure TForm1.CopyTable(FromDir, SrcTblName, ToDir, DestTblName: String);
var
  DBHandle: HDBIDB;
  ResultCode: DBIResult;
  Src, Dest, Err: Array[0..255] of Char;
  SrcTbl, DestTbl: TTable;
```

```
begin
  SrcTbl := TTable.Create(Application);
  DestTbl := TTable.Create(Application);
  try
    SrcTbl.DatabaseName := FromDir;
    SrcTbl.TableName := SrcTblName;
    SrcTbl.Open;
    DBHandle := SrcTbl.DBHandle;
    SrcTbl.Close;
    ResultCode := DbiCopyTable(DBHandle, false, StrPCopy(Src, FromDir + '\'
                  + SrcTblName), nil, StrPCopy(Dest, ToDir + '\'
                  + DestTblName));
    if (ResultCode <> DBIERR_NONE) then begin
      DbiGetErrorString(ResultCode, Err);
      raise EDatabaseError.Create('При копировании ' + FromDir + '\'
              + SrcTblName + ' в ' + ToDir + '\' + DestTblName + ' ,'
              + 'BDE сгенерировал ошибку ''' +  StrPas(Err) + '''');
    end;
  finally
    SrcTbl.Free;
    DestTbl.Free;
  end;
end;

procedure TForm1.DeleteTable(Dir, TblName: String);
var
  DBHandle: HDBIDB;
  ResultCode: DBIResult;
  tbl, Err: Array[0..255] of Char;
  SrcTbl, DestTbl: TTable;
begin
  SrcTbl := TTable.Create(Application);
  try
    SrcTbl.DatabaseName := Dir;
    SrcTbl.TableName := TblName;
    SrcTbl.Open;
    DBHandle := SrcTbl.DBHandle;
    SrcTbl.Close;
    ResultCode := DbiDeleteTable(DBHandle, StrPCopy(Tbl, Dir + '\' + TblName),
                                 nil);
    if (ResultCode <> DBIERR_NONE) then begin
      DbiGetErrorString(ResultCode, Err);
      raise EDatabaseError.Create('Удаляя ' + Dir + '\' + TblName + ', BDE ' +
        'сгенерировал ошибку ''' + StrPas(Err) + '''');
      end;
  finally
    SrcTbl.Free;
  end;

end;
```

DBFSeek и DBFLocate

Как выполнить поиск в таблице?

Надежней и быстрее (если вы ищете отдельные записи) выполнить поиск строки с помощью Seek (если найдена первая запись) или Locate (индекс не требуется).

Решение

```
{ DBFSeek - поиск величины с использованием индекса - простой путь }
function DBFSeek(const Table1: TTable; const sValue: string): boolean;
var
  sExpValue: DBIKEYEXP;
  bmPos: TBookMark;
  nOrder: integer;
begin
  Result := False;
  with Table1 do begin
    if (Active) and (Length(IndexName) > 0) then begin
      bmPos := GetBookMark;
      DisableControls;
      StrPCopy(sExpValue, sValue);
      if (DbiGetRecordForKey(Handle, True, 0, StrLen(sExpValue),
                             @sExpValue, nil) = DBIERR_NONE) then Result := True
      else GotoBookMark(bmPos);
      FreeBookMark(bmPos);
      EnableControls;
    end;
  end;
end;

{ DBFLocate - поиск величины, не связанный с ключевым полем, замена из faq
  теперь принимает FieldName, искомая величина может быть частью словосочетания }
function DBFLocate(const Table1: TTable; const sFld, sValue: string): boolean;
var
  bmPos: TBookMark;
  bFound: boolean;
  len: integer;
begin
  Result := False;
  if (sValue <> '') and (sFld <> '') then begin
    with Table1 do begin
      DisableControls;
      bFound := False;
      bmPos := GetBookMark;
      len := Length(sValue);
      First;
      while not EOF do begin
        if FieldByName(sFld).AsString <> sValue then Next
        else begin
          Result := True;
```

```
            bFound := True;
            Break;
          end;
        end;
        if (not bFound) then GotoBookMark(bmPos);
        FreeBookMark(bmPos);
        EnableControls;
      end;
    end;
  end;

  procedure TForm1.Button1Click(Sender: TObject);
  begin
    Table1.UpdateCursorPos;
    if DBFSeek(Table1, xVal1) then begin ...
    /// делаем все, что необходимо
    if DBFLocate(Table1, 'CUSTNAME', xVal2) then begin ...
    /// делаем все, что необходимо
  end;
```

Выполнение запросов к базе данных в фоновом режиме

Данный документ объясняет, как выполнить запрос в фоновом режиме, используя класс TThread. Для получения общей информации о классе TThread, пожалуйста, обратитесь к документации Borland и электронной справке. Для понимания данного документа необходимо иметь представление о том, как обращаться с компонентами для работы с базами данных, поставляемыми с Delphi 2.0.

Для осуществления потокового запроса необходимо выполнение двух требований. Во-первых, потоковый запрос должен находиться в своей собственной сессии с использованием отдельного компонента TSession. Следовательно, на вашей форме должен находиться компонент TSession, имя которого должно быть назначено свойству SessonName компонента TQuery, выполняющего потоковый запрос. Для каждого используемого в потоке компонента TQuery необходим отдельный компонент TSession. В случае применения компонента TDataBase, для отдельного потокового запроса должен также использоваться отдельный TDataBase. Второе требование заключается в том, что компонент TQuery, используемый в потоке, не должен подключаться в контексте этого потока к TDataSource. Это должно быть сделано в контексте первичного потока.

Приведенный ниже пример кода иллюстрирует описываемый процесс. Данный модуль демонстрирует форму, которая содержит по два экземпляра компонентов TSession, TDatabase, TQuery, TDataSource и TDBGrid. Данные компоненты имеют следующие значения свойств:

```
Session1
  Active        True;
  SessionName   "Ses1"
```

```
DataBase1
  AliasName     "IBLOCAL"
  DatabaseName  "DB1"
  SessionName   "Ses1"

Query1
  DataBaseName   "DB1"
  SessionName    "Ses1"
  SQL.Strings    "Select * from employee"

DataSource1
  DataSet        ""

DBGrid1
  DataSource    DataSource1

Session2
  Active         True;
  SessionName    "Ses2"

DataBase2
  AliasName      "IBLOCAL"
  DatabaseName   "DB2"
  SessionName    "Ses2"

Query2
  DataBaseName   "DB2"
  SessionName    "Ses2"
  SQL.Strings    "Select * from customer"

DataSource2
  DataSet        ""

DBGrid1
  DataSource     DataSource2
```

Обратите внимание, что свойство DataSet обоих компонентов TDataSource первоначально никуда не ссылается. Оно устанавливается во время выполнения приложения и это проиллюстрировано в коде.

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Grids, DBGrids, DB, DBTables;

type
  TForm1 = class(TForm)
    Session1: TSession;
```

```
    Session2: TSession;
    Database1: TDatabase;
    Database2: TDatabase;
    Query1: TQuery;
    Query2: TQuery;
    DataSource1: TDataSource;
    DataSource2: TDataSource;
    DBGrid1: TDBGrid;
    DBGrid2: TDBGrid;
    GoBtn1: TButton;
    procedure GoBtn1Click(Sender: TObject);
  end;

  TQueryThread = class(TThread)
  private
    FSession: TSession;
    FDatabase: TDataBase;
    FQuery: TQuery;
    FDataSource: TDatasource;
    FQueryException: Exception;
    procedure ConnectDataSource;
    procedure ShowQryError;
  protected
    procedure Execute; override;
  public
    constructor Create(Session: TSession; DataBase: TDatabase; Query: TQuery;
                       DataSource: TDataSource); virtual;
  end;

var
  Form1: TForm1;

implementation
{$R *.DFM}

constructor TQueryThread.Create(Session: TSession; DataBase: TDatabase;
                                Query: TQuery; Datasource: TDataSource);
begin
  inherited Create(True);    // Создаем поток с состоянием suspendend
  FSession := Session;       // подключаем все privat-поля
  FDatabase := DataBase;
  FQuery := Query;
  FDataSource := Datasource;
  FreeOnTerminate := True;   // Устанавливаем флаг освобождения потока после
                             // его завершения
  Resume;                    // Продолжение выполнения потока
end;

procedure TQueryThread.Execute;
begin
  try
```

```
{ Выполняем запрос и подключаем источник данных к компоненту TQuery}
    FQuery.Open;
    Synchronize(ConnectDataSource);
  except
{ Ловим исключение и его дескриптор в контексте основного потока }
    FQueryException := ExceptObject as Exception;
    Synchronize(ShowQryError);
  end;
end;

procedure TQueryThread.ConnectDataSource;
begin
  FDataSource.DataSet := FQuery;               // Подключаем DataSource к TQuery
end;

procedure TQueryThread.ShowQryError;
begin
  Application.ShowException(FQueryException);  // Обрабатываем исключение
end;

procedure RunBackgroundQuery(Session: TSession; DataBase: TDataBase; Query: TQuery;
                             DataSource: TDataSource);
begin
{ Создаем экземпляр TThread с различными параметрами. }
  TQueryThread.Create(Session, Database, Query, DataSource);
end;

procedure TForm1.GoBtn1Click(Sender: TObject);
begin
{ Запускаем два отдельных запроса, каждый в своем потоке }
  RunBackgroundQuery(Session1, DataBase1, Query1, Datasource1);
  RunBackgroundQuery(Session2, DataBase2, Query2, Datasource2);
end;
end.
```

Метод `TForm1.GoBtn1Click` является обработчиком события нажатия кнопки. Данный обработчик события дважды вызывает процедуру `RunBackgroundQuery`, это случается при каждой передаче новых параметров компонентам для работы с базой данных. `RunBackgroundQuery` создает отдельный экземпляр класса `TQueryThread`, передает различные компоненты для работы с базой данных в его конструктор, который, в свою очередь, назначает их закрытым полям `TQueryThread`.

`TQueryThread` содержит две определенные пользователем процедуры: `ConnectDataSource` и `ShowQryError`. Первая из них связывает `FDataSource.DataSet` с `FQuery`. Тем не менее, это делается в первичном потоке с помощью метода `TThread.Synchronize`. А вторая – `ShowQryError` – обрабатывает исключение в контексте первичного потока, также используя метод `Synchronize`. Конструктор `Create` и метод `Execute` снабжены подробными комментариями.

Повторный запрос к таблице

Как создать новый запрос и скопировать туда точно такие же описания полей?

Скопируйте FieldDefs и выполните цикл по

```
FieldDefs.Items[i].CreateField(Owner);
```

[Nomadic]

Контроль изменения данных

Предположим, что пользователь изменил строковое поле в Null. Как тогда в обработчике OnUpdateData *можно определить, изменилось ли это поле на строку Null, или поле не было изменено?*

Используйте свойство NewValue класса TField при чтении второй записи (той, которая содержит изменения). Если возвращаемое значение (variant) пусто или не назначено, то поле не было модифицировано. Приведем немного иллюстрирующего кода:

```
var
  NewVal: Variant;
begin
  NewVal := DataSet.FieldByName('MyStrField').NewValue;
  if VarIsEmpty(NewVal) then ShowMessage('Field was not edited')
  else if VarIsNull(NewVal) then ShowMessage('Field was blanked out')
  else ShowMessage('New Field Value: ' + String(NewVal));
end;
```

[Nomadic]

Дублирование набора записей

Можно воспользоваться вторым объектом TTable, подключенным к той же таблице, или вызвать метод объект TTable DisableControls, внести изменения и вызвать EnableControls. Для сохранения той же позиции можно попробовать воспользоваться закладкой. Например, так:

```
procedure TMyForm.MakeChanges;
var
  aBookmark: TBookmark;
begin
  Table1.DisableControls;
  aBookmark := Table.GetBookmark;
  try

{ Ваш код }
  finally
    Table1.GotoBookmark(aBookmark);
    Table1.FreeBookmark(aBookmark);
```

```
    Table1.EnableControls;
  end;
end;
```

Ошибка при добавлении или изменении записей

Почему при добавлении или изменении записей в некоторых запросах возникает ошибка «Cannot modify a read-only dataset»?

Во-первых, свойство RequestLive должно быть установлено в True. Во-вторых, чтобы запрос был редактируемым, он должен удовлетворять требованиям, которые можно найти в файлах справки при поиске по маске «result set, editing».

[Nomadic]

Поиск величины при вводе

Каким способом можно производить поиск подходящих величин в момент ввода? Табличный курсор (визуально) должен перемещаться к наиболее подходящему значению при добавлении пользователем новых символов водимой величины.

Используйте в обработчике события OnChange:

```
with MainForm.PatientTable do begin

  { начинаем поиск имени }
  IndexName := 'Name';
  FindNearest([SearchFor.Text]);
end;
```

Код подразумевает, что индексу, по которому производится поиск, присвоено имя Name. Свяжите этот код с табличной сеткой, и курсор будет перескакивать на ближайшую запись, удовлетворяющую введенной пользователем информации.

Удаление и восстановление индексов

Как быстро добавить данные в таблицу, если она индексирована?

Часто приходится удалять индексы из таблицы для ускорения процесса закачки туда данных, потом данные приходится переиндексировать. Класс TDBWork призван облегчить этот процесс:

- procedure DropIndexes — удаляет все индексы таблицы, предварительно сохраняя их в файле <имя таблицы>.set в каталоге программы;
- procedure CreateIndexes — создает все индексы, читая информацию о них из файла <имя таблицы>.set. После успешного завершения операции данный файл удаляется.

```
unit DBWork;

interface
uses
  Forms, SysUtils, Db, DbTables, IniFiles;
type
  TindexInfo = record
    Name: String;
    Fields: String;
    DescFields: String;
    Options: TIndexOptions;
end;

type
  TDBWork = class
  private
    prgpath: String;
    ininame: String;
    tbl: TTable;
    IndInfo: array of TIndexInfo;
  public
    constructor Create; overload;
    constructor Create(t: TTable); overload;
    procedure OpenTable(t: TTable);
    procedure ReadIndexInfo;
    procedure DropIndexes;
    procedure CreateIndexes;
    procedure SaveToFile;
    procedure LoadFromFile;
end;

implementation

{ TDBWork }

constructor TDBWork.Create;
begin
  prgpath := ExtractFilePath(Application.ExeName);
  tbl := Nil;
end;

constructor TDBWork.Create(t: TTable);
begin
  prgpath := ExtractFilePath(Application.ExeName);
  OpenTable(t);
end;

procedure TDBWork.CreateIndexes;
var
  i: Integer;
  e: Boolean;
```

```
  a: Boolean;
begin
  LoadFromFile;
  with tbl do begin
    a := Active;
    if Active then Close;
    e := Exclusive;
    Exclusive := true;
    for i := 0 to Length(IndInfo) - 1 do
      if(IndInfo[i].Name <> '') then
        AddIndex(IndInfo[i].Name,IndInfo[i].Fields, IndInfo[i].Options,
                 IndInfo[i].DescFields);
     Exclusive := e;
     Active := a;
  end;
  DeleteFile(ininame);
end;

procedure TDBWork.DropIndexes;
var
  i: Integer;
  e: Boolean;
  a: Boolean;
begin
  SaveToFile;
  with tbl do begin
    a := Active;
    if Active then Close;
    e := Exclusive;
    Exclusive := true;
    for i := 0 to Length(IndInfo) - 1 do
      if(IndInfo[i].Name <> '') then DeleteIndex(IndInfo[i].Name);
    Exclusive := e;
    Active := a;
  end;
end;

procedure TDBWork.LoadFromFile;
var
  i, c: Integer;
  ini: TMemIniFile;
  Section: String;
begin
  ini := TMemIniFile.Create(ininame);
  c := ini.ReadInteger('Common', 'DefsCount', 0);
  SetLength(IndInfo, c);
  for i := 0 to c - 1 do begin
    Section := 'IndexDef #' + IntToStr(i);
    IndInfo[i].Name := ini.ReadString(Section, 'Name', '');
    IndInfo[i].Fields := ini.ReadString(Section, 'Fields', '');
    IndInfo[i].DescFields := ini.ReadString(Section, 'DescFields', '');
```

```
      IndInfo[i].Options := [];
      if ini.ReadBool(Section, 'ixPrimary', false) then
        IndInfo[i].Options := [ixPrimary];
      if ini.ReadBool(Section, 'ixUnique', false) then
        IndInfo[i].Options := IndInfo[i].Options + [ixUnique];
      if ini.ReadBool(Section, 'ixDescending', false) then
        IndInfo[i].Options := IndInfo[i].Options + [ixDescending];
      if ini.ReadBool(Section, 'ixCaseInsensitive', false) then
        IndInfo[i].Options := IndInfo[i].Options + [ixCaseInsensitive];
      if ini.ReadBool(Section, 'ixExpression', false) then
        IndInfo[i].Options := IndInfo[i].Options + [ixExpression];
      if ini.ReadBool(Section, 'ixNonMaintained', false) then
        IndInfo[i].Options := IndInfo[i].Options + [ixNonMaintained];
    end;
    ini.Free;
  end;

  procedure TDBWork.OpenTable(t: TTable);
  begin
    tbl := t;
    ReadIndexInfo;
    Ininame := prgpath + tbl.TableName + '.set';
  end;

  procedure TDBWork.ReadIndexInfo;
  var
    i: Integer;
  begin
    with tbl.IndexDefs do begin
      Update;
      SetLength(IndInfo,Count);
      for i := 0 to Count - 1 do begin
        IndInfo[i].Name := Items[i].Name;
        IndInfo[i].Fields := Items[i].Fields;
        IndInfo[i].DescFields := Items[i].DescFields;
        IndInfo[i].Options := Items[i].Options;
      end;
    end;
  end;

  procedure TDBWork.SaveToFile;
  var
    i: Integer;
    ini: TMemIniFile;
    Section: String;
  begin
    ini := TMemIniFile.Create(ininame);
    ini.WriteInteger('Common', 'DefsCount', Length(IndInfo));
    for i := 0 to Length(IndInfo) - 1 do begin
      Section := 'IndexDef #' + IntToStr(i);
      ini.WriteString(Section, 'Name', IndInfo[i].Name);
```

```
      ini.WriteString(Section, 'Fields', IndInfo[i].Fields);
      ini.WriteString(Section, 'DescFields', IndInfo[i].DescFields);
      ini.WriteBool(Section, 'ixPrimary', ixPrimary in IndInfo[i].Options);
      ini.WriteBool(Section, 'ixUnique', ixUnique in IndInfo[i].Options);
      ini.WriteBool(Section, 'ixDescending', ixDescending in IndInfo[i].Options);
      ini.WriteBool(Section, 'ixCaseInsensitive',
                    ixCaseInsensitive in IndInfo[i].Options);
      ini.WriteBool(Section, 'ixExpression', ixExpression in IndInfo[i].Options);
      ini.WriteBool(Section, 'ixNonMaintained',
                    ixNonMaintained in IndInfo[i].Options);
    end;
    ini.UpdateFile;
    ini.Free;
  end;

  end.
```

Пример использования

```
procedure TForm1.Button1Click(Sender: TObject);

var
  i: TDBWork;

begin
  i := TDBWork.Create;
  i.OpenTable(DM.Table1);
  i.DropIndexes;
  i.Free;
end;

procedure TForm1.Button2Click(Sender: TObject);

var
  i: TDBWork;

begin
  i := TDBWork.Create;
  i.OpenTable(DM.Table1);
  i.CreateIndexes;
  i.Free;
end;
```

[Kostin Slava]

Странности в работе AddIndex

При попытке использования `AddIndex` *я получаю ошибку* `'Invalid Index/Tag name. Index: cusname'`. *Но у меня нет никаких проблем с этим именем в случае применения* `DeleteIndex`. *В чем причина?*

Необходимо сделать так:

```
InvTbl.AddIndex('cusname', 'name', [ixCaseInsensitive]);
```

или так:

```
InvTbl.AddIndex('name', 'name', []);
```

Особенности работы с Update

Почему не всегда верно обновляются IndexDefs *по* Update*?*

Ошибка в VCL.

Помогает добавление строки fUpdated := False; в тело процедуры TIndexDefs.Update. Можно еще удалить владельца с помощью Free и создать снова.

[Nomadic]

Простой пример работы с базой данных из DLL

Это простейшая DLL, экспортирующая единственную функцию. Вызывающий ее оператор передает функции значение ключа и строку со значением. Функция открывает демонстрационную базу данных BIOLIFE, находит по ключу запись и добавляет строку после всех записей в поле Notes:

```
library Mydll;

uses
  DBTables;

function Modify(Key: Double; const Info: String): Boolean; export;
var
  Table: TTable;
  Stream: TBlobStream;
begin
  Table := TTable.Create(nil);
  Table.DatabaseName := 'D:\';
  Table.TableName := 'BIOLIFE';
  Table.TableType := ttParadox;
  Table.Open;
  if Table.FindKey([Key]) then begin
    Result := True;
    Table.Edit;
    Stream := TBlobStream.Create(TMemoField(Table.FieldByName('Notes')),
                                 bmReadWrite);
    Stream.Seek(0, 2);
    Stream.Write(Info[1], Length(Info));
    Stream.Free;
    Table.Post;
  end else Result := False;
```

```
  Table.Free;
end;
exports
  Modify;
begin
end.
```

Пример вызова из приложения:

```
function Modify(Key: Double; const Info: String): Boolean; far;
                external 'MYDLL';
  ...

 Modify(90200, 'Васек Трубачев');
```

Значение по умолчанию для объекта TField

Как мне задать выражение по умолчанию для объекта `TField`*?*

Значение по умолчанию можно задать, если уже установлены атрибуты поля и оно ассоциировано с полем таблицы. Если вы установили значение в Инспекторе объектов, т. е. задали строку, то не думайте, что во время выполнения приложения удастся получить значение по умолчанию. Если попытаться установить свойство `TField.DefaultExpression` примерно так:

```
MyField.DefaultExpression := 'MyValue';
```

то это будет скомпилировано, но при создании в таблице новой записи, скажем, при щелчке на кнопке + в `DBNavigator`, мы значения по умолчанию не получим. Чтобы во время работы приложения все работало, код должен быть таким:

```
MyField.DefaultExpression := '''MyValue''';
```

В Инспекторе объектов нужно поместить значение `'MyValue'` (используя одинарные кавычки).

Сохранение в базе данных файла формата JPEG

Как правильно показать на экране и сохранить в базе данных картинку формата JPEG?

Можно сделать, например, так:

```
if Picture.Graphic is TJPegImage then begin
  bs := TBlobStream.Create(TBlobField(Field), bmWrite);
  Picture.Graphic.SaveToStream(bs);
  bs.Free;
end else if Picture.Graphic is TBitmap then begin
  Jpg := TJPegImage.Create;
  Jpg.CompressionQuality := ...;
  Jpg.PixelFormat := ...;
```

```
    Jpg.Assign(Picture.Graphic);
    Jpg.JPEGNeeded;
    bs := TBlobStream.Create(TBlobField(Field), bmWrite);
    Jpg.SaveToStream(bs);
    bs.Free;
    Jpg.Free;
  end else
    Field.Clear;
```

[Nomadic]

Примечание

Вы не забыли объявить переменную bs *как* TBlobStream*?*

Автоматическая вставка SEQUENCE

Как при вводе информации в базу данных автоматически вставлять SEQUENCE?

Если добавление выполняется посредством оператора INSERT (в TQuery), то надо прямо там написать, как в плюсе ('... Values(My_seq.nextval, ...').

Если добавление осуществляется через TQuery с RequestLive = True, то в BeforeInsert сделайте запрос через TQuery (select myseq.nextval from dual) и записывайте значение в свое поле.

[Nomadic]

Запись и чтение чисел в поле BLOB

Мне нужно записать серию чисел в файл Paradox в BLOB-поле. Числа получаются из значений компонентов, размещенных на форме. Затем мне нужно будет считывать числа из поля BLOB и устанавливать согласно им значения компонентов. Как мне сделать это?

Можно начать свое исследование со следующего модуля:

```
unit BlobFld;

interface

uses
  Windows, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, DBTables, DB, ExtCtrls, DBCtrls, Grids,
  DBGrids, SysUtils;

type
  TFrmBlobFld = class(TForm)
    BtnWrite: TBitBtn;
    Table1: TTable;
    DataSource1: TDataSource;
    DBNavigator1: TDBNavigator;
```

```
    LbxDisplayBlob: TListBox;
    Table1pubid: TIntegerField;
    Table1comments: TMemoField;
    Table1UpdateTime: TTimeField;
    Table1Real1: TFloatField;
    Table1Real2: TFloatField;
    Table1Real3: TFloatField;
    Table1Curr1: TCurrencyField;
    Table1Blobs: TBlobField;
    Table1Bytes: TBytesField;
    CbxRead: TCheckBox;
    procedure BtnWriteClick(Sender: TObject);
    procedure DataSource1DataChange(Sender: TObject; Field: TField);
    procedure FormShow(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  end;
var
  FrmBlobFld: TFrmBlobFld;

implementation
{$R *.DFM}

type
  ADouble = array[1..12] of double;
  PADouble = ^ADouble;

procedure TFrmBlobFld.BtnWriteClick(Sender: TObject);
var
  i: integer;
  myBlob: TBlobStream;
  v: longint;
begin
  Table1.Edit;
  myBlob := TBlobStream.Create(Table1Blobs, bmReadWrite);
  try
    v := ComponentCount;
    myBlob.Write(v, SizeOf(longint));
    for i := 0 to ComponentCount - 1 do begin
      v := Components[i].ComponentIndex;
      myBlob.Write(v, SizeOf(longint));
    end;
  finally
    Table1.Post;
    myBlob.Free;
  end;
end;

procedure TFrmBlobFld.DataSource1DataChange(Sender: TObject; Field: TField);
var
  i: integer;
  myBlob: TBlobStream;
```

```
    t: longint;
    v: longint;
  begin
    if CbxRead.Checked then begin
      LbxDisplayBlob.Clear;
      myBlob := TBlobStream.Create(Table1Blobs, bmRead);
      try
        myBlob.Read(t, SizeOf(longint));
        LbxDisplayBlob.Items.Add(IntToStr(t));
        for i := 0 to t - 1 do begin
          myBlob.Read(v, SizeOf(longint));
          LbxDisplayBlob.Items.Add(IntToStr(v));
          end;
      finally
        myBlob.Free;
      end;
    end;
  end;

  procedure TFrmBlobFld.FormShow(Sender: TObject);
  begin
    Table1.Open;
  end;

  procedure TFrmBlobFld.FormClose(Sender: TObject; var Action: TCloseAction);
  begin
    Table1.Close;
  end;
  end.
```

Поля BLOB с длинным текстом

Как записать в BLOB-поле большой текст (больше 255 байт) из Delphi?

Можно так:

```
  var
    S: TBlobStream;
    B: pointer;
    c: integer;
    ...
    Table1.Edit;
    S := TBlobStream.Create(Table1BlobField as TBlobField, bmWrite);
    C := S.Write(B, C);
    Table1.Post;
    S.Destroy;
    ...
```

или так:

```
  var
    S: TMemoryStream;
```

```
  B: pointer;
  C: integer;
  ...

  S := TMemoryStream.Create;
  ...

  Table1.Edit;
  S.Clear;
  S.SetSize(C);
  C := S.Write(B, C);
  (Table1BlobField as TBlobField).LoadFromStream(S);
  S.Clear;
  Table1.Post;
  ...
  S.Destroy;
```

[Nomadic]

Запись потока в поле BLOB

Как выполнить запись потока в BLOB-поле?

Вся хитрость заключается в использовании StrPCopy (помещении строки в PChar) и записи буфера в поток. Нельзя передать строку в PChar непосредственно, поскольку для ее размещения нужен буфер. Получить необходимый размер буфера можно при помощи <BufferName>[0] и StrLen().

Пример использования TMemoryStream и записи его в поле Blob:

```
var
  cString: String;
  oMemory: TMemoryStream;
  Buffer: PChar;

begin
  cString := 'Ну, допустим, хочу эту строку!';
{ Создаем новый поток в памяти }
  oMemory := TMemoryStream.Create;
{ Копируем строку в PChar }
  StrPCopy(Buffer, cString);
{ Пишем =буфер= и его размер в поток }
  oMemory.Write(Buffer[0], StrLen(Buffer));
{ Записываем это в поле }
  <Blob/Memo/GraphicFieldName>.LoadFromStream(oMemory);
{ Необходимо освободить ресурсы}
  oMemory.Free;
end;
```

Загрузка изображений в поля BLOB

Как загрузить изображение в BLOB-поле?

Имеется несколько способов загрузки изображения в BLOB-поле таблицы dBASE или Paradox. Перечислим три самых простых метода:

- копирование данных из буфера обмена Windows в компонент `TDBImage`, связанный с полем `BLOB`;
- применение метода `LoadFromFile` компонента `TBlobField`;
- использование метода `Assign` для копирования объекта типа `TBitmap` в значение свойства `Picture` компонента `TBDBImage`.

Первый способ, в соответствии с которым происходит копирование изображения из буфера обмена, наиболее удобен, если требуется добавить изображение в таблицу и с приложением работает конечный пользователь. В этом случае компонент `TDBImage` выступает в роли интерфейса между BLOB-полем таблицы и изображением, хранящимся в буфере обмена. Метод `PasteFromClipboard` компонента `TDBImage` как раз и занимается тем, что копирует изображение из буфера обмена в `TDBImage`. При сохранении записи изображение записывается в BLOB-поле таблицы.

Поскольку буфер обмена Windows может содержать данные различных форматов, то желательно перед вызовом метода `CopyFromClipboard` осуществлять проверку формата хранящихся в нем данных. Для этого необходимо создать объект `TClipboard` и использовать его метод `HasFormat`, позволяющий определить формат хранящихся в буфере данных. Имейте в виду, что для создания объекта `TClipboard` необходимо добавить модуль `Clipbrd` в секцию `uses` того модуля, в котором будет создаваться экземпляр объекта.

Вот исходный код примера, копирующий содержание буфера обмена в компонент `TDBImage`, если содержащиеся в буфере данные имеют формат изображения:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  C: TClipboard;
begin
  C := TClipboard.Create;
  try
    if Clipboard.HasFormat(CF_BITMAP) then DBImage1.PasteFromClipboard
    else ShowMessage('Буфер обмена не содержит изображения!');
  finally
    C.Free;
  end;
end;
```

Второй способ заполнения BLOB-поля заключается в загрузке изображения непосредственно из файла. Данный способ одинаково хорош как при создании приложения (формирование данных), так и при его использовании.

Этот способ использует метод LoadFromFile компонента TBlobField, который применяется в Delphi для работы с dBASE-таблицами и двоичными Windows-полями или таблицами Paradox и графическими Windows-полями; в обоих случаях с помощью данного метода можно загрузить изображение и сохранить его в таблице.

Методу LoadFromFile компонента TBlobField необходим единственный параметр типа String: имя загружаемого файла с изображением. Значение данного параметра может быть получено при выборе файла пользователем с помощью компонента TOpenDialog и его свойства FileName.

Пример, демонстрирующий работу метода LoadFromFile компонента TBlobField с именем Table1Bitmap (поле с именем Bitmap связано с таблицей TTable, которой присвоено имя Table1):

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  Table1Bitmap.LoadFromFile('c:\delphi\images\splash\16color\construc.bmp');
end;
```

Третий способ для копирования содержимого объекта типа TBitmap в свойство Picture компонента TDBImage использует метод Assign. Объект типа TBitmap может быть как свойством Bitmap свойства объекта Picture компонента TImage, так и отдельного объекта TBitmap. Как и в методе, копирующем данные из буфера обмена в компонент TDBImage, данные изображения компонента TDBImage сохраняются в BLOB-поле после успешного сохранения записи. Ниже приведен пример, в котором задействован метод Assign. В нашем случае используется отдельный объект TBitmap. Для помещения изображения в компонент TBitmap был вызван его метод LoadFromFile.

```
procedure TForm1.Button3Click(Sender: TObject);
var
  B: TBitmap;
begin
  B := TBitmap.Create;
  try
    B.LoadFromFile('c:\delphi\images\splashh\16color\athena.bmp');
    DBImage1.Picture.Assign(B);
  finally
    B.Free;
  end;
end;
```

Извлечение изображения из поля BLOB

Простейший способ – применение метода Assign для сохранения содержимого BLOB-поля в объекте, имеющем тип TBitmap. Отдельный объект TBitmap или свойство Bitmap объекта Picture, в свою очередь являющегося свойством компонента TImage, могут служить примером совместимой цели для данной операции.

Пример кода, демонстрирующего использование метода Assign для копирования изображения из BLOB-поля в компонент TImage.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Picture.Bitmap.Assign(Table1Bitmap);
end;
```

В этом примере объект Table1Bitmap типа TBlobField представляет собой BLOB-поле таблицы dBASE. Данный TblobField-объект был создан с помощью редактора полей (Fields Editor). Если для создания полей таблицы Fields Editor не используется, то получить к ним доступ можно с помощью метода FieldByName или свойства Fields, оба они являются членами компонентов TTable или TQuery. В случае ссылки на BLOB-поле таблицы с помощью одного из приведенных членов, перед использованием метода Assign указатель на поле должен быть прежде приведен к типу объекта TBlobField.

Решение

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Picture.Bitmap.Assign(TBLOBField(Table1.Fields[1]));
end;
```

Изображение, хранящееся в BLOB-поле, может быть скопировано непосредственно в отдельный объект TBitmap. Ниже приведен пример, демонстрирующий создание объекта TBitmap и сохранения в нем изображения из BLOB-поля.

```
procedure TForm1.Button2Click(Sender: TObject);
var
  B: TBitmap;
begin
  B := TBitmap.Create;
  try
    B.Assign(Table1Bitmap);
    Image1.Picture.Bitmap.Assign(B);
  finally
    B.Free;
  end;
end;
```

Изображение и поля BLOB в InterBase

Таблицы dBASE и Paradox имеют в своем арсенале BLOB-поля, позволяющие хранить бинарные данные. В том числе формат Bitmap, отображаемый с помощью компонента TDBImage. В Database Desktop данный тип полей указан как Binary и Graphic (для таблиц dBASE и Paradox, соответственно). Тем не менее, процесс сохранения изображений в BLOB-полях InterBase и их использование в компонентах TDBImage не так уж просты.

Таблицы InterBase не имеют простого типа BLOB-поля. Есть три варианта, или подтипа: тип 0, тип 1 и подтип, определенный пользователем. Типы 0 и 1 – «встроенные» типы. Тип 0 – BLOB-поля (тип по умолчанию) для хранения общих бинарных данных. Тип 1 – BLOB-поля для хранения текстовых BLOB-данных. Ни один из предопределенных типов не допускает автоматического извлечения данных изображения из BLOB-поля для его последующего отображения в компоненте `TDBImage`. В BLOB-полях типа 0 можно хранить данные формата `Bitmap`, но данные должны извлекаться и передаваться в объект типа `TBitmap` программным путем. Пример ручного извлечения данных изображения, хранящихся в BLOB-поле типа 0 (`Table1BlobField`) и его отображение в компоненте `TImage` (не предназначенном для работы с базой данных):

```
procedure TForm1.ExtractBtnClick(Sender: TObject);
begin
  Image1.Picture.Bitmap.Assign(Table1BlobField);
end;
```

Естественно, поскольку это должно делаться вручную, данный процесс менее желателен в приложении, по сравнению с автоматическим отображением графических данных в комбинации BDE и компонента `TDBImage`. Здесь уточняется подтип определенного пользователем BLOB-поля.

При работе с данными подтип BLOB-поля учитывается, т. к. сохраненные первыми данные устанавливают тип данных для этого поля для всей таблицы целиком. Таким образом, если данные формата `Bitmap` оказываются первым загружаемым типом, то этот формат будет единственно возможным для данного поля. До сих пор по умолчанию тип бинарного BLOB-поля (предопределенный тип 0) позволял BDE читать и отображать данные в компоненте `TDBImage` без особых проблем.

Утилиты Database Desktop допускают создание бинарных BLOB-полей только типа 0 и не позволяют определять их подтипы. Из-за такого ограничения таблицы, подразумевающие хранение и вывод изображений, должны создаваться с помощью запросов SQL. Обычно это делается посредством утилиты WISQL, но вполне достаточным является выполнение SQL-запроса с помощью компонента `TQuery`. Ниже приведен запрос SQL, создающий таблицу с определенным пользователем подтипом BLOB-поля:

```
CREATE TABLE WITHBMP(
  FILENAME CHAR(12),
  BITMAP Blob SUB_TYPE -1
)
```

После создания таблицы с совместимыми BLOB-полями, для хранения данных изображения в BLOB-поле и его вывода в компоненте `TDBImage` используются те же самые методы, что и при работе с таблицами dBASE и Paradox.

Имеется множество способов загрузки изображений в BLOB-поле. Вот три самых простых метода, которые ничем не отличаются от тех, которые были рассмотрены выше в разделе «Загрузка изображений в поля BLOB»:

- копирование данных из буфера обмена Windows в компонент TDBImage, связанный с BLOB-полем;
- использование метода LoadFromFile компонента TBlobField;
- копирование объекта типа TBitmap в значение свойства Picture компонента TBDBImage посредством метода Assign.

Клиентский запрос к серверу

Как я могу выбрать для клиента только часть данных с определенной позиции из набора данных на сервере?

Наиболее приемлемым является использование TQuery и Provider.SetParams.

Но можно сделать это иначе.

Первоначально на клиенте нужно считать с сервера только метаданные для набора данных. Это можно сделать, установив PacketRecords в 0 и затем вызвав Open. Затем нужно вызвать метод сервера (необходимо определить этот метод на сервере), который выполнит позиционирование курсора на первую нужную запись. И, наконец, установите PacketRecords в нужное положительное значение и вызовите GetNextPacket.

[Nomadic]

Получение метода сервера

Как получить методы сервера приложения из TClientDataSet*?*

Это можно сделать так:

```
RemoteServer.AppServer.MyMethod;
```

AppServer — свойство только для чтения, позволяющее получить интерфейс удаленного сервера, возвращаемый провайдером сервера приложений. Клиентские приложения могут общаться напрямую с сервером приложений через этот интерфейс.

[Nomadic]

Быстрый поиск в базах данных

Предлагается утилита быстрого поиска по базе данных. Данная технология выполняет поиск по полям, преобразуя их значения в строки (все значения преобразуются в верхний регистр, включая действительные числа). Данное решение может быть не самым быстрым, однако на поверку оно оказывается быстрее остальных. Более того, представьте, что действительное значение какого-либо поля равно 4,509375354, а значение поиска равно 7, в этом случае утилита засчитает «попадание».

Утилита удобна также тем, что она за один проход производит поиск более, чем в одном поле. Это удобно, если имеются два поля с адресами.

```
unit Finder;

interface

uses
  DB, DBTables, SysUtils;
function GrabMemoFieldAsPChar(TheField: TMemoField): PChar;
function DoFindIn(TheField: TField; SFor: String): Boolean;
function FindIt(TheTable: TDataSet; TheFields: array of integer;
  SearchBackward: Boolean; FromBeginning: Boolean; SFor: String): Boolean;

{ Применение функции FindIt:
  if FindIt(NotesSearchT, [NotesSearchT.FieldByName('Leadman').Index],
            False, True, SearchText.Text) then DoSomething; }

implementation

function GrabMemoFieldAsPChar(TheField: TMemoField): PChar;
begin
  with TBlobStream.Create(TheField, bmRead) do begin
    GetMem(Result, Size + 1);
    FillChar(Result^, Size + 1, #0);
    Read(Result^, Size);
    Free;
  end;
end;

function DoFindIn(TheField: TField; SFor: String): Boolean;
var
  PChForMemo: PChar;
begin
  Result := False;
  case TheField.DataType of
    ftString: if (Pos(SFor, UpperCase(TheField.AsString)) > 0) then
              Result := True;
   ftInteger: if (Pos(SFor, TheField.AsString) > 0) then
              Result := True;
   ftBoolean: if SFor = UpperCase(TheField.AsString) then
              Result := True;
     ftFloat: if (Pos(SFor, TheField.AsString) > 0) then
              Result := True;
  ftCurrency: if (Pos(SFor, TheField.AsString) > 0) then
              Result := True;
      ftDate .. ftDateTime:
              if (Pos(SFor, TheField.AsString) > 0) then
              Result := True;
      ftMemo: begin
                SFor[Ord(Length(SFor)) + 1] := #0;
```

```
                PChForMemo := GrabMemoFieldAsPChar(TMemoField(TheField));
                StrUpper(PChForMemo);
                if not (StrPos(PChForMemo, @SFor[1]) = nil) then
                  Result := True;
                FreeMem(PChForMemo, StrLen(PChForMemo + 1));
              end;
  end;
end;

function FindIt(TheTable: TDataSet; TheFields: array of integer;
                SearchBackward: Boolean; FromBeginning: Boolean;
                SFor: String): Boolean;
var
  i, HighTheFields, LowTheFields: integer;
  BM: TBookmark;
begin
  TheTable.DisableControls;
  BM := TheTable.GetBookmark;
  try
    LowTheFields := Low(TheFields);
    HighTheFields := High(TheFields);
    SFor := UpperCase(SFor);
    Result := False;
    if FromBeginning then TheTable.First;
    if SearchBackward then begin
      TheTable.Prior;
      while not TheTable.BoF do begin
        for i := LowTheFields to HighTheFields do begin
          if DoFindIn(TheTable.Fields[TheFields[i]], SFor) then begin
            Result := True;
            Break;
          end;
        end;
        if Result then Break else TheTable.Prior;
      end;
    end else begin
      TheTable.Next;
      while not TheTable.EoF do begin
        for i := LowTheFields to HighTheFields do begin
          if DoFindIn(TheTable.Fields[TheFields[i]], SFor) then begin
            Result := True;
            Break;
          end;
        end;
        if Result then Break else TheTable.Next;
      end;
    end;
  finally
    TheTable.EnableControls;
    if not Result then
      TheTable.GotoBookmark(BM);
```

```
    TheTable.FreeBookmark(BM);
  end;
end;
end.
```

Поиск записи в больших таблицах

Процедура позволяет переходить на любую из записей в таблице (формат Paradox или dBASE).

```
uses
  ..., BDE, DBTables;

function MoveToRec(RecNo: longint; taSingle: TDBDataSet): DBIResult;
var
  CursorProps: CurProps;
begin
  Result := DbiGetCursorProps(taSingle.Handle, CursorProps);
  if Result = DBIERR_NONE then begin
    case TTable(taSingle).TableType of
      ttParadox: Result := DbiSetToSeqNo(taSingle.Handle, RecNo);
        ttDBase: Result := DbiSetToRecordNo(taSingle.Handle, RecNo);
           else
                Result := DBIERR_NOTSUPPORTED;
     end;
     if Result = DBIERR_NONE then taSingle.Resync([rmCenter]);
  end;
end;
```

[Куприн Александр]

Изменение каталога псевдонима во время выполнения приложения

Как изменить каталог псевдонима из работающего приложения?

Решение 1

```
procedure CheckTable(var Table: TTable; var TName: string);
var
  ChangePath: boolean;
  Path: string;
  ActiveState: Boolean;
begin
  if (TName = '') then TName := Table.TableName
  else with Table do begin
    ActiveState := Active;
    Close;
    Path := ExtractFilePath(TName);
    ChangePath := HasAttr(DatabaseName, faDirectory) or
                  (CompareText(DatabaseName, Path) <> 0);
```

```
    if (Length(Path) > 0) and ChangePath then DatabaseName := Path;
    if (CompareText(ExtractFileName(TName), TableName) <> 0) then
      TableName := ExtractFileName(TName);
    Active := ActiveState;
  end;
end;
```

Примечание

Для работы этого примера необходимо в uses *указать* FmxUtils*, исходник которого лежит в* C:\Program Files\Borland\Delphi5\Demos\Doc\Filmanex.

Решение 2

```
type
  TDataMod = class(TDataModule)
    Database: TDatabase;
  public
    procedure TempAlias(NewAlias, NewDir: String);
  end;

procedure TDataMod.TempAlias(NewAlias, NewDir: String);
begin
  with Session do
    if not IsAlias(NewAlias) then begin
      ConfigMode := cmSession;    // NewAlias будет ВРЕМЕННЫМ
      try
        AddStandardAlias(NewAlias, NewDir, 'Paradox');
        Database.Close;
        Database.AliasName := NewAlias;
        Database.Open;
      finally
        ConfigMode := cmAll;
      end;
    end;
end;
```

Комментарии:

- поместите компонент Database на форму DataModule;
- задайте свойству DatabaseName имя базы данных, например, 'TempDB';
- задайте свойству DatabaseName компонента TTable значение = 'TempDB';
- для получения дополнительной информации ознакомьтесь с примером MastApp, поставляемым вместе с Delphi.

Копирование записи в пределах одной и той же таблицы

Каким образом мне копировать запись в пределах одного и того же TTable*? То есть, если я вижу текущую запись на экране и хочу ее скопировать в ту же таблицу с изменением индекса поля, какие действия мне необходимо предпринять?*

Необходимы два TTable, связанные с одной таблицей. Когда Table1 позиционируется в копируемой строке, с помощью Table2 можно выполнить операцию добавления записи.

Пример только для демонстрации идеи:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  i: Cardinal;
  srcStream: TBlobStream;
begin
  try
    with Table1 do begin
      CheckBrowseMode;
      if EoF or BoF then
        raise Exception.Create('Разместите курсор на правильной строке');
    end;
    with Table2 do begin
      Append;
      for i := 0 to Table1.FieldCount - 1 do
        if Table1.Fields[i].DataType < ftBytes then
          FieldByName(Table1.Fields[i].FieldName).Assign(Table1.Fields[i])
        else try
          srcStream := nil;
          srcStream := TBlobStream.Create(TBlobField(Table1.Fields[i]), bmRead);
          TBlobField(FieldByName(Table1.Fields[i].FieldName)).LoadFromStream
                                                              (srcStream);
        finally
          if Assigned(srcStream) then srcStream.Free;
        end;
        Post;
    end;
  except
    on E: EDBEngineError do MessageDlg(E.Message, mtError, [mbOk], 0);
    on E: Exception do MessageDlg(E.Message, mtError, [mbOk], 0);
  end;
end;
```

Текущий номер записи таблицы

Какой метод возвращает номер текущей записи?

Функция, возвращающая номер текущей записи в наборе данных DataSet.

```
function RecordNumber(Dataset: TDataset): Longint;
var
  CursorProps: CurProps;
  RecordProps: RECProps;
begin
{ Возвращаем 0, если набор данных не Paradox или dBASE }
  Result := 0;
  with Dataset do begin
```

```
{ Набор данных активен? }
    if State = dsInactive then DBError(SDataSetClosed);
{ Нам необходимо сделать этот вызов, чтобы "захватить" курсор iSeqNums }
    Check(DbiGetCursorProps(Handle, CursorProps));
{ Синхронизируем курсор BDE с курсором набора данных }
    UpdateCursorPos;
{ Заполняем RecordProps текущими свойствами записи }
    Check(DbiGetRecord(Handle, dbiNOLOCK, nil, @RecordProps));
{ С каким типом набора данных мы работаем? }
    case CursorProps.iSeqNums of
      0:  Result := RecordProps.iPhyRecNum;  // dBASE
      1:  Result := RecordProps.iSeqNum;     // Paradox
    end;
  end;
end;
```

Затем в обработчике события `OnDataChange DataSet` используем команду:

```
MyTextVariable := 'Запись ' + IntToStr(RecordNumber(tImport)) + ' из '
                  + IntToStr(tImport.RecordCount);
```

Примечание

Хочется заметить, что физический номер записи имеет смысл только для таблиц dBASE, т. к. он (этот номер) меняется достаточно редко – при упаковке таблиц. Для таблиц Paradox можно говорить лишь о логическом номере записи, поскольку месторасположение записи может меняться при добавлении индекса и при добавлении/удалении записи.

Данный пример разработан для Delphi 1. Для адаптации под современные версии надо добавить `uses ..., BDE, DBConsts` *и заменить вызов* `DBError()` *на* `DatabaseError()`.

Связь с DB2 в сети Netware

Как заставить работать DB2 *через протокол IPX?*

Возможны два варианта доступа:

- через сервер NETWARE;
- прямая адресация.

Конфигурация для доступа через сервер

Примечание

Доступ проверялся через серверы NW 3.11 и 3.12.

Сервер DB2:

- должна быть установлена OS/2 Warp или OS/2 Warp Connect;
- включена поддержка NETWARE;

- в CONFIG.SYS в переменную среды DB2COMM добавить (через запятую) IPX-SPX и перезагрузить систему;
- создать командный файл DBIPXSET.CMD следующего вида (<NWSERVER> – имя сервера):

```
db2 update dbm cfg using fileserver objectname dbserver
```

- выполнить командный файл DBIPXSET.CMD;
- перестартовать сервер базы данных;
- создать командный файл DBIPXREG.CMD следующего вида (<USERNAME> – - имя пользователя, обладающего правами администратора на сервере <NWSERVER>):

```
db2 register nwbindery user
```

- выполнить командный файл DBIPXREG. CMD;
- ответить на запрос пароля.

Windows- клиент:

- установить Windows 3.1 или Windows for WorkGroups 3.11;
- установить клиент NETWARE версии 4.x;
- при установке включить поддержку Windows;
- установить клиент DB2 для Windows;
- используя программу Client Setup, описать новый узел – сервер базы данных:

```
Name - <любое имя>
Protocol - IPX/SPX
File server - <NWSERVER>
Object name - dbserver
```

- описать базу данных и разрешить доступ к ней через ODBC.

Конфигурация для доступа через прямую адресацию

Сервер DB2:

- см. п. 1 для сервера;
- найти в директории x:\sqllib\misc программу DB2IPXAD.EXE и выполнить ее;
- записать полученный адрес;

Windows – клиент:

- см. п. 1 для клиента (первые три шага);
- используя программу Client Setup описать новый узел – сервер базы данных:

```
Name - <любое имя>
Protocol - IPX/SPX
```

```
File server - *
Object name - <адрес, полученный от DB2IPXAD.EXE>
```

- описать базу данных и разрешить доступ к ней через ODBC.

[Nomadic]

Create Trigger – чувствительность к регистру

Почему DB2 *выдает ошибку при использовании* Create Trigger*?*

Все дело в правиле написания команды Create Trigger. Если все остальные команды корректно воспринимаются на любом регистре, то эта – только набранная на верхнем регистре.

[Nomadic]

Использование MS ADO

Как работать из Delphi напрямую с MS ADO?

Если потребуется организовать в программе доступ к базе данных и нет желания (или возможности) использовать BDE, то есть довольно приятный вариант: обратиться к ActiveX Data Objects. Однако с их применением есть некоторые проблемы, и одна из них состоит в передаче необязательных (optional) параметров, которые вроде как можно не указывать. Однако если работать с ADO грамотно, а не через IDispatch.Invoke, то это превращается в головную боль. Вот как от нее избавляться:

```
var
  OptionalParam: OleVariant;
  VarData: PVarData;

begin
  OptionalParam := DISP_E_PARAMNOTFOUND;
  VarData := @OptionalParam;
  VarData^.VType := varError;
  ...
```

после этого переменную OptionalParam можно передавать вместо неиспользуемого аргумента.

Далее, самый приятный способ получения Result sets.

Оптимальным является вариант, который позволяет получить любой желаемый вид курсора (как клиентский, так и серверный):

```
var
  MyConn: Connection;
  MyComm: Command;
  MyRecSet: Recordset;
  prm1: Parameter;
```

```
begin
  MyConn := CoConnection.Create;
  MyConn.ConnectionString := 'DSN=pubs; uid=sa; pwd=;';
  MyConn.Open('', '', '', -1);
  MyCommand := CoCommand.Create;
  MyCommand.ActiveConnection := MyConn;
  MyCommand.CommandText := 'SELECT * FROM blahblah WHERE BlahID=?'
  Prm1 := MyCommand.CreateParameter('Id', adInteger.adParamInput, -1, <value>);
  MyCommand.AppendParameter(Prm1);
  MyRecSet := CoRecordSet.Create;
  MyRecSet.Open(MyCommand, OptionalParam, adOpenDynamic,
                adLockReadOnly, adCmdText);
```

В Delphi (не ниже версии 4) существует «константа» `EmptyParam`, которую можно подставлять в качестве пустого параметра.

[Nomadic]

Создание функции провайдера

После того как я использовал правый щелчок мыши для создания функции провайдера, как мне снова выполнить команду контекстного меню Export from Table*?*

Как только вы экспортировали интерфейс провайдера, эта команда контекстного меню перестает быть видимой. Чтобы снова включить ее, нужно удалить ассоциированное свойство в Редакторе библиотеки типов (Type Library Editor) и затем нажать кнопку обновления информации в Редакторе библиотеки типов (Type Library Editor's Refresh button). Можно также удалить точку вхождения `Get_XXX` в исходном тексте `RemoteDataModule`.

[Nomadic]

Передача UserName и Password в удаленный модуль данных

Как передать `UserName` *и* `Password` *в удаленный модуль данных (remote data module)?*

В удаленный модуль данных «положите» компонент `TDatabase`, затем добавьте процедуру автоматизации (пункт главного меню Edit|Add To Interface) для `Login`.

Убедитесь, что свойство `HandleShared` компонента `TDatabase` установлено в `True`.

```
procedure Login(UserName, Password: WideString);
begin
{ DB = TDatabase Something unique between clients }
  DB.DatabaseName := UserName + 'DB';
  DB.Params.Values['USER NAME'] := UserName;
  DB.Params.Values['PASSWORD'] := Password;
  DB.Open;
end;
```

После того как метод автоматизации создан, можно вызывать его с помощью:

```
RemoteServer1.AppServer.Login('USERNAME','PASSWORD');
```

[Nomadic]

Использование интерфейсов в RemoteDataModule

Как работать с новыми, своими интерфейсами, в RemoteDataModule*?*

В редакторе библиотеки типов (typelib) можно добавить свои интерфейсы и сделать их членами оригинального coClass. После этого можно обращаться к этим интерфейсам, используя следующий синтаксис:

```
(IDispatch(RemoteServer.AppServer) as IAnother)
```

Необходимо заметить, что это будет работать только в том случае, если DCOM используется как транспорт.

[Nomadic]

Модуль данных для каждого MDIChild

Когда во время разработки устанавливаете свойство DataSource в компонентах базы данных для указания на модуль данных, VCL во время выполнения приложения будет пытаться создать связь с существующим TDataModule, основываясь на его свойстве Name. Так, добавив модуль данных к проекту и переместив его в свойстве проекта из колонки автоматически создаваемых форм в колонку доступных, можно разработать форму, содержащую элементы управления для работы с базами данных, после чего несколькими строчками кода создать экземпляр формы, имеющий экземпляр собственного модуля данных.

С помощью Репозитория создайте стандартное MDI-приложение (standard MDI application), в котором модуль TMDIChild будет похож на приведенный ниже. Добавленные строки имеют комментарий {!}. Хитрости спрятаны в конструкторе Create и задании другого порядка следования операторов.

```
unit Childwin;

interface

uses
  Windows, Classes, Graphics, Forms, Controls, ExtCtrls, DBCtrls, StdCtrls,
  Mask, Grids, DBGrids, DataM; {!} // Модуль TDataModule1

type
  TMDIChild = class(TForm)
    DBGrid1: TDBGrid;
    DBGrid2: TDBGrid;
```

```
    DBEdit1: TDBEdit;
    DBEdit2: TDBEdit;
    DBNavigator1: TDBNavigator;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  public
    {!} DM:TDataModule1;
    {!} constructor Create(AOwner:TComponent); override;
  end;

implementation

{$IFDEF XOXOXOX}  // DataM должен находиться в секции interface.
                  // Необходимо для среды
uses DataM;       // времени проектирования. Определение "XOXOXOX" подразумевает,
{$ENDIF}          // что это никогда не будет определено, но чтобы компилятор видел
                  // это.
{$R *.DFM}

{!} constructor TMDIChild.Create;
{!} begin
{!}   DM := TDataModule1.Create(Application);
{!}   inherited Create(AOwner);
{!}   DM.Name := '';
{!} end;

procedure TMDIChild.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action := caFree;
end;

end.
```

5

BDE

Проверка наличия IDAPI

Решение

```
unit FindBDE;

interface

implementation

uses
  Controls, SysUtils, Windows, Dialogs;

var
  IdapiPath: array[0..255] of Char;
  IdapiHandle: THandle;

initialization
  GetProfileString('IDAPI', 'DLLPath', 'C:\', IdapiPath, 255);

{ следующие строки "изолируют" первый путь к каталогу из IdapiPath в случае,
  если их несколько }
  if Pos(';', StrPas(IdapiPath)) <> 0 then
    StrPCopy(IdapiPath, Copy(StrPas(IdapiPath), 1,
             Pred(Pos(';' ,StrPas(IdapiPath)))));
  IdapiHandle := LoadLibrary(StrCat(IdapiPath, '\IDAPI01.DLL'));
  if IdapiHandle < HINSTANCE_ERROR then
    if MessageDlg('ОШИБКА: Borland Database Engine (IDAPI) не найдена.'
               + ' Перед следующей попыткой ее необходимо установить....', mtError,
                 [mbOK], 0) = mrOK then Halt
```

```
{ IDAPI в системе не установлена }
    else FreeLibrary(IdapiHandle);
{ IDAPI Установлена в системе }
end.
```

Примечание

Этот код характерен для Win16. Win32 требует работы с реестром.

RecCount в таблицах ASCII

В Delphi 1.0 для получения количества записей в ASCII-файле (файлы .TXT *и* .SCH*) я пользовался свойством* RecordCount *компонента* TTable*. В Delphi 2.0 эта функциональность не поддерживается! Я прав или не прав? Во всяком случае, как мне получить количество записей, содержащихся в ASCII-таблице?*

В Delphi 2.0 свойство RecordCount отображается на недокументированную функцию BDE DBIGetExactRecordCount. Данное изменение было сделано для обеспечения правильных величин при работе с «живыми» запросами. Очевидно, данный API по какой-то причине не поддерживает текстовые файлы.

Можно обойти эту проблему, вызывая функцию API BDE DbiGetRecordCount напрямую (добавьте BDE к списку используемых модулей):

```
procedure TForm1.FormKeyUp(Sender: TObject; var Key: Word);
var
  RecCount: Integer;
begin
  Check(DbiGetRecordCount(Table1.Handle, RecCount));
  ...
end;
```

[News Group]

Увеличение размера LCK-файла

Если EXE-файл расположен в том же каталоге, что и таблица Paradox, и флажок Local Share установлен в True, LCK-файл с каждым запросом растет, как на дрожжах. Другое условие – имеется соединение, установленное посредством DBIOpenTable или TTable.Open.

Решение

- установите частный (private) каталог в какое-нибудь другое место;
- переместите EXE-файл в каталог, отличный от каталога с файлами таблиц;
- установите Local Share в False.

[News Group]

Локальный и общий доступ

Я так понимаю, что LocalShare *относится к ситуации, когда другие не-IDAPI приложения могут одновременно иметь доступ к одним и тем же файлам?*

Примерно на такие мысли наводит поставляемая документация... К сожалению, это не так.

Установка свойства LOCALSHARE в False говорит BDE о том, что он должен сам решать при необходимости вопрос о блокировке таблицы-записи в типичных ситуациях, например, когда BDE «думает», что таблица находится на локальном диске, он выключает блокировку для увеличения скорости доступа. К сожалению, логические диски общего пользования в сетях PTP программно идентифицируются как локальные с предсказуемо «липовыми» результатами. Установка LOCALSHARE = True заставляет блокирующий механизм «включаться» для всех дисков и, следовательно, решает эту проблему.

[News Group]

Распространение BDE

Следуйте приведенной ниже инструкции для разворачивания BDE на клиентской машине:

- Отформатируйте две дискеты. Пометьте дискеты как «Disk 1» и «Disk 2».
- Скопируйте файлы с компакт-диска Delphi, содержащиеся в каталоге \REDIST\BDEINST\DISK1, на дискету, помеченную как «Disk 1», и файлы из каталога \REDIST\BDEINST\DISK2 на дискету «Disk 2».
- Вставьте в дисковод клиентской машины дискету, помеченную как «BDE Install 1» (в нашем примере это дисковод A:).
- Убедитесь, что в Windows отсутствуют запущенные программы. В меню Пуск выберите пункт Выполнить..., введите в поле Открыть окна Запуск программы командную строку "A:\DISK1\SETUP" и нажмите кнопку OK для начала установки Borland Database Engine на клиентской машине.
- Сначала на короткое время появится окно Database Engine Installation, затем окно preparing to install... и, наконец, диалог BDE Redistributable, содержащий кнопки Continue (Продолжить) и Exit (Выход). Нажмите Continue.
- Появится диалог Borland Database Engine Location Settings, позволяющий изменить каталог установки программ BDE и конфигурационных файлов. Оставьте настройки по умолчанию и нажмите кнопку Continue (Продолжить).
- Появится диалог Borland Database Engine Installation, позволяющий вернуться к предыдущим диалогам или начать установку. Нажмите кнопку Install (Установить).

- Копирование дискеты Disk 1 будет отображаться индикатором хода процесса (progress bar).
- Появится диалог BDE Redistributable Install Request. Вставьте дискету Disk 2. Нажмите кнопку Continue (Продолжить).
- По окончании процедуры установки появится диалог Borland Database Engine Installation Notification, сообщающий об успешной установке BDE. Нажмите кнопку Exit (Выход).
- Завершите работу Windows, удалите дискету из дисковода и перезагрузите клиентскую машину.

Если настройки по умолчанию уже где-то используются, произойдут изменения, указанные ниже.

На клиентской машине появятся два новых каталога – \IDAPI и \IDAPI\LANGDRV. Обратите внимание, что утилита BDE Configuration Utility, BDECFG.EXE, располагается в каталоге \IDAPI. Языковые драйверы можно найти в каталоге \IDAPI\LANGDRV как файлы *.LD. Файлы AUTOEXEC.BAT, CONFIG.SYS и SYSTEM.INI при инсталляции не изменяются.

В файле WIN.INI в каталоге \Windows\SYSTEM появятся новые секции:

```
[IDAPI]
DLLPATH=C:\IDAPI
CONFIGFILE01=C:\IDAPI\IDAPI.CFG
[Borland Language Drivers]
LDPath=C:\IDAPI\LANGDRV
```

Примечание

Win16 (без комментариев)

Получение дескриптора соединения ODBC посредством BDE

Решение

```
unit GetProp;
interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls, Forms,
  Dialogs, Grids, DBGrids, StdCtrls, DB, DBTables, DBIProcs, DBITypes, DBIErrs;

type
  TForm1 = class(TForm)
    Table1: TTable;
    DataSource1: TDataSource;
    Button1: TButton;
    Button2: TButton;
    DBGrid1: TDBGrid;
```

```
    Edit1: TEdit;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  end;

var
  Form1: TForm1;

implementation
{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
begin
  Table1.Active := True;
end;

procedure TForm1.Button2Click(Sender: TObject);
var
  hTmpDB: hDBIDb;
  iLen: word;
begin
  Check(DbiGetProp(hDBIObj(Table1.DBhandle), dbNATIVEHNDL, @hTmpDB,
                   SizeOf(hDBIDb), iLen));
  Edit1.Text := IntToStr(Longint(hTmpdb));
end;

end.
```

Информация о псевдонимах BDE

Как через конфигурацию IDAPI получить физический каталог расположения базы данных, зная ее псевдоним?

Обратите внимание на метод `GetAliasParams` класса `TSession`. Возвращенная строка будет содержать искомый путь.

Воспользуемся следующей функцией:

```
uses
  DbiProcs, DBiTypes;

{ Возвращает каталог расположения базы данных по заданному псевдониму
 (без обратного слэша) }
function GetDataBaseDir(const Alias: string): string;
var
  sp: PChar;
  Res: pDBDesc;
begin
  try
    New(Res);
    sp := StrAlloc(Length(Alias) + 1);
```

```
    StrPCopy(sp, Alias);
    if DbiGetDatabaseDesc(sp, Res) = 0 then Result := StrPas(Res^.szPhyName)
    else Result := '';
  finally
    StrDispose(sp);
    Dispose(Res);
  end;
end;
```

Получение пути псевдонима и таблицы

Как получить псевдоним или путь к таблице?

Решение 1

Есть три способа сделать это:

- первый годится только для постоянных псевдонимов BDE;
- второй работает с BDE и локальными псевдонимами;
- третий работает с BDE и локальными псевдонимами, используя «тяжелый» путь, через вызовы DBI.

```
function GetDBPath1(AliasName: string): TFileName;
var
  ParamList: TStringList;
begin
  ParamList := TStringList.Create;
  with Session do
    try
      GetAliasParams(AliasName, ParamList);
      Result := UpperCase(ParamList.Values['PATH']) + '\';
    finally
      Paramlist.Free;
    end;
end;

function GetDBPath2(AliasName: string): TFileName;
var
  ParamList: TStringList;
  i: integer;
begin
  ParamList := TStringList.Create;
  with Session do
    try
      try
        GetAliasParams(AliasName, ParamList);
      except
        for i:=0 to pred(DatabaseCount) do
          if (Databases[i].DatabaseName = AliasName) then
            ParamList.Assign(Databases[i].Params);
      end;
```

```
      Result := UpperCase(ParamList.Values['PATH']) + '\';
    finally
      Paramlist.Free;
    end;
end;

function GetDBPath3(ATable: TTable): TFileName;
var
  TblProps: CURProps;
  pTblName, pFullName: DBITblName;
begin
  with ATable do begin
    AnsiToNative(Locale, TableName, pTblName, 255);
    Check(DBIGetCursorProps(Handle, TblProps));
    Check(DBIFormFullName(DBHandle, pTblName, TblProps.szTableType, pFullName));
    Result := ExtractFilePath(StrPas(pFullName));
  end;
end;
```

Решение 2

По таблице (фактически по Database) получить физическое местонахождение.

Примечание

Database можно создать явно, если нет – Delphi сама его создаст. Доступ к ней осуществляется по `Table(Query).Database`.

```
uses
  DbiProcs;

function GetDirByDatabase(Database: TDataBase): string;
var
  pszDir: PChar;
begin
  pszDir := StrAlloc(255);
  try
    DbiGetDirectory(Database.Handle, True, pszDir);
    Result := StrPas(pszDir);

  finally
    StrDispose(pszDir);
  end;
end;
```

По псевдониму:

```
function GetPhNameByAlias(sAlias: string): string;
var
  Database: TDataBase;
  pszDir: PChar;
```

```
begin
  Database := TDataBase.Create(nil);
  pszDir := StrAlloc(255);
  try
    Database.AliasName := sAlias;
    Database.DatabaseName := 'TEMP';
    Database.Connected := True;
    DbiGetDirectory(Database.Handle, True, pszDir);
    Database.Connected := False;
    Result := StrPas(pszDir);
  finally
    Database.Free;
    StrDispose(pszDir);
  end;
end;
```

[Nomadic]

Отображение всех псевдонимов в ComboBox

Как отобразить список псевдонимов?

Решение

```
Session.GetAliasNames(ComboBox1.Items);
```

[Громов Роман]

Задание псевдонима программным путем

Эта информация поможет разобраться в вопросе создания и использования псевдонимов баз данных в приложениях.

Вне Delphi создание и конфигурирование псевдонимов осуществляется утилитой BDECFG.EXE. Тем не менее, применяя компонент TDataBase, в приложении можно создать и использовать псевдоним, не определенный в IDAPI.CFG.

Важно понять, что, создав псевдоним, использовать его можно только в текущем сеансе приложения. Псевдонимы определяют расположение таблиц базы данных и параметры связи с сервером баз данных. В конце концов, можно воспользоваться преимуществами псевдонимов в пределах приложения без необходимости беспокоиться об их наличии в конфигурационном файле IDAPI.CFG в момент инициализации приложения. Приведем некоторые варианты решения задачи.

- **Решение 1.** Создает и конфигурирует псевдоним для базы данных STANDARD (.DB, .DBF). Псевдоним затем используется компонентом TTable.
- **Решение 2.** Создает и конфигурирует псевдоним для базы данных INTERBASE (.GDB). Псевдоним затем используется компонентом TQuery для подключения к двум таблицам базы данных.

- **Решение 3.** Создает и конфигурирует псевдоним для базы данных STANDARD (.DB, .DBF). Демонстрация ввода псевдонима пользователем и его конфигурация во время выполнения программы.

Решение 1

Используем базу данных .DB или .DBF (STANDARD).

1. Создаем новый проект.
2. Располагаем на форме следующие компоненты: TDataBase, TTable, TDataSource, TDBGrid и TButton.
3. Дважды щелкаем по компоненту TDataBase или через контекстное меню (правая кнопка мыши) вызываем редактор базы данных.
4. Присваиваем базе данных имя "MyNewAlias". Это имя будет играть роль псевдонима в свойстве DatabaseName для компонентов типа TTable, TQuery, TStoredProc.
5. Выбираем в поле Driver Name (Имя драйвера) пункт STANDARD.
6. Нажимаем кнопку Defaults. Это автоматически добавляет путь (PATH=) в секцию перекрытых параметров (окно Parameter Overrides).
7. Устанавливаем переменную PATH (PATH=C:\Program Files\Common Files\Borland Shared\Data).
8. Нажимаем кнопку OK и закрываем окно редактора.
9. В компоненте TTable свойству DatabaseName присваиваем значение "MyNewAlias".
10. В компоненте TDataSource для свойства DataSet устанавливаем значение "Table1".
11. В компоненте DBGrid свойству DataSource присваиваем значение "DataSource1".
12. Создаем в компоненте TButton обработчик события OnClick.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Table1.Tablename:= 'CUSTOMER';
  Table1.Active:= True;
end;
```

13. Запускаем приложение.

Примечание

В качестве альтернативы шагам 3–11 можно включить все эти действия в сам обработчик:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Database1.DatabaseName := 'MyNewAlias';
  Database1.DriverName := 'STANDARD';
```

```
  Database1.Params.Clear;
  Database1.Params.Add('PATH=C:\Program Files\Common Files\Borland Shared\Data');
  Table1.DatabaseName := 'MyNewAlias';
  Table1.TableName := 'CUSTOMER';
  Table1.Active := True;
  DataSource1.DataSet := Table1;
  DBGrid1.DataSource := DataSource1;
end;
```

Решение 2

Используем базу данных INTERBASE.

1. Создаем новый проект.
2. Располагаем на форме следующие компоненты: TDataBase, TQuery, TDataSource, TDBGrid и TButton.
3. Посредством двойного щелчка по компоненту TDataBase или через контекстное меню (правая кнопка мыши) вызываем редактор базы данных.
4. Присваиваем базе данных имя «MyNewAlias». Оно будет играть роль псевдонима в свойстве DatabaseName для компонентов типа TTable, TQuery, TStoredProc.
5. Выбираем в поле Driver Name (имя драйвера) пункт INTRBASE.
6. Нажимаем кнопку Defaults. Это автоматически добавляет путь (PATH=) в секцию перекрытых параметров (окно Parameter Overrides).

```
SERVER NAME=IB_SERVER:/PATH/DATABASE.GDB
USER NAME=MYNAME
OPEN MODE=READ/WRITE
SCHEMA CACHE SIZE=8
LANGDRIVER=
SQLQRYMODE=
SQLPASSTHRU MODE=SHARED AUTOCOMMIT
SCHEMA CACHE TIME=-1
MAX ROWS=-1
BATCH COUNT=200
ENABLE SCHEMA CACHE=FALSE
SCHEMA CACHE DIR=
ENABLE BCD=FALSE
BLOBS TO CACHE=64
BLOB SIZE=32
PASSWORD=
```

Устанавливаем следующие параметры:

```
SERVER NAME=C:\Program Files\Common Files\Borland Shared\Data\EMPLOYEE.GDB
USER NAME=SYSDBA
OPEN MODE=READ/WRITE
SCHEMA CACHE SIZE=8
LANGDRIVER=
SQLQRYMODE=
```

```
SQLPASSTHRU MODE=NOT SHARED
SCHEMA CACHE TIME=-1
PASSWORD=masterkey
```

7. В компоненте TDataBase свойство LoginPrompt устанавливаем в False. Если в секции перекрытых параметров (окно Parameter Overrides) задан пароль (ключ PASSWORD) и свойство LoginPrompt установлено в False, при соединении с базой данный пароль запрашиваться не будет.

Внимание

При неправильно указанном пароле в секции перекрытых параметров и неактивном свойстве LoginPrompt *вы не сможете получить доступ к базе данных, поскольку нет возможности ввести правильный пароль – диалоговое окно* Ввод пароля *отключено свойством* LoginPrompt.

8. Нажимаем кнопку OK и закрываем окно редактора.
9. В компоненте TQuery свойству DatabaseName присваиваем значение "MyNewAlias".
10. В компоненте TDataSource свойству DataSet присваиваем значение "Query1".
11. В компоненте DBGrid свойству DataSource присваиваем значение "DataSource1".
12. Создаем в компоненте TButton обработчик события OnClick.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Query1.SQL.Clear;

  Query1.SQL.ADD('SELECT DISTINCT * FROM CUSTOMER C, SALES S' +
             ' WHERE (S.CUST_NO = C.CUST_NO)' +
             ' ORDER BY C.CUST_NO, C.CUSTOMER');
  Query1.Active := True;
end;
```

13. Запускаем приложение.

Решение 3

Ввод псевдонима пользователем.

В этом решении выводится диалоговое окно и на основе информации, введенной пользователем, создается псевдоним.

Директория, имя сервера, путь, имя базы данных и другая необходимая информация для получения псевдонима может быть получена приложением из диалогового окна или конфигурационного .INI-файла.

1. Выполняем шаги 1–11 из решения 1.
2. Пишем следующий обработчик события OnClick компонента TButton:

```
procedure TForm1.Button1Click(Sender: TObject);
```

```
var
  NewString: string;
  ClickedOK: Boolean;

begin
  NewString := 'C:\';
  ClickedOK := InputQuery('Database Path', 'Path:', NewString);
  if ClickedOK then begin
    Database1.DatabaseName := 'MyNewAlias';
    Database1.DriverName := 'STANDARD';
    Database1.Params.Clear;
    Database1.Params.Add('Path=' + NewString);
    Table1.DatabaseName := 'MyNewAlias';
    Table1.TableName := 'CUSTOMER';
    Table1.Active := True;
    DataSource1.DataSet := Table1;
    DBGrid1.DataSource := DataSource1;
  end;
end;
```

3. Запускаем приложение.

Изменение псевдонима во время выполнения программы

Как с помощью Delphi проще изменить путь псевдонима (Alias Path)?

Можно создать виртуальный псевдоним (т. е. не сохраняемый в IDAPI.CFG) с помощью компонента TDataBase и затем изменить ассоциированный путь DOS на нужный.

Решение

```
var
  d: TDataBase;

begin
  d := TDataBase.Create(Application);
  d.DataBaseName := 'TEST_ALIAS';
  d.DriverName := 'STANDARD';
  d.Params.Add('PATH=C:\Program Files\Common Files\Borland Shared\Data');
  d.Connected := True;
end;
```

Приведенный код, может быть, и бесполезен, но он показывает, как это можно сделать. Также это можно сделать во время разработки приложения с помощью Инспектора объектов.

Псевдоним на лету

- Поместите компонент Database на форму DataModule.

- Задайте свойству DatabaseName имя базы данных, например, "TempDB".
- Задайте свойству DatabaseName компонента TTable значение "TempDB".
- Для получения дополнительной информации ознакомьтесь с примером MastApp, поставляемым с Delphi.

```
type
  TDataMod = class(TDataModule)
    Database: TDataBase;
  public
    procedure TempAlias(NewAlias, NewDir: String);
  end;
procedure TDataMod.TempAlias(NewAlias, NewDir: String);
begin
  with Session do
    if not IsAlias(NewAlias) then begin
      ConfigMode := cmSession;    // NewAlias будет ВРЕМЕННЫМ
      try
        AddStandardAlias(NewAlias, NewDir, 'PARADOX');
        Database.Close;
        Database.AliasName := NewAlias;
        Database.Open;
      finally
        ConfigMode := cmAll;
      end;
    end;
end;
```

Синтаксис функции DbiAddAlias

Где можно узнать о параметрах в функции DbiAddAlias*?*

Синтаксис:

```
DBIResult DbiAddAlias([hCfg], pszAliasName, pszDriverType, pszParams, bPersistent);
```

Описание:

DbiAddAlias добавляет псевдоним в конфигурационный файл, связанный с текущим сеансом.

Параметры:

hCfg — тип: hDBICfg

Для BDE 2.5 данный параметр должен быть NIL. Указывает, что конфигурация действует в течение текущего сеанса. Другие значения для BDE 2.5 не поддерживаются.

pszAliasName — тип: pCHAR

Указатель на имя псевдонима. Это имя нового псевдонима, который должен быть добавлен.

pszDriverType — тип: pCHAR

Указатель на тип устройства. Это тип устройства для добавляемого псевдонима. Если данный параметр равен NIL, псевдоним будет добавлен для базы данных STANDARD. Если указан szPARADOX, szDBASE или szASCII, будет добавлена запись в генератор псевдонимов базы данных STANDARD для указания того, что данный тип будет предпочтительным типом устройства. Если указано имя устройства, то оно должно существовать в измененном файле конфигурации.

pszParams — тип: pCHAR

Указатель на список дополнительных параметров. Данный список определяется следующим образом:

```
"AliasOption: Option Data[;AliasOption: Option Data][;...]"
```

AliasOption должен соответствовать одному из значений, возвращаемых DBIOpenCfgInfoList. Для псевдонимов базы данных STANDARD единственно необходимый параметр PATH, остальные игнорируются (без ошибок).

Пример 1. Установка пути для использования базы данных

```
STANDARD: "PATH:c:\mydata"
```

Пример 2. Установка имени сервера и имени пользователя для работы с драйверм SQL

```
"SERVER NAME: server:/path/database;USER NAME: myname"
```

bPersistent — тип: BOOL. Определяет область действия нового псевдонима. Если имеет значение True, то сохраняется в файле конфигурации для будущих сеансов. В противном случае псевдоним действует только в течение текущего сеанса. Псевдоним удаляется в конце сеанса (или при выходе из программы).

Использование

Созданный данной функцией псевдоним будет иметь параметры по умолчанию, хранимые в списке параметров драйверов DB OPEN, если только они не перекрыты в параметре pszParams. Вы можете использовать DBIOpenCfgInfoList, чтобы изменить значение по умолчанию после добавления псевдонима с помощью DBIAddAlias.

Для псевдонимов стандартной базы данных все параметры pszParams, за исключением PATH, игнорируются.

Внимание

DbiInit *должен вызываться до вызова* DbiAddAlias.

Возвращаемые значения DBIResult:

- DBIERR_INVALIDPARAM — имя псевдонима — NIL или один из следующих типов pszDriverType: szASCII, szDBASE, szPARADOX. В последнем случае используйте NIL pszDriverType для указания на базу данных STANDARD.

- DBIERR_NONE — псевдоним был успешно добавлен.
- DBIERR_NAMENOTUNIQUE — существует другой псевдоним с тем же именем (работает, если bPersistent равен True).
- DBIERR_OBJNOTFOUND — один (или более) из дополнительных параметров, указанных в pszParams, не соответствуют правильным типам в секции драйверов конфигурационного файла.
- DBIERR_UNKNOWNDRIVER — имя устройства в конфигурационном файле при сопоставлении с pszDriverType не найдено.

Добавление псевдонима с помощью функции DbiAddAlias

Как воспользоваться функцией DbiAddAlias*?*

Решение

```
var
  pszAliasName: PChar;      // Имя псевдонима
  pszDriverType: PChar;     // Тип драйвера для псевдонима
  pszParams: PChar;         // Дополнительные параметры
  bPersist: Bool;           // Постоянный или временный псевдоним
  dbiRes: Integer;          // Возвращаемый код

begin
  pszAliasName := strAlloc(25);
  pszDriverType := strAlloc(25);
  pszParams := strAlloc(100);
  try
    bPersist := True;
    strPcopy(pszAliasName, 'Lance');
    strPcopy(pszDriverType, 'PARADOX');
    strPcopy(pszParams, 'PATH:' + 'c:\Paradox');
    dbiRes := DbiAddAlias(nil, pszAliasName, pszDriverType, pszParams, bPersist);

  finally
    strDispose(pszAliasName);
    strDispose(pszDriverType);
    strDispose(pszParams);
  end;

end;
```

Копирование таблицы с помощью BDE

Решение

```
function CopyTable(tbl: TTable; dest: string): boolean;
var
  psrc, pdest: array[0..DBIMAXTBLNAMELEN] of char;
```

```
    rslt: DBIResult;
  begin
    Result := False;
    StrPCopy(pdest, dest);
    with tbl do begin
      try
        DisableControls;
        StrPCopy(psrc, TableName);
        rslt := DbiCopyTable(DBHandle, True, psrc, nil, pdest);
        Result := (rslt = 0);
      finally
        Refresh;
        EnableControls;
      end;
    end;
  end;
```

Обратные вызовы BDE32 для получения статуса операций

Данный совет показывает, как в Delphi 2.01 можно использовать функцию BDE `DBICallBack` для получения значения индикатора хода выполнения при длительных пакетных операциях, связанных с обработкой данных.

Дополнительная документация, описывающая вызовы функций BDE, находится в файле `BDE32.HLP` (расположенном в каталоге, где установлен 32-битный IDAPI).

При создании функций обратного вызова BDE, BDE будет осуществлять «обратный вызов» функций вашего приложения, позволяя тем самым извещать его о происходящих событиях, а в некоторых случаях передавать информацию обратно BDE.

BDE определяет несколько возвращаемых типов, которые могут быть установлены для обратного вызова:

- состояние больших пакетных операций;
- запросы для передачи информации вызывающему оператору.

Данный совет подробно описывает обратный вызов типа `cbGENPROGRESS`, позволяющий изменять индикатор выполнения в соответствии с состоянием операции.

Чтобы это сделать, необходимо сначала вызвать функцию `DBIGetCallBack()`, возвращающую дескриптор обратного вызова, который мог быть уже установлен (с этими параметрами), и сохранить информацию в структуре данных. Затем установить свой обратный вызов, заменяя им любой установленный до этого.

При установке обратного вызова понадобится передавать BDE указатель на структуру данных, содержащую информацию о предыдущем установленном обратном вызове, после чего, при выполнении вашей функции обратного вы-

зова, вы можете воспользоваться оригинальным обратным вызовом (если он установлен).

BDE каждый раз возвращает приложению сообщение, содержащее количество обработанных записей, или же процентное соотношение обработанных записей, также передаваемое в виде целого числа. Код должен учитывать эту ситуацию. Если процентное поле в структуре обратного вызова больше чем −1, можно сделать вывод, что передано число, и можно сразу обновить индикатор выполнения. Если же это поле меньше нуля, значит, обратный вызов получил текстовое сообщение, помещенное в поле szTMsg и содержащее количество обработанных записей. В этом случае понадобится осуществить грамматический разбор текстового сообщения, преобразовать остальные строки в целое, затем вычислить текущий процент обработанных записей, и только после этого изменить индикатор выполнения.

Наконец, после осуществления операции с данными, необходимо «деактивировать» обратный вызов и вновь установить предыдущую функцию обратного вызова (если она существует).

Для следующего примера необходимо создать форму и расположить на ней две таблицы, компоненты ProgressBar, BatchMove, Button.

```
unit Testbc1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, DB, DBTables, ComCtrls;

type
  TForm1 = class(TForm)
    Table1: TTable;
    BatchMove1: TBatchMove;
    Table2: TTable;
    Button1: TButton;
    ProgressBar1: TProgressBar;
    procedure Button1Click(Sender: TObject);
  end;

var
  Form1: TForm1;

implementation
uses
  Bde;    // Здесь расположены Dbi Types и Procs

{$R *.DFM}

{ тип структуры данных для сохранения информации о предыдущем обратном вызове }
type
```

```
  TDbiCbInfo = record
  ecbType: CBType;
  iClientData: longint;
  DataBuffLn: word;
  DataBuff: pCBPROGRESSDesc;
  DbiCbFn: pointer;
end;

type
  PDbiCbInfo = ^TDbiCbInfo;
{ Наша функция обратного вызова }
function DbiCbFn(ecbType: CBType; iClientData: Longint;
                 CbInfo: pointer): CBRType stdcall;
var
  s: string;
begin
{ Проверяем, является ли тип обратного вызова тем, который мы ожидаем }
  if ecbType = cbGENPROGRESS then begin
{ если iPercentDone меньше нуля, извлекаем число обработанных записей }
    if pCBPROGRESSDesc(cbInfo).iPercentDone < 0 then begin
      s := pCBPROGRESSDesc(cbInfo).szMsg;
      Delete(s, 1, Pos(': ', s) + 1);
{ Вычисляем процент выполненного и изменяем индикатор хода выполнения }
      Form1.ProgressBar1.Position :=
          Round((StrToInt(s)/Form1.Table1.RecordCount) * 100);
    end else
{ Устанавливаем линейку прогресса }
      Form1.ProgressBar1.Position := pCBPROGRESSDesc(cbInfo).iPercentDone;
  end;
{ существовал ли предыдущий зарегистрированный обратный вызов? }
  if PDbiCbInfo(iClientData)^.DbiCbFn <> nil then
    DbiCbFn := pfDBICallBack(PDbiCbInfo(iClientData)^.DbiCbFn)
           (ecbType, PDbiCbInfo(iClientData)^.iClientData, cbInfo)
  else DbiCbFn := cbrCONTINUE;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  CbDataBuff: CBPROGRESSDesc;    // Структура DBi
  OldDbiCbInfo: TDbiCbInfo;      // структура данных должна хранить информацию
                                 // о предыдущем обратном вызове
begin
{ Убедимся в том, что перемещаемая таблица открыта }
  Table1.Open;
{ Убедимся в том, что таблица-приемник закрыта }
  Table2.Close;
{ получаем информацию о любом установленном обратном вызове }
  DbiGetCallBack(Table2.Handle, cbGENPROGRESS, @OldDbiCbInfo.iClientData,
                 @OldDbiCbInfo.DataBuffLn, @OldDbiCbInfo.DataBuff,
                 pfDBICallBack(OldDbiCbInfo.DbiCbFn));
{ регистрируем наш обратный вызов }
```

```
  DbiRegisterCallBack(Table2.Handle, cbGENPROGRESS, longint(@OldDbiCbInfo),
        SizeOf(cbDataBuff), @cbDataBuff, @DbiCbFn);
  ProgressBar1.Position := 0;
  BatchMove1.Execute;
{ если предыдущий обратный вызов существовал, вновь устанавливаем его, }
{ в противном случае регистрируем функцию обратного вызова }
  if OldDbiCbInfo.DbiCbFn <> nil then
    DbiRegisterCallBack(Table2.Handle, cbGENPROGRESS, OldDbiCbInfo.iClientData,
              OldDbiCbInfo.DataBuffLn, OldDbiCbInfo.DataBuff, OldDbiCbInfo.DbiCbFn)
  else DbiRegisterCallBack(Table2.Handle, cbGENPROGRESS, longint(@OldDbiCbInfo),
        SizeOf(cbDataBuff), @cbDataBuff, nil);
{ Показываем наш успех! }
  Table2.Open;
end;

end.
```

Демонстрация обратного вызова BDE

Существует обратный вызов (callback) BDE, посредством которого можно получать уведомления об изменении таблиц Paradox. Тем не менее, от вас потребуется использование таймера. Функция обратного вызова инициируется при вызове функций, осуществляющих доступ к таблице. Ниже приведен код, демонстрирующий технику работы с описанным выше обратным вызовом:

```
unit tcmain;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  DB, DBTables, ExtCtrls, DBCtrls, Grids, DBGrids, BDE, StdCtrls;

const
  WM_UPDATETABLE = WM_USER + 1;
type
  TForm1 = class(TForm)
    Table1: TTable;
    DataSource1: TDataSource;
    DBGrid1: TDBGrid;
    DBNavigator1: TDBNavigator;
    Timer1: TTimer;
    Button1: TButton;
    procedure Table1AfterOpen(DataSet: TDataSet);
    procedure FormCreate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    FChgCnt: Integer;
    FCB: TBDECallback;
```

```
    function TableChangeCallBack(CBInfo: Pointer): CBRType;
    procedure UpdateTableData(var Msg: TMessage); message WM_UPDATETABLE;
  end;
var
  Form1: TForm1;
implementation

{$R *.DFM}
// Это функция, вызываемая функцией обратного вызова.
function TForm1.TableChangeCallBack(CBInfo: Pointer): CBRType;
begin
  Inc(FChgCnt);
  Caption := IntToStr(FChgCnt);
  MessageBeep(0);
// Здесь мы не можем вызвать Table1.Refresh, сделаем это позже.
  PostMessage(Handle, WM_UPDATETABLE, 0, 0);
end;

procedure TForm1.UpdateTableData(var Msg: TMessage);
begin
// Не пытайтесь вызвать обновление, если мы в "середине" редактирования.
  if (Table1.State = dsBrowse) then Table1.Refresh;
end;

procedure TForm1.Table1AfterOpen(DataSet: TDataSet);
begin
// Установка обратного вызова.
  FCB := TBDECallback.Create(Self, Table1.Handle, cbTableChanged,
                             nil, 0, TableChangeCallBack, true);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Table1.DatabaseName := ExtractFilePath(ParamStr(0));
  Table1.Open;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
var
  SeqNo: Longint;
begin
// События таймера просто осуществляют вызов DbiGetSeqNo для получения доступа к
// таблице. В противном случае мы не хотим делать обратный вызов, пока что-то
// делаем (типа прокрутки) для получения доступа к данным. DbiGetSeqNo вызывается в
// случае, если таблица не активна.
  if Table1.State <> dsInActive then DbiGetSeqNo(Table1.Handle, SeqNo);
end;
end.
```

[News Group]

Запись буфера BDE на диск

Сделанные в таблице изменения непосредственно на диск не записываются до тех пор, пока таблица не закрыта. Потеря питания или сбой в системе могут привести к утрате данных и прочим неприятностям. Чтобы избежать этого, существует два прямых вызова Database Engine, приводящих к одному и тому же результату. Это функции DbiUseIdleTime и DbiSaveChanges.

Последняя сохраняет на диске все обновления, находящиеся в буфере таблицы, связанной с курсором (hDBICur). Функция может быть вызвана из любого места программы. Например, можно при каждом обновлении записи сохранять на диске все изменения (добавьте BDE в список используемых модулей):

```
procedure TForm1.Table1AfterPost(DataSet: TDataSet);
begin
  DbiSaveChanges(Table1.Handle);
end;
```

При таком способе можно не опасаться порчи данных в случае потери питания или сбоя системы, которая может произойти после обновления записи.

При помощи функции DBISaveChanges также можно временную таблицу (созданную с помощью DBICreateTempTable) сделать постоянной.

Эта функция не применима к таблицам SQL.

DBIUseIdleTime может быть вызвана, если очередь запросов Windows (Windows Message Queue) пуста. Это позволяет Database Engine сохранить на диске «грязные буферы». Другими словами, выполняется операция DBISaveChanges, но применительно ко ВСЕМ измененным таблицам. Тем не менее, данная операция не обязательно должна выполняться после каждого обновления записи, ее нужно приберечь для «холостого» периода (период простоя, idle).

В Delphi этому можно найти такое применение (добавьте BDE в список используемых модулей):

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Application.OnIdle := UseIdle;
end;
procedure Tform1.UseIdle(Sender: TObject; var Done: Boolean);
begin
  DbiUseIdleTime;
end;
```

Примечание

Использование обоих вызовов `DBIUseIdleTime` *и* `DBISaveChanges` *(после каждого обновления записи) излишне и сопровождается необязательными вызовами функций. Если приложение выполняет множественный ввод новых записей или их редактирование в течение небольшого периода времени, рекомендуем осуществлять вызов функции* `DBIUseIdleTime` *во время простоя клиента, а вызов* `DBISaveChanges` *по завершении «пакета» обновлений.*

В случае если в таблице выполняется не слишком много изменений, клиент может использовать вызов `DBISaveChanges` *после каждого постинга*[1] *или же «прикрепить» к таймеру вызов* `DBIUseIdleTime`.

Приложения BDE32 в одноранговой сети

Сетью Peer-To-Peer (сетью, в которой каждая машина действует как клиент и как сервер) может быть одной из следующих сетей, включая другие сетевые платформы, совместимые с ними:

- Windows 95;
- Windows NT;
- Lantastic;
- Netware Lite.

BDE автоматически обнаруживает таблицы на сетевом диске, но не может их определить на выделенном сервере (dedicated server) или узле одноранговой сети. Выделенные серверы уведомляют приложение клиента о том, что файл был изменен или заблокирован. Данная функциональность отсутствует в сетях Peer-To-Peer (без выделенного сервера). Для ее включения в одноранговых сетях, установите свойство `LOCAL SHARE` в `True` в BDE Configuration Utility в узле `System`. Это должно быть сделано на всех клиентах BDE, которые имеют доступ к таблицам в сетях, указанных выше. В случае файловых серверов Novell данное требование не является необходимым.

Работая с таблицами Paradox, следует помещать их в каталог с сетевым контролем, который должен находиться в сети для всех используемых клиентских приложений. Хорошим стилем считается использование отдельного каталога для приложения, сети и таблиц. Поясним примером:

```
<Каталог общего доступа>
<Каталог таблиц>
<Каталог Exe-файлов>
<Сетевой каталог>
```

Существуют две различных среды BDE:

- работа только с 32-битными приложениями BDE;
- совместная работа 32- и 16-битных приложений BDE.

[1] posting – сохранение табличных записей.

Установка только для 32-битных приложений

32-битное BDE полностью поддерживает соглашение о путях UNC вместе с длинными именами файлов. Рекомендуется использование соглашения UNC для всех сетевых соединений BDE. UNC позволяет обойтись без подключения сетевых (mapped) дисков. Это позволяет иметь доступ к таблицам и сетевым каталогам без необходимости заставлять пользователя подключать сетевые диски. UNC имеет следующий синтаксис:

```
\<Имя сервера>\<Имя каталога общего доступа>\<Путь к каталогу> + <Имя файла>
```

Далее приведен пример стандартного псевдонима (alias) BDE с использованием UNC.

Псевдоним:	`MyUNCAlias`
Тип:	STANDARD
Путь:	\\FooServer\FooShare\Sharedir\Tables
Драйвер по умолчанию:	Paradox

Сетевой каталог может быть установлен и таким способом:

Драйвер:	Paradox
Сетевой каталог:	\\FooServer\FooShare\Sharedir\NetDir

Сетевой каталог может быть установлен во время выполнения приложения с помощью `Session.NetFileDir` (Delphi) или `DbiSetProp` (C++/Delphi).

Если по какой-либо причине UNC не может применяться в 32-битных приложениях, следуйте инструкциям для среды с 32- и 16-битными приложениями BDE, изложенными далее.

Установка для 16-битных и 32-битных приложений BDE

Поскольку 16-битное Windows API не поддерживает UNC, ее не поддерживает и 16-битное BDE. Для того чтобы приложения могли иметь общий доступ к таблицам, все клиенты должны подключить один и тот же каталог на сервере. Если сервер также используется и в качестве клиента, то все другие клиенты должны подключить его корневой каталог диска. Логический диск при этом у клиентов может быть разным. Вот несколько примеров с работающими и неработающими настройками:

Работоспособно:

	Путь
Клиент1:	X:\Каталог общего доступа\Таблицы
Клиент2:	X:\Каталог общего доступа\Таблицы

Работоспособно:

	Путь
Клиент1: (Также машина с таблицами):	X:\Каталог общего доступа\Таблицы
Клиент2:	X:\Каталог общего доступа\Таблицы

Работоспособно:

	Путь
Клиент1: (Также машина с таблицами)	C:\Каталог общего доступа\Таблицы
Клиент2:	X:\Каталог общего доступа\Таблицы
Клиент3:	R:\Каталог общего доступа\Таблицы

Неработоспособно. BDE должна иметь доступ к файлу Network Control (управление сетью).

	Путь
Клиент1:	X:\Каталог общего доступа\Таблицы
Клиент2:	X:\Таблицы (где X:\Таблицы реально – X:\Каталог общего доступа\Таблицы, но имеющий общий доступ в \Каталог общего доступа)

Итог (установки для одноранговых сетей):

16- и/или 32-битные приложения:

- в BDE Configuration Utility установите свойство LOCAL SHARE в True;
- не используйте UNC-имена;
- не используйте таблицы с длинными именами файлов;
- убедитесь в том, что все клиенты подключены к одному и тому же каталогу на сервере.

Только 32-битные приложения:

- в BDE Configuration Utility установите LOCAL SHARE в True;
- для получения доступа к сетевому каталогу и каталогу с таблицами используйте UNC-имена.

При невыполнении описанных выше шагов пользователи могут блокировать таблицы с получением следующей ошибки:

```
"Directory is controlled by other .NET file."
  (Каталог управляется другим .NET-файлом)
"File:  PDOXUSRS.LCK" ("Файл:  PDOXUSRS.LCK")
"Directory: " (Каталог:)
```

или

```
"Multiple .NET files in use." (Используются несколько .NET-файлов.)
"File: PDOXUSRS.LCK" (Файл: PDOXUSRS.LCK)
```

[News Group]

Работа с BDE в сети

Может ли мое приложение иметь доступ к файлам, расположенным на сетевых дисках?

Да.

Когда я попытался это сделать, программа выдала сообщение об ошибке «Not initialized for accessing network files» (не инициализирована для доступа к сетевым файлам).

Необходимо задать правильный путь к каталогу в секции NET DIR файла IDAPI.CFG. Директория должна быть одна и к ней должен быть открыт доступ всем пользователям приложения с применением одинаковых подключенных сетевых дисков. Если NET DIR указывает на F:\PUBLIC\NETDIR, то пользователи с подключенным сетевым диском G:\NETDIR доступа не получат.

Можно ли запустить приложение, относящееся к описываемой категории, с сетевого диска без установленного на локальной машине BDE (за исключением возможных ссылок в локальном файле WIN.INI *на копии элементов программы* BDE/IDAPI*, расположенных на сетевом диске)?*

Установите BDE в сети, затем добавьте следующие секции в файл WIN.INI каждой рабочей станции:

```
[IDAPI]
CONFIGFILE01=F:\IDAPI\IDAPI.CFG
DLLPATH=F:\IDAPI
[Borland Language Drivers]
LDPath=F:\IDAPI\LANGDRV
```

Пути должны отражать текущее месторасположение каталога IDAPI.

Для установки NET DIR мне нужно запустить BDECFG на каждой рабочей станции или просто сделать это на сервере?

С помощью утилиты BDECFG отредактируйте файл IDAPI.CFG и сохраните его в сетевом каталоге IDAPI. Следовательно, данную операцию необходимо проделать всего лишь один раз.

Если мне нужно сделать это только на сервере, то как все рабочие станции узнают о месторасположении сетевых файлов (NET DIR)?

Рабочая станция открывает файл IDAPI.CFG из каталога, указанного в WIN.INI, и уже оттуда читает настройки NET DIR.

[News Group]

Управление сетевыми каталогами (BDE)

Если два различных пользователя подключают два различных сетевых каталога (Net Control Directories, NCD), но при этом пути к каталогам одинаковые (это бывает при работе с сетью), BDE думает, что в этом случае подключен один NCD. Это может привести к серьезным проблемам.

Если два пользователя подключают один и тот же NCD, но с разными путями, BDE думает, что это два различных NCD и не позволяет второму пользователю редактировать таблицу. Например, пользователь A подключил NCD по пути G:\DATA\BDENET. Пользователь B подключил NCD по пути H:\BDENET, где H: подключен по пути G:\DATA. В этом случае оба пользователя пытаются работать с одним и тем же NCD, но BDE не знает об этом.

Если в вышеприведенном примере пользователи подключают каталог по одному и тому же пути, но с различными буквами диска, BDE позволяет работать им обоим, подразумевая, что они обращаются к одному и тому же NCD. Так, если пользователь A подключен к G:\DATA\BDENET, а пользователь B к H:\DATA\BDENET, BDE даст работать обоим.

Это полезно в одноранговой сети, где сервер также является и рабочей станцией. В этом случае некоторые операционные системы, перечисленные выше (см. начало раздела «Приложения BDE32 в одноранговой сети») не позволят серверу подключить сетевой диск к самому себе, так что сервер может использовать только диск C: (или D:, или какой-то другой локальный диск), а рабочая станция – нет, поскольку сама имеет собственный локальный диск C:.

Примечание

Никогда не допускайте ситуации (в любой сети), при которой имеется несколько пользователей, имеющих доступ к одним и тем же таблицам, но работающих с разными физическими NET-файлы. Это создает огромные проблемы, особенно в корпоративных и PTP-сетях.

Paradox DOS версии 4.0 использует ту же BDE-схему работы с сетью, что и таблицы Paradox. Необходимо учесть несколько важных моментов:

- Убедитесь, что включили опцию BDE Local Share, если вы создаете таблицы с общим доступом для приложений Pdox DOS и BDE.
- Из-за странного поведения при работе с сетевыми каталогами пути в файле контроля сети Paradox DOS у пользователей должны быть идентичны BDE-путям (например, тот же каталог и та же буква диска). Это должно быть сделано в случае, если и Paradox DOS, и BDE делают общими одни и те же таблицы и запущены оба приложения. Это может создать некоторые проблемы с установкой PTP-сетей.
- Убедитесь, что отключена опция BDE Strict Integrity, если создаете таблицы с общим доступом для приложений Paradox DOS и BDE. В противном случае BDE заблокирует пользователей Paradox DOS для редакти-

рования данных в таблицах Paradox (в любом каталоге), для которых установлена опция целостности данных (Referential Integrity).

- Убедитесь, что номер версии Paradox, имеющийся в настройках BDE, совместим с OLDEST версией Paradox DOS для использования в вашей сети. Установить ее можно, выбрав соответствующий драйвер Paradox в BDE Config Utility и проверив значение в поле LEVEL. Установите номер версии Paradox DOS, округлив его до ближайшего меньшего целого числа.

[News Group]

Решение проблемы BDE «Index out of Date»

После продолжительного исследования выяснилась причина. Оказалось, что она заключается в различных установках Paradox Language в BDE (v1 и V3) на странице Driver и System в утилите конфигурирования BDE.

[News Group]

Пример DBIDoRestructure

Как из приложения изменить размер поля или его тип?

Единственный способ изменить размер поля или его тип – обратиться к DBIDoRestructure. Вот простой пример, который поможет разобраться в этом:

```
function BDEStringFieldResize(ATable: TTable; AFieldName: string;
                              ANewSize: integer): boolean;
type
  TRestructStatus = (rsFieldNotFound, rsNothingToDo, rsDoIt);
var
  hDB: hDBIdb;
  pTableDesc: pCRTblDesc;
  pFldOp: pCROpType;           // фактически это массив array of pCROpType
  pFieldDesc: pFldDesc;        // фактически это массив array of pFldDesc
  CurPrp: CurProps;
  eRestrStatus: TRestructStatus;
  i: integer;
begin
  Result := False;
  eRestrStatus := rsFieldNotFound;
  AFieldName := UpperCase(AFieldName);
  pTableDesc := nil;
  pFieldDesc := nil;
  pFldOp := nil;
  with ATable do
    try
{ убедимся, что имеем исключительный доступ и сохраним dbhandle: }
      if Active and (not Exclusive) then Close;
```

```
      if (not Exclusive) then Exclusive := True;
      if (not Active) then Open;
      hDB := DBHandle;

{ готовим данные для DBIDoRestructure: }
      Check(DBIGetCursorProps(Handle, CurPrp));
      GetMem(pFieldDesc, CurPrp.iFields * SizeOf(FldDesc));
      Check(DBIGetFieldDescs(Handle, pFieldDesc));
      GetMem(pFldOp, CurPrp.iFields * SizeOf(CROpType));
      FillChar(pFldOp^, CurPrp.iFields * SizeOf(CROpType), 0);

{ ищем в цикле (через fielddesc) наше поле: }
      for i:=1 to CurPrp.iFields do begin

{ для ввода мы имеем серийные номера вместо Paradox ID, возвращаемых
  DbiGetFieldDescs: }
        pFieldDesc^.iFldNum := i;
        if (Uppercase(StrPas(pFieldDesc^.szName)) = AFieldName)
          and (pFieldDesc^.iFldType = fldZSTRING) then begin
            eRestrStatus := rsNothingToDo;
          if (pFieldDesc^.iUnits1 <> ANewSize) then begin
            pFieldDesc^.iUnits1 := ANewSize;
            pFldOp^ := crModify;
            eRestrStatus := rsDoIt;
          end;
        end;
        inc(pFieldDesc);
        inc(pFldOp);
      end;

{ "регулируем" массив указателей: }
      dec(pFieldDesc, CurPrp.iFields);
      dec(pFldOp, CurPrp.iFields);

{ в случае отсутствия операций возбуждаем исключение: }
      case eRestrStatus of
         rsNothingToDo:  raise Exception.Create('Ничего не сделано');
       rsFieldNotFound:  raise Exception.Create('Поле не найдено');
      end;
      GetMem(pTableDesc, SizeOf(CRTblDesc));
      FillChar(pTableDesc^, SizeOf(CRTblDesc), 0);
      StrPCopy(pTableDesc^.szTblName, TableName);
      StrPCopy(pTableDesc^.szTblType, szPARADOX);
      pTableDesc^.szTblType := CurPrp.szTableType;
      pTableDesc^.iFldCount := CurPrp.iFields;
      pTableDesc^.pecrFldOp := pFldOp;
      pTableDesc^.pfldDesc := pFieldDesc;
      Close;
      Check(DbiDoRestructure(hDB, 1, pTableDesc, nil, nil, nil, False));
    finally
      if pTableDesc <> nil then FreeMem(pTableDesc, SizeOf(CRTblDesc));
```

```
      if pFldOp<>nil then FreeMem(pFldOp, CurPrp.iFields * sizeOf(CROpType));
      if pFieldDesc <> nil then FreeMem(pFieldDesc,
        CurPrp.iFields * SizeOf(FldDesc));
      Open;
    end;        // пробуем с Table1
    Result := True;
end;
```

Пример использования DbiAddFilter

Этот пример устанавливает два фильтра. Первый, применяемый к Table1, выводит все записи, где в поле TYPE имеются значения "IN". Второй, применяемый к Table2, выводит все записи, где поле TYPE имеют значения, отличные от "IN".

```
procedure TForm1.Button1Click(Sender: TObject);
type
  TMyFilter = record
    Expr: CANExpr;
    Nodes: array[0..2] of CANNode;
    literals: array [0..7] of char;
  end;
const
  myFilter: TMyFilter =
    (Expr: (iVer: 1; iTotalSize: SizeOf(TMyFilter);
            iNodes: 3; iNodeStart: SizeOf(CANExpr);
            iLiteralStart: SizeOf(CANExpr) + 3 * SizeOf(CANNode));
     Nodes: ((canBinary: (nodeClass: nodeBinary;
                          canOP: canEQ;
                          iOperand1: SizeOf(CANNode);
               iOperand2: 2 * SizeOf(CANNode))),
             (canField: (nodeClass: nodeField;
                         canOP: canField2;
                         iFieldNum: 0; iNameOffset: 0)),
             (canConst: (nodeClass: nodeConst; canOP: canCONST2;
                         iType: fldZSTRING; iSize: 3; iOffset: 5)));
  literals: ('T', 'Y', 'P', 'E', #0, 'I', 'N', #0));

var
  dbResult: DBIResult;
  hFilter, hFilter1: hDBIFilter;
begin
{ procedure SetupFilter }
  Table1.Open;
  Table2.Open;
  dbResult := DbiAddFilter(Table1.Handle, 1, 1, False, addr(myFilter),
                           nil, hFilter);
  dbResult := DbiActivateFilter(Table1.Handle, hFilter);
  Table1.First;
  myFilter.nodes[0].canBinary.canOp := canNE;
```

```
    dbResult := DbiAddFilter(Table2.Handle, 1, 1, False, addr(myFilter),
                             nil, hFilter1);
    dbResult := DbiActivateFilter(Table2.Handle, hFilter1);
    Table2.First;
    myFilter.nodes[0].canBinary.canOp := canEQ;
    Table1.Close;
    Table2.Close;
end;
```

[News Group]

Примечание

Этот пример демонстрирует технику работы с фильтрами в Delphi до появления у компонента `Table` *свойства* `Filter`. *Переменную* `dbResult` *можно обрабатывать по своему усмотрению.*

Тот же результат сейчас можно получить, написав:

```
Table1.Filter :=  'TYPE=''IN''';
Table1.Filtered := True;
Table2.Filter :=  'TYPE<>''IN''';
Table2.Filtered := True;
```

Проблемы установки Interbase Server

После удаления родной uninstall Interbase Server 5.0 для Windows и при попытке поставить 5.1.1 вылетает ошибка: IBCheck. Что делать?

Надо запустить `regedit` и открыть ключ

```
HKEY_LOCAL_MACHINE\Environment
```

Там есть строковый параметр `PATH`. Почему-то его значение превращается в набор случайных символов. Этот параметр надо удалить и пересоздать как строчный, прописав туда прежнее содержимое.

Управление локальным сервером Interbase

Мой InterBase-сервер загружается при запуске Windows 95. Но я не вижу его пиктограмму в группе автозапуска. Как мне его выключить?

В системном реестре имеется ключ:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
```

Им можно отключить автоматическую загрузку сервера, удаляя соответствующий параметр.

Автоматический logon к локальной InterBase

Используйте компонент TDataBase. В строках Params пропишите:

```
SER NAME=sysdba
ASSWORD=masterkey
```

Затем установите свойство компонента TDataBase LoginPrompt в False. После этого с помощью свойства DataBaseName необходимо создать прикладной псевдоним (Alias) и связать TQuery/TTable с компонентом TDataBase.

Проблемы регистрации UDF

При попытке регистрации UDF возникает ошибка «udf not defined». Что не так?

Располагайте DLL в каталоге \Interbase\Bin или в одном из каталогов, в которых ОС обязательно будет произведен поиск этой библиотеки (для Windows это %SystemRoot% и %Path%);

При декларировании функции не следует указывать расширение модуля (в Windows по умолчанию .DLL):

```
declare external function f_SubStr
cstring(254), integer, integer
returns
cstring(254)
entry_point "Substr" module_name "UDF1"
```

Где UDF1 — UDF1.DLL

[Nomadic]

COLLATE PXW_CYRL по умолчанию

Как заставить Interbase принять COLLATE PXW_CYRL *по умолчанию?*

Чтобы не писать каждый раз COLLATE, создайте сохраненную процедуру:

```
create procedure fix_character_sets as begin
  update rdb$character_sets
  set rdb$default_collate_name = 'PXW_CYRL'
  where rdb$character_set_name = 'WIN1251'
  and rdb$default_collate_name = 'WIN1251';
end
```

Запустите ее один раз. Затем создайте таблицы без указания COLLATE. После восстановления из архива запустите еще раз.

[Nomadic]

Приращиваемые поля и Interbase

Я пытаюсь сгенерировать последовательный ключ для первичной ключевой колонки, но LIBS мне отвечает: «nested select is not support in this context» (вложенный выбор не поддерживается в данном контексте). Что делать?

Решение

```
CREATE TRIGGER AUTOINCREMENT FOR MYTABLE
  BEFORE INSERT AS
  DECLARE VARIABLE new_key INTEGER;
  BEGIN
    UPDATE AUTOKEYS
    SET KEY_VALUE = KEY_VALUE + 1
    WHERE (KEY_ID = "A");
  SELECT KEY_VALUE
    FROM AUTOKEYS
    WHERE KEY_ID = "A"
    INTO :new_key;
    new.my_key_column = new_key;
  END ^
```

Я пытаюсь добавить запись в таблицу InterBase, содержащую триггеры и BLOB-поля, тем не менее, всякий раз при выполнении метода post *после установки (append) значений, я получаю ошибку: «Record/Key deleted». (запись/ключ удален).*

Определение хранимой процедуры:

```
Create Procedure NewEmployeeKey Returns(EmployeeKey Integer) as begin
  EmployeeKey = Gen_Id(gnEmployeeKey, 1);
end
```

Определение триггера:

```
Create Trigger SetEmployeeKey for tbEmployee Active Before Insert Position 0 as
begin
  if (New.EmployeeKey is Null) then begin
    Execute Procedure NewEmployeeKey Returning_Values New.EmployeeKey;
  end
end
```

Код Delphi для использования в обработчике события OnNewRecord или AfterInsert, или BeforePost:

```
{ qyProviderData - это tQuery }
{ spProviderKey - это tStoredProc }
if qyProviderData.State in [dsInsert] then begin spProviderKey.ExecProc;
  qyProviderData.FieldByName('ProviderKey').AsInteger :=
       spProviderKey.ParamByName('ProviderKey').AsInteger;
end;
```

Это все, что необходимо. Хранимая процедура возвращает следующее сгенерированное значение. Триггер это гарантирует, даже если бы данные не бы-

ли доступны из Delphi-программы, первичный ключ все еще назначает значение. В коде Delphi можно проверять наличие пустого поля первичного ключа вместо `State in [dsInsert]`, хотя это тоже работает.

BLOB–поля Interbase

BLOB-поля InterBase существенно отличаются от полей другого типа. Реально BLOB-поле имеет несколько подтипов (sub-types). Знание подтипа BLOB-поля существенно при создании приложения для работы с базами данных, которые включают в себя BLOB-поля InterBase. Известны три подтипа BLOB-полей: два встроенных – подтипы 0 и 1 и пользовательский подтип.

Подтип 0 BLOB-поля создается при выполнении команды `CREATE`, когда подтип не определен. Для ясности, в синтаксисе SQL все же рекомендуется явно указывать, что BLOB-поле относится к подтипу 0. Данный подтип поля BLOB используется для хранения бинарных данных. InterBase никак не анализирует хранимые данные, просто помещает данные в BLOB-поле байт за байтом. Чаще всего BLOB-поля в приложениях Windows служат для хранения двоичных данных графических объектов, отображаемых обычно впоследствии компонентом `TDBImage`. Для этой цели подходит или BLOB-поле подтипа 0, или BLOB-поле пользовательского подтипа.

Второй встроенный подтип – 1 – разрабатывался для хранения текста. Обычно это данные свободного формата типа memo или заметок, отображаемых и редактируемых компонентом `TDBMemo`. Данный подтип BLOB-поля лучше подходит для хранения данных типа текст, чем поле, имеющее тип `VARCHAR`, поскольку, в отличие от поля типа `VARCHAR`, в режиме проектирования возможно задание ограничения по занимаемой области памяти.

С помощью синтаксиса SQL подтип 1 BLOB-поля создается с указанием типа BLOB-поля посредством ключевого слова `SUB_TYPE` и числа, указывающего на номер необходимого подтипа:

```
CREATE TABLE WITHBLOB (
  ID CHAR(3) NOT NULL PRIMARY KEY,
  MEMO BLOB SUB_TYPE 1, AMOUNT NUMERIC
)
```

Помимо двух встроенных подтипов BLOB-поля, существует также подтип, определяемый пользователем. Такой подтип задается отрицательным целым значением совместно с ключевым словом `SUB_TYPE`. Фактически учитывается только «отрицательность» целого числа, его значение может быть произвольным и остается на усмотрение того, кто создает таблицу. Указание числа -1 идентично указанию числа -2. Единственная рекомендация для применения подтипа, определяемого пользователем, – необходимо гарантировать, что в каждой строке таблицы BLOB-поле будет иметь только данный подтип, определяемый пользователем. InterBase не имеет критерия для оценки хранимого подтипа, поэтому вся ответственность по определению подходящего типа двоичных данных ложится на вас. Никакой ошибки со

стороны InterBase при загрузке неверного типа двоичных данных в пользовательский подтип BLOB-поля быть не может, но приложение может столкнуться с трудностями, если оно ожидает один тип данных, а ему передают другой.

BLOB-поле пользовательского подтипа может создаваться посредством следующего синтаксиса SQL:

```
CREATE TABLE IMAGE_DATA (
  FILENAME CHAR(12) NOT NULL PRIMARY KEY,
  BITMAP BLOB SUB_TYPE -1,
  EXEs BLOB SUB_TYPE -2,
)
```

В случае работы с таблицей, созданной с помощью приведенной выше команды, в поле `BITMAP` можно хранить один тип двоичных данных для всех записей. В нашем случае хранятся данные изображения. Поле `EXEs` подразумевает хранение выполнимых файлов, загружаемых с диска. Если приложение, использующее данную таблицу, по ошибке сохранит двоичные данные в поле `BITMAP` вместо `EXEs`, то InterBase ошибки не выдаст, но приложение при этом столкнется с серьезными трудностями при отображении в компоненте `TDBImage` сохраненного выполнимого файла.

BLOB-поля InterBase и Delphi

При определении объектов `TField` для BLOB-полей InterBase в Delphi следует относить различные подтипы BLOB-поля к производным типам `TField` следующим образом:

- Подтип 0: `TBlobField`
- Подтип 1: `TMemoField`
- Пользовательский: `TBlobField`

Поскольку как встроенный подтип 0, так и пользовательский подтип относятся к объектам `TBlobField`, то забота об определении подтипа во время проектирования приложения ложится на программиста. Единственный способ отличить подтип 0 от пользовательского подтипа заключается в просмотре информации о метаданных таблицы, что не может быть сделано с помощью Delphi. Метаданные таблицы можно просмотреть при помощи утилиты Local InterBase Server под названием WISQL.

InterBase BLOB-поля и Database Desktop

Утилита Database Desktop, поставляемая с Delphi (DBD), не создает пользовательские подтипы. При создании в Database Desktop BLOB-полей для хранения бинарных данных, включая данные изображения, используйте тип поля `BLOB`. Этим вы создадите BLOB-поле встроенного подтипа 0.

В DBD также возможно создание BLOB-поля типа `TEXT BLOB`. Это эквивалент встроенного подтипа 1 и в нем можно хранить текст свободного формата. Так как только он функционален встроенному подтипу 1 BLOB-поля, то при

просмотре таблицы утилитой WISQL обозначение его подтипа может отличаться от действительного.

Использование OLE с Interbase

Чтение из таблицы Interbase:

```
procedure TForm1.ReadOLE;
var
  BS: TBlobStream;
begin
  BS := TBlobStream.Create(Table1BLOBFIELD_BLOB, bmRead);
  OLEContainer1.LoadFromStream(BS);
  BS.Free;
end;
```

Запись в таблицу Interbase:

```
procedure TForm1.WriteOLE;
var
  BS: TBlobStream;
begin
  BS := TBlobStream.Create(Table1BLOBFIELD_BLOB, bmWrite);
  OLEContainer1.SaveToStream(BS);
  BS.Free;
end;
```

Interbase в Linux

Как скомпилировать UDF для Interbase под Linux RH 4.0?

Решение

```
#!/bin/sh
gcc -c -O -fpic udflib.c
ld -o libudf.so -shared udflib.o
cp libudf.so /usr/interbase/lib/
ldconfig -v >>/dev/null
```

[Nomadic]

Проблемы кириллицы в Oracle при работе с BDE

Как настроить Personal Oracle с русским языком на корректную работу с числами и BDE?

Определить в ключе \HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE параметр:

```
NLS_NUMERIC_CHARACTERS = '.,'
```

или после соединения с ORACLE выполнить

```
ALTER SESSION SET NLS_NUMERIC_CHARACTERS = '.,'
```

[Nomadic]

Связь Oracle с Windows 95

Откройте в Notepad или Write файл SQLNET.ORA (файл расположен в каталоге <ORA_HOME>\network\admin. Проигнорируйте любые другие разновидности этого файла). Данный файл должен выглядеть примерно следующим образом:

```
#################
# Filename......: sqlnet.ora
# Node..........: local.world
# Date..........: 24-MAY-94 13:23:20
#################
TRACE_LEVEL_CLIENT = OFF
sqlnet.expire_time = 15
names.default_domain = borland.world
name.default_zone = borland.world
```

Добавьте следующий параметр в файл SQLNET.ORA:

```
AUTOMATIC_IPC = OFF
```

После изменений файл должен выглядеть примерно так:

```
#################
# Filename......: sqlnet.ora
# Node..........: local.world
# Date..........: 24-MAY-94 13:23:20

#################
AUTOMATIC_IPC = OFF
TRACE_LEVEL_CLIENT = OFF
sqlnet.expire_time = 15
names.default_domain = borland.world
name.default_zone = borland.world
```

Сохраните измененный файл SQLNET.ORA. В дальнейшем при инициализации соединения с Oracle время соединения вместо 15 составит всего лишь 3 секунды. Скорость работы Delphi существенно увеличится.

Связь с Personal Oracle

Связаться с Personal Oracle – мудреное дело, но оно оказывается очень простым, если знать, как…

- Personal Oracle должно быть присвоено имя сервера (servername) «2;» (два и точка с запятой)
- Сетевой протокол (Net Protocol) ДОЛЖЕН БЫТЬ пустым (т. е. ничего не содержать)

Если вы работаете с Personal Oracle версии 7.1, в файле конфигурации сервера должна быть определена библиотека ORA71WIN.DLL, в противном случае выберите ORA7WIN.DLL.

Oracle в сети... Если вы можете соединиться через SQL*DBA или SQL*PLUS, то настройки вашего Oracle, вероятно, правильные. Кроме того, поставщик в файле конфигурации должен указывать версию базы данных.

```
The server name <имя сервера>:  SQL*Net V.1.x: Protocol <протокол>: ServerName <имя
сервера>: Database SID <версия>
```

Например:

```
T:222.122.22.32:DEMO
```

Помните, Protocol и SID чувствительны к регистру. Протокол T для TCP/IP. Адрес TCP/IP может быть заменен псевдонимом, если правильно сконфигурирован HOST-файл.

SQL*Net V.1.x: Здесь все сводится к Oracle-псевдониму. Используйте его как имя сервера (ServerName). Сетевой протокол (Protocol) должен отражать используемый протокол (TCP/IP, Named pipes, IPX/SPX и т. д.).

Анализ таблиц в Oracle

Как заставить Oracle анализировать все таблицы базы данных?

Конечно, можно использовать DBMS_SQL, DBMS_JOB.

А можно и так:

```
#!/bin/sh
#
# Analyze all tables
#
SQLFILE=/tmp/analyze.sql
LOGFILE=/tmp/analyze.log
echo @connect dbo/passwd@ > $SQLFILE
$ORACLE_HOME/bin/svrmgrl <> $SQLFILE
connect dbo/passwd
SELECT 'TABLE', TABLE_NAME FROM all_tables WHERE owner = 'DBO';
EOF
echo exit >> $SQLFILE
cat $SQLFILE > $LOGFILE
cat $SQLFILE | $ORACLE_HOME/bin/svrmgrl >> $LOGFILE
cat $LOGFILE | /usr/bin/mailx -s 'Analyze tables' tlk@nbd.kis.ru
rm $SQLFILE
rm $LOGFILE
```

[Nomadic]

Проблемы с Oracle в режиме отладки

В режиме отладки приложения не разрешается доступ к базе данных (открытие). Как исправить?

Необходимо отключить (деинсталлировать через Oracle Installer) Trace Service на клиенте – совет от ORACLE.

Проблемы могут быть только под Windows NT 4.xx.

[Nomadic]

SQL в Delphi

Delphi поддерживает статический и динамический SQL. В Delphi имеется объект `TQuery`, который используется для хранения и выполнения SQL запросов.

Свойство `TQuery` SQL содержит текст запросов SQL, выполняемых `TQuery`. Данное свойство имеет тип `TStrings`, означающее, что оно может хранить в списке целую серию строк. Список ведет себя подобно массиву, но в действительности это специальный класс с уникальными возможностями.

Компонент `TQuery` позволяет выполнять SQL-запросы двух типов:

- статические запросы;
- динамические запросы.

Статический запрос SQL устанавливается во время проектирования и не содержит никаких параметров или переменных. Например, следующая строка является статическим SQL-запросом:

```
SELECT * FROM CUSTOMER WHERE CUST_NO = 1234
```

Динамический запрос SQL или, как его еще называют, параметрический запрос, включает в себя параметры для колонок или имени таблицы. Например, следующая строка является динамическим SQL-запросом:

```
SELECT * FROM CUSTOMER WHERE CUST_NO = :Number
```

Переменная `Number`, указанная после двоеточия, – параметр, который назначается пользователем во время выполнения приложения. Во время выполнения запроса параметр может изменяться.

Delphi-приложения могут использовать SQL для получения доступа к следующим БД:

- Таблицы Paradox или dBASE, использующие локальный SQL. Допустимый синтаксис является подмножеством стандарта ANSI standard SQL и включает основные запросы `SELECT`, `INSERT`, `UPDATE` и `DELETE`. Для получения дополнительной информации о локальном синтаксисе SQL обратитесь к справке «Using Local SQL».

- Базы данных Local InterBase Server, включая Local InterBase Server. Допускаются любые запросы InterBase SQL. Для получения дополнительной информации о синтаксисе и ограничениях обратитесь к электронной справке «SQL Statement and Function Reference».
- Базы данных на удаленных серверах баз данных (только в версии Delphi Client/Server). По-видимому, вы установили подходящий SQL Link. В SQL-серверах допускаются любые стандартные запросы SQL. Для получения дополнительной информации о синтаксисе и ограничениях обратитесь к электронной справке вашего сервера.

Delphi также поддерживает разнородные запросы к более чем одному серверу или типу таблицы (для примера, данные из таблицы Oracle и таблицы Paradox). Для получения дополнительной информации обратитесь к электронной справке «Creating Heterogeneous Queries» (создание гетерогенных запросов).

Обработка транзакций в приложениях

Delphi-приложения поддерживают следующие способы управления транзакциями:

- неявно, автоматически стартуя и запуская транзакции, когда приложение пытается передать данные (`Post data`);
- явно, в зависимости от уровня управления, требующемуся вашему приложению (рекомендуемый метод): методы `TDataBase StartTransaction`, `Commit` и `Rollback`;
- `Passthrough` (транзитная пересылка) SQL в компоненте `TQuery`. Ваше приложение должно использовать специфические серверные SQL-запросы управления транзакциями, и вы должны понять, как управляются транзакции вашим сервером.

Зарезервированные слова Local SQL

Ниже приведен список слов в алфавитном порядке, зарезервированных Local SQL в Borland Database Engine. Имейте в виду, что данный совет публикуется «как есть».

- `ACTIVE, ADD, ALL, AFTER, ALTER, AND, ANY, AS, ASC, ASCENDING, AT, AUTO, AUTOINC, AVG`
- `BASE_NAME, BEFORE, BEGIN, BETWEEN, BLOB, BOOLEAN, BOTH, BY, BYTES`
- `CACHE, CAST, CHAR, CHARACTER, CHECK, CHECK_POINT_LENGTH, COLLATE, COLUMN, COMMIT, COMMITTED, COMPUTED, CONDITIONAL, CONSTRAINT, CONTAINING, COUNT, CREATE, CSTRING, CURRENT, CURSOR`
- `DATABASE, DATE, DAY, DEBUG, DEC, DECIMAL, DECLARE, DEFAULT, DELETE, DESC, DESCENDING, DISTINCT, DO, DOMAIN, DOUBLE, DROP`
- `ELSE, END, ENTRY_POINT, ESCAPE, EXCEPTION, EXECUTE, EXISTS, EXIT, EXTERNAL, EXTRACT`

- FILE, FILTER, FLOAT, FOR, FOREIGN, FROM, FULL, FUNCTION
- GDSCODE, GENERATOR, GEN_ID, GRANT, GROUP, GROUP_COMMIT_WAIT_TIME
- HAVING, HOUR
- IF, IN, INT, INACTIVE, INDEX, INNER, INPUT_TYPE, INSERT, INTEGER, INTO, IS, ISOLATION
- JOIN
- KEY
- LONG, LENGTH, LOGFILE, LOWER, LEADING, LEFT, LEVEL, LIKE, LOG_BUFFER_SIZE
- MANUAL, MAX, MAXIMUM_SEGMENT, MERGE, MESSAGE, MIN, MINUTE, MODULE_NAME, MONEY, MONTH
- NAMES, NATIONAL, NATURAL, NCHAR, NO, NOT, NULL, NUM_LOG_BUFFERS, NUMERIC
- OF, ON, ONLY, OPTION, OR, ORDER, OUTER, OUTPUT_TYPE, OVERFLOW
- PAGE_SIZE, PAGE, PAGES, PARAMETER, PASSWORD, PLAN, POSITION, POST_EVENT, PRECISION, PROCEDURE, PROTECTED, PRIMARY, PRIVILEGES
- RAW_PARTITIONS, RDB$DB_KEY, READ, REAL, RECORD_VERSION, REFERENCES, RESERV, RESERVING, RETAIN, RETURNING_VALUES, RETURNS, REVOKE, RIGHT, ROLLBACK
- SECOND, SEGMENT, SELECT, SET, SHARED, SHADOW, SCHEMA, SINGULAR, SIZE, SMALLINT, SNAPSHOT, SOME, SORT, SQLCODE, STABILITY, STARTING, STARTS, STATISTICS, SUB_TYPE, SUBSTRING, SUM, SUSPEND
- TABLE, THEN, TIME, TIMESTAMP, TIMEZONE_HOUR, TIMEZONE_MINUTE, TO, TRAILING, TRANSACTION, TRIGGER, TRIM
- UNCOMMITTED, UNION, UNIQUE, UPDATE, UPPER, USER
- VALUE, VALUES, VARCHAR, VARIABLE, VARYING, VIEW
- WAIT, WHEN, WHERE, WHILE, WITH, WORK, WRITE
- YEAR

Операторы:

`||`, `-`, `*`, `/`, `<>`, `<`, `>`, `,`(запятая), `=`, `<=`, `>=`, `~=`, `!=`, `^=`, `(`, `)`.

Параметризованные запросы

Как передать переменную в запрос?

Сначала нужно создать запрос, использующий параметр:

```
SELECT Test."FName", Test."Salary Of Employee"
  FROM Test
  WHERE Test."Salary of Employee" > :val
```

Примечание

Если просто написать имя поля как "Salary of Employee"*, получите ошибку* "Capability Not Supported". *Это должно быть просто* Test."Salary of Employee".

В нашем случае имя параметра val, но может быть любое другое. Затем переходите к свойству TQuery Params и устанавливаете параметр val в зависимости от требуемого типа. В примере это тип integer.

Затем нужно создать код, устанавливающий значение параметра. Для задания значения будем использовать компонент TEdit.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  with Query1 do begin
    Close;
    ParamByName('val').AsInteger := StrToInt(Edit1.Text);
    Open;
  end;
end;
```

Примечание

Рекомендуем в качестве меры предосторожности разместить приведенный выше код в блоке try..except.

В запросе можно использовать ключевое слово LIKE:

```
SELECT * FROM CUSTOMER

  WHERE Company LIKE :CompanyName
```

Примечание

Код работает с таблицей пользователя, расположенной в каталоге C:\Program Files\Common Files\Borland Shared\Data. *При этом возможно применение псевдонима* DBDEMOS.

Код Delphi:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  with Query1 do begin
    Close;
    ParamByName('CompanyName').AsString := Edit1.Text + '%';
    Open;
  end;
end;
```

Альтернативный способ передачи параметра (с последующим использованием ParamByName) – params[TheParameterNumber].

Демонстрация такого способа:

```
ParamByName('CompanyName').AsString := Edit1.Text + '%';
```

или в качестве альтернативы:

```
Params[0].AsString := Edit1.Text + '%';
```

Хитрость шаблона – в конкатенирующем знаке процента в конце параметра.

Имя таблицы в SQL-запросе

Этот SQL не работает:

```
RequestLive := True
SELECT * FROM DBO.TABLE1
```

А этот работает:

```
RequestLive := True
SELECT * FROM "DBO.TABLE1"
```

Помните о том, что кавычки необходимы для обозначения имени таблицы. Также имейте в виду, что поставляемая документация говорит о том, что для таблиц Oracle кавычки не нужны. Это неверно – нужны.

Интерактивные SQL-запросы

Как передать значение переменной в запросе SQL? К примеру, в обработчике `onClick` *клавиши вывести все записи с величиной поля большей, чем задал пользователь. Можно ли в Delphi создать что-либо подобное механизму запросов, реализованному в Paradox for Windows?*

Решение этой задачи в Delphi подобно созданию и выполнению строки запроса SQL в Paradox.

Paradox:

```
method pushButton(var eventInfo Event)

var
  s string
  q query
  d database

endvar
  d.open("MYALIAS")
  s = "select * from mytable where somefield=\"" + entryField.value + "\""
  q.readFromString(s)
  q.executeSQL(d)
endmethod
```

Delphi:

```
procedure TForm1.Button1Click(Sender: TObject);

begin
  MyQuery.Active := False;
  MyQuery.SQL.Clear;
  MyQuery.SQL.Add('select * from mytable where somefield="'+ EntryField.Text +'"');
  MyQuery.Active := True;
end;
```

SQL–запросы в изменившейся структуре базы данных

В процессе работы программы изменилась структура базы данных (alter table etc.). Программа продолжала успешно открывать таблицы, но запросы посылались в соответствии со старой схемой данных. В чем проблема?

В установках BDE (Configuration utility или BDEAdmin) можно выставить SCHEMA CACHE = FALSE (не кэшировать схему данных).

Но в некоторых случаях ошибки такого рода все-таки происходят. В таком случае необходимо вызывать метод TDataBase.FlushSchemaCache после каждого изменения метаданных.

[News Group]

SQL – суммирование вычисляемого поля

Бывают случаи, когда в приложении Delphi, которое для получения доступа к данным использует SQL, необходимо узнать сумму вычисленных данных. Другими словами, необходимо с помощью SQL создать вычисляемое поле и применить к нему функцию SUM.

При выполнении такой операции с SQL-таблицами (например, Local InterBase Server), все достаточно тривиально, и сумма вычисляется простым использованием функции SUM с указанием поля. Вот пример с демонстрационной таблицей EMPLOYEE (из базы данных EMPLOYEE.GDB):

```
SELECT SUM(SALARY / 12)
  FROM EMPLOYEE
```

Та же самая методика применима в случае возвращаемого набора данных, в котором значения группируются в другом столбце с помощью утверждения GROUP BY:

```
SELECT EMP_NO, SUM(SALARY / 12)
  FROM EMPLOYEE
  GROUP BY EMP_NO
  ORDER BY EMP_NO
```

Пока базы данных SQL поддерживают суммирование вычисляемых полей, локальный SQL этого делать не будет. Для получения результатов нужны другие методы, например копирование результатов запроса с вычисляемым полем во временную таблицу (как и в случае компонента TBatchMove) и использование компонента TQuery для вычисления суммы данных во временной таблице.

SQL – сортировка вычисляемого поля

Иногда схема данных требует, чтобы набор данных имел вычисляемый результат. В приложениях Delphi в случае применения SQL это возможно, но данная технология немного разнится в зависимости от используемого типа данных.

Для локального SQL, включая таблицы Paradox и dBASE, вычисляемому полю дают имя с указанием ключевого слова AS. При этом допускается ссылаться на такое поле для задания порядка сортировки с помощью ключевой фразы ORDER BY в запросе SQL. Пример с демонстрационной таблицей ITEMS.DB:

```
SELECT I."PARTNO", I."QTY", (I."QTY" * 100) AS TOTAL
  FROM "ITEMS.DB" I
  ORDER BY TOTAL
```

В данном примере вычисляемому полю было присвоено имя TOTAL (временно, только для ссылки), после чего оно стало доступным в запросе SQL для выражения ORDER BY.

Вышеуказанный метод не поддерживается в InterBase. Тем не менее, сортировать вычисляемые поля в таблицах InterBase все же возможно. Вместо имени вычисляемого поля в выражении ORDER BY используется число, представляющее собой позицию вычисляемого поля в списке полей таблицы. Небольшой пример с демонстрационной таблицей EMPLOYEE (расположенной в базе данных EMPLOYEE.GDB):

```
SELECT EMP_NO, SALARY, (SALARY / 12) AS MONTHLY
  FROM EMPLOYEE
  ORDER BY 3 DESCENDING
```

Тогда как в таблицах InterBase и LIBS реализован второй метод и в них не доступен первый, с обоими методами позволяет работать локальный SQL. Вот пример SQL-запроса для таблицы Paradox, приспособленного для работы с относительной позицией вычисляемого поля, а не с его именем:

```
SELECT I."PARTNO", I."QTY", (I."QTY" * 100) AS TOTAL
  FROM "ITEMS.DB" I
  ORDER BY 3
```

Синтаксис SQL–функции Substring

Каков синтаксис SQL-функции `SUBSTRING()`*?*

Решение

```
SUBSTRING('Delphi - это супер!!!' from 1 to 6)
```

SQL и расширенные символы

Я использую поле `TMemo`*. Все прекрасно, до тех пор пока я не начинаю работать с расширенными символами (с кодом больше 127). Текст с этими символами послать могу, но при последующем поиске записи получаю такое сообщение об ошибке: «General SQL error: Cannot transliterate character between character sets.» (Общая ошибка SQL: не могу сопоставить символ с имеющимися наборами символов). Как исправить?*

Нужно установить встроенный набор символов `LIBS` в `DEFAULT CHARACTER SET ISO8859_1`. Созданная впоследствии таблица будет использовать данный набор символов для хранимых `alphanumeric`-данных и BLOB-полей. Псевдоним базы данных должен быть установлен на драйвер языка BLLT1FR.

После этих изменений и пересоздания базы данных все заработает.

Может быть, не стоило публиковать эту информацию, поскольку многие пользователи работают с локальными драйверами, наборами символов и новой версией SQL. Но эта информация отсутствует в описании. Она присутствует только в LIBS-файле `Readme`.

SQL Server и проблемы StoredProc

Обнаруженная проблема заключается в ненормальной работе BDE с `TStoredProc`, когда хранимая процедура SQL получает на входе параметр типа `String`. BDE/SQL Links перед вызовом хранимой процедуры заносит в строку управляющие символы.

Чтобы обойти эту проблему, Borland предлагает использовать `TQuery`. Конечно, ничего не стоит перевести `TStoredProcs` в `TQuery` (с сохранением полного набора характеристик и без потери скорости). Но мне стала интересна причина такого поведения компонента, и я решил покопаться в `TStoredProc`, для чего добавил к хранимой процедуре дополнительный параметр, позволяющий указывать длину передаваемой процедуре строки. Затем, уже в процедуре, если реальная длина строки оказывалась больше, то дополнительно передаваемый параметр позволял в дальнейшем работать только с левой частью строки, а остальные управляющие символы игнорировались.

Решение

Приведенная ниже процедура SQL Server возвращает 1, если таблица существует, и 2 в противном случае.

```
CREATE PROCEDURE up_TableExists (@TableName varchar(50),
                                 @TableNameLen int = null) AS
  declare @CleanTblName varchar(50)
  if @TableNameLen is not null
    select @CleanTblName = SubString(@TableName, 1, @TableNameLen)
  else
    select @CleanTblName = @TableName
  if EXISTS (SELECT name FROM sysobjects WHERE name = @CleanTblName)
    RETURN 1
  else
    RETURN 2
```

В Delphi, прежде чем вызвать ExecProc, установите параметр длины строки.

Пример вызова хранимой процедуры в Delphi:

```
var
  sTableName: String;
  rc: Boolean;
  ...
  sTableName := 'MyTable';
  with StoredProc1 do begin

    ...
    ParamByName('@TableName').AsString := sTableName;
{ обход проблемы: передаем длину строки SQL Server для обработки
  хранимой процедурой }
    ParamByName('@TableNameLen').AsInteger := Length(sTableName);
    Prepare;
    ExecProc;
    rc := ParamByName('Result').AsInteger = 1;  // rc True если result = 1
    if rc then
      ....
  end;
```

[News Group]

SQL – применение функции SUBSTRING

SQL-функция SUBSTRING может использоваться в приложениях Delphi, работающих с запросами к локальной SQL, но она не поддерживается при работе с таблицами InterBase (IB) и Local InterBase Server (LIBS). Ниже приведен синтаксис функции SUBSTRING, примеры ее использования в запросах к Local SQL и альтернатива для возвращения тех же результатов для таблиц IB/LIBS.

Синтаксис функции SUBSTRING:

```
SUBSTRING(<column> FROM <start> [, FOR <length>])
```

Где:

- <column> – имя колонки таблицы, из которой должна быть получена подстрока (substring);
- <start> – место в значении колонки, начиная с которого извлекается подстрока;
- <length> – длина извлекаемой подстроки.

Функция SUBSTRING в примере ниже возвратит второй, третий и четвертый символы из колонки с именем COMPANY:

```
SUBSTRING(COMPANY FROM 2 FOR 3)
```

Функция SUBSTRING может быть использована и для списка полей в запросе SELECT, где ключевое слово WHERE допускает сравнение значения с определенным набором колонок. Функция SUBSTRING может применяться только с колонками типа String (в языке SQL тип CHAR). Вот пример функции SUBSTRING со списком колонок в запросе SELECT (используем демонстрационную таблицу Paradox CUSTOMER.DB):

```
SELECT (SUBSTRING(C."COMPANY" FROM 1 FOR 3)) AS SS
  FROM "CUSTOMER.DB" C
```

Данный SQL-запрос извлекает первые три символа из колонки COMPANY, возвращаемой как вычисляемая колонка с именем SS. Вот пример функции SUBSTRING из SQL-запроса с ключевым словом WHERE (используем ту же самую таблицу):

```
SELECT C."COMPANY"
  FROM "CUSTOMER.DB" C
  WHERE SUBSTRING(C."COMPANY" FROM 2 FOR 2) = "an"
```

Данный запрос возвратит все строки таблицы, где второй и третий символы в колонке COMPANY равны "an".

Поскольку функция SUBSTRING не поддерживается в базах данных IB и LIBS, операции с подстроками со списком колонок в запросе невозможны.

Исключение

IB может работать с подстроками через функции, определяемые пользователем, User-Defined Functions.

Но с помощью оператора LIKE и сопутствующих символьных маркеров подстановки можно работать с подстрокой и в случае WHERE. Пример на основе таблицы EMPLOYEE (в базе данных EMPLOYEE.GDB):

```
SELECT LAST_NAME, FIRST_NAME
  FROM EMPLOYEE
  WHERE LAST_NAME LIKE "_an%"
```

Данный SQL-запрос возвратит все строки таблицы, где второй и третий символы в колонке LAST_NAME равны "an", см. предыдущий пример на основе таблицы Paradox. Базам данных IB и LIBS для выполнения сравнения подстроки в операторе запроса WHERE данный метод необходим (и невозможно воспользоваться функцией SUBSTRING), в случае таблиц Paradox и dBASE (например, Local SQL) можно применить любой метод.

SQL и пробельные символы

Выполнение SQL-запросов в Delphi-компоненте TQuery (или специального средства SQL-запроса в Database Desktop, Visual dBASE или Paradox for Windows) требует специального синтаксиса для любых колонок, содержащих пробелы или специальные символы.

Пользуясь таблицей Biolife.DB из демонстрационных данных Delphi, проиллюстрируем соблюдение любых специфических требований синтаксиса. Запрос SQL Select мог бы быть сформирован следующим образом:

```
SELECT Species No, Category, Common_Name,
Species Name, Length (cm), Length_In,
    Notes, Graphic
FROM BIOLIFE
```

Причиной синтаксической ошибки могут стать пробелы в номерах и именах колонок, круглые скобки и другие символы.

Для коррекции синтаксиса в вышеприведенное SQL-выражение необходимо внести два изменения. Во-первых, любые колонки, содержащие пробелы или специальные символы, должны быть заключены в двойные или одинарные кавычки (апострофы). Во-вторых, ссылке на имя колонки должна предшествовать ссылка на саму таблицу и точка после нее. Второе требование особенно важно, поскольку заключенная в кавычки строка интерпретируется не как строковое выражение, а как значение колонки. Ниже приведено правильно отформатированное выражение:

```
SELECT BIOLIFE."Species No", BIOLIFE."Category",
     BIOLIFE."Common_Name",  BIOLIFE."Species Name",
     BIOLIFE."Length (cm)", BIOLIFE."Length_In",
     BIOLIFE."Notes", BIOLIFE."Graphic"
FROM "BIOLIFE.DB" BIOLIFE
```

В приведенном выше примере табличный псевдоним BIOLIFE используется в качестве ссылки на саму таблицу и располагается перед именем колонки. Данная ссылка может принимать форму имени псевдонима, реального имени таблицы или ссылаться на имя файла при использовании таблиц dBASE или Paradox. Следующее SQL-выражение также работает.

```
SELECT BIOLIFE."Species No", BIOLIFE.Category,
     BIOLIFE.Common_Name, BIOLIFE."Species Name",
     BIOLIFE."Length (cm)", BIOLIFE.Length_In,
```

```
    BIOLIFE.Notes, BIOLIFE.Graphic
FROM BIOLIFE
```

Примечание

Данное SQL-выражение может использоваться при условии, что необходимый псевдоним уже открыт. В случае `TQuery` *это означает, что псевдоним определен в свойстве* `DatabaseName`.

Если псевдоним недоступен, то для таблицы должен быть определен путь целиком, например, так:

```
SELECT
  "C:\Delphi\DEMOS\DATA\BIOLIFE.DB"."Species No",
  "C:\Delphi\DEMOS\DATA\BIOLIFE.DB"."Category",
  "C:\Delphi\DEMOS\DATA\BIOLIFE.DB"."Common_Name",
  "C:\Delphi\DEMOS\DATA\BIOLIFE.DB"."Species Name",
  "C:\Delphi\DEMOS\DATA\BIOLIFE.DB"."Length (cm)",
  "C:\Delphi\DEMOS\DATA\BIOLIFE.DB"."Length_In",
  "C:\Delphi\DEMOS\DATA\BIOLIFE.DB"."Notes",
  "C:\Delphi\DEMOS\DATA\BIOLIFE.DB"."Graphic"
FROM
  "C:\Delphi\DEMOS\DATA\BIOLIFE.DB"
```

Наконец, есть два средства автоматического форматирования специального синтаксиса. Первым является Visual Query Builder, включаемый в версию Client/Server Delphi. Visual Query Builder выполняет такое форматирование автоматически, по мере создания запроса. Другое средство – Database Desktop Show SQL, доступный при создании и редактировании запроса QBE-типа. После выбора пункта меню Query|Show SQL отображаемый SQL текст может быть вырезан и вставлен куда необходимо.

Неработающий SQL OR

Я заполнил таблицу 10 записями и создал запрос `Select` *с* `OR`:

```
select * from "mytable" where ((acreage=5.5) or (acreage=6))
```

Это работает. Затем на основе поля `acreage` *был создан вторичный индекс – и запрос не сработал. В чем причина?*

Наличие вторичного индекса не позволяет нормально отработать запросу SQL.

[News Group]

Функции работы с датами в SQL

Как получить месяц или год из поля `DATETIME` *с помощью SQL?*

Решение

```
SELECT SALEDATE,
  EXTRACT(DAY FROM SALEDATE) AS DD,
```

```
  EXTRACT(MONTH FROM SALEDATE) AS MM,
  EXTRACT(YEAR FROM SALEDATE) AS YY
FROM ORDERS
```

[News Group]

Сиротские Master-записи

Как с помощью SQL найти записи таблицы, которых нет в другой таблице?

Решение

```
with PeopleHiddenForm.PersonQuery.SQL do begin
  Add('Select P.Last, P.First, P.Middle, P."Suffix", P.KeyNo,
       COUNT(PersMemL.PersonKeyNo)');
  Add('From Person P Left Outer Join ');
  Add('PersMemL PersMemL');
  Add('On ((P.KeyNo = PersMemL.PersonKeyNo))');
  Add('Group By P.Last, P.First, P.Middle, P.Suffix, P.KeyNo');
  Add('Having ((Count(PersmemL.PersonKeyNo) = 0))');
```

Данный код позволяет связаться с таблицей (PersMemL), содержащей количество ключей персональной записи и запись членства. Запрос возвращает имена персон, для которых нет записей членства.

На практике этот способ оказывается очень эффективным, по крайней мере, с локальным SQL в таблицах Paradox.

[News Group]

Refresh для запросов

Как научить VCL правильно выполнять Refresh *для запросов?*

Здесь может оказаться полезным DoRefreshQuery.

```
function GetFieldNamesStr(DataSet: TDataSet): String;
var
  I: Integer;
begin
  Result := '';
  with DataSet do
    for I := 0 to FieldCount - 1 do
      Result := Result + Fields[I].FieldName + ';';
end;

procedure DoRefreshQuery(Query: TQuery; KeyFields: String;
                         BookMarkSearch: Boolean);
var
  Fields: TList;
  KeyValues: Variant;
  KeyNames: String;
```

```
  Bmk: TBookmark;
  I: Integer;
  BookmarkFound: Boolean;
  CanLocate: Boolean;
begin
  Fields := TList.Create;
  if KeyFields = '' then KeyFields := GetFieldNamesStr(Query);
  try
    Query.GetFieldList(Fields, KeyFields);
    for I := Fields.Count - 1 downto 0 do
      with TField(Fields[I]) do
        if Calculated or Lookup then Fields.Delete(I);
    CanLocate := Fields.Count > 0;
    if CanLocate then begin
      if Fields.Count = 1 then KeyValues := TField(Fields[0]).Value
      else begin
        KeyValues := VarArrayCreate([0, Fields.Count - 1], varVariant);
        KeyValues[0] := TField(Fields[0]).Value;
      end;
      KeyNames := TField(Fields[0]).FieldName;
      for I := 1 to Fields.Count - 1 do begin
        KeyNames := KeyNames + ';' + TField(Fields[I]).FieldName;
        KeyValues[I] := TField(Fields[I]).Value;
      end;
    end;
  finally
    Fields.Free;
  end;
  with Query do begin
    Bmk := nil;
    DisableControls;
    try
      BookmarkFound := False;
      if BookMarkSearch then Bmk := GetBookmark;
      Close;
      Open;
      if Assigned(Bmk) then
        try
          GotoBookMark(Bmk);
          BookmarkFound := True;
        except
      end;
      if not BookmarkFound and CanLocate then Locate(KeyNames, KeyValues, []);

    finally
      EnableControls;
      Screen.Cursor := crDefault;
      FreeBookmark(Bmk);
    end;
  end;
end;
```

```
procedure TQuery.RefreshParams;
var
  DataSet: TDataSet;
begin
  DisableControls;
  try
    if FDataLink.DataSource <> nil then begin
      DataSet := FDataLink.DataSource.DataSet;
      if DataSet <> nil then
        if DataSet.Active and (DataSet.State <> dsSetKey) then
          DoRefreshQuery(Self, GetFieldNamesStr(Self), False);
    end;
  finally
    EnableControls;
  end;
end;
```

[News Group]

Примечание

Дополнительную информацию можно почерпнуть в `dbtables.pas`.

Default Cursor после завершения выполнения запросов

Почему курсор мыши не возвращается в исходное состояние (не становится обычной стрелкой) после выполнения запроса?

При выполнении открытого запроса Delphi изменяет курсор, и произойти это может даже в середине события, например, при нажатии на кнопку. Приведенный ниже пример отобразит курсор в виде песочных часов (SQL Hourglass Icon) после того, как закроется окно с сообщением. При этом мышь будет вести себя так, как будто находится в режиме «стрелки».

```
// Добавьте к обработчику события нажатия кнопки, использование запроса при этом не
// имеет значения, например: Select * from Customer (в IBLocal)
  with Query1 do begin
    Close;
    Open;
    ShowMessage(IntToStr(RecordCount));
  end;
```

При наступлении события Delphi пробует обратно придать курсору тип стрелки (`Arrow`), при этом выводится новая форма (диалог `ShowMessage`), которая препятствует автоматическому переводу курсора в режим стрелки.

Для решения этой проблемы нужно добавить `Application.ProcessMessages` прежде, чем форма будет показана, это позволит обработать все сообщения, скопившиеся в очереди (и очистить ее), после чего курсор мыши вновь примет нормальную форму.

```
// Добавьте к обработчику события нажатия кнопки, использование запроса при этом не
// имеет значения, например: Select * from Customer (в IBLocal)
  with Query1 do begin
    Close;
    Open;
    Application.ProcessMessages;          // Добавьте эту строку.
    ShowMessage(IntToStr(RecordCount));
  end;
```

32-битное соединение с сервером Sybase

Шаги для подключения:

1. Убедитесь, что пакет SQL Links установлен на вашем локальном диске. При полной установке Delphi 2.x он должен быть уже установлен в системе.
2. Инсталлируйте клиентское программное обеспечение Sybase.
3. При появлении в процессе установки диалога выбора 16- и 32-разрядной версии Sybase links выберите только 32-битную версию (установите флажок) и убедитесь в том, что опция 16-битной версии выключена.
4. После того как клиентское программное обеспечение будет установлено на жестком диске, у вас попросят разрешение на автоматическую программную коррекцию файла AUTOEXEC.BAT. Выберите YES.
5. На запрос по поводу редактирования файла SQL.INI ответьте YES.
6. В секции Input Server Name: (Введите имя сервера) укажите псевдоним сервера. Нажмите кнопку Add (Добавить) для внесения имени сервера в список Server Entry:. Затем убедитесь, что поля редактирования Service Type: (Тип сервиса), в котором следует указать query (запрос), Platform: (Платформа) – по умолчанию обычно устанавливается NT, DOS или Win3 и Net-Library Driver: (Драйвер сетевой библиотеки), а это должен быть NLWNSCK или NLNWLINK содержат верные сведения. Заполните поле редактирования Connection Information/Network Address: (Адрес информационного/сетевого соединения), введя сетевой адрес сервера, с которым надо установить соединение. Нажмите кнопку Add Service (Добавить сервис). Теперь можно проверить установку связи с сервером, нажимая кнопку Ping. Сохраните текущие настройки и выйдите из программы.
7. Завершите работу Windows и перегрузите машину.
8. В меню Пуск выберите программную группу Delphi и запустите Database Explorer.
9. В Database Explorer перейдите на вкладку Database. Активизируйте пункт меню Object|New... В поле со списком появившегося диалогового окна должно отображаться имя STANDARD. Щелкнув по стрелке, откройте список и выберите пункт SYBASE.
10. Теперь там должен быть псевдоним для вашего соединения с Sybase с именем SYBASE1. Убедитесь в том, что это имя выделено. Перейдите в

Database Explorer на следующую вкладку. В секции Server Name (Имя сервера) выберите имя одного из серверов, помещенного в SQL.INI, доступность которого надо проверить. В секции User Name укажите имя пользователя, имеющего права на доступ к определенному в секции Server Name серверу. Убедитесь в том, что вы знаете пароль только что назначенного пользователя.

11. Дважды щелкните по имени псевдонима (SYBASE1) и в появившемся диалоговом окне введите имя пользователя и его пароль. Имя пользователя должно совпадать с именем, определенным в секции User Name для псевдонима Sybase. Введите пароль, соответствующий данному пользователю. Нажмите кнопку OK. Теперь рядом с псевдонимом Sybase (SYBASE1) вы должны увидеть пиктограмму с изображением маленького зеленого ящика. Это означает успешное установление соединения.

Тестирование вашего соединения с помощью Delphi 2.x:

1. Разместите на пустой форме компоненты TDataSource, TTable и TDBGrid.
2. В Инспекторе объектов (Object Inspector) установите для TDataSource свойство DataSet в Table1.
3. В Инспекторе объектов установите для TTable имя базы данных в SYBASE1. Переместитесь ниже до свойства TableName и дважды щелкните в поле редактирования, расположенном около данного свойства. Должно появиться диалоговое окно с требованием ввести имя пользователя и его пароль. При этом должно уже отображаться имя пользователя, которое вы определили в Database Explorer для псевдонима Sybase. Введите соответствующий пароль. Нажмите кнопку OK.
4. Теперь вы должны увидеть список, состоящий из имен таблиц. Выберите одно.
5. Щелкните по компоненту TDBGrid. Присвойте его свойству DataSource значение DataSource1.
6. Установите свойство Active компонента TTable в True.
7. Теперь можно увидеть данные в TDBGrid. После запуска приложения должно появиться диалоговое окно с требованием ввести имя пользователя и его пароль. Введите пароль и нажмите кнопку OK. Теперь вы должны увидеть данные в табличной сетке.

Сообщения об ошибках

Ошибка, связанная с невозможностью нахождения сетевой библиотеки. Данная ошибка означает, что программе не удалось найти нужную ей .DLL. Следующие файлы должны располагаться в вашем каталоге \Sybase\DLL:

```
Libblk.dll
Libcomn.dll
Libcs.dll
Libct.dll
Libintl.dll
```

```
Libsrv.dll
Libsybdb.dll
Libtcl.dll
Mscvrt10.dll
Nldecnet.dll
Nlmsnmp.dll
Nlnwadvt.exe
Nlnwlink.dll
Nlwnsck.dll
```

Внимание

Данный документ не гарантирует установления соединения с сервером, он демонстрирует самый лучший и быстрый способ сделать это.

Ошибка BDE32 $2104

При обращении к функции `dbiGetDatabaseDesc` *получаю ошибку: «Возникла ошибка при попытке инициализации Borland Database Engine (ошибка $2104)». В чем причина?*

Если при вызове любой из функций BDE вы не пользуетесь компонентами для работы с базами данных, вам необходимо инициализировать BDE вызовом `dbiInit(nil)`.

[News Group]

Ошибка ApplyUpdates

Выполняем `ApplyUpdates`. *Если при выполнении* `insert` *(*`update`*) произошла ошибка (поле* `null`*, сработал* `check`*, etc.), то BDE всегда говорит «General SQL Error» вместо нормального сообщения об ошибке. В чем причина?*

Используйте нормальную трансляцию ошибок в `Application.OnException`.

```
procedure DBExceptionTranslate(E: EDBEngineError);
  function OriginalMessage: String;
  var
    I: Integer;
    DBErr: TDBError;
    S: String;
  begin
    Result := '';
    for I := 0 to E.ErrorCount - 1 do begin
      DBErr := E.Errors[I];
      case DBErr.NativeError of
          -836: begin                              { Interbase exception }
                  S := DBErr.Message;
                  Result := #13#10 + Copy(S, Pos(#10, S) + 1, Length(S));
                  Exit;
                end;
```

```
      end;
      S := Trim(DBErr.Message);
      if S <> '' then Result := Result + #13#10 + S;
    end;
  end;
begin
  case E.Errors[0].ErrorCode of
      $2204:          E.Message := LoadStr(SKeyDeleted);
      $271E, $2734:   E.Message := LoadStr(SInvalidUserName);
      $2815:          E.Message := LoadStr(SDeadlock);
      $2601:          E.Message := LoadStr(SKeyViol);
      $2604:          E.Message := LoadStr(SFKViolation) + OriginalMessage;
    else
      begin
        E.Message := Format(LoadStr(SErrorCodeFmt), [E.Errors[0].ErrorCode]) +
                              OriginalMessage;
      end;
  end;
end;
```

[Nomadic]

Примечание

Тексты сообщений об ошибках читаются из ресурса, созданного, например, с использованием `resourcestring`.

Ошибка создания дескриптора курсора

Следует применять `ExecSQL` вместо `Open`. Например, если имя запроса `UpdateStudent`, то при необходимости обновления STUDENT.DB необходимо использовать следующий код:

```
begin
  ...
  UpdateStudent.ExecSQL;
  ...
end;
```

Если SQL-запрос может возвратить значение – используйте `Open`. В противном случае следует предпочесть `ExecSQL`.

Ваш запрос представляет собой запрос `Passtrough`, который не может возвратить установленный результат, так что это не может быть открыто, а должно быть выполнено командой `ExecSQL`.

Нарушение уникальности записи

Создайте функцию, подобную следующей:

```
procedure DBError(DataSet: TDataSet; E: EDatabaseError; var Action: TDataAction);
```

```
const
  eKeyViol = 9729;
var
  iDBIError: Integer;

begin
  if (E is EDBEngineError) then begin
    iDBIError := (E as EDBEngineError).Errors[0].ErrorCode;
    case iDBIError of
      eKeyViol:  begin
                   MessageDlg('Нарушение уникальности записи ', mtWarning,
                              [mbOK], 0);
                   Halt;
                 end;
    end;
  end;
end;
```

Затем для каждой таблицы вашего приложения создайте следующий обработчик события:

```
procedure TMainForm.Table1EditError(DataSet: TDataSet; E: EDatabaseError;
                                    var Action: TDataAction);

begin
  DBError(Table1, E, Action);
end;
```

Таким образом можно перехватить массу ошибок.

Ошибка псевдонимов

Ошибка ODBC «INVALID CONFIGURATION PARAMETER FOR ALIAS {ALIAS_NAME}» (Неверно сконфигурирован параметр для псевдонима {имя псевдонима}) возникает при попытке получения доступа к базе данных через ODBC DSN на сервере ISAPI/NSAPI. Что я делаю не так?

Для того чтобы ODBC выступал как сервис и был доступен всем пользователям (гостевая учетная запись IIS), необходимо создать нестандартный SYSTEM DSN, а не пользовательский USER DSN, предоставляемый по умолчанию.

IIS, Novell и ошибки учетной записи

Когда мой клиент обращается к серверу, ISAPI DLL выдает сообщение «INVALID FILENAME». Мои таблицы PARADOX/DBASE *расположены на сервере* NOVELL. *В чем дело?*

Вы пропустили процедуру отображения диска (drive mapping) для вашей учетной записи IUSR_XXX. Ваши таблицы расположены на сервере Novell,

поэтому вам необходимо иметь диск, отображенный на нужный том сервера (где расположены таблицы).

Сопутствующая информация:

- Ваш web-сервер выполняется IIS как сервис. Никто не сможет войти в систему интерактивно.
- Клиентская страница является результатом работы ISAPI DLL. IIS регистрирует вас на сервере под учетной записью IUSR_XXX. Для того чтобы ISAPI DLL могли работать с таблицами, расположенными на сервере Novell, необходимо отобразить для пользователя соответствующий диск.
-
-

-

6

Мультимедиа

Мультимедиа–программирование в Delphi

Вы можете легко придать программе Delphi мультимедийные черты, добавив в нее звук, видео и музыку. Данный материал расскажет вам о том, как этого можно достичь с помощью элементов управления мультимедиа, доступных из среды разработки Delphi. Полный исходный код для всех программ доступен на сервере Compuserve.

При чтении данного материала вы узнаете о трех типах файлов, используемых в мультимедиа-программах:

- Файлы с расширением .AVI, содержащие видео. Примерами файлов такого типа могут служить ELEPHANT.AVI и FLOWERS.AVI.
- Файлы с расширением .MID, включающие в себя музыкальные фрагменты в формате MIDI (Musical Instrument Digital Interface). Примерами могут служить JAZZ.MID и VIVALDI.MID.
- Файлы с расширением .WAV включают в себя звуки, созданные с помощью технологии Microsoft WAVE. Например, CRASH.WAV или SPEECH.WAV.

Из трех обсуждаемых здесь форматов файлов, по размеру файлы AVI обычно самые большие. Даже сравнительно короткие AVI-фильмы длительностью до одной минуты занимают до 5 или даже до 10 Мбайт дискового пространства. Для уменьшения размера таких файлов применяются различные методы сжатия. WAV-файлы также велики, но они все-таки меньше, чем файлы AVI. MID-файлы небольшие по размеру, компактные и обеспечивают отличное качество звучания. Тем не менее, в них не так легко поместить звуки или голос, и они очень ограничены в музыкальном плане.

Delphi и мультимедиа

Delphi содержит в своей палитре компонент TMediaPlayer, который позволяет получить доступ в программе ко всем основным возможностям мультимедиа. Данный элемент управления очень легок в использовании. Фактически, он настолько прост, что многим начинающим программистам проще создать из двух-трех строк программу, воспроизводящую видео или музыку, чем написать приложение, выводящее фразу «Здравствуй, мир!». Для того чтобы знать, как управлять таким мощным элементом управления, необходимо помнить, что под его «капотом» на вас работает технология Microsoft, известная под названием Media Control Interface (MCI). Эта технология дает программистам доступ ко всему разнообразию мультимедиа. Компонент TMediaPlayer делает применение этих технологий интуитивно понятным и чрезвычайно легким.

Компонент TMediaPlayer

Для того чтобы начать создавать наше первое мультимедиа-приложение, создайте новый проект и выберите мультимедийный компонент управления (TMediaPlayer) на вкладке Additional Палитры компонентов. Поместите компонент на форму.

После этого в Инспекторе объектов будет доступно поле свойства FileName. Заполните его именем AVI-, MIDI- или WAVE-файла. Для примера, заполните это свойство строкой G:\AVI\CLOUDS.AVI, указывающей на имя видеофайла, размещенного на CD-ROM. Щелкнув по пиктограмме в правой части редактора свойств, можно вызвать диалог поиска нужного вам файла.

Завершив простые шаги, приведенные выше, можете запускать вашу мультимедиа-программу. Запустите и нажмите зеленую кнопку в левой части MediaPlayer для старта видео- или звукового файла, который вы перед этим выбрали. Если возникает ошибка, то для нее существует три вероятных причины:

- Указано неверное имя файла.
- В системе неправильно установлены компоненты мультимедиа. Это может означать, что отсутствует надлежащее аппаратное обеспечение или в системе не установлены правильные драйверы. Конфигурирование драйверов осуществляется через Панель управления, а аппаратные требования хорошо описаны в любой книге, посвященной мультимедиа.
- Свойство AutoOpen установлено в true, но не заполнено поле свойства FileName. Это вызовет следующее сообщение об ошибке: «This command requires a parameter. Please supply one» (Данная команда требует параметр. Пожалуйста, укажите его.).

После небольших экспериментов, проведенных с программой MEDIA1, вы обнаружите, что программа может проигрывать файлы AVI, MIDI и WAVE после простого изменения имени файла. Эта гибкость является результатом способности компонента анализировать свойство FileName, определять расшире-

ние файла и передавать его на обработку соответствующему внутреннему компоненту MediaPlayer во время выполнения приложения. Поместив компонент TMediaPlayer в окно, выбрав файл и запустив программу, можно обнаружить, что медиа-проигрыватель стартует в «действующем» режиме; это означает, что большинство управляющих кнопок доступно и готово к работе. Если вы закроете программу, установите свойство AutoOpen в False и перезапустите программу, медиа-проигрыватель не будет открытым при следующем запуске программы. При некоторых обстоятельствах это оказывается полезным, поскольку экономит время и ресурсы при открытии мультимедиа-драйверов Windows.

Для получения контроля над открытием и закрытием TMediaPlayer необходимо выполнить несколько простых шагов. Для начала разместите на форме кнопку и дайте ей имя Open. Дважды щелкните по кнопке для создания обработчика события OnClick и поместите в него следующий код:

```
procedure TForm1.Button1Click(Sender: TObject):

begin
  MediaPlayer.Open;
end;
```

При запуске программы вы увидите кнопки с универсальными символами для проигрывателя, CD или видеомагнитофона. Нажмите кнопку для открытия соответствующего MCI-устройства. Подождите немного, пока не изменится цвет кнопок MediaPlayer, после чего щелкните один раз по зеленой стрелке в левой части элемента управления. После этого начнется воспроизведение файла, определенного ранее в поле свойства FileName.

Доступ к мультимедиа

Некоторые программисты пытаются предоставить пользователям удобный и легкий способ воспроизведения файлов широкого диапазона. Для этого в программу встраиваются возможности получения доступа к жесткому диску и CD-ROM, выбора файлов различных форматов и их воспроизведения. Такой тип приложений иногда называют *программами медиа-доступа*.

Чтобы начать создавать приложение MediaAccess, расположите на форме компонент TMediaPlayer и кнопку. Дайте кнопке имя BSelectFile и сопоставьте ей заголовок Select File (Выбор файла). Затем выберите компонент OpenDialog на вкладке Dialogs Палитры компонентов и поместите его на форму. Сохраните программу под именем, например, ACCESS.

Теперь создадим метод BSelectFileClick, который должен выглядеть примерно так:

```
procedure TForm1.BSelectFileClick(Sender: TObject);

begin
  MediaPlayer1.Close;
  if OpenDialog1.Execute then begin
```

```
      MediaPlayer1.FileName := OpenDialog1.FileName;
      MediaPlasyer1.Open;
    end;
end;
```

Приведенный выше код будет выполняться каждый раз при нажатии пользователем кнопки выбора файла (Select File). Принцип работы кода может быть показан в четырех шагах:

На первом шаге необходимо закрыть все MCI-устройства, которые могли бы быть к настоящему времени открытыми. Это мы делаем с помощью вызова метода `MediaPlayer1.Close`. При открытии программы эта строка не делает ничего полезного, но после того как вы открыли один файл и хотите открыть второй, вам понадобится «очистить» устройство. Вызов метода `Close` компонента `TMediaPlayer` в любом его состоянии никогда не приведет к ошибке.

На втором шаге выполняется `OpenDialog`. В диалоге выбора файла пользователи могут осуществлять поиск файла, как на жестком диске, так и на доступных CD-ROM. Найдя файл, достаточно нажать кнопку `OK`.

Третий шаг выполняется, только если пользователь выбрал файл. В случае если в диалоге `OpenDialog` нажать кнопку `Cancel`, ничего не произойдет. Быстрый взгляд на код дает понять, что выбранный пользователем файл очень просто назначить компоненту MCI. Как только будет определено подходящее имя файла, то для его открытия (или переоткрытия) MCI-устройством достаточно сделать последний шаг – осуществить вызов `MediaPlayer1.Open`. После этого пользователь уже может нажимать на зеленую стрелку для воспроизведения файла.

Теперь осталось добавить несколько небольших штрихов, чтобы программа стала более дружественной по отношению к пользователю. Например, можно предложить пользователю выбрать файл с определенным расширением, например, .AVI, .WAV или .MID. Есть два способа сделать это.

Первый следует выбрать, если требуется показать сразу все три типа файлов. Для этого заполните в Инспекторе объектов секцию Filter следующей строкой:

```
Мультимедиа файлы (*.avi;*wav;*.mid)|*.avi;*.wav;*.mid
```

Теперь пользователь может одновременно выбрать файлы WAVE, MIDI и AVI.

В качестве альтернативы можно присвоить свойству `OpenDialog.Filter` такую строку:

```
AVI файлы(*.avi)|*.avi|WAVE файлы(*.wav)|*.wav| MIDI файлы(*.MID)|*.mid
```

Для того чтобы строку было легче прочитать, ее можно разбить на три секции:

```
AVI файлы(*.avi)|*.avi
WAVE файлы(*.wav)|*.wav
MIDI файлы(*.MID)|*.mid
```

Каждая из этих секций содержит по одному символу вертикальной черты | (bar), отделяющему одну секцию от другой. Если при работающей программе активизируется OpenDialog, то первая половина строки (до разделяющего символа) будет отображена в секции List File Type (Список типов файлов) окна OpenDialog, а вторая половина – в секции FileName (Имя файла). В этом случае для выбора нужного типа файлов достаточно нажать кнопку справа от выпадающего списка и выбрать подходящее расширение.

Другое свойство OpenDialog, с помощью которого можно сделать жизнь пользователя комфортнее, это InitialDir. Данное свойство позволяет определить устройство или каталог, который будет показан при открытии диалога, направляя пользователя в каталог, где вероятность расположения файлов мультимедиа наиболее высока, а не заставлять искать их по всему компьютеру. Поскольку конкретный каталог не определен, пользователи могут выбрать его самостоятельно и выбирать в нем очередной мультимедийный файл для прослушивания.

Несколько описанных выше простых мелочей относились к работе пользователя с файлами и каталогами (при всем разнообразии методов нужно не забывать предоставлять по требованию полный доступ ко всем имеющимся на компьютере файлам). Другими приятными мелочами могли бы быть установка цвета фона формы в черный или запуск приложения в максимальном размере.

Возможности TMediaPlayer во время выполнения программы были показаны на примере очень простой программы медиа-доступа с именем ACCESS, каковую никому не составит труда скомпоновать. Программа за счет мультимедиа-компонента очень функциональна, так что она может удовлетворить потребности большинства пользователей в мультимедиа-возможностях. Тем не менее, в некоторых случаях требуется глубже знать механизм работы компонента и проигрывания файлов. В следующем разделе описывается программа с именем ACCESS2 и дается информация, которая позволит вам получить полный контроль над всеми возможностями MCI мультимедиа.

Перед началом повествования сделаем небольшое замечание.

Хотя большинство программистов сочтут эту информацию полезной, все же она адресуется в первую очередь мультимедиа-программистам. Некоторые программисты рвутся пойти дальше описанных в данном разделе простого управления компонентом и воспроизведения мультимедийных файлов. Такие энтузиасты должны понимать, что им придется перейти на более низкий уровень программирования, полностью игнорируя TMediaPlayer и работая с набором низкоуровневых команд Windows, которые можно найти в поставляемом с Delphi модуле MMSYSTEM.PAS. Рассказ о работе с этими низкоуровневыми командами выходит за рамки данной книги.

Ну вот, теперь вы готовы узнать о более тонких аспектах работы с TMediaPlayer.

Читатели, вероятно, обратили внимание, что палитра событий для MediaPlayer содержит группу специальных событий, обработкой которых можно заняться в своей программе. Эти методы подразделяются на две категории:

- Первая состоит из событий типа OnClick, которые происходят, когда пользователь нажимает какую-то область на элементе управления. Например, событие OnPlay с параметром Button посылается вашей программе при каждом нажатии на зеленую стрелку (проигрывании) компонента MediaPlayer.
- Вторая включает события типа OnNotify, содержащие сообщения mm_MciNotify. Такие сообщения посылаются вашему окну при начале или окончании проигрывания файла, или при возникновении ошибки.

Обе категории ошибок обсуждаются в следующих разделах.

Имеется восемь специализированных событий, возникающих при щелчке на элементе управления. Все эти события перехватываются и передаются в TMPBtnType:

- btPlay: нажатие зеленой стрелки Воспроизведение;
- btStop: нажатие красного квадратика на кнопке Стоп;
- btBack: нажатие голубой кнопки Назад;
- btStep: нажатие голубой кнопки Перемотка;
- btNext: нажатие голубой кнопки Следующая;
- btPrev: нажатие голубой кнопки Предыдущая;
- btPause: нажатие желтой кнопки Пауза.
- btRecord: нажатие красного кружка кнопки Запись.

Для того чтобы видеть, как возникают эти события, разместите на своей форме четыре поля редактирования (компоненты Edit). Каждый из этих элементов управления будет использован событием, но они не будут играть ключевой роли в программе, описанной ниже. Создайте для TMediaPlayer обработчик события OnClick и расположите в нем код, который должен выглядеть так:

```
procedure TForm1.MediaPlayer1Click(Sender: TObject; Button: TMPBtnType;
                                   var DoDefault: Boolean);
begin
  Edit1.Text := 'Воспроизведение';
end;
```

Ниже приведено описание метода, вызываемого всякий раз, когда пользователь нажимает кнопку медиа-проигрывателя:

```
procedure TForm1.MediaPlayer1Click(Sender: TObject; Button: TMPBtnType;
                                   var DoDefault: Boolean);
begin
  case Button of
      btPlay:    Edit1.Text := 'Воспроизведение';
```

```
      btPause:   Edit1.Text := 'Пауза';
      btStop:    Edit1.Text := 'Стоп';
      btNext:    Edit1.Text := 'Следующая';
      btPrev:    Edit1.Text := 'Предыдущая';
      btStep:    Edit1.Text := 'Перемотка';
      btBack:    Edit1.Text := 'Назад';
      btRecord:  Edit1.Text := 'Запись';
      btEject:   Edit1.Text := 'Извлечь';
  end;
end;
```

Чтобы обнаружить окончание воспроизведения или наличие ошибки, необходимо взглянуть на сообщение `OnNotify`. При работе непосредственно с операционной системой эти сообщения могут принимать четыре значения:

- `mci_Notify_Successful` – посылается при завершении работы команды;
- `mci_Notify_Superseded` – прерывание команды другой функцией;
- `mci_Notfiy_Aborted` – текущая функция прервана;
- `mci_Notify_Failure` – произошла ошибка.

Delphi не посылает эти сообщения непосредственно, а преобразует их в константы, например, `nvSuccessful`, `nvAborted` и т. д. Такой механизм был принят для обеспечения согласованности и взаимодействия различных частей библиотеки Delphi VCL.

Немного запутан вопрос о том, в какой именно момент посылается каждое сообщение. Как правило, следующие сообщения генерируются:

- `nvSuccessful`, когда команда успешно завершена;
- `nvSuperseded`, если система находилась в режиме временной остановки (паузы), а пользователь нажал кнопку воспроизведения;
- `nvAborted`, если пользователь нажимает на кнопку остановки или закрывает устройство.

Ниже приведена функция, реагирующая на сообщения `OnNotify` и отображающая короткую строку описания в поле редактирования:

```
procedure TForm1.MediaPlayer1Notify(Sender: TObject);
var
  S: String;
begin
  case MediaPlayer1.NotifyValue of
    nvSuccessful:  begin
                     Inc(Total);
                     S := 'mci_Notify_Successful ' + IntToStr(Total);
                   end;
    nvSuperseded:  S := 'mci_Notify_Superseded';
    nvAborted:     S := 'mci_Notify_Aborted';
    nvFailure:     S := 'mci_Notify_Failure';
  else
      S := 'Неизвестное уведомляющее сообщение';
  end;
```

```
    Edit2.Text := S;
    if (MediaPlayer1.NotifyValue = nvSuccessful) and
       (MediaPlayer1.Mode = mpStopped) then
          Edit1.Text := 'Окончание проигрывания файла';
end;
```

Данная функция автоматически вызывается каждый раз при возникновении значимого события, которое влияет на текущее состояние устройства MCI. Никогда не вызывайте явно эту функцию сами. Вместо этого ждите, пока ее не вызовет сама система.

Сердцем метода MediaPlayer1Notify является проверка текущего состояния поля NotifyValue у TMediaPlayer. Например, если поле NotifyValue содержит значение mci_Notify_Successful, то в поле редактирования будет помещена определенная строка. Снова повторимся: не надо предпринимать никаких шагов для изменения состояния поля NotifyValue. Оно изменяется системой автоматически. В обязанности программиста входит только ожидание события OnNotify и проверка поля NotifyValue.

Чтобы быть уверенным, что проигрывание файла завершено, дождитесь прихода сообщения mci_Notify_Successful, после чего убедитесь, что текущим является режим mci_Mode_Stop.

Текущий режим MCI-устройства определяется свойством Mode у TMediaPlayer. Вот список «общих» значений, которые могут определяться в этом поле:

```
mci_Mode_Not_Ready
mci_Mode_Stop
mci_Mode_Play
mci_Mode_Record
mci_Mode_Seek
mci_Mode_Pause
mci_Mode_Open
```

Имя каждого из этих значений указывает на соответствующий режим. Так, если поле Mode содержит mci_Mode_Stop, устройство остановлено. Если свойство содержит mci_Mode_Play, устройство в настоящий момент находится в режиме воспроизведения. Delphi не посылает эти константы напрямую, а задействует свои константы, такие как mpStopped, mpPlaying и т. д.

Если приложение получило сообщение mci_Notify_Successful, а поле Mode было установлено в mci_Mode_Stop, то можно быть уверенным, что воспроизведение текущего файла завершено. Ниже приведен метод OnNotify, проверяющий эти условия:

```
if (MediaPlayer1.NotifyValue = nvSuccessful) and
   (MediaPlayer1.Mode = mpStopped) then
      Edit1.Text := 'Воспроизведение файла окончено';
```

Программа ACCESS2 содержит также приведенный ниже метод, проверяющий текущее состояние медиа-проигрывателя:

```
procedure TForm1.SetMode;
begin
  case MediaPlayer1.Mode of
     mpNotReady:    Edit3.Text := 'mci_Mode_Not_Ready';
      mpStopped:     Edit3.Text := 'mci_Mode_Stop';
      mpPlaying:     Edit3.Text := 'mci_Mode_Play';
      mpRecording:   Edit3.Text := 'mci_Mode_Record';
      mpSeeking:     Edit3.Text := 'mci_Mode_Seek';
      mpPaused:      Edit3.Text := 'mci_Mode_Pause';
      mpOpen:        Edit3.Text := 'mci_Mode_Open';
  else
    begin
      Edit1.Text := 'Устройство неактивно'
      Edit2.Text := 'Нет специальных сообщений';
      Edit3.Text := 'Неизвестно';
      Edit4.Text := 'Файл не выбран';
    end;
  end;
end;
```

Эта программа вызывается в ответ на срабатывание таймера (TTimer), расположенного на форме. Чтобы создать таймер, просто выберите его на вкладке System Палитры компонентов, расположите на форме и создайте обработчик события OnTimer. Метод не должен делать ничего, кроме как вызывать процедуру SetMode, рассмотренную выше. Свойство Interval компонента TTimer определяет частоту генерирования событий OnTimer. Если оставить значение Interval равным 1000 миллисекунд, как установлено по умолчанию, то SetMode будет вызываться каждую секунду, тем самым гарантируя, что пользователь будет в курсе текущего состояния TMediaPlayer.

При работе с программой ACCESS2 обратите внимание на следующие ключевые моменты в коде:

- При каждом нажатии пользователем какой-либо кнопки компонента MediaPlayer автоматически генерируется событие. Например, если пользователь нажимает зеленую стрелку воспроизведения, программе посылается событие btPlay. Для того чтобы реагировать на это событие, создайте метод OnClick, просто дважды щелкнув в соответствующей строке Инспектора объектов.
- Сообщения mci_Notify_mm посылаются программе в момент возникновения ключевых событий в цикле «жизни» мультимедиа-файла. Например, при завершении проигрывания файла приложению посылается сообщение mci_Notify_Successful. Можно даже реагировать на это событие в обработчике OnNotify.
- Можно послать запрос системе на предмет текущего состояния выбранного мультимедиа-устройства. Чтобы сделать это, следует «ознакомиться» со значением свойства Mode у MediaPlayer. Например, если MediaPlayer1.Mode = mpPlaying, то вы можете быть уверены, что устройство в данный момент находится в режиме воспроизведения.

Элемент управления TMediaPlayer полностью освобождает программиста от необходимости работы с интерфейсом Microsoft MCI, описанным в модуле MMSYSTEM.PAS. Но тот, кто хочет стать экспертом в затрагиваемой области, должен изучить как этот модуль, так и способы его вызова из MPLAYER.PAS.

Заключение

Изучив данный материал, вы узнали, как работает элемент управления TMediaPlayer, и как можно заставить его проигрывать файлы WAVE, MIDI и AVI. Delphi предоставляет простой и одновременно эффективный способ создания собственных мультимедийных приложений. Используйте данный материал как отправную точку, а свое воображение – как руководство к действию.

Не забывайте, что Windows предоставляет полный набор возможностей Multimedia. Многие из них доступны через компонент TMediaPlayer. Тем не менее, Delphi не только не ограничит тех, кто захочет узнать об этой стороне операционной системы немного больше, а еще и предоставит в их распоряжение отличный инструмент для работы с низкоуровневыми командами.

Извлечение звука из динамика в Windows 9x

Решение

```
function GetPort(address: word): word;
var
  bValue: byte;
begin
  asm
    mov  dx, address
    in   al, dx
    mov  bValue, al
  end;
  GetPort := bValue;
end;

procedure SetPort(address, Value: word);
var
  bValue: byte;
begin
  bValue := trunc(Value and 255);
  asm
    mov  dx, address
    mov  al, bValue
    out  dx, al
  end;
end;

procedure Sound(Freq: word);
var
  B: byte;
```

```
begin
  if Freq > 18 then begin
    Freq := Word(1193181 div LongInt(Freq));
    B := Byte(GetPort($61));
    if (B and 3) = 0 then begin
      SetPort($61, Word(B or 3));
      SetPort($43, $B6);
    end;
    SetPort($42, Freq);
    SetPort($42, Freq shr 8);
  end;
end;

procedure NoSound;
var
  Value: word;
begin
  Value := GetPort($61) and $FC;
  SetPort($61, Value);
end;
```

[News Group]

Формат WAV-файла

Структура заголовка WAV-файла:

```
TWaveHeader = record
  Marker1: array[0..3] of Char;
  BytesFollowing: LongInt;
  Marker2: array[0..3] of Char;
  Marker3: array[0..3] of Char;
  Fixed1: LongInt;
  FormatTag: Word;
  Channels: Word;
  SampleRate: LongInt;
  BytesPerSecond: LongInt;
  BytesPerSample: Word;
  BitsPerSample: Word;
  Marker4: array[0..3] of Char;
  DataBytes: LongInt;
end;
```

Для создания собственного WAV-файла сделайте следующее:

```
DataBytes := Channels;
DataBytes := DataBytes * SampleRate;
DataBytes := DataBytes * Resolution;
DataBytes := DataBytes div 8;
DataBytes := DataBytes * Duration;
DataBytes := DataBytes div 1000;
```

```
WaveHeader.Marker1 := 'RIFF';
WaveHeader.BytesFollowing := DataBytes + 36;
WaveHeader.Marker2 := 'WAVE';
WaveHeader.Marker3 := 'fmt ';
WaveHeader.Fixed1 := 16;
WaveHeader.FormatTag := 1;
WaveHeader.SampleRate := SampleRate;
WaveHeader.Channels := Channels;
WaveHeader.BytesPerSecond := Channels;
WaveHeader.BytesPerSecond := WaveHeader.BytesPerSecond * SampleRate;
WaveHeader.BytesPerSecond := WaveHeader.BytesPerSecond * Resolution;
WaveHeader.BytesPerSecond := WaveHeader.BytesPerSecond div 8;
WaveHeader.BytesPerSample := Channels * Resolution div 8;
WaveHeader.BitsPerSample := Resolution;
WaveHeader.Marker4 := 'data';
WaveHeader.DataBytes := DataBytes;
```

Остальная часть файла представляет собой звуковые данные. Порядок следования: верхний уровень для левого канала, верхний уровень для правого канала и т. д. Для монофонических или 8-битных файлов сделайте соответствующие изменения.

Создание пустого WAV-файла

Как создать пустой WAV-*файл?*

MediaPlayer может открыть звуковой файл, если он содержит по крайней мере один байт данных. Если с помощью данного компонента пытаться создать и открыть звуковой файл, содержащий только заголовок, MediaPlayer не захочет этого делать.

Нижеприведенный код создаст звуковой файл размером 1 байт. Конечно, это не очень красиво, но это работает. Необходимо лишь добавить MMSYSTEM ко всем модулям, использующим данную функцию.

```
function TForm1.CreateNewWave(NewFileName: String): Boolean;
var
  DeviceID: Word;
  Return: LongInt;
  MciOpen: TMCI_Open_Parms;
  MciRecord: TMCI_Record_Parms;
  MciPlay: TMCI_Play_Parms;
  MciSave: TMCI_SaveParms;
  MCIResult: LongInt;
  Flags: Word;
  TempFileName: array[0..255] of char;
begin
  MediaPlayer1.Close;
  StrPCopy(TempFileName, NewFileName);
  MciOpen.lpstrDeviceType := 'waveaudio';
  MciOpen.lpstrElementName := '';
```

```
    Flags := Mci_Open_Element or Mci_Open_Type;
    MCIResult := MciSendCommand(0, MCI_OPEN, Flags, LongInt(@MciOpen));
    DeviceID := MciOpen.wDeviceId;
    MciRecord.dwTo := 1;
    Flags := Mci_To or Mci_Wait;
    MCIResult := MciSendCommand(DeviceID, Mci_Record, Flags,
                                LongInt(@MciRecord));
    mciPlay.dwFrom := 0;
    Flags := Mci_From or Mci_Wait;
    MciSendCommand(DeviceId, Mci_Play, Flags, LongInt(@MciPlay));
    mciSave.lpfileName := TempFilename;
    Flags := MCI_Save_File or Mci_Wait;
    MCIResult := MciSendCommand(DeviceID, MCI_Save, Flags,
                                LongInt(@MciSave));
    Result := MciSendCommand(DeviceID, Mci_Close, 0,
                             LongInt(nil)) = 0;
end;
```

Проигрывание WAVE-файла, помещенного в ресурс

Я пытаюсь проиграть WAVE-*файл при щелчке на кнопке моего Delphi приложения. Я установил звуковой файл и воспользовался вызовом API-функции* PlaySound(), *но мне хотелось бы поместить его в ресурс приложения, т. е. «вложить» его в* EXE-*файл и проигрывать его оттуда.*

Надо скомпилировать необходимый ресурс (например, с помощью Resource Workshop) и включить туда WAVE-файл. Затем для его вызова и проигрывания используйте следующий код:

```
  ...
var
  FindHandle, ResHandle: THandle;
  ResPtr: Pointer;
begin
  FindHandle := FindResource(HInstance, '<Имя вашего ресурса>', 'WAVE');
  if FindHandle <> 0 then begin
    ResHandle := LoadResource(HInstance, FindHandle);
    if ResHandle <> 0 then begin
      ResPtr := LockResource(ResHandle);
      if ResPtr <> Nil then SndPlaySound(PChar(ResPtr), snd_ASync or snd_Memory);
      UnlockResource(ResHandle);
    end;
    FreeResource(FindHandle);
  end;
end;
```

Но есть и лучшее решение:

```
PlaySound('S1', HInstance, SND_RESOURCE or SND_ASYNC);
```

где S1 — ID (идентификатор) звука.

Эта единственная строчка кода сама ищет, загружает, блокирует, разблокирует и освобождает ресурс.

«Декомпиляция» файла формата WAV и получение данных

Как из «звукового» файла получить данные?

Программа для Delphi 1 и Delphi 2. Читает WAV-файлы и производит выборку исходных данных. Но она не может их восстановить, используя зашитый алгоритм сжатия.

```
unit LinearSystem;

interface

const                    { Поток данных сэмпла }
  MaxN = 300;            { максимальное значение величины сэмпла }
type                     { Тип, описывающий формат WAV }
  WAVHeader = record
    nChannels: Word;
    nBitsPerSample: LongInt;
    nSamplesPerSec: LongInt;
    nAvgBytesPerSec: LongInt;
    RIFFSize: LongInt;
    fmtSize: LongInt;
    formatTag: Word;
    nBlockAlign: LongInt;
    DataSize: LongInt;
  end;
  SampleIndex = 0 .. MaxN + 3;
  DataStream = array[SampleIndex] of Real;
  Observation = record  { Переменные сопровождения }
    Name: String[40];   { Имя данного сопровождения }
    yyy: DataStream;    { Массив указателей на данные }
    WAV: WAVHeader;     { Спецификация WAV для сопровождения }
    Last: SampleIndex;  { Последний доступный индекс yyy }
    Min0, Max0: Real;   { Диапазон значений yyy }
  end;

var
  N: SampleIndex;
  K0R, K1R, K2R, K3R: Observation;
  K0B, K1B, K2B, K3B: Observation;

{ Переменные имени файла }
  StandardDatabase: String[80];
  BaseFileName: String[80];
  StandardOutput: String[80];
  StandardInput: String[80];
```

```
{ Объявления процедур }
procedure ReadWAVFile (var Ki, Kj: Observation);
procedure WriteWAVFile (var Ki, Kj: Observation);
procedure ScaleData (var Kk: Observation);
procedure InitAllSignals;
procedure InitLinearSystem;

implementation

{$R *.DFM}

uses
  SysUtils;

const                                        { Стандартный формат WAV-файла }
  MaxDataSize: LongInt = (MaxN + 1) * 2 * 2;
  MaxRIFFSiz : LongInt = (MaxN + 1) * 2 * 2 + 36;
  StandardWAV: WAVHeader = (
      nChannels: Word(2);
      nBitsPerSample: LongInt(16);
      nSamplesPerSec: LongInt(8000);
      nAvgBytesPerSec: LongInt(32000);
      RIFFSize: LongInt((MaxN + 1) * 2 * 2 + 36);
      fmtSize: LongInt(16);
      formatTag: Word(1);
      nBlockAlign: LongInt(4);
      DataSize: LongInt((MaxN + 1) * 2 * 2));

procedure ScaleData(var Kk: Observation); // Сканирование переменных сопровождения
var
  I: SampleIndex;
begin
{ Инициализация переменных сканирования }
  Kk.Max0 := Kk.yyy[0];
  Kk.Min0 := Kk.yyy[0];

{ Сканирование для получения максимального и минимального значения }
  for I := 1 to Kk.Last do begin
    if Kk.Max0 < Kk.yyy[I] then Kk.Max0 := Kk.yyy[I];
    if Kk.Min0 > Kk.yyy[I] then Kk.Min0 := Kk.yyy[I];
  end;
end; { ScaleData }

procedure ScaleAllData;
begin
  ScaleData(K0R);
  ScaleData(K0B);
  ScaleData(K1R);
  ScaleData(K1B);
  ScaleData(K2R);
  ScaleData(K2B);
```

```
  ScaleData(K3R);
  ScaleData(K3B);
end; {ScaleAllData}

var                                          { Считывание/запись WAV-данных }
  InFile, OutFile: file of Byte;
type
  Tag = (F0, T1, M1);
  FudgeNum = record
    case X:Tag of
      F0: (chrs: array[0..3] of Byte);
      T1: (lint: LongInt);
      M1: (up, dn: Integer);
    end;

var
  ChunkSize: FudgeNum;
procedure WriteChunkName(Name: String);
var
  i: Integer;
  MM: Byte;
begin
  for i := 1 to 4 do begin
    MM := ord(Name[i]);
    write(OutFile, MM);
  end;
end; { WriteChunkName }

procedure WriteChunkSize(LL: Longint);
var
  I: integer;
begin
  ChunkSize.x := T1;
  ChunkSize.lint := LL;
  ChunkSize.x := F0;
  for I := 0 to 3 do write(OutFile, ChunkSize.chrs[I]);
end;

procedure WriteChunkWord(WW:Word);
var
  I: integer;
begin
  ChunkSize.x := T1;
  ChunkSize.up := WW;
  ChunkSize.x := M1;
  for I := 0 to 1 do write(OutFile, ChunkSize.chrs[I]);
end; { WriteChunkWord }

procedure WriteOneDataBlock(var Ki, Kj: Observation);
var
  I: Integer;
```

```
begin
  ChunkSize.x := M1;
  with Ki.WAV do begin
    case nChannels of
      1:  if nBitsPerSample = 16 then begin
      { 1..2 Помещаем в буфер одноканальный 16-битный сэмпл }
            ChunkSize.up := trunc(Ki.yyy[N] + 0.5);
            if N < MaxN then ChunkSize.dn := trunc(Ki.yyy[N + 1] + 0.5);
            N := N + 2;
          end else begin
      { 1..4 Помещаем в буфер одноканальный 8-битный сэмпл }
            for I:=0 to 3 do ChunkSize.chrs[I] := trunc(Ki.yyy[N + I] + 0.5);
            N := N + 4;
          end;
      2:  if nBitsPerSample = 16 then begin
      { 2 Двухканальный 16-битный сэмпл }
           ChunkSize.dn := trunc(Ki.yyy[N] + 0.5);
           ChunkSize.up := trunc(Kj.yyy[N] + 0.5);
           N := N + 1;
         end else begin
      { 4 Двухканальный 8-битный сэмпл }
           ChunkSize.chrs[1] := trunc(Ki.yyy[N] + 0.5);
           ChunkSize.chrs[3] := trunc(Ki.yyy[N + 1] + 0.5);
           ChunkSize.chrs[0] := trunc(Kj.yyy[N] + 0.5);
           ChunkSize.chrs[2] := trunc(Kj.yyy[N + 1] + 0.5);
           N := N + 2;
        end;

    end;
  end;                  // четырехбайтовая переменная "ChunkSize" теперь заполнена
  ChunkSize.x := T1;
  WriteChunkSize(ChunkSize.lint);          // помещаем 4 байта данных
end; { WriteOneDataBlock }

procedure WriteWAVFile(var Ki, Kj: Observation);
var
  MM: Byte;
  I: Integer;
  OK: Boolean;
begin
{ Приготовления для записи файла данных }
  AssignFile(OutFile, StandardOutput);     // Файл, выбранный в диалоговом окне
  ReWrite(OutFile);
  with Ki.WAV do begin
    DataSize := nChannels * (nBitsPerSample div 8) * (Ki.Last + 1);
    RIFFSize := DataSize + 36;
    fmtSize := 16;
  end;

{ Записываем ChunkName "RIFF" }
  WriteChunkName('RIFF');
```

```
{ Записываем ChunkSize }
  WriteChunkSize(Ki.WAV.RIFFSize);

{ Записываем ChunkName "WAVE" }
  WriteChunkName('WAVE');
{ Записываем tag "fmt_" }
  WriteChunkName('fmt ');

{ Записываем ChunkSize }
  Ki.WAV.fmtSize := 16;      { должно быть 16 - 18 }
  WriteChunkSize(Ki.WAV.fmtSize);

{ Записываем formatTag, nChannels }
  WriteChunkWord(Ki.WAV.formatTag);
  WriteChunkWord(Ki.WAV.nChannels);
{ Записываем nSamplesPerSec }
  WriteChunkSize(Ki.WAV.nSamplesPerSec);

{ Записываем nAvgBytesPerSec }
  WriteChunkSize(Ki.WAV.nAvgBytesPerSec);

{ Записываем nBlockAlign, nBitsPerSample }
  WriteChunkWord(Ki.WAV.nBlockAlign);
  WriteChunkWord(Ki.WAV.nBitsPerSample);

{ Записываем метку блока данных "data" }
  WriteChunkName('data');

{ Записываем DataSize }
  WriteChunkSize(Ki.WAV.DataSize);
  N := 0;              { первая запись-позиция }

{ помещаем 4 байта и увеличиваем счетчик N }
  while N <= Ki.Last do WriteOneDataBlock(Ki, Kj);

{Освобождаем буфер файла}
  CloseFile( OutFile );
end; {WriteWAVFile}

procedure InitSpecs;
begin
end; { InitSpecs }

procedure InitSignals(var Kk: Observation);
var
  J: Integer;
begin
  for J := 0 to MaxN do Kk.yyy[J] := 0.0;
  Kk.Min0 := 0.0; Kk.Max0 := 0.0;
  Kk.Last := MaxN;
end; { InitSignals }
```

```
procedure InitAllSignals;
begin
  InitSignals(K0R);
  InitSignals(K0B);
  InitSignals(K1R);
  InitSignals(K1B);
  InitSignals(K2R);
  InitSignals(K2B);
  InitSignals(K3R);
  InitSignals(K3B);
end; {InitAllSignals}
var ChunkName: string[4];
procedure ReadChunkName;
var
  I: integer;
  MM: Byte;
begin
  ChunkName[0] := chr(4);
  for I := 1 to 4 do begin
    Read(InFile, MM);
    ChunkName[I] := chr(MM);
  end;
end; { ReadChunkName }

procedure ReadChunkSize;
var
  I: integer;
  MM: Byte;
begin
  ChunkSize.x := F0;
  ChunkSize.lint := 0;
  for I := 0 to 3 do begin
    Read(InFile, MM);
    ChunkSize.chrs[I] := MM;
  end;
  ChunkSize.x := T1;
end; { ReadChunkSize }

procedure ReadOneDataBlock(var Ki, Kj: Observation);
var
  I: Integer;
begin
  if N <= MaxN then begin
    ReadChunkSize;                          { получаем 4 байта данных }
    ChunkSize.x := M1;
    with Ki.WAV do
      case nChannels of
        1:  if nBitsPerSample = 16 then begin
        { 1..2 Помещаем в буфер одноканальный 16-битный сэмпл }
              Ki.yyy[N] := 1.0 * ChunkSize.up;
              if N < MaxN then Ki.yyy[N + 1] := 1.0 * ChunkSize.dn;
```

```
            N := N + 2;
          end else begin
        { 1..4 Помещаем в буфер одноканальный 8-битный сэмпл }
            for I := 0 to 3 do
              Ki.yyy[N + I] := 1.0 * ChunkSize.chrs[I];
            N := N + 4;
          end;
        2:  if nBitsPerSample = 16 then begin
        { 2 Двухканальный 16-битный сэмпл }
            Ki.yyy[N] := 1.0 * ChunkSize.dn;
            Kj.yyy[N] := 1.0 * ChunkSize.up;
            N := N + 1;
          end else begin
        { 4 Двухканальный 8-битный сэмпл }
            Ki.yyy[N] := 1.0 * ChunkSize.chrs[1];
            Ki.yyy[N + 1] := 1.0 * ChunkSize.chrs[3];
            Kj.yyy[N] := 1.0 * ChunkSize.chrs[0];
            Kj.yyy[N + 1] := 1.0 * ChunkSize.chrs[2];
            N := N + 2;
          end;
  end;
  if N <= MaxN then begin    { LastN := N; }
    Ki.Last := N;
    if Ki.WAV.nChannels = 2 then Kj.Last := N;
    end else begin           { LastN := MaxN; }
      Ki.Last := MaxN;
      if Ki.WAV.nChannels = 2 then Kj.Last := MaxN;
    end;
  end;
end; { ReadOneDataBlock }

procedure ReadWAVFile(var Ki, Kj :Observation);
var
  MM: Byte;
  I: Integer;
  OK: Boolean;
  NoDataYet: Boolean;
  DataYet: Boolean;
  nDataBytes: LongInt;
begin
  if FileExists(StandardInput) then
    with Ki.WAV do begin                  { Вызов диалога открытия файла }
      OK := True;                         { если не изменится где-нибудь ниже }
  { Приготовления для чтения файла данных, выбранного в диалоговом окне }
      AssignFile(InFile, StandardInput);
      Reset(InFile);
  { Считываем ChunkName "RIFF" }
      ReadChunkName;
      if ChunkName <> 'RIFF' then OK := False;
  { Считываем ChunkSize }
      ReadChunkSize;
```

```
    RIFFSize := ChunkSize.lint;          { должно быть 18,678 }
{ Считываем ChunkName "WAVE" }
    ReadChunkName;
    if ChunkName <> 'WAVE' then OK := False;
{ Считываем ChunkName "fmt_" }
    ReadChunkName;
    if ChunkName <> 'fmt ' then OK := False;
{ Считываем ChunkSize }
    ReadChunkSize;
    fmtSize := ChunkSize.lint;           { должно быть 18 }
{ Считываем formatTag, nChannels }
    ReadChunkSize;
    ChunkSize.x := M1;
    formatTag := ChunkSize.up;
    nChannels := ChunkSize.dn;
{ Считываем nSamplesPerSec }
    ReadChunkSize;
    nSamplesPerSec := ChunkSize.lint;
{ Считываем nAvgBytesPerSec }
    ReadChunkSize;
    nAvgBytesPerSec := ChunkSize.lint;
{ Считываем nBlockAlign }
    ChunkSize.x := F0;
    ChunkSize.lint := 0;
    for I := 0 to 3 do begin Read(InFile, MM);
    ChunkSize.chrs[I] := MM;
  end;
  ChunkSize.x := M1;
  nBlockAlign := ChunkSize.up;
{ Считываем nBitsPerSample }
  nBitsPerSample := ChunkSize.dn;
  for I := 17 to fmtSize do Read(InFile, MM);
  NoDataYet := True;
  while NoDataYet do begin
{ Считываем метку блока данных "data" }
    ReadChunkName;
{ Считываем DataSize }
    ReadChunkSize;
    DataSize := ChunkSize.lint;
    if ChunkName <> 'data' then begin
{ пропуск данных, не относящихся к набору звуковых данных }
      for I := 1 to DataSize do
        Read(InFile, MM);
    end else
      NoDataYet := False;
  end;
  nDataBytes := DataSize;
{ Наконец, начинаем считывать данные для байтов nDataBytes }
  if nDataBytes > 0 then DataYet := True;
  N := 0;                                { чтение с первой позиции }
  while DataYet do begin
```

```
      ReadOneDataBlock(Ki, Kj);            { получаем 4 байта }
      nDataBytes := nDataBytes - 4;
      if nDataBytes <= 4 then DataYet := False;
    end;
    ScaleData(Ki);
    if Ki.WAV.nChannels = 2 then begin
      Kj.WAV := Ki.WAV;
      ScaleData(Kj);
    end;
  { Освобождаем буфер файла }
    CloseFile(InFile);
  end else begin
    InitSpecs;                           { файл не существует }
    InitSignals(Ki);                     { обнуляем массив "Ki" }
    InitSignals(Kj);                     { обнуляем массив "Kj" }
  end;
end; { ReadWAVFile }
{ Операции с набором данных }
const
  MaxNumberOfDataBaseItems = 360;

type
  SignalDirectoryIndex = 0 .. MaxNumberOfDataBaseItems;

var
  DataBaseFile: file of Observation;
  LastDataBaseItem: LongInt;             // Номер текущего элемента набора данных
  ItemNameS: array[SignalDirectoryIndex] of String[40];

procedure GetDatabaseItem(Kk: Observation; N: LongInt);
begin
  if N <= LastDataBaseItem then begin
    Seek(DataBaseFile, N);
    Read(DataBaseFile, Kk);
  end else
    InitSignals(Kk);
end; {GetDatabaseItem}

procedure PutDatabaseItem(Kk: Observation; N: LongInt);
begin
  if N < MaxNumberOfDataBaseItems then
    if N <= LastDataBaseItem then begin
      Seek(DataBaseFile, N);
      Write(DataBaseFile, Kk);
      LastDataBaseItem := LastDataBaseItem + 1;
    end else
      while LastDataBaseItem <= N do begin
        Seek(DataBaseFile, LastDataBaseItem);
        Write(DataBaseFile, Kk);
        LastDataBaseItem := LastDataBaseItem + 1;
      end else
```

```
//        ReportError(1);              // Попытка чтения MaxNumberOfDataBaseItems
end; {PutDatabaseItem}

procedure InitDataBase;
begin
  LastDataBaseItem := 0;
  if FileExists(StandardDataBase) then begin
    Assign(DataBaseFile, StandardDataBase);
    Reset(DataBaseFile);
    while not EOF(DataBaseFile) do begin
      GetDataBaseItem(KOR, LastDataBaseItem);
      ItemNameS[LastDataBaseItem] := KOR.Name;
      LastDataBaseItem := LastDataBaseItem + 1;
    end;
    if EOF(DataBaseFile) then
      if LastDataBaseItem > 0 then
        LastDataBaseItem := LastDataBaseItem - 1;
    end;
end; {InitDataBase}

function FindDataBaseName(Nstg: String):LongInt;
var
  ThisOne: LongInt;
begin
  ThisOne := 0;
  FindDataBaseName := -1;
  while ThisOne < LastDataBaseItem do begin
    if Nstg = ItemNameS[ThisOne] then begin
      FindDataBaseName := ThisOne;
      Exit;
    end;
    ThisOne := ThisOne + 1;
  end;
end; { FindDataBaseName }

{ Инициализация модуля }
procedure InitLinearSystem;
begin
  BaseFileName:= 'c:\temp\';
  StandardOutput := BaseFileName + 'KO.wav';
  StandardInput := BaseFileName + 'KO.wav';
  StandardDataBase := BaseFileName + 'Radar.sdb';
  InitAllSignals;
  InitDataBase;
  ReadWAVFile(KOR, KOB);
  ScaleAllData;
end; {InitLinearSystem}
begin                        { инициализируемый модулем код }
  InitLinearSystem;
end.                         { Unit LinearSystem }
end.
```

Удаление содержимого WAV-файла

Как очистить содержимое WAV-файла? Просто заново создать пустой?

Пример компонента, позволяющего стирать любую часть WAV-файла:

```
unit Nickmp;

interface
uses
  Windows, SysUtils, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  MPlayer, MMSystem;
type
  TNickMediaPlayer = class(TMediaPlayer)
  public
    function DeleteWaveChunk(aFrom, aTo: LongInt): Longint;
end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('Samples', [TNickMediaPlayer]);
end;

function TNickMediaPlayer.DeleteWaveChunk(aFrom, aTo: LongInt): Longint;
var
  DeleteParms: TMCI_WAVE_DELETE_PARMS;
  Flags: LongInt;
begin
  Flags := 0;
  if Wait then Flags := mci_Wait;
  if Notify then Flags := Flags or mci_Notify;
  DeleteParms.dwCallback := Handle;
  Flags := Flags or mci_From;
  DeleteParms.dwFrom := aFrom;
  Flags := Flags or mci_To;
  DeleteParms.dwTo := aTo;
  Result := mciSendCommand(DeviceID, mci_Delete, Flags, Longint(@DeleteParms));
end;

end.
```

Получение идентификатора диска

Как получить идентификатор находящегося в CD-ROM звукового компакт-диска?

```
// метка диска
procedure GetDriveInfo(VolumeName: string; var VolumeLabel, SerialNumber,
                       FileSystem: string);
```

```
var
  VolLabel, FileSysName: array [0..255] of char;
  SerNum: pdword;
  MaxCompLen, FileSysFlags: dword;
begin
  New(SerNum);
  GetVolumeInformation(PChar(VolumeName), VolLabel, 255, SerNum, MaxCompLen,
                       FileSysFlags, FileSysName, 255);
  VolumeLabel := VolLabel;
  SerialNumber := Format('%x', [SerNum^]);
  FileSystem := FileSysName;
  Dispose(SerNum);
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  VolLabel, SN, FileSystem, S: string;
begin
  S := 'g:\';      // имя CD-дисковода
  GetDriveInfo(S, VolLabel, SN, FileSystem);
  Label1.Caption := VolLabel + ' ' + SN + ' ' + FileSystem;
end;
```

получаем:

```
VolLabel   - 'ARMSTRONG'  // метка диска
SN         - B5FF77AD     // номер серийный
FileSystem - CDFS         // тип файловой системы
```

А этот код определяет тип всех дисков:

```
procedure TForm1.GetAllDrive(Sender: TObject);
var
  i, mask: integer;
  s: string;

begin
  mask := GetLogicalDrives;
  i := 0;
  while mask <> 0 do begin
    s := chr(ord('a') + i) + ':\';

    if (mask and 1) <> 0 then
      case GetDriveType(PChar(s)) of
                0:  ListBox1.Items.Add(s + ' unknown.');
                1:  ListBox1.Items.Add(s + ' not exist.');
  DRIVE_REMOVABLE:  ListBox1.Items.Add(s + ' removable.');  // floppy, zip
     DRIVE_FIXED:  ListBox1.Items.Add(s + ' fixed.');
    DRIVE_REMOTE:  ListBox1.Items.Add(s + ' network.');
     DRIVE_CDROM:  ListBox1.Items.Add(s + ' CD-ROM.');
   DRIVE_RAMDISK:  ListBox1.Items.Add(s + ' RAM.');
      end;
```

```
      inc(i);
      mask := mask shr 1;
    end;
  end;
```

В ListBox1 получаем все диски на данном компьютере.

[Pincuk Vasili]

Определение типа CD

Как определить, является ли компакт-диск звуковым?

Далее приведен пример с использованием MediaPlayer.

```
function IsAudioCD(Drive: char): bool;
var
  DrivePath: string;
  MaximumComponentLength: DWORD;
  FileSystemFlags: DWORD;
  VolumeName: string;
begin
  Result := false;
  DrivePath := Drive + ':\';
  if GetDriveType(PChar(DrivePath)) <> DRIVE_CDROM then exit;
  SetLength(VolumeName, 64);
  GetVolumeInformation(PChar(DrivePath), PChar(VolumeName), Length(VolumeName),
                       nil, MaximumComponentLength, FileSystemFlags, nil, 0);
  if lStrCmp(PChar(VolumeName), 'Audio CD') = 0 then Result := true;
end;

function PlayAudioCD(Drive: char): bool;
var
  mp: TMediaPlayer;
begin
  Result := false;
  Application.ProcessMessages;
  if not IsAudioCD(Drive) then exit;
  mp := TMediaPlayer.Create(nil);
  mp.Visible := false;
  mp.Parent := Application.MainForm;
  mp.Shareable := true;
  mp.DeviceType := dtCDAudio;
  mp.FileName := Drive + ':';
  mp.Shareable := true;
  mp.Open;
  Application.ProcessMessages;
  mp.Play;
  Application.ProcessMessages;
  mp.Close;
  Application.ProcessMessages;
  mp.Free;
```

```
  Result := true;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  if not PlayAudioCD('G') then ShowMessage('Not an Audio CD');
end;
```

Серийный номер AudioCD

Как получить серийный номер звукового компакт-диска?

Решение

```
uses
  ...,
  MMSystem,
  MPlayer;

procedure TForm1.Button1Click(Sender: TObject);

var
  mp: TMediaPlayer; msp: TMCI_INFO_PARMS;
  MediaString: array[0..255] of char; ret: longint;

begin
  mp := TMediaPlayer.Create(nil);
  mp.Visible := false;
  mp.Parent := Application.MainForm;
  mp.Shareable := true;
  mp.DeviceType := dtCDAudio;
  mp.FileName := 'H:';          // CD-ROM
  mp.Open;
  Application.ProcessMessages;
  FillChar(MediaString, SizeOf(MediaString), #0);
  FillChar(msp, SizeOf(msp), #0);
  msp.lpstrReturn := @MediaString;
  msp.dwRetSize := 255;
  ret := mciSendCommand(Mp.DeviceId, MCI_INFO, MCI_INFO_MEDIA_IDENTITY,
                        longint(@msp));
  if Ret <> 0 then begin
    MciGetErrorString(ret, @MediaString, SizeOf(MediaString));
    Memo1.Lines.Add(StrPas(MediaString));
  end else
    Memo1.Lines.Add(StrPas(MediaString));
  mp.Close;
  Application.ProcessMessages;
  mp.Free;
end;
```

Контроль джойстика в Delphi

Есть ли возможность в Delphi контролировать джойстик?

Действительно, Delphi это позволяет. Нижеприведенный код был взят из действующего приложения, его можно переписать под свои нужды. Главное – он показывает технологию работы с джойстиком.

```
uses
  ..., MMSystem;

var
  myjoy: YJoyInfo;

begin
  JoyGetPos(Joystickid1,@myjoy);
  TrackBar1.Position := myjoy.wYpos;
  TrackBar2.Position := myjoy.wXpos;
  RadioButton1.Checked := (myjoy.wButtons and Joy_Button1) > 0;
  RadioButton2.Checked := (myjoy.wButtons and Joy_Button2) > 0;
end;
```

Примечание

Для получения дополнительной информации о работе с джойстиком в Delphi необходимо изучить модуль `MMSystem` *(раздел* `Joystick support`*).*

7

Аппаратное обеспечение

Дата BIOS из приложения

Определить дату BIOS, равно как и другие данные, находящиеся по фиксированному физическому адресу, в Delphi можно так:

```
var
  BiosDate: array [0..7] of char absolute $FFFF5;
  PCType: byte absolute $FFFFE;

procedure TForm1.FormCreate(Sender: TObject);
var
  S: string;
begin
  case PCType of
      $FC:  S := 'AT';
      $FD:  S := 'PCjr';
      $FE:  S := 'XT =8-O';
      $FF:  S := 'PC';
  else S := 'Нестандартный';
  end;
  Caption := 'Дата BIOS: ' + BiosDate + ' Тип ПК: ' + S;
end;
```

Получение списка процессов

Решение

```
procedure TForm1.Button1Click(Sender: TObject);
var
  handler: THandle;
```

```
  data: TProcessEntry32;

  function return_name: string;
  var
    i: byte;
    names: string;
  begin
    names := '';
    i := 0;
    while data.szExeFile[i] <> '' do begin
      names := names + data.szExeFile[i];
      inc(i);
    end;
    return_name := names;
  end;

begin
  handler := CreateToolhelp32Snapshot(TH32CS_SNAPALL, 0);
  if Process32First(handler, data) then begin
    ListBox1.Items.Add(return_name);
    while Process32Next(handler, data) do
      ListBox1.Items.Add(return_name);
    end else
      ShowMessage('Ошибка получения информации');
end;
```

Примечание

Для работы примера укажите в uses *модуль* TLHELP32.

Определение загрузки ресурсов GDI и USER

Как из приложения определить загрузку ресурсов GDI и USER?

Решение

```
program res;
{$APPTYPE CONSOLE}
// индикатор ресурсов

function MyGetFreeSystemResources32(Id: integer): integer;
          stdcall; external 'rsrc32' name '_MyGetFreeSystemResources32@4';
const
  rSystem = 0;
  rGDI = 1;
  rUSER = 2;

begin
  Writeln('free resources');
  Writeln('System:', MyGetFreeSystemResources32(rSystem), '%');
  Writeln('GDI:', MyGetFreeSystemResources32(rGDI), '%');
```

```
  Writeln('USER:', MyGetFreeSystemResources32(rUSER), '%');
  Readln;
end.
```

Получение информации о процессоре

Если необходимо не только «вычислить» частоту процессора, но и узнать о процессоре как можно больше – воспользуйтесь следующим модулем:

```
unit ExpandCPUInfo;

interface

type
  TCPUInfo = packed record
    IDString: array [0..11] of Char;
    Stepping: Integer;
    Model: Integer;
    Family: Integer;
    FPU: Boolean;
    VirtualModeExtensions: Boolean;
    DebuggingExtensions: Boolean;
    PageSizeExtensions: Boolean;
    TimeStampCounter: Boolean;
    K86ModelSpecificRegisters: Boolean;
    MachineCheckException: Boolean;
    CMPXCHG8B: Boolean;
    APIC: Boolean;
    MemoryTypeRangeRegisters: Boolean;
    GlobalPagingExtension: Boolean;
    ConditionalMoveInstruction: Boolean;
    MMX: Boolean;
    SYSCALLandSYSRET, FPConditionalMoveInstruction, AMD3DNow: Boolean;
    CPUName: String;
  end;

function ExistCPUID: Boolean;      { информация об идентификации процессора }
function CPUIDInfo(out info: TCPUInfo): Boolean;    { технологии процессора }
function ExistMMX: Boolean;
function Exist3DNow: Boolean;
function ExistKNI: Boolean;
procedure EMMS;
procedure FEMMS;
procedure PREFETCH(p: Pointer); register;

implementation

function ExistCPUID: Boolean;
asm
    pushfd
    pop   eax
```

```
    mov   ebx, eax
    xor   eax, 00200000h
    push  eax
    popfd
    pushfd
    pop   ecx
    mov   eax, 0
    cmp   ecx, ebx
    jz    @NO_CPUID
    inc   eax
@NO_CPUID:
end;

function CPUIDInfo(out info: TCPUInfo): boolean;

  function ExistExtendedCPUIDFunctions: boolean;
  asm
    mov  eax, 080000000h
  db  $0F, $A2
  end;

var
  name: array [0..47] of Char;
  p: Pointer;
begin
  if ExistCPUID then
    asm
      jmp   @Start
  @BitLoop:
      mov   al, dl
      and   al, 1
      mov   [edi], al
      shr   edx, 1
      inc   edi
      loop  @BitLoop
      ret
  @Start:
      mov   edi, info
      mov   eax, 0
  db  $0F, $A2
      mov   [edi],ebx
      mov   [edi + 4], edx
      mov   [edi + 8], ecx
      mov   eax, 1
  db  $0F, $A2
      mov   ebx, eax
      and   eax, 0fh;
      mov   [edi + 12], eax;
      shr   ebx, 4
      mov   eax, ebx
      and   eax, 0fh
```

```
    mov   [edi + 12 + 4], eax
    shr   ebx, 4
    mov   eax, ebx
    and   eax, 0fh
    mov   [edi + 12 +8 ], eax
    add   edi, 24
    mov   ecx, 6
    call  @BitLoop
    shr   edx, 1
    mov   ecx, 3
    call  @BitLoop
    shr   edx, 2
    mov   ecx, 2
    call  @BitLoop
    shr   edx, 1
    mov   ecx, 1
    call  @BitLoop
    shr   edx, 7
    mov   ecx, 1
    call  @BitLoop
    mov   p, edi
end;
if (info.IDString = 'AuthenticAMD') and ExistExtendedCPUIDFunctions then begin
  asm
    mov   edi, p
    mov   eax, 080000001h
db  $0F, $A2
    mov   eax, edx
    shr   eax, 11
    and   al, 1
    mov   [edi], al
    mov   eax, edx
    shr   eax, 16
    and   al, 1
    mov   [edi + 1], al
    mov   eax, edx
    shr   eax, 31
    and   al, 1
    mov   [edi + 2], al
    lea   edi, name
    mov   eax, 0
    mov   [edi], eax
    mov   eax, 080000000h
db  $0F, $A2
    cmp   eax, 080000004h
    jl    @NoString
    mov   eax, 080000002h
db  $0F, $A2
    mov   [edi], eax
    mov   [edi + 4], ebx
    mov   [edi + 8], ecx
```

```
      mov   [edi + 12], edx
      add   edi, 16
      mov   eax, 080000003h
  db  $0F, $A2
      mov   [edi], eax
      mov   [edi + 4], ebx
      mov   [edi + 8], ecx
      mov   [edi + 12], edx
      add   edi, 16
      mov   eax, 080000004h
  db  $0F, $A2
      mov   [edi], eax
      mov   [edi + 4], ebx
      mov   [edi + 8], ecx
      mov   [edi + 12], edx
    @NoString:
    end;
    info.CPUName := name;
  end else
   with info do begin
    SYSCALLandSYSRET := False;
    FPConditionalMoveInstruction := False;
    AMD3Dnow := False;
    CPUName := '';
  end;
  Result := ExistCPUID;
end;

function ExistMMX: Boolean;
var
  info: TCPUInfo;
begin
  if CPUIDInfo(info) then Result := info.MMX
  else Result := False;
end;

function Exist3DNow: Boolean;
var
  info: TCPUInfo;
begin
  if CPUIDInfo(info) then Result := info.AMD3DNow
  else Result := False;
end;

function ExistKNI: Boolean;
begin
  Result := False;
end;

procedure EMMS;
asm
```

```
  db $0F, $77
end;

procedure FEMMS;
asm
  db $0F, $03
end;

procedure PREFETCH(p: Pointer); register;
asm
  // PREFETCH byte ptr [eax]
end;

end.
```

[Popov Igor]

Определяем процессор

Как определить тип установленного процессора?

Ниже приведен код модуля Delphi, позволяющий определить тип CPU. Данный код ориентирован на встроенный ассемблер Delphi и на процессоры не старше Pentium.

```
unit CpuId;

interface

type
{ Все типы известны. В случае создания новых дайте им пригодные имена и
  соответственно расширьте функцию CpuTypeString. }

TCpuType = (cpu8086, cpu80286, cpu386, cpu486, cpuPentium);

{ Возвращает тип текущего CPU }
function CpuType: TCpuType;
{ Возвращаем тип как короткую строку }
function CpuTypeString: String;

implementation

uses
  SysUtils;

function CpuType: TCpuType; assembler;
asm
    push ds
{ Во-первых, проверяем наличие 8086 CPU }
{ Биты 12-15 в регистре FLAGS всегда выставлены в процессоре 8086. }
    pushf                   // сохраняем EFLAGS
```

```
    pop    bx              // сохраняем EFLAGS в BX
    mov    ax, 0fffh       // сбрасываем биты 12-15
    and    ax, bx          // в EFLAGS
    push   ax              // сохраняем в стеке значение EFLAGS
    popf                   // замещаем текущее значение EFLAGS
    pushf                  // устанавливаем новый EFLAGS
    pop    ax              // сохраняем новый EFLAGS в AX
    and    ax, 0f000h      // если биты 12-15 выставлены, то CPU
    cmp    ax, 0f000h      // 8086/8088
    mov    ax, cpu8086     // вывешиваем флаг 8086/8088
    je     @@End_CpuType

{ Проверка 80286 CPU }
{ Биты 12-15 регистра FLAGS всегда очищены в случае процессора 80286. }
    or     bx, 0f000h      // пробуем установить биты 12-15
    push   bx
    popf
    pushf
    pop    ax
    and    ax, 0f000h       // если биты 12-15 сброшены, CPU = 80286
    mov    ax, cpu80286     // включаем флаг 80286
    jz     @@End_CpuType

{ Для определения процессоров 386 и выше необходимо использовать 32-битные
  инструкции, но 16-битный ассемблер Delphi не признает 32-битные коды операций и
  операнды. Взамен этого используется префикс размера операнда 66H, который
  позволяет преобразовать каждую инструкцию в ее 32-битный эквивалент. Для
  непосредственных 32-битных операндов нужно также хранить первое слово
  операнда сразу после следующей за ним инструкции. 32-битная инструкция показана в
  комментариях после инструкции 66H. }

{ Проверяем на i386 CPU }
{ Бит AC и бит #18 - новые биты, введенные в регистр EFLAGS в i486 DX CPU для
  компенсационного выравнивания. Этот бит не может быть выставлен в i386 CPU. }

  db 66h                   // pushfd
    pushf
  db 66h                   // pop eax
    pop    ax              // получаем оригинальный EFLAGS
  db 66h                   // mov ecx, eax
    mov    cx, ax          // сохраняем оригинальный EFLAGS
  db 66h                   // xor eax, 40000h
    xor    ax, 0h          // перебрасываем бит AC в EFLAGS
  dw 0004h
  db 66h                   // push eax
    push   ax              // сохраняем для EFLAGS
  db 66h                   // popfd
    popf                   // копируем в EFLAGS
  db 66h                   // pushfd
    pushf                  // выталкиваем EFLAGS
  db 66h                   // pop eax
```

```
    pop ax                  // получаем новое значение EFLAGS
  db 66h                    // xor eax, ecx
    xor    ax, cx           // невозможно переключить бит AC, CPU=Intel386
    mov    ax, cpu386       // включаем 386 флаг
    je     @@End_CpuType

{ Производим проверку i486 DX CPU / i487 SX MCP и i486 SX CPU }
{ Проверяем способность устанавливать/сбрасывать флаг ID (бит 21) в EFLAGS, который
  указывает присутствие процессора с помощью инструкции CPUID. }
  db 66h                    // pushfd
    pushf                   // выталкиваем оригинальный EFLAGS
  db 66h                    // pop eax
    pop    ax               // получаем оригинальный EFLAGS в eax
  db 66h                    // mov ecx, eax
    mov    cx, ax           // сохраняем оригинальный EFLAGS в ecx
  db 66h                    // xor eax, 200000h
    xor    ax, 0h           // перебрасываем бит ID в EFLAGS
  dw 0020h
  db 66h                    // push eax
    push   ax               // сохраняем для EFLAGS
  db 66h                    // popfd
    popf                    // копируем в EFLAGS
  db 66h                    // pushfd
    pushf                   // выталкиваем EFLAGS
  db 66h                    // pop eax
    pop    ax               // получаем новое значение EFLAGS
  db 66h                    // xor eax, ecx
    xor    ax, cx
    mov    ax, cpu486       // вывешиваем флаг i486
    je     @@End_CpuType    // если бит ID не может быть изменен, CPU=486

{ инструкция CPUID без функционального назначения }
{ Выполняем инструкцию CPUID для определения поставщика, семейства, модели и
  версии.
  Инструкция CPUID для этих целей может быть использована, начиная с версии B0
  процессора P5. }
  db 66h                    // mov eax, 1
    mov    ax, 1            // устанавливаем для инструкции CPUID
  dw 0
  db 66h                    // cpuid
  db 0Fh                    // код операции Hardcoded для инструкции CPUID
  db 0a2h
  db 66h                    // and eax, 0F00H
    and    ax, 0F00H        // все маскируем
  dw 0
  db 66h                    // shr eax, 8
    shr    ax, 8            // сдвигаем тип cpu вплоть до нижнего байта
    sub    ax, 1            // вычитаем 1 для отображения на TCpuType
@@End_CpuType:
    pop    ds
end;
```

```
function CpuTypeString: String;
var
  kind: TCpuType;
begin
  kind := CpuType;
  case kind of
    cpu8086:     Result := '8086';
    cpu80286:    Result := '80286';
    cpu386:      Result := '386';
    cpu486:      Result := '486';
    cpuPentium:  Result := 'Pentium';
  else
{ Пытаемся добавить "гибкости" для будущих версий процессоров, например, для P6. }
    Result := Format('P%d', [Ord(kind)]);
  end;
end;

end.
```

[News Group]

Примечание

Данный код не работает для современных процессоров (выше Pentium) без существенной модификации. Необходимо добавить модули детектирования новых CPU и несколько изменить сам алгоритм определения. Этот пример сохранен для демонстрации еще одного подхода определения CPU.

Загрузка процессора

Как получить информацию о загрузке процессора?

Используйте информацию из реестра

```
HKEY_DYN_DATA\PerfStats\StatData, смотрим ключ Kernel\CPUUsage.
```

[News Group]

Примечание

Максимальное значение ключа равно 64

Работа с портами микропроцессора

Модуль для работы с портами микропроцессора.

Работает под Windows 9x и не работает под Windows NT.

```
unit Ports;

interface
```

```
type
  TPort = class
  private
    procedure Set_(index_: word; value: byte); register;
    function Get_(index_: word): byte; register;
  public
    property Element[index_: word]: byte read Get_ write Set_; default;
  end;
  TPortW = class
  private
    procedure Set_(index_: word; value: Word); register;
    function Get_(index_: word): word; register;
  public
    property Element[index_: word]: word read Get_ write Set_; default;
  end;

var
  Port: TPort;
  PortW: TPortW;
implementation

procedure TPort.Set_(index_: word; value: byte);
begin
  asm
    mov  dx, index_
    mov  al, value
    out  dx, al
  end;
end;

function TPort.Get_(index_: word): byte;
begin
  asm
    mov  dx, index_
    in   al, dx
    mov  @Result, al
  end;
end;

procedure TPortW.Set_(index_: word; value: word);
begin
  asm
    mov  dx, index_
    mov  ax, value
    out  dx, ax
  end;
end;

function TPortW.Get_(index_: word): word;
begin
  asm
```

```
    mov  dx, index_
    in   ax, dx
    mov  @Result, ax
  end;
end;

initialization
  Port := TPort.Create;
  PortW := TPortW.Create;

finalization
  Port.free;
  PortW.free;

end.
```

[Zolotorenki Pavlo]

CPU Speed

Ниже приводится код, позволяющий определить рабочую тактовую частоту CPU:

```
function GetCPUSpeed: Double;
const
  DelayTime = 500;
var
  TimerHi: DWORD;
  TimerLo: DWORD;
  PriorityClass: Integer;
  Priority: Integer;
begin
  PriorityClass := GetPriorityClass(GetCurrentProcess);
  Priority := GetThreadPriority(GetCurrentThread);
  SetPriorityClass(GetCurrentProcess, REALTIME_PRIORITY_CLASS);
  SetThreadPriority(GetCurrentThread, THREAD_PRIORITY_TIME_CRITICAL);
  Sleep(10);
  asm
    DW   310Fh            // rdtsc
    MOV  TimerLo, EAX
    MOV  TimerHi, EDX
  end;
  Sleep(DelayTime);
  asm
    DW   310Fh            // rdtsc
    SUB  EAX, TimerLo
    SBB  EDX, TimerHi
    MOV  TimerLo, EAX
    MOV  TimerHi, EDX
  end;
  SetThreadPriority(GetCurrentThread, Priority);
```

```
  SetPriorityClass(GetCurrentProcess, PriorityClass);
  Result := TimerLo / (1000.0 * DelayTime);
end;
// использование ...
  Label1.Caption := Format('CPU speed: %f MHz', [GetCPUSpeed]);
```

[News Group]

Форматирование носителя

Как отформатировать носитель под Win32?

Используйте ShFormatDrive:

```
function SHFormatDrive(hWnd: HWND; Drive: Word; fmtID: Word;
     Options: Word): Longint; stdcall; external 'Shell32.dll' name 'SHFormatDrive';
const
  SHFMT_DRV_A = 0;
  SHFMT_DRV_B = 1;
  SHFMT_ID_DEFAULT = $FFFF;
  SHFMT_OPT_QUICKFORMAT = 0;
  SHFMT_OPT_FULLFORMAT = 1;
  SHFMT_OPT_SYSONLY = 2;
  SHFMT_ERROR = -1;
  SHFMT_CANCEL = -2;
  SHFMT_NOFORMAT = -3;

procedure TForm1.Button1Click(Sender: TObject);
var
  FmtRes: longint;
begin
  try
    FmtRes := ShFormatDrive(Handle, SHFMT_DRV_A, SHFMT_ID_DEFAULT,
                            SHFMT_OPT_QUICKFORMAT);
    case FmtRes of
        SHFMT_ERROR:    ShowMessage('Error formatting the drive');
        SHFMT_CANCEL:   ShowMessage('User canceled formatting the drive');
        SHFMT_NOFORMAT: ShowMessage('No Format')
      else
        ShowMessage('Disk has been formatted');
    end;
  except
  end;
end;
```

Определение свободного места на диске

Как определить количество свободного места на диске размером более 2 Гбайт?

Для этого потребуется использовать GetDiskFreeSpaceEx() с последующим преобразованием целочисленных значений к типу Double.

```
GetDiskFreeSpaceEx: function (Directory: PChar; var FreeAvailable,
          TotalSpace: TLargeInteger; TotalFree: PLargeInteger): Bool stdcall = nil;

procedure GetDiskSizeAvail(TheDrive: PChar; var TotalBytes: double;
                           var TotalFree: double);
var
  AvailToCall, TheSize: TLargeInteger;
  FreeAvail: PLargeInteger;
begin
  GetDiskFreeSpaceEx(TheDrive, AvailToCall, TheSize, FreeAvail);
{$IFOPT Q+}
{$DEFINE TURNOVERFLOWON}
{$Q-}
{$ENDIF}
  if TheSize >= 0 then TotalBytes := TheSize
  else if TheSize = -1 then begin
    TotalBytes := $7FFFFFFF;
    TotalBytes := TotalBytes * 2;
    TotalBytes := TotalBytes + 1;
  end else begin
    TotalBytes := $7FFFFFFF;
    TotalBytes := TotalBytes + abs($7FFFFFFF - TheSize);
  end;
  if AvailToCall >= 0 then TotalFree := AvailToCall
  else if AvailToCall = -1 then begin
    TotalFree := $7FFFFFFF;
    TotalFree := TotalFree * 2;
    TotalFree := TotalFree + 1;
  end

  else begin
    TotalFree := $7FFFFFFF;
    TotalFree := TotalFree + abs($7FFFFFFF - AvailToCall);
  end;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  TotalBytes: double;
  TotalFree: double;
begin
  GetDiskSizeAvail('C:\', TotalBytes, TotalFree);
  ShowMessage('Total bytes: ' + FloatToStr(TotalBytes));
  ShowMessage('Total bytes free: ' + FloatToStr(TotalFree));
end;
```

Примечание

Функция GetDiskFreeSpaceEx *описана в* SysUtils.

Серийный номер тома

Как с помощью Delphi получить серийный номер жесткого диска?

Решение

```
procedure TForm1.Button1Click(Sender: TObject);
var
  SerialNum: dword;
  a, b: dword;
  Buffer: array [0..255] of char;
begin
  if GetVolumeInformation('c:\', Buffer, SizeOf(Buffer), @SerialNum,
                          a, b, nil, 0) then
    Label1.Caption := IntToStr(SerialNum);
end;
```

Управление дисководом

Как проверить готовность диска A:*?*

Решение 1

```
function DiskInDrive(const Drive: char): Boolean;
var
  DrvNum: byte;
  EMode: Word;
begin
  Result := false;
  DrvNum := Ord(Drive) - Ord('A') + 1;
  EMode := SetErrorMode(SEM_FAILCRITICALERRORS);
  try
    if DiskSize(DrvNum) = -1 then Result := true
    else MessageBeep(0);
  finally
    SetErrorMode(EMode);
  end;
end;
```

Решение 2

```
type
  TDriveState = (DS_NO_DISK, DS_UNFORMATTED_DISK, DS_EMPTY_DISK,
                 DS_DISK_WITH_FILES);

function DriveState(driveletter: Char): TDriveState;
var
  mask: String[6];
  sRec: TSearchRec;
  oldMode: Cardinal;
  retcode: Integer;
begin
  oldMode := SetErrorMode(SEM_FAILCRITICALERRORS);
```

```
  mask := '?:\*.*';
  mask[1] := driveletter;
{$I-}                              { не возбуждаем исключение при неудаче }
  retcode := FindFirst(mask, faAnyfile, SRec);
  FindClose(SRec);
{$I+}
  case retcode of
      0:  Result := DS_DISK_WITH_FILES;
    -18:  Result := DS_EMPTY_DISK;
  else
    Result := DS_NO_DISK;
  end;
  SetErrorMode(oldMode);
end;
```

Управление метками томов диска

Данный совет содержит исходный код модуля, который позволяет получить, установить и удалить метку тома гибкого или жесткого диска. Код получения метки тома содержит функцию Delphi `FindFirst`, код для установки и удаления метки тома использует вызов прерывания DOS 21h (функции 16h и 13h, соответственно). Поскольку функция 16h не поддерживается Windows, она должна вызываться через DPMI-прерывание 31h, функцию 300h.

```
unit VolLabel;

interface

uses
  Classes, SysUtils, WinProcs;
type
  EInterruptError = class(Exception);
  EDPMIError = class(EInterruptError);
  Str11 = String[11];

procedure SetVolumeLabel(NewLabel: Str11; Drive: Char);
function GetVolumeLabel(Drive: Char): Str11;
procedure DeleteVolumeLabel(Drv: Char);

implementation

type
  PRealModeRegs = ^TRealModeRegs;
  TRealModeRegs = record
  case Integer of
    0: (
      EDI, ESI, EBP, EXX, EBX, EDX, ECX, EAX: Longint;
      Flags, ES, DS, FS, GS, IP, CS, SP, SS: Word);
    1: (
      DI, DIH, SI, SIH, BP, BPH, XX, XXH: Word;
```

```
      case Integer of
        0: (
          BX, BXH, DX, DXH, CX, CXH, AX, AXH: Word);
        1: (
          BL, BH, BLH, BHH, DL, DH, DLH, DHH,
          CL, CH, CLH, CHH, AL, AH, ALH, AHH: Byte));
  end;
PExtendedFCB = ^TExtendedFCB;
TExtendedFCB = Record
  ExtendedFCBflag: Byte;
  Reserved1: array[1..5] of Byte;
  Attr: Byte;
  DriveID: Byte;
  FileName: array[1..8] of Char;
  FileExt: array[1..3] of Char;
  CurrentBlockNum: Word;
  RecordSize: Word;
  FileSize: LongInt;
  PackedDate: Word;
  PackedTime: Word;
  Reserved2: array[1..8] of Byte;
  CurrentRecNum: Byte;
  RandomRecNum: LongInt;
end;

procedure RealModeInt(Int: Byte; var Regs: TRealModeRegs);
{ процедура работает с прерыванием 31h, функцией 0300h для имитации }
{ прерывания режима реального времени для защищенного режима. }
var
  ErrorFlag: Boolean;
begin
  asm
    mov  ErrorFlag, 0  { успешное завершение }
    mov  ax, 0300h     { функция 300h }
    mov  bl, Int       { прерывание режима реального времени, которое }
                       { необходимо выполнить }
    mov  bh, 0         { требуется }
    mov  cx, 0         { помещаем слово в стек для копирования, принимаем ноль }
    les  di, Regs      { es:di = Regs }
    int  31h           { DPMI-прерывание 31h }
    jnc  @@End         { адрес перехода установлен в error }
@@Error:
    mov  ErrorFlag, 1  { возвращаем false в error }
@@End:
  end;
  if ErrorFlag then
    raise EDPMIError.Create('Неудача при выполнении DPMI-прерывания');
end;

function DriveLetterToNumber(DriveLet: Char): Byte;
{ функция преобразования символа буквы диска в цифровой эквивалент. }
```

```
begin
  if DriveLet in ['a'..'z'] then
    DriveLet := Chr(Ord(DriveLet) -32);
  if not (DriveLet in ['A'..'Z']) then
    raise EConvertError.CreateFmt('Не могу преобразовать %s в ' +
                                  'числовой эквивалент диска', [DriveLet]);
  Result := Ord(DriveLet) - 64;
end;

procedure PadVolumeLabel(var Name: Str11);
{ процедура заполнения метки тома диска строкой с пробелами }
var
  i: integer;
begin
  for i := Length(Name) + 1 to 11 do
    Name := Name + ' ';
end;

function GetVolumeLabel(Drive: Char): Str11;
{ функция возвращает метку тома диска }
var
  SR: TSearchRec;
  DriveLetter: Char;
  SearchString: String[7];
  P: Byte;
begin
  SearchString := Drive + ':\*.*';
{ ищем метку тома }
  if FindFirst(SearchString, faVolumeID, SR) = 0 then begin
    P := Pos('.', SR.Name);
    if P > 0 then begin                    { если у него есть точка... }
      Result := '       ';                 { пространство между именами }
      Move(SR.Name[1], Result[1], P - 1);  { и расширениями }
      Move(SR.Name[P + 1], Result[9], 3);
    end else begin
      Result := SR.Name;              { в противном случае обходимся без пробелов }
      PadVolumeLabel(Result);
    end;
  end else Result := '';
end;

procedure DeleteVolumeLabel(Drv: Char);
{ процедура удаления метки тома с данного диска }
var
  CurName: Str11;
  FCB: TExtendedFCB;
  ErrorFlag: WordBool;
begin
  ErrorFlag := False;
  CurName := GetVolumeLabel(Drv);          { получение текущей метки тома }
  FillChar(FCB, SizeOf(FCB), 0);           { инициализируем FCB нулями }
```

```
  with FCB do begin
    ExtendedFCBflag := $FF;                 { всегда }
    Attr := faVolumeID;                     { Атрибут Volume ID }
    DriveID := DriveLetterToNumber(Drv);    { Номер диска }
    Move(CurName[1], FileName, 8);          { необходимо ввести метку тома }
    Move(CurName[9], FileExt, 3);
  end;
  asm
    push  ds                                { сохраняем ds }
    mov   ax, ss                            { помещаем сегмент FCB (ss) в ds }
    mov   ds, ax
    lea   dx, FCB                           { помещаем смещение FCB в dx }
    mov   ax, 1300h                         { функция 13h }
    Call  DOS3Call                          { вызываем int 21h }
    pop   ds                                { восстанавливаем ds }
    cmp   al, 00h                           { проверка на успешность выполнения }
    je    @@End
@@Error:                                    { устанавливаем флаг ошибки }
    mov   ErrorFlag, 1
@@End:
  end;
  if ErrorFlag then
    raise EInterruptError.Create('Не могу удалить имя тома');
end;

procedure SetVolumeLabel(NewLabel: Str11; Drive: Char);
{ процедура присваивания метки тома диска. Имейте в виду, что }
{ данная процедура удаляет текущую метку перед установкой новой. }
{ Это необходимое требование для функции установки метки. }
var
  Regs: TRealModeRegs;
  FCB: PExtendedFCB;
  Buf: Longint;
begin
  PadVolumeLabel(NewLabel);
  if GetVolumeLabel(Drive) <> '' then           { если имеем метку... }
    DeleteVolumeLabel(Drive);                   { удаляем метку }
  Buf := GlobalDOSAlloc(SizeOf(PExtendedFCB));  { распределяем реальный буфер }
  FCB := Ptr(LoWord(Buf), 0);
  FillChar(FCB^, SizeOf(FCB), 0);               { инициализируем FCB нулями }
  with FCB^ do begin
    ExtendedFCBflag := $FF;                     { требуется }
    Attr := faVolumeID;                         { Атрибут Volume ID }
    DriveID := DriveLetterToNumber(Drive);      { Номер диска }
    Move(NewLabel[1], FileName, 8);             { устанавливаем новую метку }
    Move(NewLabel[9], FileExt, 3);
  end;
  FillChar(Regs, SizeOf(Regs), 0);
  with Regs do begin                            { Сегмент FCB }
    ds := HiWord(Buf);                          { отступ = ноль }
    dx := 0;
```

```
    ax := $1600;                                { Функция 16h }
  end;
  RealModeInt($21, Regs);                       { создаем файл }
  if (Regs.al <> 0) then                       { проверка на успешность выполнения }
    raise EInterruptError.Create('Не могу создать метку тома');
end;

end.
```

Примечание

Совет дан для Delphi 1, поэтому нуждается в переработке под современные версии Delphi.

Проблемы утечки памяти при работе с FindFirst, FindNext

Как избавиться от утечек памяти при использовании `FindFirst`*,* `FindNext`*?*

Необходимо применять эти функции, явно указывая их определение в модуле `SysUtils`.

```
...
begin
  Result := SysUtils.FindFirst(Path, Attr, SearchRec);
  while Result = 0 do begin
    ProcessSearchRec(SearchRec);
    Result := SysUtils.FindNext(SearchRec);
  end;
  SysUtils.FindClose(SearchRec);
end;
```

Копирование с диска на дискету и обратно

Вместо побайтного чтения следует открыть файл с размером записи, равным 64 Кбайт или около того и читать блоками.

Пример процедуры копирования файла `CopyFile`:

```
function CopyFile(FromPath, ToPath: String): integer;
var
  F1, F2: file;
  NumRead, NumWritten: integer;
  Buf: pointer;
  BufSize: longint;
  Totalbytes, TotalRead: longint;
begin
  Result := 0;
  AssignFile(f1, FromPath);
  AssignFile(F2, ToPath);
  Reset(F1, 1);
  TotalBytes := FileSize(F1);
  Rewrite(F2, 1);
```

```
  BufSize := 16384;
  GetMem(Buf, BufSize);
  TotalRead :=0;
  repeat
    BlockRead(F1, Buf^, BufSize, NumRead);
    inc(TotalRead, NumRead);
    BlockWrite(F2, Buf^, NumRead, NumWritten);
  until (NumRead = 0) or (NumWritten <> NumRead);
  if (NumWritten <> NumRead) then begin
{ ошибка }
    Result := -1;
  end;
  CloseFile(f1);
  CloseFile(f2);
end;
```

Для нетипизированных файлов можно использовать функцию Blockread, которая позволяет устанавливать размер буфера, равный 64 Кбайт. Ниже представлен «быстрый» способ достижения цели. Рекомендуем утилиту Compress (которую можно найти в поставке Delphi или на сайте Microsoft), позволяющую создавать файлы типа filename.ex. Это означает, что для копирования информации требуется гораздо меньше усилий.

Ниже приведен код, позволяющий копировать файлы. Для работы процедуры необходимо, чтобы файлы были несжатыми.

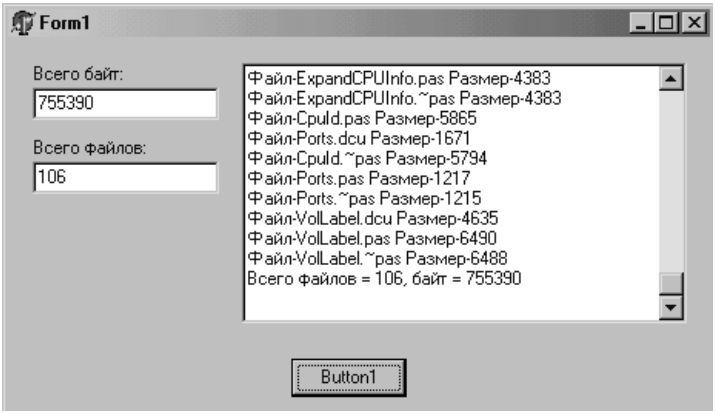
```
function TInstallForm.UnCompress(src, dest: String; var Error: LongInt): Boolean;
var
  s, d: TOFStruct;
  fs, fd: Integer;
  fnSrc, fnDest: PChar;
begin
  src := src + #0;
  dest := dest + #0;
  fnSrc := @src[1];                        { Хитро преобразуем строки в ASCIIZ }
  fnDest := @dest[1];
  fs := LZOpenFile(fnSrc, s, OF_READ);     { Получаем дескриптор файла }
  fd := LZOpenFile(fnDest, d, OF_CREATE);
  Error := LZCopy(fs, fd);                 { Вот магический вызов API }
  Result := (Error > -1);
  LZClose(fs);                             { Убедитесь, что закрыли! }
  LZClose(fd);
end;

procedure UnCompressError(Error: LongInt);
begin
  case Error of
    LZERROR_BADINHANDLE:  S := 'Неверный дескриптор исходного файла';
   LZERROR_BADOUTHANDLE:  S := 'Неверный дескриптор целевого файла';
       LZERROR_BADVALUE:  S := 'Входной параметр вышел за границы диапазона';
      LZERROR_GLOBALLOC:  S := 'Недостаточно памяти для требуемого буфера';
```

```
        LZERROR_GLOBLOCK:  S := 'Неверный дескриптор структуры внутренних данных';
            LZERROR_READ:  S := 'Неверный формат исходного файла';
      LZERROR_UNKNOWNALG:  S := 'Исходный файл был упакован с использованием'
                              + 'неизвестного алгоритма сжатия';
           LZERROR_WRITE:  S := 'Недостаточно места для выходного файла'
                    else
                           S := 'Неизвестная проблема с распаковкой'
  end;
  MessageDlg(S, mtConfirmation, [mbOK], 0);
  Close;
end;

function CopyFile(SrcName, DestName: string): boolean;
{ базовый метод копирования файла требует полный путь & имя для исходного
  & целевого файла }
var
  Buf: array[1..1024*4] of byte;
{ Этот размер может быть изменен. Объявляя указатель, вы можете использовать
  GetMem для создания в куче большого буфера }
  TotalRead: longint;
  NumRead, NumWritten: word;
  TotalWritten: longint;
  FromFileSize: longint;
  FrF, ToF: file;
  FileTime: longint;
begin
  FGetTime(SrcName, FileTime);
  Assign(FrF, SrcName);
  Reset(FrF, 1);
  FromFileSize := FileSize(FrF);
  Assign(ToF, DestName);
  Rewrite(ToF, 1);
  TotalRead := 0;
  TotalWritten := 0;
  repeat
    BlockRead (FrF, Buf, SizeOf(Buf), NumRead);
    TotalRead := TotalRead + NumRead;
    BlockWrite(ToF, Buf, NumRead, NumWritten);
    TotalWritten := TotalWritten + NumWritten;
  until (NumRead = 0) or (NumWritten <> NumRead);
  Close(FrF);
  Close(ToF);
{ возвращаем true, если они равны, в противном случае возвращаем false }
  CopyFile := (TotalWritten = FromFileSize);
end;
```

Примечание

Код работает только для Delphi 1. Для современных версий Delphi нужна переработка.

Получение размера файла



Код для определения информации о группе файлов:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Fhnd2: File;
  sPath: String;
  tpath: string;
  SearchRec: TSearchRec;
  tempsearch: string;
  tempfiles: Integer;
  tempbytes: LongInt;
  wBytes: Word;
  sTemp: String;
  iLen: Integer;
  szString: array[0..128] of Char;
  ec: integer;
  SysDir: string;
  Files2bProc, Bytes2bProc: integer;
begin
  Memo1.Clear;
  Memo1.Lines.Add('Запуск');
  sTemp := ParamStr(0);
  iLen := Length(sTemp);
  while sTemp[iLen] <> '\' do Dec(iLen);
  StrPCopy(szString, sTemp);
  szString[iLen] := #0;
  SysDir := StrPas(szString);
  tempbytes := 0;
  tempfiles := 0;
  Files2bProc := 0;
  Bytes2bProc := 0;
  Memo1.Lines.Add('Калькулируем файлы для обработки');
```

```
{ Подсчитываем, сколько файлов и байт должны быть обработаны }
  tempsearch := SysDir + '*.*';
  ec := FindFirst(tempsearch, faSysFile, SearchRec);
  while ec = 0 do begin
    if ((SearchRec.Name <> '.') and (SearchRec.Name <> '..')) then begin
      tempfiles := tempfiles + 1;
      tempbytes := tempbytes + SearchRec.Size;
      TotalInBytes.Text := IntToStr(tempbytes);
      TotalInFiles.Text := IntToStr(tempfiles);
      Memo1.Lines.Add('Файл-' + SearchRec.Name + ' Размер-' +
                      IntToStr(SearchRec.Size));
    end;
    ec := FindNext(SearchRec);
  end;
  Memo1.Lines.Add('Всего файлов = ' + IntToStr(tempfiles) + ', байт = ' +
                  IntToStr(tempbytes));
end;
```

В зависимости от предпочтений, можно выбрать одну из двух приведенных ниже функций. В первой реализуется идея временного изменения атрибутов файла, необходимого для его чтения. Вторая использует Windows API, но не содержит средств проверки ошибок.

```
function FileGetSize1(Filename: String): LongInt;
var
  F: File;
  OldFileAttr: Integer;

begin
  if FileExists(Filename) then begin
    OldFileAttr := FileGetAttr(Filename);
    FileSetAttr(Filename, OldFileAttr and (faReadOnly xor $FFFF));
    try
      AssignFile(F, Filename);
      Reset(F, 1);
      Result := FileSize(F);
      CloseFile(F);
    finally
      FileSetAttr(Filename, OldFileAttr);
    end;
  end else
    Result := 0;
end;

function FileGetSize2(Filename: String): LongInt;
var
  FileHandle: Integer;
begin
  if FileExists(Filename) then begin
    FileName := FileName + chr(0);
    FileHandle := _lopen(@FileName[1], 0);
```

```
    Result := _llseek(FileHandle, 0, 2);
    _lclose(FileHandle);
  end else
    Result := 0;
end;
```

[News Group]

Примечание

Можно воспользоваться функцией `FileSize` *из* `System`. *Она возвращает длину файла в байтах или число записей в файле, но таким способом нельзя «измерять» текстовые файлы.*

Определение устройства CD-ROM

Это должно помочь:

```
function IsCDROM(DriveNum: Integer): Boolean; assembler;
asm
  mov  ax, 1500h     { смотрим на предмет MSCDEX }
  xor  bx, bx
  int  2fh
  or   bx, bx
  jz   @Finish
  mov  ax, 150Bh     { проверяем использование драйвера CD }
  mov  cx, DriveNum
  int  2fh
  or   ax, ax
@Finish:
end;
```

или:

```
function IsCdRom(DriveNum: Word): Boolean;
var
  F: WordBool;
begin
  asm
    mov   ax, 1500h     { тест на наличие MSCDEX }
    xor   bx, bx
    int   2fh
    mov   ax, bx        { если bx = нулю, MSCDEX не присутствует }
    or    ax, ax        { возвращаем FALSE }
    jz    @no_mscdex
    mov   ax, 150bh     { проверка наличия драйвера MSCDEX }
    mov   cx, DriveNum  { cx содержит номер устройства }
    int   2fh
@no_mscdex:
    mov   f, ax
  end;
```

```
  Result := F;          { Назначаем возвращаемое функцией значение }
end;
```

Под Win32 функция GetDriveType правильно возвращает устройство CD-ROM.

Открытие и закрытие привода CD-ROM

Есть ли в API Win32 функция, позволяющая не только открыть, но и закрыть CD-ROM? Компонентом TMediaPlayer *пользоваться не хочу, тем более что компакт он может только извлечь...*

Для открытия CD-ROM:

```
mciSendString('Set cdaudio door open wait', nil, 0, handle);
```

Для закрытия CD-ROM:

```
mciSendString('Set cdaudio door closed wait', nil, 0, handle);
```

Не забудьте включить MMSystem в список используемых модулей.

При наличии более одного CD-ROM в системе рекомендуем обратиться к следующим функциям:

```
unit DriveTools;

interface

uses
  Windows, SysUtils, MMSystem;

function CloseCD(Drive: Char): Boolean;
function OpenCD(Drive: Char): Boolean;

implementation

function OpenCD(Drive: Char): Boolean;
var
  Res: MciError;
  OpenParm: TMCI_Open_Parms;
  Flags: DWord;
  S: String;
  DeviceID: Word;
begin
  Result := False;
  S := Drive + ':';
  Flags := mci_Open_Type or mci_Open_Element;
  with OpenParm do begin
    dwCallback := 0;
    lpstrDeviceType := 'CDAudio';
    lpstrElementName := PChar(S);
  end;
```

```
  Res := mciSendCommand(0, mci_Open, Flags, Longint(@OpenParm));
  if Res <> 0 then exit;
  DeviceID := OpenParm.wDeviceID;
  try
    Res := mciSendCommand(DeviceID, MCI_SET, MCI_SET_DOOR_OPEN, 0);
    if Res = 0 then exit;
    Result := True;
  finally
    mciSendCommand(DeviceID, mci_Close, Flags, Longint(@OpenParm));
  end;
end;

function CloseCD(Drive: Char): Boolean;
var
  Res: MciError;
  OpenParm: TMCI_Open_Parms;
  Flags: DWord;
  S: String;
  DeviceID: Word;
begin
  Result := false;
  S := Drive + ':';
  Flags := mci_Open_Type or mci_Open_Element;
  with OpenParm do begin
    dwCallback := 0;
    lpstrDeviceType := 'CDAudio';
    lpstrElementName := PChar(S);
  end;
  Res := mciSendCommand(0, mci_Open, Flags, Longint(@OpenParm));
  if Res <> 0 then exit;
  DeviceID := OpenParm.wDeviceID;
  try
    Res := mciSendCommand(DeviceID, MCI_SET, MCI_SET_DOOR_CLOSED, 0);
    if Res = 0 then exit;
    Result := True;
  finally
    mciSendCommand(DeviceID, mci_Close, Flags, Longint(@OpenParm));
  end;
end;

end.
```

[News Group]

Использование клавиш для управления компонентами

Если у меня есть некая кнопка (или флажок, переключатель и т. п.), то почему я не могу с помощью клавиш курсора управлять ею?

После некоторых экспериментов был создан метод, который приводится ниже. Он способен перехватывать в форме все нажатые клавиши позициониро-

вания и с их помощью управлять выбранным в настоящий момент элементом управления. Имейте в виду, что элементы управления (кроме компонентов Label) должны поддерживать возможность их выбора. Прежде чем выбрать GroupBox или другой компонент, убедитесь, что их свойство TabStop установлено в True. Можно переместить фокус на GroupBox, но, поскольку он не выделяется целиком, узнать, что он действительно имеет фокус, достаточно непросто. Если не требуется передавать управление в контейнерные элементы (нижеследующий код исходит из этого предположения), то можно управлять элементами, просто передавая управление в сам GroupBox.

В рассматриваемом здесь коде FormActivate является обработчиком события формы OnActivate, тогда как ProcessFormMessages никакого отношения к событиям формы не имеет. Не забудьте поместить объявление процедуры ProcessFormMessages в секцию Private класса вашей формы.

```
procedure TForm1.FormActivate(Sender: TObject);
begin
{ Делаем ссылку на нового обработчика сообщений }
  Application.OnMessage := ProcessFormMessages;
end;
procedure TForm1.ProcessFormMessages(var Msg: tMsg; var Handled: Boolean);
var
  Increment: Byte;
  TheControl: tWinControl;
begin
{ проверка наличия системного сообщения KeyDown }
  case Msg.Message of
    WM_KEYDOWN:  if Msg.wParam in [VK_UP, VK_DOWN, VK_LEFT, VK_RIGHT] then begin
{ изменяем величину приращения в зависимости от состояния клавиши Shift }
      if GetKeyState(VK_SHIFT) and $80 = 0 then Increment := 8
      else Increment := 1;
{ Этот код перемещает управление на родительский GroupBox, если один из его
  контейнерных элементов получает фокус. Если вам необходимо управлять элементами
  внутри контейнера, удалите блок IF и измените в блоке CASE TheControl на
  ActiveControl }
      if (ActiveControl.Parent is tGroupBox) then
        TheControl := ActiveControl.Parent
      else TheControl := ActiveControl;
      case Msg.wParam of
           VK_UP:  TheControl.Top := TheControl.Top - Increment;
         VK_DOWN:  TheControl.Top := TheControl.Top + Increment;
         VK_LEFT: TheControl.Left := TheControl.Left - Increment;
        VK_RIGHT: TheControl.Left := TheControl.Left + Increment;
      end;
{ сообщаем о том, что сообщение обработано }
      Handled := True;
    end;
  end;
end;
```

[News Group]

Как перехватить нажатия клавиш в системе

Для этого используется функция GetAsyncKeyState(KeyCode), которой в качестве параметра передаются коды клавиш. GetAsyncKeyState возвращает ненулевое значение, если во время ее вызова нажата указанная клавиша.

```
if GetAsyncKeyState(65) <> 0 then ShowMessage('A - pressed');
```

[Nikolaev Igor]

Особенности использования KeyPreview

В обработчик события FormCreate вставьте следующую строчку кода:

```
KeyPreview := True;
```

Это позволит всем событиям, связанным с нажатием клавиш, в первую очередь передаваться форме, обработчики которой могут выполнить какое-то заранее заданное действие или «подавить» нажатия клавиш. Только после этого они передаются выбранному элементу управления. Чтобы полностью «подавить» клавишу, используйте событие OnKeyPress, где код нажатой клавиши имеет тип Char, и для того чтобы «подавить» его, просто напишите key := #0.

Перехват клавиатуры

Решение

Project1.dpr

```
program Project1;

uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};
{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

SendKey.dpr

```
library SendKey;

uses
  SysUtils, Classes, Windows, Messages;

const
{ пользовательские сообщения }
  wm_LeftShow_Event = wm_User + 133;
  wm_RightShow_Event = wm_User + 134;
```

```
  wm_UpShow_Event = wm_User + 135;
  wm_DownShow_Event = wm_User + 136;
{ handle для ловушки }
  HookHandle: hHook = 0;

var
  SaveExitProc: Pointer;

{ собственно ловушка }
function Key_Hook(Code: integer; wParam: word; lParam: Longint): Longint;
                 stdcall; export;
var
  H: HWND;
begin
{если Code >= 0, то ловушка может обработать событие}
  if (Code >= 0) and (lParam and $40000000 = 0) then begin
{ ищем окно по имени класса и по заголовку (Caption формы управляющей программы
  должен быть равен 'XXX' !!!!)}
    H := FindWindow('TForm1', 'XXX');
{ это те клавиши? }
    case wParam of
       VK_Left:  SendMessage(H, wm_LeftShow_Event, 0, 0);
      VK_Right:  SendMessage(H, wm_RightShow_Event, 0, 0);
         VK_Up:  SendMessage(H, wm_UpShow_Event, 0, 0);
       VK_Down:  SendMessage(H, wm_DownShow_Event, 0, 0);
    end;
{ если 0, то система должна дальше обработать это событие, если 1 - нет }
    Result := 0;
  end else if Code < 0 then   { если Code < 0, то нужно вызвать следующую ловушку }
    Result := CallNextHookEx(HookHandle, Code, wParam, lParam);
end;

procedure LocalExitProc; far;              { при выгрузке DLL надо снять ловушку }
begin
  if HookHandle <> 0 then begin
    UnhookWindowsHookEx(HookHandle);
    ExitProc := SaveExitProc;
  end;
end;

exports Key_Hook;

begin        { инициализация DLL при загрузке ее в память }
  HookHandle := SetWindowsHookEx(wh_Keyboard, @Key_Hook, hInstance, 0);
  if HookHandle = 0 then
    MessageBox(0, 'Unable to set hook!', 'Error', mb_Ok)
  else begin
    SaveExitProc := ExitProc;
    ExitProc := @LocalExitProc;
  end;
end.
```

Unit1.dfm

```
object Form1: TForm1
  Left = 200
  Top = 104
  Width = 544
  Height = 375
  Caption = 'XXX'
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  PixelsPerInch = 96
  TextHeight = 13
    object Label1: TLabel
      Left = 128
      Top = 68
      Width = 32
      Height = 13
      Caption = 'Label1'
  end
end
```

Unit1.pas

```
unit Unit1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls, Forms,
   Dialogs, StdCtrls;
{ пользовательские сообщения }
const
  wm_LeftShow_Event = wm_User + 133;
  wm_RightShow_Event = wm_User + 134;
  wm_UpShow_Event = wm_User + 135;
  wm_DownShow_Event = wm_User + 136;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    procedure FormCreate(Sender: TObject);

  private
    procedure WM_LeftMSG(Var M: TMessage); message wm_LeftShow_Event;
    procedure WM_RightMSG(Var M: TMessage); message wm_RightShow_Event;
    procedure WM_UpMSG(Var M: TMessage); message wm_UpShow_Event;
    procedure WM_DownMSG(Var M: TMessage); message wm_DownShow_Event;
  end;
```

```
var
  Form1: TForm1;
  P: Pointer;

implementation

{$R *.DFM}

//Загрузка DLL
function Key_Hook(Code: integer; wParam: word; lParam: Longint): Longint;
                  stdcall; external 'SendKey' name 'Key_Hook';
procedure TForm1.WM_LeftMSG(Var M: TMessage);
begin
  Label1.Caption := 'Left';
end;

procedure TForm1.WM_RightMSG(Var M: TMessage);
begin
  Label1.Caption := 'Right';
end;

procedure TForm1.WM_UpMSG(Var M: TMessage);
begin
  Label1.Caption := 'Up';
end;

procedure TForm1.WM_DownMSG (Var M : TMessage);
begin
  Label1.Caption := 'Down';
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
{ если не использовать вызов процедуры из DLL в программе, то компилятор удалит
  загрузку DLL из программы }
  P := @Key_Hook;
end;

end.
```

[Bogachev]

Блокирование ввода информации

Как заблокировать ввод?

Недокументированная функция из User32.dll, которая блокирует ввод (мышь, клавиатуру кроме <Ctrl>+<Alt>+<Del>). При нажатии <Ctrl>+ +<Alt>+<Del> все разблокируется.

```
procedure BlockInput; external 'user32.dll';
```

Передаем параметры в стек вручную через push (1 – заблокировать; 0 – разблокировать):

```
procedure Block;
asm
  push 1
  call BlockInput
end;

procedure UnBlock;
asm
  push 0
  call BlockInput
end;
```

[Pastushenko Andrew]

Имитация нажатия клавиши

У меня есть набор кнопок (`Caption = '0'..'9'`*), и я хотел бы имитировать их нажатие во время нажатия пользователем соответствующей клавиши. То есть, когда пользователь нажимает клавишу <1>, кнопка с таким заголовком должна быть нажата на экране. Как мне это сделать без нового компонента* `TButton`*?*

Вероятно, лучше использовать десять элементов управления TSpeedButton или их массив, поскольку этот тип кнопок имеет свойство Down. Для начала установите свойство KeyPreview вашей формы в True. Затем создайте обработчик события OnKeyDown примерно такого вида:

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  case Key of
    VK_NUMPAD0: btn0.Down := True;
    VK_NUMPAD1: btn1.Down := True;
    VK_NUMPAD2: btn2.Down := True;
    VK_NUMPAD3: btn3.Down := True;
    VK_NUMPAD4: btn4.Down := True;
    VK_NUMPAD5: btn5.Down := True;
    VK_NUMPAD6: btn6.Down := True;
    VK_NUMPAD7: btn7.Down := True;
    VK_NUMPAD8: btn8.Down := True;
    VK_NUMPAD9: btn9.Down := True;
  end;
end;
```

В этом случае обработчик события OnKeyUp будет выглядеть так:

```
procedure TForm1.FormKeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  case Key of
```

```
      VK_NUMPAD0: btn0.Down := False;
      VK_NUMPAD1: btn1.Down := False;
      VK_NUMPAD2: btn2.Down := False;
      VK_NUMPAD3: btn3.Down := False;
      VK_NUMPAD4: btn4.Down := False;
      VK_NUMPAD5: btn5.Down := False;
      VK_NUMPAD6: btn6.Down := False;
      VK_NUMPAD7: btn7.Down := False;
      VK_NUMPAD8: btn8.Down := False;
      VK_NUMPAD9: btn9.Down := False;
    end;
  end;
```

Поэкспериментируйте со свойствами `AllowAllUp` и `GroupIndex` для получения необходимого эффекта.

Кроме того, массив кнопок `TSpeedButtons` является, вероятно, наиболее изящным решением данной задачи, поскольку в этом случае можно использовать константу `VK_ constant` в качестве индекса, благодаря чему обработчики обоих событий помещаются всего в одну строчку – `Button[VK_x].Down := True` (или `False`).

Индикация статуса клавиш

Где найти код, который помог бы мне связать текст строки состояния с состоянием клавиш <CapsLock>, <NumLock> *и др.?*

Событие `OnIdle` происходит каждый раз, когда приложение «не работает». С помощью обработчика данного события можно сделать так, чтобы во время «простоя» приложение могло выполнять второстепенные задачи. В это время оно находится в ожидании какого-то события, например, ввода пользователем новой команды.

`TIdleEvent` – процедурный тип, имеющий логический параметр `Done` со значением по умолчанию `True`. Если `Done` равен `True`, после обработки события `OnIdle` вызывается функция Windows API `WaitMessage`, передающая управление другим приложениям до тех пор, пока в очереди сообщений вашего приложения не появится новое сообщение. Если `Done` равно `False`, `WaitMessage` не вызывается.

Как же решить нашу задачу в свете вышесказанного?

Добавьте четыре компонента `CheckBox` к вашему компоненту `StatusBar` и включите следующее объявление в секцию `Private` вашей формы:

```
procedure AppOnIdle(Sender: TObject; var Done: Boolean);
```

Добавьте в секцию реализации:

```
procedure TForm1.AppOnIdle(Sender: TObject; var Done: Boolean);
begin
  CheckBox1.Checked := Odd(GetKeyState(VK_CAPITAL));
  CheckBox2.Checked := Odd(GetKeyState(VK_SHIFT));
```

```
  CheckBox3.Checked := Odd(GetKeyState(VK_NUMLOCK));
  CheckBox4.Checked := Odd(GetKeyState(VK_SCROLL));
  Done := False;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Application.OnIdle := AppOnIdle;
end;
```

Перехват курсорных клавиш

Как в приложении перехватить нажатие клавиш курсора?

Решение 1

Нужно перехватить и обработать сообщение `WM_GETDLGCODE`:

```
procedure WMGetDlgCode(var Msg: TMessage); message WM_GETDLGCODE;
```

затем реализовать его:

```
procedure TMyControl.WMGetDlgCode(var Msg: TMessage);
begin
  Msg.Result := DLGC_WANTARROWS;
end;
```

Решение 2

Создайте `HandleMessages` как метод формы и затем назначьте его объекту: `Application.HandleMessages`.

```
procedure TForm1.HandleMessages(Var Msg: tMsg; Var Handled: Boolean);
begin
  if (Msg.Message = WM_KeyDown)
    and (Msg.wParam in [VK_UP, VK_DOWN, VK_LEFT, VK_RIGHT]) then begin
    case Msg.wParam of
           VK_UP:  ShowMessage('Нажата стрелка вверх');
         VK_DOWN:  ShowMessage('Нажата стрелка вниз');
         VK_LEFT:  ShowMessage('Нажата стрелка влево');
        VK_RIGHT:  ShowMessage('Нажата стрелка вправо');
    end;
    Handled := True;
  end;
end;
```

[News Group]

Создание собственных «горячих» клавиш

Как перехватывать нажатие созданных в программе «горячих» клавиш?

Во-первых, установите свойство формы `KeyPreview := True`;

Затем сделайте что-то подобное следующему:

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if (ssCtrl in Shift) and (chr(Key) in ['A', 'a']) then ShowMessage('Ctrl-A');
end;
```

Обход элементов формы

Требуется, чтобы при нажатии клавиши <Enter> в любом из нескольких полей редактирования на форме, фокус переходил в следующий. Как это сделать?

Решение

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
  if Key = #13 then Begin
    Key := #0;
    SelectNext(Sender as TWinControl, True, True);
{ или Perform(WM_NEXTDLGCTL, 0, 0); }
  end;
end;
```

Не забудьте установить нужный порядок в списке `Tab Order`.

Отключение клавиш <Ctrl>+<Alt>+<Del>, <Alt>+<Tab>, <Ctrl>+<Esc> из приложения

Как подавить реакцию Windows на <Ctrl>+<Alt>+<Del>, <Alt>+<Tab>, <Ctrl>+<Esc>?

В некоторых случаях (например, при работе в полноэкранном режиме, показе своей презентации или экранной заставки) бывает полезно заблокировать перечисленные комбинации клавиш. Они блокируются при работе системы в режиме «экранная заставка», который в свою очередь несложно включить и выключить:

```
// Включение режима
SystemParametersInfo(SPI_SCREENSAVERRUNNING, 1, 0, 0);
// Выключение режима
SystemParametersInfo(SPI_SCREENSAVERRUNNING, 0, 0, 0);
```

Отключение <Alt>+<F4>

Обратите внимание на событие формы `CloseQuery`, – если оно возвращает `False`, то форму можно закрыть только командой `Close` из приложения – в этом случае при попытке закрытия формы вы могли бы успеть произвести необходимые действия.

Недоступность комбинации <Alt>+<Tab>

Как сделать недоступной комбинацию клавиш <Alt>+<Tab>?

Решение

```
program small;
var
  Dummy: integer;
```

```
begin
  Dummy := 0;
{Отключаем ALT-TAB}
  SystemParametersInfo( SPI_SETFASTTASKSWITCH, 1, @Dummy, 0);
{Отключаем CTRL-ALT-DEL}
  SystemParametersInfo( SPI_SCREENSAVERRUNNING, 1, @Dummy, 0);
end.
```

Хочу добавить, что этот код отключает не только <Alt>+<Tab>, <Ctrl>+<Alt>+<Del> но и <Ctrl>+<Esc> и клавиши вызова программ из меню Пуск, что в сочетании с удалением самой кнопки Пуск создает замечательный эффект.

А как все вернуть в исходное состояние?

Решение

```
SystemParametersInfo(SPI_SETFASTTASKSWITCH, 0, 0, 0);
SystemParametersInfo(SPI_SCREENSAVERRUNNING, 0, 0, 0);
```

[Subfire]

Управление клавишей <Caps Lock>

Как включить индикатор Caps Lock?

Решение

```
procedure TForm1.Button1Click(Sender: TObject);
var
  KeyState: TKeyboardState;
begin
  GetKeyboardState(KeyState);
  if (KeyState[VK_CAPITAL] = 0) then KeyState[VK_CAPITAL] := 1
  else KeyState[VK_CAPITAL] := 0;
  SetKeyboardState(KeyState);
end;
```

Для <Num Lock> замените VK_CAPITAL на VK_NUMLOCK.

Чтение и установка клавиши <Num Lock>

Решение

```
var
  KS: TKeyboardState;
  ...

begin
  GetKeyboardState(KS);
  if Odd(KS[VK_NUMLOCK]) then                    { NumLock включен }
  ...
  KS[VK_NUMLOCK] := KS[VK_NUMLOCK] XOR 1;        { переключение NumLock }
  KS[VK_NUMLOCK] := KS[VK_NUMLOCK] OR 1;         { включение NumLock }
```

```
  KS[VK_NUMLOCK] := KS[VK_NUMLOCK] AND (NOT 1);     { выключение NumLock }
  SetKeyboardState(KS);
  ...
end;
```

Определение нажатия клавиши <PrintScreen>

Как определить нажатие <PrintScreen>*?*

Решение

```
type
  TForm1 = class(TForm)
      Button1: TButton;
      procedure FormCreate(Sender: TObject);
    private
      procedure AppIdle(Sender: TObject; var Done: Boolean);
end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
  Application.OnIdle := AppIdle;
end;
procedure TForm1.AppIdle(Sender: TObject; var Done: Boolean);
begin
  if GetAsyncKeyState(VK_SNAPSHOT) <> 0 then Form1.Caption := 'SnapShot';
  Done := True;
end;
```

Управление индикаторами на клавиатуре

Как включать/выключать индикаторы NumLock, CapsLock и т. д.?

Решение

```
procedure SetNumLock(bState: Boolean);

var
  KeyState: TKeyboardState;

begin
  GetKeyboardState(KeyState);
  if ((bState) and (not ((KeyState[VK_NUMLOCK] and 1) = 1))
      or ((not (bState)) and ((KeyState[VK_NUMLOCK] and 1) = 1))) then

// имитация нажатия клавиши
    keybd_event(VK_NUMLOCK, $45, (KEYEVENTF_EXTENDEDKEY or 0), 0);
```

```
// имитация отпускания клавиши
  keybd_event(VK_NUMLOCK, $45, (KEYEVENTF_EXTENDEDKEY or KEYEVENTF_KEYUP), 0);
  SetKeyboardState(KeyState);
end;
```

Для других клавиш заменяйте VK_NUMLOCK.

[News Group]

Перехват нажатия клавиши <Tab>

Перехватить нажатие клавиши <Tab> из программы можно только в обработчике Application.OnMessages.

```
unit Unit1;

interface

uses
  Windows, SysUtils, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    Label1: TLabel;
    procedure FormCreate(Sender: TObject);
  private
    procedure AppMessage(var Msg: TMsg; var Handled: Boolean);
end;

var
  Form1: TForm1;

implementation
{$R *.DFM}

procedure TForm1.AppMessage(var Msg: TMsg; var Handled: Boolean);
const
  shiftPressed: boolean = false;

begin
  if Msg.Message = WM_KEYDOWN then
    if not shiftPressed and (Msg.wParam = VK_SHIFT) then begin
      shiftPressed := True;
      Exit;
    end else begin
      if Msg.wParam = VK_TAB then
        if ActiveControl = Edit1 then begin
          if shiftPressed then Label1.Caption := 'BACKTAB!'
          else Label1.Caption := 'TAB!';
```

```
          Handled := True;
        end else
          Label1.Caption := '';
        shiftPressed := false;
    end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Application.OnMessage := AppMessage;
end;
end.
```

[News Group]

Переключение языка

Решение 1

Для переключения языка применяется вызов LoadKeyboardLayout:

```
var
  russian, latin: HKL;
  ...
  russian := LoadKeyboardLayout('00000419', 0);
  latin := LoadKeyboardLayout('00000409', 0);
```

Где-то в программе:

```
SetActiveKeyboardLayout(russian);
```

[Nikolaev Igor]

Примечание

Для переключения надо использовать функцию API ActivateKeyboardLayout(russian, 0)*, где* russian *– см. выше. Для вызова* SetActiveKeyboardLayout(russian) *необходимо определить такую функцию и передать ей в качестве параметра* russian*. Второй параметр –* 0 *(можно не передавать).*

Решение 2

Предыдущее решение является очень простым, но не совсем корректным, т. к в нем задаются конкретные значения раскладок, а именно Русская и США (международная), в то время как на машине пользователя может быть установлена раскладка (например) США (101 клавиша) или любая другая. А как быть, если раскладок не две, а больше? Для этого и предназначен компонент LangTrigger.

```
{*******************************************************************************}
{            LanguageTrigger (With overpatching and add-ins)                    }
{            Переключатель раскладки клавиатуры (исправленный и дополненный)   }
{ Version:   1.2                                                                }
```

```
{ Co-Author: Michael Chumak, Rybinsk, Yaroslavl reg., Russia                  }
{            Михаил Чумак, Рыбинск, Ярославская обл., Россия                 }
{ E-Mail:    chuka@mail.ru, chuka@chat.ru                                     }
{ Modified:  13.09.2000                                                       }
{ Legal:     Chuka Rec.Ltd © '72-'00                                          }
{ License:   компонент и исходный код распространяются свободно, но с условием }
{            уведомления соавтора (т. е. меня) при внесении изменений.        }
{            Автору я сколько ни писал, письма не доходят... :(               }
{*****************************************************************************}
{ Краткое описание:                                                           }
{  - поместите компонент на форму;                                            }
{  - поместите на форму TToolbar и создайте новую кнопку TToolButton;         }
{  - в TLanguageTrigger в свойстве ToolButton выберите данную кнопку;         }
{  - на кнопке отображается текущая раскладка клавиатуры в соответствии с тем, }
{    как она хранится в системе, т.е: русская - RUS;                          }
{                                     США - ENU;                              }
{                                     английская - ENG и т.п.                 }
{  - при щелчке по кнопке или при смене раскладки с помощью Alt-Shift         }
{    (или Ctrl-Shift) меняются текущая раскладка и изображение на кнопке;     }
{  - если для свойства Style кнопки установить значение DropDown, то          }
{    в выпадающем меню будутот ображаться значки и названия раскладок         }
{    (это актуально, если их много).  }
{*****************************************************************************}
//*************************************************************//
// Platforms : Windows 9X, NT
// Delphi : D4, D5
// Author :Ahmed Ammar 8/2/2000
// E-mail Ahmed_Ammar@Hotmail.com
// This unit is the implementation of TLanguageTrigger component
// About :Easeful component to switch between local keyboard languages,
//        You need only ToolBar and ToolButton with this component.
// Installation:
//        1 - put this file and LangTrigger.Res file in \Lib directory
//        2 - from Component|Instal component menu select this file and
//            compile it.
// Usage:
//        1 - from Samples put TLanguageTrigger component on form1.
//        2 - put TToolbar on form1 and create new TToolButton.
//        3 - set ToolButton property of TLanguageTrigger to point
//            to Toolbutton.
//        4- run your application and click on toolbutton and see
//           the changes on task bar, change keyboard language by
//           click Alt+Shift keys and see the changes on Icon on toolButton.
//
// License: this component and source code is free, but if you using it
// please E-mail me send any notes at Email:Ahmed_Ammar@hotmail.com
//*************************************************************//

unit LangTrigger;

interface
```

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, Registry, StdCtrls, Menus, ComCtrls;

type
  TLanguageTrigger = class;

  TLanguageEntry = class
  private
    FParent: TLanguageTrigger;
    FCountryName: String;
    FLanguageID: LongInt;
    FLanguageStr: String; // Добавлено (Is added)
    FISOCode: String;
  public
    constructor Create(aParent:TLanguageTrigger);
    destructor Destroy; override;
    property CountryName: String read FCountryName;
    property LanguageID: LongInt read FLanguageID;
    property LanguageStr: String read FLanguageStr; // Добавлено (Is added)
    property ISOCode: String read FISOCode;
  end;

  TLanguageTrigger = class(TComponent)
  private
    FAbout: String;
    FEntryList: TList;
    FDefaultLanguage: Integer;
    FImageList: TImageList;
    FPopupMenu: TPopupMenu;
    FToolButton: TToolButton;
    procedure DoNewPopupItem(Sender: TMenuItem);
    procedure DoButtonClick(Sender: TObject);
    procedure DoOnLanguageChange;
  protected
    function GetAbout: String;
    function GetCount: Integer;
    function Get(Index: Integer): TLanguageEntry;
    procedure Put(Index: Integer; Item: TLanguageEntry);
    function GetCountryNames(Index: Integer): String;
    function GetISOCodes(Index: Integer): String;
    function GetLanguageIDs(Index: Integer): LongInt;
    procedure InstallCallBack;
    procedure UnInstallCallBack;
    procedure DoLanguageChange(LangID: Word);
    procedure SetToolButton(aToolButton: TToolButton);
    procedure FillPopupMenu;
    procedure MakeImages;
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
```

```
      procedure ReadLocals;
      procedure Fill; // Добавлено (Is added)
      procedure DoCreate;
      procedure Add(aEntry: TLanguageEntry);
      procedure Delete(Index: Integer);
      procedure Clear;
      procedure Move(CurIndex, NewIndex: Integer);
      procedure Exchange(Index1, Index2: Integer);
      procedure SetAsDefault(Index: Integer);
      function IndexOf(Item: TLanguageEntry): Integer;
      function IndexByLangID(LangID: LongInt): Integer;
      property EntryList: TList read FEntryList;
      property ISOCodes[Index: Integer]: String read GetISOCodes;
      property CountryNames[Index: Integer]: String read GetCountryNames;
      property LanguageIDs[Index: Integer]: LongInt read GetLanguageIDs;
      property Count: Integer read GetCount;
      property Items[Index: Integer]: TLanguageEntry read Get write Put; default;
    published
      property DefaultLanguage: Integer read FDefaultLanguage write SetAsDefault;
      property About: string Read GetAbout write FAbout;
      property ToolButton: TToolButton read FToolButton write SetToolButton;
    end;

    function ShellProc(ncode:Integer; wParam:LongInt; lParam:LongInt):LResult; far;
stdcall;

  var
    hhk: HHOOK;
    dLanguageList: TLanguageTrigger;
    BH,BH2,BW: Integer; // Добавлено (Is added)

  procedure Register;

  implementation
  {$R LangTrigger.Res}

  function ShellProc(ncode:Integer; wParam:LongInt; lParam:LongInt):LResult;
  begin
    Result:= CallNextHookEx(hhk, nCode, wParam, lParam);
    if (ncode = HSHELL_LANGUAGE) and assigned(dLanguageList)
      then dLanguageList.DoLanguageChange(LOWORD(lParam));
  end;

  {TLanguageEntry}

  constructor TLanguageEntry.Create(aParent:TLanguageTrigger);
  begin
    FCountryName:= '';
    FParent:= aParent;
    FParent.Add(Self);
  end;
```

```
destructor TLanguageEntry.Destroy;
begin
//
end;

{TLanguageTrigger}

constructor TLanguageTrigger.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  FEntryList:= TList.Create;
  dLanguageList:= Self;
  FDefaultLanguage:= -1;
end;

destructor TLanguageTrigger.Destroy;
var
  i: Integer;
begin
  UnInstallCallBack;
  for i:=0 to Count-1 do
    TLanguageEntry(Items[i]).Destroy;
  Clear;
  FEntryList.Free;
  inherited Destroy;
end;

procedure TLanguageTrigger.DoNewPopupItem(Sender: TMenuItem);
begin
  (Sender as TMenuItem).OnClick:= DoButtonClick;
end;

procedure TLanguageTrigger.DoButtonClick(Sender :TObject);
begin
  if (Sender is TToolButton)
    then FToolButton.Tag:= (FToolButton.Tag + 1) mod Count;
  DefaultLanguage:= TComponent(Sender).Tag;
end;

procedure TLanguageTrigger.DoOnLanguageChange;
begin
  FToolButton.ImageIndex:= DefaultLanguage;
  FToolButton.Hint:= FPopupMenu.Items[DefaultLanguage].Caption;
  FToolButton.Tag:= DefaultLanguage;
end;

procedure TLanguageTrigger.SetToolButton(aToolButton :TToolButton);
begin
  if FToolButton <> aToolButton
    then begin
      FToolButton:= aToolButton;
```

```
      BH:= aToolButton.Height;
      BH2:= BH div 2;
      aToolButton.Width:= BH + BH2;
      BW:= aToolButton.Width;
      Fill;
      TToolBar(FToolButton.Parent).Images:= FImageList;
      if aToolButton.Style = tbsDropDown
        then FToolButton.DropdownMenu:= FPopupMenu;
      FToolButton.OnClick:= DoButtonClick;
    end;
end;

procedure TLanguageTrigger.Fill;
begin
  FImageList:= TImageList.Create(Self);
  FPopupMenu:= TPopupMenu.Create(Self);
  FImageList.Height:= BH-6;
  FImageList.Width:= BW-7;
  FPopupMenu.Images:= FImageList;
  if not (csDesigning in ComponentState) then DoCreate;
//  DoCreate;
end;

procedure TLanguageTrigger.DoCreate;
begin
  ReadLocals;
  InstallCallBack;
end;

procedure TLanguageTrigger.ReadLocals;
const
  lcISOLang      = LOCALE_NOUSEROVERRIDE or LOCALE_SABBREVLANGNAME;
  lcFullLangName = LOCALE_NOUSEROVERRIDE or LOCALE_SLANGUAGE;
var
  HWStr: string;
  lpList: array[0..63]of HKL;
  lpLCData: array [0..63] of Char;
  Count_1,i,HW,LW: Integer;
  NewEntry: TLanguageEntry;
begin
  Count_1:= GetKeyboardLayoutList(64, lpList);
  for i:=0 to Count_1-1 do
    begin
      NewEntry:= TLanguageEntry.Create(Self);
      HW:= HiWord(lpList[I]);
      LW:= LoWord(lpList[I]);
      if HW = LW
        then begin
          NewEntry.FLanguageStr:= '00000' + Format('%0x',[LW])
        end
        else begin
```

```
          HWStr:= Format('%0x',[HW]);
          HWStr[1]:= '0';
          NewEntry.FLanguageStr:= HWStr + '0' + Format('%0x',[LW]);
        end;
      NewEntry.FLanguageID:= LoWord(lpList[I]);
      GetLocaleInfo(NewEntry.LanguageID, lcFullLangName, @lpLCData, 64);
      NewEntry.FCountryName:= StrPas(lpLCData);
      GetLocaleInfo(NewEntry.LanguageID, lcISOLang, @lpLCData, 64);
      NewEntry.FISOCode:= StrPas(lpLCData);
    end;
  MakeImages;
  FillPopupMenu;
end;

procedure TLanguageTrigger.FillPopupMenu;
var
  i: Integer;
  NewItem: TMenuItem;
begin
  if Assigned(FPopupMenu) and not(csDesigning in ComponentState)
    then begin
      for i:=0 To Count-1 do
        begin
          NewItem:= TMenuItem.Create(Application);
          NewItem.Caption:= CountryNames[i];
          NewItem.Tag:= i;
          NewItem.ImageIndex:= i;
          FPopupMenu.Items.Add(NewItem);
          DoNewPopupItem(NewItem);
        end;
    end;
end;

procedure TLanguageTrigger.MakeImages;
var
  i: Integer;
  NewBmp: TBitmap;
  Pen: TPen;
  Brush: TBrush;
  Font: TFont;
  TextBounds: TRect;
begin
  if Assigned(FImageList) and not(csDesigning in ComponentState)
    then begin
      Pen:= TPen.Create;
      Brush:= TBrush.Create;
      font:= TFont.Create;
      Brush.Style:= bsSolid;
  //  Brush.Color:= clMaroon; // clActiveCaption;
      Brush.Color:= clNavy;
      Pen.Style:= psSolid;
```

```
        Pen.Color:= clRed;
        Pen.Width:= 1;
        Font.Color:= clWhite;
        Font.Size:= BH2-2;
        Font.Style:= [fsBold];
        Font.Name:= 'Arial';
        FImageList.AllocBy:= Count;
        for i:=0 to Count-1 do
          begin
            NewBmp:= TBitmap.Create;
            NewBmp.Height:= FImageList.Height;
            NewBmp.Width:= FImageList.Width;
            NewBmp.Canvas.Brush.Assign(Brush);
            NewBmp.Canvas.Pen.Assign(Pen);
            NewBmp.Canvas.Font.Assign(Font);
            NewBmp.Canvas.FillRect(NewBmp.Canvas.ClipRect);
            TextBounds:= NewBmp.Canvas.ClipRect;
            DrawText(NewBmp.Canvas.Handle, PChar(ISOCodes[i]), Length(ISOCodes[i]),
                     TextBounds, DT_CENTER or DT_VCENTER or DT_SINGLELINE);
            FImageList.Insert(i, NewBmp, nil);
          end;
        Pen.Free;
        Brush.Free;
        Font.Free;
      end;

end;

function TLanguageTrigger.GetAbout: String;
begin
//  Result:= 'Ahmed Ammar 8/2/2000';
  Result:= 'Переключение раскладок клавиатуры';
end;

function TLanguageTrigger.IndexByLangID(LangID: LongInt): Integer;
begin
  Result:= 0;
  while (Result < Count) and (Items[Result].LanguageID <> LangID) do
    Inc(Result);
  if Result = Count
    then Result:= -1;
end;

procedure TLanguageTrigger.SetAsDefault(Index:Integer);
begin
  if Index <> FDefaultLanguage
    then begin
      if (Index < Count) and (Index > -1)
        then begin
          FDefaultLanguage:= Index;
//         ActivateKeyboardLayout(Items[FDefaultLanguage].LanguageID, KLF_REORDER);
```

```
          LoadKeyboardLayout(PChar(Items[FDefaultLanguage].LanguageStr),
                             KLF_ACTIVATE);
        end;
    end;
end;

procedure TLanguageTrigger.InstallCallBack;
(*
var
  PC: array[0..$FFF] of char;
  ActiveWindowTitle: String;
  Wnd: hWnd;
begin
  {$IFDEF Win32}
  Wnd:= GetForegroundWindow;
  {$ELSE}
  Wnd:= GetActiveWindow;
  {$ENDIF}
  SendMessage(Wnd, wm_GetText, $FFF, LongInt(@PC));
  ActiveWindowTitle:= StrPas(PC);
*)

begin
  if (hhk = 0) and not (csDesigning in ComponentState)
    then hhk:= SetWindowsHookEx(WH_SHELL, @ShellProc,
                                GetWindowLong(Application.Handle, GWL_HINSTANCE),
//                                GetWindowLong(Wnd, GWL_ID),
                                GetCurrentThreadId);
end;

procedure TLanguageTrigger.UnInstallCallBack;
begin
  if hhk <> 0
    then UnhookWindowsHookEx(hhk);
end;

procedure TLanguageTrigger.DoLanguageChange(LangID: Word);
begin
  DefaultLanguage:= IndexByLangID(LangID);
  DoOnLanguageChange;
end;

function TLanguageTrigger.GetLanguageIDs(Index: Integer):LongInt;
begin
  Result:= Items[Index].FLanguageID;
end;

function TLanguageTrigger.GetCountryNames(Index: Integer):String;
begin
  Result:= Items[Index].FCountryName;
end;
```

```
function TLanguageTrigger.GetISOCodes(Index: Integer):String;
begin
  Result:= Items[Index].ISOCode;
end;

function TLanguageTrigger.GetCount:Integer;
begin
  Result:= FEntryList.Count;
end;

function TLanguageTrigger.Get(Index: Integer): TLanguageEntry;
begin
  Result:= TLanguageEntry(FEntryList.Items[Index]);
end;

procedure TLanguageTrigger.Put(Index: Integer; Item: TLanguageEntry);
begin
   FEntryList.Items[Index]:= Item;
end;
(*
constructor TLanguageTrigger.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  FEntryList:= TList.Create;
  dLanguageList:= Self;
  FDefaultLanguage:= -1;
  RegisterClass(TImageList);
  RegisterClass(TPopupMenu);
  FImageList:= TImageList.Create(AOwner);
  FPopupMenu:= TPopupMenu.Create(AOwner);
{
  RegisterClass(TImageList);
  RegisterClass(TPopupMenu);
  FImageList:= TImageList.Create(AOwner);
  FImageList.Height:= 22; //20;
  FImageList.Width:= 23; //30;
  FPopupMenu:= TPopupMenu.Create(AOwner);
  FPopupMenu.Images:= FImageList;
  if not (csDesigning in ComponentState)
    then DoCreate;
}
end;

destructor TLanguageTrigger.Destroy;
var
  i: Integer;
begin
  UnInstallCallBack;
  for i:=0 to Count-1 do
    TLanguageEntry(Items[i]).Destroy;
  Clear;
```

```
  FEntryList.Free;
  UnRegisterClass(TImageList);
  UnRegisterClass(TPopupMenu);
  inherited Destroy;
end;
*)

procedure TLanguageTrigger.Add(aEntry: TLanguageEntry);
begin
  FEntryList.Add(aEntry);
end;

procedure TLanguageTrigger.Delete(Index:Integer);
begin
  FEntryList.Delete(Index);
end;

procedure TLanguageTrigger.Clear;
begin
  FEntryList.Clear;
end;

procedure TLanguageTrigger.Move(CurIndex, NewIndex: Integer);
begin
  FEntryList.Move(CurIndex, NewIndex);
end;

procedure TLanguageTrigger.Exchange(Index1, Index2: Integer);
begin
  FEntryList.Exchange(Index1, Index2);
end;

function TLanguageTrigger.IndexOf(Item: TLanguageEntry): Integer;
begin
  Result:= FEntryList.IndexOf(Item);
end;

procedure Register;
begin
  RegisterComponents('Chuka', [TLanguageTrigger]);
end;

initialization
  dLanguageList:= nil;
  hhk:= 0;
finalization

end.
```

[Чумак Михаил]

Управление кнопкой Windows Пуск из приложения

Процедура, которая убирает или показывает кнопку Пуск (Start):

```
var
  hTaskBar, hButton: HWND;
begin
  hTaskBar := FindWindow('Shell_TrayWnd', nil);
  hButton := GetWindow(hTaskBar, GW_CHILD);
// Нажать кнопку "Пуск"
  SendMessage(hButton, WM_LBUTTONDOWN, MK_LBUTTON, LoWord(5) +
              HiWord(Screen.Height - 20));
// Убрать кнопку "Пуск"
  ShowWindow(hButton, SW_HIDE);
// Показать кнопку "Пуск"
  ShowWindow(hButton, SW_NORMAL);
end;
```

[News Group]

Имитация ввода с клавиатуры для приложений DOS

Как имитировать ввод с клавиатуры в программе, выполняющейся в окне DOS?

Решение

```
const
  ExtendedKeys: set of Byte =
    [VK_INSERT, VK_DELETE, VK_HOME, VK_END, VK_PRIOR, VK_NEXT,
     VK_LEFT, VK_UP, VK_RIGHT, VK_DOWN, VK_NUMLOCK];
procedure SimulateKeyDown(Key: byte);
var
  flags: DWORD;
begin
  if Key in ExtendedKeys then
    flags := KEYEVENTF_EXTENDEDKEY
  else
    flags := 0;
  keybd_event(Key, MapVirtualKey(Key, 0), flags, 0);
end;
procedure SimulateKeyUp(Key: byte);
var
  flags: DWORD;
begin
  if Key in ExtendedKeys then flags := KEYEVENTF_EXTENDEDKEY
  else flags := 0;
  keybd_event(Key, MapVirtualKey(Key, 0), KEYEVENTF_KEYUP or flags, 0);
end;

procedure SimulateKeystroke(Key: byte);
var
  flags: DWORD;
  scancode: BYTE;
```

```
begin
  if Key in ExtendedKeys then
    flags := KEYEVENTF_EXTENDEDKEY
  else
    flags := 0;
  scancode := MapVirtualKey(Key, 0);
  keybd_event(Key, scancode, flags, 0);
  keybd_event(Key, scancode, KEYEVENTF_KEYUP or flags, 0);
end;
```

[News Group]

«Замена» кнопок мыши

Как поменять кнопки мыши местами?

Используйте SwapMouseButton:

```
SwapMouseButton(true);
```

Поменять обратно:

```
SwapMouseButton(false);
```

[Nomadic]

Перехват событий мыши

При вызове из приложения функции SetHook и каждом нажатии левой кнопки мыши будет раздаваться сигнал – до тех пор, пока приложение не вызовет функцию UnHookHook. В действующем приложении возвращаемое функцией CallNextHookEx отрицательное значение свидетельствует об отсутствии манипуляций с мышью.

```
library Hookdemo;

uses
  Beeper;

exports
  SetHook index 1,
  UnHookHook index 2,
  HookProc index 3;

begin
  HookedAlready := False;
end.

unit Beeper;

interface
```

```
uses
  Windows, Messages;

function SetHook: Boolean; export;
function UnHookHook: Boolean; export;
function HookProc(Code: integer; wParam: Word; lParam: Longint): Longint; export;

var
  HookedAlready: Boolean;

implementation

var
  ourHook: HHook;

function HookProc(Code: integer; wParam: Word; lParam: Longint): Longint;
begin
  if (wParam = WM_LBUTTONDOWN) then MessageBeep(0);
  result := CallNextHookEx(ourHook, Code, wParam, lParam);
end;

function SetHook: Boolean;
begin
  if HookedAlready then exit;
  ourHook := SetWindowsHookEx(WH_MOUSE, @HookProc, HInstance, 0);
  HookedAlready := True;
end;

function UnHookHook: Boolean;
begin
  UnHookWindowsHookEx(ourHook);
  HookedAlready := False;
end;
end.
```

Мышь над формой

Как определить, что мышь находится над формой?

Используйте функцию GetCapture.

```
procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
  if GetCapture = 0 then SetCapture(Form1.Handle);
  if PtInRect(Rect(Form1.Left, Form1.Top, Form1.Left + Form1.Width,
              Form1.Top + Form1.Height), ClientToScreen(Point(x, y))) then
    Form1.Caption := 'Mouse is over form'
  else
    Form1.Caption := 'Mouse is outside of form';
end;
```

Подсветка компонента во время перемещения над ним указателя мыши

Хочу создать компонент, реагирующий на перемещение мыши над его областью, но не знаю, как это сделать.

Нужно обрабатывать сообщения CM_MOUSEENTER и CM_MOUSELEAVE примерно таким образом:

```
TYourObject = class(TAnyControl)
  ...
private
  FMouseInPos : Boolean;
  procedure CMMouseEnter(var AMsg: TMessage); message CM_MOUSEENTER;
  procedure CMMouseLeave(var AMsg: TMessage); message CM_MOUSELEAVE;
  ...
end;
implementation

procedure TYourObject.CMMouseEnter(var AMsg: TMessage);
begin
  FMouseInPos := True;
  Refresh;
end;

procedure TYourObject.CMMouseLeave(var AMsg: TMessage);
begin
  FMouseInPos := False;
  Refresh;
end;
```

Затем читать параметр FMouseInPos при прорисовке области компонента или использовать иное решение.

Выход указателя мыши за границы компонента

Как определить момент выхода указателя мыши за границы компонента?

В момент выхода курсора мыши за пределы кнопки возникает событие OnMouseMove у панели, на которой кнопка расположена.

```
procedure TForm1.Panel1MouseMove(Sender: TObject; Shift: TShiftState;
                                 X, Y: Integer);
begin
  if (X = 0) or (X = Panel1.Width) or (Y = 0) or (Y = Panel1.Width) then
    Screen.Cursor := crHSplit
  else
    Screen.Cursor := crDefault;
end;
```

Примечание

При работе этого примера вид курсора изменяется при позиционировании его над «покидаемым» компонентом.

Добавление события OnMouseLeave

Все потомки TComponent могут посылать сообщения CM_MOUSEENTER и CM_MOUSELEAVE во время пересечения курсора мыши границ компонента. Для того чтобы компоненты обладали реакцией на эти события, необходимо написать соответствующие обработчики.

```
...
procedure CMMouseEnter(var msg: TMessage); message CM_MOUSEENTER;
procedure CMMouseLeave(var msg: TMessage); message CM_MOUSELEAVE;
...

procedure MyComponent.CMMouseEnter(var msg: TMessage);
begin
  inherited;
  { действия на вход мыши в область компонента }
end;

procedure MyComponent.CMMouseLeave(var msg: TMessage);
begin
  inherited;
  { действия на выход мыши из области компонента }
end;
```

Дополнение

Нередко возникает ситуация, когда необходимо обработать два важных события для визуальных компонентов:

- MouseEnter — куросор мыши входит в пределы визуального компонента;
- MouseLeave — курсор мыши оставляет его пределы.

Известно, что все версии Delphi объявляют эти сообщения в виде CM_MouseEnter и CM_MouseLeave.

То есть все визуальные компоненты, порожденные от TControl, могут «отлавливать» эти события. Следующий пример показывает, как создать наследника от TLabel и добавить два необходимых обработчика событий OnMouseLeave и OnMouseEnter.

```
unit BS_Label;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;
type
  TBS_Label = class(TLabel)
  private
    FOnMouseLeave: TNotifyEvent;
    FOnMouseEnter: TNotifyEvent;
    procedure CMMouseEnter(var Message: TMessage); message CM_MOUSEENTER;
```

```
    procedure CMMouseLeave(var Message: TMessage); message CM_MOUSELEAVE;
  published
    property OnMouseLeave: TNotifyEvent read FOnMouseLeave write FOnMouseLeave;
    property OnMouseEnter: TNotifyEvent read FOnMouseEnter write FOnMouseEnter;
end;

procedure Register;
implementation

procedure Register;
begin
  RegisterComponents('Custom', [TBS_Label]);
end;

{ TBS_Label }

procedure TBS_Label.CMMouseEnter(var Message: TMessage);
begin
  if Assigned(FOnMouseEnter) then FOnMouseEnter(Self);
end;

procedure TBS_Label.CMMouseLeave(var Message: TMessage);
begin
  if Assigned(FOnMouseLeave) then FOnMouseLeave(Self);
end;

end.
```

[Briculski]

Определение и использование курсора

Сначала (поскольку многие попадаются в эту ловушку) убедитесь в том, что имя .RES-файла, в котором хранится ваш курсор, отличается от имени .RES-файла проекта, т. е., если проекту дано имя MyApp.DPR, то не сохраняйте новые ресурсы в файле MyApp.RES. Нужно создать отдельный .RES-файл с другим именем (например, MyApp01.RES) и включить его в проект, например, так:

```
implementation

{$R MyApp01.Res}
```

Нельзя назначить курсор свойству компонента Cursor или DragCursor из .RES-файла напрямую. Необходимо выполнить несколько промежуточных шагов. В каждом проекте Delphi определяет глобальный объект с именем Screen (тип TScreen), который, между прочим, определяет массив курсоров, называемый, как ни «странно», Cursors. Если щелкнуть по свойству Cursor/DragCursor в Инспекторе объектов, выпадающий список и есть список элементов указанного массива.

Для предустановленных курсоров Delphi использует элементы массива с индексами от –1 и ниже (т. е. только отрицательные числа), поэтому нумерацию собственных курсоров можно вести от нуля и выше.

Для начала определите константу, допустим, так:

```
const
  MyCursor = 1;
```

Далее необходимо загрузить курсор. Сделать это можно в обработчике события формы OnCreate:

```
Screen.Cursors[MyCursor] := LoadCursor(HInstance, 'MYCURSOR');
```

Затем просто установите в свойстве DragCursor любого элемента управления:

```
MyListbox.DragCursor := MyCursor;
```

Примечание

Имя курсора всегда следует писать в ВЕРХНЕМ регистре, как при вызове LoadCursor, *так и в его названии в* .RES-*файле.*

Использование анимированных курсоров

Пусть анимированный курсор содержится в файле mycursor.ani Его (курсор) можно создать с помощью программы Microsoft aniedit.exe.

Решение

```
...
const
  crMyCursor = 1;
procedure TForm1.FormCreate(Sender: TObject);
begin
// Загружаем курсор.
  Screen.Cursors[crMyCursor] := LoadCursorFromFile('c:\mystuff\mycursor.ani');
// Используем курсор на форме
  Cursor := crMyCursor;
end;
```

Управление MouseOver посредством Hint

Существует ли какой-либо способ зафиксировать момент попадания курсора в область компонента? А его выход оттуда? Мне необходимо убедиться в видимости элемента управления под курсором мыши и совершить над ним некоторые действия.

Можно приспособить для этих целей метод OnHint класса TApplication. Данный метод вызывается при перемещении мыши над компонентом и, обычно в ответ на это, отображается всплывающая подсказка, а заголовок компонента присваивается величине Application.Hint. Тем не менее, нельзя поль-

зоваться этим решением. Зато при наступлении события можно проверять величину Hint для определения того, над каким компонентом в данный момент находится курсор мыши.

Вот некоторые детали для усвоения идеи. На форме компоненты Button, Edit и Label, их свойства Hint установлены и создано несколько дубликатов этих компонентов. Далее добавлен компонент TRadioGroup, содержащий три элемента: Button, Edit и Label. Метод приложения OnHint реализован следующим образом:

```
procedure TForm1.AppOnHint(Sender: TObject);
begin
  with Application do
    if Hint = 'Button' then RadioGroup1.ItemIndex := 0
    else if Hint = 'Edit' then RadioGroup1.ItemIndex := 1
    else if Hint = 'Label' then RadioGroup1.ItemIndex := 2
    else RadioGroup1.ItemIndex := -1;
end;
```

Теперь, во время перемещения мыши над формой, соответствующий элемент RadioGroup показывает, над чем в данный момент времени находится курсор мыши.

Событие OnShowHint в данном контексте, по-моему мнению, должно подойти больше, поскольку HintControl уже доступен в записи THintInfo объекта TShowHintEvent. К тому же, с его помощью гораздо удобнее производить проверку типа и др. При желании можно использовать всплывающие подсказки по назначению.

[News Group]

Количество заданий на печать

Как определить количество заданий на печать?

Windows передает WM_SPOOLERSTATUS каждый раз, когда добавляется или удаляется задание на печать. В примере мы попытаемся перехватывать это сообщение.

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
  private
    procedure WM_SpoolerStatus(var Msg: TWMSPOOLERSTATUS);
              message WM_SPOOLERSTATUS;
end;
```

```
var
  Form1: TForm1;

implementation
{$R *.DFM}

procedure TForm1.WM_SpoolerStatus(var Msg: TWMSPOOLERSTATUS);
begin
  Label1.Caption := IntToStr(msg.JobsLeft) + ' Jobs currenly in spooler';
  msg.Result := 0;
end;

end.
```

Замена принтера по умолчанию

Как поменять принтер Windows по умолчанию?

Решение

```
procedure TForm1.Button1Click(Sender: TObject);
var
  WinIni: TIniFile;
  WinIniFileName: array[0..MAX_PATH] of char;
  s: array[0..64] of char;
begin
  GetWindowsDirectory(WinIniFileName, SizeOf(WinIniFileName));
  StrCat(WinIniFileName, '\win.ini');
  WinIni := TIniFile.Create(WinIniFileName);
  try
    WinIni.WriteString('windows', 'device', 'HP LaserJet Series II, HPPCL, LPT1:');
  finally
    WinIni.Free;
  end;
  StrCopy(S, 'windows');
  SendMessage(HWND_BROADCAST, WM_WININICHANGE, 0, LongInt(@S));
end;
```

Примечание

Необходимо добавить в uses *модуль* IniFiles. *Помните, что это «грубый» метод, корректирующий напрямую системный файл.*

Замена порта принтера

Как программно изменить порт, используемый принтером?

Решение

```
uses
  Printers;
```

```
{$IFNDEF WIN32}
  const MAX_PATH = 144;
{$ENDIF}

procedure TForm1.Button1Click(Sender: TObject);
var
  pDevice: pChar;
  pDriver: pChar;
  pPort: pChar;
  hDMode: THandle;
  PDMode: PDEVMODE;
begin
  if PrintDialog1.Execute then begin
    GetMem(pDevice, cchDeviceName);
    GetMem(pDriver, MAX_PATH);
    GetMem(pPort, MAX_PATH);
    Printer.GetPrinter(pDevice, pDriver, pPort, hDMode);
    Printer.SetPrinter(pDevice, PDriver, 'FILE:', hDMode);
    FreeMem(pDevice, cchDeviceName);
    FreeMem(pDriver, MAX_PATH);
    FreeMem(pPort, MAX_PATH);
    Printer.BeginDoc;
    Printer.Canvas.TextOut(100, 100, 'Delphi Is RAD!');
    Printer.EndDoc;
  end;
end;
```

AT-команды модема

Команда	Описание
A	Команда ответа (Answer Command)
Bn	Настройка связи (Communications Options)
D	Команда набора (Dial Command)
En	Команда выбора символа эха (Select Command Character Echo Option)
Hn	Управление Switchhook – эмуляция нажатия телефонного рычага (Control The Switchhook)
I0	Идентификация кода продукта (Identify The Product Code)
I2	Выполнение теста контрольной суммы ROM (Perform ROM Checksum Test)
I7	Номер версии (Version Number)

продолжение

Команда	Описание
Ln	Выбор уровня громкости динамика (Select Speaker Volume Level)
Mn	Функция выбора опций динамика (Select Speaker Function Option)
Nn	Выбор опций для установления связи (Select Negotiate Handshake Option)
On	Переход к онлайновым командам (Go Online Command)
P	Выбор метода пульсового набора (Select Pulse Dialing Method)
Qn	Выбор опции результирующего кода (Select Result Code Option)
Sn=	Запись в S-регистр (Write To An S-Register)
Sn?	Чтение S-регистра (Read An S-Register)
T	Выбор метода тонового набора (Select Tone Dialing Method)
Vn	Выбор опции формата ответа (Select Response Format Option)
Wn	Выбор расширенного результирующего кода (Select Extended Result Code)
Xn	Выбор опции модемного вызова (Select Call Progress Option)
Yn	Выбор опции бездействия для разъединения (Select Long Space Disconnect Option)
Zn	Выполнение мягкого сброса (Perform Soft Reset)
&An	Выбор роли автоответчика (Select Originate/Answer Role For Autoanswer)
&Cn	Выбор опции определения передаваемых данных (Select Data Carrier Detect Option)
&Dn	Выбор опции готовности терминала данных (Select Data Terminal Ready Option)
&F	Загрузка заводских установок (Load Factory Default Profile)
&Gn	Выбор опции защиты тонового набора (Select Guard Tone Option)
&Kn	Выбор опций потока ConTDol (Select Flow ConTDol Option)

продолжение

Команда	Описание
&Pn	Выбор параметров пульсового набора (Select Pulse Dialing Parameters)
&Qn	Выбор опций режима связи (Select Communications Mode Option)
&Rn	Выбор опций RTS/CTS (Select RTS/CTS Option)
&Sn	Выбор опций готовности передачи данных (Select Data Set Ready Option)
&T0	Тест завершения в процессе (Terminate Test In Process)
&T1	Инициирование локального аналога сетевой петли (Initiate Local Analog Loopback)
&T3	Выполнение локальной цифровой сетевой петли (Perform Local Digital Loopback)
&T4	Включение предоставления RDL-запросов (Enable Granting Of RDL Requests)
&T5	Запрет предоставления RDL-запросов (Deny Granting Of RDL Requests)
&T6	Инициирование удаленной цифровой сетевой петли (Initiate Remote Digital Loopback)
&T7	Иниицирование внутреннего теста RDL (Initiate RDL With Self Test)
&T8	Внутренний тест локальной сетевой петли (Local Loopback With Self Test)
&T19	Выполнение теста RTS/CTS кабеля (Perform RTS/CTS Cable Test)
&Un	Отмена Tdellis-кодирования (Disable TDellis Coding)
&V	Просмотр профилей конфигурации (View Configuration Profiles)
&Wn	Сохранение активного профиля (Store Active Profile)
&Xn	Выбор источника синхронизации времени TDansmit (Store Active Profile)
&Yn	Выбор сохранения профиля для аппаратного перезапуска (Select Stored Profile For Hard Reset)
&Zn=	Сохранение телефонного номера (Store Telephone Number)

продолжение

Команда	Описание
,	Пауза (Perform Pause)
=	Запись в S-регистр (Write To An S-Register)
?	Чтение S-регистра (Read An S-Register)
P	Выбор пульсового набора (Select Pulse Dialing)
T	Тоновый набор (Tone)

S-регистры модема

Регистр	Описание
S0	Звонок, на который необходимо ответить (Ring After Which To Answer)
S1	Количество звонков (Ring Count)
S2	Символ отмены (Hayes Escape Character)
S3	Символ перевода строки (Carriage Return Character)
S4	Символ пропуска строки (Line Feed Character)
S5	Символ пробела (Backspace Character)
S6	Ожидание перед вызовом (Wait Before Blind Dialing)
S7	Ожидание ответа (Wait For Carrier)
S8	Время паузы для запятой (Pause Time For Comma)
S9	Время восстановления (Carrier Recovery Time)
S10	Время задержки для поднятия трубки после потери соединения (Lost Carrier Hang Up Delay)
S11	Время DTMF-соединения (DTMF Dialing Speed)
S12	Время защиты отмены (Hayes Escape Guard Time)
S16	Выполнение теста (Test in Progress)

продолжение

Регистр	Описание
S18	Тест таймера модема (Modem Test Timer)
S19	Настройки автосинхронизации (AutoSync Options)
S25	Обнаружено изменение DTD (Detect DTD Change)
S26	Интервал задержки RTS для CTS (RTS To CTS Delay Interval)
S30	Неактивное время ожидания (Inactivity Timeout)
S31	Символ XON (XON Character)
S32	Символ XOFF (XON Character)
S36	Ошибка согласования TDeatment (Negotiation Failure TDeatment)
S37	Ускорение DCE-линии (Desired DCE Line Speed)
S38	Время ожидания снятия трубки (Hang-up Timeout)
S43	Текущая скорость линии (Current Line Speed)
S44	Техническая конструкция (Framing Technique)
S46	Выбор протокола/компрессии (Protocol/Compression Selection)
S48	Действие характеристики согласования (Feature Negotiation Action)
S49	Нижний предел буфера (Buffer Low Limit)
S50	Верхний предел буфера (Buffer High Limit)
S70	Максимальное число ReTDansmissions (Maximum Number of ReT-Dansmissions)
S73	Неактивное время ожидания (No Activity Timeout)
S82	Выбор прерывания (Break Selection)
S86	Код причины неудачной связи (Connection Failure Cause Code)
S91	Выбор уровня TDansmit коммутируемой линии (Select Dial-up Line TDansmit Level)

продолжение

Регистр	Описание
S95	Расширенный результат кода битовой карты (Extended Result Code Bit Map)
S97	Позднее время соединения – снятия трубки (V.32 Late Connecting Handshake Timing)
S105	Размер кадра (Frame Size)
S108	Селектор качества сигнала (Signal Quality Selector)
S109	Селектор скорости соединения (Carrier Speed Selector)
S110	Селектор V.32/V.32 bis (V.32/V.32 bis Selector)
S113	Тональный вызов ConTDol (Calling Tone ConTDol)
S121	Использование DTD (Use of DTD)
S141	Таймер фазы обнаружения (Detection Phase Timer)
S142	Онлайновый формат символов (Online Character Format)
S144	Выбор скорости автобода (Autobaud Speed Group Selection)

Примечание

Следует помнить, что список команд модемов постоянно расширяется, добавляются новые протоколы, новые регистры. Чтобы не отстать от жизни, необходимо читать инструкции к модемам, которые вы собираетесь использовать. Помните, что у некоторых моделей одного класса разных производителей могут быть «свои взгляды» на некоторые аспекты работы.

Список установленных модемов

Как получить список установленных модемов в Windows 9x?

Решение

```
unit PortInfo;

interface

uses
  Windows, SysUtils, Classes, Registry;
```

```
function EnumModems: TStrings;

implementation

function EnumModems: TStrings;
var
  R: TRegistry;
  s: ShortString;
  N: TStringList;
  i, j: integer;

begin
  Result := TStringList.Create;
  R := TRegistry.Create;

  try
    with R do begin
      RootKey := HKEY_LOCAL_MACHINE;
      if OpenKey('\System\CurrentControlSet\Services\Class\Modem', False) then
        if HasSubKeys then begin
          N := TStringList.Create;

          try
            GetKeyNames(N);
            for I := 0 to N.Count - 1 do begin
              CloseKey;
              OpenKey('\System\CurrentControlSet\Services\Class\Modem', False);
              OpenKey(N[i], False);
              s := ReadString('AttachedTo');
              for j := 1 to 4 do
                if Pos(Chr(j + Ord('0')), s) > 0 then Break;
              Result.AddObject(ReadString('DriverDesc'), TObject(j));
              CloseKey;
            end;

          finally
            N.Free;
          end;
        end;
    end;

  finally
    R.Free;
  end;
end;

end.
```

[Nomadic]

Определяем состояние модема

Как узнать состояние модема под Win32?

Решение

```
procedure TForm1.Button1Click(Sender: TObject);
var
  CommPort: string;
  hCommFile: THandle;
  ModemStat: DWord;
begin
  CommPort := 'COM2';
{ Open the comm port }
  hCommFile := CreateFile(PChar(CommPort), GENERIC_READ, 0, nil, OPEN_EXISTING,
                          FILE_ATTRIBUTE_NORMAL, 0);
  if hCommFile = INVALID_HANDLE_VALUE then begin
    ShowMessage('Unable to open '+ CommPort);
    exit;
  end;
{ Get the Modem Status }
  if GetCommModemStatus(hCommFile, ModemStat) <> false then begin
    if ModemStat and MS_CTS_ON <> 0 then
      ShowMessage('The CTS (clear-to-send) is on.');
    if ModemStat and MS_DSR_ON <> 0 then
      ShowMessage('The DSR (data-set-ready) is on.');
    if ModemStat and MS_RING_ON <> 0then
      ShowMessage('The ring indicator is on.');
    if ModemStat and MS_RLSD_ON <> 0 then
      ShowMessage('The RLSD (receive-line-signal-detect) is on.');
  end;
{ Close the comm port }
  CloseHandle(hCommFile);
end;
```

Набор номера модемом

Как набрать номер модемом?

Решение

```
var
  hCommFile: THandle;

procedure TForm1.Button1Click(Sender: TObject);
var
  PhoneNumber: string;
  CommPort: string;
  NumberWritten: PDWORD;
begin
  PhoneNumber := 'ATDT 1-555-555-1212' + #13 + #10;
  CommPort := 'COM2';
```

```
{ Open the comm port }
  hCommFile := CreateFile(PChar(CommPort), GENERIC_WRITE, 0, nil, OPEN_EXISTING,
                          FILE_ATTRIBUTE_NORMAL, 0);
  if hCommFile = INVALID_HANDLE_VALUE then begin
    ShowMessage('Unable to open ' + CommPort);
    exit;
  end;
{ Dial the phone }
  New(NumberWritten);
  NumberWritten^ := 0;
  if WriteFile(hCommFile, PChar(PhoneNumber)^, Length(PhoneNumber),
               NumberWritten^, nil) = false then
    ShowMessage('Unable to write to ' + CommPort);
  Dispose(NumberWritten);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
{ Close the port }
  CloseHandle(hCommFile);
end;
```

Использование TAPI

Как использовать TAPI *для набора телефона, чтобы поговорить голосом?*

Решение

```
{ TAPI Errors }
  const TAPIERR_CONNECTED = 0;
  const TAPIERR_DROPPED = -1;
  const TAPIERR_NOREQUESTRECIPIENT = -2;
  const TAPIERR_REQUESTQUEUEFULL = -3;
  const TAPIERR_INVALDESTADDRESS = -4;
  const TAPIERR_INVALWINDOWHANDLE = -5;
  const TAPIERR_INVALDEVICECLASS = -6;
  const TAPIERR_INVALDEVICEID = -7;
  const TAPIERR_DEVICECLASSUNAVAIL = -8;
  const TAPIERR_DEVICEIDUNAVAIL = -9;
  const TAPIERR_DEVICEINUSE = -10;
  const TAPIERR_DESTBUSY = -11;
  const TAPIERR_DESTNOANSWER = -12;
  const TAPIERR_DESTUNAVAIL = -13;
  const TAPIERR_UNKNOWNWINHANDLE = -14;
  const TAPIERR_UNKNOWNREQUESTID = -15;
  const TAPIERR_REQUESTFAILED = -16;
  const TAPIERR_REQUESTCANCELLED = -17;
  const TAPIERR_INVALPOINTER = -18;

{ TAPI size constants }
  const TAPIMAXDESTADDRESSSIZE = 80;
```

```
  const TAPIMAXAPPNAMESIZE = 40;
  const TAPIMAXCALLEDPARTYSIZE = 40;
  const TAPIMAXCOMMENTSIZE = 80;
  const TAPIMAXDEVICECLASSSIZE = 40;
  const TAPIMAXDEVICEIDSIZE = 40;

function tapiRequestMakeCallA(DestAddress: PAnsiChar; AppName: PAnsiChar;
                              CalledParty: PAnsiChar; Comment: PAnsiChar): LongInt;
                              stdcall; external 'TAPI32.DLL';
function tapiRequestMakeCallW(DestAddress: PWideChar; AppName: PWideChar;
                              CalledParty: PWideChar; Comment: PWideChar): LongInt;
                              stdcall; external 'TAPI32.DLL';
function tapiRequestMakeCall(DestAddress: PChar; AppName: PChar;
                             CalledParty: PChar; Comment: PChar): LongInt;
                             stdcall; external 'TAPI32.DLL';

procedure TForm1.Button1Click(Sender: TObject);
var
  DestAddress: string;
  CalledParty: string;
  Comment: string;
begin
  DestAddress := '1-555-555-1212';
  CalledParty := 'Frank Borland';
  Comment := 'Calling Frank';
  tapiRequestMakeCall(pChar(DestAddress), PChar(Application.Title),
                      pChar(CalledParty), PChar(Comment));
end;

end.
```

Управление динамиком PC

Как «умертвить» PC Speaker?

Выключение динамика:

```
SyStemParametersInfo(SPI_SETBEEP, 0, nil, SPIF_UPDATEINIFILE);
```

Включение:

```
SyStemParametersInfo(SPI_SETBEEP, 1, nil, SPIF_UPDATEINIFILE);
```

[Nomadic]

8

Операционная система

Определение версии ОС

Как определить версию ОС?

Решение приведено на сайте http://www.swissdelphicenter.ch/en/.

Определение размера оперативной памяти

Как определить размер свободной оперативной памяти?

Решение

```
unit MemInfo;

interface
  procedure FreeMemory(Var lTotalMemory: LongInt; Var lFreeMemory: LongInt);

implementation

uses
  WinTypes, WinProcs, ToolHelp;

function Min(Number1, Number2: LongInt): LongInt;

{ Возвращаем минимум из Number1 & Number2 }
begin
  if (Number1 <= Number2) then Min := Number1
  else Min := Number2;
end;
```

```
procedure FreeMemory(Var lTotalMemory: LongInt; Var lFreeMemory: LongInt);
{ Вычисляем и возвращаем сумму полной и свободной памяти (Total & Free Memory)
  в байтах (ie. divide each by 1024 of Kilobytes) NB: Total Memory будет равна 0,
  если Windows запущена в стандартном режиме (Standard Mode), поскольку в этом
  случае Total Memory не может быть определена.}
var
  lWinFlags: LongInt;
  mmiMemManInfo: TMemManInfo;
begin
{ Инициализируем переменные }
  lTotalMemory := 0;
  lFreeMemory := 0;
  lWinFlags := GetWinFlags;
  if (0 <> (lWinFlags and WF_ENHANCED)) then begin
{ Инициализируем структуру MemManInfo }
    mmiMemManInfo.dwSize := SizeOf(TMemManInfo);
    mmiMemManInfo.dwLargestFreeBlock := 0;
    mmiMemManInfo.dwMaxPagesAvailable := 0;
    mmiMemManInfo.dwMaxPagesLockable := 0;
    mmiMemManInfo.dwTotalLinearSpace := 0;
    mmiMemManInfo.dwTotalUnlockedPages := 0;
    mmiMemManInfo.dwFreePages := 0;
    mmiMemManInfo.dwTotalPages := 0;
    mmiMemManInfo.dwFreeLinearSpace := 0;
    mmiMemManInfo.dwSwapFilePages := 0;
    mmiMemManInfo.wPageSize := 0;
    MemManInfo(@mmiMemManInfo);          { Получение информации диспетчера памяти }
{ Вычисление Total Memory }
    lTotalMemory := (Min(mmiMemManInfo.dwTotalLinearSpace,
                     mmiMemManInfo.dwTotalPages + mmiMemManInfo.dwSwapFilePages) *
                     mmiMemManInfo.wPageSize);
{ Вычисляем Free Memory }
    lFreeMemory := GetFreeSpace(0);
  end else begin
{ Полная память = 0 }
    lTotalMemory := 0;
{ Вычисляем Free Memory }
    lFreeMemory := GetFreeSpace(0);
  end;
end;

end.
```

Примечание

Пример работает только для Win16 (был написан в Delphi 1). При адаптации под Win32 необходимо изменить ToolHelp *в секции* uses *на* TlHelp32 *и вызовы используемых функций, отсутствующих в Win32.*

Откуда инсталлировалась Windows

Как узнать, откуда инсталлировалась Windows?

Решение

```
procedure TForm1.Button1Click(Sender: TObject);
var
  reg: TRegistry;
begin
  reg := TRegistry.Create;
  reg.RootKey := HKEY_LOCAL_MACHINE;
  reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\SETUP', false);
  ShowMessage(reg.ReadString('SourcePath'));
  reg.CloseKey;
  reg.Free;
end;
```

Примечание

Добавьте в uses *модуль* Registry.

Имя программы и расширение

Как получить имя программы, с которой ассоциировано то или иное расширение?

Решение

```
uses
{$IFDEF WIN32}
  Registry; { We will get it from the registry }
{$ELSE}
  IniFiles; { We will get it from the win.ini file }
{$ENDIF}

{$IFNDEF WIN32}
const MAX_PATH = 144;
{$ENDIF}

function GetProgramAssociation(Ext: string): string;
var
{$IFDEF WIN32}
  reg: TRegistry;
  s: string;
{$ELSE}
  WinIni: TIniFile;
  WinIniFileName: array[0..MAX_PATH] of char;
  s: string;
{$ENDIF}
begin
{$IFDEF WIN32}
  s := '';
```

```
  reg := TRegistry.Create;
  reg.RootKey := HKEY_CLASSES_ROOT;
  if reg.OpenKey('.' + ext + '\shell\open\command', false) <> false then begin
{ The open command has been found }
    s := reg.ReadString('');
    reg.CloseKey;
  end else begin
{ perhaps thier is a system file pointer }
    if reg.OpenKey('.' + ext, false) <> false then begin
      s := reg.ReadString('');
      reg.CloseKey;
      if s <> '' then begin
{ A system file pointer was found }
        if reg.OpenKey(s + '\shell\open\command', false) <> false then
{ The open command has been found }
          s := reg.ReadString('');
        reg.CloseKey;
      end;
    end;
  end;
{ Delete any command line, quotes and spaces }
  if Pos('%', s) > 0 then Delete(s, Pos('%', s), length(s));
  if ((length(s) > 0) and (s[1] = '"')) then Delete(s, 1, 1);
  if ((length(s) > 0) and (s[length(s)] = '"')) then Delete(s, Length(s), 1);
  while ((length(s) > 0) and ((s[length(s)] = #32) or (s[length(s)] = '"'))) do
    Delete(s, Length(s), 1);
{$ELSE}
  GetWindowsDirectory(WinIniFileName, sizeof(WinIniFileName));
  StrCat(WinIniFileName, '\win.ini');
  WinIni := TIniFile.Create(WinIniFileName);
  s := WinIni.ReadString('Extensions', ext, '');
  WinIni.Free;
{ Delete any command line }
  if Pos(' ^', s) > 0 then Delete(s, Pos(' ^', s), length(s));
{$ENDIF}
  result := s;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  ShowMessage(GetProgramAssociation('gif'));
end;
```

Изменения в реестре

После того как я вношу изменения в реестр, некоторые программы не видят их до перезагрузки, что можно сделать?

Необходимо послать всем окнам сообщение `WM_WININICHANGE` с указанием полного мени измененного ключа.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  SendMessage(HWND_BROADCAST, WM_WININICHANGE, 0,
              LongInt(PChar('RegistrySection')));
end;
```

Загрузка приложения при запуске Windows

Как сделать, чтобы приложение загружалось при каждой загрузке Windows?

В данном случае необходимо следить за соответствующими записями в реестре. Пример для Win16 и Win32:

```
uses
  Registry, {For Win32}
  IniFiles; {For Win16}

{$IFNDEF WIN32}
const MAX_PATH = 144;
{$ENDIF}

{ For Win32 }
procedure TForm1.Button1Click(Sender: TObject);
var
  reg: TRegistry;
begin
  reg := TRegistry.Create;
  reg.RootKey := HKEY_LOCAL_MACHINE;
  reg.LazyWrite := false;
  reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\Run', false);
  reg.WriteString('My App', Application.ExeName);
  reg.CloseKey;
  reg.free;
end;

{ For Win16 }
procedure TForm1.Button2Click(Sender: TObject);
var
  WinIni: TIniFile;
  WinIniFileName: array[0..MAX_PATH] of char;
  s: string;
begin
  GetWindowsDirectory(WinIniFileName, sizeof(WinIniFileName));
  StrCat(WinIniFileName, '\win.ini');
  WinIni := TIniFile.Create(WinIniFileName);
  s := WinIni.ReadString('windows', 'run', '');
  if s = '' then s := Application.ExeName else
  s := s + ';' + Application.ExeName;
  WinIni.WriteString('windows', 'run', s);
  WinIni.Free;
end;
```

Панель управления

Как поместить приложение Delphi в Панель управления?

Для Delphi версии 3 и выше добавьте модуль Cpl в файл проекта.

Решение

```
library Project1;

uses
  Cpl, Windows, Forms,
  Unit1 in 'Unit1.pas' {Form1};
{$R *.RES}

procedure ExecuteApp;
begin
  Application.Initialize;
  Application.CreateForm(TForm1,Form1);
  Application.Run;
end;

{ Callback-функция для экспорта в Панель управления }
function CPlApplet(hwndCPl: THandle; uMsg: DWORD;
                   lParam1, lParam2: LongInt): LongInt; stdcall;
var
  NewCplInfo: PNewCplInfo;
begin
  Result := 0;
  case uMsg of
        CPL_INIT: Result := 1;     { Инициализация должна возвращать True. }
    CPL_GETCOUNT: Result := 1;     { Число апплетов }
  CPL_NEWINQUIRE: { Помещаем информацию об этом апплете в Панель управления. }
                  begin
                    NewCplInfo := PNewCplInfo(lParam2);
                    with NewCplInfo^ do begin
                      dwSize := SizeOf(TNewCplInfo);
                      dwFlags := 0;
                      dwHelpContext := 0;
                      lData := 0;
                  { Иконка для отображения на Панели Управления. }
                      hIcon := LoadIcon(HInstance, 'MAINICON');
                  { Имя апплета }
                      szName := 'Project1';
                  { Описание этого апплета. }
                      szInfo := 'Это тестовый апплет.';
                      szHelpFile := '';
                    end;
                  end;

      CPL_DBLCLK: { Выполнение апплета. }
                  ExecuteApp;
```

```
    else
      Result := 0;
    end;
  end;

  exports CPlApplet;    { Экспортирование функции CplApplet }

  begin
  end.
```

Для использования апплета измените его расширение с .dll на .cpl и поместите в системную директорию.

Апплет будет добавлен к списку уже существующих (Display, Fonts, Mouse, System и др.).

Панель управления из приложения

Как открывать Панель управления и ее компоненты из приложения?

Окно Свойства: Экран, Фон:

```
WinExec('C:\WINDOWS\CONTROL.EXE desk.cpl', SW_SHOW);
```

или

```
WinExec('C:\WINDOWS\CONTROL.EXE desk.cpl,,0', SW_SHOW);
```

Окно Свойства: Экран, Заставка (ScreenSaver – хранитель экрана):

```
WinExec('C:\WINDOWS\CONTROL.EXE desk.cpl,,1', SW_SHOW);
```

Окно Свойства: Экран, Оформление:

```
WinExec('C:\WINDOWS\CONTROL.EXE desk.cpl,,2', SW_SHOW);
```

Окно Свойства: Экран, Настройка:

```
WinExec('C:\WINDOWS\CONTROL.EXE desk.cpl,,3', SW_SHOW);
```

Установка времени:

```
WinExec('C:\WINDOWS\CONTROL.EXE TIMEDATE.CPL', sw_ShowNormal);
```

Свойства мыши:

```
WinExec('C:\WINDOWS\CONTROL.EXE MOUSE', sw_ShowNormal);
```

Принтеры:

```
WinExec('C:\WINDOWS\CONTROL.EXE PRINTERS', sw_ShowNormal);
```

Определение имени Группы Запуска

В различных локализациях Windows папка \StartUp *называется по-разному. Как определить ее текущее имя?*

Следует обратиться к реестру:

```
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion
                 \Explorer\Shell Folders\ Startup
```

Путь/имя папки My Computer

Материал довольно сложный, и если это не актуально для вашего приложения, то не тратьте драгоценное время, блуждая в дебрях сложной и запутанной информации. Тем не менее, заинтересованных направим на путь истинный.

Операционная система Windows основывается на оболочке, которая использует виртуальные папки, такие, как my computer (Мой компьютер), desktop (Рабочий Стол) и recycle bin (Корзина). Некоторые из них являются частью физической файловой системы. Другими словами, им соответствует реальный каталог в файловой системе. Это относится, например, к системным папкам desktop и recycle bin. Данные каталоги могут быть использованы как InitialDir в TOpenDialog, но сначала необходимо получить их физическое месторасположение, которое может не совпадать на разных компьютерах. Их реальное месторасположение на локальном диске можно выяснить посредством некоторых специальных вызовов API (см. пример ниже). Другие папки, типа my computer и printers не являются частью файловой системы, они чисто виртуальные. Обращаю ваше внимание на то, что такие папки можно использовать в TOpenDialog, но никак не в InitialDir.

Виртуальные папки (если упростить) имеют тип SHITEMID (идентификатор элемента). Доступ к ним можно получить, используя указатель на элемент списка идентификаторов (Pointers to Item Identifiers List, PIDL). Для того чтобы получить PIDL специальной папки, следует обратиться к функции SHGetSpecialFolder. Физическое месторасположение соответствующей директории можно получить, передавая PIDL в качестве входного параметра функции GetPathFromIDList. Если папка является частью файловой системы, функция возвращает путь к ней в виде строки (которая впоследствии может использоваться как InitialDir). Но если предполагается применение OpenDialog только с виртуальными папками (например, с my computer), то в принципе надо использовать PIDL как InitialDir, но это работать не будет. Возможно, дело в том, что TOpenDialog использует PIDLs только для просмотра, а для InitialDir требуются только реальные (физические) каталоги.

Пример показывает, как получить путь к recent documents (последние документы) и использовать его в качестве InitialDir:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  PIDL: PItemIDList;
```

```
  Path: LPSTR;
const
  CSIDL_RECENT = $0008;
begin
  Path := StrAlloc(MAX_PATH);
  SHGetSpecialFolderLocation(Handle, CSIDL_RECENT, PIDL);
  if SHGetPathFromIDList(PIDL, Path) then begin
// возвращает False, если папка не является частью файловой системы
    OpenDialog1.InitialDir := Path;
    OpenDialog1.Execute;
  end;
  StrDispose(Path);
end;
```

Необходимо создать класс-оболочку для этих вызовов API. Они располагаются в shell32.dll. Наилучший совет, который можно дать тем, кто будет изучать этот вопрос – «копните» поглубже файл ShlObj.h. Даже если вы не программируете на C, то почерпнете оттуда немало ценной информации.

Вот некоторые константы, которые вам могут понадобиться:

```
CSIDL_DESKTOP = $0000;
CSIDL_PROGRAMS = $0002;
CSIDL_CONTROLS = $0003;
CSIDL_PRINTERS = $0004;
CSIDL_PERSONAL = $0005;
CSIDL_STARTUP = $0007;
CSIDL_RECENT = $0008;
CSIDL_SENDTO = $0009;
CSIDL_BITBUCKET = $000a;
CSIDL_STARTMENU = $000b;
CSIDL_DESKTOPDIRECTORY = $0010;
CSIDL_DRIVES = $0011;                    // Мой компьютер
CSIDL_NETWORK = $0012;
CSIDL_NETHOOD = $0013;
CSIDL_FONTS = $0014;
CSIDL_TEMPLATES = $0015;
```

Примечание

Необходимо добавить в секцию uses *модуль* ShlObj*, а «копать» легче в* SHLOBJ.PAS. *Константы, приведенные выше, уже определены в этом модуле (кто не верит – уберите раздел* const *с определением константы* CSIDL_RECENT*).*

Вызов стандартного системного окна О программе

Как показать стандартное окно О программе*?*

Решение

```
{ Не забудьте поместить ShellAPI в uses }
procedure ShowAbout;
```

```
begin
  ShellAbout(Form1.Handle, 'Напиши здесь название программы', 'Авторские права на
            программу' + #13#10 + '(можно в две строки)', Application.Icon.Handle);
end;
```

[News Group]

Замена обоев на Рабочем столе

Как программно заменить обои на Рабочем столе?

Решение

```
uses
  ComObj, ShlObj;

procedure ChangeActiveWallpaper;
const
  CLSID_ActiveDesktop: TGUID = '{75048700-EF1F-11D0-9888-006097DEACF9}';
var
  ActiveDesktop: IActiveDesktop;
begin
  ActiveDesktop := CreateComObject(CLSID_ActiveDesktop) as IActiveDesktop;
  ActiveDesktop.SetWallpaper('c:\windows\forest.bmp', 0);
  ActiveDesktop.ApplyChanges(AD_APPLY_ALL or AD_APPLY_FORCE);
end;
```

[Рыбант Владимир]

Управление хранителем экрана

Как отключить хранитель экрана?

Решения

```
procedure TForm1.Button1Click(Sender: TObject);
begin
{ Turn it off }
  SystemParametersInfo(SPI_SETSCREENSAVEACTIVE, 0, nil, 0);
{ Turn it on }
  SystemParametersInfo(SPI_SETSCREENSAVEACTIVE, 1, nil, 0);
end;
```

Пример показывает возможность включения, одновременно проверяя доступность хранителя экрана.

```
function TurnScreenSaverOn: boolean;
var
  b: boolean;
begin
  Result := false;
  if SystemParametersInfo(SPI_GETSCREENSAVEACTIVE, 0, @b, 0) <> true then exit;
```

```
  if not b then exit;
  PostMessage(GetDesktopWindow, WM_SYSCOMMAND, SC_SCREENSAVE, 0);
  Result := true;
end;
```

Окно свойств компьютера из приложения

Как вывести окно свойств компьютера?

Решение

```
WinExec('C:\WINDOWS\CONTROL.EXE SYSDM.CPL', sw_ShowNormal);
```

Очистка Корзины (Recycle Bin)

Есть функция SHEmptyRecycleBin (в SHELL32.DLL), но она не документирована.

Кнопки в панели задач Windows 9.x

В Delphi 2 или 3 требуется создать кнопку в панели задач так, как это делает PowerDesk 2.0 Toolbar.

Решение

```
// Это необходимо объявить в секции public в верхней части вашего pas-файла
procedure IconCallBackMessage(var Mess: TMessage); message WM_USER + 100;
procedure TForm1.FormCreate(Sender: TObject);
var
  nid: TNotifyIconData;
begin
  with nid do begin
    cbSize := SizeOf(TNotifyIconData);
    Wnd := Form1.Handle;
    uID := 1;
    uFlags := NIF_ICON or NIF_MESSAGE or NIF_TIP;
    uCallbackMessage := WM_USER + 100;
    hIcon := Application.Icon.Handle;
    szTip := 'Текст всплывающей подсказки';
  end;
  Shell_NotifyIcon(NIM_ADD, @nid);
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
var
  nid: TNotifyIconData;
begin
  with nid do begin
    cbSize := SizeOf(TNotifyIconData);
    Wnd := Form1.Handle;
    uID := 1;
    uFlags := NIF_ICON or NIF_MESSAGE or NIF_TIP;
```

```
    uCallbackMessage := WM_USER + 100;
    hIcon := Application.Icon.Handle;
    szTip := 'Текст всплывающей подсказки';
// Все, что указано выше, не является обязательным
  end;
  Shell_NotifyIcon(NIM_DELETE, @nid);
end;

procedure TForm1.IconCallBackMessage(var Mess: TMessage);
var
  sEventLog: String;
begin
  case Mess.lParam of
// Поместите сюда все что хотите. Например, вызов контекстного меню при
// нажатии правой кнопки мыши.
    WM_LBUTTONDBLCLK: sEventLog := 'Двойной щелчок левой кнопкой';
      WM_LBUTTONDOWN: sEventLog := 'Нажатие левой кнопки мыши';
        WM_LBUTTONUP: sEventLog := 'Отжатие левой кнопки мыши';
    WM_MBUTTONDBLCLK: sEventLog := 'Двойной щелчок средней кнопкой';
      WM_MBUTTONDOWN: sEventLog := 'Нажатие средней кнопки мыши';
        WM_MBUTTONUP: sEventLog := 'Отжатие средней кнопки мыши';
        WM_MOUSEMOVE: sEventLog := 'Перемещение мыши';
       WM_MOUSEWHEEL: sEventLog := 'Вращение колесика мыши';
    WM_RBUTTONDBLCLK: sEventLog := 'Двойной щелчок правой кнопкой';
      WM_RBUTTONDOWN: s
        WM_RBUTTONUP: sEventLog := 'Отжатие правой кнопки мыши';
  end;
end;
```

Примечание

Добавьте в секцию uses *модуль* ShellAPI.

Замена изображения на кнопке Пуск

Как изменить изображение кнопки Пуск*?*

Решение

```
{ объявляем глобальные переменные }

var
  StartButton: hWnd;
  OldBitmap: THandle;
  NewImage: TPicture;

{ добавляем следующий код в событие формы OnCreate }
procedure TForm1.FormCreate(Sender: TObject);
begin
  NewImage := TPicture.Create;
  NewImage.LoadFromFile('C:\Windows\Треугольники.BMP');
  StartButton := FindWindowEx(FindWindow('Shell_TrayWnd', nil), 0, 'Button', nil);
```

```
  OldBitmap := SendMessage(StartButton, BM_SetImage, 0, NewImage.Bitmap.Handle);
end;

{ Событие OnDestroy }
procedure TForm1.FormDestroy(Sender: TObject);
begin
  SendMessage(StartButton, BM_SetImage, 0, OldBitmap);
  NewImage.Free;
end;
```

[Nikolaev Igor]

Управляем кнопкой Пуск

Как спрятать кнопку Пуск*?*

Пример, показывающий, как сделать это, а заодно и некоторые другие вещи:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Rgn: hRgn;
begin

{ Hide the start button }
  Rgn := CreateRectRgn(0, 0, 0, 0);
  SetWindowRgn(FindWindowEx(FindWindow('Shell_TrayWnd', nil),
               0, 'Button', nil), Rgn, true);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin

{ Turn the start button back on }
  SetWindowRgn(FindWindowEx(FindWindow('Shell_TrayWnd', nil),
               0, 'Button', nil), 0, true);
end;

procedure TForm1.Button3Click(Sender: TObject);
begin

{ Disable the start button }
  EnableWindow(FindWindowEx(FindWindow('Shell_TrayWnd', nil),
               0, 'Button', nil), false);
end;

procedure TForm1.Button4Click(Sender: TObject);
begin

{ Enable the start button }
  EnableWindow(FindWindowEx(FindWindow('Shell_TrayWnd', nil),
               0, 'Button', nil), true);
end;
```

Управляем пунктом меню Документы

Как добавить информацию в меню Пуск|Документы*?*

Решение

```
uses
  ShlOBJ;
procedure TForm1.Button1Click(Sender: TObject);
var
  s: string;
begin
  s := 'C:\Мои документы\risunok.tif';                  // добавляемый документ
  SHAddToRecentDocs(SHARD_PATH, pointer(s));
end;
```

Как очистить папку \Документы*?*

Используйте SHAddToRecentDocs() с параметром имени файла nil.

```
uses
  ShlOBJ;
procedure TForm1.Button1Click(Sender: TObject);
begin
  SHAddToRecentDocs(SHARD_PATH, nil);
end;
```

Примечание

Надо помнить, что в папку \Документы *записываются только файлы зарегистрированных типов (операционная система «знает», какому приложению их сопоставить).*

Поиск файла из приложения

Как запустить диалог поиска файла?

Решение

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  with TDDEClientConv.Create(Self) do begin
    ConnectMode := ddeManual;
    ServiceApplication := 'explorer.exe';
    SetLink('Folders', 'AppProperties');
    OpenLink;
    ExecuteMacro('[FindFolder(, C:\WINDOWS)]', False);
    CloseLink;
    Free;
  end;
end;
```

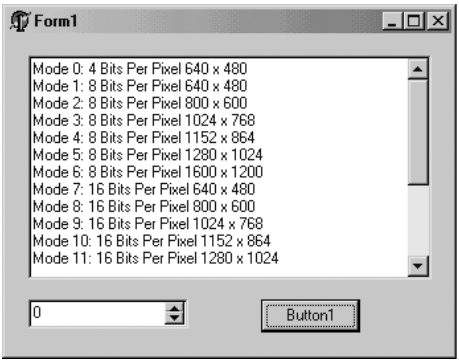
Примечание

Секция Uses *должна содержать ссылку на модуль* DdeMan.

Определение изменений на дисплее

Как определить появление каких-либо изменений на дисплее?

Решение




```
  type
    TForm1 =
class(TForm)
      Button1: TButton;
      ...
    private
      procedure
WMDisplayChange(var Mes-
sage: TMessage); message
WM_DISPLAYCHANGE;
  end;
  var
    Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.WMDisplayChange(var Message: TMessage);
begin
{ Do Something here }
  inherited;
end;
```

Управляем режимами дисплея

Как программно переключать режимы дисплея?

Нужно использовать `EnumDisplaySettings()` для получения полного списка доступных режимов. Затем `ChangeDisplaySettings()` – для изменения текущего

режима. Обратите внимание, что многие драйверы не меняют режим без перезагрузки.

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Spin;
type
  PDmArray = ^TDmArray;
  TDmArray = array[0..0] of TDeviceMode;
type
  TForm1 = class(TForm)
    Memo1: TMemo;
    SpinEdit1: TSpinEdit;
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    lpDmArray: PDmArray;
    NumModes: integer;
end;

var
  Form1: TForm1;

implementation
{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
var
```

```
  i: integer;
  MoreModes: boolean;
  dm: TDeviceMode;
begin
  Memo1.Lines.Clear;
  MoreModes := True;
  i := 0;
  while MoreModes do begin
    MoreModes := EnumDisplaySettings(nil, i, dm);
    Memo1.Lines.Add(' Mode ' + IntToStr(i) + ': ' + IntToStr(dm.dmBitsPerPel) +
                    ' Bits Per Pixel ' + IntToStr(dm.dmPelsWidth) + ' x ' +
                    IntToStr(dm.dmPelsHeight));
    Inc(i);
  end;
  NumModes := i;
  SpinEdit1.MinValue := 0;
  SpinEdit1.MaxValue := NumModes;
  GetMem(lpDmArray, SizeOf(TDeviceMode) * NumModes);
  FillChar(lpDmArray^, SizeOf(TDeviceMode) * NumModes, #0);
{$IFOPT R+}
{$DEFINE CKRANGE}
{$R-}
{$ENDIF}
  for i := 0 to (NumModes - 1) do
    EnumDisplaySettings(nil, i, lpDmArray[i]);
{$IFDEF CKRANGE}
{$UNDEF CKRANGE}
{$R+}
{$ENDIF}
end;
procedure TForm1.Button1Click(Sender: TObject);
var
  ReturnVal: LongInt;
begin
{$IFOPT R+}
{$DEFINE CKRANGE}
{$R-}
{$ENDIF}
  ReturnVal := ChangeDisplaySettings(lpDmArray[SpinEdit1.Value],
                                     CDS_UPDATEREGISTRY);
{$IFDEF CKRANGE}
{$UNDEF CKRANGE}
{$R+}
{$ENDIF}
  with Memo1.Lines do begin
    case ReturnVal of
        DISP_CHANGE_SUCCESSFUL:    Add('DISP_CHANGE_SUCCESSFUL');
        DISP_CHANGE_RESTART:       Add('DISP_CHANGE_RESTART');
        DISP_CHANGE_BADFLAGS:      Add('DISP_CHANGE_BADFLAGS');
        DISP_CHANGE_FAILED:        Add('DISP_CHANGE_FAILED');
        DISP_CHANGE_BADMODE:       Add('DISP_CHANGE_BADMODE');
```

```
          DISP_CHANGE_NOTUPDATED:    Add('DISP_CHANGE_NOTUPDATED');
      end;
    end;
  end;

  procedure TForm1.FormDestroy(Sender: TObject);
  begin
    FreeMem(lpDmArray, SizeOf(TDeviceMode) * NumModes);
  end;
  end.
```

Прячем Панель задач

Можно ли спрятать Панель задач при запуске своего приложения? Когда пользователь закрывает приложение, панель задач должна снова стать видимой.

Решение 1

Сначала объявим переменную типа HWND, в которой будем хранить дескриптор окна Панели задач Windows 95:

```
TForm1 = class(TForm)
...
private
  hTaskBar: HWND;
...
end;
```

В обработчике события OnCreate() главной формы напишем код, подобный этому:

```
hTaskBar := FindWindow('Shell_TrayWnd', nil);
ShowWindow(hTaskBar, SW_HIDE);
```

В обработчик события главной формы OnDestroy() необходимо вставить следующую строчку:

```
ShowWindow(hTaskBar, SW_SHOW);
```

Решение 2

```
procedure HideWin95TaskBar;
var
  WindowHandle: hWnd;
begin
{ Скрытие панели задач Windows 95 }
  WindowHandle := FindWindow('Shell_TrayWnd', '');
  if WindowHandle <> 0 then ShowWindow(WindowHandle, SW_HIDE);
end; { HideWin95TaskBar }

procedure ShowWin95TaskBar;
var
  WindowHandle: hWnd;
```

```
begin
{ Восстанавливаем видимость панели задач Windows 95 }
  WindowHandle := FindWindow('Shell_TrayWnd', '');
  if WindowHandle <> 0 then ShowWindow(WindowHandle, SW_RESTORE)
end; { ShowWin95TaskBar }
```

Пиктограмма приложения в Панели задач

Если вы не хотите, чтобы приложение было представлено пиктограммой в Панели задач, добавьте следующие строки в исходный код проекта:

```
Application.CreateHandle;
ShowWindow(Application.Handle, SW_HIDE);
Application.ShowMainForm := False;
```

При нормальном поведении TApplication создает дескриптор и показывает окно прежде, чем что-то начнет «происходить». Чтобы избежать этого, необходимо создать модуль, содержащий единственную строчку в секции initialization:

```
IsLibrary := True;
```

Поместить этот модуль первым в .DPR-файле в списке используемых модулей. Так мы обманываем TApplication. Оно «думает», что оно запущено из DLL, тем самым, изменяя свое обычное поведение.

[News Group]

Удаление пиктограммы из Панели задач

Решение

```
ShowWindow(Application.Handle, SW_HIDE);
```

Сохранение приложения в виде пиктограммы

Как после запуска сохранять все время приложение в виде пиктограммы?

- В свойствах формы вы должны установить для WindowState значение wsMinimized.
- В секции класса формы private поместите следующее объявление:

```
...
{ private declarations }
procedure WMQueryOpen(var Msg: TWMQueryOpen); message WM_QUERYOPEN;
```

- В секции implementation поместите следующую реализацию метода:

```
procedure TForm1.WMQueryOpen(var Msg: TWMQueryOpen);
begin
  Msg.Result := 0;
end;
```

Это все! Форма всегда будет пиктограммой.

Загрузка пиктограммы

Если пиктограмма хранится в компоненте Image (видимым или иным способом), напишите:

```
Application.Icon := Image1.Picture.Icon;
```

А если в файле ресурса, то:

```
Application.Icon.Handle := LoadIcon(hInstance, 'ICONNAME');
```

В любом случае, чтобы форсировать отображение пиктограммы, необходимо вызвать следующую функцию:

```
InvalidateRect(Application.Handle, NIL, True);
```

Пиктограмма, расположенная в .RES-файле, должна быть видима в .EXE-файле. Пиктограмма, расположенная в компоненте Image, в этом случае не видна.

[News Group]

Создание ярлыков

Решение

```
uses
  ShlObj, ComObj, ActiveX;
procedure CreateLink(const PathObj, PathLink, Desc, Param: string);

var
  IObject: IUnknown;
  SLink: IShellLink;
  PFile: IPersistFile;

begin
  IObject := CreateComObject(CLSID_ShellLink);
  SLink := IObject as IShellLink;
  PFile := IObject as IPersistFile;
  with SLink do begin
    SetArguments(PChar(Param));
    SetDescription(PChar(Desc));
    SetPath(PChar(PathObj));
  end;
  PFile.Save(PWChar(WideString(PathLink)), False);
end;
```

Примечание

В данном примере PathObj *– объект и путь к нему,* PathLink *– рабочий каталог,* Desc *– размещение ярлыка,* Param *– параметры.*

Изменение координат ярлыков на Рабочем столе

Как программным путем задавать координаты ярлыкам на Рабочем столе?

Рабочий стол перекрыт сверху компонентом ListView. Необходимо получить Handle этого органа управления.

Решение

```
function GetDesktopListViewHandle: THandle;
var
  S: String;
begin
  Result := FindWindow('ProgMan', nil);
  Result := GetWindow(Result, GW_CHILD);
  Result := GetWindow(Result, GW_CHILD);
  SetLength(S, 40);
  GetClassName(Result, PChar(S), 39);
  if PChar(S) <> 'SysListView32' then Result := 0;
end;
```

Получив Handle, можно обратиться к API этого ListView, определенному в модуле CommCtrl, для того чтобы манипулировать Рабочим столом. См. тему «LVM_xxxx messages» в оперативной справке по Win32. Например, следующая строка кода:

```
{ Не забудьте в uses добавить CommCtrl }
ListView_SetItemPosition(GetDesktopListViewHandle, i, x, y);
```

для ярлыка с индексом i задаст координаты (x, y).

[Сахаров Сергей]

Примечание

Строка с вызовом GetWindow *повторяется дважды. Это не опечатка!*

Дополнение

Код:

```
SendMessage(GetDesktopListViewHandle, LVM_ARRANGE, LVA_ALIGNLEFT, 0);
```

разместит пиктограммы по левой стороне Рабочего стола Windows.

[Nomadic]

Пиктограмма приложения в окне Tray

Как поместить пиктограмму в Tray*?*

Решение

```
function TaskBarAddIcon(hWindow: THandle; ID: Cardinal; ICON: hicon;
                        CallbackMessage: Cardinal; Tip: String): boolean;
var
  NID: TNotifyIconData;
```

```
begin
  FillChar(NID, SizeOf(TNotifyIconData), 0);
  with NID do begin
    cbSize := SizeOf(TNotifyIconData);
    Wnd := hWindow;
    uID := ID;
    uFlags := NIF_MESSAGE or NIF_ICON or NIF_TIP;
    uCallbackMessage := CallbackMessage;
    hIcon := Icon;
    if Length(Tip) > 63 then SetLength(Tip, 63);
    StrPCopy(szTip, Tip);
  end;
  Result := Shell_NotifyIcon(NIM_ADD, @NID);
end;
```

[News Group]

Примечание

В uses *необходимо добавить* ShellAPI.

Всплывающее меню и Tray

Иногда мое приложение, помещенное в Tray*, не убирает всплывающее меню после потери фокуса. Что делать?*

Во время обработки сообщений PopUp меню надо назначить активное окно, а потом, после всплытия меню, послать сообщение WM_NULL.

```
procedure TForm1.WndProc(var Msg: TMessage);
var
  p: TPoint;
begin
  case Msg.Msg of
    WM_USER + 1:
      case Msg.lParam of
        WM_RBUTTONDOWN: begin
                          SetForegroundWindow(Handle);
                          GetCursorPos(p);
                          PopupMenu1.Popup(p.x, p.y);
                          PostMessage(Handle, WM_NULL, 0, 0);
                        end;
      end;
  end;
  inherited;
end;
```

Рисование на минимизированной пиктограмме

Есть ли у кого-нибудь пример рисования на значке минимизированного приложения с помощью Delphi?

Когда Delphi-приложение минимизировано, пиктограмма, которую вы видите, – реальное главное окно, объект TApplication. Поэтому необходимо его использовать. Таким образом, чтобы удостовериться, что приложение минимизировано, вызовите IsIconic(Application.Handle). Если функция возвратит True, значит, так оно и есть. Для рисования на пиктограмме создайте обработчик события Application.OnMessage. Здесь можно проверять наличие сообщения WM_Paint и при его возникновении рисовать пиктограмму. Это должно выглядеть приблизительно так:

```
  ...
{ private declarations }
  procedure AppOnMessage(var Msg: TMsg; var Handled: Boolean);
  ...

procedure TForm1.AppOnMessage(var Msg: TMsg; var Handled: Boolean);
var
  DC: hDC;
  PS: TPaintStruct;
begin
  if (Msg.Message = WM_PAINT) and IsIconic(Application.Handle) then begin
    DC := BeginPaint(Application.Handle, PS);
// осуществляем отрисовку с помощью вызовов Windows GDI ...
    EndPaint(Application.Handle, PS);
    Handled := True;
  end;
end;

procedure TForm1.OnCreate(Sender : TObject);
begin
  Application.OnMessage := AppOnMessage;
end;
```

[News Group]

Метка диска под Win32

По моему глубокому убеждению, для получения метки диска в среде Win95 необходимо использовать FindFile. *Но это не работает, так?*

Функция FindFile в Win32 больше не возвращает имя диска, поскольку в файловых системах, отличных от FAT (например, в NTFS), это работает иначе. Обратитесь к функции API GetVolumeInformation.

[News Group]

Процедура форматирования

В Shell32.dll спрятана функция WinAPI SHFormatDrive, вызывающая стандартный диалог форматирования сменного накопителя.

```
{ раздел реализации }
   ...

const
  SHFMT_ID_DEFAULT = $FFFF;

// Опции форматирования
  SHFMT_OPT_QUICKFORMAT = $0000;
  SHFMT_OPT_FULL = $0001;
  SHFMT_OPT_SYSONLY = $0002;
// Коды ошибок
  SHFMT_ERROR = $FFFFFFFF;
  SHFMT_CANCEL = $FFFFFFFE;
  SHFMT_NOFORMAT = $FFFFFFFD;

function SHFormatDrive(Handle: HWND; Drive, ID, Options: Word): LongInt;
                       stdcall; external 'shell32.dll' name 'SHFormatDrive'
procedure TForm1.btnFormatDiskClickClick(Sender: TObject);
var
  retCode: LongInt;
begin
  retCode := SHFormatDrive(Handle, 0, SHFMT_ID_DEFAULT,
                           SHFMT_OPT_QUICKFORMAT);
  if retCode < 0 then
    ShowMessage('Не могу отформатировать диск');
end;
end.
```

Подсчет размера директории

Как подсчитать занимаемое директорией место?

Попробуйте следующий код. Он просматривает скрытые, системные, архивные и обычные файлы, использует рекурсивный алгоритм для просмотра всех вложенных поддиректорий: достаточно указать стартовый каталог, и функция возвратит результат в переменной `DirBytes`. Имейте в виду, что для определения типа (файл или директория) код использует функции `FileExists` и `DirectoryExists` вместо просмотра атрибутов файла. Причина этого проста – при просмотре CD-ROM функции `FindFirst` и `FindNext` иногда заявляют, что файл является каталогом.

```
var
  DirBytes: integer;
function DirSize(Dir: string): integer;
var
  SearchRec: TSearchRec;
  Separator: string;
begin
  if Copy(Dir, Length(Dir), 1) = '\' then Separator := ''
  else Separator := '\';
```

```
  if FindFirst(Dir + Separator + '*.*', faAnyFile, SearchRec) = 0 then begin
    if FileExists(Dir + Separator + SearchRec.Name) then begin
      DirBytes := DirBytes + SearchRec.Size;
{ Memo1.Lines.Add(Dir + Separator + SearchRec.Name); }
    end else if DirectoryExists(Dir + Separator + SearchRec.Name) then begin
      if (SearchRec.Name <> '.') and (SearchRec.Name <> '..') then begin
        DirSize(Dir + Separator + SearchRec.Name);
      end;
    end;
    while FindNext(SearchRec) = 0 do begin
      if FileExists(Dir + Separator + SearchRec.Name) then begin
        DirBytes := DirBytes + SearchRec.Size;
{ Memo1.Lines.Add(Dir + Separator + SearchRec.Name); }
      end else if DirectoryExists(Dir + Separator + SearchRec.Name) then begin
        if (SearchRec.Name <> '.') and (SearchRec.Name <> '..') then begin
          DirSize(Dir + Separator + SearchRec.Name);
        end;
      end;
    end;
  end;
  FindClose(SearchRec);
end;
```

Примечание

Добавьте в uses *модуль* FileCtrl. *Вызывается этот пример так:*

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  DirBytes := 0;
  DirSize('c:\windows');
  Label1.Caption := IntToStr(DirBytes);
end;
```

Как видно из примера, возвращаемое значение нас не интересует. Такая вот функция...

Поиск загрузочного диска

Есть какая-либо функция или вызов API для поиска загрузочного диска?

Решение

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Setup
```

Значение параметра BootDir и есть искомая величина.

Поиск на жестком диске

Я ищу метод или компонент, выполняющий поиск каких-либо файлов на диске, например (.exe*).*

Решение

```
unit Audit1;

interface

uses
  Windows, SysUtils, Dialogs;

var
  dest: string;

procedure DoRecurse(dir: string);

implementation

procedure Process(dir: string; SearchRec: TSearchRec);
begin
  ShowMessage(SearchRec.Name);
  case SearchRec.Attr of
      $10: if (SearchRec.Name <> '.') and (SearchRec.Name <> '..') then begin
             DoRecurse(dir + '\' + SearchRec.Name);
             Writeln(dir);
           end;
  end;
end;

procedure DoRecurse(dir: string);
var
  SearchRec: TSearchRec;
  pc: array[0..79] of Char;
begin
  StrPCopy(pc, dir + '\*.*');
  FindFirst(pc, FaAnyfile, SearchRec);
  Process(dir, SearchRec);
  while FindNext(SearchRec) <> - 18 do
    Process(dir, SearchRec);
end;

procedure StartSearch;
begin
  DoRecurse(ParamStr(1));
end;

begin
  StartSearch;
end.
```

Управление каталогами и файлами

Решение

```
unit win95;
{ Копирование, перемещение и удаление файлов и каталогов по образцу Проводника
  (Explorer) в Windows 95 }

interface

uses
  Classes, ShellApi, ShlObj, Registry, Windows;

type
  Str10 = String[10];

const
  fpRootKey ='\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders';
  fpDesktop: Str10 = 'DESKTOP';
  fpFavorites: Str10 = 'FAVORITES';
  fpFonts: Str10 = 'FONTS';
  fpPersonal: Str10 = 'PERSONAL';
  fpPrograms: Str10 = 'PROGRAMS';
  fpRecent: Str10 = 'RECENT';
  fpSendTo: Str10 = 'SENDTO';
  fpStartMenu: Str10 = 'START MENU';
  fpStartup: Str10 = 'STARTUP';
  fpTemplates: Str10 = 'TEMPLATES';

{ Пути к системным папкам }
function GetFolderPath(Const FolderName: Str10): String;

{ Функции для работы с файлами }
procedure Win95AddToRecentDocs(Const Filename: string);
procedure Win95ClearRecentDocs;

{ Для манипулирования несколькими файлами разделите их имена символом "#0" }
function Win95Copy(Owner: Integer; FromFile,ToFile: String; RenameOnCollision,
                   Confirm: boolean): Boolean;
function Win95Move(Owner: Integer; FromFile,ToFile: String; RenameOnCollision,
                   Confirm: boolean): Boolean;
{ Если SendToRecycleBin = True, то файлы будут отправлены в Корзину (RecycleBin),
  в противном случае они будут стерты }
function Win95Erase(Owner: Integer; WichFiles: String; SendToRecycleBin,
                    Confirm: Boolean): Boolean;

implementation

function GetFolderPath(Const FolderName: Str10): String;
begin
  with TRegistry.Create do
    try
      RootKey := HKEY_CURRENT_USER;
      OpenKey(fpRootKey, False);
      Result := ReadString(FolderName);
```

```
    finally
      Free;
    end;
end;

procedure Win95AddToRecentDocs(Const Filename: string);
begin
  SHAddToRecentDocs(SHARD_PATH, @Filename[1]);
end;

procedure Win95ClearRecentDocs;
begin
  SHAddToRecentDocs(SHARD_PATH, nil);
end;

function Win95Copy(Owner: Integer; FromFile, ToFile: String; RenameOnCollision,
                   Confirm: boolean): Boolean;
const
  Aborted: Boolean = False;
var
  Struct: TSHFileOpStructA;
begin
  while pos(';', FromFile) > 0 do
    FromFile[pos(';', FromFile)] := #0;
  while pos(';', ToFile) > 0 do
    ToFile[pos(';', ToFile)] := #0;
  FromFile := FromFile + #0#0;
  ToFile := ToFile + #0#0;
  with Struct do begin
    wnd := Owner;
    wFunc := FO_Copy;
    pFrom := PChar(FromFile);
    pTo := PChar(ToFile);
    fFlags := FOF_ALLOWUNDO or FOF_FILESONLY;
    if RenameOnCollision then fFLags := fFlags or FOF_RENAMEONCOLLISION;
    if not Confirm then fFLags := fFlags or FOF_NOCONFIRMATION;
    fAnyOperationsAborted := Aborted;
    hNameMappings := nil;
    lpszProgressTitle := nil;
  end;
  result := (SHFileOperationA(Struct) = 0) and (not Aborted);
end;

function Win95Move(Owner: Integer; FromFile, ToFile: String; RenameOnCollision,
                   Confirm: boolean): Boolean;
const
  Aborted: Boolean = False;
var
  Struct: TSHFileOpStructA;
begin
```

```
  while pos(';', FromFile) > 0 do
    FromFile[pos(';', FromFile)] := #0;
  while pos(';', ToFile)>0 do
    ToFile[pos(';', ToFile)] := #0;
  FromFile := FromFile + #0#0;
  ToFile := ToFile + #0#0;
  with Struct do begin
    wnd := Owner;
    wFunc := FO_Move;
    pFrom := PChar(FromFile);
    pTo := PChar(ToFile);
    fFlags := FOF_ALLOWUNDO or FOF_FILESONLY;
    if RenameOnCollision then fFLags := fFlags or FOF_RENAMEONCOLLISION;
    if Confirm then fFLags := fFlags or FOF_NOCONFIRMATION;
    fAnyOperationsAborted := Aborted;
    hNameMappings := nil;
    lpszProgressTitle := nil;
  end;
  result := (SHFileOperationA(Struct) = 0) and (not Aborted);
end;
function Win95Erase(Owner: Integer; WichFiles: String; SendToRecycleBin,
                    Confirm: Boolean): Boolean;
const
  Aborted: Boolean = False;
var
  Struct: TSHFileOpStructA;
begin
  while pos(';', WichFiles) > 0 do
    WichFiles[pos(';', WichFiles)] := #0;
  WichFiles := WichFiles + #0#0;
  with Struct do begin
    wnd := Owner;
    wFunc := FO_Delete;
    pFrom := PChar(WichFiles);
    pTo := nil;
    if not Confirm then fFlags := FOF_NOCONFIRMATION;
    if SendToRecycleBin then fFLags := fFlags or FOF_ALLOWUNDO or FOF_FILESONLY
    else fFlags := fFlags or 0 or FOF_FILESONLY;
    fAnyOperationsAborted := Aborted;
    hNameMappings := nil;
    lpszProgressTitle := nil;
  end;

  result := (SHFileOperationA(Struct) = 0) and (not Aborted);
end;

end.
```

Получение времени создания файла

Как получить время создания файла?

Решение

```
function GetFileDate(TheFileName: string): string;
var
  FHandle: integer;
begin
  FHandle := FileOpen(TheFileName, 0);
  Result := DateTimeToStr(FileDateToDateTime(FileGetDate(FHandle)));
  FileClose(FHandle);
end;
```

Определение размера файла

Стандартная функция Delphi FileSize не может быть использована для определения размера текстовых файлов. Ниже приведен текст функции, определяющей размер любых файлов посредством вызова соответствующих функций WinAPI. Для использования должен быть подключен модуль Windows.

```
function AnyFileSize(FileName: PChar): LongWord;
var
  hFile: THandle;
begin
  hFile := CreateFile(FileName, GENERIC_READ, FILE_SHARE_READ + FILE_SHARE_WRITE,
                      Nil, OPEN_EXISTING, 0, 0);
// Попытка получить размер файла:
  Result := GetFileSize(hFile, Nil); CloseHandle(hFile);
// Возможно, произошла ошибка...
  if Result = $FFFFFFFF then begin
// Произошла ошибка - возвращаем нулевой размер
    Result := 0;
    Exit;
  end;
end;
```

[Kostin Slava]

Управление атрибутом файла date/time

Как написать функцию, которая устанавливает дату одного файла, равную дате другого файла?

Рассмотрим функцию, использующую в качестве параметров две строки с полными путями/именами файлов. Файл, дату которого требуется установить, указывается как второй параметр. А тот файл, дата которого используется, – первым.

```
procedure CopyFileDate(const Source, Dest: String);
var
  SourceHand, DestHand: word;
begin
  SourceHand := FileOpen(Source, fmOutput);          { открываем исходный файл }
  DestHand := FileOpen(Dest, fmInput);               { открываем целевой файл }
  FileSetDate(DestHand, FileGetDate(SourceHand));    { получаем/устанавливаем дату }
```

```
    FileClose(SourceHand);                          { закрываем исходный файл }
    FileClose(DestHand);                            { закрываем целевой файл }
  end;
```

Ассоциирование типов файлов

В общих чертах, необходимо добавить два ключа в ветку реестра HKEY_CLASSES_ROOT. Зарегистрируйте в корне расширение типа .<ext> (создайте ключ с именем расширения):

```
HKEY_CLASSES_ROOT\.<ext>\
```

и запишите в строковый параметр По умолчанию созданного ключа «внутреннее имя» вашего типа файлов – например, MyApp.Document:

```
HKEY_CLASSES_ROOT\.ext\(По умолчанию) = "MyApp.Document"
```

Затем создайте другой ключ с этим именем:

```
HKEY_CLASSES_ROOT\MyApp.Document\
```

Создайте подключ с именем "shell", в нем другой подключ с именем "open", а в нем, в свою очередь, еще один подключ с именем "command". Значение по умолчанию (default) – путь и имя вашего приложению с ключом "%1", представляющим параметр «имя файла», позволяя системе подставлять вызванный файл:

```
HKEY_CLASSES_ROOT\MyApp.Document\shell\open\command\(По умолчанию) =
  "C:\myapp\myapp.exe %1"
```

Все манипуляции в коде производятся с помощью объекта TRegistry или при использовании InstallShield, который сделает это за вас автоматически. Применяйте оба способа, поскольку пользователь может внести в реестр всякий мусор.

Еще один совет. Самый простой путь достижения цели – модифицировать секцию Extensions в файле win.ini, расположенном в директории Windows. Это также работает под Windows 95, автоматически обновляя реестр при перезапуске системы. Взгляните на секцию Extensions файла win.ini для определения формата записи. Поместите IniFiles в секцию используемых модулей и создайте код, подобный следующему:

```
var
  INIFile: TIniFile;
begin
  try
    INIFile := TInifile.Create('WIN.INI');
    INIFile.WriteString('Extensions', 'txt', 'c:\windows\notepad.exe ^.txt');
  finally
    INIFile.Free;
  end;
end;
```

Это ассоциирует файлы, имеющие расширение *.TXT, с Блокнотом (Notepad) Windows. Например, для ассоциации приложения MyApp, расположенного в каталоге C:\MyApps, с файлами *.MAP необходимо выполнить следующее:

```
var
  INIFile: TIniFile;

begin
  try
    INIFile := TInifile.Create('WIN.INI');
    INIFile.WriteString('Extensions', 'map', 'c:\myapps\myapp.exe ^.map');

  finally
    INIFile.Free;
  end;

end;
```

Дополнение

Чтобы описание типа файлов появилось в списке Типы файлов Windows 98, укажите в параметре По умолчанию ключа HKEY_CLASSES_ROOT\MyApp.Document\ '<краткое описание типа файлов (файл MyApp)>'.

Чтобы ассоциированные файлы имели соответствующую приложению пиктограмму, нужно в параметре По умолчанию ключа HKEY_CLASSES_ROOT\MyApp.Document\DefaultIcon\ указать путь к ней. Пиктограммы соответствующих файлов обновятся после перезагрузки Windows. Windows 98 делает это без перезагрузки системы.

Добавьте 'Registry' в секцию Uses. Объявите переменную RegFile.

Пример кода (проверено в Delphi 5 под Windows 98):

```
procedure TForm1.Button1Click(Sender: TObject);

var
  RegFile: TRegIniFile;

begin
  RegFile := TRegIniFile.Create;
  RegFile.RootKey := HKEY_CLASSES_ROOT;                //устанавливаем текущий ключ
  RegFile.WriteString('.ext', '', 'MyApp.Document');   // определяем расширение
// Описание ассоциированных файлов (для списка "Типы файлов" Windows 98).
  RegFile.WriteString('MyApp.Document', '', 'Описание файлов');
// устанавливаем икону для ассоциированных файлов
  RegFile.WriteString('MyApp.Document \DefaultIcon', '',
                      'Путь к пиктограмме для файлов');
// Исполняемый файл (полный путь)
  RegFile.WriteString(' MyApp.Document \Shell\Open\Command', '' ,
                      'исполняемый файл "%1"');
  RegFile.CloseKey;
  RegFile.Free;
```

```
end;
```

[Шпанер Михаил]

Сокращенные имена файлов

Как преобразовать <Long File Name>.pas *в* <longfi~1>.pas*?*

Решение

```
function GetShortFileName(Const FileName: String): String;
var
  aTmp: array[0..255] of char;
begin
  if GetShortPathName(PChar(FileName), aTmp, SizeOf(aTmp) - 1) = 0 then
    Result := FileName
  else
    Result := StrPas(aTmp);
end;
function GetLongFileName(Const FileName: String): String;
var
  aInfo: TSHFileInfo;
begin
  if SHGetFileInfo(PChar(FileName), 0, aInfo, SizeOf(aInfo),
                   SHGFI_DISPLAYNAME) <> 0 then
    Result := String(aInfo.szDisplayName)
  else
    Result := FileName;
end;
```

Примечание

Модуль ShellAPI *необходимо добавить в* uses.

Восстановление длинных имен файлов по известным коротким

Решение

```
// Восстанавливает длинные имена файлов по известным коротким (8.3)
function RestoreLongName(fn: string): string;

  function LookupLongName(const filename: string): string;
  var
    sr: TSearchRec;
  begin
    if FindFirst(filename, faAnyFile, sr) = 0 then Result := sr.Name
    else Result := ExtractFileName(filename);
    SysUtils.FindClose(sr);
  end;

  function GetNextFN: string;
  var
```

```
    i: integer;
  begin
    Result := '';
    if Pos('\\', fn) = 1 then begin
      Result := '\\';
      fn := Copy(fn, 3, length(fn) - 2);
      i := Pos('\', fn);
      if i <> 0 then begin
        Result := Result + Copy(fn, 1, i);
        fn := Copy(fn, i + 1, length(fn) - i);
      end;
    end;
    i := Pos('\', fn);
    if I <> 0 then begin
      Result := Result + Copy(fn, 1, i - 1);
      fn := Copy(fn, i + 1, length(fn) - i);
    end else begin
      Result := Result + fn;
      fn := '';
    end;
  end;

var
  name: string;
begin
  fn := ExpandFileName(fn);
  Result := GetNextFN;
  repeat
    name := GetNextFN;
    Result := Result + '\' + LookupLongName(Result + '\' + name);
  until length(fn) = 0;
end;
```

Блокировка файла

Для того чтобы получить монопольный доступ к файлу, нужно воспользоваться переменной FileMode.

Решение

```
type
  FileShareType = (DenyCompatibility, DenyAll, DenyWrite, DenyRead, DenyNone);
  FileAccessType = (ReadOnly, WriteOnly, ReadWrite);

procedure SetFileAccess(AccessMode: FileAccessType; ShareMode: FileShareType);
{ Устанавливаем режим доступа к файлу для следующего вызова открытия файла }
begin
  FileMode := ord(AccessMode) or (ord(ShareMode) shl 4);
end;
```

Поиск строки в текстовом файле

Кто-нибудь знает быстрый способ поиска строки в текстовом файле?

Решение

```
unit BMSearch;
{ Поиск строки методом Boyer-Moore.
  Это один из самых быстрых алгоритмов поиска строки. }

interface

type
  {$ifdef WINDOWS}
  size_t = Word;
{$else}
  size_t = LongInt;
{$endif}

type
  TTranslationTable = array[char] of char;       { таблица перевода }
  TSearchBM = class(TObject)
  private
    FTranslate: TTranslationTable;               { таблица перевода }
    FJumpTable: array[char] of Byte;             { таблица переходов }
    FShift_1: integer;
    FPattern: pchar;
    FPatternLen: size_t;
  public
    procedure Prepare(Pattern: pchar; PatternLen: size_t; IgnoreCase: Boolean);
    procedure PrepareStr(const Pattern: string; IgnoreCase: Boolean);
    function Search(Text: pchar; TextLen: size_t): pchar;
    function Pos(const S: string): integer;
  end;

implementation
uses
  SysUtils;

{ Игнорируем регистр таблицы перевода }
procedure CreateTranslationTable(var T: TTranslationTable; IgnoreCase: Boolean);
var
  c: char;
begin
  for c := #0 to #255 do T[c] := c;
  if not IgnoreCase then exit;
  for c := 'a' to 'z' do T[c] := UpCase(c);
{ Связываем все нижние символы с их эквивалентом верхнего регистра }
  T['Б'] := 'A';
  T['А'] := 'A';
  T['Д'] := 'A';
  T['В'] := 'A';
  T['б'] := 'A';
  T['а'] := 'A';
  T['д'] := 'A';
```

```
  T['в'] := 'A';
  T['Й'] := 'E';
  T['И'] := 'E';
  T['Л'] := 'E';
  T['К'] := 'E';
  T['й'] := 'E';
  T['и'] := 'E';
  T['л'] := 'E';
  T['к'] := 'E';
  T['Н'] := 'I';
  T['М'] := 'I';
  T['П'] := 'I';
  T['О'] := 'I';
  T['н'] := 'I';
  T['м'] := 'I';
  T['п'] := 'I';
  T['о'] := 'I';
  T['У'] := 'O';
  T['Т'] := 'O';
  T['Ц'] := 'O';
  T['Ф'] := 'O';
  T['у'] := 'O';
  T['т'] := 'O';
  T['ц'] := 'O';
  T['ф'] := 'O';
  T['Ъ'] := 'U';
  T['Щ'] := 'U';
  T['Ь'] := 'U';
  T['Ы'] := 'U';
  T['ъ'] := 'U';
  T['щ'] := 'U';
  T['ь'] := 'U';
  T['ы'] := 'U';
  T['с'] := 'C';
end;

{ Подготовка таблицы переходов }
procedure TSearchBM.Prepare(Pattern: pchar; PatternLen: size_t;
                            IgnoreCase: Boolean);
var
  i: integer;
  c, lastc: char;
begin
  FPattern := Pattern;
  FPatternLen := PatternLen;
  if FPatternLen < 1 then FPatternLen := strlen(FPattern);
{ Данный алгоритм базируется на наборе из 256 символов }
  if FPatternLen > 256 then exit;
{ 1. Подготовка таблицы перевода }
  CreateTranslationTable(FTranslate, IgnoreCase);
{ 2. Подготовка таблицы переходов }
```

```
  for c := #0 to #255 do FJumpTable[c] := FPatternLen;
  for i := FPatternLen - 1 downto 0 do begin
    c := FTranslate[FPattern[i]];
    if FJumpTable[c] >= FPatternLen - 1 then FJumpTable[c] := FPatternLen - 1 - i;
  end;
  FShift_1 := FPatternLen - 1;
  lastc := FTranslate[Pattern[FPatternLen - 1]];
  for i := FPatternLen - 2 downto 0 do
    if FTranslate[FPattern[i]] = lastc then begin
      FShift_1 := FPatternLen - 1 - i;
      break;
    end;
  if FShift_1 = 0 then FShift_1 := 1;
end;
procedure TSearchBM.PrepareStr(const Pattern: string; IgnoreCase: Boolean);
var
  str: pchar;
begin
  if Pattern <> '' then begin
{$ifdef Windows}
    str := @Pattern[1];
{$else}
    str := pchar(Pattern);
{$endif}
    Prepare(str, Length(Pattern), IgnoreCase);
  end;
end;
{ Поиск последнего символа & просмотр справа налево }
function TSearchBM.Search(Text: pchar; TextLen: size_t): pchar;
var
  shift, m1, j: integer;
  jumps: size_t;
begin
  result := nil;
  if FPatternLen > 256 then exit;
  if TextLen < 1 then TextLen := strlen(Text);
  m1 := FPatternLen - 1;
  shift := 0;
  jumps := 0;
{ Поиск последнего символа }
  while jumps <= TextLen do begin
    Inc(Text, shift);
    shift := FJumpTable[FTranslate[Text^]];
    while shift <> 0 do begin
      Inc(jumps, shift);
      if jumps > TextLen then exit;
      Inc(Text, shift);
      shift := FJumpTable[FTranslate[Text^]];
    end;
{ Сравниваем справа налево FPatternLen - 1 символов }
    if jumps >= m1 then begin
```

```
        j := 0;
        while FTranslate[FPattern[m1 - j]] = FTranslate[(Text - j)^] do begin
          Inc(j);
          if j = FPatternLen then begin
            result := Text - m1;
            exit;
          end;
        end;
      end;
      shift := FShift_1;
      Inc(jumps, shift);
    end;
  end;

  function TSearchBM.Pos(const S: string): integer;
  var
    str, p: pchar;
  begin
    result := 0;
    if S <> '' then begin
  {$ifdef Windows}
      str := @S[1];
  {$else}
      str := pchar(S);
  {$endif}
      p := Search( str, Length(S));
      if p <> nil then result := 1 + p - str;
    end;
  end;

  end.
```

Чтение атрибута файла Last Accessed

Если в Проводнике Windows 9x на любом файле нажать правую кнопку мыши и выбрать пункт Свойства*, можно увидеть время последнего доступа к данному файлу. Как прочесть атрибут файла «Last Accessed» в Delphi?*

Решение

```
procedure TForm1.Button1Click(Sender: TObject);
var
  FileHandle: THandle;
  LocalFileTime: TFileTime;
  DosFileTime: DWORD;
  LastAccessedTime: TDateTime;
  FindData: TWin32FindData;
begin
  FileHandle := FindFirstFile('AnyFile.FIL', FindData);
  if FileHandle <> INVALID_HANDLE_VALUE then begin
    Windows.FindClose(Handle);
    if (FindData.dwFileAttributes and FILE_ATTRIBUTE_DIRECTORY) = 0 then begin
```

```
        FileTimeToLocalFileTime(FindData.ftLastWriteTime, LocalFileTime);
        FileTimeToDosDateTime(LocalFileTime, LongRec(DosFileTime).Hi,
                              LongRec(DosFileTime).Lo);
        LastAccessedTime := FileDateToDateTime(DosFileTime);
        Label1.Caption := DateTimeToStr(LastAccessedTime);
       end;
    end;
  end;
```

Копирование файлов

Как скопировать файл?

Решение 1

Три способа копирования файлов:

```
{ Данный способ использует файловый поток. }
procedure FileCopy(Const sourcefilename, targetfilename: String);
var
  S, T: TFileStream;
begin
  S := TFileStream.Create(sourcefilename, fmOpenRead);
  try
    T := TFileStream.Create(targetfilename, fmOpenWrite or fmCreate);
    try
      T.CopyFrom(S, S.Size);
    finally
      T.Free;
    end;
  finally
    S.Free;
  end;
end;
{ Данный способ для чтения/записи использует блоки памяти. }
procedure FileCopy(const FromFile, ToFile: string);
var
  FromF, ToF: file;
  NumRead, NumWritten: Integer;
  Buf: array[1..2048] of Char;
begin
  AssignFile(FromF, FromFile);
  Reset(FromF, 1);                    { Размер записи = 1 }
  AssignFile(ToF, ToFile);            { Открываем выходной файл }
  Rewrite(ToF, 1);                    { Размер записи = 1 }
  repeat
    BlockRead(FromF, Buf, SizeOf(Buf), NumRead);
    BlockWrite(ToF, Buf, NumRead, NumWritten);
  until (NumRead = 0) or (NumWritten <> NumRead);
  CloseFile(FromF);
  CloseFile(ToF);
end;
```

```
{ Данный способ использует LZCopy, добавьте в список USES модуль LZExpand. }
procedure CopyFile(FromFileName, ToFileName: string);
var
  FromFile, ToFile: File;
begin
  AssignFile(FromFile, FromFileName);  { Присваиваем FromFile FromFileName }
  AssignFile(ToFile, ToFileName);      { Присваиваем ToFile ToFileName }
  Reset(FromFile);                     { Открываем файл для чтения }
  try
    Rewrite(ToFile);                   { Создаем файл для записи }
    try
{ копируем файл, если возвращена отрицательная величина возбуждаем исключение }
      if LZCopy(TFileRec(FromFile).Handle, TFileRec(ToFile).Handle) < 0 then
        raise EInOutError.Create('Ошибка использования LZCopy')
    finally
      CloseFile(ToFile);               { Закрываем ToFile }
    end;
  finally
    CloseFile(FromFile);               { Закрываем FromFile }
  end;
end;
```

Решение 2

```
CopyFile('c:\1.txt', 'c:\files\2.txt', True);
```

где первый параметр – путь и имя нужного файла, второй – путь и имя нового файла, третий – признак перезаписи файла в случае его присутствия.

Если необходимо задавать имена файлов через Edit, то:

```
CopyFile(PChar(Edit1.Text), PChar(Edit2.Text), True);
```

[Nikolaev Igor]

Удаление файлов

Как удалить все файлы из директории?

Решение

```
procedure TfrmMain.DelDir(DirName: string);
var
  SearchRec: TSearchRec;
  GotOne: integer;
begin
  GotOne := FindFirst(DirName + '\*.*', faAnyFile, SearchRec);
  while GotOne = 0 do begin
    if ((SearchRec.Attr and faDirectory) = 0) then
      DeleteFile(DirName + '\' + SearchRec.Name)
    else if (SearchRec.Name <> '.') and (SearchRec.Name <> '..') then
      DelDir(DirName + '\' + SearchRec.Name);
    GotOne:= FindNext(SearchRec);
  end;
```

```
    FindClose(SearchRec);
  end;
```

Если впоследствии понадобится директорию удалить, попробуйте сделать так:

```
  ...
  DelDir('C:\WASTE');
  {-I}
  RmDir('C:\WASTE');
  {+I}
  if IOResult <> 0 then raise Exception.Create('Ошибка удаления каталога');
  ...
```

[News Group]

Особенности использования DeleteFile()

Почему при вызове `DeleteFile()` *генерируется ошибка несовместимых типов?*

Надо учитывать, что определение этой функции есть и в `SysUtils` и в `Windows`.

Пример ниже показывает использование обоих вариантов.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  s: string;
  a: array[0..MAX_PATH - 1] of char;
begin
  s := 'C:\SomeFile';
  SysUtils.DeleteFile(s);
  a := 'C:\SomeFile';
  Windows.DeleteFile(@a);
end;
```

Открытие файла общего доступа

Как открыть файл общего доступа?

Проверьте переменную `FileMode` и сравните ее значение с приведенным ниже списком:

```
const
  fmReadOnly = $00;
  fmWriteOnly = $01;
  fmReadWrite = $02;
  fmDenyAll = $10;
  fmDenyWrite = $20;
  fmDenyRead = $30;
  fmDenyNone = $40;
  fmNoInherit = $80;
```

Можно использовать их совместно. Например, так:

```
FileMode := fmReadWrite + fmDenyAll;
```

или так:

```
FileMode := fmReadOnly + fmDenyNone;
```

[News Group]

Открытие файла только на чтение

Перед открытием или созданием файла установите переменную FileMode. Чтобы установить ее, воспользуйтесь 'File Open Mode constants' (константы режима открытия файла). Взгляните в описание модуля Sysutils, там находится перечень 'File Open Mode constants'. Константы лучше связывать логическим оператором OR. Например, с fmOpen или с константой fmShare.

Ознакомьтесь в файле помощи с описанием переменной FileMode. Если перед открытием файла ее установили в ноль, файл будет открыт только для чтения. По умолчанию для нетипизированных файлов установлен доступ read/write (чтение/запись).

Решение

```
AssignFile(F, FileName);
FileMode := 0; { устанавливаем доступ к файлу только для чтения }
Reset(F);
  ...
CloseFile(F);
```

Чтение данных из файла

Можно использовать процедуру Flush, которая работает с открытыми файлами. В руководстве четко не сказано, передает ли Flush данные непосредственно на диск. Если это не так, то данные сохраняются в других временных буферах.

Решение

```
uses
  SysUtils;

var
  F: text;

procedure TextFlush(F: Text);
var
  fhandle: word;
begin
  Flush(F);
  fhandle := TTextRec(F).Handle;        { получаем дескриптор msdos }
  asm
    mov   ax, $6800
    mov   bx, handle
    call  DOS3CALL
```

```
    end;
  end;
```

Если файл является «блочным», пропускаем шаг с командой Flush, и используем TFileRec вместо TTextRec.

Переменная FileMode определяет режим открытия файла (по умолчанию режим эксклюзивный). К сожалению, это не срабатывает для текстовых файлов, поэтому необходимо, используя BlockReads, писать в буфер и затем конвертировать части буфера в строку, если вы хотите работать с ним как с текстовым файлом.

Assign или AssignFile, как вы знаете, не может использоваться с файлом, который уже открыт. В данном случае рекомендуем обратиться к вызову API OpenFile.

Если это текстовый файл, сбросьте сначала текстовый буфер на диск командой Flush:

```
Flush(f);
```

Чтение и запись файлов

Как направить выходной поток программы в файл?

Решение

```
program Project1;
{$APPTYPE CONSOLE}
uses SysUtils;

var
  outfile: TextFile;

begin
  AssignFile(outfile, 'c:\outfile.txt');
  Rewrite(outfile);
  Writeln(outfile, 'Привет из Delphi');
  Writeln(outfile, 'Моя программа работает и выводит '
              +  'данный текст, чтобы доказать это...');
  CloseFile(outfile);
end.
```

Как направить выходной поток программы на принтер?

Иногда необходимо печатать генерируемые программой данные непосредственно на принтер. Вот как это можно сделать:

```
program Project1;

{$APPTYPE CONSOLE}
uses SysUtils;

var
```

```
  outfile: TextFile;

begin
  AssignFile(outfile, 'LPT1');
  Rewrite(outfile);
  Writeln(outfile, 'Привет из Delphi');
  Writeln(outfile, 'Моя программа работает и выводит '
                + 'данный текст, чтобы доказать это...');
  CloseFile(outfile);
end.
```

Как прочитать данные из входного файла?

Весьма полезным будет иметь функцию чтения из внешнего файла:

```
program Project1;
{$APPTYPE CONSOLE}
uses SysUtils;

var
  infile, outfile: TextFile;
  num_lines, x: integer;
  line: string;

begin
  AssignFile(infile, 'C:\INFILE.TXT');
  AssignFile(outfile, 'C:\OUTFILE.TXT');
  Reset(infile);         { перемещаем указатель в начало файла }
  Rewrite(outfile);      { очищаем содержимое файла }
  Readln(infile, num_lines);
  for x := 1 to num_lines do begin
    Readln(infile, line);
    Writeln(outfile, line);
  end;
  CloseFile(infile);
  CloseFile(outfile);
end.
```

Чтение длинной строки из файла

Для решения этой задачи на помощь можно призвать потоки (`TFileStream`, `TMemoryStream`). Для определения конца строк нужно искать пары `CR/LF`:

```
  ...
  StartPos := Stream.Position;
  EndPos := StartPos;
  Repeat
    Stream.Read(Buffer^, 1024);

{ FindCR возвращает 0..1023 для CR, и 1024, если он не найден }
    CRPos := FindCR(Buffer^);
    Inc(EndPos, CRPos);
```

```
  Until CRPos < 1024;
  GetMem(MyPChar, EndPos - StartPos);
  Stream.Seek(StartPos);
  Stream.Read(MyPChar^, EndPos - StartPos)
  ...
```

Затем установите CR в конце MyPChar в 0 и выполните Seek в позицию (End + 1) или что-то еще, чтобы пропустить LF.

[News Group]

Файл с множеством записей

Обычно используют файл с заголовком, который затем загружают в память, и тогда уже отыскивают необходимую запись.

```
type
  TSaveHeader = record
    scene: Integer;
    hotspots: LongInt;
    talk: LongInt;
    hype: LongInt;
  end;

var
  SaveHeader: TSaveHeader;

procedure OpenSaveFile(fname: String);
var
  f: File;
  i: Integer;
begin
  AssignFile(f, fname);
  Reset(f, 1);
  BlockRead(f, SaveHeader, SizeOf(TSaveHeader));

{ получаем один набор записи }
  Seek(f, SaveHeader.hotspots);
  for i := 1 to 50 do
    BlockRead(f, somevar, SizeOf(hotspotrec);

{ и так далее }
  CloseFile(f);
end;

{ предположим, что файл открыт }
procedure GetHotspotRec(index: LongInt; var hotspotrec: THotspot);
var
  offset: LongInt;
begin
  offset := SaveHeader.hotspots + index * SizeOf(THotSpot);
  Seek(f, offset);
```

```
  BlockRead(f, hotspotrec, SizeOf(THotspot));
end;
```

Доступ к нетипизированному файлу

Можно организовать доступ к нетипизированному файлу таким образом:

```
var
  MyFile: file;

begin
  Assign(MyFile, Filename);
  Reset(MyFile, 1);

{ для записи }
  BlockWrite(MyFile, item, SizeOf(item));
{ для чтения }
  BlockRead(MyFile, item, SizeOf(item));
  Close(MyFile);
end;
```

Имейте в виду, что для чтения/записи нетипизированного файла следует применять функции `BlockRead` и `BlockWrite`, т. к. для использования функций `Read/Write` компилятору необходимо знать тип файла.

Readln для более чем 255 символов

Как мне воспользоваться функцией `Readln()`*, если длина строк в файле превышает 255 символов?*

Функция `Readln` принимает массив символов `array [0..something] of Char` и использует его в качестве буфера для чтения символов, замыкая цепочку терминирующим нулем. Единственное ограничение: компилятор должен иметь возможность вычисления размера буфера во время компиляции. Это делает невозможным объявление переменой типа `PChar` и ее распределение во время выполнения программы.

Обходной путь:

```
type
{ используем самое большое кол-во символов в строке, с которым можно иметь дело }
  TLine = Array [0..1024] of Char;
  PLine = ^TLine;

var
  pBuf: PLine;
  ...
  New(pBuf);
  ...
  ReadLn(F, pBuf^);
  ...
```

Для передачи pBuf функциям, которым требуется параметр типа PChar, используйте приведение типа – примерно так: PChar(pBuf).

Примечание

Можно объявить переменную типа TLine *или непосредственно массив символов, но я предпочитаю распределять из кучи нечто большее, чем 4 байта…*

Импорт больших файлов с разделителями

Две функции, которые можно использовать в проектах. Работать с ними очень просто:

```
function GetField(InpString: String; fieldpos: Integer): String;
var
  c: Char;
  curpos, i: Integer;
begin
  curpos := 1;
  for i := 1 to fieldpos do begin
    result := '';
    if curpos > Length(InpString) then Break;
    repeat
      c := InpString[curpos];
      Inc(curpos, 1);
      if (c = '"') or (c = #13) or (c = #10) then c := ' ';
      if c <> ',' then result := result + c;
    until (c = ',') or (curpos > Length(InpString))
  end;
  if (curpos > Length(InpString)) and (i < fieldpos) then result := '';
  result := Trim(result);
end;

function Trim(inp_str: String): String;
var
  i: Integer;
begin
  for i := 1 to Length(inp_str) do
    if inp_str[i] <> ' ' then Break;
  if i > 1 then Delete(inp_str, 1, i - 1);
  for i := Length(inp_str) downto 1 do
    if inp_str[i] <> ' ' then Break;
  if i < Length(inp_str) then Delete(inp_str, i + 1, Length(inp_str));
  result := inp_str;
  if result = ' ' then result := '';
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  s: String;
  f: TextFile;
```

```
begin
  AssignFile(f, 'D:\INPUT.TXT');
  Reset(f);
  while not EoF(f) do begin
    ReadLn(f, s);
    ShowMessage(GetField(s, 1));     { Первое поле }
    ShowMessage(GetField(s, 6));     { Шестое поле }
    ShowMessage(GetField(s, 25));    { возвратит '', если нет 25 колонки }
  end;
  CloseFile(f);
end;
```

[News Group]

Слияние двух двоичных файлов

Самым простым способом является открытие первого, перемещение в его конец и копирование с этого места второго файла.

```
  ...
var
  f1, f2: File;
  xfer: Word;
  buf: PChar;
begin
  AssignFile(f1, name1);
  Reset(f1);
  Seek(f1, Filesize(f1));
  AssignFile(f2, name2);
  Reset(f2);
  GetMem(buf, 65000);
  Repeat
    BlockRead(f1, buf^, 65000, xfer);
    BlockWrite(f2, buf^, xfer);
  Until xfer < 65000;
  CloseFile(f1);
  CloseFile(f2);
end;
```

[News Group]

Объекты и TRegistry

Как сохранить общие настройки шрифтов формы/панели/списка и пр. в реестре; конечно, можно легко обойтись построчным сохранением, но, к примеру, в случае сохранения свойств шрифтов количество строк выйдет за пределы разумного – кто-нибудь может подсказать мне решение покороче и полегче?

Delphi имеет маленький секрет, позволяющий рекурсивно сохранять в реестре любые общедоступные свойства объектов. В качестве примера покажем, как это работает для `TFont`.

```
uses TypInfo;
{ Определяем тип-набор для доступа к битам целого. }

const
  BitsPerByte = 8;

type
  TIntegerSet = set of 0..SizeOf(Integer) * BitsPerByte - 1;

{ Сохраняем набор свойств в виде подключа. Каждый элемент перечислимого типа -
  отдельная логическая величина. Истина означает что элемент включен в набор,
  Ложь - элемент в наборе отсутствует. Это позволит пользователю с помощью
  редактора реестра (REGEDIT) легко изменять конфигурацию. }
procedure SaveSetToRegistry(const Name: string; Value: Integer;
                            TypeInfo: PTypeInfo; Reg: TRegistry);
var
  OldKey: string;
  I: Integer;
begin
  TypeInfo := GetTypeData(TypeInfo)^.CompType;
  OldKey := '\' + Reg.CurrentPath;
  if not Reg.OpenKey(Name, True) then
    raise ERegistryException.CreateFmt('Не могу создать ключ: %s', [Name]);
{ Организуем цикл для всех элементов перечислимого типа. }
  with GetTypeData(TypeInfo)^ do
    for I := MinValue to MaxValue do
{ Записываем логическую величину для каждого установленного элемента. }
      Reg.WriteBool(GetEnumName(TypeInfo, I), I in TIntegerSet(Value));
{ Возвращаем родительский ключ. }
  Reg.OpenKey(OldKey, False);
end;

{ Сохраняем объект в реестре в отдельном ключе. }
procedure SaveObjToRegistry(const Name: string; Obj: TPersistent; Reg: TRegistry);
var
  OldKey: string;
begin
  OldKey := '\' + Reg.CurrentPath;
{ Открываем подключ для объекта. }
  if not Reg.OpenKey(Name, True) then
    raise ERegistryException.CreateFmt('Не могу создать ключ: %s', [Name]);
{ Сохраняем свойства объекта. }
  SaveToRegistry(Obj, Reg);
{ Возвращаем родительский ключ. }
  Reg.OpenKey(OldKey, False);
end;

{ Сохраняем в реестре метод путем записи его имени. }
procedure SaveMethodToRegistry(const Name: string; const Method: TMethod;
                               Reg: TRegistry);
var
```

```
  MethodName: string;
begin
{ Если указатель на метод содержит nil, сохраняем пустую строку. }
  if Method.Code = nil then MethodName := ''
  else MethodName := TObject(Method.Data).MethodName(Method.Code);
  Reg.WriteString(Name, MethodName);
end;

{ Сохраняем в реестре каждое свойство в виде значения текущего ключа. }
procedure SavePropToRegistry(Obj: TPersistent; PropInfo: PPropInfo;
                             Reg: TRegistry);
begin
  with PropInfo^ do
    case PropType^.Kind of
      tkInteger, tkChar, tkWChar:
{ Сохраняем порядковые свойства в виде целочисленного значения. }
              Reg.WriteInteger(Name, GetOrdProp(Obj, PropInfo));
                  tkEnumeration:
{ Сохраняем имена перечислимых величин. }
              Reg.WriteString(Name, GetEnumName(PropType, GetOrdProp(Obj,
                               PropInfo))); tkFloat:
{ Сохраняем реальные числа как Doubles. }
              Reg.WriteFloat(Name, GetFloatProp(Obj, PropInfo));
          tkString, tkLString:
{ Сохраняем строки как строки. }
              Reg.WriteString(Name, GetStrProp(Obj, PropInfo));
                     tkVariant:
{ Сохраняем вариантные величины как строки. }
              Reg.WriteString(Name, GetVariantProp(Obj, PropInfo)); tkSet:
{ Сохраняем набор как подключ. }
              SaveSetToRegistry(Name, GetOrdProp(Obj, PropInfo), PropType, Reg);
                      tkClass:
{ Сохраняем класс как подключ, а его свойства в виде значений подключа. }
              SaveObjToRegistry(Name, TPersistent(GetOrdProp(Obj, PropInfo)), Reg);
                     tkMethod:
{ Сохраняем в реестре метод путем записи его имени. }
              SaveMethodToRegistry(Name, GetMethodProp(Obj, PropInfo), Reg);
    end;
end;

{ Записываем объект в реестр, сохраняя опубликованные свойства. }
procedure SaveToRegistry(Obj: TPersistent; Reg: TRegistry);
var
  PropList: PPropList;
  PropCount: Integer;
  I: Integer;
begin
{ Получаем список опубликованных свойств. }
  PropCount := GetTypeData(Obj.ClassInfo)^.PropCount;
  GetMem(PropList, PropCount*SizeOf(PPropInfo));
  try
```

```
    GetPropInfos(Obj.ClassInfo, PropList);
{ Сохраняем каждое свойство в виде значения текущего ключа. }
    for I := 0 to PropCount - 1 do
      SavePropToRegistry(Obj, PropList^[I], Reg);
  finally
    FreeMem(PropList, PropCount * SizeOf(PPropInfo));
  end;
end;

{ Сохраняем опубликованные свойства в виде значения данного ключа. Корневой улей -
  HKEY_CURRENT_USER. }
procedure SaveToKey(Obj: TPersistent; const KeyPath: string);
var
  Reg: TRegistry;
begin
  Reg := TRegistry.Create;
  try
    if not Reg.OpenKey(KeyPath, True) then
      raise ERegistryException.CreateFmt('Не могу создать ключ: %s', [KeyPath]);
    SaveToRegistry(Obj, Reg);
  finally
    Reg.Free;
  end;
end;
```

Примечание

Пример написан для Delphi 1, поэтому его необходимо адаптировать для старших версий, добавив в uses *модуль* Registry.

Работа с RegIniFile

Мне кто-нибудь покажет, как работать с реестром Windows 95?

TRegIniFile работает почти так же, как и объект TIniFile. Пример кода, который может быть вам полезен:

```
  ...
var
  Reg: TRegIniFile;
begin
  Reg := TRegIniFile.Create('');
// ReadString(Section, Ident, Default)
  Result := Reg.ReadString('\Software\MyApp', 'InitialDir', 'c:\windows');
// WriteString(Section, Ident, Value)
  Reg.WriteString('Software\MyApp', 'InitialDir', 'c:\windows');
  Reg.Free;
end;
```

[News Group]

Примечание

Добавляем `uses Registry`.

Registry, работающий со значениями типа REG_MULTI_SZ

Решение

```
//-------------------------------------------------------------------------
// TReg - расширенный TRegistry
// умеет работать со значениями типа REG_MULTI_SZ в реестре
//-------------------------------------------------------------------------
unit Reg;
{$R-,T-,H+,X+}
interface
uses Registry, Classes, Windows, Consts, SysUtils;
type
TReg = class(TRegistry)
public
procedure ReadStringList(const name : string; list : TStringList);
procedure WriteStringList(const name : string; list : TStringList);
end;
implementation
//*** TReg ***************************************************************
//-------------------------------------------------------------------------
// Запись TStringList в виде значения типа REG_MULTI_SZ в реестр
//-------------------------------------------------------------------------
procedure TReg.WriteStringList(const name : string; list : TStringList);
var
Buffer : Pointer;
BufSize : DWORD;
i, j, k : Integer;
s : string;
p : PChar;
begin
if list = nil then begin
// записываем пустой список
Buffer := nil;
BufSize := 0;
end
else begin
// подготовим буфер к записи
BufSize := 0;
for i:=0 to list.Count-1 do
inc(BufSize, Length(list[i])+1);
inc(BufSize);
GetMem(Buffer, BufSize);
k := 0;
p := Buffer;
```

```
for i:=0 to list.Count-1 do begin
s := list[i];
for j:=0 to Length(s)-1 do begin
p[k] := s[j+1];
inc(k);
end;
p[k] := chr(0);
inc(k);
end;
p[k] := chr(0);
end;
{запись в реестр}
try
if RegSetValueEx(CurrentKey, PChar(name), 0, REG_MULTI_SZ, Buffer, BufSize) <>
   ERROR_SUCCESS then
raise ERegistryException.CreateFmt( 'Ошибка записи значения %s в реестр', [name]);
finally
if Buffer<> nil then
FreeMem(Buffer);
end;
end;
//---------------------------------------------------------------------------
// Чтение TStringList в виде значения типа REG_MULTI_SZ из реестра
//---------------------------------------------------------------------------
procedure TReg.ReadStringList(const name : string; list : TStringList);
var
BufSize,
DataType : DWORD;
Len, i : Integer;
Buffer : PChar;
s : string;
begin
if list = nil then
Exit;
{чтение из реестра}
Len := GetDataSize(Name);
if Len < 1 then
Exit;
Buffer := AllocMem(Len); //GetMem(Buffer, Len+1);
if Buffer = nil then
Exit;
try
DataType := REG_NONE;
BufSize := Len; // обязательно задаем размер буфера перед вызовом!!!
if RegQueryValueEx(CurrentKey, PChar(name), nil, @DataType, PByte(Buffer),
@BufSize) <> ERROR_SUCCESS then
raise ERegistryException.CreateFmt('Ошибка чтения значения %s из реестра', [name]);
if DataType <> REG_MULTI_SZ then
raise ERegistryException.CreateResFmt(@SInvalidRegType, [name]);
{запись в TStringList}
list.Clear;
```

```
s := '';
for i:=0 to BufSize-2 do begin // BufSize-2 т.к. последние два нулевых символа
if Buffer[i] = chr(0) then begin
list.Add(s);
s := '';
end
else
s := s + Buffer[i];
end;
finally
FreeMem(Buffer);
end;
end;
end.
```

[Кондратюк Виталий]

Сообщения Windows

Как Delphi работает с Windows Messages (системными сообщениями типа `WM_xxx`*)?*

Все вы хорошо знаете, что Windows строится на принципе событийного управления. Наверняка не один раз вы создавали обработчики событий `OnKeyPress`, `OnThis`, `OnThat` и т. п. В исходном коде VCL можно легко обнаружить, что механизм работы событий в Delphi основан на обработке конкретных системных сообщений, посылаемых элементу управления. Как раз здесь и заложено главное достоинство объектно-ориентированного программирования – можно создать новый компонент на основе существующего и «научить» его обрабатывать другие, необходимые вам системные сообщения. Windows постоянно посылает сообщения в ответ на действия пользователя и ждет соответствующей реакции от приложений Delphi (и всех остальных приложений Windows), заключающейся в их «приеме» и соответствующей обработке. Delphi имеет «оболочки» для большинства системных сообщений, создавая «механизм оповещения элемента управления о приеме сообщения на его адрес» – события для компонентов, как было описано выше.

Кроме приема сообщений, также существует возможность их отправления посредством функций `SendMessage` и `PostMessage` (обе являются Win API функциями), а также метода Delphi `Perform`. Первые два требуют в качестве параметра `Handle` указывать дескриптор компонента, которому посылается сообщение, тогда как метод `Perform` принадлежит самому компоненту. Сообщения передаются в стандартную очередь системных сообщений и обрабатываются подобно другим сообщениям.

Вот тривиальный пример: требуется вставлять в `TMemo` символ `'y'` каждый раз после набора цифры `'4'`. Можно, конечно, поработать с Memo-свойством `Lines`, но это не так красиво и достаточно громоздко. Вот как выглядит например с использованием сообщений:

```
procedure TForm1.Memo1KeyPress(Sender: TObject; var Key: Char);
```

```
begin
  if Key = '4' then SendMessage(Memo1.Handle, WM_CHAR, Word('y'), 0);
end;
```

Другой пример демонстрирует работу с компонентом ComboBox. Мы хотим, чтобы он автоматически выпадал при нажатии пользователем какой-либо клавиши. Это поведение, к сожалению, нестандартно. Вот что мы делаем:

```
procedure TFormEffortRates.ComboBox1KeyDown(Sender: TObject; var Key: Word;
                                            Shift: TShiftState);
var
  iShowing: integer;
{ какой-то код, затем... }
begin
{ С помощью сообщения узнаем состояние ("раскрытость") ComboBox'а }
  iShowing := SendMessage((Sender as TComboBox).Handle, CB_GETDROPPEDSTATE, 0, 0);
  if iShowing = 0 then
{ раскрываем ComboBox }
    SendMessage((Sender as TComboBox).Handle, CB_SHOWDROPDOWN, 1,0);
end;
```

Другой пример демонстрирует способ получения строки и колонки TMemo. Для этого трюка мы воспользуемся API. Вот реализация этого метода (возможно, не самый эффективный метод, но он приведен ради демонстрации работы сообщений):

```
function TForm1.GetMemoColumn(const TheMemo: TMemo): integer;
begin
  Result := TheMemo.SelStart -
            (SendMessage(TheMemo.Handle, EM_LINEINDEX, GetMemoLine(TheMemo), 0));
end;

function TForm1.GetMemoLine(const TheMemo: TMemo): integer;
begin
  Result := SendMessage(TheMemo.Handle, EM_LINEFROMCHAR, TheMemo.SelStart, 0);
end;
```

Список и описание всех сообщений приведены в электронной справке по API. Сообщения API позволяют тонко управлять пользовательскими приложениями, выполняя именно те задачи, которые необходимо решить (метод «точечной наводки»). Программист должен лишь выбрать цель и передать свою «просьбу» элементу управления (или обрабатывать такие сообщения самостоятельно).

Сообщение для всех главных окон

Как послать сообщение всем главным окнам в Windows?

Решение

```
var
```

```
  FM_FINDPHOTO: Integer;
// Для того чтобы использовать hwnd_Broadcast, нужно сперва зарегистрировать
// уникальное сообщение.
initialization
  FM_FindPhoto := RegisterWindowMessage('MyMessageToAll');
// Чтобы поймать это сообщение в другом приложении (приемнике), нужно перекрыть
// DefaultHandler
procedure TForm1.DefaultHandler(var Message);
begin
  with TMessage(Message) do begin
    if Msg = Fm_FindPhoto then MyHandler(WPARAM, LPARAM)
    else inherited DefaultHandler(Message);
  end;
end;
// А теперь можно в приложении-передатчике
  SendMessage(HWND_BROADCAST, FM_FINDPHOTO, 0, 0);
```

[Nomadic]

Центрирование информационного диалога (MessageDlg)

Как вывести сообщение в центр экрана?

Решение

```
unit kns;
{$R-}

interface

uses
  Forms, Dialogs;

{ Центрирование информационного диалога }
function MessageDlgCtr(const Msg: string; DlgType: TMsgDlgType;
                       Buttons: TMsgDlgButtons; HelpCtx: Longint): Integer;
implementation

uses
  Consts;

{ Функция MessageDlg располагает диалог над центром активного окна }
function MessageDlgCtr(const Msg: string; DlgType: TMsgDlgType;
                       Buttons: TMsgDlgButtons; HelpCtx: Longint): Integer;
begin
  with CreateMessageDialog(Msg, DlgType, Buttons) do
    try
      HelpContext := HelpCtx;
      Left := Screen.ActiveForm.Left + (Screen.ActiveForm.Width div 2)
              - (Width div 2);
```

```
        Top := Screen.ActiveForm.Top + (Screen.ActiveForm.Height div 2)
               - (Height div 2);
        Result := ShowModal;
      finally
        Free;
      end;
  end;
  end.
```

MessageDlg в обработчике OnExit

Я пытаюсь использовать MessageDlg *в обработчике* OnExit *компонента* TEdit. *При показе диалогового окна пользователь нажимает одну из кнопок, после чего, по идее, должно возникнуть событие* OnEnter *компонента, но оно не возникает! Если вызов диалога сопровождается комментарием, событие* OnEnter *инициализируется верно. В любом случае, событие* OnExit *завершает весь код.*

Фактически (в момент показа диалога), фокус имеет поле редактирование, но курсор при этом не выводится. Передавая фокус «вперед» и снова «назад», вы получите желаемый результат.

Например: в обработчике события OnExit поля редактирования после вызова MessageDlg попробуйте вызвать следующие функции:

```
PostMessage(Handle, WM_NEXTDLGCTL, 0, 0) ;
PostMessage(Handle, WM_NEXTDLGCTL, 1, 0) ;
```

[News Group]

Текст на кнопках MessageDlg

Как сменить текст на кнопках диалогового окна MessageDlg*? Английский язык для текста кнопок пользователь хочет заменить на родной.*

Текст кнопок извлекается из списка строк, расположенных в файле ...BORLAND\DELPHI5\SOURCE\VCL\CONSTS.PAS. Отредактируйте его, после чего пересоберите VCL.

[News Group]

Дополнение:

Можно ничего не менять. Вместо MessageDlg вызовите функцию Windows MessageBox. И, если Windows адаптирована под национальный язык, то надписи на кнопках в диалоговых окнах будут отображаться на языке адаптации Windows.

[VS]

Использование Shell_NotifyIcon

У меня есть несколько вопросов по функции Shell_NotifyIcon*:*

– Как добавить пиктограмму в системную область панели задач?

– Как изменить пиктограмму?

– Как удалить пиктограмму?

Чтобы получать сообщения, надо добавить флаг NIF_MESSAGE в notify-структуру и записать то сообщение, которое требуется послать.

```
procedure TMainForm.UpdateTaskBar;    // обновление области системных пиктограмм
var
  NotifyData: TNotifyIconData;

begin
  with NotifyData do begin            // устанавливаем структуру данных
    cbSize := SizeOf(TNotifyIconData);
    Wnd := MyForm.Handle;
    uID := 0;
// установки, которые необходимо изменить
    uFlags := NIF_ICON or NIF_MESSAGE or NIF_TIP;
    uCallbackMessage := WM_MY_MESSAGE;             // возвращаемое сообщение
    hIcon := hMyIcon;
    szTip := 'Текст всплывающей подсказки';        // и соответствующий текст
  end;
  Shell_NotifyIcon(dwMessage, @NotifyData);       // теперь производим обновление
end;
```

WM_MYMESSAGE — определенное пользователем сообщение. Обычно определяется так:

```
const
  WM_MYMESSAGE = WM_USER + <какой-то номер, может быть нулем>;
```

Примечание

Необходимо присоединить модуль ShellAPI.

ProcessMessages

Для чего нужен ProcessMessages*?*

Многие начинающие программисты не знают о методе Application.ProcessMessages() и сталкиваются с проблемами, которых не могут объяснить. Например, мы хотим написать что-то на форме и через 5 секунд стереть. Нужно сделать так:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Label1.Caption := 'Started';
```

```
    Application.ProcessMessages();
    Sleep(1000);
    Label1.Caption := 'Finished';
  end;
```

Попробуйте убрать `Application.ProcessMessages()`, и вы никогда не увидите слова «Started». Это связано с тем, что Windows ждет, пока накопятся сообщения в очереди, а не обрабатывает их сразу. `Application.ProcessMessages()` заставляет обработать все сообщения, которые накопились в данный момент. Подробности можно найти в файлах Справки.

Еще это можно применить так:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  i: integer;
begin
  for i := 1 to 100000 do Application.ProcessMessages();
end;
```

если убрать `ProcessMessages()`, то, пока выполняется цикл, мы не сможем сдвинуть или свернуть форму.

[Vozny Alexander]

Избавление от системного окна с ошибкой

Как избавиться от окна с сообщением о системной ошибке?

Имеется функция Windows API, преобразующая «уродливые» Windows-окна, информирующие об ошибках, в привычные исключения Delphi, что, по крайней мере, более эстетично и полезно (поскольку в этом случае ошибка может быть перехвачена и обработана вашим приложением).

```
SetErrorMode(SEM_FAILCRITICALERRORS);
```

Эта функция сообщает Windows о том, что вызвавшая ошибку программа будет сама обрабатывать критические ошибки.

[News Group]

Функции InputBox и InputQuery

Данный совет демонстрирует три очень мощных и полезных процедуры, интегрированных в Delphi.

В диалоговых окнах `InputBox` и `InputQuery` пользователь может вводить данные.

Функция `InputBox` используется в том случае, когда не имеет значения, что пользователь выбирает для закрытия диалогового окна – кнопку `OK` или `Cancel` (или нажатие клавиши <Esc>). Для того чтобы узнать это, вызовите функцию `InputQuery`.

ShowMessage — другой простой путь отображения сообщения для пользователя.

```
procedure TForm1.Button1Click(Sender: TObject);

var
  s, s1: string;
  b: boolean;

begin
  s := Trim(InputBox('Новый пароль', 'Пароль', 'masterkey'));
  b := s <> '';
  s1 := s;
  if b then b := InputQuery('Повторите пароль', 'Пароль', s1);
  if not b or (s1 <> s) then ShowMessage('Пароль неверен');
end;
```

Примечание

Trim — *функция отсекания лишних пробелов справа и слева.*

Работа со шрифтами

Как определить, большие или маленькие шрифты используются в системе?

Решение

```
function SmallFonts: boolean;
{ возвращает true, если установлены маленькие шрифты, и false если большие }
var
  DC: HDC; { используется для проверки количества доступных цветов }
begin
  DC := GetDC(0);
  Result := (GetDeviceCaps(DC, LOGPIXELSX) = 96);
{ Если используются большие шрифты, LOGPIXELSX будет равен 120 }
  ReleaseDC(0, DC);
end;
```

Добавляем новый шрифт

Как добавить шрифт в Windows 9x?

Для начала нужно скопировать шрифт в системную папку \Windows\Fonts. Затем добавить соответствующее значение в реестр HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Fonts, вызвать функцию AddFontRecource() и передать сообщение WM_FONTCHANGE. Затем выполнить RemoveFontRecource() для снятия флажка (блокировки) с системного ресурса и второй раз послать сообщение WM_FONTCHANGE.

```
uses
  Registry;

procedure TForm1.Button1Click(Sender: TObject);
```

```
var
  reg: TRegistry;
  b: bool;
begin
  CopyFile('C:\DOWNLOAD\FP000100.TTF', 'C:\WINDOWS\FONTS\FP000100.TTF', b);
  reg := TRegistry.Create;
  reg.RootKey := HKEY_LOCAL_MACHINE;
  reg.LazyWrite := false;
  reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\Fonts', false);
  reg.WriteString('TESTMICR (TrueType)','FP000100.TTF');
  reg.CloseKey;
  reg.free;
{ Добавляем шрифт }
  AddFontResource('c:\windows\fonts\FP000100.TTF');
  SendMessage(HWND_BROADCAST, WM_FONTCHANGE, 0, 0);
{ Удаляем шрифт }
  RemoveFontResource('c:\windows\fonts\FP000100.TTF');
  SendMessage(HWND_BROADCAST, WM_FONTCHANGE, 0, 0);
end;
```

Использование шрифтов в приложении

Как установить и использовать шрифты во время работы программы? Я пробовал копировать файлы *.ttf *в директорию пользователя* \windows\system *(Win16), но мое приложение так и не смогло их увидеть и выбрать для дальнейшей работы.*

Ниже приведен код для Delphi 1, который динамически устанавливает шрифты, загружаемые только во время работы приложения. Можно расположить файлы шрифтов в каталоге приложения. Они будут инсталлированы при загрузке формы и выгружены при ее закрытии. Возможно, придется модифицировать код для работы с Delphi 2, поскольку там используются вызовы Windows API, которые могут измениться. Многоточие в коде (. . .) означает, что в этом месте может располагаться какой-либо код, не относящийся к существу вопроса.

Замените MYFONT на реальное имя файла шрифта.

```
type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
   ...
  private
    bLoadedFont: boolean;
  end;

procedure TForm1.FormCreate(Sender: TObject);
var
  sAppDir: string;
 sFontRes: string;
begin
  sAppDir := Application.ExeName;
```

```
  sAppDir := Copy(sAppDir, 1, rpos('\', sAppDir));
  sFontRes := sAppDir + 'MYFONT.FOT';
  if not FileExists(sFontRes) then begin
    sFontRes := sFontRes + #0;
    sFont := sAppDir + 'MYFONT.TTF' + #0;
    CreateScalableFontResource(0, @sFontRes[1], @sFont[1], nil);
  end;
  sFontRes := sAppDir + 'MYFONT.FOT';
  if FileExists(sFontRes) then begin
    sFontRes := sFontRes + #0;
    if AddFontResource(@sFontRes[1]) = 0 then bLoadedFont := false

    else begin
      bLoadedFont := true;
      SendMessage(HWND_BROADCAST, WM_FONTCHANGE, 0, 0);
    end;
  end;
   ...
end;
procedure TForm1.FormDestroy(Sender: TObject);
var
  sFontRes: string;
begin
  if bLoadedFont then begin
    sFontRes := sAppDir + 'MYFONT.FOT' + #0;
    RemoveFontResource(@sFontRes[1]);
    SendMessage(HWND_BROADCAST, WM_FONTCHANGE, 0, 0);
  end;
end;
```

Примечание

Электронная справка по продукту InstallShield сообщает, что в системах Windows 9x/NT файл FOT *не нужен. Нужен только файл* TTF.

В результате процедура FormCreate выглядит так:

```
var
  sAppDir, sFontRes: string;
begin
{...другой код...}
  sAppDir := ExtractFilepath(Application.ExeName);
  sFontRes := sAppDir + 'MYFONT.TTF';
  if FileExists(sFontRes) then begin
    sFontRes := sFontRes + #0;
    if AddFontResource(@sFontRes[1]) = 0 then bLoadedFont := false
    else begin
      bLoadedFont := true;
      SendMessage(HWND_BROADCAST, WM_FONTCHANGE, 0, 0);
    end;
  end;
```

```
  ...
end;
```

А FormDestroy так:

```
var
  sFontRes, sAppDir: string;
begin
{...другой код...}
  if bLoadedFont then begin
    sAppDir := ExtractFilepath(Application.ExeName);

    sFontRes := sAppDir + 'MYFONT.TTF' + #0;
    RemoveFontResource(@sFontRes[1]);
    SendMessage(HWND_BROADCAST, WM_FONTCHANGE, 0, 0);
  end;
{...другой код...}
end;
```

Для упрощения создана простая функция, совмещающая обе этих задачи. Она возвращает логическое значение, говорящее об успехе или, наоборот, о неудаче операции загрузки или выгрузки шрифта.

```
function LoadFont(sFontFileName: string; bLoadIt: boolean): boolean;
var
  sFont, sAppDir, sFontRes: string;
begin
  result := true;
  if bLoadIt then begin
{ Загрузка шрифта }
    if FileExists(sFontFileName) then begin
      sFontRes := sFontFileName + #0;
      if AddFontResource(@sFontRes[1]) = 0 then result := false
      else SendMessage(HWND_BROADCAST, WM_FONTCHANGE, 0, 0);
    end;
  end else begin

{ Выгрузка шрифта }
    sFontRes := sFontFileName + #0;
    result := RemoveFontResource(@sFontRes[1]);
    SendMessage(HWND_BROADCAST, WM_FONTCHANGE, 0, 0);
  end;
end;
```

Свойства шрифта Style/Color в виде строки

Как получить значение Font.Style *и* Font.Color *в виде строки? Я хотел бы присвоить его заголовку компонента* Label*, но* Style *и* Color *не являются строковыми величинами.*

Решение

```
const
  fsTextName: array[TFontStyle] of string[11] =
```

```
                ('fsBold', 'fsItalic', 'fsUnderline', 'fsStrikeOut');

    fpTextName: array[TFontPitch] of string[10] =
                ('fpDefault', 'fpVariable', 'fpFixed');
```

Затем в коде:

```
var
  TFPitch: TFontPitch;
  TFStyle: TFontStyle;
  FString: String;
  ...
  FString := '';
  for TFStyle := fsBold to fsStrikeOut do
    if TFStyle in Canvas.Font.Style then
      Fstring := Fstring+fsTextName[TFStyle] + ',';
  if Fstring <> '' then
    dec(FString[0]);        { убираем лишний разделитель ',' }
  something := FString;
  FString := fpTextName[Canvas.Font.Pitch];
  something := FString;
  ...
```

Примерно так же нужно поступить и с именованными цветами типа `TColor`.

[News Group]

Включение шрифта как ресурса в *.EXE

Можно ли включить шрифт в файл ресурсов?

Создайте *.RC-файл, описывающий шрифт:

```
MY_FONT ANYOL1 "Bauhs93.ttf"
```

Первые два параметра могут быть любыми. Они будут использоваться в программе позже.

Затем создайте .RES-файл с помощью компилятора командной строки BRCC32.EXE, поставляемого с Delphi. Если файл на предыдущем этапе был назван MyFont.rc, то командная строка в сеансе DOS должна выглядеть так:

```
BRCC32 MyFont
```

Программа добавит в компилируемый файл созданный ресурс .rc и создаст файл с тем же именем, но с расширением .RES: MyFont.res

В программе добавьте директиву компилятора, чтобы включить вновь созданный файл:

```
{$R MyFont.res}
```

Правильным будет разместить его в секции реализации после строчки

```
{$R *.DFM}.
```

Добавьте процедуру создания файла из ресурса, делающую шрифт доступным для использования.

Решение

```
procedure TForm1.FormCreate(Sender: TObject);
var
  Res: TResourceStream;
begin
  Res := TResourceStream.Create(hInstance, 'MY_FONT', PChar('ANYOL1'));
  Res.SaveToFile('Bauhs93.ttf');
  Res.Free;
  AddFontResource(PChar('Bauhs93.ttf'));
  SendMessage(HWND_BROADCAST, WM_FONTCHANGE, 0, 0);
end;
```

Теперь можно использовать данный шрифт в своем приложении:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Button1.Font.Name := 'Bauhaus 93';
end;
```

Предупреждение

Приведенный пример не предусматривает никакой проверки и защиты от возможных ошибок.

Обратите внимание – имя файла не такое же, как имя шрифта. Предполагается, что вы знаете имя шрифта и имя его TTF*-файла. Можно определить это, дважды щелкнув по файлу в окне Проводника.*

Изменение шрифта всплывающей подсказки

Как изменить шрифт всплывающей подсказки?

Решение

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm1 = class(TForm)
      Edit1: TEdit;
      procedure FormCreate(Sender: TObject);
  end;
```

```
var
  Form1: TForm1;

implementation

{$R *.DFM}

type
  TMyHintWindow = Class(THintWindow)
    constructor Create (AOwner: TComponent); override;
  end;

constructor TMyHintWindow.Create(AOwner: TComponent);
begin
  inherited Create(Aowner);
  Canvas.Font.Name := 'Times New Roman';
  Canvas.Font.Size := 14;
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
  Application.ShowHint := False;
  HintWindowClass := TMyHintWindow;
  Application.ShowHint := True;
end;

end.
```

Масштабирование формы и шрифтов

Когда мои программы работают на системах с установленными маленькими шрифтами, выводимые ими формы приобретают странный вид. К примеру, расположенные на форме компоненты `Label`*, становятся малы для размещения указанного теста, обрезая его в правой или нижней части.* `StringGrid` *не осуществляет положенного выравнивания и т. д.*

Попробуйте следующий код. Он масштабирует как размер формы, так и размер шрифтов.

```
unit geScale;

interface

uses
  Forms, Controls;

procedure geAutoScale(MForm: TForm);
implementation

type
  TFooClass = class(TControl);    { необходимо выяснить защищенность }
```

```
procedure geAutoScale(MForm: TForm);
const
  cScreenWidth: integer = 800;
  cScreenHeight: integer = 600;
  cPixelsPerInch: integer = 96;
  cFontHeight:integer = -11;   { В режиме проектирования значение из Font.Height }

var
  i: integer;

begin
{ Установите в Инспекторе объектов свойство Scaled TForm в FALSE. Следующая
  программа масштабирует форму так, чтобы она выглядела одинаково вне зависимости
  от размера и разрешения экрана. Расположенный ниже участок кода проверяет,
  отличается ли размер экрана во время выполнения от размера во время
  проектирования. Если да, Scaled устанавливается в True и компоненты снова
  масштабируются так, чтобы они выводились в той же позиции экрана, что и во время
  проектирования. }
  if (Screen.Width > cScreenWidth) or
     (Screen.PixelsPerInch <> cPixelsPerInch) then begin
    MForm.Scaled := True;
    MForm.Height := MForm.Height * Screen.Height div cScreenHeight;
    MForm.Width := MForm.Width * Screen.Width div cScreenWidth;
    MForm.ScaleBy(screen.Width, cScreenWidth);
  end;

{ Этот код проверяет, отличается ли размер шрифта во время выполнения от размера во
  время проектирования. Если во время выполнения PixelsPerInch формы отличается от
  PixelsPerInch во время проектирования, шрифты снова масштабируются так, чтобы
  форма не отличалась от той, которая была во время разработки. Масштабирование
  производится исходя из коэффициента, получаемого путем деления значения
  Font.Height во время проектирования на Font.Height во время выполнения. Font.Size
  в этом случае работать не будет, так как это может дать результат больший, чем
  текущие размеры компонентов, при этом текст может оказаться за границами области
  компонента. Например, форма создана при размерах экрана 800x600 с установленными
  маленькими шрифтами, имеющими размер Font.Size = 8. Когда вы запускаете в системе
  с 800x600 и большими шрифтами, Font.Size также будет равен 8, но текст будет
  большим, чем при работе в системе с маленькими шрифтами. Данное масштабирование
  позволяет иметь один и тот же размер шрифтов при различных установках системы. }

  if (Screen.PixelsPerInch <> cPixelsPerInch) then begin
    for i := MForm.ControlCount - 1 downto 0 do
      TFooClass(MForm.Controls[i]).Font.Height :=
                                  (MForm.Font.Height div cFontHeight) *
                                   TFooClass(MForm.Controls[i]).Font.Height;
  end;

end;

end.
```

PopupComponent и шрифты

Font является защищенным свойством, введенным в классе TControl иерархии компонентов. Можно получить доступ к свойству Font только после создания элемента управления (класса) с опубликованным свойством Font. Есть одна хитрость, позволяющая обойти это и иметь доступ к свойству Font любого потомка TControl, но будьте осторожны, т. к. при изменении свойства Font класса, который не хочет, чтобы его изменяли, может случиться любая неприятность.

Хитрость заключается в создании в секции implementation следующего описания класса:

```
type
  THack = class(TControl)
  public
    property Font;
end;
```

Затем код может выглядеть приблизительно так:

```
if Popup1.PopupControl is TControl then
  if FontDialog1.Execute then
    THack(Popup1.PopupControl).Font := FontDialog1.Font;
```

[News Group]

Прием файлов из Program Manager

Для того чтобы реализовать функцию Drag&Drop из File Manager (Windows 3.1x) или из Проводника (Windows 9x), нужно «зарегистрировать» дескриптор принимаемой формы (свойство Handle) Windows API функцией

```
DragAcceptFiles(Handle, True);
```

После этого можно получать сообщения WM_DROPFILES при перетаскивании файлов из File Manager на форму. Для «отключения» этой характеристики необходимо вторично вызвать ту же API-функцию, но со вторым параметром равным False.

Для получения имен перетаскиваемых файлов необходимо воспользоваться Windows-функцией DragQueryFile, а для получения информации об окончании операции Drag&Drop воспользуйтесь функцией DragFinish. DragQueryPt подскажет, в каком месте формы были «брошены» перетаскиваемые файлы.

Необходимо принимать файлы даже при свернутой форме? Для этого нужно дополнительно создать обработчик события OnMessage объекта Application. Нижеприведенный пример предполагает, что на форме имеется компонент ListBox, свойство Align которого установлено в alClient:

```
{ Private declarations }
  procedure WMDropFiles(var Msg: TWMDropFiles); message WM_DROPFILES;
```

```
    procedure AppOnMessage(var Msg: TMsg; var Handled: Boolean);
  ...

implementation

uses
  ShellAPI;

procedure TForm1.WMDropFiles(var Msg: TWMDropFiles);
var
  N: Word;
  Buffer: array[0..80] of char;
begin
  with Msg do begin
    for N := 0 to DragQueryFile(Drop, $FFFFFFFF, Buffer, SizeOf(Buffer)) - 1 do begin
      DragQueryFile(Drop, N, Buffer, SizeOf(Buffer));
      ListBox1.Items.Add(StrPas(Buffer));
    end;
    DragFinish(Drop);
  end;
end;

procedure TForm1.AppOnMessage(var Msg: TMsg; var Handled: Boolean);
var
  WMD: TWMDropFiles;
begin
  if Msg.message = WM_DROPFILES then begin
    MessageBeep(0);
    WMD.Msg := Msg.message;
    WMD.Drop := Msg.wParam;
    WMD.Unused := Msg.lParam;
    WMD.Result := 0;
    WMDropFiles(WMD);
    Handled := True;
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  DragAcceptFiles(Handle, True);
  DragAcceptFiles(Application.Handle, True);
  Application.OnMessage := AppOnMessage;
end;
```

Drag & Drop с Windows 95 Explorer

Как заставить окно принимать файлы, перетаскиваемые из проводника?

Решение

```
procedure TForm1.CreateParams(var Params: TCreateParams);
```

```
begin
  inherited
  CreateParams(Params);
{ сделаем окно способным принимать файлы }
  Params.ExStyle := Params.ExStyle or WS_EX_ACCEPTFILES;
end;

procedure TForm1.WMDropFiles(var Message: TWMDropFiles);
var
  aFile: array [0..255] of Char;
// FilesCount : Integer;

begin
  inherited;
{ так можно узнать, сколько файлов перетягивается }
// FilesCount := DragQueryFile(Message.drop, $FFFFFFFF, nil, 0);
begin
{ здесь можно организовать цикл открытия всех перетаскиваемых файлов
  for N := 0 to FilesCount - 1 do DragQueryFile(Message.drop, N, aFile, 256);
  а в данном случае открывается только первый файл в списке }
    DragQueryFile(Message.drop, 0, aFile, 256);
    Memo1.Lines.LoadFromFile(aFile);
  end;
  DragFinish(Message.Drop);
end;

procedure TForm1.FormCreate(Sender: TObject); { Form1.OnCreate }
begin
{ сделаем окно неравнодушным к пролетающим над ним файлам }
  DragAcceptFiles(Handle, True);
end;
```

[News Group]

Примечание

В uses *необходимо добавить* ShellAPI*, а определение* TForm1 *должно выглядеть так:*

```
type
  TForm1 = class(TForm)
    Memo1: TMemo;
    procedure FormCreate(Sender: TObject);
  private
    procedure CreateParams(var Params: TCreateParams);
    procedure WMDropFiles(var Message: TWMDropFiles); message WM_DROPFILES;
  end;
```

Drag & Drop для ярлыков

Как принимать ярлыки при перетаскивании их на активный элемент?

Решение

```
TForm1 = class(TForm)
  ...
private
  procedure WMDropFiles(var M : TWMDropFiles); message WM_DROPFILES;
  ...
end;

var
  Form1: TForm1;

implementation

uses
  ShellAPI, ComObj, ShlObj, ActiveX;

procedure TForm1.FormCreate(Sender: TObject);
begin
  ...
  DragAcceptFiles(Handle, True);
  ...
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  ...
  DragAcceptFiles(Handle, False);
  ...
end;

procedure TForm1.WMDropFiles(var M: TWMDropFiles);
var
  hDrop: Cardinal;
  n: Integer;
  s: string;
begin
  hDrop := M.Drop;
  n := DragQueryFile(hDrop, 0, nil, 0);
  SetLength(s, n);
  DragQueryFile(hDrop, 0, PChar(s), n + 1);
  DragFinish(hDrop);
  M.Result := 0;
  FileOpen(s);
end;

function ResolveShortcut(Wnd: HWND; ShortcutPath: string): string;
var
  obj: IUnknown;
  isl: IShellLink;
  ipf: IPersistFile;
  pfd: TWin32FindDataA;
begin
```

```
  Result := '';
  obj := CreateComObject(CLSID_ShellLink);
  isl := obj as IShellLink;
  ipf := obj as IPersistFile;
  ipf.Load(PWChar(WideString(ShortcutPath)), STGM_READ);
  with isl do begin
    Resolve(Wnd, SLR_ANY_MATCH);
    SetLength(Result, MAX_PATH);
    GetPath(PChar(Result), Length(Result), pfd, SLGP_UNCPRIORITY);
    Result := PChar(Result);
  end;
end;
procedure TForm1.FileOpen(FileName: string);
var
  DocName: string;
begin
  if CompareText(ExtractFileExt(FileName), '.lnk') = 0 then
    FileName := ResolveShortcut(Application.Handle, FileName);
  DocName := ExtractFileName(FileName);
  Caption := Application.Title + ' - ' + DocName;
  ...
end;
```

Drag&Drop с минимизированным приложением

Необходимо понимать, что окно главной формы минимизированного приложения не работает. Если при проверке окна главной формы вы обнаруживаете, что оно имеет прежний размер, не удивляйтесь – оно просто невидимо. Пиктограмма минимизированного Delphi-приложения принадлежит объекту `Application`, дескриптор окна которого – `Application.Handle`.

Вот пример кода из программы, который с помощью компонента `CheckBox` проверяет возможность принятия перетаскиваемых файлов минимизированным приложением:

```
{ Вызывается, только если TApplication не получает drag/drop }
procedure TForm1.WMDropFiles(var Msg: TWMDropFiles);
begin
  RecordDragDrop(Msg.Drop, False);           { внутренняя функция }
  Msg.Result := 0;
end;

{ Когда активно, получаем сообщения WM_DROPFILES, посылаемые форме или
  минимизированному приложению }
procedure TForm1.AppOnMessage(var Msg: TMsg; var Handled: Boolean);
begin
  if Msg.message = WM_DROPFILES then begin
    RecordDragDrop(Msg.wParam, Msg.hWnd = Application.Handle);
    Handled := True;
  end;
end;
```

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  DragAcceptFiles(Handle, True);
  DragAcceptFiles(Application.Handle, False);
  Application.OnMessage := NIL;
end;
```

[News Group]

Перетаскивание элементов управления с рамкой контура

Как перетаскивать элементы управления с контурной рамкой, «приклеенной» к курсору, по форме?

В общих чертах, нужно рисовать на всей поверхности формы и даже рабочего стола, для чего необходимо сделать растровую копию окна или рабочей поверхности и рисовать на ней.

Начните с новой формы. Опустите на нее компонент Notebook и установите его свойство Align в alClient. Разработайте форму на первой странице компонента Notebook. Создайте вторую страницу в Notebook, поместите туда Paintbox и установите его свойство Align в alClient. Далее добавьте нижеследующие строчки в секцию Private формы:

```
Img: TBitmap;
DragX, DragY, DragW, DragH, XOff, YOff: Integer;
```

В обработчике формы OnCreate:

```
Img := TBitmap.Create;
```

Для всех перетаскиваемых компонентов обработчик события OnMouseDown:

```
if not (ssShift in Shift) then Exit;
Img := GetFormImage;
Notebook1.PageIndex := 1;
with Sender AS TControl do begin
  DragW := Width;
  DragH := Height;
  XOff := X;
  YOff := Y;
  BeginDrag(True);
end;
```

Для всех перетаскиваемых компонентов обработчик события EndDrag:

```
Notebook1.PageIndex := 0;
with Sender as Tcontrol do begin
  Left := X - XOff;
  Top := Y - YOff;
end;
```

Поместите следующую строку в обработчик события OnPaint компонента PaintBox:

```
PaintBox1.Canvas.Draw(0, 0, Img);
```

И, наконец, поместите следующую строчку в обработчик OnDragOver компонента PaintBox:

```
if (X = DragX) and (Y = DragY) then Exit;
with PaintBox1.Canvas do begin
  DrawFocusRect(Bounds(DragX - XOff, DragY - YOff, DragW, DragH);
  DragX := X; DragY := Y;
  DrawFocusRect(Bounds(DragX - XOff, DragY - YOff, DragW, DragH);
end;
```

Примечание

В данном случае необходимо перемещать элемент, удерживая клавишу <Shift> *(*ssShift *в* Shift*).*

Вопросы по Drag & Drop

Я пытаюсь перемещать TPanel*, используемую в качестве панели инструментов, и всегда почему-то получаю пиктограмму с перечеркнутым кругом. Я понимаю, что это означает невозможность перетаскивания. К сожалению, в документации я не нашел, как решить эту проблему. Я пробовал и ручные, и автоматические настройки (*DragMode = dmManual/dmAutomatic*), но все без толку.*

Причина появления курсора crNoDrop в том, что элемент управления под курсором не готов принять перетаскиваемый компонент. Чтобы исправить эту ситуацию, дважды щелкните (в Инспекторе объектов) на событии формы или компонента OnDragOver и установите параметр Accept:

```
procedure TForm1.FormDragOver(Sender, Source: TObject; X, Y: Integer;
                              State: TDragState; var Accept: Boolean);
begin
  Accept := True;
end;
```

*Есть ли возможность во время операции перетаскивания (*PaintBox1DragOver*) работать с элементами управления, находящимися под* PaintBox *с тем, чтобы они также изменяли курсор и также могли бы принимать перетаскиваемый элемент?*

Можно как-то определить, с каким конкретно элементом управления, расположенным под PaintBox*, взаимодействует в данный момент перетаскиваемый элемент?*

Можно получить координаты в методе OnDragOver при сравнении BoundsRect с областью компонентов. Например, вы не хотите принимать перетаскиваемый компонент кнопкой, перекрывающей любую другую имеющуюся кнопку. В обработчике OnDragOver напишите примерно следующее:

```
for N := 0 to ComponentCount - 1 do
```

```
  if COmponents[N] is TButton then
    if IntersectRect(DummyRect, TControl(Components[N]).BoundsRect,
          (Bounds(X - XOff, Y - YOff, DragW, DragH)) > 0 then
       Accept := False;
```

В этом случае курсор будет изменяться на перечеркнутый кружок при пересечении перетаскиваемым элементом границы интересующей вас кнопки. Необходимо добавить аналогичную логику в обработчике `EndDrag` или `OnDragDrop` компонента `PainBox`.

Перемещение формы не за заголовок

Решение

```
TForm1 = class(TForm)

private
  procedure WMNCHitTest(var M: TWMNCHitTest); message wm_NCHitTest;
end;
{ реализация обработчика события }
procedure TForm1.WMNCHitTest(var M: TWMNCHitTest);
begin
  inherited;                        { вызвали унаследованный обработчик сообщения, }
  if M.Result = htClient then       { щелкнув в клиентской области окна? }
    M.Result := htCaption;          { если так, то мы заставили Windows думать, }
                                    { что щелчок был произведен по заголовку окна. }
end;
```

Это заставляет Windows думать, что курсор мыши находится в области заголовка окна. Но это может повлечь за собой другую проблему, поскольку предполагается, что мышь будет считаться расположенной в области заголовка при любом ее нахождении в области клиента. Тем не менее, это решение элегантно, поскольку при перетаскивании формы ее границы изменяются на «резиновые». Если описаный способ не помогает, попробуйте работать с другим сообщением, которое может дать тот же результат.

- Выключите все `BorderIcons` формы.
- Убедитесь в том, что заголовок является пустой строкой.

  ```
  BorderStyle = bsNone
  ```

- Перекройте процедуру формы `CreateParams`, как показано ниже:

```
type
  TForm1 = class(TForm)
  ...

  protected
    procedure CreateParams(var Params: TCreateParams); override;
  ...

end;
```

```
implementation

procedure TForm1.CreateParams(var Params: TCreateParams);
begin
  inherited CreateParams(Params);
  with Params do
    Style := Style or ws_Border or ws_ThickFrame;
end;

end.
```

Использование обработчиков событий мыши:

```
var
  Moving: Boolean;
  OldX, OldY: Integer;
  ...

procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
                               Shift: TShiftState; X, Y: Integer);
begin
  if Button = mbLeft then begin          { нас интересует только левая кнопка }
    OldLeft := X;                        { сохраняем текущую позицию }
    OldTop := Y;
    Moving := True;
  end;
end;

procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
{ Если необходимо переместить окно относительно своей оригинальной позиции }
  if Moving then Self.SetBounds(Self.Left + X - OldLeft, Self.Top + Y - OldTop,
                                Self.Width, Self.Height);
end;

procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton;
                             Shift: TShiftState; X, Y: Integer);
begin
  if Button = mbLeft then
    Moving := False;                     { Останавливаем перемещение }
end;
```

Не забудьте назначить эти методы каждому элементу управления вашей формы:

```
Self.Panel1.OnMouseDown := Self.OnMouseDown;
```

Рассуждения о потоках

Умножение потоков не умножает CPU

Если на вашем компьютере установлено более одного процессора, это замечательно. В противном случае, множество параллельных потоков не ускоряет работу приложения, поскольку в отдельно взятый промежуток времени возможно выполнение только одного потока. Кроме того, чем больше потоков, тем больше нагрузка на систему из-за затрат на переключение между ними. Если проект имеет более двух постоянно работающих потоков, то такая мультизадачность не сделает программу быстрее. Обратное же утверждение истинно!

Использование потоков для перекрытия обработки I/O

В этом случае нужно вначале понять, для чего необходим поток. Поток, осуществляющий обработку, может проигнорировать необходимость быстро реагировать на I/O-запросы системы. Сам же он может не реагировать на эти запросы, полностью доверив это операционной системе. Потоки позволяют программе отзываться на просьбы пользователя и устройств, но при этом (в том числе) сильно загружают процессор. Потоки позволяют компьютеру одновременно обслуживать множество устройств, и созданный вами поток, отвечающий за обработку специфического устройства, в качестве минимума может потребовать столько времени, сколько системе необходимо для обработки запросов всех устройств.

Расставляйте приоритеты потоков в зависимости от задач

Потокам можно назначить определенный приоритет для того, чтобы наименее значимые процессы выполнялись в фоновом режиме. Это путь честного разделения ресурсов CPU. Просто необходимо осознать тот факт, что процессор один, а потоков, эдаких «мальчиков на побегушках», ну очень много. Если в программе главная процедура передает нечто для обработки в низкоприоритетный поток, то сама программа становится просто неуправляемой.

Конкуренция за обладание ресурсами

Потоки хорошо работают, когда они независимы. Но они начинают работать непродуктивно (становятся не лучше (а может быть, хуже), чем очередь системных сообщений), когда они вынуждены синхронизироваться для доступа к общим ресурсам. Блокировка и критические секции отнюдь не добавляют бодрости системе.

Борьба за пользовательский интерфейс

Один из наиболее горячо оспариваемых ресурсов в системе – интерфейс пользователя. В конце концов, сама Windows представляет собой набор высокоприоритетных задач, являющихся вашими прямыми конкурентами и наравне с вами лезущих в драку за обладание ресурсами. И если несколько потоков одновременно попытаются обновить одно и то же окно, то визуальный хаос вам обеспечен.

Управляйте нагрузкой... если есть инструмент для этого

Слишком много потоков – и Windows становится похожей на умирающего лебедя. Это вызывает нарушения и нестабильную работу. Вы не можете создать поток просто «по просьбе пользователя» и надеяться, что Windows его обслужит в первую очередь. Так не будет. Для эффективного управления и нормальной продуктивности необходимо контролировать поступающую нагрузку, отслеживать текущее состояние компьютера и варьировать приоритеты.

Это наводит на мысль, что для грамотного и эффективного управления потоками необходим механизм (MVS и DOS/VS реализовали его так давно, что, наверное, некоторые из читателей еще не родились к тому моменту...), или «монитор транзакций», или другие части программного обеспечения, встраиваемые в ваше приложение и позволяющие организовывать очереди и регулировать рабочие потоки для многопользовательского или серверного режима.

Помните также, что если процесс не может быть реально завершен в режиме параллельной работы..., то ему необходим не отдельный поток, ему необходима очередь сообщений.

Помните, что память виртуальна

Механизм виртуальной памяти следит за тем, какая часть памяти должна находиться в RAM, а какая должна быть сброшена в файл подкачки. Потоки усложняют ситуацию, если они обращаются в одно и то же время к разным адресам виртуальной памяти. Это значительно увеличивает нагрузку на систему. Помните, что реально память не всегда «свободна», как это пишут в окошках О системе. Всегда отождествляйте доступ к памяти с доступом к файлу на диске и создавайте приложение с учетом вышесказанного.

Заявляйте о себе открыто

Всякий раз, когда любой из ваших потоков пытается воспользоваться общим ресурсом, вы обязаны каким-то образом легализовать и защитить вашу деятельность. Хорошим средством для этого являются критические секции, семафоры и очереди сообщений. Если вы протестировали приложение в «стерильном» режиме и не напоролись ни на одну из ошибок синхронизации, то знайте, в реальном режиме, когда пользователю вздумается запустить что угодно, вы попадете впросак, ибо ошибки будут обязательно!

Делегируйте обязанности все время

Итак, необходимо предусмотреть «одновременный» доступ множества потоков к общим ресурсам. Хорошей идеей в данном случае является делегирование функций доступа специализированным потокам, специально создаваемым для этих целей, умеющим создавать очереди, работать в дискретном режиме с определенным кругом устройств и иметь небольшой набор обязанностей, с которыми они отлично справляются. Это лучшее решение, чем поток, похожий на «царя горы», способный заблокировать множество других автономных задач.

Вызов процедуры в другом потоке

Решение

```
CreateThread(nil, 0, @MyProcedure, 0, 0, nil);
```

Использование собственных курсоров в приложении

Как использовать свои курсоры в приложении?

С помощью программы Image Editor упакуйте курсор в RES-файл. В следующем примере подразумевается, что курсор под именем cursor_1 сохранен в файле MYFILE.RES.

```
{$R c:\programs\delphi\MyFile.res}             { Это ваш RES-файл }
const
  PutTheCursorHere_Dude = 1;                   { произвольное положительное число }
procedure stuff;
begin
  Screen.Cursors[PutTheCursorHere_Dude] := LoadCursor(hInstance,
                                                      PChar('cursor_1'));
  Screen.Cursor := PutTheCursorHere_Dude;
end;
```

Использование MemoryStream

Думайте о потоке памяти как о расположенном в памяти файле. Команды для работы с потоком очень похожи на команды для работы с файлами. В действительности, это ближе к команде BlockWrite.

Вот «медленный» путь записи строки в поток:

```
for i := 1 to Length(s) do Memstream.Write(s[i], 1);
```

Данный код за один проход пишет один символ строки. Просто и легко для понимания, но немного медленно в работе. Для ускорения процесса можно сделать следующее:

```
Memstream.Write(s[1], Length(s));
```

Две строчки делают то же самое, они добавляют символы в поток. Если вы не перемещали внутренний курсор потока (Seek), символы просто добавляются в конец.

Теперь, чтобы соблюсти разбиение на строки, надо добавлять эти спецсимволы (0D0Ah) самостоятельно:

```
Memstream.Write(#13, 1);
Memstream.Write(#10, 1);
```

Можно прибегнуть и к более изощренному методу:

```
procedure StreamWriteStr(var ms: TMemoryStream; s: string);
begin
  ms.Write(s[1], Length(s));
```

```
end;
procedure StreamWriteLnStr(var ms: TMemoryStream; s: string);
begin
  StreamWriteStr(ms, s + #13#10);
end;
```

Также вы можете создать собственный класс-потомок TMemoryStream с методом записи строк.

Преобразование координат

Поверьте, достаточно просто преобразовать координаты X,Y, передаваемые в параметрах событий OnDragOver и OnDragDrop, в координаты формы.

Работайте со свойствами Left и Top компонента, над которым перемещается курсор. Приведу простой пример. Поместите на форму компонент Memo и присвойте свойству Align значение alTop. Поместите на форму панель, также присвойте свойству Align значение alTop и задайте небольшое значение свойству Height, скажем, 6 или 7 пикселов. Установите DragMode на dmAutomatica и DragCursor на crVSplit. Поместите другой компонент Memo и установите Align на alClient. Одновременно выберите оба Memo-компонента, панель и создайте общий обработчик события OnDragOver, как показано ниже:

```
procedure TForm1.Memo1DragOver(Sender, Source: TObject; X, Y: Integer;
                               State: TDragState; var Accept: Boolean);
begin
  if Source = Panel1 then
    with Sender as TControl do begin
      Accept := True;
      Memo1.Height := Y + Top;
    end else Accept := False;
end;
```

[News Group]

Вызов контекстного меню в координатах курсора мыши

Решение

```
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
                               Shift: TShiftState; X,Y: integer);
var
  FCursor: TPoint;
begin
  if Button = mbRight then begin
    GetCursorPos(FCursor);
    PopupMenu1.Popup(FCursor.X, FCursor.Y);
  end;
end;
```

[Тимошенко Александр]

Размер диалогового окна

Как правильно определить размер диалогового окна для вывода на экран с различным разрешением и различными шрифтами?

Начнем с «Microsoft Windows User Interface Guidelines» (Руководящие принципы построения интерфейса пользователя Microsoft Windows) и допустим, что мы создаем диалоговое окно, содержащее компонент TMemo, занимающий бо́льшую часть площади формы, и кнопки OK и Cancel, размещенные в ее нижней части.

Несколько примечаний из только что упомянутого документа:

- Диалоговые окна должны быть основаны на базовых диалоговых модулях, (Dialog Base Units, DBU), которые создаются с учетом размера шрифта и разрешения экрана.
- Диалоговые окна должны быть созданы, по возможности, на основе одного из нескольких стандартных размеров. Для нашего окна используем размер 212×188 DBU.
- Все элементы управления должны располагаться как минимум на расстоянии 7 DBU от края окна.
- Зазор между всеми элементами управления должен составлять минимум 4 DBU.
- Кнопки должны иметь высоту 14 DBU. (Про ширину кнопок вышеуказанный источник умалчивает. В обычном случая я использую кнопки шириной 40 DBU.)

Вот необходимая для создания формы и элементов управления информация, которую мы можем получить во время выполнения приложения:

```
procedure TMyForm.FormCreate(Sender: TObject);
var
  BaseUnit, Margin, Spacing, BtnW, BtnH: Integer;
begin
  BaseUnit := Canvas.TextHeight('0');       { 1 BaseUnit = 8 DBU определениям }
  Width := (212 * BaseUnit) div 8;
  Height := (188 * BaseUnit) div 8;
  Margin := (7 * BaseUnit) div 8 - GetSystemMetrics(SM_CXFIXEDFRAME);
  Spacing := (4 * BaseUnit) div 8;
  BtnW := (40 * BaseUnit) div 8;
  BtnH := (14 * BaseUnit) div 8;
  Memo1.SetBounds(Margin, Margin, ClientWidth - 2 * Margin,
                  ClientHeight - 2 * Margin - Spacing - BtnH);
  OkButton.SetBounds(ClientWidth - Margin - Spacing - 2 * BtnW,
                  ClientHeight - Margin - BtnH, BtnW, BtnH);
  CancelButton.SetBounds(ClientWidth - Margin - BtnW,
                  ClientHeight - Margin - BtnH, BtnW, BtnH);
end;
```

Данный код позволяет создать диалоговое окно с правильными размерами и пропорциями, независимо от разрешения экрана и шрифтов.

[News Group]

Заголовок диалогового окна

Заголовок диалогового окна устанавливается в момент вызова CreateMessageDialog, код которой приведен в Dialogs.pas. При этом выполняется вызов LoadStr, который получает WarningCaption, CautionCaption и пр., так что у вас есть два пути: или вы изменяете Dialogs.pas, или редактируете строки в .RES-файле.

Запуск приложения в полноэкранном режиме

Запуск приложения в полноэкранном режиме означает, что окно приложения полностью занимает рабочий стол. Это бывает необходимо для обеспечения поддержки функций акселератора видеокарты, которая может ускорить работу лишь полной области экрана, но не только для этого. Например, для того чтобы сделать видимой для пользователя только вашу программу. Кстати, полноэкранный запуск, в общих чертах, имеет отношение не только к OpenGL, DirectX и 3D. Строго говоря, полноэкранный режим требует лишь установки флага состояния окна wsMaximize, и все.

Но есть другой вопрос, подразумеваемый требованиями для полноэкранных приложений. Это наличие возможности выбора пользователем специфического разрешения экрана и глубины цвета или возможность запуска приложения в фиксированном разрешении. Последнее важно в каждом конкретном случае, поскольку не все видеокарты поддерживают все разрешения и часто игра или другое 3D-приложение хотят работать в другом разрешении (в основном, более низком), чем пользователю требуется в каждодневной работе.

Так что полностью вопрос должен читаться так: как запустить полноэкранное приложение в специфичном разрешении экрана и глубине цвета (без перезагрузки)? Ключевой является функция ChangeDisplaySettings. В зависимости от видеодрайвера, можно динамически установить один из режимов, не перегружая компьютер:

```
function ScreenModeChanged(ModeIndex: Integer): Boolean;
// изменение видеорежима, задаваемого 'ModeIndex'
var
  DeviceMode: TDevMode;
begin
  with DeviceMode do begin
    dmSize := SizeOf(DeviceMode);
    dmBitsPerPel := VideoModes[ModeIndex].ColorDepth;
    dmPelsWidth := VideoModes[ModeIndex].Width;
    dmPelsHeight := VideoModes[ModeIndex].Height;
    dmFields := DM_BITSPERPEL or DM_PELSWIDTH or DM_PELSHEIGHT;
```

```
// при неудачной смене режима переходим в режим текущего разрешения
    Result := ChangeDisplaySettings(DeviceMode, CDS_FULLSCREEN) =
              DISP_CHANGE_SUCCESSFUL;
    if ModeIndex = 0 then Result := False;
  end;
end;
```

Если вы обратили внимание, в этом примере присутствует глобальная переменная VideoModes. Ее наличие обусловлено необходимостью перечисления всех доступных режимов, которые могут быть установлены динамически и загружены в структуру, подобную VideoModes, для гарантии использования только описанных режимов.

```
const
  MaxVideoModes = 200;                  // это не очень актуально

type
  TVideoMode = record
    Width, Height, ColorDepth: Word;
    Description: String[20];
end;
var
  VideoModes: array[0..MaxVideoModes] of TVideoMode;
  NumberVideomodes: Integer = 1;       // 1, поскольку есть режим по умолчанию
```

Как видите, это делает наш пример более функциональным. При необходимости можно заменить в вышеуказанной функции VideoModes на фиксированные значения (скажем, на 640, 480, 16). Перечисление всех видеорежимов осуществляется при помощи EnumDisplaySettings:

```
procedure ReadVideoModes;
var
  I, ModeNumber: Integer;
  done: Boolean;
  DeviceMode: TDevMode;
  DeskDC: HDC;
begin
// создание режима "по умолчанию"
  with VideoModes[0] do
  try
    DeskDC := GetDC(0);
    ColorDepth := GetDeviceCaps(DeskDC, BITSPIXEL);
    Width := Screen.Width;
    Height := Screen.Height;
    Description := 'default';
  finally
    ReleaseDC(0,DeskDC);
  end;
// перечисляем все доступные видеорежимы
  ModeNumber := 0;
  done := False;
```

```
    repeat
      done := not EnumDisplaySettings(nil, ModeNumber, DeviceMode);
      TryToAddToList(DeviceMode);
      Inc(ModeNumber);
    until(done or (NumberVideomodes >= MaxVideoModes));
  // режимы низкого разрешения не всегда перечислимы, о них запрашивают явно
    with DeviceMode do begin
      dmBitsPerPel := 8;
      dmPelsWidth := 42;
      dmPelsHeight := 37;
      dmFields := DM_BITSPERPEL or DM_PELSWIDTH or DM_PELSHEIGHT;
  // тест видеодрайвера: убедимся, что он справится со всеми видеорежимами
      if ChangeDisplaySettings(DeviceMode, CDS_TEST or CDS_FULLSCREEN) <>
                               DISP_CHANGE_SUCCESSFUL then begin
        I := 0;
        while (I < NumberLowResModes - 1)
               and (NumberVideoModes < MaxVideoModes) do begin
          dmSize := Sizeof(DeviceMode);
          dmBitsPerPel := LowResModes[I].ColorDepth;
          dmPelsWidth := LowResModes[I].Width;
          dmPelsHeight := LowResModes[I].Height;
          dmFields := DM_BITSPERPEL or DM_PELSWIDTH or DM_PELSHEIGHT;
          TryToAddToList(DeviceMode);
          Inc(I);
        end;
      end;
    end;
  end;
```

Функция не трудна для понимания. Есть две части, которые нужно рассмотреть. Сначала – стандартное перечисление видеорежимов. В следующей части удостоверяемся, что все режимы низкого разрешения также протестированы:

```
  type
    TLowResMode = record
      Width, Height, ColorDepth: Word;
    end;

  const
    NumberLowResModes = 60;
    LowResModes: array[0..NumberLowResModes - 1] of TLowResMode =
    ((Width: 320; Height: 200; ColorDepth: 8),
     (Width: 320; Height: 200; ColorDepth: 15),
     (Width: 320; Height: 200; ColorDepth: 16),
     (Width: 320; Height: 200; ColorDepth:24),
     (Width: 320; Height: 200; ColorDepth: 32),
     (Width: 320; Height: 240; ColorDepth: 8),
     (Width: 320; Height: 240; ColorDepth: 15),
     (Width: 320; Height: 240; ColorDepth:16),
     (Width: 320; Height: 240; ColorDepth: 24),
```

```
(Width: 320; Height: 240; ColorDepth:32),
(Width: 320; Height: 350; ColorDepth: 8),
(Width: 320; Height: 350; ColorDepth:15),
(Width: 320; Height: 350; ColorDepth: 16),
(Width: 320; Height: 350; ColorDepth:24),
(Width: 320; Height: 350; ColorDepth: 32),
(Width: 320; Height: 400; ColorDepth: 8),
(Width: 320; Height: 400; ColorDepth: 15),
(Width: 320; Height: 400; ColorDepth:16),
(Width: 320; Height: 400; ColorDepth: 24),
(Width: 320; Height: 400; ColorDepth:32),
(Width: 320; Height: 480; ColorDepth: 8),
(Width: 320; Height: 480; ColorDepth:15),
(Width: 320; Height: 480; ColorDepth: 16),
(Width: 320; Height: 480; ColorDepth:24),
(Width: 320; Height: 480; ColorDepth: 32),
(Width: 360; Height: 200; ColorDepth: 8),
(Width: 360; Height: 200; ColorDepth: 15),
(Width: 360; Height: 200; ColorDepth:16),
(Width: 360; Height: 200; ColorDepth: 24),
(Width: 360; Height: 200; ColorDepth: 32),
(Width: 360; Height: 240; ColorDepth: 8),
(Width: 360; Height: 240; ColorDepth: 15),
(Width: 360; Height: 240; ColorDepth: 16),
(Width: 360; Height: 240; ColorDepth: 24),
(Width: 360; Height: 240; ColorDepth: 32),
(Width: 360; Height: 350; ColorDepth: 8),
(Width: 360; Height: 350; ColorDepth: 15),
(Width: 360; Height: 350; ColorDepth: 16),
(Width: 360; Height: 350; ColorDepth: 24),
(Width: 360; Height: 350; ColorDepth: 32),
(Width: 360; Height: 400; ColorDepth: 8),
(Width: 360; Height: 400; ColorDepth: 15),
(Width: 360; Height: 400; ColorDepth: 16),
(Width: 360; Height: 400; ColorDepth: 24),
(Width: 360; Height: 400; ColorDepth: 32),
(Width: 360; Height: 480; ColorDepth: 8),
(Width: 360; Height: 480; ColorDepth: 15),
(Width: 360; Height: 480; ColorDepth: 16),
(Width: 360; Height: 480; ColorDepth: 24),
(Width: 360; Height: 480; ColorDepth: 32),
(Width: 400; Height: 300; ColorDepth: 8),
(Width: 400; Height: 300; ColorDepth: 15),
(Width: 400; Height: 300; ColorDepth: 16),
(Width: 400; Height: 300; ColorDepth: 24),
(Width: 400; Height: 300; ColorDepth: 32),
(Width: 512; Height: 384; ColorDepth: 8),
(Width: 512; Height: 384; ColorDepth: 15),
(Width: 512; Height: 384; ColorDepth: 16),
(Width: 512; Height: 384; ColorDepth: 24),
(Width: 512; Height: 384; ColorDepth: 32));
```

Остается функция TryToAddToList:

```
procedure TryToAddToList(DeviceMode: TDevMode);
// Добавление видеорежима к списку, это не дубликат и режим действительно
// может быть установлен.

var
  I: Integer;

begin
// Смотрим на предмет дублирования видеорежима (такое может быть из-за показателя
// частоты смены кадров или т. к. мы явно пробуем все режимы низкого разрешения)
  for I := 1 to NumberVideomodes - 1 do
    with DeviceMode do
      if ((dmBitsPerPel = VideoModes[I].ColorDepth)
           and (dmPelsWidth = VideoModes[I].Width)
           and (dmPelsHeight = VideoModes[I].Height)) then Exit;
// повтор видеорежима (дубликат)
// устанавливаем тестируемый режим (на самом деле мы не устанавливаем данный режим,
// а хотим получить сообщение о его поддержке видеокартой).
  if ChangeDisplaySettings(DeviceMode, CDS_TEST
     or CDS_FULLSCREEN) <> DISP_CHANGE_SUCCESSFUL then Exit;
// если это новый, поддерживаемый режим, то добавляем его к списку
  with DeviceMode do begin
    VideoModes[NumberVideomodes].ColorDepth := dmBitsPerPel;
    VideoModes[NumberVideomodes].Width := dmPelsWidth;
    VideoModes[NumberVideomodes].Height := dmPelsHeight;
    VideoModes[NumberVideomodes].Description :=
      Format('%d x %d, %d bpp', [dmPelsWidth, dmPelsHeight, dmBitsPerPel]);
  end;
  Inc(NumberVideomodes);
end;
```

Для завершения реализации проекта необходима функция, восстанавливающая видеорежим по умолчанию при завершении работы приложения:

```
procedure RestoreDefaultMode;
// восстанавливаем видеорежим по умолчанию
var
  T: TDevMode absolute 0;          // маленькая хитрость: создаем указатель на ноль
begin
// Так как первый параметр является переменной, мы не можем использовать ноль
// непосредственно. Взамен мы используем переменную с абсолютным адресом нуля.
  ChangeDisplaySettings(T, CDS_FULLSCREEN);
end;
```

Добавление своих пунктов в системное меню окна

Добавьте в листинг константу:

```
const
```

```
  My_MenuItem = $4000;
```

Константу можете назвать по-своему и дать ей другой номер, но есть некоторые номера, которые зарезервированы Windows для собственных пунктов меню – не попадите на них.

Обязательно поместите в секцию private строку:

```
procedure HookSysCommand(var message: TwmSysCommand); message WM_SysCommand;

procedure TForm1.HookSysCommand(var message: TwmSysCommand);
begin
  inherited;
  case Message.CmdType of
    My_MenuItem: ShowMessage('Пункт активизирован');
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  SysMenu: THandle;
begin
  SysMenu := GetSystemMenu(Handle, False);
  AppendMenu(SysMenu, mf_SEPARATOR, 0, #0);
  AppendMenu(SysMenu, mf_BYPOSITION, My_MenuItem, 'Новый пункт');
{ AppendMenu добавляет новый пункт в конец, а для вставки своего пункта в другую
  порядковую позицию воспользуйтесь InsertMenu пример:
  InsertMenu(SMenu, 1, mf_BYPOSITION, My_MenuItem, 'Новый пункт'); }
end;
```

[Алексей]

Получение различных диалогов из шаблона формы

Решение

```
  ...
Application.CreateForm(TFormX, FormX);
FormX.ShowModal; FormX.Free;
  ...
```

Не забывайте, что свойство Visible такой формы при ее создании должно быть установлено в False, в противном случае при попытке отобразить ее в модальном режиме вы получите ошибку.

Задержка без использования времени CPU

Я хочу создать серверное приложение (текстовый режим) в Delphi 3, ожидающее некоторые данные из последовательного порта. Во время ожидания мне необходимо «усыплять» приложение, дабы дать возможность другим программам полноценно использовать время CPU (простое зацикливание `repeat ... until` *не подходит).*

- `repeat ... until` не заблокирует другие процессы системы из-за вытесняющей многозадачности Win32, но вы действительно загрузите CPU на 100% и заблокируете текущий процесс, а это означает, например, что в текущем приложении не будет происходить перерисовка окна.
- Обрабатывая приходящие сообщения в теле вашего цикла, вы решите проблемы перерисовки приложения, но загрузка CPU так и останется на уровне 100%. Сделать это можно примерно так:

```
repeat
  while PeekMessage(Msg, 0, 0, 0, pm_Remove) do begin
    TranslateMessage(Msg);
    DispatchMessage(Msg);
  end;
until ThereIsSomethingGoingOnOnTheSerialLine;
```

- Можно задать паузу (например, `Sleep(100)`) перед каждым опросом последовательного порта. Это значительно уменьшает нагрузку на CPU.
- Если какой-то компонент или объект посылает приложению системное сообщение при наличии необходимой информации, то наилучший способ решения проблемы состоит в создании обработчика события, получающего управление вместе с данным сообщением и выполняющего необходимые действия, но поскольку все это происходит в консольном режиме, вам понадобится написать оконную функцию и указать ее консольному окну, например, так:

```
function MyWndProc(Wnd: HWnd; Msg, wParam, lParam: Integer): Integer;
begin
  case Msg of
    wm_SerialLineReceivesData: begin
                               ...
                               end;
            else
              Result := CallWindowProc(OldWndProc, Wnd,
                        Msg, wParam, lParam);
  end;
end;
```

В вашей главной программе:

```
  ...
var
  OldWndProc: Pointer;
begin
  OldWndProc := Pointer(SetWindowLong(GetActiveWindow, gwl_WndProc,
                      Integer(@MyWndProc)));
  ...
  SetWindowLong(GetActiveWindow, gwl_WndProc, Integer(OldWndProc));
end.
```

Моментальный снимок экрана

Как получить снимок экрана (типа программ перехвата экрана)?

Обратитесь к стандартным функциям Windows API:

- `hWnd := GetDesktopWindow` для получения дескриптора рабочего стола;
- `hDC := GetDC(hWnd)` для получения `HDC` (дескриптора контекста экрана);
- и не забывайте освобождать (уничтожать дескриптор) `HDC` после выполнения задачи.

Используя `TCanvas.Handle` в качестве `HDC`, можно при помощи WinAPI реализовать функции рисования или, если это возможно, присвоить `HDC` свойству `Handle` непосредственно при создании `TCanvas`.

Количество цветов в системе

Мне необходимо узнать, как сконфигурирована система: для работы с 16 или 256 цветами.

Приведенное ниже выражение возвращает количество цветов, используемое в текущий момент в системе. `Canvas` должен быть холстом экранного элемента управления, например, формы.

```
NumberOfColors := (1 shl (GetDeviceCaps(Form1.Canvas.Handle, BITSPIXEL) *
                          GetDeviceCaps(Form1.Canvas.Handle, PLANES));
```

Быстрый способ вывода графики

Как быстро выводить графику?

Предлагаем пример заполнения формы точками случайного цвета:

```
type
  TRGB = record
    b, g, r: byte;
  end;
  PRGB = ^TRGB;
var
  b: TBitMap;

procedure TForm1.FormCreate(sender: TObject);
begin
  b := TBitMap.Create;
  b.PixelFormat := pf24bit;
  b.Width := ClientWidth;
  b.Height := ClientHeight;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
var
```

```
    p: PRGB;
    x, y: integer;
  begin
    Randomize;
    for y := 0 to b.Height - 1 do begin
      p := b.Scanline[y];
      for x := 0 to b.Width - 1 do begin
        p.r := Random(256);
        p.g := Random(256);
        p.b := Random(256);
        inc(p);
      end;
    end;
    Canvas.Draw(0, 0, b);
  end;
  procedure TForm1.FormDestroy(Sender: TObject);
  begin
    b.Free;
  end;
```

[News Group]

Как бороться с «квадратичностью» Image

При вставке какой-либо картинки в Image с произвольными очертаниями, можно легко обратиться к самой картинке, а не к ненужному «куску» Image.

Решение

Вставляем картинку, например с белым фоном, Transparent := True и в обработчике пишем:

```
if Image1.Picture.Bitmep.Canvas.Pixels[X, Y] <> clwhite then
   Image1.Cursor := crHourGlass
else
  Image1.Cursor := crDefault;
```

Программа проверяет цвет пиксела под указателем, и, если он отличается от белого, указатель меняет свой вид. Кстати, свойство Image1.Picture.Bitmap.Canvas.Pixels[X, Y] доступно не только для чтения, но и для изменения. Его также можно использовать при рисовании.

[Аркадий]

Копирование содержимого экрана на форму

Решение 1

```
var
  Image2: TImage;
```

```
procedure TForm1.CopyScreen;
var
  DeskTopDC: HDC;
  DeskTopCanvas: TCanvas;
  DeskTopRect: TRect;
begin
  Image2 := TImage.Create(Form1);
  with Image2 do begin
    Height := Screen.Height;
    Width := Screen.Width;
  end;
  Image2.Canvas.CopyMode := cmSrcCopy;
  DeskTopDC := GetWindowDC(GetDeskTopWindow);
  DeskTopCanvas := TCanvas.Create;
  DeskTopCanvas.Handle := DeskTopDC;
  Image2.Canvas.CopyRect(Image2.Canvas.ClipRect, DeskTopCanvas,
                         DeskTopCanvas.ClipRect);
  Image1.Picture.Assign(Image2.Picture);
{ Image1 расположен на целевой форме и выровнен по области клиента }
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Image2.Free;
end;
```

Решение 2

В Delphi, если вы хотите получить изображение клиентской области формы, надо вызвать `GetFormImage`. Но иногда возникает необходимость получения снимка формы целиком, вместе с заголовком, контуром и всем содержимым. Или целиком всего экрана. В условиях дефицита времени мы бы посоветовали отобразить диалоговое окно с надписью «Теперь нажмите кнопку Print Screen!», после чего работать с изображением, помещенным в буфер обмена.

Но мы никуда не спешим. Комбинирование холстов Delphi с несколькими функциями GDI сводят задачу получения снимка экрана всего к одной строчке кода.

`CaptureScreenRect` демонстрирует это. Код получает экранный контекст устройства с помощью `GetDC(0)`, а затем копирует прямоугольную область этого `DC` на холст изображения (`Bitmap`). Для копирования используется `BitBlt`. Смысл применения `BitBlt` (и любой функции `GDI`) в том, что Delphi помнит, что дескриптор холста есть `DC`, необходимый Windows.

Остальные функции копирования экрана захватывают прямоугольник и отдают реальную работу на откуп `CaptureScreenRect`. `CaptureScreen` захватывает для прямоугольника целый экран. А `CaptureClientImage` и `CaptureControlImage` захватывают прямоугольник области клиента и элемента управления, соответственно.

Эти четыре функции могут быть использованы для захвата любой произвольной области экрана, а также экранных областей форм, кнопок, полей редактирования, ComboBox и т. д. Не забывайте после работы освобождать используемые ресурсы (Bitmap).

```
unit ScrnCap;

interface

uses
  Windows, Forms, Classes, Graphics, Controls;
function CaptureScreenRect(ARect: TRect): TBitmap;
function CaptureScreen: TBitmap;
function CaptureClientImage(Control: TControl): TBitmap;
function CaptureControlImage(Control: TControl): TBitmap;
implementation

{ используем следующий код для захвата прямоугольной области экрана }

function CaptureScreenRect(ARect: TRect): TBitmap;
var
  ScreenDC: HDC;
begin
  Result := TBitmap.Create;
  with Result, ARect do begin
    Width := Right - Left;
    Height := Bottom - Top;
    ScreenDC := GetDC(0);
    try
      BitBlt(Canvas.Handle, 0, 0, Width, Height, ScreenDC, Left, Top, SRCCOPY);
    finally
      ReleaseDC(0, ScreenDC);
    end;
  end;
end;

{ используем следующий код для захвата целого экрана }

function CaptureScreen: TBitmap;
begin
  with Screen do
    Result := CaptureScreenRect(Rect(0, 0, Width, Height));
end;

{ следующий код – для захвата клиентской области формы или элемента управления }

function CaptureClientImage(Control: TControl): TBitmap;
begin
  with Control, Control.ClientOrigin do
    Result := CaptureScreenRect(Bounds(X, Y, ClientWidth, ClientHeight));
end;
```

```
{ используем следующий код для захвата целой формы или элемента управления }

function CaptureControlImage(Control: TControl): TBitmap;
begin
  with Control do
    if Parent = nil then
      Result := CaptureScreenRect(Bounds(Left, Top, Width, Height))
    else
      with Parent.ClientToScreen(Point(Left, Top)) do
        Result := CaptureScreenRect(Bounds(X, Y, Width, Height));
end;
end.
```

[News Group]

Обзор сети (типа Network Neighborhood)

Пример может служить отправной точкой для создания рабочего варианта.

```
unit netres_main_unit;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, StdCtrls, Buttons, Menus, ExtCtrls;

type
  TfrmMain = class(TForm)
    tvResources: TTreeView;
    btnOK: TBitBtn;
    btnClose: TBitBtn;
    Label1: TLabel;
    barBottom: TStatusBar;
    popResources: TPopupMenu;
    mniExpandAll: TMenuItem;
    mniCollapseAll: TMenuItem;
    mniSaveToFile: TMenuItem;
    mniLoadFromFile: TMenuItem;
    grpListType: TRadioGroup;
    grpResourceType: TRadioGroup;
    dlgOpen: TOpenDialog;
    dlgSave: TSaveDialog;
    procedure FormCreate(Sender: TObject);
    procedure btnCloseClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure mniExpandAllClick(Sender: TObject);
    procedure mniCollapseAllClick(Sender: TObject);
    procedure mniSaveToFileClick(Sender: TObject);
    procedure mniLoadFromFileClick(Sender: TObject);
```

```
    procedure btnOKClick(Sender: TObject);
  private
    ListType, ResourceType: DWORD;
    procedure ShowHint(Sender: TObject);
    procedure DoEnumeration;
    procedure DoEnumerationContainer(NetResContainer: TNetResource);
    procedure AddContainer(NetRes: TNetResource);
    procedure AddShare(TopContainerIndex: Integer; NetRes: TNetResource);
    procedure AddShareString(TopContainerIndex: Integer; ItemName: String);
    procedure AddConnection(NetRes: TNetResource);
end;

var
  frmMain: TfrmMain;

implementation
{$R *.DFM}

procedure TfrmMain.ShowHint(Sender: TObject);
begin
  barBottom.Panels.Items[0].Text := Application.Hint;
end;

procedure TfrmMain.FormCreate(Sender: TObject);
begin
  Application.OnHint := ShowHint;
  barBottom.Panels.Items[0].Text := '';
end;

procedure TfrmMain.btnCloseClick(Sender: TObject);
begin
  Close;
end;

{ Перечисляем все сетевые ресурсы }
procedure TfrmMain.DoEnumeration;
var
  NetRes: Array[0..2] of TNetResource;
  Loop: Integer;
  r, hEnum, EntryCount, NetResLen: DWORD;
begin
  case grpListType.ItemIndex of
       { Подключенные ресурсы: }
      1: ListType := RESOURCE_CONNECTED;

       { Возобновляемые ресурсы: }
      2: ListType := RESOURCE_REMEMBERED;
       { Глобальные: }
    else ListType := RESOURCE_GLOBALNET;
  end;
  case grpResourceType.ItemIndex of
```

```
        { Дисковые ресурсы: }
       1: ResourceType := RESOURCETYPE_DISK;
        { Принтерные ресурсы: }
       2: ResourceType := RESOURCETYPE_PRINT;
        { Все: }
     else ResourceType := RESOURCETYPE_ANY;
   end;
   Screen.Cursor := crHourGlass;
   try
 { Удаляем любые старые элементы из дерева }
     for Loop := tvResources.Items.Count - 1 downto 0 do
       tvResources.Items[Loop].Delete;
   except
   end;
 { Начинаем перечисление: }
   r := WNetOpenEnum(ListType, ResourceType, 0, nil, hEnum);
   if r <> NO_ERROR then begin
     if r = ERROR_EXTENDED_ERROR then
       MessageDlg('Невозможно выполнить обзор сети.' + #13
                + 'Произошла сетевая ошибка.', mtError, [mbOK], 0)
     else
       MessageDlg('Невозможно сделать обзор сети.', mtError, [mbOK], 0);
     Exit;
   end;
   try
 { Мы получили правильный дескриптор перечисления; опрашиваем ресурсы }
     while (1 = 1) do begin
       EntryCount := 1;
       NetResLen := SizeOf(NetRes);
       r := WNetEnumResource(hEnum, EntryCount, @NetRes, NetResLen);
       case r of
           0: begin             { Это контейнер, организуем итерацию: }
                if NetRes[0].dwUsage = RESOURCEUSAGE_CONTAINER then
                  DoEnumerationContainer(NetRes[0])
                else        { Здесь получаем подключенные и возобновляемые ресурсы }
                  if ListType in [RESOURCE_REMEMBERED, RESOURCE_CONNECTED] then
                    AddConnection(NetRes[0]);
              end;
     ERROR_NO_MORE_ITEMS: { Получены все ресурсы: }
                Break;
          else begin       { Другие ошибки: }
                MessageDlg('Ошибка опроса ресурсов.', mtError, [mbOK], 0);
                break;
              end;
       end;
     end;
   finally
     Screen.Cursor := crDefault;
 { Закрываем дескриптор перечисления: }
     WNetCloseEnum(hEnum);
   end;
```

```
  end;

{ Перечисление заданного контейнера. Эта функция обычно вызывается рекурсивно }
procedure TfrmMain.DoEnumerationContainer(NetResContainer: TNetResource);
var
  NetRes: Array[0..10] of TNetResource;
  TopContainerIndex: Integer;
  r, hEnum, EntryCount, NetResLen: DWORD;
begin
{ Добавляем имя контейнера к найденным сетевым ресурсам }
  AddContainer(NetResContainer);
{ Делаем этот элемент текущим корневым уровнем }
  TopContainerIndex := tvResources.Items.Count-1;
{ Начинаем перечисление: }
  if ListType = RESOURCE_GLOBALNET then
{ Перечисляем глобальные объекты сети: }
    r := WNetOpenEnum(ListType, ResourceType, RESOURCEUSAGE_CONTAINER,
                      @NetResContainer, hEnum)
  else
{ Перечисляем подключаемые и возобновляемые ресурсы
  (другие получить здесь невозможно) }
  r := WNetOpenEnum(ListType, ResourceType, RESOURCEUSAGE_CONTAINER, nil, hEnum);
{ Невозможно перечислить ресурсы данного контейнера, выводим соответствующее
  предупреждение и едем дальше }
  if r <> NO_ERROR then begin
    AddShareString(TopContainerIndex, '<Не могу опросить ресурсы (Ошибка #'
                 + IntToStr(r) + '>');
    WNetCloseEnum(hEnum);
    Exit;
  end;
{ Мы получили правильный дескриптор перечисления; опрашиваем ресурсы }
  while (1 = 1) do begin
    EntryCount := 1;
    NetResLen := SizeOf(NetRes);
    r := WNetEnumResource(hEnum, EntryCount, @NetRes, NetResLen);
    case r of
           0: begin {Другой контейнер для перечисления, необходим рекурсивный вызов}
                if (NetRes[0].dwUsage = RESOURCEUSAGE_CONTAINER)
                   or (NetRes[0].dwUsage=10) then
                  DoEnumerationContainer(NetRes[0])
                else
                  case NetRes[0].dwDisplayType of
                  { Верхний уровень }
                  RESOURCEDISPLAYTYPE_GENERIC,
                   RESOURCEDISPLAYTYPE_DOMAIN,
                   RESOURCEDISPLAYTYPE_SERVER:        AddContainer(NetRes[0]);
                  { Ресурсы общего доступа: }
                    RESOURCEDISPLAYTYPE_SHARE:
                                       AddShare(TopContainerIndex, NetRes[0]);
                  end;
              end;
```

```
      ERROR_NO_MORE_ITEMS: Break;
        else begin
               MessageDlg('Ошибка #' + IntToStr(r) + ' при перечислении ресурсов.',
                          mtError, [mbOK], 0);
               Break;
             end;
    end;
  end;
{ Закрываем дескриптор перечисления }
  WNetCloseEnum(hEnum);
end;

procedure TfrmMain.FormShow(Sender: TObject);
begin
  DoEnumeration;
end;
{ Добавляем элементы дерева; помечаем, что это контейнер }
procedure TfrmMain.AddContainer(NetRes: TNetResource);
var
  ItemName: String;
begin
  ItemName := Trim(String(NetRes.lpRemoteName));
  if Trim(String(NetRes.lpComment))<>'' then begin
    if ItemName <> '' then ItemName := ItemName + ' ';
    ItemName := ItemName + '(' + String(NetRes.lpComment) + ')';
  end;
  tvResources.Items.Add(tvResources.Selected,ItemName);
end;

{ Добавляем дочерние элементы к контейнеру, обозначенному как текущий
  верхний уровень }
procedure TfrmMain.AddShare(TopContainerIndex: Integer; NetRes: TNetResource);
var
  ItemName: String;
begin
  ItemName := Trim(String(NetRes.lpRemoteName));
  if Trim(String(NetRes.lpComment)) <> '' then begin
    if ItemName <> '' then ItemName := ItemName + ' ';
    ItemName := ItemName + '(' + String(NetRes.lpComment) + ')';
  end;
  tvResources.Items.AddChild(tvResources.Items[TopContainerIndex], ItemName);
end;

{ Добавляем дочерние элементы к контейнеру, обозначенному как текущий верхний
  уровень; это просто добавляет строку для таких задач, как, например, перечисление
  контейнера. То есть, некоторые контейнерные ресурсы общего доступа нам не доступны}
procedure TfrmMain.AddShareString(TopContainerIndex: Integer; ItemName: String);
begin
  tvResources.Items.AddChild(tvResources.Items[TopContainerIndex], ItemName);
end;
```

```
{ Добавляем соединения к дереву. По большому счету, к этому моменту все сетевые
  ресурсы типа возобновляемых и текущих соединений уже отображены. }
procedure TfrmMain.AddConnection(NetRes: TNetResource);
var
  ItemName: String;
begin
  ItemName := Trim(String(NetRes.lpLocalName));
  if Trim(String(NetRes.lpRemoteName)) <> '' then begin
    if ItemName <> '' then ItemName := ItemName + ' ';
    ItemName := ItemName + '-> ' + Trim(String(NetRes.lpRemoteName));
  end;
  tvResources.Items.Add(tvResources.Selected, ItemName);
end;

{ Раскрываем все контейнеры дерева }
procedure TfrmMain.mniExpandAllClick(Sender: TObject);
begin
  tvResources.FullExpand;
end;

{ Сворачиваем все контейнеры дерева }
procedure TfrmMain.mniCollapseAllClick(Sender: TObject);
begin
  tvResources.FullCollapse;
end;

{ Записываем дерево в выбранный файл }
procedure TfrmMain.mniSaveToFileClick(Sender: TObject);
begin
  if dlgSave.Execute then tvResources.SaveToFile(dlgSave.FileName);
end;

{ Загружаем дерево из выбранного файла }
procedure TfrmMain.mniLoadFromFileClick(Sender: TObject);
begin
  if dlgOpen.Execute then tvResources.LoadFromFile(dlgOpen.FileName);
end;

{ Обновляем }
procedure TfrmMain.btnOKClick(Sender: TObject);
begin
  DoEnumeration;
end;

end.
```

Определение собственного IP-адреса

Как определить собственный IP-адрес из Delphi?

```
function my_ip_address: longint;
```

```
const
  bufsize = 255;

var
  buf: pointer;
  RemoteHost: PHostEnt;                          // Не освобождайте это!

begin
  buf := NIL;
  try
    GetMem(buf, bufsize);
    WinSock.GetHostName(buf, bufsize);           // это может работать и без сети
    RemoteHost := WinSock.GetHostByName(buf);
    if RemoteHost = NIL then
      my_ip_address := WinSock.htonl($07000001)  // 127.0.0.1
    else
      my_ip_address := longint(pointer(RemoteHost^.h_addr_list^)^);
  finally
    if buf <> NIL then FreeMem(buf, bufsize);
  end;
  Result := Winsock.ntohl(Result);
end;
```

Вначале возвращается локальный сетевой адрес компьютера, а затем, если он не равен 127.0.0.1, стандартный IP-адрес.

Единственное, что вам необходимо, это наличие winsock.dcu/winsock.pas, т. к. они не включаются в поставку Delphi ранних версий. В Delphi 5 просто пишем uses WinSock.

Остановка и запуск сервисов

Решение

unit1.dfm

```
object Form1: TForm1
  Left = 192
  Top = 107
  Width = 264
  Height = 121
  Caption = 'Сервис'
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  OldCreateOrder = False
  PixelsPerInch = 96
  TextHeight = 13
  object Label1: TLabel
    Left = 2
```

```
    Top = 8
    Width = 67
    Height = 13
    Caption = 'Имя сервиса'
  end
  object Button1: TButton
    Left = 4
    Top = 56
    Width = 95
    Height = 25
    Caption = 'Стоп сервис'
    TabOrder = 0
    OnClick = Button1Click
  end
  object Button2: TButton
    Left = 148
    Top = 56
    Width = 95
    Height = 25
    Caption = 'Старт сервис'
    TabOrder = 1
    OnClick = Button2Click
  end

  object Edit1: TEdit
    Left = 0
    Top = 24
    Width = 241
    Height = 21
    TabOrder = 2
    Text = 'Messenger'
  end
end
```

unit1.pas

```
unit Unit1;
interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Winsvc;

type
  TForm1 = class(TForm)
      Button1: TButton;
      Button2: TButton;
      Edit1: TEdit;
      Label1: TLabel;
      procedure Button1Click(Sender: TObject);
      procedure StopService(ServiceName: String);
      procedure Button2Click(Sender: TObject);
```

```
      procedure StartService(ServiceName: String);
  end;

var
  Form1: TForm1;

implementation
{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
begin
  StopService(Edit1.Text);
end;

procedure TForm1.StopService(ServiceName: String);
var
  schService, schSCManager: DWORD;
  p: PChar;
  ss: _SERVICE_STATUS;
begin
  p := nil;
  schSCManager := OpenSCManager(nil, nil, SC_MANAGER_ALL_ACCESS);
  if schSCManager = 0 then RaiseLastWin32Error;
  try
    schService := OpenService(schSCManager, PChar(ServiceName),
                              SERVICE_ALL_ACCESS);
    if schService = 0 then RaiseLastWin32Error;
    try
      if not ControlService(schService, SERVICE_CONTROL_STOP, SS) then
        RaiseLastWin32Error;
    finally
      CloseServiceHandle(schService);
    end;
  finally
    CloseServiceHandle(schSCManager);
  end;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  StartService(Edit1.Text);
end;

procedure TForm1.StartService(ServiceName: String);
var
  schService, schSCManager: Dword;
  p: PChar;
begin
  p := nil;
  schSCManager := OpenSCManager(nil, nil, SC_MANAGER_ALL_ACCESS);
  if schSCManager = 0 then RaiseLastWin32Error;
```

```
  try
    schService := OpenService(schSCManager, PChar(ServiceName),
                              SERVICE_ALL_ACCESS);
    if schService = 0 then RaiseLastWin32Error;
    try
      if not Winsvc.startService(schService, 0, p) then RaiseLastWin32Error;
    finally
      CloseServiceHandle(schService);
    end;
  finally
    CloseServiceHandle(schSCManager);
  end;
end;

end.
```

[News Group]

Определение доступных серверов приложений

Как определить доступные серверы приложений на этой машине через реестр?

Просмотрите ключ HKEY_CLASSES_ROOT\CLSID\... и найдите вхождения подключа Borland DataBroker. Эти вхождения и являются серверами приложений.

Ниже пример, который загружает имена доступных серверов приложений в ListBox:

```
uses
  Registry;

procedure TForm1.FormCreate(Sender: TObject);
var
  I: integer;
  TempList: TStringList;
begin
  TempList := TStringList.Create;
  try
    with TRegistry.Create do
      try
        RootKey := HKEY_CLASSES_ROOT;
        if OpenKey('CLSID', False) then GetKeyNames(TempList);
        CloseKey;
        for I := 1 to TempList.Count - 1 do
          if KeyExists('CLSID\' + TempList[I] + '\Borland DataBroker') then begin
            if OpenKey('CLSID\' + TempList[I] + '\ProgID', False) then begin
              Listbox1.Items.Add(ReadString(''));
              CloseKey;
            end;
          end;
      finally
```

```
        Free;
      end;
  finally
    TempList.Free;
  end;
end;
```

Как определить доступность сетевых ресурсов?

Решение

```
type
  PNetResourceArray = ^TNetResourceArray;
  TNetResourceArray = array[0..MaxInt div SizeOf(TNetResource) - 1]
                        of TNetResource;

procedure EnumResources(LpNR: PNetResource);
var
  NetHandle: THandle;
  I, BufSize, NetResult: Integer;
  NetResources: PNetResourceArray;
  NewItem: TListItem;
  Count, Size: DWORD;
begin
  if WNetOpenEnum(RESOURCE_GLOBALNET, RESOURCETYPE_ANY, 0, LpNR,
                  NetHandle) <> NO_ERROR then Exit;
  try
    BufSize := 50 * SizeOf(TNetResource);
    GetMem(NetResources, BufSize);
    try
      while True do begin
        Count := $FFFFFFFF;
        Size := BufSize;
        NetResult := WNetEnumResource(NetHandle, Count, NetResources, Size);
        if NetResult = ERROR_MORE_DATA then begin
          BufSize := Size;
          ReallocMem(NetResources, BufSize);
          Continue;
        end;
        if NetResult <> NO_ERROR then Exit;
        for I := 0 to Count-1 do begin
          with NetResources^[I] do begin
            if RESOURCEUSAGE_CONTAINER = (DwUsage and RESOURCEUSAGE_CONTAINER) then
              EnumResources(@NetResources^[I]);
            if dwDisplayType = RESOURCEDISPLAYTYPE_SHARE then begin
// ^^^^^^^^^^^^^^^^^^^^^^^^^ - ресурс
// RESOURCEDISPLAYTYPE_SERVER - компьютер
// RESOURCEDISPLAYTYPE_DOMAIN - рабочая группа
// RESOURCEDISPLAYTYPE_GENERIC - сеть
              NewItem := Form1.ListView1.Items.Add;
              NewItem.Caption := LpRemoteName;
```

```
            end;
          end;
        end;
      end;
    finally
      FreeMem(NetResources, BufSize);
    end;
  finally
    WNetCloseEnum(NetHandle);
  end;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  OldCursor: TCursor;
begin
  OldCursor := Screen.Cursor;
  Screen.Cursor := crHourGlass;
  with ListView1.Items do begin
    BeginUpdate;
    Clear;
    EnumResources(nil);
    EndUpdate;
  end;
  Screen.Cursor := OldCursor;
end;
```

[Nomadic]

Примечание

Добавляем в uses *модуль* ComCtrls.

Получение сетевого имени пользователя

Можете попробовать этот код. Работа программы зависит от наличия NWCALLS.DLL на машине пользователя, но если он работал с сетью хоть раз, данная библиотека должна присутствовать на его машине.

```
unit GetLogin;
{ Данный модуль инкапсулирует несколько внешних функций библиотеки NWCALLS.DLL }
{ Модуль содержит функции, возвращающие Netware User ID и полное имя пользователя. }
interface

uses
  SysUtils, Messages, Dialogs;

function GetUserLogin: string;
function GetUserFullName(SomeUser: string): string;

implementation
```

```
type
  NWTimeStamp = record
    Year: byte;
    Month: byte;
    Day: byte;
    Hour: byte;
    Minute: byte;
    Second: byte;
    DayOfWeek: byte;
  end;

{ Netware API - требуется NWCALLS.DLL }
function NWGetDefaultConnectionID(var Connection: word): word;
                far; external 'NWCALLS';
function NWGetConnectionNumber(Connection: word; var ConnectionNumber: word): word;
                far; external 'NWCALLS';
function NWGetConnectionInformation(Connection: word; ConnectionNumber: word;
                ObjectName: PChar; var ObjectType: word; var ObjectID: word;
                var LoginTime: NWTimeStamp): word; far; external 'NWCALLS';
function NWReadPropertyValue(Connection: word; ObjectName: PChar;
                ObjectType: word; PropertyName: PChar; DataSetIndex: byte;
                DataBuffer: PChar; var More: byte; var Flags: byte): word;
               far; external 'NWCALLS';       { конец секции работы с Netware API }

function GetUserLogin: string;
var
  ConnectionID: word;
  ConnectionNumber: word;
  RC: word;
  Name: array[0..50] of Char;
  ObjectType: word;
  ObjectID: word;
  LoginTime: NWTimeStamp;
begin
  RC := NWGetDefaultConnectionID(ConnectionID);
  RC := NWGetConnectionNumber(ConnectionID, ConnectionNumber);
  RC := NWGetConnectionInformation(ConnectionID, ConnectionNumber, Name,
                                   ObjectType, ObjectID, LoginTime);
  Result := StrPas(Name);
end;

function GetUserFullName( SomeUser: string): string;
{ Реально имя пользователя является свойством 'IDENTIFICATON'. Вы должны вызывать
  NWReadPropertyValue с параметрами (между прочим) вашего ConnectionID, имени
  объекта (такое же, как и логин пользователя, сетевое имя которого мы пытаемся
  узнать) и свойство name, которое нам необходимо получить, в нашем случае
  'IDENTIFICATION' (это и есть искомая величина - полное имя пользователя). }
var
  ConnectionID: word;
  RC: word;
  Name: array[0..50] of Char;
```

```
  ObjectType: word;
  PropName: array[0..14] of Char;
  DataSetIndex: byte;
  FullName: array[0..127] of Char;
  More: byte;
  Flags: byte;
begin
  RC := NWGetDefaultConnectionID(ConnectionID);
  ObjectType := 256; {пользователь}
  StrPCopy(PropName, 'IDENTIFICATION');
  DataSetIndex := 1;
  StrPCopy(Name, SomeUser);
  RC := NWReadPropertyValue(ConnectionID, Name, ObjectType, PropName, DataSetIndex,
                            FullName, More, Flags);
  if RC = 35324 then MessageDlg('Пользователь ' + SomeUser
                    + ' на этом сервере не обнаружен!', mtError, [mbOK], 0);
  Result := StrPas(FullName);
end;

end.
```

Список пользователей в Windows NT/2000

Решение

```
unit Func;

interface

uses
  SysUtils, Classes, StdCtrls, ComCtrls, Graphics, Windows;

{$EXTERNALSYM NetUserEnum}
function NetUserEnum(servername: LPWSTR; level, filter: DWORD; bufptr: Pointer;
      prefmaxlen: DWORD; entriesread, totalentries, resume_handle: LPDWORD): DWORD;
      stdcall; external 'NetApi32.dll' Name 'NetUserEnum';
function NetApiBufferFree(Buffer: Pointer{LPVOID}): DWORD; stdcall;
      external 'NetApi32.dll' Name 'NetApiBufferFree';
procedure GetLocalUserList(ulist: TStringList);

implementation

// возвращает список пользователей локального хоста
procedure GetLocalUserList(ulist: TStringList);
const
  NERR_SUCCESS = 0;
  FILTER_TEMP_DUPLICATE_ACCOUNT = $0001;
  FILTER_NORMAL_ACCOUNT = $0002;
  FILTER_PROXY_ACCOUNT = $0004;
  FILTER_INTERDOMAIN_TRUST_ACCOUNT = $0008;
  FILTER_WORKSTATION_TRUST_ACCOUNT = $0010;
```

```
  FILTER_SERVER_TRUST_ACCOUNT = $0020;

type
  TUSER_INFO_10 = record
      usri10_name,
      usri10_comment,
      usri10_usr_comment,
      usri10_full_name: PWideChar;
  end;

  PUSER_INFO_10 = ^TUSER_INFO_10;

var
  dwERead, dwETotal, dwRes, res: DWORD;
  inf: PUSER_INFO_10;
  info: Pointer;
  p: PChar;
  i: Integer;
begin
  if ulist = nil then Exit;
  ulist.Clear;
  info := nil;
  dwRes := 0;
  res := NetUserEnum(nil, 10, FILTER_NORMAL_ACCOUNT, @info, 65536, @dwERead,
                     @dwETotal, @dwRes);
  if (res <> NERR_SUCCESS) or (info = nil) then Exit;
  p := PChar(info);
  for i:=0 to dwERead - 1 do begin
    inf := PUSER_INFO_10(p + i * SizeOf(TUSER_INFO_10));
    ulist.Add(WideCharToString(PWideChar((inf^).usri10_name)));
  end;
  NetApiBufferFree(info);
end;

end.
```

[Кондратюк Виталий]

Определение имени компьютера средствами WinAPI

Решение

```
procedure TForm1.Button1Click(Sender: TObject);
var
  compnm: array[0..50] of Char;
  i: DWORD;
  comp: boolean;
begin
  comp := false;
  for i := 0 to 50 do begin
    comp := GetComputerName(compnm, i);
    if comp = true then begin
```

```
      Edit1.Text := StrPas(compnm);
      Exit;
    end;
  end;
end;
```

[Jungle Vitaliy]

Примечание

А можно и проще:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  compnm: array[0..MAX_COMPUTERNAME_LENGTH] of Char;
  i: DWORD;
  comp: boolean;
begin
  comp := false;
  i := SizeOf(compnm);
  comp := GetComputerName(compnm, i);
  if comp = true then Edit1.Text := StrPas(compnm);
end;
```

Подключение сетевого диска в Delphi

Решение 1

Данный код показывает, как создавать кнопку Сеть, вызывающую диалог подключения сетевого диска и указания логического диска для подключаемого сетевого ресурса. Этот код создавался на Delphi 2.

Создайте кнопку с именем NetBtn и DriveComboBox (выпадающий список с дисками) с именем DriveBox. Затем напишите следующий обработчик события OnClick кнопки:

```
procedure TForm1.NetBtnClick(Sender: TObject);
var
  OldDrives: TStringList;
  i: Integer;
begin
  OldDrives := TStringList.Create;
  OldDrives.Assign(DriveBox.Items);      // Запоминаем список дисков
// Показываем диалог подключения
  if WNetConnectionDialog(Handle, RESOURCETYPE_DISK) = NO_ERROR then begin
    DriveBox.TextCase := tcLowerCase;    // Обновляем список дисков
    for i := 0 to DriveBox.Items.Count - 1 do begin
      if OldDrives.IndexOf(Drivebox.Items[i]) = -1 then begin
// Ищем свободный логический диск
        DriveBox.ItemIndex := i;    // Показываем первый найденный логический диск
        DriveBox.Drive := DriveBox.Text[1]; // Каскадируем обновление на список
                                            // подключенных каталогов и др.
```

```
        end;
      end;
      DriveBox.SetFocus;
    end;
    OldDrives.Free;
  end;
```

Самое большое неудобство заключается в том, что у DriveComboBox отсутствует функция обновления. Меняя значение свойства TextCase, заставляем компонент обновляться.

Решение 2

Можно воспользоваться вызовом API WNetAddConnection2. Прототип вызова API расположен в Windows.pas:

```
function WNetAddConnection2W(var lpNetResource: TNetResourceW;
               lpPassword, lpUserName: PWideChar; dwFlags: dword): dword; stdcall;
```

Перед тем как сделать вызов, надо заполнить структуру lpNetResource минимальным количеством параметров, как это показано в примере ниже. В эту структуру передаются, начиная с первого параметра, пароль, имя пользователя, и флаг, указывающий на необходимость восстановления подключения сетевого диска при каждой регистрации машины в сети. Для получения более подробной информации об этой функции обратитесь к справке «Windows Programmers Reference» (найдите функцию в Windows.pas, разместите на ней курсор, и нажмите клавишу <F1> для вызова справки по этой функции).

```
procedure TForm1.Button1Click(Sender: TObject);
var
  NRW: TNetResource;
begin
  with NRW do begin
    dwType := RESOURCETYPE_ANY;
    lpLocalName := 'X:';                    // подключаемся к диску с этой буквой
    lpRemoteName := '\\MyServer\MyDirectory';
// Необходимо заполнить. В случае пустой строки используется значение lpRemoteName.
    lpProvider := '';
  end;
  WNetAddConnection2(NRW, 'MyPassword', 'MyUserName', CONNECT_UPDATE_PROFILE);
end;
```

[News Group]

Перезагрузка Windows из приложения

Решение

```
ExitWindowsEx(EWX_FORCE + EWX_REBOOT, 0)
```

Ошибка отключения сетевого диска

Когда задаете начальный каталог в диалоге открытия файла (в нашем случае это сетевой диск), происходит изменение текущего каталога (что логично). Но, в момент выполнения диалога (когда нажата кнопка OK или Cancel), вы все еще подключены к диску/каталогу, который был указан для данного диалога. Затем, при попытке разорвать соединение с текущим диском, WNetCancelConnection возвращает вам WN_NET_ERROR.

В этом случае проблему может решить ChDir – после диалога открытия файла или перед разрывом сетевого подключения.

[News Group]

Plugins

Выбираем все DLL из каталога с программой, загружаем каждую и пытаемся найти в ней функцию (через API GetProcAddress) с заранее жестко определенным именем (например, IsPluginForMyStuff). Если нашлась, то DLL считается подключаемым модулем («плагином», plug-in), если нет, то мы ее выгружаем и идем дальше.

Набор вызываемых функций, по идее, одинаков для всех подобных модулей. И основная программа «в курсе», какие именно функции она ищет в DLL. Если даже и не так, то ничто не мешает определить в DLL функцию, подобную GetFeatures, возвращающую список строк-названий поддержанных «плагином» процедур.

Часть кода по работе с подключаемыми модулями:

```
type
// Процедурные типы для хранения ссылок на функции плагинов
  TGetNProc = function: shortstring;
  TGetSProc = function: integer;
  TProcessProc = procedure(config: pointer; request: PRequest; var reply: PReply);
  TConfigProc = procedure(defcfg: PSysConfig; var config: pointer);
  TSaveLoadProc = procedure(inifile: pointer; var config: pointer);

// Информация об отдельном плагине
  TPlugin = record
    Name: shortstring;                // Полное название
    Filename: shortstring;            // Имя файла
    Handle: integer;                  // Дескриптор загруженной DLL
    CFGSize: integer;                 // Размер конфигурации в RAM
    ProcessProc: TProcessProc;        // Адрес процедуры обработки
    ConfigProc: TConfigProc;          // Адрес процедуры настройки
    LoadCFG, SaveCFG: TSaveLoadProc;  // Адреса процедур чтения/записи cfg
  end;

  Pplugin = ^TPlugin;
```

```
// Список загруженных плагинов
  TPlugins = class(TList);

  ...

var
  Plugins: TPlugins;
  sr: TSearchRec;
  lib: integer;
  pgetn: TGetNProc;
  pgets: TGetSProc;
  plugin: PPlugin;
  ...
// Читаем плагины и создаем их список.
  Plugins := TPlugins.Create;
  if FindFirst('*.dll', faAnyFile, sr) <> 0 then begin
    ShowMessage('Не найдено подключаемых модулей.');
    Close;
  end;
  repeat
    lib := LoadLibrary(PChar(sr.Name));
    if lib <> 0 then begin
      @pgetn := GetProcAddress(lib, 'GetPluginName');
      if @pgetn = nil then FreeLibrary(lib)              // Не плагин
      else begin
        New(plugin);
        @pgets := GetProcAddress(lib, 'GetCFGSize');
        plugin.Name := pgetn;
        plugin.Filename := sr.Name;
        plugin.CFGSize := pgets;
        plugin.Handle := lib;
        plugin.ConfigProc := GetProcAddress(lib, 'Configure');
        plugin.ProcessProc := GetProcAddress(lib, 'Process');
        plugin.SaveCFG := GetProcAddress(lib, 'SaveCFG');
        plugin.LoadCFG := GetProcAddress(lib, 'LoadCFG');
        Plugins.Add(plugin);
      end;
    end;
  until FindNext(sr) <> 0;
  FindClose(sr);
  ...
```

Минимизация ресурсов, используемых IDE Delphi

Если у вас открыты все формы (показаны или минимизированы), а в редакторе кода открыты все модули, ресурсы очень быстро исчерпываются. Попробуйте закрыть все формы и модули, и открыть только те, которыми будете пользоваться. В противном случае при компиляции можно подвесить Delphi и саму машину.

Hard mode без перерыва

При отладке я наткнулся на сообщение, похожее на системное сообщение Windows, звучащее приблизительно так: «Unable to stop at breakpoint due to Hard Mode» (невозможно остановиться в точке прерывания из-за Hard Mode – дословно «жесткий режим»). Чья это страшилка (Windows или Delphi) и что это такое – «Hard Mode»?

Это Windows. Есть определенные легальные способы заставить перейти Windows в Hard Mode. В этом режиме отладчик прервать невозможно. Тем не менее, можно попробовать знаменитый «Маневр Мебиуса». Он заключается в установке дополнительной точки останова на строке перед той, которая сообщает системе о переходе в режим Hard Mode. Вызов системного модального диалога обычно заставляет систему выйти из этого режима, после чего вторая точка останова сработает как обычно.

Дополнение

Hard Mode представляет собой режим Windows, в котором не происходит никакой обработки сообщений. Это имеет место при отрисовке меню или некоторых операциях ядра. Как следствие, в этом состоянии Delphi не может «заморозить» пользовательское приложение, не блокируя Windows. Обычно это возникает по причине многочисленных вызовов `SendMessage`. В этом случае для выхода из Hard Mode необходимо «встряхнуть» систему. Вполне достаточно, если отладчик отобразит системное модальное окно (`messagebox`), сообщающее о том, что вы находитесь в Hard Mode! Для этого попробуйте поставить дополнительную точку останова (breakpoint) на строчке, предшествующей этой точке останова. В этом случае вы получите предупреждение о том, что система находится в Hard Mode, и этот же диалог «вышибет» систему из данного состояния. При нажатии кнопки OK вторая точка останова сработает как положено.

Примечание

Поскольку работа отладчика построена на обработке сообщений, то он не может остановить работу в точке останова, если он «думает», что система вошла в режим Hard Mode. В этом случае вы не сможете ничего сделать, и система просто-напросто зависнет.

[News Group]

Зависание Delphi 4, 5

Delphi 4, 5 виснут при запуске. Видеокарта S3 Virge.

Решение

Добавьте в реестр строку:

```
[HKEY_CURRENT_CONFIG\Display\Settings]"BusThrottle"="on"
```

Если это не помогает, попробуйте добавить в system.ini:

```
[Display]
"BusThrottle"="On"
```

Эта проблема устранена в Delphi 4 sp3.

[Сахаров Сергей]

Ошибка 1157

При компиляции получаю сообщение «Cannot open c:\delphi 2.0\bin\cmplib32.dll Error code 1157». Что за ошибка такая с кодом 1157? Я пробовал удалить все DCU-*файлы и переустановить* PAS- *и* DFM-*файлы, но ошибка не исчезла. Как это исправить?*

Убедитесь, что все требуемые DLL находятся в пути, установленном в опциях IDE.

[News Group]

Борьба с SoftIce

Полный листинг модуля против SOFTICE. При обнаружении отладчика модуль перезагружает компьютер:

```
unit StopIce;

interface

implementation

uses
  Windows;

function IsSoftIce95Loaded: boolean;
var
  hFile: Thandle;
begin
  result := False;
  hFile := CreateFileA('\\.\SICE', GENERIC_READ or GENERIC_WRITE,
                       FILE_SHARE_READ or FILE_SHARE_WRITE, nil, OPEN_EXISTING,
                       FILE_ATTRIBUTE_NORMAL, 0);
  if(hFile <> INVALID_HANDLE_VALUE) then begin
    CloseHandle(hFile);
    result := True;
  end;
end;

function IsSoftIceNTLoaded: boolean;
var
  hFile: Thandle;
```

```
begin
  result := False;
  hFile := CreateFileA('\\.\NTICE', GENERIC_READ or GENERIC_WRITE,
                       FILE_SHARE_READ or FILE_SHARE_WRITE, nil, OPEN_EXISTING,
                       FILE_ATTRIBUTE_NORMAL, 0);
  if (hFile <> INVALID_HANDLE_VALUE) then begin
    CloseHandle(hFile);
    result := True;
  end;
end;

function WinExit(flags: integer): boolean;

  function SetPrivilege(privilegeName: string; enable: boolean): boolean;
  var
    tpPrev, tp: TTokenPrivileges;
    token: THandle;
    dwRetLen: DWord;
  begin
    result := False;
    OpenProcessToken(GetCurrentProcess, TOKEN_ADJUST_PRIVILEGES or TOKEN_QUERY,
                     token);
    tp.PrivilegeCount := 1;
    if LookupPrivilegeValue(nil, pchar(privilegeName),
                            tp.Privileges[0].LUID) then begin
      if enable then tp.Privileges[0].Attributes := SE_PRIVILEGE_ENABLED
      else tp.Privileges[0].Attributes := 0;
      dwRetLen := 0;
      result := AdjustTokenPrivileges(token, False, tp, SizeOf(tpPrev),
                                      tpPrev, dwRetLen);
    end;
    CloseHandle(token);
  end;

begin
  if SetPrivilege('SeShutdownPrivilege', true) then begin
    ExitWindowsEx(flags, 0);
    SetPrivilege('SeShutdownPrivilege', False);
  end;
end;

initialization
  if IsSoftIce95Loaded or IsSoftIceNTLoaded then begin
    WinExit(EWX_SHUTDOWN or EWX_FORCE);
    Halt;
  end;

end.
```

[News Group]

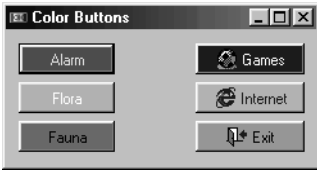
9

Компоненты

Цветная кнопка

В книгах Калверта, Свана и других авторов можно найти текст, смысл которого сводится примерно к следующему: «Изменить цвет кнопок Button, BitBtn нельзя, т. к. их рисует Windows». Если нельзя, но очень нужно, то можно.

Небольшой компонент ColorBtn дает возможность создавать цветные кнопки. Кроме того, представлено новое свойство – Frame3D, позволяющее получить более реалистичный вид нажатой кнопки. В отличие от API, при изменении значения свойства Frame3D не требуется переоткрытие компонента.




```
unit ColorBtn;
interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons;

type
  TColorBtn = class(TButton)
  private
    IsFocused: boolean;
    FCanvas: TCanvas;
    F3DFrame: boolean;
    FButtonColor: TColor;
    procedure Set3DFrame(Value: boolean);
    procedure SetButtonColor(Value: TColor);
```

```
    procedure CNDrawItem(var Message: TWMDrawItem); message CN_DRAWITEM;
    procedure WMLButtonDblClk(var Message: TWMLButtonDblClk);
                              message WM_LBUTTONDBLCLK;
    procedure DrawButtonText(const Caption: string; TRC: TRect;
                             State: TButtonState; BiDiFlags: Longint);
    procedure CalcuateTextPosition(const Caption: string;
                                   var TRC: TRect;
                                   BiDiFlags: Longint);

  protected
    procedure CreateParams(var Params: TCreateParams); override;
    procedure SetButtonStyle(ADefault: boolean); override;

  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
  published
    property ButtonColor: TColor read FButtonColor write SetButtonColor
                          default clBtnFace;
    property Frame3D: boolean read F3DFrame write Set3DFrame default False;
end;

procedure Register;

implementation
{ TColorBtn }

constructor TColorBtn.Create(AOwner: TComponent);
begin
  Inherited Create(AOwner);
  FCanvas := TCanvas.Create;
  FButtonColor := clBtnFace;
  F3DFrame := False;
end;

destructor TColorBtn.Destroy;
begin
  FCanvas.Free;
  Inherited Destroy;
end;

procedure TColorBtn.CreateParams(var Params: TCreateParams);
begin
  Inherited CreateParams(Params);
  with Params do Style := Style or BS_OWNERDRAW;
end;

procedure TColorBtn.Set3DFrame(Value: boolean);
begin
  if F3DFrame <> Value then F3DFrame := Value;
end;
```

```
procedure TColorBtn.SetButtonColor(Value: TColor);
begin
  if FButtonColor <> Value then begin
    FButtonColor := Value; Invalidate;
  end;
end;

procedure TColorBtn.WMLButtonDblClk(var Message: TWMLButtonDblClk);
begin
  Perform(WM_LBUTTONDOWN, Message.Keys, Longint(Message.Pos));
end;

procedure TColorBtn.SetButtonStyle(ADefault: Boolean);
begin
  if IsFocused <> ADefault then IsFocused := ADefault;
end;

procedure TColorBtn.CNDrawItem(var Message: TWMDrawItem);
var
  RC: TRect;
  Flags: Longint;
  State: TButtonState;
  IsDown, IsDefault: Boolean;
  DrawItemStruct: TDrawItemStruct;
begin
  DrawItemStruct := Message.DrawItemStruct^;
  FCanvas.Handle := DrawItemStruct.HDC;
  RC := ClientRect;
  with DrawItemStruct do begin
    IsDown := ItemState and ODS_SELECTED <> 0;
    IsDefault := ItemState and ODS_FOCUS <> 0;
    if not Enabled then State := bsDisabled
    else if IsDown then State := bsDown
    else State := bsUp;
  end;
  Flags := DFCS_BUTTONPUSH or DFCS_ADJUSTRECT;
  if IsDown then Flags := Flags or DFCS_PUSHED;
  if DrawItemStruct.ItemState and ODS_DISABLED <> 0 then
     Flags := Flags or DFCS_INACTIVE;
  if IsFocused or IsDefault then begin
    FCanvas.Pen.Color := clWindowFrame;
    FCanvas.Pen.Width := 1;
    FCanvas.Brush.Style := bsClear;
    FCanvas.Rectangle(RC.Left, RC.Top, RC.Right, RC.Bottom);
    InflateRect(RC, -1, -1);
  end;
  if IsDown then begin
    FCanvas.Pen.Color := clBtnShadow;
    FCanvas.Pen.Width := 1;
    FCanvas.Rectangle(RC.Left, RC.Top, RC.Right, RC.Bottom);
    InflateRect(RC, -1, -1);
```

```
    if F3DFrame then begin
      FCanvas.Pen.Color := FButtonColor;
      FCanvas.Pen.Width := 1;
      DrawFrameControl(DrawItemStruct.HDC, RC, DFC_BUTTON, Flags);
    end;
  end else
    DrawFrameControl(DrawItemStruct.HDC, RC, DFC_BUTTON, Flags);
  FCanvas.Brush.Color := FButtonColor;
  FCanvas.FillRect(RC); InflateRect(RC, 1, 1);
  if IsFocused then begin
    RC := ClientRect;
    InflateRect(RC, -1, -1);
  end;
  if IsDown then OffsetRect(RC, 1, 1);
  FCanvas.Font := Self.Font;
  DrawButtonText(Caption, RC, State, 0);
  if IsFocused and IsDefault then begin
    RC := ClientRect;
    InflateRect(RC, -4, -4);
    FCanvas.Pen.Color := clWindowFrame;
    Windows.DrawFocusRect(FCanvas.Handle, RC);
  end;
  FCanvas.Handle:= 0;
end;

procedure TColorBtn.CalcuateTextPosition(const Caption: string; var TRC: TRect;
                                         BiDiFlags: Integer);
var
  TB: TRect; TS, TP: TPoint;
begin
  with FCanvas do begin
    TB := Rect(0, 0, TRC.Right + TRC.Left, TRC.Top + TRC.Bottom);
    DrawText(Handle, PChar(Caption), Length(Caption), TB,
             DT_CALCRECT or BiDiFlags);
    TS := Point(TB.Right - TB.Left, TB.Bottom - TB.Top);
    TP.X := ((TRC.Right - TRC.Left) - TS.X + 1) div 2;
    TP.Y := ((TRC.Bottom - TRC.Top) - TS.Y + 1) div 2;
    OffsetRect(TB, TP.X + TRC.Left, TP.Y + TRC.Top);
    TRC := TB;
  end;
end;

procedure TColorBtn.DrawButtonText(const Caption: string; TRC: TRect;
                                   State: TButtonState; BiDiFlags: Integer);
begin
  with FCanvas do begin
    CalcuateTextPosition(Caption, TRC, BiDiFlags);
    Brush.Style := bsClear;
    if State = bsDisabled then begin
      OffsetRect(TRC, 1, 1);
      Font.Color := clBtnHighlight;
```

```
        DrawText(Handle, PChar(Caption), Length(Caption), TRC,
                 DT_CENTER or DT_VCENTER or BiDiFlags);
        OffsetRect(TRC, -1, -1);
        Font.Color := clBtnShadow;
        DrawText(Handle, PChar(Caption), Length(Caption), TRC,
                 DT_CENTER or DT_VCENTER or BiDiFlags);
      end else
        DrawText(Handle, PChar(Caption), Length(Caption), TRC,
                 DT_CENTER or DT_VCENTER or BiDiFlags);
    end;
  end;

  procedure Register;
  begin
    RegisterComponents('Controls', [TColorBtn]);
  end;

  end.
```

[VS]

Примечание

Кнопку по-прежнему рисует Windows, а раскрашивает ее компонент `ColorBtn`.

Нажатие кнопки

Я знаю, как нажать кнопку через `KeyPress`*, определил необходимые действия в обработчике события* `OnClick`*. Но с кнопкой на форме не происходит видимых изменений. Кто-нибудь может мне помочь?*

Можно сделать кнопку «нажатой» или «не нажатой», посылая ей сообщение `BM_SETSTATE`. Определить ее текущее состояние можно, послав ей сообщение `BM_GETSTATE`.

Нажатая кнопка:

```
Button1.Perform(BM_SETSTATE, 1, 0);
```

Отжатая кнопка:

```
Button1.Perform(BM_SETSTATE, 0, 0);
```

Чтобы обнаружить нажатие кнопки:

```
ButtonPressed := Button1.Perform(BM_GETSTATE, 0, 0) = 1;
```

[News Group]

Обработка нажатия нескольких кнопок

Необходимо убедиться, что события OnClick привязаны к каждой кнопке и указывают на общий обработчик события.

В разделяемом обработчике события получите заголовок обрабатываемой кнопки следующим образом:

```
Edit1.Text := TButton(Sender).Caption;
```

В этом случае самым разумным будет использование свойства Tag каждой кнопки.

Назначьте уникальный Tag для каждой кнопки (например, эквивалент арабских цифр), тогда:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if (Sender is TButton) then
    with (Sender as TButton) do
{ используем Tag }

end;
```

Если вам нужен только заголовок, то есть изящный способ получить к нему доступ. Подключите общий обработчик события для всех кнопок и используйте приведение типа, как показано ниже:

```
procedure TForm1.Edit1Click(Sender: TObject);
begin
  Edit1.Text := (Sender as TButton).Caption;
end;
```

Конструкции Sender.Caption будет недостаточно, поскольку компилятор не знает о том, имеет ли Sender свойство Caption.

Смена пиктограммы BitBtn во время работы приложения

Пиктограмма компонента является инкапсулированным объектом, хранящимся в некоторой области памяти. Следовательно, при замене пиктограммы память, связанная с первоначальной пиктограммой, должна возвратиться в кучу, а для новой пиктограммы требуется новое распределение памяти. По правилам Delphi, этим должен заниматься метод Assign.

Поместите на форму компоненты OpenDialog, ImageList, Button (2 шт.), BitBtn, Label. Создайте обработчики событий как показано ниже. Первый обработчик демонстрирует процесс замены пиктограммы, второй создан для удобства демонстрации.

```
const
  n: integer = 0;   // счетчик пиктограмм
```

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Image: TBitmap;
begin
  // Изменение пиктограммы в BitBtn1
  Image := TBitmap.Create;
  // пробуем загрузить пиктограммы из ImageList
  if n < ImageList1.Count then ImageList1.GetBitmap(n, Image);
  BitBtn1.Glyph.Assign(Image);
  // Примечание: для изменения свойств объекта используется метод Assign
  inc(n, 2);        // Кнопка содержит две пиктограммы
  if n > ImageList1.Count then n := 0;
  Image.Free;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  // добавляем новую пиктограмму кнопки в список ImageList1
  if OpenDialog1.Execute then
    ImageList1.FileLoad(rtBitMap, OpenDialog1.FileName, clBtnFace);
  Label1.Caption := 'Количество пиктограмм = ' + IntToStr(ImageList1.Count);
end;
```

Кнопка с несколькими строками текста

Как на кнопке поместить текст в нескольких строках?

Решение 1

Кнопка с двумя или более строками текста. Разместите на форме компонент `TBitBtn` и задайте для него достаточно длинный заголовок. Создайте обработчик формы `OnCreate`, как показано ниже:

```
procedure TForm1.FormCreate(Sender: TObject);
var
  R: TRect;
  N: Integer;
  Buff: array[0..255] of Char;
  { другие переменные }
begin
  { ваш текст }
  with BitBtn1 do begin
    Glyph.Canvas.Font := Self.Font;
    Glyph.Width := Width - 6;
    Glyph.Height := Height - 6;
    R := Bounds(0, 0, Glyph.Width, 0);
    StrPCopy(Buff, Caption);
    Caption := '';
    DrawText(Glyph.Canvas.Handle, Buff, StrLen(Buff), R,
             DT_CENTER or DT_WORDBREAK or DT_CALCRECT);
    OffsetRect(R, (Glyph.Width - R.Right) div 2, (Glyph.Height - R.Bottom) div 2);
```

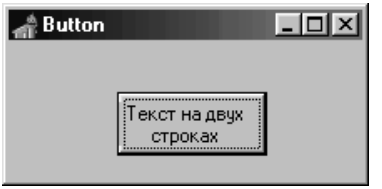
```
    DrawText(Glyph.Canvas.Handle, Buff, StrLen(Buff), R,
             DT_CENTER or DT_WORDBREAK);
  end;
  { ваш текст }
end;
...
```

[News Group]

Примечание

Недостаток этого способа в том, что кнопка не может содержать пиктограмму.

Решение 2




```
unit C_wrapb;
interface

uses
  SysUtils, WinTypes, WinProcs, Messages,
  Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Buttons;

type
  TWrapBtn = class(TBitBtn)
    private
      function GetGlyph: String;
      function GetMargin: Integer;
      function GetSpacing: Integer;
      function GetKind: TBitBtnKind;
      function GetLayout: TButtonLayout;
      function GetNumGlyphs: TNumGlyphs;
      procedure CMTextChanged(var Message: TMessage); message CM_TEXTCHANGED;
      procedure CMFontChanged(var Message: TMessage); message CM_FONTCHANGED;
      procedure WMSize(var Msg: TWMSize); message WM_SIZE;
      procedure CaptionGlyph;
    published
      property Glyph: String Read GetGlyph;
      property Margin: Integer Read GetMargin;
      property Spacing: Integer Read GetSpacing;
      property Kind: TBitBtnKind Read GetKind;
      property Layout: TButtonLayout Read GetLayout;
      property NumGlyphs: TNumGlyphs Read GetNumGlyphs;
  end;

procedure Register;
implementation

procedure TWrapBtn.CaptionGlyph;
var
  GP: TBitmap;
  R: TRect;
```

```
  Buff: array[0..255] of Char;
begin
  GP := TBitmap.Create;
  try
    with GP do begin
      Canvas.Font := Self.Font;
      StrPCopy(Buff, Caption);
      Inherited Margin := 0;
      Inherited Spacing := GetSpacing;
      Width := Self.Width - GetSpacing;
      Height := Self.Height - GetSpacing;
      R := Bounds(0, 0, Width, 0);
      DrawText(Canvas.Handle, Buff, StrLen(Buff), R,
               DT_CENTER or DT_WORDBREAK or DT_CALCRECT);
      OffsetRect(R, (Width - R.Right) div 2, (Height - R.Bottom) div 2);
      DrawText(Canvas.Handle, Buff, StrLen(Buff), R, DT_CENTER or DT_WORDBREAK);
    end;
    Inherited Glyph := GP;
    Inherited NumGlyphs := 1;
  finally
    GP.Free;
  end;
end;

function TWrapBtn.GetGlyph: String;
begin
  Result := '(Н/Д)';
end;

procedure TWrapBtn.CMTextChanged(var Message: TMessage);
begin
  Inherited;
  CaptionGlyph;
end;

procedure TWrapBtn.CMFontChanged(var Message: TMessage);
begin
  Inherited;
  CaptionGlyph;
end;

procedure TWrapBtn.WMSize(var Msg: TWMSize);
begin
  Inherited;
  CaptionGlyph;
end;

function TWrapBtn.GetMargin: Integer;
begin
  Result := 0;
end;
```

```
function TWrapBtn.GetSpacing: Integer;
begin
{$IFDEF Win32}
  Result := 12;
{$ELSE}
  Result := 6;
{$ENDIF}
end;

function TWrapBtn.GetKind: TBitBtnKind;
begin
  Result := bkCustom;
end;

function TWrapBtn.GetLayout: TButtonLayout;
begin
  Result := blGlyphLeft;
end;

function TWrapBtn.GetNumGlyphs: TNumGlyphs;
begin
  Result := 1;
end;

procedure Register;
begin
  RegisterComponents('FAQ', [TWrapBtn]);
end;

end.
```

[News Group]

Примечание

В этом примере тоже нельзя отобразить пиктограмму на кнопке, но при этом мы видим все во время проектирования формы, и «запрещенные» свойства кнопки скрыты от наших глаз.

Альтернатива кнопкам в Delphi

Вместо компонентов `TButton`, `TBitBtn` и т. д. можно использовать компоненты `TImage`, расположив их по 2 или по 3 один над одним, в них помещаются рисунки, заменяющие кнопки:

- простое изображение;
- подсвеченное изображение;
- имитация нажатия.

При проектировании свойству `Visible` второго и третьего компонентов присвоить `False`;

```
procedure TForm1.Image1MouseMove(Sender: TObject; Shift: TShiftState;
                                 X, Y: Integer);
begin
  Image1.Visible := False;
  // При наведении указателя мыши появляется второе изображение
  Image2.Visible := True;
end;

procedure TForm1.Image2MouseDown(Sender: TObject; Button: TMouseButton;
                                 Shift: TShiftState; X, Y: Integer);
begin
  Image2.Visible := False;       // При нажати ЛКМ появляется третье изображение
  Image3.Visible := True;
end;

procedure TForm1.Image2MouseUp(Sender: TObject; Button: TMouseButton;
                               Shift: TShiftState; X, Y: Integer);
begin
  Image2.Visible := False;
  // При отпускании ЛКМ активизируется подсвеченное изображение
  Image3.Visible := True;
end;
```

При перемещении указателя манипулятора на форму активизируется первое изображение:

```
procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
  Image2.Visible := False;
  Image3.Visible := False;
  Image1.Visible := True;
end;
```

[Polyanskiy Alex]

Программное открытие ComboBox

Как программно открыть ComboBox*?*

Используйте недокументированное свойство DroppedDown:

```
ComboBox1.DroppedDown := True;
```

или

```
ComboBox1.Perform(CB_SHOWDROPDOWN, 1, 0);
```

Выпадающий список ComboBox

Хочу реализовать правильный выпадающий список (combo). Как это сделать?

Небольшой пример. Его основная задача – показать принцип работы, а все остальное читатель может реализовать самостоятельно.

```
unit edit1;

interface
uses
  Windows, Messages, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls,
  SysUtils;

type
  TPopupListBox = class(TCustomListbox)
    protected
      procedure CreateParams(var Params: TCreateParams); override;
      procedure CreateWnd; override;
      procedure MouseUp(Button: TMouseButton; Shift: TShiftState;
                        X, Y: Integer); override;
  end;

  TTestDropEdit = class(TEdit)
    private
      FPickList: TPopupListbox;
      procedure CMCancelMode(var Message: TCMCancelMode); message CM_CancelMode;
      procedure WMKillFocus(var Message: TMessage); message WM_KillFocus;
    protected
      procedure CloseUp(Accept: Boolean);
      procedure DropDown;
      procedure WndProc(var Message: TMessage); override;
    public
      constructor Create(Owner: TComponent); override;
      destructor Destroy; override;
  end;

procedure Register;

implementation
{ TPopupListBox }

procedure TPopupListBox.CreateParams(var Params: TCreateParams);
begin
  inherited;
  with Params do begin
    Style := Style or WS_BORDER;
    ExStyle := WS_EX_TOOLWINDOW or WS_EX_TOPMOST;
    WindowClass.Style := CS_SAVEBITS;
  end;
end;

procedure TPopupListbox.CreateWnd;
begin
  inherited CreateWnd;
  Windows.SetParent(Handle, 0);
  CallWindowProc(DefWndProc, Handle, WM_SETFOCUS, 0, 0);
end;
```

```
procedure TPopupListbox.MouseUp(Button: TMouseButton; Shift: TShiftState;
                                X, Y: Integer);
begin
  inherited MouseUp(Button, Shift, X, Y);
  TTestDropEdit(Owner).CloseUp((X >= 0) and (Y >= 0) and (X < Width)
                                and (Y < Height));
end;

{ TTestDropEdit }

constructor TTestDropEdit.Create(Owner: TComponent);
begin
  inherited Create(Owner);
  Parent := Owner as TWinControl;
  FPickList := TPopupListbox.Create(nil);
  FPickList.Visible := False;
  FPickList.Parent := Self;
  FPickList.IntegralHeight := True;
  FPickList.ItemHeight := 11;
  FPickList.Items.CommaText := '1,2,3,4,5,6,7,8,9,0';
end;

destructor TTestDropEdit.Destroy;
begin
  FPickList.Free;
  inherited;
end;

procedure TTestDropEdit.CloseUp(Accept: Boolean);
begin
  if FPickList.Visible then begin
  if GetCapture <> 0 then SendMessage(GetCapture, WM_CANCELMODE, 0, 0);
  SetWindowPos(FPickList.Handle, 0, 0, 0, 0, 0, SWP_NOZORDER
              or SWP_NOMOVE or SWP_NOSIZE or SWP_NOACTIVATE or SWP_HIDEWINDOW);
  if FPickList.ItemIndex <> -1 then
    Text := FPickList.Items.Strings[FPickList.ItemIndex];
  FPickList.Visible := False;
  Invalidate;
  end;
end;

procedure TTestDropEdit.DropDown;
var
  P: TPoint;
  Y: Integer;
begin
  if Assigned(FPickList) and (not FPickList.Visible) then begin
    FPickList.Width := Width;
    FPickList.Color := Color;
    FPickList.Font := Font;
    FPickList.Height := 6 * FPickList.ItemHeight + 4;
```

```
    FPickList.ItemIndex := FPickList.Items.IndexOf(Text);
    P := Parent.ClientToScreen(Point(Left, Top));
    Y := P.Y + Height;
    if Y + FPickList.Height > Screen.Height then Y := P.Y - FPickList.Height;
    SetWindowPos(FPickList.Handle, HWND_TOP, P.X, Y, 0, 0,
                 SWP_NOSIZE or SWP_NOACTIVATE or SWP_SHOWWINDOW);
    FPickList.Visible := True;
    Invalidate;
    Windows.SetFocus(Handle);
  end;
end;

procedure TTestDropEdit.CMCancelMode(var Message: TCMCancelMode);
begin
  if (Message.Sender <> Self) and (Message.Sender <> FPickList) then
    CloseUp(False);
end;

procedure TTestDropEdit.WMKillFocus(var Message: TMessage);
begin
  inherited;
  CloseUp(False);
end;

procedure TTestDropEdit.WndProc(var Message: TMessage);
  procedure DoDropDownKeys(var Key: Word; Shift: TShiftState);
  begin
    case Key of
        VK_UP, VK_DOWN:  if ssAlt in Shift then begin
                           if FPickList.Visible then CloseUp(True) else DropDown;
                           Key := 0;
                         end;
  VK_RETURN, VK_ESCAPE:  if FPickList.Visible and not (ssAlt in Shift) then begin
                           CloseUp(Key = VK_RETURN);
                           Key := 0;
                         end;
    end;
  end;

begin
  case Message.Msg of
      WM_KeyDown, WM_SysKeyDown, WM_Char:
                        with TWMKey(Message) do begin
                          DoDropDownKeys(CharCode, KeyDataToShiftState(KeyData));
                          if (CharCode <> 0) and FPickList.Visible then begin
                            with TMessage(Message) do
                              SendMessage(FPickList.Handle, Msg, WParam, LParam);
                              Exit;
                          end;
                        end;
  end;
```

```
  inherited;
end;

procedure Register;
begin
  RegisterComponents('Controls', [TTestDropEdit]);
end;

end.
```

[Nomadic]

Hint в выпадающем списке ComboBox

В практике программирования довольно часто встречается ситуация, когда информация, предназначенная для отображения в имеющемся компоненте, не помещается по длине. Обычно это имеет место при работе с базами данных. В таких случаях выручают всплывающие подсказки (свойство `Hint`). Но в некоторых случаях даже такая возможность не спасает. К таким ситуациям можно отнести работу с выпадающим списком в `DBComboBox`.

Представьте, что размер поля увеличился, а изменить ширину `DBComboBox` на форме, по тем или иным причинам, нет возможности. Конечно, можно увеличить ширину выпадающего списка. Но выглядит это не всегда красиво, да и не делает чести разработчику.

Предлагаемая идея позволит создать более изящный компонент. Взгляните на рисунок:

В демонстрационном примере в выпадающем списке появляется всплывающая подсказка для строки, не помещающейся по длине.

При работе с `Hint` нужно помнить, что попытки использования `ToolTip` из API – бесполезная затея, т. к Delphi их игнорирует. Для этих целей в Delphi предусмотрен класс `THintWindow`.

В своем компоненте объявите `FHint`:

```
type
  TVSComboBox = class(TCustomComboBox)
```

```
...
  private
   FHint: THintWindow;
...
  protected
   procedure WMCTLCOLORLISTBOX(var Message: TMessage); message WM_CTLCOLORLISTBOX;
...
```

и не забудьте выполнить инициализацию компонента в конструкторе:

```
...
begin
  inherited Create(AOwner);
  FHint := THintWindow.Create(Self);
...
```

Чтобы получить информацию об активной строке в выпадающем списке ComboBox, перехватите сообщение WM_CTLCOLORLISTBOX. В процедуре сообщения проанализируйте строку и, если ее длина больше ширины выпадающего списка, передайте «длинную» строку в FHint и активизируйте его:

```
FHint.ActivateHint(TextRC, Items[ItemIndex]);
```

где

- TextRC – прямоугольник для строки подсказки;
- Items[ItemIndex] – «длинная» строка из выпадающего списка.

Если активная строка в выпадающем списке «короткая» – спрячьте FHint:

```
FHint.ReleaseHandle;
```

Для получения более подробной информации о классе THintWindow обратитесь к справке Delphi.

[VS]

Автоматический формат даты в компоненте Edit

Как в компоненте Edit *выполнить форматирование даты?*

Решение

```
procedure TForm1.Edit1Exit(Sender: TObject);
begin
  if Edit1.Text <> '' then begin
    try
      StrToDate(Edit1.Text);
    except
      Edit1.SetFocus;
      MessageBeep(0);
      raise Exception.Create('"' + Edit1.Text + '" - некорректная дата');
    end;
```

```
    Edit1.Text := DateToStr(StrToDate(Edit1.Text));
  end;
end;
```

Примечание

Для этого предназначен компонент MaskEdit. *Надо настроить маску* EditMask – *и нет проблем*.

Работа с массивом компонентов

На примере TEdit продемонстрируем работу с массивом компонентов.

```
procedure DoSomethingWithEditControls;
var
  K: Integer;
  EditArray: array[0..99] of TEdit;
begin
  try
    for K:=0 to 99 do begin
      EditArray[K] := TEdit.Create(Self);
      EditArray[K].Parent := Self;
      SetSomeOtherPropertiesOfTEdit;   { Устанавливаем необходимые свойства TEdit }
      Left := 100;
      Top := K * 10;
    { Что-то делаем при перемещении мыши }
      OnMouseMove := WhatToDoWhenMouseIsMoved;
    end;
    { Делаем все что хотим с полученным массивом Edit-компонентов }
    DoWhateverYouWantToDoWithTheseEdits;
  finally
    for K:=0 to 99 do EditArray[K].Free;
  end;
end;
```

Примечание

Узнать доступные свойства компонента можно непосредственно в Инспекторе объектов и (или) в текстовом режиме вашей формы (щелкните на форме правой кнопкой мыши и выберите пункт View as Text*)*

Расположение текста в правой части TEdit

Есть какое-нибудь простое решение для расположения текста в правой части TEdit*?*

Решение 1

Вместо этого используйте однострочный TMemo, у которого WordWrap = False, WantReturns = False, Alignment = taRightJustify.

Решение 2

```
TEdit1 = class(TEdit)
  public
    procedure CreateParams(var Params: TCreateParams); Override;
end;
procedure TEdit1.CreateParams(var Params: TCreateParams);
begin
  inherited CreateParams(Params);
  Params.Style := Params.Style or ES_MULTILINE or ES_RIGHT;
end;
```

Ограничение TEdit на ввод нецифровой информации

Как сделать, чтобы в компоненте `TEdit` *можно было вводить только цифры?*

Вставьте следующую строку в обработчик события `OnKeyPress`:

```
if not (Key in ['0'..'9', #8]) then Key := #0;
```

Числовая маска компонента TEdit с помощью OnKeyPress

Решение 1

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
var
  P, L: Byte;
begin
  with Sender as TEdit do begin
    L := Length(Text);
    P := Pos('.', Text);
  end;
  case Key of
    '0'..'9': if (P > 0) and (L - P > 1) then Key := #0;
         '.': if P > 0 then Key := #0;
          #8: { Backspace };
  else Key := #0;
  end;
end;
```

Команда `Key := #0` в обработчике события `OnKeyPress` блокирует символ нажатой клавиши, кроме цифровых символов, символа десятичной точки, символа «забой». Также блокируется цифровая клавиша, если уже присутствуют десятичная точка и две цифры после нее.

[News Group]

Решение 2

Расширенный вариант числовой маски компонента `TEdit` с помощью `OnKeyPress`. В отличие от предыдущего решения, приведенный код не «запирает» поле ввода при заполнении десятичной части, преобразует точку в запятую

(для удобства пользователя), не позволяет поставить десятичную запятую перед числом и позволяет стирать знаки в поле ввода клавишей <BackSpace>.

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
var                                  //цифровая маска
  vrPos, vrLength, vrSelStart: byte;
const
  I: byte = 1;  // I + 1 = количество знаков после запятой (в данном случае 2)
begin
  with Sender as TEdit do begin
    vrLength := Length(Text);        // определяем длину текста
    vrPos := Pos(',', Text);         // проверяем наличие запятой
    vrSelStart := SelStart;          // определяем положение курсора
  end;
  case Key of '0'..'9': begin
// проверяем положение курсора и количество знаков после запятой
                  if (vrPos > 0) and (vrLength - vrPos > I)
                    and (vrSelStart >= vrPos) then
                      Key := #0; // "погасить" клавишу
                end;
      ',', '.': begin
// если запятая уже есть или запятую пытаются поставить перед числом или никаких
// цифр в поле ввода еще нет
                  if (vrPos > 0) or (vrSelStart = 0)
                      or (vrLength =0 ) then Key:= #0   //"погасить" клавишу
                  else Key := #44;            //всегда заменять точку на запятую
                end;
            #8: ;                 // разрешить удаление знаков клавишей 'Backspace'
          else  Key := #0;        // "погасить" все остальные клавиши
  end;
end;
```

[Шпанер Михаил]

Использование SetFocus в OnExit компонента Edit

Я пробую выполнить EditBox.SetFocus *и/или* EditBox.Clear*, но это не дает никакого эффекта (по крайней мере, видимого). Что я делаю неправильно?*

Вы посылаете команду на изменение фокуса внутри обработчика, который сам устанавливает фокус, получаете банальную рекурсию.

Избежать этого можно путем отправления собственного сообщения из обработчика OnExit. После чего в обработчике сообщения надо выставить логический флажок, предохраняющий код от рекурсии. Данный флажок контролируется в обработчике OnExit.

Следующие строки содержат необходимый код:

```
interface
  ...
const
  WM_MyExitRtn = WM_USER + 1001;
```

```
  ...
type
  TForm1 = class(TForm)
  ...
    private
      bExitInProgress: Boolean;          { предохраняемся от рекурсий сообщений }
    public
      procedure WMMyExitRtn(var msg:TMessage); message WM_MyExitRtn;
  end;
  ...
implementation
  ...
procedure TForm1.DBEdit1Exit(Sender: TObject);
begin
 if not bExitInProgress then PostMessage(Handle, WM_MyExitRtn, 0, LongInt(Sender));
end;
procedure TForm1.WMMyExitRtn(var msg:TMessage);
begin
  bExitInProgress := True;               { предохраняемся от рекурсивного вызова }
{ здесь содержится необходимый код }
  bExitInProgress := False;              { сбрасываем флаг }
end;
```

Матрица на основе TEdit

Массив ячеек `TEdit` *– это замечательно. Но я не могу понять, как мне их создавать и делать видимыми?*

Допустим, что они имеют имена с `Edit1` по `Edit9`. Тогда можно попробовать сделать следующее:

```
var
  eds: array[1..3, 1..3] of TEdit;
  ix: integer;
  ed: TEdit;
  ...
  for ix := 0 to 8 do begin
    ed := FindComponent('Edit' + IntToStr(ix + 1)) as TEdit;
    if ed <> nil then eds[ix div 3 + 1, ix mod 3 + 1] := ed;
  end;
  ...
```

Затем, допустим, вам захотелось скопировать текст из строки 1 в строку 2:

```
for ix := 1 to 3 do eds[2, ix].Text := eds[1, ix].Text;
```

[News Group]

Примечание

Управлять «видимостью» и «доступностью» таких ячеек можно посредством стандартных свойств `Tedit` *–* `Visible` *и* `Enabled`.

Отслеживаем позицию курсора в EditBox

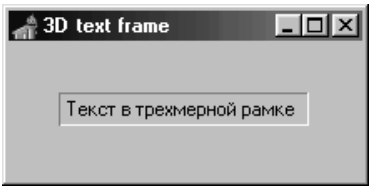
В форму добавляются компоненты TEdit и TLabel, при этом TLabel постоянно отображает позицию курсора в элементе редактирования.

```
var
  CurPos: integer;
...

procedure TForm1.Edit1Change(Sender: TObject);
begin
  CurPos := Edit1.SelStart;
  Label1.Caption := IntToStr(CurPos);
end;

procedure TForm1.Edit1KeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if (Key = VK_LEFT) and (CurPos <> 0) then dec(CurPos);
  if (Key = VK_RIGHT) and (CurPos <= Length(Text)) then inc(CurPos);
  Label1.Caption := IntToStr(CurPos);
end;
```

[Nikolaev Igor]

Трехмерная рамка для текстовых компонентов

Один из примеров создания текстового компонента с декоративной трехмерной контурной рамкой. Он демонстрирует только принцип ее получения.




```
unit IDSLabel;

interface
uses
  Windows, Messages, SysUtils, Classes,
  Graphics, Controls, Forms, Dialogs,
  ExtCtrls;

type
  TIDSLabel = class(TBevel)
    private
      FAlignment: TAlignment;
      FCaption: String;
      FFont: TFont;
      FOffset: Byte;
      FOnChange: TNotifyEvent;
      procedure SetAlignment(taIn: TAlignment);
      procedure SetCaption(const strIn: String);
      procedure SetFont(fntNew: TFont);
      procedure SetOffset(bOffNew: Byte);
    protected
      procedure Paint; override;
```

```
    public
      constructor Create(compOwn: TComponent); override;
      destructor Destroy; override;
    published
      property Alignment: TAlignment read FAlignment write SetAlignment
                          default taLeftJustify;
      property Caption : String read FCaption write SetCaption;
      property Font : TFont read FFont write SetFont;
      property Offset : Byte read FOffset write SetOffset;
      property OnChange : TNotifyEvent read FOnChange write FOnChange;
  end;

procedure Register;

implementation

constructor TIDSLabel.Create;
begin
  inherited Create(compOwn);
  FFont := TFont.Create;
  with compOwn as TForm do FFont.Assign(Font);
  Offset := 4;
  Height := 15;
end;
destructor TIDSLabel.Destroy;
begin
  FFont.Free;
  inherited Destroy;
end;

procedure TIDSLabel.Paint;
var
  wXPos, wYPos: Word;
begin
{ Рисуем рамку }
  inherited Paint;
{ Назначаем шрифт }
  Canvas.Font.Assign(Font);
{ Вычисляем вертикальную позицию }
  wYPos := (Height - Canvas.TextHeight(Caption)) div 2;
{ Вычисляем горизонтальную позицию }
  wXPos := Offset;
  case Alignment of
      taRightJustify: wXPos := Width - Canvas.TextWidth(Caption) - Offset;
            taCenter: wXPos := (Width - Canvas.TextWidth(Caption)) div 2;
  end;

  Canvas.Brush := Parent.Brush;
  Canvas.TextOut(wXPos, wYPos, Caption);
end;
```

```
procedure TIDSLabel.SetAlignment;
begin
  FAlignment := taIn;
  Invalidate;
end;

procedure TIDSLabel.SetCaption;
begin
  FCaption := strIn;
  if Assigned(FOnChange) then FOnChange(Self);
  Invalidate;
end;

procedure TIDSLabel.SetFont;
begin
  FFont.Assign(fntNew);
  Invalidate;
end;

procedure TIDSLabel.SetOffset;
begin
  FOffset := bOffNew;
  Invalidate;
end;

procedure Register;
begin
  RegisterComponents('Controls', [TIDSLabel]);
end;

end.
```

TLabel + TEdit без контейнера

При размещении на форме создается TLabel, расположенный выше поля редактирования. При перемещении поля редактирования TLabel «следует» за ним. При удалении поля редактирования TLabel также удаляется. Имеется свойство LabelCaption, так что вы можете редактировать заголовок TLabel. Вероятно, читателю потребуются и другие свойства TLabel, типа Font, но этот код только демонстрирует технологию, так что развивайте его по своему усмотрению.

```
unit LblEdit;
interface

uses
  Windows, Messages, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls;

type
  TLabelEdit = class(TEdit)
```

```
    private
      FLabel: TLabel ;
      procedure WMMove(var Msg: TWMMove); message WM_MOVE;
    protected
      procedure SetParent(Value: TWinControl); override;
      function GetLabelCaption: string; virtual ;
      procedure SetLabelCaption(const Value: string); virtual ;
    public
      constructor Create(AOwner: TComponent); override;
      destructor Destroy; override;
    published
      property LabelCaption: string read GetLabelCaption write SetLabelCaption;
  end;

procedure Register;

implementation
constructor TLabelEdit.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
{ создаем TLabel }
  FLabel := TLabel.Create(NIL);
  FLabel.Caption := 'Edit label';
end;

procedure TLabelEdit.SetParent(Value: TWinControl);
begin
{ убеждаемся, что TLabel имеет того же родителя, что и TEdit }
  if (Owner = NIL) or not (csDestroying in Owner.ComponentState) then
    FLabel.Parent := Value ;
  inherited SetParent(Value);
end ;

destructor TLabelEdit.Destroy ;
begin
  if (FLabel <> NIL) and (FLabel.Parent = NIL) then FLabel.Free;
  inherited Destroy ;
end ;

function TLabelEdit.GetLabelCaption: string;
begin
  Result := FLabel.Caption;
end ;

procedure TLabelEdit.SetLabelCaption(const Value: string);
begin
  FLabel.Caption := Value;
end;
procedure TLabelEdit.WMMove(var Msg: TWMMove);
begin
  inherited;
```

```
{ заставляем TLabel 'прилипнуть' к верху TEdit }
  if FLabel <> NIL then with FLabel do
  SetBounds(Msg.XPos, Msg.YPos - Height, Width, Height);
end ;

procedure Register;
begin
  RegisterComponents('Samples', [TLabelEdit]);
end;
initialization
{ Используем TLabel, поэтому для обеспечения "поточности" необходима регистрация }
  RegisterClass(TLabel);
end.
```

[News Group]

«Бегущая» строка

Как создать бегущую строку?

С помощью TLabel и TTimer.

```
procedure TForm1.Timer1Timer(Sender: TObject);
const
  LengthGoString = 10;
  GoString = 'В конце строку желательно повторить,'
           + ' чтобы получить эффект кольцевого движения! В конце строки';
  i: Integer = 1;
begin
  Label1.Caption := Copy(GoString, i, LengthGoString); Inc(i);
  if Length(GoString) - LengthGoString < i then i := 1;
end;
```

[Nikolaev Igor]

Примечание

«Окно» просмотра задается константой LengthGoString*, скорость – параметром* Interval *компонента* TTimer.

Советы по работе с палитрой

- Не применяйте свойство stretch для масштабирования в меньшую сторону, для этого следует предпочесть свойство stretchdraw холста изображения.
- Не используйте выходящие за рамки экрана изображения без масштабирования их до размера экрана.
- Если вы хотите изменить изображение, вызовите метод update после метода hide. К сожалению, это не указано в документации.

Изменение палитры при выводе изображения

В обработчике сообщения WM_PaletteChanged можно убедиться, что видимая палитра TImage.Picture.Bitmap.Palette всегда «реализована».

```
  private
    procedure WMPaletteChanged(var Msg: TWMPaletteChanged);
                               message WM_PaletteChanged;
  ...

procedure Form1.WMPaletteChanged(var Msg: TWMPaletteChanged);
begin
  if Msg.PalChg <> Form1.Handle then begin
    PaletteChanged(true);
    Msg.Result := 0;
  end;
end;
```

Теперь можно масштабировать не отображенную картинку как угодно. Следует только помнить, что для вывода не отображенной графики на видимый TImage необходимо вызвать PaletteChanged снова после того, как изображение будет выведено. Исходя из сказанного, приведенный выше код можно дополнить следующим образом:

```
Image1.Picture.Bitmap := obitmap;
PaletteChanged(true);
```

Без этого вызова изображение будет выведено с неправильной палитрой.

Особеннности вывода изображения

В обработчике события формы OnCreate (или, по крайней мере, прежде чем вы покажете форму) попробуйте установить:

```
Image1.ControlStyle := Image1.ControlStyle + [csOpaque];
```

Компоненты TImage изначально прозрачные (за исключением области, занятой изображением), поэтому, сообщая Delphi, что компонент непрозрачен, вы тем самым помешаете что-либо «рисовать» на области, не занятой собственно изображением (типа клиентской области формы).

Рисование прямоугольника на изображении

Как предоствить пользователю возможность нарисовать на изображении прямоугольник, чтобы в дальнейшем выбранную часть, например, масштабировать. Я хотел бы достичь такого же эффекта, как в других программах, где можно при нажатой левой кнопке мыши выделить какую-либо область изображения.

Определите глобальные переменные ImageMouse и ImageRect. Бросьте на форму компонент Image, загрузите в него изображение. Используйте метод DrawFo-

cusRect для рисования контура. Следующий код работает одинаково для левой и правой кнопок мыши:

```
...
var ImageRect: TRect;
    ImageMouse: boolean;
...

procedure TForm1.Image1MouseDown(Sender: TObject; Button: TMouseButton;
                                 Shift: TShiftState; X, Y: Integer);
begin
  ImageMouse := True;
  ImageRect.Left := X;
  ImageRect.Top := Y;
  ImageRect.Right := X;
  ImageRect.Bottom := Y;
  Image1.Canvas.DrawFocusRect(ImageRect);
end;

procedure TForm1.Image1MouseMove(Sender: TObject; Shift: TShiftState;
                                 X, Y: Integer);
var
  NewRect: TRect;
begin
  if ImageMouse then
    if (X > ImageRect.Left) and (Y > ImageRect.Top) then begin
      { Восстанавливаем фон }
      Image1.Canvas.DrawFocusRect(ImageRect);
      { Меняем прямоугольник }
      ImageRect.Right := X;
      ImageRect.Bottom := Y;
      { Рисуем прямоугольник фокуса }
      Image1.Canvas.DrawFocusRect(ImageRect);
    end;
end;

procedure TForm1.Image1MouseUp(Sender: TObject; Button: TMouseButton;
                               Shift: TShiftState; X, Y: Integer);
begin { Восстанавливаем фон }
  if ImageMouse then begin
    ImageRect.Right := X;
    ImageRect.Bottom := Y;
    Image1.Canvas.DrawFocusRect(ImageRect);
    ImageMouse := False;
    Image1.Canvas.CopyRect(Image1.Canvas.ClipRect, Image1.Canvas, ImageRect);
  end;
end;
```

Примечание

Код позволяет лишь увеличивать выделенный фрагмент. Это всего лишь демонстрация технологии.

Множественный выбор в ListBox

Данный пример выводит сообщение для каждого элемента ListBox, выбранного пользователем.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Loop: Integer;
begin
  for Loop := 0 to ListBox1.Items.Count - 1 do begin
    if ListBox1.Selected[Loop] then
      ShowMessage(ListBox1.Items.Strings[Loop]);
  end;
end;
```

Примечание

Необходимо в Object Inspector для ListBox1 *установить свойство* MultiSelect *в* True.

Изменение позиций элементов ListBox с помощью Drag&Drop

Я хотел бы изменить порядок следования элементов в неотсортированном списке ListBox *методом Drag & Drop, т. е. просто перетаскивая их мышью на нужное место. Как это сделать?*

Решение 1

```
procedure TForm1.ListBox1DragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  with (Sender as TListBox) do Items.Move(ItemIndex, ItemAtPos(Point(x, y), True));
end;

procedure TForm1.ListBox1DragOver(Sender, Source: TObject; X, Y: Integer;
                                  State: TDragState; var Accept: Boolean);
begin
  Accept := Sender = Source;
end;
```

Не забудьте в ListBox присвоить свойству DragMode значение dmAutomatic.

Решение 2

Чтобы заставить элемент перемещаться в позицию другого элемента, необходимо сопоставлять область текущего элемента с текущим положением курсора мыши. Для организации автоматической прокрутки также необходимо вычислять текущие координаты курсора.

```
unit Draglb;

interface
```

```
uses
  Windows, Messages, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls;
type
  TDragListBox = class(TListBox)
    public
      procedure DragOver(Sender, Source: TObject; X, Y: Integer; State: TDragState;
                         var Accept: Boolean);
      procedure DragDrop(Sender, Source: TObject; X, Y: Integer);
      constructor Create(AOwner: TComponent); override;
end;

procedure Register;
implementation

procedure Register;
begin
  RegisterComponents('Custom', [TDragListBox]);
end;

constructor TDragListBox.Create(AOwner: TComponent);
begin
  Inherited Create(AOwner);
  DragMode := dmAutomatic;
  OnDragDrop := DragDrop;
  OnDragOver := DragOver;
end;

procedure TDragListBox.DragOver(Sender, Source: TObject; X, Y: Integer;
                                State: TDragState; var Accept: Boolean);
begin
  Accept := Source = Self;
end;

procedure TDragListBox.DragDrop(Sender, Source: TObject; X, Y: Integer);
var
  Value: Integer;
begin
  if Sender = Self then begin
    Value := Self.ItemAtPos(Point(x, y), True);
    if Value = -1 then begin
      Self.Items.Add(Self.Items[Self.ItemIndex]);
      Self.Items.Delete(Self.ItemIndex);
    end else begin
      Self.Items.Insert(Value {+ 1}, Self.Items[Self.ItemIndex]);
      Self.Items.Delete(Self.ItemIndex);
    end;
  end;
end;
end.
```

[News Group]

Решение 3

```
var
  DraggedPM: integer;

...

procedure TForm1.ListBox1MouseDown(Sender: TObject; Button: TMouseButton;
                                   Shift: TShiftState; X, Y: Integer);
begin
  if Button = mbLeft then
    with Sender as TListBox do begin
      DraggedPM := ItemAtPos(Point(X, Y), True);
      if DraggedPM >= 0 then BeginDrag(False);
    end;
end;

procedure TForm1.ListBox1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  if Source = ListBox1 then Accept := True;
end;

procedure TForm1.ListBox1DragDrop(Sender, Source: TObject; X, Y: Integer);
var
  NewIndex: integer;
begin
  NewIndex := ListBox1.ItemAtPos(Point(X, Y), False);
  if NewIndex > ListBox1.Items.Count - 1 then
    NewIndex := ListBox1.Items.Count - 1;
  ListBox1.Items.Move(DraggedPM, NewIndex);
  ListBox1.ItemIndex := NewIndex;
end;
```

[News Group]

Улучшение компонента ListBox

Как вставить графику в `ListBox` *или* `ComboBox`*?*

Пример, иллюстрирующий данную технологию...

- Создайте форму.
- Расположите на ней компоненты `ComboBox` и `ListBox`.
- Измените свойство `Style` компонента `ComboBox` на `csOwnerDrawVariable` и свойство `Style` компонента `ListBox` на `lbOwnerDrawVariable`. Обработчик события `OnDrawItem` компонентов `TListBox` или `TComboBox` позволяет осуществить вывод как объекта (графики), так и строки элемента. В данном примере осуществляется вывод как графического объекта, так и строки.

- Создайте 5 переменных типа TBitmap в var-секции вашей формы.
- Создайте обработчики для событий формы OnCreate и OnClose.
- Создайте для ComboBox и ListBox обработчики события OnDrawItem.
- Создайте для ComboBox и ListBox обработчики события OnMeasureItem.

```
unit Unit1;

interface

uses
  Windows, Messages, Classes, Graphics, Controls, Forms, Dialogs, ExtCtrls,
  StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    ComboBox1: TComboBox;
    ListBox1: TListBox;
    procedure FormCreate(Sender: TObject);
    procedure ComboBox1MeasureItem(Control: TWinControl; Index: Integer;
      var Height: Integer);
    procedure ComboBox1DrawItem(Control: TWinControl; Index: Integer;
      Rect: TRect; State: TOwnerDrawState);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure ListBox1DrawItem(Control: TWinControl; Index: Integer;
      Rect: TRect; State: TOwnerDrawState);
    procedure ListBox1MeasureItem(Control: TWinControl; Index: Integer;

      var Height: Integer);
 end;

var
  Form1: TForm1;
  TheBitMap1, TheBitMap2, TheBitMap3, TheBitMap4, TheBitMap5: TBitMap;

implementation
{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
const
  path = 'C:\Program Files\Common Files\Borland Shared\images\buttons\';
begin
  TheBitmap1 := TBitmap.Create;
  TheBitmap1.LoadFromFile(path + 'globe.bmp');
  TheBitmap2 := TBitmap.Create;
  TheBitmap2.LoadFromFile(path + 'video.bmp');
  TheBitmap3 := TBitmap.Create;
  TheBitmap3.LoadFromFile(path + 'gears.bmp');
  TheBitmap4 := TBitmap.Create;
  TheBitmap4.LoadFromFile(path + 'key.bmp');
```

```
  TheBitmap5 := TBitmap.Create;
  TheBitmap5.LoadFromFile(path + 'tools.bmp');
  ComboBox1.Items.AddObject('Изображение1: Глобус', TheBitmap1);
  ComboBox1.Items.AddObject('Изображение2: Видео', TheBitmap2);
  ComboBox1.Items.AddObject('Изображение3: Механизм', TheBitmap3);
  ComboBox1.Items.AddObject('Изображение4: Ключ', TheBitmap4);
  ComboBox1.Items.AddObject('Изображение5: Инструмент', TheBitmap5);
  ListBox1.Items.AddObject('Изображение1: Глобус', TheBitmap1);
  ListBox1.Items.AddObject('Изображение2: Видео', TheBitmap2);
  ListBox1.Items.AddObject('Изображение3: Механизм', TheBitmap3);
  ListBox1.Items.AddObject('Изображение4: Ключ', TheBitmap4);
  ListBox1.Items.AddObject('Изображение5: Инструмент', TheBitmap5);
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  TheBitmap1.Free;
  TheBitmap2.Free;
  TheBitmap3.Free;
  TheBitmap4.Free;
  TheBitmap5.Free;
end;

procedure TForm1.ComboBox1DrawItem(Control: TWinControl; Index: Integer;
                                   Rect: TRect; State: TOwnerDrawState);
var
  Bitmap: TBitmap;
  Offset: Integer;
begin
  with (Control as TComboBox).Canvas do begin
    FillRect(Rect);
    Bitmap := TBitmap(ComboBox1.Items.Objects[Index]);
    if Bitmap <> nil then begin
      BrushCopy(Bounds(Rect.Left + 2, Rect.Top + 2, Bitmap.Width, Bitmap.Height),
                Bitmap, Bounds(0, 0, Bitmap.Width, Bitmap.Height), clRed);
      Offset := Bitmap.width + 8;
    end;
    { отображаем текст }
    TextOut(Rect.Left + Offset, Rect.Top, Combobox1.Items[Index])
  end;
end;

procedure TForm1.ComboBox1MeasureItem(Control: TWinControl; Index: Integer;
                                      var Height: Integer);
begin
  height := 20;
end;

procedure TForm1.ListBox1DrawItem(Control: TWinControl; Index: Integer;
                                  Rect: TRect; State: TOwnerDrawState);
```

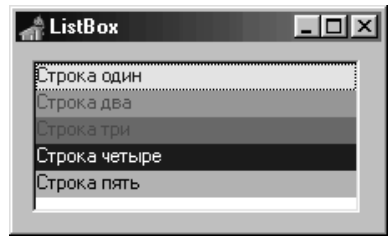
```
var
  Bitmap: TBitmap;
  Offset: Integer;
begin
  with (Control as TListBox).Canvas do begin
    FillRect(Rect);
    Bitmap := TBitmap(ListBox1.Items.Objects[Index]);
    if Bitmap <> nil then begin
      BrushCopy(Bounds(Rect.Left + 2, Rect.Top + 2, Bitmap.Width, Bitmap.Height),
                Bitmap, Bounds(0, 0, Bitmap.Width, Bitmap.Height), clRed);
      Offset := Bitmap.width + 8;
    end;
    { отображаем текст }
    TextOut(Rect.Left + Offset, Rect.Top, Listbox1.Items[Index])
  end;
end;

procedure TForm1.ListBox1MeasureItem(Control: TWinControl; Index: Integer;
                                     var Height: Integer);
begin
  height := 20;
end;
end.
```

Использование цвета в ListBox

Что нужно сделать, чтобы каждая строка в ListBox *имела свой цвет?*

Приведем пример обработчика события OnDrawItem.

```
procedure TForm1.ListBox1DrawItem(Control:
              TWinControl; Index: Integer;
     Rect: TRect; State: TOwnerDrawState);

begin
  with (Control As TListBox).Canvas do begin
    case (Index mod 3) of
      0: begin
           Font.Color := clBlue;
           Brush.Color := clYellow;
         end;
      1: begin
           Font.Color := clRed;
           Brush.Color := clLime;
         end;
      2: begin
           Font.Color := clGreen;
           Brush.Color := clFuchsia;
         end;
    end;
```



```
      FillRect(Rect);
      TextOut(Rect.Left, Rect.Top, (Control as TListBox).Items[Index]);
    end;

end;
```

Вышеприведенный код устанавливает различный цвет для фона и текста в зависимости от номера строки, но он не работает с выделенными/выбранными строками. (Не забудьте установить значение свойства `ListBox1.Style` равным `lbOwnerDrawFixed`.) Для примера выбраны три цвета, которыми циклически закрашиваются строки списка.

Есть несколько вещей, достойных упоминания.

Используется приведение типов `(Control as TListBox).Items[Index]`. Такой способ позволяет сделать общим обработчик события для нескольких компонентов `TListBox`, например, если имеется пара `ListBox` на различных страницах `TNoteBook`.

Можно создать небольшое свободное пространство вокруг текста. Для этого увеличьте `TListBox.ItemHeight` и, соответственно, область вывода текста – `TempLeft := Rect.Left + 3` и `TempTop := Rect.Top + 1`.

Инкрементный поиск в ListBox

Решение 1

Предположим, что `ListBox` сортируется (`Sorted = True`). Необходимо разместить компонент `Edit` выше `ListBox` и создать следующий обработчик его события `OnChange`:

```
procedure TForm1.Edit1Change(Sender: TObject);

var
  Ndx: Word;

begin
  with (Sender as TEdit) do begin
    Ndx := ListBox1.Items.Add(Text);
    ListBox1.Items.Delete(Ndx);
    if CompareText(Text, Copy(ListBox1.Items[Ndx], 1, Length(Text))) = 0 then
      ListBox1.ItemIndex := Ndx
    else ListBox1.ItemIndex := -1;
  end;

end;
```

Пытаясь вставить часть текста, надо просто просматривать список на предмет его наличия. Если актуальный элемент в этой позиции содержит «частичный» текст, мы выводим его, в противном случае делаем так, чтобы `ListBox` не имел выделенного (`ItemIndex`) элемента.

Решение 2

Я видел приложение, в котором ListBox *позволял осуществлять инкрементный поиск. При вводе очередного символа* ListBox *позиционирует вас на первую ячейку, начало значения которой совпадает с введенным пользователем текстом, или выделяет все строки с текстом, содержащим введенный текст. Как это осуществить на Delphi?*

Здесь придется прибегнуть к Win API. Установите свойство формы KeyPreview в True и сделайте примерно следующее:

```
unit LbxSrch;

interface

uses
  Windows, Messages, SysUtils, Classes, Controls, Forms, StdCtrls;

type
  TFrmLbxSrch = class(TForm)
      Edit1: TEdit;
      Edit2: TEdit;
      ListBox1: TListBox;
      Label1: TLabel;
      procedure FormKeyPress(Sender: TObject; var Key: Char);
      procedure ListBox1Enter(Sender: TObject);
    private
      FPrefix: array[0..255] of char;
end;

var
  FrmLbxSrch: TFrmLbxSrch;

implementation
{$R *.DFM}

procedure TFrmLbxSrch.FormKeyPress(Sender: TObject; var Key: Char);
{ Помните о том, что свойство KeyPreview должно быть установлено в True }
var
  curKey: array[0..1] of char;
  ndx: integer;
begin
  if ActiveControl = ListBox1 then begin
    if key = #8 { Backspace (клавиша возврата) } then begin
      if FPrefix[0] <> #0 then begin
        FPrefix[StrLen(FPrefix) - 1] := #0;
      end;
    end else begin
      curKey[0] := Key;
      curKey[1] := #0;
      StrCat(FPrefix, curKey);
      ndx := SendMessage(ListBox1.Handle, LB_FINDSTRING, -1, longint(@FPrefix));
```

```
      if ndx <> LB_ERR then ListBox1.ItemIndex := ndx;
    end;
    Label1.Caption := StrPas(FPrefix);
    Key := #0;
  end;
end;
procedure TFrmLbxSrch.ListBox1Enter(Sender: TObject);
begin
  FPrefix[0] := #0;
  Label1.Caption := StrPas(FPrefix);
end;
end.
```

Примечание

У этого примера есть один незаметный, на первый взгляд, недостаток. Если вводить какое-либо значение, состоящее более чем из одного символа, а затем начать стирать его (ошибочный символ), то поиск никак не отреагирует на это, т. е., чтобы опять начал работать поиск, необходимо удалить всю строку и набрать ее заново, что не есть хорошо. Предлагаю модернизированный вариант процедуры `TFrmLbxSrch.FormKeyPress` *без указанного недостатка:*

```
procedure TFrmLbxSrch.FormKeyPress(Sender: TObject; var Key: Char);
{ Помните о том, что свойство KeyPreview должно быть установлено в True }
var
  curKey: array[0..1] of char;
  ndx: integer;
begin
  if ActiveControl = ListBox1 then begin
    if key = #8 { Backspace (клавиша возврата) } then begin
      if FPrefix[0] <> #0 then FPrefix[StrLen(FPrefix) - 1] := #0;
    end else begin
      curKey[0] := Key;
      curKey[1] := #0;
      StrCat(FPrefix, curKey);
    end;
    ndx := SendMessage(ListBox1.Handle, LB_FINDSTRING, -1, longint(@FPrefix));
    if ndx <> LB_ERR then ListBox1.ItemIndex := ndx;
    Label1.Caption := StrPas(FPrefix);
    Key := #0;
  end;
end;
```

Уменьшение мерцания ListBox в обработчике OwnerDraw

Предположим, `ListBox` содержит в своем списке два элемента, элемент 0 имеет фокус, активен другой компонент, и вы щелкаете по элементу 1. При этом происходит пятикратный вызов `OnDrawItem`. Ниже приведены изменения состояний двух элементов:

Index	State
0	[odSelected, odFocused]
0	[odSelected]
0	[]
1	[odSelected]
1	[odSelected, odFocused]

В случае единственного элемента в списке ListBox получается конфуз, поскольку при щелчке по нему мы получаем тот же самый сценарий, только вместо двух индексов присутствует один, нулевой.

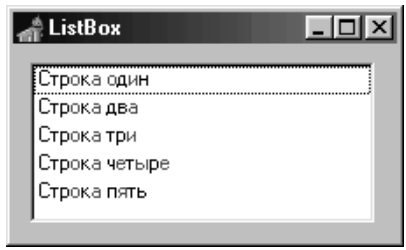
Имея эту информацию, можно свести к минимуму количество вызовов процедуры отрисовки. Предположим, что в компоненте ListBox запрещен множественный выбор элементов (MultiSelect = False). В этом случае, если элемент находится в состоянии [odSelected], отрисовывать его не нужно. Действительно, поскольку в нашем случае исключаются ситуации, когда элемент одновременно имеет состояния selected и focused или не имеет ни одного из них. В этом вам поможет технология отслеживания в обработчике OnDrawItem предыдущего отрисованного элемента, и если предыдущий запомненный элемент равен текущему, то отрисовывать его не обязательно, например:

```
  ...
const
  LastIndex: LongInt = -1;
begin
  if Index = LastIndex then
  ...
  else
  ...
  LastIndex := Index;
end;
```

[News Group]

Пример Ownerdraw для Listbox

Пример обработчика OnDrawItem, выводящий гласные красным цветом:



```
procedure TForm1.ListBox1DrawItem(Control: TWinControl; Index: Integer;
  Rect: TRect; State: TOwnerDrawState);
var
  S: String;
  Sym: Char;
  N: Word;
  WasColor: TColor;
begin
  with (Control as TListBox), Canvas do begin
    S := Items[Index];
    FillRect(Rect);
    MoveTo(Rect.Left + 2, Rect.Top);
    SetTextAlign(Canvas.Handle, TA_LEFT OR TA_UPDATECP);
    WasColor := Font.Color;
    for N := 1 to Length(S) do begin
      Sym := AnsiUpperCase(S[N])[1];
      case Sym of
      'А','Е','И','Й','О','У','Ы','Ю','Я': Font.Color := clRed;
                                       else Font.Color := WasColor;
      end;

      Windows.TextOut(Canvas.Handle, 0, 0, @S[N], 1);
    end;
  end;
end;
```

Обратите внимание, что для использования стиля TA_UPDATECP (при котором каждый следующий вызов TextOut выводит текст в позицию, расположенную после предшествующей), необходимо вызывать функцию API Windows.TextOut вместо метода объекта Delphi Canvas.TextOut.

[News Group]

Примечание

Не забудьте установить свойство Style *компонента* ListBox *в* lbOwnerDrawFixed *или* lbOwnerDrawVariable.

Прокрутка в TListBox

Как определить, что пользователь двигает полосы прокрутки в TListbox*?*

Например, так:

```
unit Listbob;

interface

uses
  Windows, Messages, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls;
```

```
type
  TListBob = class(TListBox)
    private
      FOnHScroll: TNotifyEvent;
      FOnVScroll: TNotifyEvent;
    protected
      procedure WMHScroll(var Message: TWMHScroll); message WM_HSCROLL;
      procedure WMVScroll(var Message: TWMVScroll); message WM_VSCROLL;
    public
      constructor Create(AOwner: TComponent); override;
    published
      property OnHScroll: TNotifyEvent read FOnHScroll write FOnHScroll;
      property OnVScroll: TNotifyEvent read FOnVScroll write FOnVScroll;
  end;

procedure Register;
implementation
constructor TListBob.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  FOnHScroll := nil;
  FOnVScroll := nil;
end;

procedure TListBob.WMHScroll(var Message: TWMHScroll);
{ помните, что данное сообщение вызывается дважды!! }
begin
  if Assigned(FOnHScroll) then FOnHScroll(Self);
  DefaultHandler(Message);
end;

procedure TListBob.WMVScroll(var Message: TWMHScroll);
{ помните, что данное сообщение вызывается дважды!! }
begin
  if Assigned(FOnVScroll) then FOnVScroll(Self);
  DefaultHandler(Message);
end;

procedure Register;
begin
  RegisterComponents('Controls', [TListBob]);
end;
end.
```

[News Group]

Щелчок в пустой области TListBox

Я хочу, чтобы TListBox *имитировал поведение окна* Watch List *из IDE Delphi, отвечающего на двойной щелчок открытием диалогового окна. Я добился такого эффекта, добавив к форме обработку события* OnDoubleClick. *Но в случае двойного щелчка в области, расположенной ниже последнего элемента списка, я хочу открывать диалоговое окно* Добавить новый элемент. *Но вот это никак не получается. Метод* OnDoubleClick *не вызывается, если я щелкаю в* TListBox, *но щелчок приходится не на элемент списка, а на пустое место.*

Добавьте следующий обработчик OnMouseDown в ваш ListBox:

```
procedure TForm1.ListBox1MouseDown(Sender: TObject; Button: TMouseButton;
                                   Shift: TShiftState; X, Y: Integer);
begin
  if (Button = mbLeft) then begin
    with Sender as TListBox do begin
      if ((ItemAtPos(Point(X, Y), True) = -1) and (ssDouble in Shift)) then begin
{ двойной щелчок в пустой области }
      end;
    end;
  end;
end;
```

[News Group]

Использование выбранных элементов TListBox

Как переместить выбранные элементы в StringList. *Или, например, вообще удалить невыбранные элементы из* ListBox?

Следующий код перемещает все выбранные элементы из ListBox1 в ListBox2:


```
procedure TForm1.Button1Click(Sender: TObject);
var
  ix: integer;
```

```
begin
  with ListBox1 do begin
    if SelCount < 1 then exit;
    for ix := 0 to Items.Count - 1 do
      if Selected[ix] then ListBox2.Items.Add(Items[ix]);
    for ix := Items.Count-1 downto 0 do
      if Selected[ix] then Items.Delete(ix);
  end;
end;
```

[News Group]

Примечание

Свойство `MultiSelect` *должно быть установлено в* `True`.

Расширение TListBox

Я пытаюсь осуществить предварительную загрузку элементов наследника `TListBox` *с множеством строк. Для этого я перекрываю конструктор, добавляя в него строки типа* `Items.Add('foo');` *но когда я выполняю это, то получаю исключение «Window has no parent window» (окно не имеет родительского окна). Почему?*

`ListBox` сохраняет элементы, передавая их Windows. При этом требуется дескриптор окна, а дескриптору окна требуется родитель. Родитель не устанавливается даже после возврата из конструктора.

Решение:

```
  ...
SaveVis := Visible;
Visible := False;
Parent := Owner;

  // заполнение ListBox
Parent := Nil;
Visible := SaveVis;
  ...
```

[News Group]

Табуляция в графическом ListBox

Использование табуляции в `ListBox`*, когда компонент находится в стандартном режиме, не составляет труда. Но что делать, если надо отображать элементы списка как графику?*

Решение

```
procedure TForm1.ListBox1DrawItem(Control: TWinControl; Index: Integer;
  Rect: TRect; State: TOwnerDrawState);
```

```
var
  S, Ss: String;
  P: Integer;                        // Флаг символа разделителя
begin
  ListBox1.Canvas.FillRect(Rect);
// Отрисовка графики
  ...
  S := ListBox1.Items.Strings[Index];
  P := Pos('|', S);
  if P = 0 then Ss := S
  else Ss := Copy(S, 1, P - 1);     // Если нет табуляции, то пишем всю строку,
                                    // иначе отрезаем фрагмент до разделителя
  ListBox1.Canvas.TextOut(Rect.Left + 20, Rect.Top + 2, Ss);
  if P > 0 then
    ListBox1.Canvas.TextOut(ListBox1.TabWidth, Rect.Top + 2,
                            Copy(S, P + 1, Length(S) - P + 2));
  ...
end;
```

[Virtualik]

Примечание

*Стиль (*Style*)* ListBox1 *установите в* lbOwnerDrawVariable *и введите несколько строк в* Items.

Выравнивание в ListBox

Перед тем как вычислить позицию фразы, необходимо с помощью функции TextWidth вычислить ее ширину.

```
  ...
var
  J, TempInt, LongPrefixLen, CurrPrefixLen: Integer;
begin
{ Вычисляем TextWidth по ключевой строке. Устанавливаем CurrPrefixLen в TextWidth
  ключевого слова строки Indexth }
  LongPrefixLen := 0;
  for J := 0 to ListBox1.Items.Count - 1 do
    with ListBox1 do begin
      TempInt := Canvas.TextWidth(Copy(Items[J], 1, Pos(KeyString, Items[J] - 1)));
      if LongPrefixLen < TempInt then LongPrefixLen := TempInt;
      if J = Index then CurrPrefixLen := TempInt;
    end;
{ PrevTextLeft - TextLeft = Где мы хотим вывести новый элемент }
  TextOut(LongPrefixLen - CurrPrefixLen, Y, Items[I]);
end;
```

ListBox с графикой

Мне нужно в ListBox *вставить картинку, но как это сделать?*

Необходимо установить свойство ListBox Style в lbOwnerDrawFixed. Затем в обработчике события DrawItem попытаемся нарисовать изображение.

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    ListBox1: TListBox;
    Button1: TButton;
    procedure FormActivate(Sender: TObject);
    procedure ListBox1DrawItem(Control: TWinControl; Index: Integer;
      Rect: TRect; State: TOwnerDrawState);
  end;

var
  Form1: TForm1;

implementation
{$R *.dfm}

procedure TForm1.FormActivate(Sender: TObject);
var
  Dibujo: TIcon;
begin
  with ListBox1.Items do begin
    Dibujo := TIcon.Create;
    Dibujo.LoadFromFile('c:\Program Files\Common Files
                         \Borland Shared\Images\Icons\chemical.ico');
    AddObject('Poder Legislativo y Partidos Politicos', Dibujo);
    Dibujo := TIcon.Create;
    Dibujo.LoadFromFile('c:\Program Files\Common Files
                        \Borland Shared\Images\Icons\chip.ico');
    AddObject('Poder Ejecutivo Nacional', Dibujo);
  end;
end;

procedure TForm1.ListBox1DrawItem(Control: TWinControl; Index: Integer;
                                  Rect: TRect; State: TOwnerDrawState);
var
  Icon: TIcon;
  Offset: Integer;                    { ширина отступа текста }
```

```
begin
  with (Control as TListBox).Canvas do begin
 { рисуем на холсте элемента управления, не на форме }
    FillRect(Rect);                    { очищаем прямоугольник }
    Offset := 2;                       { обеспечиваем отступ по умолчанию }
{ получаем пиктограмму для данного элемента }
    Icon := TIcon((Control as TListBox).Items.Objects[Index]);
    if Icon <> nil then begin
      Draw(Rect.Left + 1, Rect.Top + 2,
           TIcon((Control as TListBox).Items.Objects[Index]));
{ добавляем пикселы между пиктограммой и текстом }
      Offset := Icon.Width + 9;
    end;
    TextOut(Rect.Left + Offset, Rect.Top + 7,
           (Control as TListBox).Items[Index]) { выводим текст }
  end;
end;

end.
```

Воспользуйтесь событием OnDrawItem объекта ListBox (ComboBox или др.). В его обработчике рисовать графику так же легко, как писать текст. (Полное управление можно получить, лишь подключив обработку события OnMeasureItem.)

```
procedure ListDrawItem(Control: TWinControl; Index: Integer;
                       Rect: TRect; State: TOwnerDrawState);
var
  BitMap: TBitMap;

begin
{ Здесь инициализируем Bitmap... например, загружаем в него изображение }
  with (Control as TListBox).Canvas do begin
    FillRect(Rect);
    Draw(Rect.Left, Rect.Top, BitMap);
{ DstList - имя списка }
    TextOut(Rect.Left + 2 + BitMap.Width, Rect.Top, DstList.Items.Strings[Index]);
  end;
end;
```

Горизонтальная полоса прокрутки в TListBox

Как добавить горизонтальную полосу прокрутки в TListBox*?*

Компонент TListBox автоматически реализует вертикальную полосу прокрутки. Полоса прокрутки появляется, когда окно списка слишком мало для отображения всех элементов списка. Однако окно списка не содержит горизонтальной полосы прокрутки, если какие-либо элементы списка имеют большую ширину, чем само окно списка. Но есть возможность добавить горизонтальную полосу прокрутки. Добавьте следующий код в обработчик события OnCreate своей формы:

```
procedure TForm1.FormCreate(Sender: TObject);
var
  i, MaxWidth: integer;
begin
  MaxWidth := 0;
  for i := 0 to ListBox1.Items.Count - 1 do
    if MaxWidth < ListBox1.Canvas.TextWidth(ListBox1.Items.Strings[i]) then
      MaxWidth := ListBox1.Canvas.TextWidth(ListBox1.Items.Strings[i]);
  SendMessage(ListBox1.Handle, LB_SETHORIZONTALEXTENT, MaxWidth + 2, 0);
end;
```

[Nomadic]

Динамическое добавление пунктов меню

Пример программы, создающей структуру меню большой вложенности двумя различными способами. Она даст вам пищу для размышлений. Форма содержит компонент MainMenu.

```
unit Istopmnu;

interface

uses
  Windows, SysUtils, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Menus;

type
  TForm1 = class(TForm)
    MainMenu1: TMainMenu;
    procedure AClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
end;

var
  Form1: TForm1;

implementation
{$R *.DFM}

procedure TForm1.AClick(Sender: TObject);
var
  TM: TMenuItem;
  Lev: Word;
begin
  MessageBeep(0);
  TM := Sender as TMenuItem;
  if TM.Count > 0 then Caption := 'подменю'
  else Caption := 'элемет меню';
  Caption := Caption + ' с именем "' + TM.Name + '"';
  Lev := 0;
```

```
  while (TM.Parent <> NIL) and (TM.Parent is TMenuItem) do begin
    TM := TM.Parent;
    Inc(Lev);
  end;
  case Lev of
    1: Caption := 'Верхний уровень ' + Caption;
    2: Caption := '2-й уровень ' + Caption;
    3: Caption := '3-й уровень ' + Caption;
  else Caption := Format('%d-й уровень %s', [Lev, Caption]);
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  TM: TMenuItem;
  N: Integer;
begin
  TM := MainMenu1.Items;
  TM.Add(NewItem('&Элемент', 0, False, True, AClick, 0, 'MenuItem2'));
  for N := 2 to 5 do begin
    TM.Add(TMenuItem.Create(nil));
    TM := TM.Items[TM.Count - 1];
    TM.Caption := '&Меню';
    TM.Name := 'SubMenu' + IntToStr(N);
    TM.OnClick := AClick;
    TM.Add(NewItem('&Элемент', 0, False, True, AClick, 0, 'MenuItem' +
                   IntToStr(N + 1)));
  end;
  MainMenu1.Items.Add(NewSubMenu('Меню&2', 0, 'SM1',
    [NewItem('&Элемент', 0, False, True, AClick,0,'MI2'),
     NewSubMenu('&Меню', 0, 'SM2',
    [NewItem('&Элемент', 0, False, True, AClick,0,'MI3'),
     NewSubMenu('&Меню', 0, 'SM3',
    [NewItem('&Элемент', 0, False, True, AClick, 0, 'MI4'),
     NewSubMenu('&Меню', 0, 'SM4',
    [NewItem('&Элемент', 0, False, True, AClick, 0, 'MI5'),
     NewSubMenu('&Меню', 0, 'SM5',
    [NewItem('&Элемент', 0, False, True, AClick, 0, 'MI6')])])])])]));
  TM := MainMenu1.Items[1];
  while True do begin
    TM.OnClick := AClick;
    if TM.Count < 2 then Break;
    TM := TM.Items[1];
  end;
end;

end.
```

[News Group]

Очень длинные меню

Данный код изменяет количество пунктов меню в зависимости от текущего разрешения. В нижеприведенном коде mnuView – выводимое меню, HandleMenuClick – назначенный обработчик для события OnClick.

```
procedure TfrmMain.LoadViewMenu;
var
  itemNum: integer;
  mnu: TMenuItem;
  menuItemHeight: integer;
  itemsPerColumn: integer;
begin
{ удаляем все видимые пункты меню }
  while mnuView.Count > 0 do begin
{ метод Free удаляет пункт меню }
    mnuView.Items[0].Free;
  end;
{ находим высоту каждого пункта меню. Значение 2 получено в результате
  экспериментов }
  menuItemHeight := GetSystemMetrics(SM_CYMENU) + 2;
{ вычисляем количество пунктов в колонке меню }
  itemsPerColumn := Screen.Height div menuItemHeight;
{ создаем пункты меню }
  for itemNum := 0 to 99 do begin
    mnu := TMenuItem.Create(Self);
    mnu.Caption := 'Пункт ' + intToStr(itemNum);
{ при необходимости начинаем с новой колонки }
    if (itemNum mod itemsPerColumn = 0) and (itemNum > 0) then begin
      mnu.Break := mbBarBreak;
    end;
{ назначаем обработчик события OnClick }
    mnu.OnClick := HandleMenuClick;
    mnuView.Add(mnu);
  end;
end;
```

Примечание

Код, приведенный выше, написан для Delphi 1, поэтому для работы со старшими версиями требует адаптации.

Слияние MDI-меню

Delphi не совсем корректно выполняет объединение меню в MDI-приложениях. Если окно MDIChild максимально развернуто и добавляется другое окно MDIChild, управляющее меню MDIChild (родительское основное меню) или исчезает совсем, или делает это в момент его выбора.

По всей видимости, для слияния меню Delphi использует функцию InsertMenu() с параметром MF_POSITION. Тем не менее, если дочернее MDI-окно макси-

мально развернуто, всплывающее (pop-up) меню, называемое еще контекстным, добавляется к меню MDI-приложения, на одну позицию дальше, чем необходимо. Это стандартное поведение системы, поскольку системное меню активного дочернего окна включается в первую позицию панели меню MDI-окна.

Согласно WinSDK, если активное дочернее окно максимально развертывается, вставляется новое всплывающее меню. При этом к значению позиции добавляется 1 (единица).

Назначение обработчика MenuItem OnClick

Имейте в виду, что SomeMethod должен быть методом объекта, а не автономной процедурой, и иметь ту же форму, что и TNotifyEvent, т. е. метод должен принимать единственный параметр (Sender) типа TObject.

```
SomeMenuItem.OnClick := SomeMethod;
```

Пиктограммы в пунктах меню

Как рисовать картинки в пунктах меню (через OwnerDraw*)?*

Решение

```
unit DN_Win;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Menus, StdCtrls,
type
  TDNForm = class(TForm)
    MainMenu1: TMainMenu;
    cm_MainExit: TMenuItem;
    procedure FormCreate(Sender: TObject);
    procedure cm_MainExitClick(Sender: TObject);
  public
    BM:TBitmap;
    procedure WMDrawItem(var Msg:TWMDrawItem); message wm_DrawItem;
    procedure WMMeasureItem(var Msg:TWMMeasureItem); message wm_MeasureItem;
  end;

var
  DNForm: TDNForm;

implementation
{$R *.DFM}

var
  Comm, yMenu: word;
```

```
procedure TDNForm.FormCreate(Sender: TObject);
begin
{ картинку в меню }
  yMenu := GetSystemMetrics(SM_CYMENU);
  comm. := cm_MainExit.Command;
  ModifyMenu(MainMenu1.Handle, 0, mf_ByPosition or mf_OwnerDraw, comm, 'Go');
end;

procedure TDNForm.cm_MainExitClick(Sender: TObject);
begin
  DNForm.Close;
end;

{ для прорисовки меню }
procedure TDNForm.WMMeasureItem(var Msg: TWMMeasureItem);
begin
  with Msg.MeasureItemStruct^ do begin
    if ItemID = comm then begin
      ItemWidth := yMenu;
      Itemheight := yMenu;
    end;
  end;
end;

procedure TDNForm.WMDrawItem(var Msg: TWMDrawItem);
var
  MemDC: hDC; BM: hBitMap; mtd: longint;
begin
  with Msg.DrawItemStruct^ do begin
    if ItemID = comm then begin
      BM := LoadBitMap(hInstance, 'dver');
{ hDC входит в структуру TDrawItemStruct }
      MemDC := CreateCompatibleDC(hDC);
      SelectObject(MemDC, BM);
{ rcItem входит в структуру TDrawItemStruct }
      if ItemState = ods_Selected then mtd := NotSrcCopy
      else mtd := SrcCopy;
      StretchBlt(hDC, rcItem.left, rcItem.top, yMenu, yMenu,
                 MemDC, 0, 0, 24, 23, mtd);
      DeleteDC(MemDC);
      DeleteObject(BM);
    end;
  end;
end;
end.
```

[Nomadic]

Примечание

В современных версиях Delphi этот код может сыграть роль «пятого колеса», но для того чтобы понять идею, может пригодиться.

Исправление пиктограмм в недоступных пунктах меню

В случае применения ListImage в недоступных пунктах меню пиктограммы отображаются некорректно. Ошибка связана с получением монохромного изображения в процедуре TCustomImageList.DoDraw. Однако ее можно исправить, если воспользоваться небольшой «заплаткой» – ScrambleBitmap.

Это можно видеть на рисунках:


```
procedure TCustomImageList.DoDraw(Index: Integer; Canvas: TCanvas; X, Y: Integer;
                                  Style: Cardinal; Enabled: Boolean);

  procedure ScrambleBitmap(var BMP: TBitmap);
  const
    RMask = $0000FF;
    RAMask = $FFFF00;
    GMask = $00FF00;
    GAMask = $FF00FF;
    BMask = $FF0000;
    BAMask = $00FFFF;
  var
    R, C: integer;
    Color: LongWord;
  begin
    with Bmp.Canvas do begin
      for C := 0 to Bmp.Height - 1 do
        for R := 0 to Bmp.Width - 1 do begin
          Color := Pixels[R, C];
          if ((Color = 0) or (Color = $FFFFFF)) then Continue;
          if (((Color and RMask > $7F) and (Color and RAMask > $0))
           or ((Color and GMask > $7F00) and (Color and GAMask > $0))
           or ((Color and BMask > $7F0000) and (Color and BAMask > $0))) then
            Pixels[R, C] := $FFFFFF
          else Pixels[R, C] := 0;
        end;
    end;
  end;
```

```
const
  ROP_DSPDxax = $00E20746;
var
  R: TRect; DestDC, SrcDC: HDC;
begin
  if HandleAllocated then begin
    if Enabled then
      ImageList_DrawEx(Handle, Index, Canvas.Handle, X, Y, 0, 0,
                       GetRGBColor(BkColor), GetRGBColor(BlendColor), Style)
    else begin
      if FMonoBitmap = nil then begin
        FMonoBitmap := TBitmap.Create;
        with FMonoBitmap do begin
       Monochrome := True;                 // закомментировать!
          Width := Self.Width; Height := Self.Height;
        end;
      end;
{ Store masked version of image temporarily in FBitmap }
{ Временно сохраняем маскированную версию изображения в FBitmap }
      FMonoBitmap.Canvas.Brush.Color := clWhite;
      FMonoBitmap.Canvas.FillRect(Rect(0, 0, Self.Width, Self.Height));
      ImageList_DrawEx(Handle, Index, FMonoBitmap.Canvas.Handle, 0, 0, 0, 0,
                       CLR_DEFAULT, 0, ILD_NORMAL);
      ScrambleBitmap(FMonoBitmap);                                  // заплатка
      R := Rect(X, Y, X + Width, Y + Height);
      SrcDC := FMonoBitmap.Canvas.Handle;
      BitBlt(SrcDC, 0, 0, Width, Height, SrcDC, 0, 0, DSTINVERT);    // добавить!
{ Convert Black to clBtnHighlight }
      Canvas.Brush.Color := clBtnHighlight;
      DestDC := Canvas.Handle;
      Windows.SetTextColor(DestDC, clWhite);
      Windows.SetBkColor(DestDC, clBlack);
      BitBlt(DestDC, X+1, Y+1, Width, Height, SrcDC, 0, 0, ROP_DSPDxax);
{ Convert Black to clBtnShadow }
      Canvas.Brush.Color := clBtnShadow;
      DestDC := Canvas.Handle;
      Windows.SetTextColor(DestDC, clWhite);
      Windows.SetBkColor(DestDC, clBlack);
      BitBlt(DestDC, X, Y, Width, Height, SrcDC, 0, 0, ROP_DSPDxax);
    end;
  end;
end;
```

«Заплатку» можно использовать и в ToolBar, где ошибка аналогична.

Сравните ToolBar без «заплатки» и с ней:

[VS]

Вызов всплывающего меню

Я создал общий обработчик события для всех своих компонентов EditBox. *В обработчике при нажатии правой кнопки мыши мне нужно вызвать всплывающее меню, создать список доступных значений, отметить текущее и вернуть обратно в* EditBox *выбранное, т. е. мне нужна двухсторонняя связь.*

Добавьте объявление переменной в секцию private формы:

```
ActivePopUp: TPopUpMenu;
```

Затем укажите в обработчике события OnPopUp на общий TPopUpMenu, как показано ниже:

```
procedure TForm1.OnPopUp(Sender: TObject);
begin
  ActivePopUp := Sender as TPopUpMenu;
end;
```

Затем из обработчика события любого элемента контекстного меню вы можете сослаться на компонент, над областью которого оно и было вызвано:

```
ActivePopUp.PopUpComponent;
```

[News Group]

Динамическое создание пункта всплывающего меню

Как динамически создать пункт всплывающего меню?

Для динамического создания пункта меню необходимо создать процедуру в объекте (частный метод формы), примерно так:

```
procedure MyClick(Sender: TObject);
```

Затем, при создании нового пункта меню, назначить ему собственное событие OnClick следующим образом:

```
NewItem := TMenuItem.Create(Self);
NewItem.Caption := 'Пункт меню';
NewItem.OnClick := MyClick;
```

Обработчик динамически созданного пункта меню

Как «подключить» код к пункту меню, который был создан динамически?

Приведенный ниже метод используется для «подключения» кода к пунктам меню, динамически создаваемым во время выполнения программы. Сохраняя результат выполнения пункта меню в глобальной переменной, можно воспользоваться им позже в нужное время в любом месте программы.

Примечание

Этот пример – один из возможных вариантов. Вообще же обработчик для пункта меню подключается следующим кодом:

```
MyItem.OnClick := MyClick;
```

где `MyItem` *– нужный пункт меню,* `MyClick` *– обработчик сообщения* `OnClick` *этого пункта. Необходимые условия для обработчика указаны в предыдущем совете, пример – в следующем. Надо заметить, что старшие версии Delphi позволяют аналогично подключать обработчики и для некоторых других событий.*

```
unit Tunit1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Menus;

type
  TForm1 = class(TForm)
    Edit1: TEdit;           { Просто "место для щелчка" и отображения результатов }
    procedure Edit1Click(Sender: TObject);
  private
    FPopupMenu: TPopupMenu;   { Общий Popup для использования "кем нужно" }
    FPopupResult: Longint;    { Результат последнего выполненного FPopupMenu }
    procedure FPopupMenuClick(Sender: TObject);
  end;

var
  Form1: TForm1;

implementation
{$R *.DFM}

procedure TForm1.FPopupMenuClick(Sender: TObject);
begin
  with (Sender as TMenuItem) do
    FPopupMenu.Tag := Tag;     { передаем значение TMenuItem.Tag в FPopupMenu.Tag }
end;

procedure TForm1.Edit1Click(Sender: TObject);
var
  tx, ty, tz: integer;
  FMenuItem: TMenuItem;
begin
  tx := Left + (Width - ClientWidth) + (Sender as TEdit).Left;
  ty := Top + (Height- ClientHeight)+ (Sender as TEdit).Top;
  FPopupMenu := TPopupMenu.Create(Self);
  FPopupMenu.AutoPopup := false;
```

```
  FPopupMenu.Tag := 0;
  for tz := 1 to 5 do begin
    FMenuItem := TMenuItem.Create(Self);
    with FMenuItem do begin
      Tag := tz;
      OnClick := FPopupMenuClick;        { все сделает один OnClick }
      Caption := 'Выбор #' + IntToStr(tz);
    end;
    FPopupMenu.Items.Add(FMenuItem)
  end;
  FPopupMenu.Popup(tx, ty);
  Application.Processmessages;         { даем время для обработки события OnClick }
  if FPopupMenu.Tag <> 0 then begin      { они действительно выбрали что-то? }
    FPopupResult := FPopupMenu.Tag;
    Edit1.Text := ' Выбор #' + IntToStr(FPopupResult);
  end;
{ FPopupMenu.Tag может храниться в ГЛОБАЛЬНОЙ переменной и использоваться позже как
  порядковое значение в блоках CASE OF или IF THEN для организации в коде условного
  перехода. }
  FPopupMenu.Free;
  FPopupMenu:= nil;
end;

end.
```

[News Group]

Динамическое создание пунктов подменю во всплывающем меню

Как динамически создавать пункты подменю всплывающего меню?

Решение

```
procedure TForm1.PopupMenu1Popup(Sender: TObject);
var
  mi, msub: TmenuItem;
begin
  with (Sender as TPopupMenu) do begin
// Удаляем все пункты меню
    while Items.Count > 0 do Items[0].Free;
// Создаем обычный пункт "Первый"
    mi := TMenuItem.Create(Self);
    with mi do begin
      Caption := 'Первый';
      OnClick := MyClick;
    end;
    Items.Insert(0, mi);
// Создаем подменю "Подменю" с двумя пунктами "Подменю1" и "Подменю2"
    mi := TMenuItem.Create(Self);
    with mi do begin
```

```
        Caption := 'Подменю';
        msub := TMenuItem.Create(Self);
        with msub do begin
          Caption := 'Подменю1';
          OnClick := MyClick;
        end;
        Insert(0, msub);
        msub := TMenuItem.Create(self);
        with msub do begin
          Caption := 'Подменю2';
          OnClick := MyClick;
        end;
        Insert(1, msub);
      end;
      Items.Insert(1, mi);
    end;
  end;

  procedure TForm1.MyClick(Sender: TObject);
  begin
    beep;
  end;
```

Использование контекстного меню с VBX

Я хочу, чтобы при щелчке правой кнопкой мыши на моем VBX появлялось контекстное меню. При вызове контекстного меню формы я не могу определить, где был произведен щелчок. Я же хочу отображать меню при щелчке правой кнопкой на VBX. Как перехватить вызов меню?

Решение

```
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
                               Shift: TShiftState; X, Y: Integer);
begin
  if button = mbRight then
    with (Sender AS TControl) do
      with ClientToScreen(Point(X, Y)) do begin
        PopupMenu1.PopupComponent := TComponent(Sender);
        PopupMenu1.Popup(X, Y);
      end;
end;
```

Примечание

Свойство формы `PopupMenu` *должно быть пустым, иначе контекстное меню будет появляться везде. Для того чтобы форма была единственным местом, где бы появлялось контекстное меню, поместите данный метод в обработчике события формы* `OnMouseDown`. *Если же требуется, чтобы единственно возможным местом появления контекснного меню был VBX, то надо поместить приведенный выше метод в обработчик события* `OnMouseDown` *самого VBX, ну и так далее по аналогии.*

Вызов контекстного меню в позиции курсора

Как правильно вызвать контекстное меню в координатах курсора мыши?

Решение 1

```
procedure TForm1.Form1MouseDown(Sender: TObject; Button: TMouseButton;
                                Shift: TShiftState; X, Y: Integer);
begin
  if Button = mbRight then
    with (Sender as TControl).ClientToScreen(Point(X, Y)) do
      PopupMenu1.Popup(X, Y);
end;
```

Решение 2

Вызов контекстного меню связан с координатами экрана. Координаты, получаемые в обработчике события, вероятно, относятся к объекту, который создал это сообщение. Для преобразования координат необходимо воспользоваться функцией `ClientToScreen`.

Вот пример вызова контекстного меню, осуществляемого при щелчке правой кнопкой мыши по узлу `TTreeView`. Этот пример не в точности отвечает на поставленный вопрос, но предложенную идею можно развить.

```
procedure TForm1.TreeViewMouseDown(Sender: TObject; Button: TMouseButton;
                                   Shift: TShiftState; X, Y: Integer);
var
  P: TPoint;
begin
  if Button <> mbRight then exit;
  TreeMenu.AutoPopup := False;
  if TreeView.GetNodeAt(X, Y) <> NIL then begin
    TreeView.Selected := TreeView.GetNodeAt(X, Y);
    P.X := X;
    P.Y := Y;
    P := TreeView.ClientToScreen(P);
    TreeMenu.Popup(P.X, P.Y);
  end;
end;
```

Событие OnKeyPress и курсорные клавиши в TMemo

Как обновлять текущую строку во время перемещения по ней с помощью курсорных клавиш?

Потребуется перехватывать ряд событий. В приведенном ниже коде созданы обработчики всех возможных для данной операции событий, выберите необходимые сами. Некоторые события могут иметь общий обработчик.

Данный пример отображает в заголовке текущие координаты курсора (столбец, строка).

```
unit Unit1;

interface

uses
  Windows, SysUtils, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm1 = class(TForm)
    Memo1: TMemo;
    procedure Memo1Change(Sender: TObject);
    procedure Memo1Click(Sender: TObject);
    procedure Memo1Enter(Sender: TObject);
    procedure Memo1KeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
    procedure Memo1KeyUp(Sender: TObject; var Key: Word; Shift: TShiftState);
    procedure Memo1MouseDown(Sender: TObject; Button: TMouseButton;
                             Shift: TShiftState; X, Y: Integer);
    procedure Memo1MouseUp(Sender: TObject; Button: TMouseButton;
                           Shift: TShiftState; X, Y: Integer);
  private
    function GetLineIndex: Word;
    function GetStrInsertIndex: Word;
    procedure GetCursorCoord;
  end;

var
  Form1: TForm1;

implementation
{$R *.DFM}

function TForm1.GetLineIndex: Word;
begin
  Result := SendMessage(Memo1.Handle, EM_LINEFROMCHAR, Memo1.SelStart, 0);
end;

function TForm1.GetStrInsertIndex: word;
begin
  Result := Memo1.SelStart - SendMessage(Memo1.Handle, EM_LINEINDEX,
                                         GetLineIndex, 0);
end;

procedure TForm1.GetCursorCoord;
var
  LineIndex: word;
  LineChar: byte;
  SelSt: word;
begin
  SelSt := Memo1.SelStart;
  LineIndex := GetLineIndex;
```

```
  LineChar := GetStrInsertIndex;
  if Memo1.SelText > '' then Caption := 'Выбранный текст'
  else Caption := 'Колонка ' + IntToStr(LineChar + 1) + ' , ' + 'Строка '
                + IntToStr(LineIndex + 1);
end;

procedure TForm1.Memo1Change(Sender: TObject);
begin
  GetCursorCoord;
end;

procedure TForm1.Memo1Click(Sender: TObject);
begin
  GetCursorCoord;
end;

procedure TForm1.Memo1Enter(Sender: TObject);
begin
  GetCursorCoord;
end;

procedure TForm1.Memo1KeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  GetCursorCoord;
end;

procedure TForm1.Memo1KeyUp(Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  GetCursorCoord;
end;

procedure TForm1.Memo1MouseDown(Sender: TObject; Button: TMouseButton;
                                Shift: TShiftState; X, Y: Integer);
begin
  GetCursorCoord;
end;

procedure TForm1.Memo1MouseUp(Sender: TObject; Button: TMouseButton;
                              Shift: TShiftState; X, Y: Integer);
begin
  GetCursorCoord;
end;

end.
```

Поиск и замена текста в TMemo

Решение

```
procedure TForm1.FindDialog1Find(Sender: TObject);
var
```

```
    Buff, P, FT: PChar;
    BuffLen: Word;
  begin
    with Sender as TFindDialog do begin
      GetMem(FT, Length(FindText) + 1);
      StrPCopy(FT, FindText);
      BuffLen := Memo1.GetTextLen + 1;
      GetMem(Buff, BuffLen);
      Memo1.GetTextBuf(Buff, BuffLen);
      P := Buff + Memo1.SelStart + Memo1.SelLength;
      P := StrPos(P, FT);
      if P = NIL then MessageBeep(0)
      else begin
        Memo1.SelStart := P - Buff;
        Memo1.SelLength := Length(FindText);
      end;
      FreeMem(FT, Length(FindText) + 1);
      FreeMem(Buff, BuffLen);
    end;
  end;

  procedure TForm1.ReplaceDialog1Replace(Sender: TObject);
  begin
    with Sender as TReplaceDialog do
      while True do begin
        if Memo1.SelText <> FindText then FindDialog1Find(Sender);
        if Memo1.SelLength = 0 then Break;
        Memo1.SelText := ReplaceText;
        if not (frReplaceAll in Options) then Break;
      end;
  end;
```

Текущая позиция курсора в TMemo

Как определить текущую позицию курсора в TMemo*?*

Решение 1

```
CurrentLine := SendMessage(Memo1.Handle, EM_LINEFROMCHAR, Memo1.SelStart, 0);
```

Это вызов возвращает номер строки, содержащей курсор. Следующий код вернет позицию символа текущей строки, около которого находится курсор:

```
ColNum:= Memo1.SelStart-SendMessage(Memo1.Handle, EM_LINEINDEX, CurrentLine, 0)+1;
```

Описания EM_LINEFROMCHAR и EM_LINEINDEX можно найти в файлах помощи по Windows API.

Решение 2

```
procedure TForm1.Button1Click(Sender: TObject);
var
  iLine: Integer;
```

```
begin
  iLine := Memo1.Perform(em_LineFromChar, $FFFF, 0);
{ Примечание: первая строка нулевая }
  MessageDlg('Номер строки: ' + IntToStr(iLine), mtInformation, [mbOK], 0);
end;
```

Примечание

Аналогично, номер символа в строке (`iCharNum`*):*

```
iCharNum := Memo1.Perform(EM_LINEINDEX, iLine, 0) + 1;
```

TMemo и StringList

Если требуется заменить все содержимое компонента `TMemo`:

```
Memo1.Lines := StringList1.Strings;
```

А вот так можно добавить строки к списку компонента:

```
Memo1.Lines.AddStrings(StringList1.Strings);
```

Использование встроенного отката в TMemo

Реализовать интегрированный откат легче, чем осуществить комбинацию <Ctrl>+<Z>:

```
Memo1.Perform(EM_UNDO, 0, 0);
```

Предварительно можно осведомиться о доступности отмены (т. е. о наличии предыдущих состояний) для включения/выключения соответствующего пункта меню:

```
Undo1.Enabled := Memo1.Perform(EM_CANUNDO, 0, 0) <> 0;
```

[News Group]

TMemo со свойствами Строка/Колонка

Решение

```
unit C_rcmemo;

interface

uses
  Windows, SysUtils, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TRCMemo = class(TMemo)
```

```
  private
    function GetRow: Integer;
    procedure SetRow(value: Integer);
    function GetCol: Integer;
    procedure SetCol(value: Integer);
    function GetPosn: LongInt;
    procedure SetPosn(value: LongInt);
  published
    property Row: Integer Read GetRow Write SetRow;
    property Col: Integer Read GetCol Write SetCol;
    property Posn: LongInt Read GetPosn Write SetPosn;
  end;

procedure Register;

implementation

function TRCMemo.GetRow: Integer;
begin
  Result := Perform(EM_LINEFROMCHAR, $FFFF, 0);
end;

procedure TRCMemo.SetRow(value: Integer);
begin
  SelStart := GetCol + Perform(EM_LINEINDEX, Value, 0);
end;

function TRCMemo.GetCol: Integer;
begin
  Result := SelStart - Perform(EM_LINEINDEX, GetRow, 0);
end;

procedure TRCMemo.SetCol(value: Integer);
begin
  SelStart := Perform(EM_LINEINDEX, GetRow, 0) + Value;
end;

function TRCMemo.GetPosn: LongInt;
var
  ro, co: Integer;
begin
  ro := GetRow;
  co := SelStart - Perform(EM_LINEINDEX, ro, 0);
  Result := MakeLong(co,ro);
end;

procedure TRCMemo.SetPosn(value: LongInt);
begin
  SelStart := Perform(EM_LINEINDEX, HiWord(Value), 0) + LoWord(Value);
end;
```

```
procedure Register;
begin
  RegisterComponents('NJR', [TRCMemo]);
end;
end.
```

Ограничение длины и количества строк в TMemo

Как ограничить длину и количество строк в `TMemo`*?*

Решение

```
unit Unit1;
interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, ComCtrls, StdCtrls;

type
  TForm1 = class(TForm)
    Memo1: TMemo;
    procedure FormCreate(Sender: TObject);
    procedure Memo1KeyPress(Sender: TObject; var Key: Char);
  public
    MaxCharsPerLine, MaxLines: Integer;
    function MemoLine: Integer;
    function LineLen(r: Integer): Integer;
    function NRows: Integer;
  end;

var
  Form1: TForm1;

implementation
{$R *.DFM}

function TForm1.NRows: Integer;
begin
  with Memo1 do
    Result := 1 + SendMessage(Handle, EM_LINEFROMCHAR, GetTextLen - 1, 0);
end;

function TForm1.LineLen(r: Integer): Integer;
var
  r1, r2: Integer;

begin
  with Memo1 do begin
    r1 := SendMessage(Handle, EM_LINEINDEX, r, 0);
    if (r > NRows-1) then
```

```
      r2 := SendMessage(Handle, EM_LINEINDEX, r + 1, 0) - 2 {-CR/LF}
    else
      r2 := GetTextLen;
  end;
  Result := r2 - r1;
end;

function TForm1.MemoLine: Integer;
begin
  with Memo1 do Result := SendMessage(Handle, EM_LINEFROMCHAR, SelStart, 0);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  MaxCharsPerLine := 8;
  MaxLines := 4;
end;

procedure TForm1.Memo1KeyPress(Sender: TObject; var Key: Char);
begin
  with Memo1 do
    case Key of
        ' '..#255: if (LineLen(MemoLine) >= MaxCharsPerLine) then Key := #0;
         #10, #13: if (NRows >= MaxLines) then Key := #0;
               #8: if (SelStart = SendMessage(Handle, EM_LINEINDEX, MemoLine, 0))
                   then Key := #0;
    end;
end;

end.
```

Использование шрифтов и стилей в TMemo

Кто-нибудь знает, как использовать различные шрифты и стили в TMemo*?*

Создайте собственный TxxxMemo. Наследуйте от стандартного TMemo и перекройте метод Paint.

Пример, изменяющий цвет каждой строки:

```
unit Todrmemo;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls;

type
  TOwnerDrawMemo = class(TMemo)
  private
```

```
    procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
end;

procedure Register;

implementation

procedure TOwnerDrawMemo.WMPaint(var Message: TWMPaint);
var
  Buffer: Array[0..255] of Char;
  PS: TPaintStruct;
  DC: HDC;
  i: Integer;
  X, Y, Z: Word;
  OldColor: LongInt;
begin
  DC := Message.DC;
  if DC = 0 then DC := BeginPaint(Handle, PS);
  try
    X := 1;
    Y := 1;
    SetBkColor(DC, Color);
    SetBkMode(DC, Transparent);
    OldColor := Font.Color;
    for i:=0 to Pred(Lines.Count) do begin
      if odd(i) then SetTextColor(DC, clRed)
      else SetTextColor(DC, OldColor);
      Z := Length(Lines[i]);
      StrPCopy(Buffer, Lines[i]);
      Buffer[Z] := #0;                    { реально не нужно }
      TextOut(DC, X,Y, Buffer, Z);
      Inc(Y, abs(Font.Height));
    end;
  finally
    if Message.DC = 0 then EndPaint(Handle, PS);
  end;
end;

procedure Register;
begin
  RegisterComponents('Controls', [TOwnerDrawMemo]);
end;

end.
```

[News Group]

Примечание

Пример написан для Delphi 1, поэтому желающие могут, разобравшись в написанном, адаптировать его к более современным версиям. А я рекомендую использовать `RichEdit`.

Добавление строк в TMemo

Быстрый способ добавить в TMemo дополнительный текст заключается в его загрузке в невидимый TMemo и выполнении следующего кода:

```
Memo1.Lines.AddStrings(Memo2.Lines);
```

Вам необходима функциональность второго TMemo, а не просто TStringList, поскольку первый инкапсулирует множество функций для работы с текстом. В противном случае можно было бы загрузить файл в TStringList и искать первые 255 символов каждого абзаца самостоятельно.

Чтобы было по-настоящему все удобно, необходимо создать временный TMemoStrings. К сожалению, TMemoStrings определен в секции implementation файла StdCtrls.pas и, таким образом, недоступен.

Бывает и так, что два временных TStringLists удобнее одного временного TMemo. Например, такой случай:

```
TS1 := TStringList.Create;
TS2 := TStringList.Create;
TS1.Assign(Memo1.Lines);
Memo1.Lines.LoadFromFile('BULLRUN.TXT');
TS2.Assign(Memo1.Lines);
Memo1.Lines.Assign(TS1);
Memo1.Lines.AddStrings(TS2);
TS2.Free;
TS1.Free;
```

Гвоздь программы – возможность разрывать строки в свойстве Memo Lines при добавлении нового текста. Итак, мы записываем существующий текст во временную переменную, считываем новый текст в TMemo. Снова передаем новый текст временной переменной, восстанавливаем оригинальный текст и, наконец, добавляем новый текст.

[News Group]

Вставка текста в TMemo

Как вставить какой-либо текст в TMemo *в позицию курсора во время выполнения программы?*

Можно воспользоваться сообщениями Windows:

```
SendMessage(Memo.Handle, EM_REPLACESEL, 0, PChar('Текст'));
```

[News Group]

Импортирование файла в TMemo

Как импортировать файл в TMemo*, начиная с позиции курсора? Метод* LoadFromFile *заменяет содержимое* TMemo *содержимым текстового файла. Я хочу включить текстовый файл или в позицию курсора или, если выбран текст, заменить этот текст содержимым текстового файла. Все это должно быть похоже на работу функции* PasteFromClipboard.

Самый простой путь вставки текста в компонент TMemo заключается в посылке ему сообщения EM_REPLACESEL:

```
{ Если вы хотите заменить выбранный в Memo текст, передайте
  в параметре ReplaceSel значение True. False необходим для простой вставки текста }
procedure InsertFileInMemo(Memo: TMemo; FileName: string; ReplaceSel: Boolean);
var
  Stream: TMemoryStream;
  NullTerminator: Char;
begin
  Stream := TMemoryStream.Create;
  try
{ Загружаем текст... }
    Stream.LoadFromFile(FileName);
{ Добавляем в конец текста терминирующий ноль... }
    Stream.Seek(0, 2);
    NullTerminator := #0;
    Stream.Write(NullTerminator, 1);
{ Вставляем текст в Memo... }
    if not ReplaceSel then Memo.SelLength := 0;
    SendMessage(Memo.Handle, EM_ReplaceSel, 0, LongInt(Stream.Memory));
  finally
    Stream.Free;
  end;
end;
```

Создание страниц TNoteBook во время работы приложения

Как создать страницу в компоненте TNoteBook *из приложения?*

Решение

```
procedure TForm1.Button1Click(Sender: TObject);
var
  NewPage: TWinControl;
begin
  TabbedNotebook1.Pages.Add(Edit1.Text);
  NewPage := TabbedNotebook1.Pages.Objects[TabbedNotebook1.Pages.Count - 1]
             as TWinControl;
  with TLabel.Create(Self) do begin
    Left := 20;
    Top := 20;
```

```
    Caption := '&Телефон: ';
    Parent := NewPage;
  end;

  with TEdit.Create(Self) do begin
    Left := 100;
    Top := 20;
    Text := '1-800-555-1212';
    Parent := NewPage;
  end;
end;
```

[News Group]

Проблема с освобождением ресурсов TNoteBook

Как правильно освободить ресурсы в компоненте TNoteBook?

Освободите дескрипторы окон невидимых страниц компонента. Следующий код поясняет, как это можно сделать – обработчик события компонента OnClick освобождает дескрипторы невидимых страниц при каждом изменении видимой страницы:

```
procedure TForm1.TabSet1Click(Sender: TObject);
var
  i: integer;
begin
  Notebook1.PageIndex := TabSet1.TabIndex;
  with Notebook1, Pages do
    for i := 0 to Count - 1 do
      if (i <> PageIndex) then TForm1(Objects[i]).DestroyHandle;
end;
```

Как можно заметить, тип страницы NoteBook (Objects[i]) приведен к объектам TForm1, что на самом деле не так. Тем не менее, эта небольшая хитрость позволяет иметь доступ к защищенным членам извне метода класса, где они определены, в данном случае DestroyHandle – защищенный метод TWinControl.

Это работает, поскольку TForm1 является наследником TWinControl и позволяет иметь доступ к защищенным членам TForm1 и ее наследникам.

TNoteBook как контейнер для форм

Как разместить подклассы форм на страницах компонента TTabbedNotebook*?*

Попробуйте следующее решение. Оно работает с компонентами, являющимися частью формы, содержащей TTabbedNotebook, но не работает с дочерними формами:

```
ChildForm1[i].Parent := TWinControl(BrowseTabNotebook.Pages.Objects[i]);
```

В дочерней форме должен быть следующий код:

```
  ...
  private
    procedure CreateParams(var Params: TCreateParams); override;
  ...
procedure TChildForm1.CreateParams(var Params: TCreateParams);
begin
  inherited CreateParams(Params);       { сначала вызываем унаследованные методы. }
  with Params do begin
    WndParent := Application.Mainform.Handle;
    Style := (Style or WS_CHILD) and not WS_POPUP;
  end;
end;
```

[News Group]

Добавление и удаление страниц в TNoteBook

Как во время выполнения программы динамически добавлять и удалять страницы динамически созданного TNoteBook*?*

Добавляем новую страницу к TNoteBook и новую вкладку к TabSet (параметр PageName задает имя страницы), размещаем на странице компонент TMemo и выводим новую страницу на передний план. Подразумевается, что TabSet (набор закладок) имеет ровно по одной закладке на каждую страницу TNoteBook с точным сохранением порядка.

```
procedure AddPage(nbk: TNotebook; tabset: TTabSet; const pagename: string);
var
  memo: TMemo;
  page: TPage;
begin
  if nbk <> nil then begin
    nbk.Pages.Add(pagename);                { добавляем страницу в TNotebook }
    nbk.PageIndex := nbk.Pages.Count - 1;   { делаем новую страницу текущей }
    if tabset <> nil then begin
      tabset.Tabs.Add(pagename);            { добавляем соответствующую закладку }
      tabset.TabIndex := nbk.PageIndex;     { делаем новую закладку текущей }
    end;
    if nbk.PageIndex > -1 then begin        { убедимся что страница существует }
      page := TPage(nbk.Pages.Objects[nbk.PageIndex]); { получаем объект страницы }
      memo := TMemo.Create(page);           { создаем TMemo (и страницу в
                                              качестве родителя) }
      try
        memo.Parent := page;          { устанавливаем страницу в качестве Parent }
{ устанавливаем выравнивание для заполнения области клиента }
        memo.Align := alClient;
      except
        memo.Free;             { освобождаем TMemo, если что-то идет неправильно }
      end;
```

```
      page.Visible := True;        { показываем страницу }
    end;
  end;
end;

procedure DeletePage(nbk: TNotebook; tabset: TTabSet; index: integer);
{ Удаляем страницу, чей PageIndex = index из nbk и tabset. Подразумевается, что
  TabSet имеет ровно по одной закладке на каждую страницу NoteBook с точным
  сохранением порядка. }
var
  switchto: integer;
begin
  if nbk <> nil then begin
    if (index >= 0) and (index < nbk.Pages.Count) then begin
      if index = nbk.PageIndex then begin
        { если страница не последняя в списке}
        if index < nbk.Pages.Count - 1 then begin
          { выводим страницу за текущей, ставшей ею после удаления }
          switchto := nbk.PageIndex;
          if (index = 0) and (nbk.Pages.Count > 1) then { если первая страница }
            nbk.PageIndex := 1;            { теперь показываем вторую страницу }
          end else
            { в противном случае показываем страницу, расположенную перед текущей }
            switchto := nbk.PageIndex - 1;
        end;
        { освобождаем страницу и все принадлежавшие ей элементы управления }
        nbk.Pages.Delete(index);
       if tabset <> nil then begin
         if index < tabset.Tabs.Count then
           tabset.Tabs.Delete(index);          { удаляем соответствующую закладку }
       end;
       nbk.PageIndex := switchto;
    end;
  end;
end;
```

[News Group]

TPaintBox в буфер обмена

Как копировать изображение из TPaintBox *в буфер обмена?*

Решение

```
procedure TForm1.Button1Click(Sender: TObject);
var
  pbRect: TRect;
  BitMap: TBitMap;
begin
  pbRect := Rect(0, 0, PaintBox1.Width, PaintBox1.Height);
  BitMap := TBitMap.Create;
```

```
  try
    BitMap.Width := PaintBox1.Width;
    BitMap.Height := PaintBox1.Height;
    BitMap.Canvas.CopyRect(pbRect, PaintBox1.Canvas, pbRect);
    ClipBoard.Assign(BitMap);
  finally
    BitMap.Free;
  end;
end;
```

[News Group]

Отрисовка TOutline

Мне понадобилось изменить поведение компонента при его отрисовке, например, предоставить пользователю возможность выделить несколько значений и «покрасить» их в другой цвет. Я с удивлением обнаружил, что использование функции `GetItem` *для определения отрисовываемой строки на редкость просто и понятно. Прежде же для этой цели я создавал цикл, отбирал из всех индексов нужный и затем передавал его процедуре. Но в случае с несколькими значениями это становилось медленным, неуклюжим и запутанным.*

Пример такой программы. Она позволяет рисовать символы плюс/минус и сам текст. В нее включен код для создания, загрузки и освобождения пиктограмм с символами плюса и минуса.

```
var
  BitmapPlus, BitmapMinus: TBitMap;

procedure TForm1.FormCreate(Sender: TObject);
begin
  BitmapPlus := TBitmap.Create;
  BitmapPlus.LoadFromFile('c:\delphi\images\default\outplus.bmp');
  BitmapMinus := TBitmap.Create;
  BitmapMinus.LoadFromFile('c:\delphi\images\default\outminus.bmp');
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  BitMapPlus.Destroy;
  BitMapMinus.Destroy;
end;

procedure TForm1.Outline1DrawItem(Control: TWinControl; Index: Integer;
                                  Rect: TRect; State: TOwnerDrawState);
var
  iLeft, iht: integer;
  s: string;
  NodeInView: TOutlineNode;
  col: TColor;
```

```
  IndexInFull: integer;
begin
  with (Control as TOutline).Canvas do begin
{ рисуем на холсте элемента управления, не на форме }
    IndexInFull := outline1.getitem(rect.left + 1, rect.top + 1);
    NodeInView := Outline1.Items[IndexInFull];
    Font := Outline1.Font;
    iLeft := Rect.Left;
    iht := Outline1.ItemHeight;
    inc(iLeft, iht * (NodeInView.level - 1));
    s := NodeInView.Text;
    Col := clWindow;
    Font.Color := clWindowText;
    Brush.Color := col;
    FillRect(Rect);               { очищаем прямоугольник }
    if NodeInView.HasItems and not NodeInView.Expanded then
{ Draw(iLeft, Rect.Top, BitmapPlus) }
      BrushCopy(Bounds(iLeft + 1, Rect.Top + 1, iht - 2, iht - 2), BitMapPlus,
                Bounds(0, 0, BitMapPlus.Width, BitMapPlus.Height),
                BitMapPlus.TransparentColor);
    if NodeInView.Expanded and NodeInView.HasItems then
      BrushCopy(Bounds(iLeft + 1, Rect.Top + 1, iht - 2, iht - 2), BitMapMinus,
                Bounds(0, 0, BitMapMinus.Width, BitMapMinus.Height),
                BitMapMinus.TransparentColor);
    if Copy(NodeInView.Text, Length(NodeinView.Text), 1) = 'y' then Col := clAqua;
    if Outline1.SelectedItem = IndexInFull then begin
      Col := clHighlight;
      Font.Color := clHighlightText
    end;
    Brush.Color := col;
    inc(ILeft, iht);
    TextOut(iLeft, Rect.Top, s);
  end;
end;
```

Примечание

Надо всего лишь указать расположение пиктограмм с «плюсом» и «минусом» в методе `FormCreate`. *И все-таки, я рекомендую использовать современный компонент* `TreeView`.

Поточность TOutline

Я обратил внимание на то, что `TOutline` *является потомком* `TPersistent`. *Поэтому теоретически возможно сохранение в потоке дерева объектов и его последующая загрузка. И все равно, я не пойму, как в этом случае работать с двоичным файлом?*

Ниже приведен простейший путь решения вашего вопроса. Имейте в виду, что в коде не предусмотрена проверка на ошибки чтения и принадлежность компонента типу `TOutline`.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  S: TFileStream;
begin
  S := TFileStream.Create('C:\myoutlin.dat', fmCreate);
  try
    S.WriteComponent(Outline1);
  finally
    S.Free;
  end;
end;

procedure TForm1.Button2Click(Sender: TObject);
var
  S: TFileStream;
begin
  S := TFileStream.Create('C:\myoutlin.dat', fmOpenRead);
  try
    S.ReadComponent(Outline1);
  finally
    S.Free;
  end;
end;
```

[News Group]

Раскрытие пути к элементу TOutline по его индексу

Сначала присвоим элементу желаемый путь:

```
"My Computer\Hardware\SoundCard\Base Address"
```

На первом шаге возвращается приведенный выше путь. Затем изолируется подстрока "My Computer". Затем с помощью метода GetTextItem определяется индекс TOutlineNode "My Computer". Метод Expand раскрывает это дерево. Впоследствии "My Computer" вырезается из пути, и новым путем становится Hardware\SoundCard\Base Adress.

Затем определяется индекс "Hardware", раскрывается и снова вырезается. Данная процедура повторяется до тех пор, пока не останется пути, который можно раскрыть. После чего полностью раскрывается путь, передаваемый компоненту TOutlineNode.

Следующая процедура в качестве параметра принимает индекс, после чего раскрывает путь к элементу с этим индексом.

Процедура подразумевает работу с объектом TOutline, с именем Outline.

```
procedure TForm1.ExpandPathToFoundItem(const FoundItemIndex: Longint);
{ Открываем путь к данному элементу (элемент определяется номером индекса).
До корневого элемента необходимо раскрывать только родителей. }
var
  ItemIndex: Longint;
```

```
  Found: Boolean;
  LastCh: Longint;
  Path: String;
  ItemText: String;
  SepPos: Integer;
  OldSep: String;
begin
{ Сохраняем старый ItemSeparator }
  OldSep := Outline.ItemSeparator;
{ Устанавливаем новый ItemSeparator }
  Outline.ItemSeparator := '\';
{ Получаем полный путь к TOutlineNode и добавляем '\'. Это делается для упрощения
  последующего алгоритма }
  Path := Outline.Items[FoundItemIndex].FullPath + '\';
{ Зацикливаемся до тех пор, пока не будет достигнут конец пути }
  while Length(Path) > 0 do begin
{ Определяем в пути позицию первого '\' }
    SepPos := Pos('\', Path);
{ Изолируем элемент TOutlineNode }
    ItemText := Copy(Path, 1, SepPos - 1);
{ Определяем индекс TOutlineNode }
    ItemIndex := Outline.GetTextItem(ItemText);
{ Раскрываем его }
    Outline.Items[ItemIndex].Expand;
{ Вырезаем из строки раскрытый TOutlineNode }
    Path := Copy(Path, SepPos + 1, Length(Path) - SepPos + 1);
  end;
{ Восстанавливаем оригинальный ItemSeparator }
  Outline.ItemSeparator := OldSep;
end;
```

Перемещение панели мышью на форме во время выполнения программы

Решение

```
procedure TForm1.Panel1MouseDown(Sender: TObject; Button: TMouseButton;
                                 Shift: TShiftState; X, Y: Integer);
const
  SC_DRAGMOVE = $F012;
begin
  ReleaseCapture;
  Panel1.Perform(WM_SYSCOMMAND, SC_DRAGMOVE, 0);
end;
```

Примечание

На самом деле таким образом можно перемещать по форме любые визуальные компоненты.

[Aleksey]

Панель с изменяющимися размерами

Код для создания панелей с изменяющимися размерами. Выровняйте панель по области клиента свойством alClient, расположите на ней несколько элементов управления и посмотрите, как меняются размеры во время изменения размера формы при выполнении программы. Можно расширить код и запретить изменение размеров во время проектирования.

```
unit Elastic;

interface

uses
  Windows, SysUtils, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls;

type
  TElasticPanel = class(TPanel)
  private
    FHorz: boolean;
    FVert: boolean;
    nOldWidth: integer;
    nOldHeight: integer;
    bResized: boolean;
  protected
    procedure WMSize(var message: TWMSize); message WM_SIZE;
  public
    nCount: integer;
    constructor Create(AOwner: TComponent); override;
  published
    property ElasticHorizontal: boolean read FHorz write FHorz default True;
    property ElasticVertical: boolean read FVert write FVert default True;
  end;

procedure Register;

implementation

constructor TElasticPanel.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  FHorz := True;
  FVert := True;
  nOldWidth := Width;
  nOldHeight := Height;
  bResized := False;
end;

procedure TElasticPanel.WMSize(var message: TWMSize);
var
  bResize: boolean;
```

```
    xRatio: real;
    i: integer;
    ctl: TWinControl;
  begin
    Inc(nCount);
    if Align = alNone then bResize := True
    else bResize := bResized;
    if not (csDesigning in ComponentState) and bResize then begin
      if FHorz then begin
        xRatio := Width / nOldWidth;
        for i := 0 to ControlCount - 1 do begin
          ctl := TWinControl(Controls[i]);
          ctl.Left := Round(ctl.Left * xRatio);
          ctl.Width := Round(ctl.Width * xRatio);
        end;
      end;
      if FVert then begin
        xRatio := Height / nOldHeight;
        for i := 0 to ControlCount - 1 do begin
          ctl := TWinControl(Controls[i]);
          ctl.Top := Round(ctl.Top * xRatio);
          ctl.Height := Round(ctl.Height * xRatio);
        end;
      end;
    end else begin
      nOldWidth := Width;
      nOldHeight := Height;
    end;
    bResized := True;
    nOldWidth := Width;
    nOldHeight := Height;
  end;

  procedure Register;
  begin
    RegisterComponents('Additional', [TElasticPanel]);
  end;

  end.
```

Компонент с вложенной панелью

Небольшой компонент представляет собой панель, содержащую другую, вложенную панель. Во вложенной панели могут быть размещены другие компоненты, читаться они будут правильно. Ключевым моментом здесь является перекрытие метода ReadState.

```
  unit RzPnlPnl;

  interface
```

```
uses
  Windows, SysUtils, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls;

type
  TSubPanel = class(TPanel)
  protected
    procedure ReadState(Reader: TReader); override;
  end;

  TPanelPanel = class(TPanel)
  private
    FSubPanel: TSubPanel;
  protected
    procedure WriteComponents(Writer: TWriter);
    procedure ReadState(Reader: TReader); override;
  public
    constructor Create(AOwner: TComponent); override;
  end;

procedure Register;

implementation

procedure TSubPanel.ReadState(Reader: TReader);
var
  OldOwner: TComponent;
begin
{ Сохраняем старого владельца, что необходимо для PanelPanel }
  OldOwner := Reader.Owner;
  Reader.Owner := Reader.Root;            { Задаем в качестве владельца форму }
  try
    inherited ReadState(Reader);
  finally
    Reader.Owner := OldOwner;
  end;
end;

constructor TPanelPanel.Create(AOwner: TComponent);
const
  Registered: Boolean = False;
begin
  inherited Create(AOwner);
  FSubPanel := TSubPanel.Create(Self);
  FSubPanel.Parent := Self;
  FSubPanel.SetBounds(20, 20, 200, 100);
  FSubPanel.Name := 'SomeName';
  if not Registered then begin

{ так TSubPanel может храниться в файле формы }
    Classes.RegisterClasses([TSubPanel]);
```

```
    Registered := True;
  end;
end;

procedure TPanelPanel.ReadState(Reader: TReader);
var
  OldOwner: TComponent;
  I: Integer;
begin
  for I := 0 to ControlCount - 1 do
    Controls[0].Free;
  OldOwner := Reader.Owner;

{ Для чтения субкомпонентов установите данный экземпляр в качестве родителя }
  Reader.Owner := Self;
  try
    inherited ReadState(Reader);
  finally
    Reader.Owner := OldOwner;
  end;
end;

procedure TPanelPanel.WriteComponents(Writer: TWriter);
var
  I: Integer;
begin
  for I := 0 to ControlCount - 1 do Writer.WriteComponent(Controls[I]);
end;

procedure Register;
begin
  RegisterComponents('Samples', [TPanelPanel]);
end;

end.
```

[News Group]

Индикатор хода выполнения в строке состояния

Решение

```
procedure TForm1.Button1Click(Sender: TObject);
var
  pb: TProgressBar;
begin
  ...
  pb := TProgressBar.Create(Self);
  with pb do begin
    Parent := StatusBar1;
    Position := 30;
```

```
      Top := 2;
      Left := 0;
      Height := StatusBar1.Height - Top;
      Width := StatusBar1.Panels[0].Width - Left;
    end;
    pb.Visible := True;
    ...
end;
```

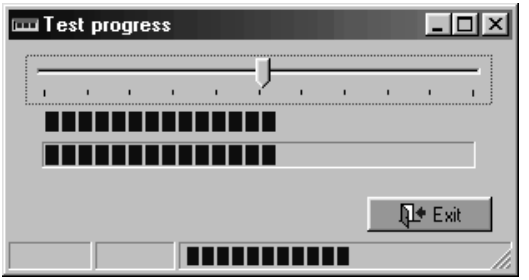
[Лихолетов Алексей]

Примечание

*Для работы необходимо создать хотя бы одну панель (*StatusPanel*) в* StatusBar.

ProgressBar с невидимой рамкой

ProgressBar всегда вычерчивает рамку и не имеет методов управления ею. Предлагаемое решение позволяет использовать ProgressBar в StatusBar без показа рамки.




```
unit vsprogress;
interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ComCtrls;
type
  TNProgressBar = class(TProgressBar)

    procedure WMNCPAINT(var Msg: TMessage); message WM_NCPAINT;
  private
    FShowFrame: boolean;
    procedure SetShowFrame(Value: boolean);
  public
    constructor Create(AOwner: TComponent); override;
  published
    property ShowFrame: boolean read FShowFrame write SetShowFrame;
  end;

procedure Register;

implementation
```

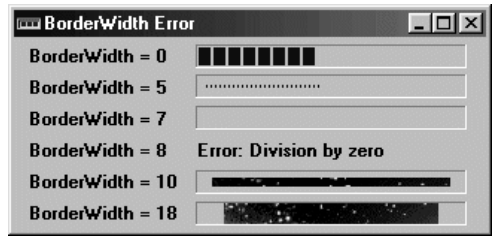
```
{ TNProgressBar }

constructor TNProgressBar.Create(AOwner: TComponent);
begin
  inherited;
  FShowFrame := True;
end;

procedure TNProgressBar.SetShowFrame(Value: boolean);
begin
  if FShowFrame <> Value then begin
    FShowFrame := Value;
    RecreateWnd;
  end;
end;

procedure TNProgressBar.WMNCPAINT(var Msg: TMessage);
var
  DC: HDC;
  RC: TRect;
begin
  if ShowFrame then begin
    inherited;
    Invalidate;
  end else begin
    DC := GetWindowDC(Handle);
    try
      Windows.GetClientRect(Handle, RC);
      with RC do begin
        Inc(Right, 2);
        Inc(Bottom, 2);
      end;
      Windows.FillRect(DC, RC, Brush.Handle);

    finally
      ReleaseDC(Handle, DC);
    end;
  end;
end;

procedure Register;
begin
  RegisterComponents('Controls', [TNProgressBar]);
end;

end.
```

[VS]

Некорректность реализации свойства BorderWidth

Применяя в компонентах свойство BorderWidth, будьте внимательны. В большинстве компонентов (TControlBar, TProgressBar, TStatusBar, TToolBar, TTrackBar и т. д.), это свойство реализовано некорректно.

Можно получить забавные результаты или большие неприятности. На рисунке показаны варианты, получающиеся при использовании различных значений BorderWidth в компоненте ProgressBar. Высота компонента ProgressBar.Height = 16.



В последних двух вариантах вместо индикатора – изображение под активным окном. Не думайте, что компонент стал «прозрачным». Это «моментальный снимок» при создании окна.

Если есть желание, то некорректность можно исправить в ComCtrls.pas, переопределив BorderWidth.

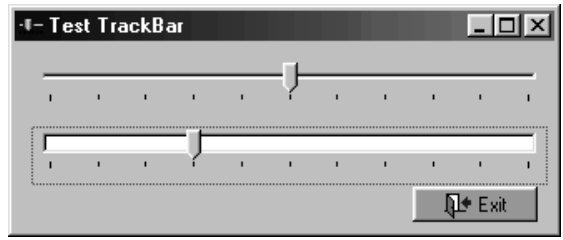
```
  ...
  TProgressBar = class(TWinControl)
  private
  ...
    FBorderWidth: TBorderWidth;
  ...
    procedure SetBorderWidth(Value: TBorderWidth);
  ...
  published
  ...
    property BorderWidth: TBorderWidth read FBorderWidth write SetBorderWidth;
  ...

constructor TProgressBar.Create(AOwner: TComponent);
begin
  ...
  FBorderWidth := inherited BorderWidth;
  ...
end;

procedure TProgressBar.SetBorderWidth(Value: TBorderWidth);
begin
  if Value > (Height div 2) - 3 then Exit;
  if Value <> inherited BorderWidth then begin
    inherited BorderWidth := Value;
    FBorderWidth := inherited BorderWidth;
  end;
end;
```

Аналогичный подход можно использовать в других компонентах. В процедуре SetBorderWidth вместо Exit можно создать исключение, но это уже на любителя.

[VS]

TrackBar для эстетов

В стандартном компоненте всегда присутствует диапазон выбора (см. на рис. нижний TrackBar). К сожалению, в Delphi не предусмотрена возможность его отключения. Если нет необходимости в использовании диапазона выбора, то поможет небольшая доработка компонента.




```
unit NTrackBar;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ComCtrls, CommCtrl;

type
  TNTrackBar = class(TTrackBar)
  private
    FSelRangeVisible: Boolean;
    procedure SetSelRangeVisible(const Value: Boolean);
  protected
    procedure CreateParams(var Params: TCreateParams); override;
  public
    constructor Create(AOwner: TComponent); override;
  published
    property SelRangeVisible: Boolean read FselRangeVisible
                                      write SetSelRangeVisible;
  end;

procedure Register;

implementation
{ TNTrackBar }

constructor TNTrackBar.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  FSelRangeVisible := False;
  ThumbLength := 18;
end;
```

```
procedure TNTrackBar.CreateParams(var Params: TCreateParams);
begin
  inherited CreateParams(Params);
  with Params do begin
    if not FSelRangeVisible then Style := Style xor TBS_ENABLESELRANGE
    else Style := Style or TBS_ENABLESELRANGE;
  end;
end;

procedure TNTrackBar.SetSelRangeVisible(const Value: Boolean);
begin
  if FSelRangeVisible <> Value then begin
    FSelRangeVisible := Value;
    RecreateWnd;
  end;
end;

procedure Register;
begin
  RegisterComponents('Controls', [TNTrackBar]);
end;

end.
```

[VS]

Чтение текста RTF из базы данных

Запись текста RTF в файл и сохранение этого файла в базе данных является наиболее простым способом сохранения текста компонента в таблице, но тот же способ может быть достигнут и без использования промежуточного файла, а именно с помощью TBLOBStream. Пример, приведенный ниже, демонстрирует чтение RTF из таблицы.

```
procedure ReadRichEditFromTable(Table: TTable; var RichEdit: TRichEdit);
var
  BLOBStream: TBLOBStream;
begin
  try
    BLOBStream := TBLOBStream.Create(Table.FieldByName('BODY') as TBLOBField,
                                     bmRead);
    if (not Table.FieldByName('BLOBFieldName').IsNull) then begin
      RichEdit.Lines.LoadFromStream(BLOBStream);
    end;

  finally
    BLOBStream.Free;
  end;
end;
```

[Лагонский Сергей]

Подсчет слов в TRichEdit

Как подсчитать количество слов в компоненте `TRichEdit`*?*

Решение

```
var
  f: text;
  Count: word;

function GetWord: boolean;
var
  s: string;
  c: char;
begin
  result := False;
  s := ' ';
  while not EoF(f) do begin
    read(f, c);
    if not (c in ['a'..'z', 'A'..'Z'{, ...}]) then break;
    s := s + c;
  end;
  result := (s <> ' ');
end;

procedure GetWordCount(TextFile: string);
begin
  Count := 0;
  AssignFile(f, TextFile);
  Reset(f);
  while not EoF(f) do if GetWord then inc(Count);
  CloseFile(f);
end;
```

Примечание

Решение не очень красивое, т. к. количество слов подсчитывается в файле, т. е. текст из `TRichEdit` *надо сначала сохранить в файл, а потом сосчитать в нем слова.*

Ошибка TRichEdit в Windows NT

Я написал программу, передающую невидимому `TRichEdit` *информацию, введенную пользователем с помощью кнопок и нескольких* `TCheckBox` *(флажков, или кнопок с независимой фиксацией). Программа разрабатывалась и эксплуатировалась под Windows 95 и работала без проблем. Но в Windows NT 4.0 строка* `RichEdit1.Print('');` *возвращала ошибку «Divide by Zero» (деление на ноль). Единственный выход из создавшегося положения заключался в сохранении файла с последующей его загрузкой и печатью с помощью MS Word. Кому-нибудь приходилось решать эту проблему?*

Чтобы решить эту проблему, необходимо небольшое вмешательство в VCL – модуль ComCtrls.pas.

```
procedure TCustomRichEdit.Print(const Caption: string);
var
  Range: TFormatRange;
  LastChar, MaxLen, LogX, LogY: Integer;
begin
  FillChar(Range, SizeOf(TFormatRange), 0);
  with Printer, Range do begin
    LogX := GetDeviceCaps(Handle, LOGPIXELSX);
    LogY := GetDeviceCaps(Handle, LOGPIXELSY);
// позиционированный вызов BeginDoc для обеспечения совместимости под NT4.0 и Win95
    BeginDoc;
    hdc := Handle;
    hdcTarget := hdc;
    if IsRectEmpty(PageRect) then begin
      rc.right := PageWidth * 1440 div LogX;
      rc.bottom := PageHeight * 1440 div LogY;
    end else begin
      rc.left := PageRect.Left * 1440 div LogX;
      rc.top := PageRect.Top * 1440 div LogY;
      rc.right := PageRect.Right * 1440 div LogX;
      rc.bottom := PageRect.Bottom * 1440 div LogY;
    end;
    rcPage := rc;
    Title := Caption;
// Оригинальная позиция BeginDoc
{ BeginDoc; }
    LastChar := 0;
    MaxLen := GetTextLen;
    chrg.cpMax := -1;
    repeat
      chrg.cpMin := LastChar;
      LastChar := SendMessage(Self.Handle, EM_FORMATRANGE, 1, Longint(@Range));
      if (LastChar < MaxLen) and (LastChar <> -1) then NewPage;
    until (LastChar >= MaxLen) or (LastChar = -1);
    EndDoc;
  end;
  SendMessage(Handle, EM_FORMATRANGE, 0, 0);
end;
```

Проблема печати RTF

Проблема заключается в том, что компонент для работы с .RTF, входящий в поставку Delphi (да и другие аналогичные свободные или коммерческие компоненты) перед печатью требует загрузки файла целиком. Файл, созданный ранее, может быть очень большим и попросту не поместиться в память. Опция быстрой печати – это хорошо, но и в этом случае приходится ждать, пока файл не загрузится в память полностью.

Решение

```
ShellExecute(MainForm.Handle, nil, 'write.exe', 'myfile.rtf /p', nil, SW_HIDE);
```

Параметр /p является недокументированным. Код сначала запустит WordPad, напечатает файл и затем закроет WordPad. SW_HIDE является единственным параметром, позволяющим хоть как-то прятать работу WordPad.

WordPad загружает в память файл, но осуществляется это благодаря механизму сегментации, когда в необходимый момент времени подгружается другая часть файла. При работе с WordPad файл загружался и выводился на печать довольно быстро. Проблем совместимости не существует – WordPad имеется на каждой системе Windows 9x.

Исправление загрузки текста RTF через поток

В версии Borland Delphi 3 Client/Server обнаружено, что при загрузке текста формата RTF методом LoadFromStream в компонент TRichEdit он не интерпретируется как RTF, а отображается полностью (со всеми управляющими символами). Ниже приведен исправленный текст реализации метода TRichEditStrings.LoadFromStream (измененные строки отмечены комментарием {!}):

```
procedure TRichEditStrings.LoadFromStream(Stream: TStream);

var
  EditStream: TEditStream;
  Position: Longint;
  TextType: Longint;
  StreamInfo: TRichEditStreamInfo;
  Converter: TConversion;

begin
  StreamInfo.Stream := Stream;
  if FConverter <> nil then Converter := FConverter
  else Converter := RichEdit.DefaultConverter.Create;
  StreamInfo.Converter := Converter;
  try
    with EditStream do begin
      dwCookie := LongInt(Pointer(@StreamInfo));
      pfnCallBack := @StreamLoad;
      dwError := 0;
    end;
    Position := Stream.Position;
    if PlainText then TextType := SF_TEXT
    else TextType := SF_RTF;
    SendMessage(RichEdit.Handle, EM_STREAMIN, TextType, Longint(@EditStream));
    if (TextType = SF_RTF) and (EditStream.dwError <> 0) then begin
      Stream.Position := Position;
{!}   if PlainText then TextType := SF_TEXT
{!}   else TextType := SF_RTF;
      SendMessage(RichEdit.Handle, EM_STREAMIN, TextType, Longint(@EditStream));
```

```
        if EditStream.dwError <> 0 then
          raise EOutOfResources.Create(sRichEditLoadFail);
      end;

    finally
      if FConverter = nil then Converter.Free;
    end;
  end;
```

[Лагонский Сергей]

Ограничение размера текста в TRichEdit

Я разместил на форме компонент TRichEdit*, присвоил ему большой (104 Кбайт) текст, но все попытки загрузить его приводили к уменьшению размера текста. Это проблема распределения памяти? Есть решение этой проблемы?*

Решение 1

Ответ на этот вопрос нужно искать в Windows API:

```
SendMessage(MyRichEdit.Handle, EM_EXLIMITTEXT, 0, NewBigSize);
```

[News Group]

Решение 2

У этого компонента есть свойство MaxLength, которое работает некорректно. Поэтому лучше сделать так:

```
RichEdit.Perform(EM_LIMITTEXT, { нужный размер }, 0);
```

Причем перед каждым открытием файла это действие необходимо повторять.

Если передать в качестве размера 0, то ОС ограничивает размер значением OS Specific Default Value. Реально следует по результатам экспериментов поставить размер чуть меньший, чем доступная виртуальная память.

Для того чтобы не повторять этот вызов (EM_LIMITTEXT), можно воспользоваться сообщением EM_EXLIMITTEXT.

[Nomadic]

Вставка текста в TRichEdit

Как вставить в нужное место RTF-текст в TRichEdit*?*

Можно послать сообщение EM_STREAMIN с параметром SFF_SELECTION методом Perform для замены текущего Selection.

[Nomadic]

Позиция курсора в TRichEdit

Как определить позицию курсора в TRichEdit*?*

Решение

```
procedure TForm1.GetPosition(Sender: TRichEdit);
var
  iX, iY: Integer;
  TheRichEdit: TRichEdit;
begin
  iX := 0;
  iY := 0;
  TheRichEdit := TRichEdit(Sender);
  iY := SendMessage(TheRichEdit.Handle, EM_LINEFROMCHAR, TheRichEdit.SelStart, 0);
  iX := TheRichEdit.SelStart - SendMessage(TheRichEdit.Handle, EM_LINEINDEX, iY, 0);
  Label1.Caption := IntToStr(iY + 1) + ':' + IntToStr(iX + 1) ;
end;

procedure TForm1.RichEdit1MouseDown(Sender: TObject; Button: TMouseButton;
                                    Shift: TShiftState; X, Y: Integer);
begin
  GetPosition(RichEdit1);
end;

procedure TForm1.RichEdit1KeyUp(Sender: TObject; var Key: Word;
                                Shift: TShiftState);
begin
  GetPosition(RichEdit1);
end;
```

Прокрутка TRichEdit

Как обеспечить прокрутку в TRichEdit*?*

Используйте сообщение EM_SCROLL:

```
SendMessage(RichEdit1.Handle, EM_SCROLL, SB_LINEDOWN, 0);
```

Модернизация компонента TRichEdit

В моем компоненте TRichEdit *расположен текст на русском языке с вкраплением английских слов. В компоненте я указал нужный мне язык, но при перемещении курсора по тексту компонент сам меняет его значение. Как мне исправить это?.*

Решение

```
unit RichEditEx;

interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
```

```
  StdCtrls, ComCtrls;
type
  TLangChangeEvent = procedure(Sender: TObject; Lang: HKL) of object;
  TRichEditEx = class(TRichEdit)
  private
    FOnLangChange: TLangChangeEvent;
    procedure WMLangRequest(var M: TMessage); message WM_INPUTLANGCHANGEREQUEST;
    procedure WMLangChange(var M: TMessage); message WM_INPUTLANGCHANGE;
  published
    property OnLangChange: TLangChangeEvent read FOnLangChange write FOnLangChange;
  end;

procedure Register;
implementation

procedure TRichEditEx.WMLangRequest(var M: TMessage);
begin
  if Assigned(FOnLangChange) then FOnLangChange(Self, M.LParam);
  inherited;
end;

procedure TRichEditEx.WMLangChange(var M: TMessage);
begin
  M.Result := 1;
end;

procedure Register;
begin
  RegisterComponents('Samples', [TRichEditEx]);
end;

end.
```

[Гуменюк Максим]

Группа переключателей и ActiveControl

На форме имеется группа переключателей (radiobuttons). Я хотел бы вызывать контекстно-зависимую подсказку, если пользователь нажал клавишу <F1>. *Для данной конкретной группы переключателей я установил* `HelpContext` *равным 22, но при любом вызове* `ActiveControl.HelpContext` *возвращается 0. Все другие элементы управления работают как положено. Что я делаю неправильно?*

Проблема в том, что `ActiveControl` – `RadioButton`, а не `RadioButtonGroup`. Поместите следующий код в обработчик события формы `OnShow`.

```
procedure TForm1.FormShow(Sender: TObject);
var
  c: integer;
begin
  with RadioGroup1 do begin
```

```
      for c := 0 to ControlCount - 1 do
        TRadioButton(Controls[c]).HelpContext := HelpContext;
    end;
  end;
```

[News Group]

Синхронизация двух компонентов ScrollBox

Решить задачу помогут обработчики событий OnScroll (в данном примере два компонента ScrollBox – ScrollBar1 и ScrollBar2 – расположены на форме TMainForm):

```
procedure TMainForm.ScrollBar1Scroll(Sender: TObject; ScrollCode: TScrollCode;
                                     var ScrollPos: Integer);
begin
  ScrollBar2.Position := ScrollPos;
end;

procedure TMainForm.ScrollBar2Scroll(Sender: TObject; ScrollCode: TScrollCode;
                                     var ScrollPos: Integer);
begin
  ScrollBar1.Position := ScrollPos;
end;
```

Мерцание ScrollBar

TScrollBar *в Delphi мигает при получении фокуса. Как избежать этого мерцания?*

Такая же проблема и при перемещении стандартного бегунка полосы прокрутки. Исправляется одинаково – установкой свойства TabStop в False.

Двойной щелчок на TSpeedButton

Событие OnDblClick компонента TSpeedButton происходит только тогда, когда кнопка является частью группы (одна и более кнопок имеют GroupIndex > 0).

SpeedButton и Glyph

Могу ли я из ресурсов поочередно загружать глифы для кнопок SpeedButton. *Если да, то как это сделать?*

Если в проекте используется DBNavigator, то пиктограммы кнопок компонента будут связываться с вашей программой. Их можно загрузить для применения в ваших SpeedButton следующим образом:

```
SpeedButton1.Caption := '';
SpeedButton1.Glyph.LoadFromResourcename(HInstance, 'DBN_REFRESH');
SpeedButton1.NumGlyphs := 2;
```

Другие зарезервированные имена: DBN_PRIOR, DBN_DELETE, DBN_CANCEL, DBN_EDIT, DBN_FIRST, DBN_INSERT, DBN_LAST, DBN_NEXT, DBN_POST.

Все имена должны быть набраны в верхнем регистре.

[News Group]

Обработчик события OwnerDraw в компоненте StatusBar

Обработчик должен выглядеть примерно так:

```
procedure TForm1.StatusBar1DrawPanel(StatusBar: TStatusBar; Panel: TStatusPanel;
                                     const Rect: TRect);
begin
  with StatusBar1.Canvas do begin
    Brush.Color := clRed;
    FillRect(Rect);
    TextOut(Rect.Left, Rect.Top, 'Панель ' + IntToStr(Panel.Index));
  end;
end;
```

Примечание

Создайте StatusPanel *на* StatusBar *и установите ее свойство* Style = psOwnerDraw.

Отображение всплывающих подсказок в строке состояния

Решение 1

Данный пример показывает, как сделать, чтобы строка состояния (StatusBar) отображала все всплывающие подсказки (Hint) элементов управления формы при наведении курсора мыши на область компонента.

Расположите TStatusBar на всех формах, для которых в строке состояния должны отображаться подсказки. Установите свойство SimplePanel в True и присвойте компоненту другое имя, например SBStatus.

Создайте необходимые подсказки в свойствах Hint. Не забудьте вставить символ |, если вам необходима комбинация короткого и длинного текста.

Примечание

*Если сделать так, как сказано в Решении 1, то «поиск» строки для отображения произойдет автоматически (*StatusBar *будет назначен в* DisplayHint*).*

Поместите следующую строку в обработчик события FormCreate вашей формы:

```
Application.OnHint := DisplayHint;
```

Создайте эту процедуру. Пожалуйста, обратите внимание на комментарии.

```
procedure TFrmMain.DisplayHint(Sender: TObject);
var
  Counter, NumComps: integer;
begin
  with Screen.ActiveForm do begin
    NumComps := ControlCount - 1;
    for Counter := 0 to NumComps do
{ SBStatus - имя всех моих компонентов TStatusBar. При необходимости его можно
  изменить }
      if (TControl(Controls[Counter]).Name = 'SBStatus') then begin
        if (Application.Hint = '') then
{ ConWorkingName - используемая константа. При необходимости ее можно
  изменить }
          TStatusBar(Controls[Counter]).SimpleText := ConWorkingName
        else
          TStatusBar(Controls[Counter]).SimpleText := Application.Hint;
        break;
      end;
  end;
end;
```

Не забудьте поместить `procedure DisplayHint(Sender: TObject)` в секцию `public`.

Это все, что вы должны сделать. Для того чтобы придать такую функциональность другим формам, просто поместите на них `TStatusBar` и установите свойство `Hint` для необходимых компонентов.

Примечание

Переменная `NumComps` *является избыточной в данном случае. Если пользователю необходим «короткий»* `Hint` *на элементе и «длинный»* `Hint` *на* `StatusBar`*, то от всего предыдущего обработчика можно оставить строку:*

```
StatusBar1.SimpleText := Application.Hint;
```

Хочется добавить, что не обязательно вся строка состояния `StatusBar` *должна представлять собой* `SimplePanel`*, можно отвести для этого отдельную* `StatusPanel` `(Style = psText)`*, в которой и отображать* `Hint`*, внеся соответствующие исправления в процедуру* `DisplayHint`*.*

Решение 2

Обратитесь к стандартному методу `TForm.FindComponent`. В этом случае метод `ShowHint` выглядит проще:

```
procedure TAnyForm.ShowHint;
var
  C: TStatusBar;
begin
// ищем наш StatusBar1 на активной форме
  C := TStatusBar(Screen.ActiveForm.FindComponent('StatusBar1'));
// если не найден - ищем на основной форме
  if not Assigned(C) then
```

```
    C := TStatusBar(Application.MainForm.FindComponent('StatusBar1'));
// если что-то обнаружено - рисуем на нем наш текст
  if Assigned(C) then C.SimpleText := '  ' + Application.Hint;
end;
```

[Иваненко Федор]

Дополнительная информация в строке состояния

Предположим, у нас есть StatusBar с четырьмя панелями, плюс таймер. Тогда можно сделать следующее:

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  with StatusBar1 do begin
    if GetKeyState(VK_CAPITAL) <> 0 then Panels[0].Text := ' CAP'
    else Panels[0].Text := '';
    if GetKeyState(VK_NUMLOCK) <> 0 then Panels[1].Text := ' NUM'
    else Panels[1].Text := '';
    if GetKeyState(VK_SCROLL) <> 0 then Panels[2].Text := ' SCRL'
    else Panels[2].Text := '';
    Panels[3].Text := ' ' + DateTimeToStr(Now);
  end;
end;
```

О том, как можно изменить формат вывода даты, доходчиво и с примерами изложено в электронной справке, в разделе, посвященном датам. Обратите внимание, что свойство Text имеет тип строки, поэтому нельзя написать Panels[0].Text := Now, т. к. дата/время имеет тип Double.

Пример компонента Status представлен ниже:

```
unit Status;
interface
uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, ExtCtrls, Menus, Gauges;
type
  TStatus = class(TCustomPanel)
  private
    FDate: Boolean;
    FKeys: Boolean;
    FTime: Boolean;
    FResources: Boolean;
    DateTimePanel: TPanel;
    ResPanel: TPanel;
    ResGauge: TGauge;
    CapPanel: TPanel;
    NumPanel: TPanel;
    InsPanel: TPanel;
    HelpPanel: TPanel;
```

```
    UpdateWidth: Boolean;
    FTimer: TTimer;
    procedure SetDate(A: Boolean);
    procedure SetKeys(A: Boolean);
    procedure SetTime(A: Boolean);
    procedure SetResources(A: Boolean);
    procedure SetCaption(A: String);
    function GetCaption: String;
    procedure CMFontChanged(var Message: TMessage); message CM_FONTCHANGED;
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    procedure SetupPanelFields(ThePanel: TPanel);
    procedure SetupPanel(ThePanel: TPanel; WidthMask: String);
    procedure UpdateStatusBar(Sender: TObject);
  published
    property ShowDate: Boolean read FDate write SetDate default True;
    property ShowKeys: Boolean read FKeys write SetKeys default True;
    property ShowTime: Boolean read FTime write SetTime default True;
    property ShowResources: Boolean read FResources write SetResources
                            default True;
    property Caption: string read GetCaption write SetCaption;
  end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('Additional', [TStatus]);
end;
procedure TStatus.SetupPanelFields(ThePanel: TPanel);
begin
  with ThePanel do begin
    Alignment := taCenter;
    Caption := '';
    BevelInner := bvLowered;
    BevelOuter := bvNone;
{ Установите все в True, чтобы все это отразилось на TStatus }
    ParentColor := True;
    ParentFont := True;
    ParentCtl3D := True;
  end;
end;

procedure TStatus.SetupPanel(ThePanel: TPanel; WidthMask: String);
begin
  SetupPanelFields(ThePanel);
  with ThePanel do begin
    Width := Canvas.TextWidth(WidthMask);
```

```
    Align := alRight;
  end;
end;

constructor TStatus.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  Parent := TWinControl(AOwner);
  FTime := True;
  FDate := True;
  FKeys := True;
  FResources := True;    { Заставляем строку состояния выровняться по нижнему краю }
  Align := alBottom;
  Height := 19;
  BevelInner := bvNone;
  BevelOuter := bvRaised;
{ Если UpdateWidth равен TRUE, StatusBar пересчитывает только ширину панелей }
  UpdateWidth := True;
  Locked := True;
  TabOrder := 0;
  TabStop := False;
  Font.Name := 'Arial';
  Font.Size := 8;
{ Создаем панель, которая будет отображать дату и время }
  DateTimePanel := TPanel.Create(Self);
  DateTimePanel.Parent := Self;
  SetupPanel(DateTimePanel, ' 00/00/00 00:00:00 дп ');
{ Создаем панель, которая будет содержать графику ресурсов }
  ResPanel := TPanel.Create(Self);
  ResPanel.Parent := Self;
  SetupPanel(ResPanel, '                    ');
{ Создаем 2 Gauges, которые размещаем на Resource Panel }
  ResGauge := TGauge.Create(Self);
  ResGauge.Parent := ResPanel;
  ResGauge.Align := alClient;
  ResGauge.ParentFont := True;
  ResGauge.BackColor := Color;
  ResGauge.ForeColor := clLime;
  ResGauge.BorderStyle := bsNone;
{ Создаем панель, которая будет отображать состояние CapsLock }
  CapPanel := TPanel.Create(Self);
  CapPanel.Parent := Self;
  SetupPanel(CapPanel, ' Cap ');
{ Создаем панель, которая будет отображать состояние NumLock }
  NumPanel := TPanel.Create(Self);
  NumPanel.Parent := Self;
  SetupPanel(NumPanel, ' Num ');
{ Создаем панель, которая будет отображать состояние Insert/Overwrite }
  InsPanel := TPanel.Create(Self);
  InsPanel.Parent := Self;
  SetupPanel(InsPanel, ' Ins ');
```

```
{ Создаем панель, которая будет отображать текст состояния }
  HelpPanel := TPanel.Create(Self);
  HelpPanel.Parent := Self;
  SetupPanelFields(HelpPanel);
{ Имеем вспомогательную панель, занимающую все остальное пространство }
  HelpPanel.Align := alClient;
  HelpPanel.Alignment := taLeftJustify;
{ Это таймер, который регулярно обновляет строку состояния }
  FTimer := TTimer.Create(Self);
  if FTimer <> Nil then begin
    FTimer.OnTimer := UpdateStatusBar;
{ Обновление происходит дважды в секунду }
    FTimer.Interval := 500;
    FTimer.Enabled := True;
  end;
end;
destructor TStatus.Destroy;
begin
  FTimer.Free;
  HelpPanel.Free;
  InsPanel.Free;
  NumPanel.Free;
  CapPanel.Free;
  ResGauge.Free;
  ResPanel.Free;
  DateTimePanel.Free;
  inherited Destroy;
end;

procedure TStatus.SetDate(A: Boolean);
begin
  FDate := A;
  UpdateWidth := True;
end;
procedure TStatus.SetKeys(A: Boolean);
begin
  FKeys := A;
  UpdateWidth := True;
end;

procedure TStatus.SetTime(A: Boolean);
begin
  FTime := A;
  UpdateWidth := True;
end;

procedure TStatus.SetResources(A: Boolean);
begin
  FResources := A;
  UpdateWidth := True;
end;
```

```
{ Если мы получаем или устанавливаем заголовок TStatus, то вместо этого задаем
  заголовок HelpPanel }
procedure TStatus.SetCaption(A: String);
begin
  HelpPanel.Caption := ' ' + A;
end;

function TStatus.GetCaption: String;
begin
  GetCaption := HelpPanel.Caption;
end;

{ Данная процедура устанавливает соответствующие заголовки }
procedure TStatus.UpdateStatusBar(Sender: TObject);
begin
  if ShowDate and ShowTime then DateTimePanel.Caption := DateTimeToStr(Now)
  else if ShowDate and not ShowTime then DateTimePanel.Caption := DateToStr(Date)
  else if not ShowDate and ShowTime then DateTimePanel.Caption := TimeToStr(Time)
  else DateTimePanel.Caption := '';
  if UpdateWidth then
    with DateTimePanel do
      if ShowDate or ShowTime then Width := Canvas.TextWidth(' ' + Caption + ' ')
      else Width := 0;
  if ShowResources then begin
    ResGauge.Progress := GetFreeSystemResources(GFSR_SYSTEMRESOURCES);
    if ResGauge.Progress < 20 then ResGauge.ForeColor := clRed
    else ResGauge.ForeColor := clLime;
  end;
  if UpdateWidth then
    if ShowResources then
      ResPanel.Width := Canvas.TextWidth('                    ')
    else
      ResPanel.Width := 0;
  if ShowKeys then begin
    if (GetKeyState(vk_NumLock) and $01) <> 0 then NumPanel.Caption := ' Num '
    else NumPanel.Caption := '';
    if (GetKeyState(vk_Capital) and $01) <> 0 then  CapPanel.Caption := ' Cap '
    else CapPanel.Caption := '';
    if (GetKeyState(vk_Insert) and $01) <> 0 then InsPanel.Caption := ' Ins '
    else InsPanel.Caption := '';
  end;
  if UpdateWidth then
    if ShowKeys then begin
      NumPanel.Width := Canvas.TextWidth(' Num ');
      InsPanel.Width := Canvas.TextWidth(' Ins ');
      CapPanel.Width := Canvas.TextWidth(' Cap ');
    end else begin
      NumPanel.Width := 0;
      InsPanel.Width := 0;
      CapPanel.Width := 0;
    end;
```

```
  UpdateWidth := False;
end;

{ Позволяем изменять шрифты, используемые панелями для вывода текста }
procedure TStatus.CMFontChanged(var Message: TMessage);
begin
  inherited;
  UpdateWidth := True;
end;

end.
```

Примечание

Код написан в Delphi 1, поэтому потребуется небольшая доработка для работы в старших версиях Delphi.

Установка атрибутов «только для чтения» для столбцов компонента StringGrid

Манипулирование вышеуказанным атрибутом возможно в обработчике события `OnSelectCell`:

```
if ACol mod 2 = 0 then
  StringGrid1.Options := StringGrid1.Options + [goEditing]
else
  StringGrid1.Options := StringGrid1.Options - [goEditing];
```

Помещение изображения в ячейку StringGrid

Как поместить изображение в одну из ячеек компонента `StringGrid`*?*

Это можно сделать с помощью обработчика события `OnDrawCell`. Приводим скелет кода, демонстрирующий принцип вывода изображения в ячейке компонента:

```
with StringGrid1.Canvas do begin
  ...

  Draw(Rect.Left, Rect.Top, Image1.Picture.Graphic);
  ...

end;
```

Достичь цели позволяют методы `Draw()` и `StretchDraw()` объекта `TCanvas`. В приведенном примере переменная `Image1` класса `TImage` содержит заранее загруженное изображение.

Сохранение и чтение StringGrid

Как сохранить `StringGrid` *со всеми ячейками в файле?*

Решение

```
procedure TForm1.SaveGrid;
var
  f: TextFile;
  X, Y: integer;
begin
  AssignFile(f, 'Filename');
  Rewrite(f);
  Writeln(f, StringGrid.ColCount);
  Writeln(f, StringGrid.RowCount);
  for X := 0 to StringGrid.ColCount - 1 do
    for Y := 0 to StringGrid.RowCount - 1 do
      Writeln (f, StringGrid.Cells[x, y]);
  CloseFile(f);
end;

procedure TForm1.LoadGrid;
var
  f: TextFile;
  temp, X, Y: integer;
  tempstr: string;
begin
  AssignFile(f, 'Filename');
  Reset(f);
  Readln(f, temp);
  StringGrid.ColCount := temp;
  Readln(f, temp);
  StringGrid.RowCount := temp;
  for X := 0 to StringGrid.ColCount - 1 do
    for Y := 0 to StringGrid.RowCount - 1 do begin
      Readln(F, tempstr);
      StringGrid.Cells[x, y] := tempstr;
    end;
  CloseFile(f);
end;
```

TStringGrid с переносом текста в ячейках

Компонент `TWrapGrid`, функционально идентичный `TStringGrid`, но умеющий переносить текст в ячейках. Это предварительный вариант. Работая с компонентом, не забывайте про `RowHeights` (или `DefaultRowHeight`), т. к. при переносе текста потребуется отобразить несколько строк.

```
{ Компонент позволяет переносить текст в ячейках TStringGrid.
  Вы можете использовать его вместо стандартного TStringGrid. }
unit Wrapgrid;
```

```
interface

uses
  SysUtils, Windows, Messages, Classes, Graphics, Controls, Forms, Dialogs, Grids;

type
  TWrapGrid = class(TStringGrid)
  protected
    procedure DrawCell(ACol, ARow: Longint; ARect: TRect;
                       AState: TGridDrawState); override;
  public
    constructor Create(AOwner: TComponent); override;
  end;
procedure Register;

implementation

constructor TWrapGrid.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  DefaultDrawing := False;
end;

{ Процедура DrawCell осуществляет перенос текста в ячейке }
procedure TWrapGrid.DrawCell(ACol, ARow: Longint; ARect: TRect;
                             AState: TGridDrawState);
var
  Sentence,                   { Выводимый текст }
  CurWord: String;            { Текущее выводимое слово }
  SpacePos,                   { Позиция первого пробела }
  CurX,                       { X координата 'курсора' }
  CurY: Integer;              { Y координата 'курсора' }
  EndOfSentence: Boolean;     { Величина, указывающая на заполненность ячейки }
begin
{ Инициализируем шрифт, чтобы он был управляющим шрифтом }
  Canvas.Font := Font;
  with Canvas do begin
{ Если это фиксированная ячейка, тогда используем фиксированный цвет }
    if gdFixed in AState then begin
      Pen.Color := FixedColor;
      Brush.Color := FixedColor;
    end else begin            { в противном случае используем нормальный цвет }
      Pen.Color := Color;
      Brush.Color := Color;
    end;
{ Рисуем подложку цветом ячейки }
    Rectangle(ARect.Left, ARect.Top, ARect.Right, ARect.Bottom);
  end;
{ Начинаем рисование с верхнего левого угла ячейки }
  CurX := ARect.Left;
  CurY := ARect.Top;
```

```
{ Здесь мы получаем содержание ячейки }
  Sentence := Cells[ACol, ARow];
{ для каждого слова ячейки }
  EndOfSentence := False;
  while (not EndOfSentence) do begin
{ для получения следующего слова ищем пробел }
    SpacePos := Pos(' ', Sentence);
    if SpacePos > 0 then begin
{ получаем текущее слово плюс пробел }
      CurWord := Copy(Sentence, 0, SpacePos);
{ получаем остальную часть предложения }
      Sentence := Copy(Sentence, SpacePos + 1, Length(Sentence) - SpacePos);
    end else begin
{ это - последнее слово в предложении }
      EndOfSentence := TRUE;
      CurWord := Sentence;
    end;
    with Canvas do begin
{ если текст выходит за границы ячейки }
      if (TextWidth(CurWord) + CurX) > ARect.Right then begin
{ переносим на следующую строку }
        CurY := CurY + TextHeight(CurWord);
        CurX := ARect.Left;
      end;
{ выводим слово }
      TextOut(CurX, CurY, CurWord);
{ увеличиваем X координату курсора }
      CurX := CurX + TextWidth(CurWord);
    end;
  end;
end;

procedure Register;
begin
  RegisterComponents('Samples', [TWrapGrid]);
end;

end.
```

StringGrid и файловый поток

Каково наилучшее решение для сохранения экземпляра `TStringGrid` *(150×10)?*

Сохранение на диске:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  myStream: TFileStream;
begin
  myStream := TFileStream.Create('grid1.sav', fmCreate);
  myStream.WriteComponent(StringGrid1);
```

```
  myStream.Destroy;
end;
```

Для чтения:

```
  ...
myStream := TFileStream.Create('grid1.sav', fmOpenRead);
StringGrid1 := myStream.ReadComponent(StringGrid1) as TStringGrid;
  ...
```

Выравнивание текста в колонках StringGrid

Решение 1

Организуйте обработчик события сетки OnDrawCell. Поместите код, подобный этому, в обработчик (выравнивание по правому краю):

```
procedure TForm1.StringGrid1DrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
var
  Txt: array [0..255] of Char;
begin
  StrPCopy(Txt, StringGrid1.Cells[ACol, ARow]);
  SetTextAlign(StringGrid1.Canvas.Handle, GetTextAlign(StringGrid1.Canvas.Handle)
      and not (TA_LEFT or TA_CENTER) or TA_RIGHT);
  ExtTextOut(StringGrid1.Canvas.Handle, Rect.Right - 2, Rect.Top + 2,
             ETO_CLIPPED or ETO_OPAQUE, @Rect, Txt, StrLen(Txt), nil);
end;
```

[News Group]

Решение 2

Нижеприведенный код выравнивает данные компонента по правому краю:

```
procedure TForm1.StringGrid1DrawCell(Sender: TObject; ACol, ARow: Integer;
                              Rect: TRect; State: TGridDrawState);
var
  lRow, lCol: Longint;
begin
  lRow := ARow;
  lCol := ACol;
  with Sender as TStringGrid, Canvas do begin
    if (gdSelected in State) then Brush.Color := clHighlight
    else if (gdFixed in State) then Brush.Color := FixedColor
         else Brush.Color := Color;
    FillRect(Rect);
    SetBkMode(Handle, TRANSPARENT);
    SetTextAlign(Handle, TA_RIGHT);
    TextOut(Rect.Right - 2, Rect.Top + 2, Cells[lCol, lRow]);
  end;
end;
```

Хитрость заключается в установке выравнивания текста TA_RIGHT, позволяющей осуществлять вывод текста, начиная с правой стороны (от правой границы).

Решение 3

```
procedure TForm1.StringGrid1DrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
var
  dx: integer;
begin
  with (Sender as TStringGrid).Canvas do begin
    Font := StringGrid1.Font;
    Pen.Color := clBlack;
    if (ACol = 0) or (ARow = 0) then begin
{ Рисуем заголовок }
      Brush.Color := clBtnFace;
      FillRect(Rect);
      TextOut(Rect.Left, Rect.Top, StringGrid1.Cells[ACol, ARow])
    end else begin
{ Рисуем ячейку с правым выравниванием }
      Brush.Color := clWhite;
      FillRect(Rect);
      dx := TextWidth (StringGrid1.Cells[ACol, ARow]) + 2;
      TextOut(Rect.Right - dx, Rect.Top, StringGrid1.Cells[ACol, ARow])
    end;
  end;
end;
```

Решение 4

Код компонента для Delphi на основе стандартного TStringGrid. Компонент позволяет переносить текст в TStringGrid. В качестве «отправной точки» был выбран компонент TWrapGrid (автор Luis J. de la Rosa, e-mail: *delarosa@ix.netcom.com*), в исходный код которого были внесены изменения.

Для использования:

- выберите в Delphi пункты меню Component/Install Component...;
- найдите и выберите файл с именем NewStringGrid.pas;
- нажмите кнопку OK;
- после этого вы увидите компонент на вкладке Other палитры компонентов Delphi.

```
unit NewStringGrid;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, Grids;
```

```
type
  TAlignText = (atLeft, atCenter, atRight, atWrapTop, atWrapCenter, atWrapBottom);
  TNewStringGrid = class(TStringGrid)
  private
    FAlignText: TAlignText;
    FAlignCaption: TAlignText;
    FCenter: Boolean;
    procedure SetAlignText(Value: TAlignText);
    procedure SetAlignCaption(Value: TAlignText);
    procedure SetCenter(Value: Boolean);
  protected
    procedure DrawCell(ACol, ARow: Longint; ARect: TRect;
                       AState: TGridDrawState); override;
  public
    constructor Create(AOwner: TComponent); override;
  published
    property AlignText: TAlignText read FAlignText write SetAlignText;
    property AlignCaption: TAlignText read FAlignCaption write SetAlignCaption;
    property Center: Boolean read FCenter write SetCenter;
  end;

procedure Register;
implementation

procedure Register;
begin
  RegisterComponents('Other', [TNewStringGrid]);
end;
{ TNewStringGrid }

constructor TNewStringGrid.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  AlignText := atLeft;
  AlignCaption := atCenter;
  Center := False;
  DefaultColWidth := 80;
  DefaultRowHeight := 18;
  Height := 100;
  Width := 408;
  DefaultDrawing := False;
end;

{ Процедура DrawCell осуществляет перенос текста в ячейке }
procedure TNewStringGrid.DrawCell(ACol, ARow: Integer; ARect: TRect;
                                  AState: TGridDrawState);
var
  CountI,                   { Счетчик }
  CountWord: Integer;       { Счетчик }
  Sentence,                 { Выводимый текст }
  CurWord: String;          { Текущее выводимое слово }
```

```
SpacePos,                  { Позиция первого пробела }
CurXDef,                   { X-координата 'курсора' по умолчанию }
CurYDef,                   { Y-координата 'курсора' по умолчанию }
CurX,                      { X-координата 'курсора' }
CurY: Integer;             { Y-координата 'курсора' }
EndOfSentence: Boolean;    { Величина, указывающая на заполненно сть ячейки }
Alig: TAlignText;          { Тип выравнивания текста }
ColPen: TColor;            { Цвет карандаша по умолчанию }
MassWord: array [0..255] of String;
MassCurX, MassCurY: array [0..255] of Integer;
LengthText: Integer;       { Длина текущей строки }
MassCurYDef: Integer;
MeanCurY: Integer;

procedure VisualCanvas;
begin
{ Прорисовываем ячейку и придаем ей 3D-вид }
  with Canvas do begin
{ Запоминаем цвет пера для последующего вывода текста }
    ColPen := Pen.Color;
    if gdFixed in AState then begin
      Pen.Color := clWhite;
      MoveTo(ARect.Left, ARect.Top);
      LineTo(ARect.Left, ARect.Bottom);
      MoveTo(ARect.Left, ARect.Top);
      LineTo(ARect.Right, ARect.Top);
      Pen.Color := clBlack;
      MoveTo(ARect.Left, ARect.Bottom);
      LineTo(ARect.Right, ARect.Bottom);
      MoveTo(ARect.Right, ARect.Top);
      LineTo(ARect.Right, ARect.Bottom);
    end;
    Pen.Color := ColPen;   { Восстанавливаем цвет пера }
  end;
end;

procedure VisualBox;
begin
{ Инициализируем шрифт, чтобы он был управляющим шрифтом }
  Canvas.Font := Font;
  with Canvas do begin
{ Если это фиксированная ячейка, тогда используем фиксированный цвет }
    if gdFixed in AState then begin
      Pen.Color := FixedColor;
      Brush.Color := FixedColor;
    end
{ в противном случае используем нормальный цвет }
    else begin
      Pen.Color := Color;
      Brush.Color := Color;
    end;
```

```
{ Рисуем подложку цветом ячейки }
    Rectangle(ARect.Left,ARect.Top,ARect.Right,ARect.Bottom);
  end;
end;

procedure VisualText(Alig: TAlignText);
begin
  case Alig of
      atLeft:  begin
                 with Canvas do
               { выводим текст }
                   TextOut(CurX, CurY, Sentence);
                 VisualCanvas;
               end;
     atRight:  begin
                 with Canvas do
               { выводим текст }
                   TextOut(ARect.Right - TextWidth(Sentence) - 2, CurY,Sentence);
                   VisualCanvas;
               end;
    atCenter:  begin
                 with Canvas do
               { выводим текст }
                   TextOut(ARect.Left + ((ARect.Right - ARect.Left
                         - TextWidth(Sentence)) div 2), CurY, Sentence);
                 VisualCanvas;
               end;
   atWrapTop:  begin
              { для каждого слова ячейки }
                 endOfSentence := False;
                 CountI := 0;
                 while CountI <= SpacePos do begin
                   MassWord[CountI] := '';
                   CountI := CountI + 1;
                 end;
                 CountI := 0; CountWord := CurY;
                 while (not endOfSentence) do begin
              { для получения следующего слова ищем пробел }
                   SpacePos := Pos(' ', Sentence);
                   if SpacePos>0 then begin
              { получаем текущее слово плюс пробел }
                     CurWord := Copy(Sentence, 0, SpacePos);
              { получаем остальную часть предложения }
                     Sentence := Copy(Sentence, SpacePos + 1,
                     Length(Sentence) - SpacePos);
                   end
                   else begin
              { это - последнее слово в предложении }
                     endOfSentence := True;
                     CurWord := Sentence;
                   end;
```

```
                with Canvas do begin
            { если текст выходит за границы ячейки }
                  LengthText := TextWidth(CurWord) + CurX + 2;
                  if LengthText > ARect.Right then begin
            { переносим на следующую строку }
                    CurY := CurY + TextHeight(CurWord);
                    CurX := CurXDef + 2;
                  end;
                  if CountWord <> CurY then CountI := CountI + 1;
                  MassWord[CountI] := MassWord[CountI] + CurWord;
            { увеличиваем X-координату курсора }
                  CurX := CurX + TextWidth(CurWord);
                  CountWord := CurY;
                end;
              end;
              with Canvas do begin
                CountWord := 0;
                CurY := CurYDef + 2;
                CurX := CurXDef+2;
                while CountWord <= CountI do begin
                  case Center of
                    True:  begin
                             CurWord := MassWord[CountWord];
                             if Copy(CurWord, Length(CurWord) - 1, 1)=' ' then
                               MassWord[CountWord] := Copy(CurWord, 0,
                                                      Length(CurWord) - 1);
                           MassCurX[CountWord] := ARect.Left + ((ARect.Right -
                       - ARect.Left - TextWidth (MassWord[CountWord])) div 2);
                             MassWord[CountWord] := CurWord;
                           end;
                    False:  MassCurX[CountWord] := CurX;
                  end;
                 MassCurY[CountWord] := CurY;                  { выводим слово }
                             TextOut(MassCurX[CountWord], MassCurY[CountWord],
                          MassWord[CountWord]);
                  CurY := CurY + TextHeight(MassWord[CountWord]);
                  CountWord := CountWord + 1;
                end;
              end;
              VisualCanvas;
            end;
atWrapCenter:  begin
            { для каждого слова ячейки }
              endOfSentence := False;
              CountI := 0;
              while CountI <= SpacePos do begin
                MassWord[CountI] := '';
                CountI := CountI + 1;
              end;
              CountI := 0;
              CountWord := CurY;
```

```
  while (not endOfSentence) do begin
{ для получения следующего слова ищем пробел }
    SpacePos := Pos(' ', Sentence);
    if SpacePos>0 then begin
{ получаем текущее слово плюс пробел }
      CurWord := Copy(Sentence, 0, SpacePos);
{ получаем остальную часть предложения }
      Sentence := Copy(Sentence, SpacePos + 1, Length(Sentence) -
                       SpacePos);
    end else begin
{ это - последнее слово в предложении }
      endOfSentence := True;
      CurWord := Sentence;
    end;
    with Canvas do begin
{ если текст выходит за границы ячейки }
      LengthText := TextWidth(CurWord) + CurX + 2;
      if LengthText>ARect.Right then begin
{ переносим на следующую строку }
        CurY := CurY + TextHeight(CurWord);
        CurX := CurXDef + 2;
      end;
      if CountWord <> CurY then CountI := CountI + 1;
      MassWord[CountI] := MassWord[CountI] + CurWord;
{ увеличиваем X-координату курсора }
      CurX := CurX + TextWidth(CurWord);
      CountWord := CurY;
    end;
  end;
  with Canvas do begin
    CountWord := 0;
    CurX := CurXDef + 2;
    while CountWord <= CountI do begin
      case Center of
        True: begin
                CurWord := MassWord[CountWord];
               if Copy(CurWord, Length(CurWord) - 1, 1) = ' ' then
                  MassWord[CountWord] := Copy(CurWord, 0,
                                              Length(CurWord) - 1);
                MassCurX[CountWord] := ARect.Left + ((ARect.Right -
            - ARect.Left - TextWidth(MassWord[CountWord])) div 2);
                  MassWord[CountWord] := CurWord;
              end;
         False: MassCurX[CountWord] := CurX;
      end;
      MassCurY[CountWord] := TextHeight(MassWord[CountWord]);
      CountWord := CountWord + 1;
    end;
    CountWord := 0;
    MassCurYDef := 0;
    while CountWord <= CountI do begin
```

```
          MassCurYDef := MassCurYDef + MassCurY[CountWord];
          CountWord := CountWord + 1;
        end;
        MassCurYDef := (ARect.Bottom - ARect.Top - MassCurYDef) div 2;
        CountWord := 0;
        MeanCurY := 0;
        while CountWord <= CountI do begin
          MassCurY[CountWord] := ARect.Top + MeanCurY + MassCurYDef;
          MeanCurY := MeanCurY + TextHeight(MassWord[CountWord]);
          CountWord := CountWord + 1;
        end;
        CountWord := -1;
        while CountWord<=CountI do begin
          CountWord := CountWord + 1;
          if MassCurY[CountWord] < (ARect.Top + 2) then Continue;
      { выводим слово }
          TextOut(MassCurX[CountWord], MassCurY[CountWord],
                  MassWord[CountWord]);
        end;
      end;
      VisualCanvas;
    end;

atWrapBottom:  begin
     { для каждого слова ячейки }
      endOfSentence := False;
      CountI := 0;
      while CountI <= SpacePos do begin
        MassWord[CountI] := '';
        CountI := CountI + 1;
      end;
      CountI := 0;
      CountWord := CurY;
      while (not endOfSentence) do begin
    { для получения следующего слова ищем пробел }
        SpacePos := Pos(' ', Sentence);
        if SpacePos > 0 then begin
    { получаем текущее слово плюс пробел }
          CurWord := Copy(Sentence, 0, SpacePos);
    { получаем остальную часть предложения }
          Sentence := Copy(Sentence,SpacePos + 1,
                      Length(Sentence) - SpacePos);
        end else begin
    { это - последнее слово в предложении }
          endOfSentence := True;
          CurWord := Sentence;
        end;
        with Canvas do begin
    { если текст выходит за границы ячейки }
          LengthText := TextWidth(CurWord) + CurX + 2;
          if LengthText > ARect.Right then begin
```

```
{ переносим на следующую строку }
        CurY := CurY + TextHeight(CurWord);
        CurX := CurXDef + 2;
      end;
      if CountWord <> CurY then CountI := CountI + 1;
      MassWord[CountI] := MassWord[CountI] + CurWord;
      CurX := CurX + TextWidth(CurWord);
      CountWord := CurY;
    end;
  end;
  with Canvas do begin
    CountWord := 0;
    CurX := CurXDef + 2;
    while CountWord <= CountI do begin
      case Center of
        True:  begin
                 CurWord := MassWord[CountWord];
               if Copy(CurWord, Length(CurWord) -1, 1)=' ' then
                   MassWord[CountWord] := Copy(CurWord, 0,
                                          Length(CurWord) - 1);
             MassCurX[CountWord] := ARect.Left + ((ARect.Right -
        - ARect.Left - TextWidth (MassWord[CountWord])) div 2);
                 MassWord[CountWord] := CurWord;
               end;
        False:  MassCurX[CountWord] := CurX;
      end;
      MassCurY[CountWord] := TextHeight(MassWord[CountWord]);
      CountWord := CountWord + 1;
    end;
    CountWord := 0; MassCurYDef := 0;
    while CountWord <= CountI do begin
      MassCurYDef := MassCurYDef + MassCurY[CountWord];
      CountWord := CountWord + 1;
    end;
    MassCurYDef := ARect.Bottom - MassCurYDef - 2;
    CountWord := 0;
    MeanCurY := -MassCurY[CountWord];
    while CountWord <= CountI do begin
      MeanCurY := MeanCurY + MassCurY[CountWord];
      MassCurY[CountWord] := MassCurYDef + MeanCurY;
      CountWord := CountWord + 1;
    end;
    CountWord := -1;
    while CountWord <= CountI do begin
      CountWord := CountWord + 1;
      if MassCurY[CountWord] < (ARect.Top + 2) then Continue;
{ выводим слово }
      TextOut(MassCurX[CountWord], MassCurY[CountWord],
              MassWord[CountWord]);
    end;
  end;
```

```
                  VisualCanvas;
                end;
    end;
  end;
begin
  VisualBox;
  VisualCanvas;
{ Начинаем рисование с верхнего левого угла ячейки }
  CurXDef := ARect.Left;
  CurYDef := ARect.Top;
  CurX := CurXDef + 2;
  CurY := CurYDef + 2;
{ Здесь мы получаем содержание ячейки }
  Sentence := Cells[ACol, ARow];
{ Если ячейка пуста, выходим из процедуры }
  if Sentence='' then Exit;
{ Проверяем длину строки (не более 256 символов) }
  if Length(Sentence) > 256 then begin
     MessageBox(0, 'Число символов не должно быть более 256.','Ошибка в таблице',
                 mb_OK);
     Cells[ACol, ARow] := '';
     Exit;
  end;
{ Узнаем, сколько в предложении слов, и задаем размерность массивов }
  SpacePos := Pos(' ',Sentence);
{ Узнаем тип выравнивания текста }
  if gdFixed in AState then Alig := AlignCaption
  else Alig := AlignText;
  VisualText(Alig);
end;

procedure TNewStringGrid.SetAlignCaption(Value: TAlignText);
begin
  if Value<>FAlignCaption then FAlignCaption := Value;
end;

procedure TNewStringGrid.SetAlignText(Value: TAlignText);
begin
  if Value<>FAlignText then FAlignText := Value;
end;
procedure TNewStringGrid.SetCenter(Value: Boolean);
begin
  if Value<>FCenter then FCenter := Value;
end;
end.
```

[Pavel Stont]

Помещение компонентов в StringGrid

Возможно ли поместить элемент управления, например, CheckBox или ComboBox внутрь компонента xxxGrid*?*

1. При создании компонента (в обработчике OnCreate) создайте его объекты Objects[C, R], например, TCheckBox.Create(Self). Имейте в виду, что необходимо присвоить ячейкам Cells[C, R] какие-либо значения, для того чтобы получить доступ к Objects[C, R]. Установите у вновь созданного компонента свойство Visible в False, а свойство Parent в Self. Осуществите другую необходимую инициализацию. Кроме того, надо внести необходимые модули в список uses, если создается тип компонента, которого нигде, кроме как на форме, нет.
2. Создайте процедуру, принимающую координаты колонки/строки и правильно позиционирующую соответствующий объект в пределах прямоугольника ячейки, например:

```
procedure TForm1.FixObjPosn(vCol, vRow: LongInt);
{ Размещаем содержимое компонента в области прямоугольника ячейки }
var
  R: TRect;
begin
  R := StringGrid1.CellRect(vCol, vRow);
  if StringGrid1.Objects[vCol, vRow] is TControl then
    with TControl(StringGrid1.Objects[vCol, vRow]) do
      if R.Right = R.Left then {прямоугольник ячейки невидим}
        Visible := False
      else begin
        InflateRect(R, -1, -1);
        OffsetRect(R, StringGrid1.Left + 1, StringGrid1.Top + 1);
        BoundsRect := R;
        Visible := True;
      end;
end;
```

Примечание

Смещение позиции необходимо, поскольку CellRect *расчитывается относительно верхнего левого угла строки сетки, и родителем компонента является форма.*

3. В обработчике события сетки OnSelectCell проверьте, располагается ли элемент Objects в текущей колонке Col и строке Row TControl. Если так, установите его свойство Visible в False. Теперь вызовите процедуру установления координат из шага 2 для новых Col и Row, передавая их из параметров обработчика события в параметры функции.
4. В обработчике OnTopLeftChanged просто вызовите FixObjPosn;
5. В обработчике события OnDrawCell, если ячейка выбрана, выполняется инструкция Exit, а если элемент ячейки Objects не TControl, также Exit. В

противном случае нужно создать код, обеспечивающий отрисовку «фасада» каждого типа элемента управления, который вы разместили в сетке;

6. Обратите внимание на то, что если вы делаете что-либо с элементом управления, на который влияют другие элементы управления (например, изменение статуса какого-либо переключателя из группы или операции `Enable/Disable`), вы должны вызвать метод сетки `Refresh`.

[News Group]

Выбор строки/колонки компонента TStringGrid

Процедура, выбирающая при нажатии кнопки первую строку сетки. Ее работа не зависит от размера сетки и количества фиксированных строк/колонок.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  NewSel: TGridRect;
begin
  with StringGrid1 do begin
    NewSel.Left := FixedCols;
    NewSel.Top := FixedRows;
    NewSel.Right := ColCount - 1;
    NewSel.Bottom := FixedRows;
    Selection := NewSel;
  end;
end;
```

Примечание

`StringGrid1.Row` – *номер строки от нуля;*

`StringGrid1.Col` – *номер столбца от нуля.*

Ширина колонок StringGrid

Я использую компонент `StringGrid` *и хотел бы менять ширину его колонок в соответствии с расположенным в них текстом. Другими словами, я хочу, чтобы весь текст в них был виден, но как это сделать?*

```
procedure TForm1.StringGrid1SelectCell(Sender: TObject; ACol, ARow: Longint;
                                       var CanSelect: Boolean);
var
  Wid: Integer;
begin
  with Sender as TStringGrid do begin
    Wid := Canvas.TextWidth(Cells[Col, Row] + ' ');
    if Wid > ColWidths[Col] then ColWidths[Col] := Wid;
  end;
end;
```

```
procedure TForm1.StringGrid1KeyPress(Sender: TObject; var Key: Char);
var
  Wid: Integer;
begin
  if Key = #13 then
    with Sender as TStringGrid do begin
      Wid := Canvas.TextWidth(Cells[Col, Row] + ' ');
      if Wid > ColWidths[Col] then ColWidths[Col] := Wid;
  end;
end;
```

[News Group]

Цвет неактивной ячейки StringGrid

Если я щелкаю мышью в любой ячейке StringGrid2*, то последняя выбранная ячейка в* StringGrid1 *становится синей. Как избавиться от этого эффекта?*

Создайте обработчик (если он отсутствует) события сетки OnDrawCell и включите в него следующий код:

```
procedure TForm1.StringGrid1DrawCell(Sender: TObject; ACol, ARow: Integer;
                                     Rect: TRect; State: TGridDrawState);
begin
  if Sender = ActiveControl then Exit;
  if not (gdSelected in State) then Exit;
  with Sender AS TStringGrid do begin
    Canvas.Brush.Color := Color;
    Canvas.Font.Color := Font.Color;
    Canvas.TextRect(Rect, Rect.Left + 2, Rect.Top + 2, Cells[ACol, ARow]);
  end;
end;
```

Данный метод выполняется немедленно после того, как сетка становится неактивной, или выбор переходит к другим ячейкам. В любом из этих случаев необходимо нарисовать невыбранную ячейку для неактивной сетки – т. е. в тех случаях, когда получается «неправильный» цвет. Разработчик просто берет работу Delphi по закрашиванию ячеек на себя, пропуская DefaultDrawing (отрисовку по умолчанию) для таких ячеек, но в то же время разрешая Delphi поработать «за себя» во всех остальных случаях.

[News Group]

Вставка и удаление строк в StringGrid

Я не нашел никаких методов для вставки и удаления строк. Что мне делать?

Поскольку свойство Cols[x] компонента TStringGrid имеет тип TStrings, все методы TStrings применимы также и к Cols[x].

```
{ Всякий раз при удалении Row (строки) или Column (колонки) проверяйте наличие и
  удаляйте любые объекты, которые могли быть назначены любой ячейке в строке или
  колонке, которые вы собираетесь удалять, поскольку данный код не может знать ни
  размера, ни типа ассоциированных с ними объектов. }
unit GridFunc;

interface
uses
  Sysutils, Windows, Grids;

procedure InsertRow(Sender: TStringGrid; ToIndex: Longint);
procedure DeleteRow(Sender: TStringGrid; FromIndex: Longint);
procedure InsertColumn(Sender: TStringGrid; ToIndex: Longint);
procedure DeleteColumn(Sender: TStringGrid; FromIndex: Longint);

implementation

type
  TCSGrid = class(TStringGrid)
  public
    procedure MoveRow(FromIndex, ToIndex: Longint);
    procedure MoveColumn(FromIndex, ToIndex: Longint);
  end;

procedure TCSGrid.MoveRow(FromIndex,ToIndex: Longint);
begin
  RowMoved(FromIndex, ToIndex);              { Защищенный метод TStringGrid }
end;

procedure TCSGrid.MoveColumn(FromIndex, ToIndex: Longint);
begin
  ColumnMoved(FromIndex, ToIndex);           { Защищенный метод TStringGrid }
end;

procedure InsertRow(Sender: TStringGrid; ToIndex: Longint);
var
  xx, yy: Integer;
begin
  if ToIndex >= 0 then
    with TCSGrid(Sender) do
      if (ToIndex <= RowCount) then begin
        RowCount := RowCount + 1;
        xx := RowCount - 1;
        for yy := 0 to ColCount - 1 do begin
          Cells[yy, xx] := ' ';
          ObJects[yy, xx] := nil;
        end;
        if ToIndex < RowCount - 1 then MoveRow(RowCount - 1, ToIndex);
      end else MessageBeep(0)
  else MessageBeep(0);
end;
```

```
procedure DeleteRow(Sender: TStringGrid; FromIndex: Longint);
begin
  if FromIndex >= 0 then
    with TCSGrid(Sender) do
      if (RowCount > 0) and (FromIndex < RowCount) then begin
        if (FromIndex < RowCount - 1) then
          MoveRow(FromIndex, RowCount - 1);
        Rows[RowCount - 1].Clear;
        RowCount := RowCount - 1;
      end else MessageBeep(0)
  else MessageBeep(0);
end;

procedure InsertColumn(Sender: TStringGrid; ToIndex: Longint);
var
  xx, yy: Integer;
begin
  if ToIndex >= 0 then
    with TCSGrid(Sender) do
      if (ToIndex<=ColCount) then begin
        ColCount := ColCount + 1;
        xx := ColCount - 1;
        Cols[xx].BeginUpdate;
        for yy := 0 to RowCount - 1 do begin
          Cells[xx, yy] := ' ';
          ObJects[xx,yy] := nil;
        end;
        Cols[xx].EndUpdate;
        if ToIndex < ColCount - 1 then
          MoveColumn(ColCount - 1, ToIndex);
        end else MessageBeep(0)
  else MessageBeep(0);
end;

procedure DeleteColumn(Sender: TStringGrid; FromIndex: Longint);
begin
  if FromIndex >= 0 then
    with TCSGrid(Sender) do
      if (ColCount > 0) and (FromIndex < ColCount) then begin
        if (FromIndex < ColCount - 1) then
          MoveColumn(FromIndex, ColCount - 1);
        Cols[ColCount - 1].Clear;
        ColCount := ColCount - 1;
      end
      else MessageBeep(0)
  else MessageBeep(0);
end;

end.
```

[News Group]

Обновление картинки в ячейке StringGrid

Если в таблице используется событие OnDrawCell для помещения в ячейку рисунков, причем различных, в зависимости, например, от соответствующего значения в двумерном массиве, и необходимо, чтобы после изменения значения в массиве обновилось изображение (Refresh не подходит, т. к. будет наблюдаться мелькание), то измените значение ячейки (DrawGrid не годится):

```
StringGrid1.Cells[i, j] := '';
```

или

```
StringGrid1.Cells[i, j] := StringGrid1.Cells[i, j];
```

если там что-то хранится.

[SottNick]

Многострочность в заголовках колонок StringGrid

У меня есть StringGrid*, который выглядит очень красивым, за исключением заголовков колонок. Я хочу, чтобы каждый заголовок находился в одной ячейке и мог занимать несколько строк. Как это сделать?*

Решение 1

Рисовать сами ячейки можно в обработчике события OnDrawCell. Для определения ячейки, обрабатываемой в текущий момент, используйте параметр GridState.

```
type
  TFTVerticalAlignment = (vaTop, vaMiddle, vaBottom);

procedure DrawTextAligned(const Text: string; Canvas: TCanvas; var Rect: TRect;
                          Alignment: TAlignment; VerticalAlignment:
                          TFTVerticalAlignment; WordWrap: Boolean);
var
  P: array[0..255] of Char;
  H: Integer;
  T: TRect;
  F: Word;
begin
  StrPCopy(P, Text);
  T := Rect;
  with Canvas, Rect do begin
    F := DT_CALCRECT or DT_EXPANDTABS or DT_VCENTER or TextAlignments[Alignment];
    if WordWrap then F := F or DT_WORDBREAK;
    H := DrawText(Handle, P, -1, T, F);
    H := MinInt(H,Rect.Bottom - Rect.Top);
    if VerticalAlignment = vaMiddle then begin
      Top := ((Bottom + Top) - H) div 2;
```

```
      Bottom := Top + H;
    end else if VerticalAlignment = vaBottom then
      Top := Bottom - H - 1;
    F := DT_EXPANDTABS or DT_VCENTER or TextAlignments[Alignment];
    if WordWrap then F := F or DT_WORDBREAK;
    DrawText(Handle, P, -1, Rect, F);
  end;
end;
```

Примечание

Пример для Delphi 1.

Решение 2

Ниже приведен пример, который создает многострочный центрированный заголовок, набранный жирным шрифтом:

```
procedure TForm1.StringGrid1DrawCell(Sender: TObject; ACol, ARow: Longint;
                                     Rect: TRect; State: TGridDrawState);
var
  l_oldalign: word;
  l_YPos, l_XPos, i: integer;
  s, s1: string;
  l_col, l_row: longint;

begin
  l_col := ACol;
  l_row := ARow;
  with Sender as TStringGrid do begin
    if (l_row = 0) then
      Canvas.Font.Style := Canvas.Font.Style + [fsbold];
    if l_row = 0 then begin
      l_oldalign := SetTextAlign(Canvas.Handle, ta_center);
      l_XPos := Rect.Left + (Rect.Right - Rect.Left) div 2;
      s := Cells[l_col, l_row];
      while s <> '' do begin
        if Pos(#13, s) <> 0 then begin
          if Pos(#13, s) = 1 then s1 := ''
          else begin
            s1 := Trim(Copy(s, 1, Pred(Pos(#13, s))));
            Delete(s, 1, Pred(Pos(#13, s)));
          end;
          Delete(s, 1, 2);
        end else begin
          s1 := Trim(s);
          s := '';
        end;
        l_YPos := Rect.top + 2;
        Canvas.TextRect(Rect, l_Xpos, l_YPos, s1);
        inc(Rect.top, RowHeights[l_row] div 3);
      end;
```

```
        SetTextAlign(Canvas.Handle, l_oldalign);
      end else
        Canvas.TextRect(Rect, Rect.Left + 2, Rect.Top + 2, Cells[l_col, l_row]);
      Canvas.Font.Style := Canvas.Font.Style - [fsbold];
    end;
  end;
```

StringGrid без выделенной ячейки

Я пытаюсь показать `StringGrid` *без выделенной ячейки, а первая нефиксированная ячейка всегда отображается «инвертированным» цветом. Я не хочу давать пользователю возможность редактирования сетки, но эта выделенная ячейка производит впечатление, что сетку можно редактировать...*

Необходимо создать обработчик события `OnDrawCell`. Это легче, чем вы думаете.

```
procedure TForm1.StringGrid1DrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);

var
  Col: integer absolute ACol;
  Row: integer absolute ARow;
  Buf: array[byte] of char;

begin
  if (gdFixed in State) then Exit;
  with StringGrid1 do begin
    Canvas.Font := Font;
    Canvas.Font.Color := clWindowText;
    Canvas.Brush.Color := clWindow;
    Canvas.FillRect(Rect);
    StrPCopy(Buf, Cells[ACol, ARow]);
    DrawText(Canvas.Handle, Buf, -1, Rect, DT_SINGLELINE or DT_VCENTER or DT_NOCLIP
             or DT_LEFT);
  end;

end;
```

[News Group]

Один щелчок на StringGrid вместо трех

Как сделать, чтобы после первого щелчка на ячейке можно было начать редактировать ее содержимое?

Включите `goAlwaysShowEditor` в свойство `TStringGrid.Options`.

[News Group]

StringGrid как DBGrid

Это может выглядеть приблизительно так:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  i, _Row: word;
begin
  Table1.First;
  _Row := 0;
  StringGrid1.RowCount := Table1.RecordCount;
  while not Table1.EoF do begin
    for i := 0 to Table1.FieldCount - 1 do
      StringGrid1.Cells[i, _Row] := Table1.Fields[i].AsString;
    inc(_Row);
    Table1.Next;
  end;
end;
```

Имеются причины использования TStringGrid вместо DBGrid. Код, который загружает данные из отфильтрованной таблицы, представлен ниже. Он не очень изящен, т. к. на самом деле является лишь черновиком, но работает очень быстро, даже если надо загрузить сотню колонок.

```
var
  fDB: TTable;

procedure TForm1.FillCells;
var
  Row, i, w: integer;
  grid: TStringGrid;
begin
  StringGrid1.RowCount := 0;
  if not Assigned(fDB) then exit;
  Row := 0;
// Данный временный объект-сетка используется для предохранения от огромного
// количества подразумеваемых событий Application.ProcessMessages,
// инициируемых базой данных и вызывающих противное моргание объекта doGrid. Итак,
// мы загружаем данные в объект-сетку и затем копируем их в стобцы, начиная с
// верхней части.
  grid := TStringGrid.Create(Self);
  grid.Visible := False;
  with fDB do
    try
      grid.ColCount := Fields.Count;
      DisableControls;
// Фильтр был установлен с помощью свойства Self.Filter
      First;
      while not EoF do
        try
          grid.RowCount := Row + 1;
```

```
            for i := 0 to grid.ColCount - 1 do begin
              grid.Cells[i, Row] := Fields[i].AsString;
              w := StringGrid1.Canvas.TextWidth(grid.Cells[i, Row]);
              if StringGrid1.ColWidths[i] < w then StringGrid1.ColWidths[i] := w;
            end;
            Inc(Row);
          finally
            Next;
          end;

      finally
        StringGrid1.RowCount := grid.RowCount;
        StringGrid1.ColCount := grid.ColCount;
        for i := 0 to grid.ColCount - 1 do begin
          StringGrid1.Cols[i] := grid.Cols[i];
          StringGrid1.ColWidths[i] := StringGrid1.ColWidths[i] + 4
        end;
        grid.Free;
        EnableControls;
      end;

  end;
```

[News Group]

«Авторазмер» для StringGrid

Устанавливать размер ячеек «руками» утомительно. Есть ли какие-нибудь предложения по этой теме?

Проблему можно решить программным путем (это нужно делать после того, как вы загрузите данные):

```
var
  i, j, temp, max: integer;

  ...
  for i := 0 to Grid.ColCount - 1 do begin
    max := 0;
    for j := 0 to Grid.RowCount - 1 do begin
      temp := Grid.Canvas.TextWidth(Grid.Cells[i, j]);
      if temp > max then max := temp;
    end;

    Grid.ColWidths[i] := max + Grid.GridLineWidth + 1;
  end;
  ...
```

Вероятно, надо будет добавить 1, чтобы текст не прилипал к границам ячеек.

Раскрашенный StringGrid

Решение 1

У StringGrid имеется свойство Objects, позволяющее назначать объекты. Создайте объект, содержащий переменную типа TColor, и назначьте его Objects[Col, Row], что позволит иметь к нему доступ в любое время. Назначьте событие OnDrawCell компонента StringGrid, позволяющее рисовать текст ячейки нужного цвета. Чтобы убедиться, что ячейка выбрана, воспользуйтесь свойством Selection, содержащим данные о выборе пользователя. Все это должно выглядеть приблизительно так:

```
type
  TStrColor = class(TObject)
  public
    Color: TColor;
{ вы могли бы также определить частные и открытые методы доступа }
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  i, j: Integer;
begin
  with StringGrid1 do
    for i := 0 to ColCount - 1 do
      for j := 0 to RowCount - 1 do
        Objects[i, j] := TStrColor.Create;
  end;

procedure TForm1.StringGrid1DrawCell(Sender: TObject; ACol, ARow: Longint;
                                     Rect: TRect; State: TGridDrawState);
var
  OldColor: TColor;

begin
  with StringGrid1.Canvas do begin
    OldColor := Font.Color;
    Font.Color := (StringGrid1.Objects[ACol, ARow] as TStrColor).Color;
    TextOut(Rect.Left + 2, Rect.Top + 2, StringGrid1.Cells[ACol, ARow]);
    Font.Color := OldColor;
  end;
end;

procedure TForm1.ProcessSelection(Sender: TObject);
var
  i, j: Integer;
begin
  with StringGrid1.Selection do
    for i := Left to Right do
      for j := Top to Bottom do
        MessageDlg(IntToStr(i) + ',' + IntToStr(j) + '-'
```

```
                + IntToStr((StringGrid1.Objects[i, j] as TStrColor).Color),
                  mtInformation, [mbOk], 0);
end;
```

Этот компонент не поддерживает множественный выбор.

Решение 2

В данном обработчике демонстрируется техника изменения цвета выводимого в StringGrid текста.

```
procedure TForm1.StringGrid1DrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
const
  CharOffset = 3;
begin
  with StringGrid1.Canvas do begin
    Font.Color := clMaroon;
    TextOut(Rect.Left + CharOffset, Rect.Top + CharOffset, 'L');
    Font.Color := clNavy;
    TextOut(Rect.Left + CharOffset + TextWidth('L'),
    Rect.Top + CharOffset, 'loyd');
  end;
end;
```

Использование <Tab> в StringGrid как <Enter>

Данный код перемещает ввод на другую колонку. Из последней колонки строки ввод переходит на следующую строку. При достижении самого конца сетки управление перемещается в ее самое начало. Естественно, программист может изменить такое поведение и передавать управление в этом случае другому элементу управления.

```
procedure TForm1.StringGrid1KeyPress(Sender: TObject; var Key: Char);
begin
  if Key = #13 then
    with StringGrid1 do
      if Col < ColCount - 1 then
      { следующая колонка }
        Col := Col + 1
      else if Row < RowCount - 1 then begin
      { следующая строка }
        Row := Row + 1;
        Col := 1;
      end else begin
      { Конец сетки - Снова перемещаемся наверх }
        Row := 1;
        Col := 1;
{ или вы можете передать управление другому элементу управления }
      end;
end;
```

Поиск в StringGrid по маске

Решение

```
  ...
  list: TStringList;              { TStringList для поиска }
  target: string;                 { строка для поиска, например 'LeftPelvic' }

{ Устанавливаем позицию для начала поиска, например, 0 - для старта с самого
  начала, или другое целое для старта с любого места списка.
  Код ниже делает поиск нечувствительным к регистру, если вам нужно обратное,
  уберите вызов AnsiUpperCase }
  target := AnsiUpperCase(target);
  for i := start to list.Count - 1 do
    if AnsiUpperCase(Copy(list.Item[i], 1, Length(target))) = target then
// мы нашли это!
  ...
```

Для того чтобы организовать простой поиск, т. е. искать `'LeftPelvic'` везде, а не только в начале слов, воспользуйтесь следующим кодом:

```
if Pos(target, AnsiUpperCase(list.Item[i])) > 0 then ...
// мы нашли это;
```

Потеря визуального курсора в StringGrid

Если `DefaultRowHeight <= 14`, визуальный курсор исчезает! Решение заключается в установке `DefaultRowHeight` в 15 и больше.

Разрешение экрана и StringGrid

Когда Delphi масштабирует свойство `Font` объекта `StringGrid`, то свойства `DefaultColWidth` и `DefaultRowHeight` не изменяются. Таким образом, сам `StringGrid` размера не меняет.

Решение

```
with StringGrid1 do begin
  DefaultColWidth := DefaultColWidth * MyScaleFactor;
  DefaultRowHeight := DefaultRowHeight * MyScaleFactor;
end;
```

Форматирование ячеек TStringGrid

Надо сделать две вещи:

- Обработчик `OnDrawCell`, отображающий отформатированные данные, выровненные по правому краю.
- В ячейки сетки загружайте неформатированные строки, представляющие собой ваши данные.

В обработчике OnDrawCell создайте код, подобный тому, что приведен ниже. Этот код отображает разделенные запятой числа, выровненные по правому краю:

```
procedure TForm1.StringGrid1DrawCell(Sender: TObject; ACol, ARow: Integer;
                                     Rect: TRect; State: TGridDrawState);
var
  strText: string; X, Y: integer;
begin
  if (ARow > 0) and (ACol > 0) and (StringGrid1.Cells[ACol, ARow] <> '') then begin
{ Формат строки с числом с плавающей точкой }
  strText:= FloatToStrF(StrToFloat(StringGrid1.Cells[ACol,ARow]), ffNumber, 13, 2);
{ Устанавливаем шрифт }
    StringGrid1.Canvas.Font.Name := 'Courier';
    if StrToFloat(StringGrid1.Cells[ACol, ARow]) < 0 then
      StringGrid1.Canvas.Font.Color := clRed;
    StringGrid1.Canvas.Font.Style := StringGrid1.Canvas.Font.Style - [fsBold];
{ Центрируем текст в ячейке по вертикали, по правому полю и отодвигаем его от
  правого поля на два пиксела. }
    X := Rect.Right - StringGrid1.Canvas.TextWidth(strText);
    Y := Rect.Top + ((Rect.Bottom - Rect.Top -
         StringGrid1.Canvas.TextHeight(strText)) div 2);
    Dec(Rect.Right, 2);
    StringGrid1.Canvas.TextRect(Rect, X, Y, strText);
  end;
end;
```

Убедитесь в том, что DefaultDrawing := True, – тогда вы сможете в ячейках выводить только текст, остальную отрисовку выполнит VCL.

При попытке пользователя отредактировать число в ячейке оно будет отображаться в неформатированном виде (если у вас нет необходимости в обработчике события OnGetEditText).

Но при этом может окоазаться, что проверка данных для этого способа может быть проблематична (например, что будет, если пользователь введет 'TX' в числовой колонке). Эта ситуация усугубляется тем фактом, что DrawGrid не позволяет реализовать «последовательное поведение» для различных путей навигации по сетке (например, инициализировать различные события, если вы пользуетесь курсорными стрелками, мышью или просто нажимаете клавишу <Enter>). Но и это все решается простой посылкой другого сообщения.

Добавление элементов управления в TTabbedNotebook и TNotebook

Как добавить элементы управления в TTabbedNotebook *или* TNotebook *во время выполнения программы?*

Добавление элементов управления в TTabbedNotebook во время проектирования – красивая и простая задача. Нужно лишь установить свойство PageIn-

dex или ActivePage на необходимую страницу и начать заполнять ее элементами управления.

Добавление элементов управления во время выполнения приложения также очень просто. Тем не менее, в прилагаемой документации по Delphi вы не найдете рецептов типа «Что-и-Как». Видимо, для того чтобы окончательно запутать начинающих программистов, фирма-изготовитель даже не удосужилась включить исходный код TTabbedNotebook в библиотеку VCL. Таким образом, TTabbedNotebook остается для некоторых тайной за семью печатями.

Первым шагом к раскрытию тайны послужит просмотр файла \DELPHI\DOC\TABNOTBK.INT, интерфейсной секции модуля TABNOTBK.PAS, в котором определен класс TTabbedNotebook. Беглый просмотр позволяет обнаружить класс TTabPage, описанный как хранилище элементов управления отдельной страницы TTabbedNotebook.

Вторым шагом в исследовании TTabbedNotebook может стать факт наличия свойства Pages типа TStrings. В связи с этим отметим, что Delphi-классы TStrings и TStringList соорганизуются с двумя свойствами: Strings и Objects. Другими словами, для каждой строки в TStrings есть указатель на соответствующий Objects. Во многих случаях этот дополнительный указатель игнорируется, нам же он очень пригодится.

После небольшого эксперимента выясняем, что свойство Objects указывает на нашу копию TTabPage и ссылается на имя страницы в свойстве Strings. Блестяще! Всегда полезно знать, что ищешь. Теперь посмотрим, что мы можем сделать:

```
{ Данная процедура добавляет кнопку в случайной позиции на текущей странице данного
  TTabbedNotebook }
procedure AddButton(tabNotebook: TTabbedNotebook);
var
  tabpage: TTabPage;
  button: TButton;
begin
  with tabNotebook do
    tabpage := TTabPage(Pages.Objects[PageIndex]);
  button := TButton.Create(tabpage);
  try
    with button do begin
      Parent := tabpage;
      Left := Random(tabpage.ClientWidth - Width);
      Top := Random(tabpage.ClientHeight - Height);
    end;
  except
    button.Free;
  end;
end;
```

Операция по заполнению элементами управления компонента TNotebook почти такая же, как и в TTabbedNotebook – разница лишь в типе класса – TPage

вместо TTabPage. Тем не менее, в DELPHI\DOC\EXTCTRLS.INT декларацию класса TPage вы не найдете. По неизвестной причине Borland не включил определение TPage и в -файлы документов (каталог ...\Doc), поставляемые с Delphi. Декларация TPage в EXTCTRLS.PAS (можно найти в библиотеке VCL исходников ...\Delphi\Source\VCL), правда, расположена в интерфейсной части модуля. Мы восполним пропущенную информацию о классе TPage:

```
TPage = class(TCustomControl)
private
  procedure WMNCHitTest(var Message: TWMNCHitTest); message WM_NCHITTEST;
protected
  procedure ReadState(Reader: TReader); override;
  procedure Paint; override;
public
  constructor Create(AOwner: TComponent); override;
published
  property Caption;
  property Height stored False;
  property TabOrder stored False;
  property Visible stored False;
  property Width stored False;
end;
```

Теперь, по аналогии с вышеприведенной процедурой, попробуем добавить кнопку на TNotebook. Нам надо лишь заменить "TTabbedNotebook" на "TNotebook" и "TTabPage" на "TPage". Вот что должно получиться:

```
{ Данная процедура добавляет кнопку в случайной позиции на текущей странице данного
  TNotebook }
procedure AddButton(Notebook1: TNotebook);
var
  page: TPage;
  button: TButton;
begin
  with Notebook1 do
    page := TPage(Pages.Objects[PageIndex]);
  button := TButton.Create(page);
  try
    with button do begin
      Parent := page;
      Left := Random(page.ClientWidth - Width);
      Top := Random(page.ClientHeight - Height);
    end;
  except
    button.Free;
  end;
end;
```

Недоступная страница в TabbedNotebook

Есть ли возможность в компоненте TTabbedNotebook *сделать какую-либо вкладку недоступной? То есть, запретить пользователю щелкать на ней и видеть ее содержимое?*

Да, такая возможность существует. Самый простой путь – удалить страницу (вкладку), например так:

```
with TabbedNotebook do Pages.Delete(PageIndex);
```

и снова включить ее (при необходимости), перегрузив форму.

Блокировка (а не удаление) немного мудренее, поскольку необходима организация цикла в процедуре создания формы, присваивающая имена страницам компонента TabbedNotebook. Например, так:

```
  ...
J := 0;
with TabbedNotebook do
  for I := 0 to ComponentCount - 1 do
    if Components[I].ClassName = 'TTabButton' then begin
      Components[I].Name :=
        ValidIdentifier(TTabbedNotebook(Components[I].Owner).Pages[J]) + 'Tab';
      Inc(J);
    end;
  ...
```

где ValidIdentifier – функция, которая возвращает правильный Pascal-идентификатор, производный от строки "Tab":

```
function ValidIdentifier (theString: str63): str63;
{ Конвертирует строку в правильный Pascal-идентификатор, удаляя все неправильные
  символы и добавляя символ '_', если первый символ - цифра }
var
  I, Len: Integer;
begin
  Len := Length(theString);
  for I := Len downto 1 do
    if not (theString[I] in LettersUnderscoreAndDigits) then
      Delete(theString, I, 1);
    if not (theString[1] in LettersAndUnderscore) then
     theString := '_' + theString;
  ValidIdentifier := theString;
end;
```

Затем мы можем сделать страницу компонента TabbedNotebook недоступной:

```
  ...
with TabbedNotebook do begin
  TabIdent := ValidIdentifier(Pages[PageIndex]) + 'Tab';
  TControl(FindComponent(TabIdent)).Enabled := False;
{ Переключаемся на первую доступную страницу: }
  for I := 0 to Pages.Count - 1 do begin
```

```
      TabIdent := ValidIdentifier(Pages[I]) + 'Tab';
      if TControl(FindComponent(TabIdent)).Enabled then begin
        PageIndex := I;
        Exit;
      end;
    end;
  end;
    ...
```

Следующий код восстанавливает доступность страницы:

```
    ...
  with TabbedNotebook do
    for I := 0 to Pages.Count - 1 do begin
      TabIdent := ValidIdentifier(Pages[I]) + 'Tab';
      if not TControl(FindComponent(TabIdent)).Enabled then
        TControl(FindComponent(TabIdent)).Enabled := True;
    end;
    ...
```

Динамическое создание объектов в TabbedNotebook

Как динамически поместить TEdit *на* TabbedNotebook*?*

Решение

```
procedure TForm1.TabbedNotebook1Click(Sender: TObject);
var
  myE: TEdit;
begin
  with TabbedNotebook1 do begin
    if PageIndex = 1 then begin
      myE := TEdit.Create(Self);
      myE.Left := 12;
      myE.Top := 12;
      myE.Parent := Pages.Objects[PageIndex] as TWinControl;
      myE.Show;
    end;
  end;
end;
```

Как поместить кнопку (во время выполнения программы) на страницу TabbedNotebook*?*

Решение

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Button2: Tbutton;
begin
  Button2 := TButton.Create(Self);
  Button2.Parent := TabbedNotebook1.Pages.Objects[0] as TTabPage;
  Button2.SetBounds(30, 30, 60, 30);
end;
```

Доступ к страницам TabbedNotebook

При добавлении компонентов во время выполнения программы необходимо для каждого компонента присвоить свойству Parent (родитель) страницу компонента Notebook, а не сам Notebook.

Можно сделать это следующим образом (пример для кнопки):

```
MyButton := TButton.Create(Form1);           { как обычно... }
  ...
MyButton.Parent := TTabPage(TabbedNotebook1.Pages.Objects[n]);
{ <== где 'n' - индекс желаемой страницы ==> }
```

Свойство Pages компонента Notebook имеет тип StringList и содержит список заголовков и объектов TTabPage.

При добавлении компонента на страницу TabbedNotebook во время выполнения приложения указатель на желаемую страницу для свойства Parent нового компонента должен быть назначен перед тем, как он будет реально показан. Доступ ко всем страницам TTabbedNotebook во время выполнения программы можно получить с помощью свойства-массива Objects свойства TabbedNotebook.Pages. Другими словами, страничные компоненты хранятся как объекты, присоединенные к имени страницы в списке строк свойства Pages. В следующем коде показано создание кнопки на второй странице компонента TabbedNotebook1:

```
var
  NewButton: TButton;
begin
  NewButton := TButton.Create(Self);
  NewButton.Parent := TWinControl(TabbedNotebook1.Pages.Objects[1]);
  ...
```

Вот как страница TNotebook может быть использована в качестве родителя для вновь создаваемого на ней компонента:

```
NewButton.Parent := TWinControl(Notebook1.Pages.Objects[1]);
```

То же самое, но для страницы (вкладки) TTabSet:

```
NewButton.Parent := TWinControl(TabSet1.Tabs.Objects[1]);
```

Перемещение на страницу TabSet по имени

Разместите компоненты TabSet1 и Edit1 на форме. Измените свойство Tabs компонента TabSet для размещения в списке заголовков следующих четырех вкладок:

- Hello
- World
- Of
- Delphi

Создайте обработчик события onChange компонента Edit1, как показано ниже:

```
procedure TForm1.Edit1Change(Sender: TObject);
var
  I: Integer;
begin
  for I := 0 to TabSet1.Tabs.Count - 1 do
    if Edit1.Text = TabSet1.Tabs[I] then TabSet1.TabIndex := I;
end;
```

Теперь при наборе любого из существующих имен в TEdit1 соответствующая вкладка будет выведена на передний план.

Изменение количества вкладок в TabSet во время выполнения программы

Поместите в код следующее объявление:

```
TabSet1: TTabSet;              { подразумевается, что это принадлежит Form1 }
```

Следующей строкой мы очищаем заголовки всех вкладок:

```
Form1.TabSet1.Tabs.Clear;
```

Для того чтобы добавить новую вкладку с определенным именем, воспользуйтесь следующим кодом:

```
Form1.TabSet1.Tabs.Add('какой-то заголовок');
```

Поскольку TabSet1.Tabs имеет тип TStrings, вы можете применить любой из доступных методов этого типа (AddObject, LoadFromFile и т. д.).

Ускорение работы TreeView

Немного переработанный компонент TreeView, работающий быстрее своего собрата из стандартной поставки Delphi. Кроме того, была добавлена возможность вывода текста узлов и пунктов в жирном начертании.

```
unit HETreeView;

{$R-}
interface

uses
  SysUtils, Windows, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  ComCtrls, CommCtrl;
type
  THETreeView = class(TTreeView)
  private
    FSortType: TSortType;
    procedure SetSortType(Value: TSortType);
```

```
  protected
    function GetItemText(ANode: TTreeNode): string;
  public
    constructor Create(AOwner: TComponent); override;
    function AlphaSort: Boolean;
    function CustomSort(SortProc: TTVCompare; Data: Longint): Boolean;
    procedure LoadFromFile(const AFileName: string);
    procedure SaveToFile(const AFileName: string);
    procedure GetItemList(AList: TStrings);
    procedure SetItemList(AList: TStrings);
// Жирное начертание шрифта 'Bold' должно быть свойством TTreeNode, но...
    function IsItemBold(ANode: TTreeNode): Boolean;
    procedure SetItemBold(ANode: TTreeNode; Value: Boolean);
  published
    property SortType: TSortType read FSortType write SetSortType default stNone;
  end;

procedure Register;
implementation

function DefaultTreeViewSort(Node1, Node2: TTreeNode;
                             lParam: Integer): Integer; stdcall;
begin
{ with Node1 do
    if Assigned(TreeView.OnCompare) then
      TreeView.OnCompare(Node1.TreeView, Node1, Node2, lParam, Result)
    else }
  Result := lstrcmp(PChar(Node1.Text), PChar(Node2.Text));
end;
constructor THETreeView.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  FSortType := stNone;
end;

procedure THETreeView.SetItemBold(ANode: TTreeNode; Value: Boolean);
var
  Item: TTVItem;
  Template: Integer;
begin
  if ANode = nil then Exit;
  if Value then Template := -1
  else Template := 0;
  with Item do begin
    mask := TVIF_STATE;
    hItem := ANode.ItemId;
    stateMask := TVIS_BOLD;
    state := stateMask and Template;
  end;
  TreeView_SetItem(Handle, Item);
end;
```

```
function THETreeView.IsItemBold(ANode: TTreeNode): Boolean;
var
  Item: TTVItem;

begin
  Result := False;
  if ANode = nil then Exit;
  with Item do begin
    mask := TVIF_STATE;
   hItem := ANode.ItemId;
   if TreeView_GetItem(Handle, Item) then Result := (state and TVIS_BOLD) <> 0;
  end;
end;

procedure THETreeView.SetSortType(Value: TSortType);
begin
  if SortType <> Value then begin
    FSortType := Value;
    if ((SortType in [stData, stBoth]) and Assigned(OnCompare))
       or (SortType in [stText, stBoth]) then AlphaSort;
  end;
end;

procedure THETreeView.LoadFromFile(const AFileName: string);
var
  AList: TStringList;
begin
  AList := TStringList.Create;
  Items.BeginUpdate;
  try
    AList.LoadFromFile(AFileName);
    SetItemList(AList);
  finally
    Items.EndUpdate;
    AList.Free;
  end;
end;

procedure THETreeView.SaveToFile(const AFileName: string);
var
  AList: TStringList;
begin
  AList := TStringList.Create;
  try
    GetItemList(AList);
    AList.SaveToFile(AFileName);
  finally
    AList.Free;
  end;
end;
```

```
procedure THETreeView.SetItemList(AList: TStrings);
var
  ALevel, AOldLevel, i, Cnt: Integer;
  S: string;
  ANewStr: string;
  AParentNode: TTreeNode;
  TmpSort: TSortType;

  function GetBufStart(Buffer: PChar; var ALevel: Integer): PChar;
  begin
    ALevel := 0;
    while Buffer^ in [' ', #9] do begin
      Inc(Buffer);
      Inc(ALevel);
    end;
    Result := Buffer;
  end;

begin
// Удаление всех элементов - в обычной ситуации подошло бы Items.Clear, но уж очень
// медленно
  SendMessage(handle, TVM_DELETEITEM, 0, Longint(TVI_ROOT));
  AOldLevel := 0;
  AParentNode := nil;
// Снятие флага сортировки
  TmpSort := SortType;
  SortType := stNone;
  try
    for Cnt := 0 to AList.Count - 1 do begin
      S := AList[Cnt];
      if (Length(S) = 1) and (S[1] = Chr($1A)) then Break;
      ANewStr := GetBufStart(PChar(S), ALevel);
      if (ALevel > AOldLevel) or (AParentNode = nil) then begin
        if ALevel - AOldLevel > 1 then
          raise Exception.Create('Неверный уровень TreeNode');
      end else begin
        for i := AOldLevel downto ALevel do begin
          AParentNode := AParentNode.Parent;
          if (AParentNode = nil) and (i - ALevel > 0) then
            raise Exception.Create('Неверный уровень TreeNode');
        end;
      end;
      AParentNode := Items.AddChild(AParentNode, ANewStr);
      AOldLevel := ALevel;
    end;
  finally
// Возвращаем исходный флаг сортировки...
    SortType := TmpSort;
  end;
end;
```

```
procedure THETreeView.GetItemList(AList: TStrings);
var
  i, Cnt: integer;
  ANode: TTreeNode;
begin
  AList.Clear;
  Cnt := Items.Count - 1;
  ANode := Items.GetFirstNode;
  for i := 0 to Cnt do begin
    AList.Add(GetItemText(ANode));
    ANode := ANode.GetNext;
  end;
end;

function THETreeView.GetItemText(ANode: TTreeNode): string;
begin
  Result := StringOfChar(' ', ANode.Level) + ANode.Text;
end;

function THETreeView.AlphaSort: Boolean;
var
  I: Integer;
begin
  if HandleAllocated then Result := CustomSort(nil, 0)
  else Result := False;
end;

function THETreeView.CustomSort(SortProc: TTVCompare; Data: Longint): Boolean;
var
  SortCB: TTVSortCB;
  I: Integer;
  Node: TTreeNode;
begin
  Result := False;
  if HandleAllocated then begin
    with SortCB do begin
      if not Assigned(SortProc) then lpfnCompare := @DefaultTreeViewSort
      else lpfnCompare := SortProc;
      hParent := TVI_ROOT;
      lParam := Data;
      Result := TreeView_SortChildrenCB(Handle, SortCB, 0);
    end;
    if Items.Count > 0 then begin
      Node := Items.GetFirstNode;
      while Node <> nil do begin
        if Node.HasChildren then Node.CustomSort(SortProc, Data);
        Node := Node.GetNext;
      end;
    end;
  end;
end;
```

```
procedure Register;
begin
  RegisterComponents('Win95', [THETreeView]);
end;
end.
```

Поточность TreeView

На пустой форме располагается TTreeView. *Я сохраняю ее в файле, используя* WriteComponent. *Все работает, как положено. Просматривая полученный файл, можно увидеть строки TtreeView, имя объекта и т. д. По крайней мере, файл записывается и создается, и он чем-то заполнен.*

Затем я освобождаю компонент TTreeView, *открываю поток, вызываю* ReadComponent *и затем* InsertControl. *И... получаю исключение «TreeView1 has no parent window» (TreeView1 не имеет родительского окна). В чем причина?*

Это происходит из-за того, что при установке определенных свойств TreeView требуется дескриптор окна элемента управления, а для этого необходимо иметь родителя. Решение заключается в создании пустого TreeView и передаче его в качестве параметра ReadComponent.

Попробуйте этот код:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  TreeView : TTreeView;
begin
  with TFileStream.Create('JUNK.STR', fmCreate) do
    try
      WriteComponent(TreeView1);
      TreeView1.Name := 'TreeView';
      Position := 0;
      TreeView := TTreeView.Create(Self);
      TreeView.Visible := False;
      TreeView.Parent := Self;
      ReadComponent(TreeView);
      TreeView.Top := TreeView1.Top + TreeView1.Height + 10;
      TreeView.Visible := True;
    finally
      Free;
    end;
end;
```

Примечание

Убедитесь в отсутствии конфликта имен. Данный код делает форму владельцем второго TreeView *и при ее освобождении разрушает компонент. Перед загрузкой «клона» переименовывается существующий* TreeView.

Свойство Visible *установлено в* False *перед установкой свойства* parent, *этим предотвращен показ только что созданного* TreeView *до момента загрузки его из потока.*

Получение доступа к узлам TreeView

Если вам надо выполнить поиск по дереву, может быть, для того чтобы найти узел, соответствующий определенному критерию, то не делайте это таким образом:

```
for i := 0 to Pred(MyTreeView.Items.Count) do
  if MyTreeView.Items[i].Text = 'Банзай' then Break;
```

Следующий код значительно быстрее:

```
  ...
Noddy := MyTreeView.Items[0];
Searching := True;
while (Searching) and (Noddy <> nil) do begin
  if Noddy.Text = SearchTarget then begin
    Searching := False;
    MyTreeView.Selected := Noddy;
    MyTreeView.SetFocus;
  end else Noddy := Noddy.GetNext
end;
  ...
```

[News Group]

Изменение шрифта в TreeView для выделения узлов

Как выделять строки в TreeView *жирным или обычным (бледным) шрифтом?*

Решение

```
procedure SetNodeState(node: TTreeNode; Flags: Integer);
var
  tvi: TTVItem;
begin
  FillChar(tvi, SizeOf(tvi), 0);
  tvi.HItem := Node.ItemID;
  tvi.Mask := TVIF_STATE;
  tvi.StateMask := TVIS_BOLD or TVIS_CUT;
  tvi.State := Flags;
  TreeView_SetItem(Node.Handle, tvi);
end;
```

Пример вызова:

```
SetNodeState(TreeView1.Selected, TVIS_BOLD);      // Текст выделить жирным
SetNodeState(TreeView1.Selected, TVIS_CUT);       // Пиктограмму сделать бледной
```

[Nomadic]

Отмена вставки нового узла в TreeView из приложения

Как реализовать отмену вставки нового узла в TreeView *по нажатию клавиши* <Esc>*?*

```
unit BetterTreeView;
interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ComCtrls, CommCtrl;
type
  TTVNewEditCancelEvent = procedure(Sender: TObject; Node: TTreeNode;
                                    var Delete: Boolean) of object;
  TBetterTreeView = class(TTreeView)
  protected
    FIsEditingNew: Boolean;
    FOnEditCancel: TTVChangedEvent;
    FOnNewEditCancel: TTVNewEditCancelEvent;
    procedure Edit(const Item: TTVItem); override;

  public
    function NewChildAndEdit(Node: TTreeNode; const S: String): TTreeNode;
  published
    property IsEditingNew: Boolean read FIsEditingNew;
    property OnEditCancel: TTVChangedEvent read FOnEditCancel
                                           write FOnEditCancel;
    property OnNewEditCancel: TTVNewEditCancelEvent read FOnNewEditCancel
                                           write FOnNewEditCancel;
  end;

implementation

procedure TBetterTreeView.Edit(const Item: TTVItem);
var
  Node: TTreeNode;
  Action: Boolean;
begin
  with Item do begin
    if (state and TVIF_PARAM) <> 0 then Node := Pointer(lParam)
    else Node := Items.GetNode(hItem);
    if pszText = nil then begin
      if FIsEditingNew then begin
        Action := True;
        if Assigned(FOnNewEditCancel) then FOnNewEditCancel(Self, Node, Action);
        if Action then Node.Destroy
      end else
        if Assigned(FOnEditCancel) then FOnEditCancel(Self, Node);
     end else inherited;
  end;
  FIsEditingNew := False;
end;
```

```
function TBetterTreeView.NewChildAndEdit(Node: TTreeNode;
                                         const S: String): TTreeNode;
begin
  SetFocus;
  Result := Items.AddChild(Node, S);
  FIsEditingNew := True;
  Node.Expand(False);
  Result.EditText;
  SetFocus;
end;

end.
```

[Nomadic]

Динамическое создание компонента TTable

Решение 1

Любой компонент можно создать и без (вне) формы или любого другого дочернего компонента. Для этого в методе Create надо указать параметр nil:

```
FSession := TSession.Create(nil);
FDatabase := TDatabase.Create(nil);
FSession.SessionName := 'DBSession'
FDatabase.Connected := False;
FDatabase.AliasName := Database;
FDatabase.DatabaseName := USER_DATABASE;
FDatabase.SessionName  := FSession.SessionName;
FUserTBL := TTable.Create(nil);
FUserTBL.DatabaseName := FDatabase.DatabaseName;
FUserTBL.SessionName := FSession.SessionName;
FUserTBL.TableName := USERTBL;
FUserTBL.IndexName := USERSpIndex;
FUserSource := TDataSource.Create(nil);
FUserSource.DataSet := FUserTBL;
```

Решение 2

Можно использовать TTable, не размещая компонент на форме:

```
function TForm1.TotalPopulation: double;
var
  Tbl: TTable;
begin
  Result := 0.0;
  Tbl := TTable.Create(nil);
  try
    Tbl.DatabaseName := 'DBDEMOS';
    Tbl.TableName := 'COUNTRY';
    Tbl.Open;
    Tbl.First;
```

```
    while not Tbl.EOF do begin
      Result := Result + Tbl.FieldByName('Population').AsFloat;
      Tbl.Next;
    end;
    Tbl.Close;
  finally
    Tbl.Free;
  end;
end;
```

Динамическое создание файла базы данных

Для создания таблицы программным способом надо сделать приблизительно следующее:

```
with Table1 do begin
  Close;                                          // при необходимости
  TableName := 'test.db';
  TableType := ttParadox;
  DatabaseName := 'Sample';
  FieldDefs.Clear;
  FieldDefs.Add('Field1', ftString, 10, False);
  ...
  IndexDefs.Add('', 'Field1', [ixPrimary]);       { Создание первичного индекса }
  CreateTable;
end;
```

Синхронизация таблицы и StringList

Допустим, у вас есть TTable с именем Table1 и DBGrid с именем DBGrid1:

- В секции модуля interface объявите переменную:

```
FieldLst: TStringList;
```

- Установите свойство формы KeyPreview в True.
- В обработчик события формы OnCreate добавьте:

```
FieldLst := TStringList.Create;
```

- В обработчик события формы OnDestroy добавьте:

```
FieldLst.Free;
```

- В обработчик события формы OnKeyUp добавьте:

```
 if (ssCtrl in Shift) and (Key in [Ord('D'), Ord('d')]) then
   if (FieldLst.Count > 0 ) then begin
{ Если вам необходимы все предыдущие данные полей
  for nFld := 0 to Table1.FieldCount - 1 do
  Table1.Fields[nFld].AsString := FieldLst.Strings[nFld]; }
{ Если вы хотите только поле, с которым сейчас имеете дело }
     DBGrid1.Fields[DBGrid1.SelectedIndex].AsString :=
          FieldLst.Strings[DBGrid1.SelectedIndex];
   end;
```

- Обработчик события таблицы BeforeInsert должен выглядеть следующим образом:

```
procedure TForm1.Table1BeforeInsert(DataSet: TDataset);
var
  nFld: Integer;
  bmPos: TBookMark;
begin
  if (not Table1.BOF) and (Assigned(FieldLst)) then
    try
      bmPos := Table1.GetBookMark;
      Table1.DisableControls;
      Table1.Prior;
      FieldLst.Clear;
      for nFld := 0 to Table1.FieldCount - 1 do
        FieldLst.Add(Table1.Fields[nFld].AsString);
      Table1.GotoBookMark(bmPos);
      Table1.FreeBookMark(bmPos);
      Table1.EnableControls;
    except
      on E: EOutOfMemory do ShowMessage(E.Message);
    end;
end;
```

Обратите внимание, что при обработке события OnKeyUp можно воспользоваться закомментированными строками, которые позволят с помощью комбинации клавиш <Ctrl>+<D> получить все предыдущие данные полей. Убрав этот комментарий, не забудьте закомментировать строку с DBGrid1.

Создание индекса во время выполнения программы

Код обработчика кнопки OnClick, с помощью которого строится индекс:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  bActive, bExclusive: Boolean;
begin
  bActive := Table1.Active;
  bExclusive := Table1.Exclusive;
  Table1.IndexDefs.Update;
  with Table1 do begin
    Close;
{ таблица dBASE должна быть открыта в монопольном (exclusive) режиме }
    Exclusive := True;
    Open;
    if Table1.IndexDefs.IndexOf('FNAME') <> 0 then
      Table1.AddIndex('FNAME', 'FNAME', []);
    Close;
    Exclusive := bExclusive;
    Active := bActive;
  end;
end;
```

Собираясь запускать проект из Delphi, убедитесь в том, что свойство таблицы Active в режиме проектирования установлено в False.

[News Group]

Проверка изменения данных таблицы

В обработчик события формы OnClose поместите следующий код:

```
  ...
if Table1.State in dsEditModes then
  if MessageDlg('Сохранить изменения?', mtInformation,
                [mbYes, mbNo], 0 ) = mrYes then
    Table1.Post
  else
    Table1.Cancel;
  ...
```

Использование DBIOpenLockList

Пример поиска пользователей данной таблицы. Имейте в виду, что свойство TStringList.Duplicate установлено в dupIgnore, поскольку пользователь может иметь более одной блокировки таблицы. При работе с dBase возвращается только блокировка текущего сеанса, тогда как с Paradox функция покажет всех пользователей, получивших доступ к этому же .NET-файлу.

```
procedure GetTableUserList(ATable: TTable; AStringList: TStringList);
var
  hUserCur: hDBICur;
  pUserBuf: pByte;
begin
  AStringList.Clear;
  AStringList.Duplicates := dupIgnore;
  Check(DBIOpenLockList(ATable.Handle, True, True, hUserCur));
  GetMem(pUserBuf, SizeOf(LOCKDesc));
  try
    while (DBIGetNextRecord(hUserCur, dbiNOLOCK, pUserBuf,nil) = DBIERR_NONE) do
      AStringList.Add(StrPas(pLOCKDesc(pUserBuf)^.szUserName))
    finally
      FreeMem(pUserBuf,SizeOf(LOCKDesc));
      DBICloseCursor(hUserCur);
  end;
end;
```

[News Group]

Заполнение DBComboBox и DBListBox

Большинство Delphi-компонентов для работы с базами данных заполняются сами после открытия «прилинкованного» набора данных. Тем не менее,

DbListBox и DbComboBox не поддерживают данную характеристику. Данные два компонента не предназначены для отображения пользовательского набора данных, но, тем не менее, они заполняются на основе полученных данных. При обновлении вашей таблицы значения DbListBox и DbComboBox будут записываться в соответствующее поле.

Заполняются DbComboBox и DbListBox точно так же, как и обычные ComboBox и ListBox. Строки текста в ListBox и ComboBox представляют собой список TString. Данный список у наших компонентов хранится в свойстве Items. Новые строки добавляются к списку посредством метода Add. Если вы хотите использовать другой тип данных, не String, то он должен быть преобразован во время выполнения программы. Если в конце списка имеется пустая строка, рекомендуем установить свойство IntegralHeight в True.

Заполнение DbListBox строками программным путем могло бы выглядеть примерно так:

```
DbListBox1.Items.Add('первая строка');
...
DbListBox1.Items.Add('строка xxx');
```

Заполнение DbListBox во время разработки требует использования Инспектора объектов. Двойной щелчок на свойстве компонента Items позволит вызвать String List Editor (редактор списка строк) и ввести необходимые строки.

К сожалению, если заполнять данным способом ComboBox, значение по умолчанию отсутствует. Данный результат возможен при установке свойства Text компонента DbComboBox (это свойство недоступно в Инспекторе объектов, оно должно быть установлено программным путем). Установка значения по умолчанию для первого элемента в DbComboBox должна выглядеть примерно таким образом:

```
DbComboBox1.Text := DbComboBox1.Items[0];
```

Часто полезным бывает заполнить DBListBox из набора данных. Это можно сделать с помощью цикла:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  with table2 do begin
    Open;
    while not EoF do begin
      DBListBox1.Items.Add(FieldByName('Name').AsString);
      Next;
    end;
  end;
end;
```

Ошибка в DBComboBox или особенность работы?

Форма предназначена для ввода новой информации и работает с двумя таблицами – master и detail. Необходимо, чтобы все компоненты формы (DBEdit, DBComboBox) при открытии отображали пустые строки. Информация из таблицы детализации содержится в DBComboBox. Список создается стандартным способом:

```
DBComboBox.Items.Add(Fields[1].AsString.
```

Если DBComboBox.ItemIndex := -1, то должна отображаться пустая строка. При открытии формы в DBComboBox отображается информация последней записи таблицы детализации. Если заменить в процедуре инициализации списка:

```
DBComboBox.ItemIndex := -1
```

на

```
DBComboBox.Text := ''
```

то появляется другая странность: пользователь раскрывает список и, ничего не выбрав, переходит на другой компонент – в DBComboBox вновь появляется информация последней записи таблицы детализации.

Исправить это поведение DBComboBox можно, если в модуле DBCTRLS.PAS изменить SetComboText:

```
procedure TDBComboBox.SetComboText(const Value: string);
var
  I: Integer;
  Redraw: Boolean;
begin
  if Value <> GetComboText then begin
    if Style <> csDropDown then begin
      Redraw := (Style <> csSimple) and HandleAllocated;
      if Redraw then SendMessage(Handle, WM_SETREDRAW, 0, 0);
      try
        if Value = '' then I := -1 else I := Items.IndexOf(Value);
        ItemIndex := I;
      finally
        if Redraw then begin
          SendMessage(Handle, WM_SETREDRAW, 1, 0);
          Invalidate;
        end;
      end;
      if I >= 0 then Exit;
    end;
    if Style in [csDropDown, csSimple] then Text := Value;
    if ItemIndex = -1 then Text := '';              // добавить
  end;
end;
```

[VS]

Перевод в верхний регистр первого вводимого символа в DBEdit

Решение

```
procedure TForm1.DBEdit1KeyPress(Sender: TObject; var Key: Char);
begin
  if (DBEdit1.SelStart = 0) then Key := UpCase(Key);
end;
```

Исправление DBEdit MaxLength

Не могу получить свойство `MaxLength`*, чтобы работать с компонентами* `TDBEdit`*. В* `TEdit` *это работает как положено, но попытки увеличить максимальную длину текста в* `TDBEdit` *не удаются.*

Дело в том, что в `TDBEdit.DataChange` есть такой код:

```
  ...
if FDataLink.Field <> nil then begin

  ...
  if FDataLink.Field.DataType = ftString then
    MaxLength := FDataLink.Field.Size
  else
    MaxLength := 0;

  ...
  end else begin

  ...
   MaxLength := 0;
  ...
end;
```

иногда значение устанавливается на ноль... Похоже, все будет работать, если изменить строки:

```
MaxLength := 0;
```

на

```
MaxLength := inherited MaxLength;
```

Для того чтобы изменения вступили в силу, необходимо перекомпилировать библиотеку компонентов с измененным DBCTRLS.PAS.

Если вы хотите использовать `MaxLength` со `StringField`, изменений необходимо сделать немного больше:

```
  ...
if (FDataLink.Field.DataType = ftString) and (inherited MaxLength = 0) then
  MaxLength := FDataLink.Field.Size
```

```
else
  MaxLength := inherited MaxLength;
  ...
```

Или использовать что-то типа EditMask...

[News Group]

Поиск и управление TDBEdit/TField

Я хотел бы менять цвет компонентов TDBEdit *и* TEdit*, расположенных на форме, на другой, «отчетливый» цвет, в том случае, если с их помощью требуется ввести какие-либо данные.*

Представляю вашему вниманию два метода. Первый задает цвет каждому DBEdit, имеющему требуемое поле. Второй (более сложный) задает цвет каждому компоненту базы данных, имеющему необходимое поле.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Control: Integer;
begin
  for Control := 0 to ControlCount - 1 do
    if Controls[Control] is TDBEdit then
      with TDBEdit(Controls[Control]) do
        if DataSource.DataSet.FieldByName(DataField).Required then
          Color := clRed;
end;

{ Данный метод будет работать только в случае, если БД-компонент обладает тремя
  полями: DataSource (типа TDataSource), DataField (типа String) и Color (типа
  TColor) - это не должно быть проблемой. Также вам необходимо включить TypInfo в
  список используемых модулей }
procedure TForm1.Button1Click(Sender: TObject);
var
  Control: Integer;
  DataSource: TDataSource;
  DataField: String;

  function GetDataSource(Instance: TComponent): Boolean;
  var
    PropInfo: PPropInfo;
  begin
    Result := False;
    PropInfo := TypInfo.GetPropInfo(Instance.ClassInfo, 'DataSource');
    if (PropInfo <> Nil) and (PropInfo^.PropType^.Kind = tkClass) then begin
      DataSource := TDataSource(TypInfo.GetOrdProp(Instance, PropInfo));
      Result := DataSource <> Nil;
    end;
  end;
```

```
  function GetDataField(Instance: TComponent): Boolean;
  var
    PropInfo: PPropInfo;
  begin
    Result := False;
    PropInfo := TypInfo.GetPropInfo(Instance.ClassInfo, 'DataField');
    if (PropInfo <> Nil) and (PropInfo^.PropType^.Kind = tkString) then begin
      DataField := TypInfo.GetStrProp(Instance, PropInfo);
      Result := True;
    end;
  end;

  procedure SetColor(Instance: TComponent; Color: TColor);
  var
    PropInfo: PPropInfo;
  begin
    PropInfo := TypInfo.GetPropInfo(Instance.ClassInfo, 'Color');
    if (PropInfo <> Nil) and (PropInfo^.PropType^.Kind = tkInteger) then
      TypInfo.SetOrdProp(Instance, PropInfo, Ord(Color));
  end;
begin
  for Control := 0 to ControlCount - 1 do
    if GetDataSource(Controls[Control])
        and GetDataField(Controls[Control])
        and (DataSource.DataSet <> Nil)
        and DataSource.DataSet.FieldByName(DataField ).Required then
      SetColor(Controls[Control], clRed);
end;
```

[News Group]

Insert/Overwrite с помощью DBEdit

Windows не позволяет это сделать, но найден обходной путь.

Сначала добавим к форме свойство:

```
  ...
private
  FinsertMode: boolean;
  procedure SetInsertMode(value: boolean);
public
  property insertMode: boolean read FinsertMode write SetInsertMode;
  ...
```

В обработчике создания события формы инициализируем его (свойство):

```
procedure TForm1.FormCreate(Sender: TObject);
begin
{ инициализация }
  insertMode := True;
end;
```

Также для этого свойства создадим процедуру SetInsertMode, которая с помощью TPanel с именем Panel1 извещает пользователя о текущем режиме работы:

```
procedure TForm1.SetInsertMode(value: boolean);
begin
  FinsertMode := value;
  if FinsertMode then Panel1.Caption := 'ВСТАВКА'
  else Panel1.Caption := 'ЗАМЕНА';
end;
```

Затем добавим три обработчика событий (OnKeyDown, OnKeyPress, OnEnter) для каждого DBEdit (можно при наличии нескольких компонентов создать один общий обработчик для всех):

```
procedure TForm1.DBEditKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if (Key = VK_INSERT) then
  insertMode := not insertMode;
end;

procedure TForm1.DBEditKeyPress(Sender: TObject; var Key: Char);
begin
  if (not insertMode) and (Sender is TDBEdit) then
    (Sender as TDBEdit).SelLength := 1
  else
    (Sender as TDBEdit).SelLength := 0;
end;

procedure TForm1.DBEditEnter(Sender: TObject);
begin
  insertMode := True;
end
```

Все вышесказанное должно также работать без проблем и с компонентами Edit.

[News Group]

Использование опции MultiSelect в DBGrid

Данный пример реализует множественный выбор записей в табличной сетке и отображение второго поля набора данных.

Метод DisableControls применяется для того, чтобы DBGrid не обновлялся во время изменения набора данных. Последняя позиция набора данных сохраняется как Bookmark.

Метод IndexOf вызывается для проверки существования вкладки. Решение о выборе метода IndexOf (или Refresh) должно определяться спецификой приложения.

```
procedure TForm1.SelectClick(Sender: TObject);
var
  x: word;
  TempBookmark: TBookMark;
begin
  DBGrid1.DataSource.DataSet.DisableControls;
  with DBGrid1.SelectedRows do
    if Count <> 0 then begin
      TempBookmark := DBGrid1.DataSource.DataSet.GetBookmark;
        for x := 0 to Count - 1 do begin
          if IndexOf(Items[x]) > -1 then begin
            DBGrid1.DataSource.DataSet.Bookmark := Items[x];
            ShowMessage(DBGrid1.DataSource.DataSet.Fields[1].AsString);
          end;
        end;
    end;
  DBGrid1.DataSource.DataSet.GotoBookmark(TempBookmark);
  DBGrid1.DataSource.DataSet.FreeBookmark(TempBookmark);
  DBGrid1.DataSource.DataSet.EnableControls;
end;
```

[News Group]

Помещение компонентов в DBGrid

Как поместить компоненты в TDBGrid*?*

Данный совет и сопутствующий код показывают, как просто поместить любой компонент в ячейку сетки данных. Компонент в данном контексте может означать любой видимый элемент управления – от простого ComboBox до сложного диалогового окна. Методы, описанные ниже, применимы практически к любому визуальному компоненту. Если его можно поместить на форму, то, вероятно, можно поместить и в ячейку DBGrid.

Здесь нет новых идей, фактически основная технология работы заключается в переносе (интеграции) внешних компонентов в DBGrid. Идея в том, чтобы получить контроль над табличной сеткой. Практически DBGrid состоит из набора компонентов TDBEdit. Вводя данные в ячейку, вы работаете непосредственно с TDBEdit. Остальные (без фокуса) ячейки в данный момент реально являются статичной картинкой. В данном совете рассказывается, как поместить в сфокусированную ячейку другой, отличный от TDBEdit, визуальный компонент

Компонент TDBLookupCombo

Потребуется форма с компонентом DBGrid на ней. Создайте новый проект и поместите на основную форму DBGrid.

Далее поместите на форму TTable, установите псевдоним (Alias) в DBDEMOS, TableName в GRIDDATA.DB и присвойте свойству Active значение True. Поместите DataSource и сошлитесь в свойстве DataSet на Table1. Вернитесь к DBGrid и ука-

жите в свойстве DataSource компонент DataSource1. Данные из GRIDDATA.DB должны появиться в табличной сетке.

Первый элемент, который мы собираемся поместить в DBGrid – TDBLookupCombo, т. к. нам нужна вторая таблица для поиска. Поместите второй TTable на форму. Установите псевдоним (Alias) в DBDEMOS, TableName в CUSTOMER.DB и присвойте свойству Active значение True. Поместите второй DataSource и сошлитесь в свойстве DataSet на Table2.

Теперь нужно поместить компонент TDBLookupCombo из палитры Data Controls на любое место формы – это не имеет никакого значения, т. к. он или будет невидим, или будет нами имплантирован в табличную сетку. Установите свойства компонента DBLookupCombo следующим образом:

```
DataSource      DataSource1
DataField       CustNo
LookupSource    DataSource2
LookupField     CustNo
LookupDisplay   CustNo
{ Вы можете изменить это на Company позже, но сейчас
                            пусть это будет CustNo)
```

Пока мы только настроили компоненты. Теперь давайте создадим код.

Во-первых, необходимо сделать так, чтобы DBLookupCombo, который вы поместили на форму, во время запуска приложения оставался невидимым. Для этого выберите Form1 в Инспекторе объектов, перейдите на вкладку Events (События) и дважды щелкните по событию onCreate. Delphi немедленно создаст и отобразит скелет кода будущего обработчика события onCreate.

Присвойте свойству Visible значение False в DBLookupCombo следующим образом:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  DBLookupCombo1.Visible := False;
end;
```

Теперь надо «прикрутить» компонент к нашей табличной сетке. Наша задача – автоматически отобразить DBLookupCombo в ячейке во время получения фокуса (или при перемещении курсора). Для этого необходимо написать код для обработчиков двух событий: OnDrawDataCell и OnColExit. Первым делом обработаем событие OnDrawDataCell. Дважды щелкните в строчке OnDrawDataCell в Инспекторе объектов и введите следующий код:

```
procedure TForm1.DBGrid1DrawDataCell(Sender: TObject; const Rect: TRect;
                                     Field: TField; State: TGridDrawState);
begin
  if (gdFocused in State) then begin
    if (Field.FieldName = DBLookupCombo1.DataField) then begin
      DBLookupCombo1.Left := Rect.Left + DBGrid1.Left;
      DBLookupCombo1.Top := Rect.Top + DBGrid1.top;
      DBLookupCombo1.Width := Rect.Right - Rect.Left;
```

```
{ DBLookupCombo1.Height := Rect.Bottom - Rect.Top; }
      DBLookupCombo1.Visible := True;
    end;
  end;
end;
```

Причины чрезмерного использования конструкций begin/end скоро станут понятны. В коде «говорится», что если параметр State принимает значение gdFocused, то данная ячейка имеет фокус (в любой момент времени только одна ячейка в табличной сетке может иметь фокус). Далее, если это выделенная ячейка, которая имеет то же имя поля, как и поле данных DBLookupCombo, то DBLookupCombo необходимо поместить над этой ячейкой и сделать его видимым. Обратите внимание на определение позиции DBLookupCombo. Она определяется относительно формы, а не ячейки. Так, положение левой стороны LookupCombo должно учитывать положение сетки (DBGrid1.Left) плюс положение соответствующей ячейки относительно сетки (Rect.Left).

Также обратите внимание на то, что определение высоты LookupCombo в коде закомментировано. Причина в том, что LookupCombo имеет минимальную высоту. Вы просто не сможете сделать ее меньше. Минимальная высота LookupCombo больше высоты ячейки. Если раскомментировать строку, касающуюся высоты LookupCombo, то данный код изменит размер компонента и Delphi немедленно его перерисует. Это вызовет неприятное моргание компонента. Бороться с этим невозможно. Пусть LookupCombo будет немного больше, чем ячейка. Это выглядит немного странным, но это работает.

Теперь запустите программу. Заработала? Сразу после запуска переместите курсор на одну из ячеек табличной сетки. Вы ожидали чего-то большего? Да! Мы только в середине пути. Теперь нам нужно спрятать LookupCombo при выходе курсора за пределы колонки. Напишем обработчик события OnColExit. Это должно выглядеть примерно так:

```
procedure TForm1.DBGrid1ColExit(Sender: TObject);
begin
  if DBGrid1.SelectedField.FieldName = DBLookupCombo1.DataField then
    DBLookupCombo1.Visible := false;
end;
```

Код ассоциирует имя поля ячейки (FieldName) с нашим LookupCombo при помощи свойства TDBGrids.SelectedField. Код «говорит»: «Если ячейка была в колонке с DBLookupCombo (имя поля ячейки совпадает с именем поля DBLookupCombo), его необходимо сделать невидимым».

Теперь снова запустите приложение. Чувствуете эффект?

Теперь вроде бы все правильно, но мы забыли об одной вещи. Попробуйте ввести новое значение в одно из LookupCombo. Проблема в том, что нажатие клавиши обрабатывает DBGrid, а не LookupCombo. Чтобы исправить это, нам нужно написать для табличной сетки обработчик события onKeyPress. Это должно выглядеть примерно так:

```
procedure TForm1.DBGrid1KeyPress(Sender: TObject; var Key: Char);
begin
  if (key <> Chr(9)) then begin
    if (DBGrid1.SelectedField.FieldName = DBLookupCombo1.DataField) then begin
      DBLookupCombo1.SetFocus;
      SendMessage(DBLookupCombo1.Handle, WM_Char, word(Key), 0);
    end;
  end;
end;
```

В данном коде «говорится», что если нажатая клавиша не является клавишей <Tab> (Chr(9)), а текущее поле в табличной сетке – LookupCombo, то надо установить фокус на LookupCombo и передать сообщение с кодом нажатой клавиши LookupCombo. Здесь используется функция Windows API. Нам не обязательно знать, как это работает, достаточно того, что это просто работает.

Небольшое пояснение. Для того чтобы функция Windows SendMessage послала сообщение «куда надо», ей в качестве параметра необходим дескриптор («адрес») нужного компонента. Используйте свойство компонента Handle. Затем нужно сообщить компоненту, что мы от него хотим. В нашем случае это Windows-сообщение WM_CHAR, извещающее LookupCombo о том, что ему посылается символ. Наконец, мы передаем ему сам символ нажатой клавиши – Word(Key), выполняя приведение к типу Word параметра Key события нажатия клавиши. Все достаточно просто. Все, что вам действительно необходимо сделать, – это заменить имя DBLookupCombo1 нашего вымышленного компонента на имя реального компонента, который будет участвовать в «модернизации» табличной сетки. Более подробную информацию о функции SendMessage можно почерпнуть из электронной справки, поставляемой вместе с Delphi.

Запустите приложение снова и попробуйте что-нибудь ввести. Это работает! Экспериментируя, вы увидите, что с помощью клавиши <Tab> можно перейти из режима редактирования в режим перемещения курсора и наоборот.

Компонент TDBCombo

Здесь не будет обсуждаться технология имплантации DBCombo, т. к. как она практически не отличается от той, что была показана выше. Все написанное выше имеет силу и здесь.

Пошаговая разработка кода для вашего компонента:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  DBLookupCombo1.Visible := False;
  DBComboBox1.Visible := False;
end;

procedure TForm1.DBGrid1DrawDataCell(Sender: TObject; const Rect: TRect;
                                     Field: TField; State: TGridDrawState);
begin
  if (gdFocused in State) then begin
```

```
    if (Field.FieldName = DBLookupCombo1.DataField) then begin
      DBLookupCombo1.Left := Rect.Left + DBGrid1.Left;
      DBLookupCombo1.Top := Rect.Top + DBGrid1.top;
      DBLookupCombo1.Width := Rect.Right - Rect.Left;
      DBLookupCombo1.Visible := True;
    end else if (Field.FieldName = DBComboBox1.DataField) then begin
      DBComboBox1.Left := Rect.Left + DBGrid1.Left;
      DBComboBox1.Top := Rect.Top + DBGrid1.top;
      DBComboBox1.Width := Rect.Right - Rect.Left;
      DBComboBox1.Visible := True;
    end;
  end;
end;

procedure TForm1.DBGrid1ColExit(Sender: TObject);
begin
  if DBGrid1.SelectedField.FieldName = DBLookupCombo1.DataField then
    DBLookupCombo1.Visible := False
  else if DBGrid1.SelectedField.FieldName = DBComboBox1.DataField then
    DBComboBox1.Visible := false;
end;

procedure TForm1.DBGrid1KeyPress(Sender: TObject; var Key: Char);
begin
  if (key <> chr(9)) then begin
    if (DBGrid1.SelectedField.FieldName = DBLookupCombo1.DataField) then begin
      DBLookupCombo1.SetFocus;
      SendMessage(DBLookupCombo1.Handle, WM_Char, word(Key), 0);
    end

    else if (DBGrid1.SelectedField.FieldName = DBComboBox1.DataField) then begin
      DBComboBox1.SetFocus;
      SendMessage(DBComboBox1.Handle, WM_Char, word(Key), 0);
    end;
  end;
end;
```

Компонент TDBCheckBox

Технология работы с компонентом DBCheckBox более интересна. В этом случае нам необходимо дать пользователю понять о наличии компонента DBCheckBox в ячейках без фокуса. Можно вставлять статическое изображение компонента или динамически изменять изображение в зависимости от логического состояния элемента управления. Обсудим второй вариант. Создадим два BMP-файла: включенный (TRUE.BMP) и выключенный (FALSE.BMP) DBCheckBox. Поместите два компонента TImage на форму, присвойте им имена ImageTrue и ImageFalse и назначьте соответствующие BMP-файлы в свойстве Picture. Также необходимо поместить на форму два компонента DBCheckBox. Установите набор данных обоих компонентов в DataSource1 и присвойте свойству Color значение clWindow. Для начала создадим для формы обработчик события OnCreate:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  DBLookupCombo1.Visible := False;
  DBCheckBox1.Visible := False;
  DBComboBox1.Visible := False;
  ImageTrue.Visible := False;
  ImageFalse.Visible := False;
end;
```

Теперь нам нужен обработчик события OnDrawDataCell, чтобы делать что-то с ячейками, не имеющими фокуса. Здесь подойдет следующий код:

```
procedure TForm1.DBGrid1DrawDataCell(Sender: TObject; const Rect: TRect;
                                     Field: TField; State: TGridDrawState);
begin
  if (gdFocused in State) then begin
    if (Field.FieldName = DBLookupCombo1.DataField) then begin
...
// смотри выше
...
    end else if (Field.FieldName = DBCheckBox1.DataField) then begin
      DBCheckBox1.Left := Rect.Left + DBGrid1.Left + 1;
      DBCheckBox1.Top := Rect.Top + DBGrid1.top + 1;
      DBCheckBox1.Width := Rect.Right - Rect.Left{ - 1};
      DBCheckBox1.Height := Rect.Bottom - Rect.Top{ - 1};
      DBCheckBox1.Visible := True;
    end else if (Field.FieldName = DBComboBox1.DataField) then begin
...
// смотри выше
...
    end;
  end else begin
{ в этом месте помещаем статическое изображение компонента }
    if (Field.FieldName = DBCheckBox1.DataField) then begin
      if TableGridDataCheckBox.AsBoolean then
        DBGrid1.Canvas.Draw(Rect.Left,Rect.Top,ImageTrue.Picture.Bitmap)
      else
        DBGrid1.Canvas.Draw(Rect.Left,Rect.Top,ImageFalse.Picture.Bitmap)
    end;
  end;
end;
```

Самое интересное место – последний участок кода. Он выполняется в случае, когда состояние не равно gdFocused и сам CheckBox находится в колонке. В нем осуществляется проверка данных поля: если они равны True, то выводится рисунок TRUE.BMP, в противном случае – FALSE.BMP. Предварительно созданы два изображения, представляющие собой «слепок» двух логических состояний компонента, теперь будет не очень трудно обнаружить отсутствие компонента в ячейках с фокусом и без оного. Теперь напишем обработчик события OnColExit:

```
procedure TForm1.DBGrid1ColExit(Sender: TObject);
begin
  if DBGrid1.SelectedField.FieldName = DBLookupCombo1.DataField then
    DBLookupCombo1.Visible := False
  else If DBGrid1.SelectedField.FieldName = DBCheckBox1.DataField then
    DBCheckBox1.Visible := False
  else If DBGrid1.SelectedField.FieldName = DBComboBox1.DataField then
    DBComboBox1.Visible := False;
end;
```

Организуйте обработку события OnKeyPress,,0 как показано ниже:

```
procedure TForm1.DBGrid1KeyPress(Sender: TObject; var Key: Char);
begin
  if (key <> chr(9)) then begin
    if (DBGrid1.SelectedField.FieldName = DBLookupCombo1.DataField) then begin
      DBLookupCombo1.SetFocus;
      SendMessage(DBLookupCombo1.Handle, WM_Char, word(Key), 0);
    end
    else if (DBGrid1.SelectedField.FieldName = DBCheckBox1.DataField) then begin
      DBCheckBox1.SetFocus; SendMessage(DBCheckBox1.Handle, WM_Char, word(Key), 0);
    end
    else if (DBGrid1.SelectedField.FieldName = DBComboBox1.DataField) then begin
      DBComboBox1.SetFocus; SendMessage(DBComboBox1.Handle, WM_Char, word(Key), 0);
    end;
  end;
end;
```

Наконец, последняя хитрость. Для удобства пользователя заголовку компонента нужно присвоить текущее логическое значение. С самого начала у меня была идея поручить это обработчику события OnChange, но проблема в том, что событие может возникнуть не один раз. Поэтому снова воспользуемся функцией Windows API и пошлем компоненту соответствующее значение, вызвав функцию SendMessage(DBCheckBox1.Handle, BM_GetCheck, 0, 0), которая возвращает 0 в случае, если компонент не включен, и любое другое число в противном случае.

```
procedure TForm1.DBCheckBox1Click(Sender: TObject);

begin
  if SendMessage(DBCheckBox1.Handle, BM_GetCheck, 0, 0) = 0 then
    DBCheckBox1.Caption := ' ' + 'Ложь'
  else
    DBCheckBox1.Caption := ' ' + 'Истина'
end;
```

Ревизия

Вышеприведенный код оказался несвободным от недостатков – после первой публикации совета дотошные программисты обнаружили две значительные недоработки.

Первая проблема – если компонент имеет фокус, то для перемещения на следующую ячейку необходимо двукратное нажатие клавиши <Tab>. Вторая проблема возникает при добавлении новой записи.

Проблема 1. Необходимость двойного нажатия клавиши <Tab>

Действительно, компонент, используемый для «имплантации», существует сам по себе, а не является частью табличной сетки. В случае, когда DBGrid имеет фокус, то для перемещения на следующую ячейку необходимо двукратное нажатие клавиши <Tab>. Первое нажатие клавиши перемещает фокус из имплантированного компонента на текущую ячейку, находящуюся под этим компонентом, и только второе нажатие клавиши <Tab> перемещает нас в следующую ячейку. Попробуем это исправить.

Для начала в форме, содержащей табличную сетку, опишем логическую переменную WasInFloater следующим образом:

```
type
  TForm1 = class(TForm)
  ...
private
  WasInFloater : Boolean;
  ...
end;
```

Затем для компонента LookupCombo напишем обработчик события OnEnter, где присвоим переменной WasInFloater значение True. Это позволит нам понять, где в данный момент находится фокус.

```
procedure TForm1.DBLookupCombo1Enter(Sender: TObject);
begin
  WasInFloater := True;
end;
```

И, наконец, создаем хитрый обработчик события OnKeyUp, позволяющий исправить досадный недостаток.

```
procedure TForm1.DBGrid1KeyUp(Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if (Key in [VK_TAB]) and WasInFloater then begin
    SendMessage(DBGrid1.Handle, WM_KeyDown, Key, 0);
    WasInFloater := False;
  end;
end;
```

Данный код реагирует на нажатие клавиши и позволяет в случае, когда фокус передался из имплантированного элемента управления табличной сетке, вторично эмулировать нажатие клавиши <Tab> (передается код нажатой клавиши, т. е. <Tab>). Это работает как для отдельной клавиши <Tab>, так и для комбинации <Shift>+<Tab>.

Проблема 2. Новая запись исчезает, когда компонент получает фокус

Если нажать в навигаторе кнопку добавить, запись добавляется, но если потом щелкнуть по одному из компонентов, имплантированных в табличную сетку, новая запись таинственным образом исчезнет. Причина этого – флаг dgCancelOnExit в опциях DBGrid, который имеет значение True по умолчанию. Установите его в False, и вышеназванная проблема исчезает.

Наверное, в Borland поступили неправильно, установив такое значение по умолчанию, флаг должен иметь значение False. Данная опция действует в случае потери компонентом фокуса и отменяет последние результаты редактирования. Во всяком случае, первым делом сбрасывайте данный флаг.

Сортировка колонок в DBGrid

Многие профессиональные приложения отображают данные в полях табличной сетки и разрешают сортировать любую колонку, просто щелкая по ее заголовку. То, что здесь изложено, – не наилучший путь для решения задачи, данная технология не что иное, как простая имитация такого поведения компонента.

Главное препятствие в решении задачи – сам DBGrid. Проблема в отсутствии событий OnClick или OnMouseDown, позволяющих реагировать на элементарные манипуляции с заголовком. Правда, существует событие OnDoubleClick, но для данной цели оно не слишком изящно. Нам нужно лишь создать заголовок, реагирующий на однократный щелчок мышью. Обратимся к компоненту THeaderControl.

Этот компонент, введенный в палитру компонентов еще в Delphi 2.0, и обеспечивает необходимые нам функции. Главное достоинство – реакция компонента при щелчке по отдельным панелям. Панели также обеспечивают визуальное отображение подобно кнопке (могут вдавливаться и отжиматься). Нам необходимо «прицепить» THeaderControl к DBGrid. Вот как это сделать:

Во-первых, создайте новое приложение. Положите THeaderControl на форму. Он автоматически выровняется по верхнему краю формы. Затем поместите на форму DBGrid и присвойте свойству Align значение alClient. Затем добавьте компоненты TTable и TDataSource. В компоненте TTable присвойте свойству DatabaseName значение DBDEMOS, а свойству TableName – значение EVENTS.DB. В TDataSource укажите в свойстве DataSet на компонент Table1, а в TDBGrid в свойстве DataSource на DataSource1. Если свойство Active компонента TTable равно True, выключите его (значение False).

Сделаем так, чтобы компонент THeaderControl выглядел похожим на заголовок компонента DBGrid. Произведем необходимые манипуляции в момент создания формы. Дважды щелкните на событии OnCreate формы и введите следующий код:

```
procedure TForm1.FormCreate(Sender: TObject);
var
  TheCap: String;
```

```
    TheWidth, a: Integer;
  begin
    DBGrid1.Options := DBGrid1.Options - [dgTitles];
    HeaderControl1.Sections.Add;
    HeaderControl1.Sections.Items[0].Width := 12;
    Table1.Active := False;
    Table1.Exclusive := True;
    Table1.Active := True;
    for a := 1 to DBGrid1.Columns.Count do begin
      with DBGrid1.Columns.Items[a - 1] do begin
        TheCap := Title.Caption;
        TheWidth := Width;
      end;
      with HeaderControl1.Sections do begin
        Add;
        Items[a].Text := TheCap;
        Items[a].Width := TheWidth + 1;
        Items[a].MinWidth := TheWidth + 1;
        Items[a].MaxWidth := TheWidth + 1;
      end;
      try
        Table1.AddIndex(TheCap, TheCap, []);
      except
        HeaderControl1.Sections.Items[a].AllowClick := False;
      end;
    end;
    Table1.Active := False;
    Table1.Exclusive := False;
    Table1.Active := True;
  end;
```

После того как `THeaderControl` заменил стандартный заголовок `DBGrid`, в первую очередь мы сбрасываем (устанавливаем в `False`) флаг `dgTitles` в свойстве `Options` компонента `DBGrid`. Затем мы добавляем колонку в `HeaderControl` и устанавливаем ее ширину равной 12. Это будет пустой колонкой, которая имеет ту же ширину, что и левая колонка статуса в `DBGrid`.

Затем нужно убедиться, что таблица открыта для эксклюзивного доступа (никакие другие пользователи работать с ней не могут). Причина будет объяснена немного позже.

Теперь добавляем секции в `HeaderControl`. Для каждой добавленной колонки мы создаем в заголовке тот же текст, что и в соответствующей колонке `DBGrid`. В цикле мы проходим по всем колонкам `DBGrid` и повторяем текст заголовка колонки и его высоту. Мы также устанавливаем для `HeaderControl` значения свойств `MinWidth` и `MaxWidth` равным ширине соответствующей колонки в `DBGrid`. Это предохранит колонки от изменения их ширины. Для изменяющих размер колонок нужно дополнительное кодирование. Этот вопрос читателю предлагается решить самостоятельно.

Теперь самое интересное. Мы собираемся создать индекс для каждой колонки в DBGrid. Имя индекса будет таким же, как и название колонки. Данный код мы должны заключить в конструкцию try..finally, поскольку существуют некоторые поля, которые не могут быть проиндексированы (например, поля BLOB и Memo). При попытке индексации этих полей генерируется исключительная ситуация. Мы перехватываем это исключение и не допускаем возможности щелчка в данной колонке. Это означает, что колонки, содержащие неиндексированные поля, не будут реагировать на щелчок мышью. Создание этих индексов объясняет, почему таблица должна быть открыта в режиме эксклюзивного (монопольного) доступа. И в заключение мы закрываем таблицу, сбрасываем флаг эксклюзивности и снова делаем таблицу активной.

Последний шаг. При щелчке на HeaderControl нам необходимо включить правильный индекс таблицы. Создадим обработчик события OnSectionClick компонента HeaderControl как показано ниже:

```
procedure TForm1.HeaderControl1SectionClick(HeaderControl: THeaderControl;
                                            Section: THeaderSection);
begin
  Table1.IndexName := Section.Text;
end;
```

После щелчка в заголовке колонки значение свойства таблицы IndexName становится равным заголовку компонента HeaderControl.

Просто и красиво. Тем не менее, есть масса мест, требующих улучшения. Например, вторичный щелчок должен возобновлять порядок сортировки. Или возможность изменения размера самих колонок.

Улучшения

Здесь приведен код, улучшенный по сравнению с предыдущей версией «Совета». Суть заключается в том, что в качестве имени индекса используется имя поля (а не заголовка).

Это улучшает гибкость.

```
procedure TForm1.FormCreate(Sender: TObject);
var
  TheCap: String;
  TheFn: String;
  TheWidth: Integer;
  a: Integer;
begin
  Table1.Active := True;
  DBGrid1.Options := DBGrid1.Options - [DGTitles];
  HeaderControl1.Sections.Add;
  HeaderControl1.Sections.Items[0].Width := 12;
  for a := 1 to DBGrid1.Columns.Count do begin
    with DBGrid1.Columns.Items[a - 1] do begin
      TheFn := FieldName;
```

```
        TheCap := Title.Caption;
        TheWidth := Width;
      end;
      with Headercontrol1.Sections do begin
        Add;
        Items[a].Text     := TheCap;
        Items[a].Width    := TheWidth + 1;
        Items[a].MinWidth := TheWidth + 1;
        Items[a].MaxWidth := TheWidth + 1;
      end;
      try
{ Используем индексы с тем же именем, что и имя поля }
{ Пробуем задать имя индекса }
       (DataSource1.Dataset as TTable).IndexName := TheFn;
      except
        HeaderControl1.Sections.Items[a].AllowClick := False; { Индекс недоступен }
      end;
    end;
end;
```

Используем свойство `FieldName` компонента `DBGrid` для задания индекса с тем же именем, что и имя поля.

```
procedure TfrmDoc.HeaderControl1SectionClick(HeaderControl: THeaderControl;
                                             Section: THeaderSection);
begin
  (DataSource1.Dataset as TTable).IndexName :=
         DBGrid1.Columns.Items[Section.Index - 1].FieldName;
end;
```

Примечание

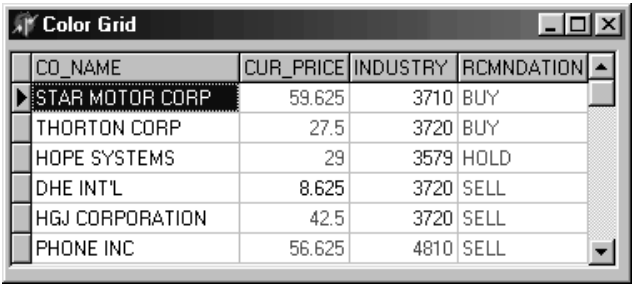
Работу этой программы можно еще улучшить, если предусмотреть реакцию на изменение размеров формы и работу с полосами прокрутки.

DBGrid с цветными ячейками

Есть ли какой-либо способ придать ячейке `DBGrid` *другой цвет? Мне хотелось бы выделить отдельные ячейки строки по определенному признаку. Типа флага, который, если счет просрочен свыше 90 дней, делает строчку красной.*

Решение 1

На рисунке показано, как изменить цвет отдельных ячеек `GBGrid` без создания нового компонента.



Создайте форму, поместите на нее компонент TTable и укажите ему на таблицу EMPLOYEE.DB в базе данных DBDEMOS. Затем разместите на форме DataSource и DBGrid, «соедините» их и вы получите «живые» данные.

Для демонстрации данной технологии выбрано поле Номер служащего в таблице EMPLOYEE.DB «покрашены» ячейки с нечетными числами. То есть, если число нечетное, красим ячейку в зеленый цвет.

Единственный код расположился в обработчике события OnDrawColumnCell компонента DBGrid и выглядел он так:

```
procedure TForm1.DBGrid1DrawColumnCell(Sender: TObject; const Rect: TRect;
                        DataCol: Integer; Column: TColumn; State: TGridDrawState);
var
  holdColor: TColor;
begin
  holdColor := DBGrid1.Canvas.Brush.Color;          { сохраняем оригинальный цвет }
  if Column.FieldName = 'EmpNo' then

{ "раскрашиваем" ячейки только для поля EmpNo }
    if (Column.Field.AsInteger mod 2 <> 0) then begin
      DBGrid1.Canvas.Brush.Color := clGreen;
      DBGrid1.DefaultDrawColumnCell(Rect, DataCol, Column, State);
      DBGrid1.Canvas.Brush.Color := holdColor;
    end;
end;
```

В данном случае вызывается метод DefaultDrawColumnCell компонента TCustomDBGrid, являющийся родителем для TDBGrid. Он раскрашивает зеленым цветом нечетные ячейки поля EmpNo.

Решение 2

Обработайте событие OnDrawDataCell. Вот пример, который использует демонстрационную таблицу COUNTRY и выводит текст красным цветом во всех строках, содержащих страны с населением свыше 10 миллионов человек:

```
procedure TForm1.DBGrid1DrawDataCell(Sender: TObject; const Rect: TRect;
  Field: TField; State: TGridDrawState);
begin
  if Table1.FieldByName('Population').AsFloat > 10000000 then
```

```
    DBGrid1.Canvas.Font.Color := clRed;
  dbGrid1.DefaultDrawDataCell(Rect, Field, State);
end;
```

Примечание

Borland не рекомендует использовать в новых разработках обработчик события `OnDrawDataCell`*, которому пришел на смену обработчик* `OnDrawColumnCell`*. Старый вызов сохранен для совместимости.*

Отображение графики в DBGrid

`DBGrid`, который позволяет выводить изображения:

```
unit DBPicGrd;

interface

uses
  Windows, DBGrids, DB, DBTables, Grids, Classes, Graphics;

type
  TDBPicGrid = class(TDBGrid)
  protected
    procedure DrawDataCell(const Rect: TRect; Field: TField;
                           State: TGridDrawState); override;
  public
    constructor Create(AOwner: TComponent); override;
  published
    property DefaultDrawing default False;
end;

procedure Register;

implementation

constructor TDBPicGrid.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  DefaultDrawing := False;
end;

procedure TDBPicGrid.DrawDataCell(const Rect: TRect; Field: TField;
                                  State: TGridDrawState);
var
  bmp: TBitmap;
begin
  with Canvas do begin
    FillRect(Rect);
    if Field is TGraphicField then
```

```
        try
          bmp := TBitmap.Create;
          bmp.Assign(Field);
          Draw(Rect.Left, Rect.Top, bmp);
        finally
          bmp.Free;
        end
      else
        TextOut(Rect.Left, Rect.Top, Field.Text);
    end;
  end;

  procedure Register;
  begin
    RegisterComponents('Custom', [TDBPicGrid]);
  end;
  end.
```

Пример формы запроса на Delphi

Встроенные инструменты обработки данных в Delphi делают эту задачу намного труднее, чем, например, в таких ресурсоемких инструментальных средствах, как Oracle Forms (Формы Oracle). Данный модуль не такой мощный, как встроенные QBF-возможности форм Oracle, но он заполняет значительную брешь в функциональности Delphi.

Имейте в виду, что модуль охраняется авторским правом, как требует того лицензия, и желательно, чтобы вы сохранили полностью весь код данного модуля. Имейте также в виду, что авторские условия допускают безвозмездное использование данного модуля для любых целей.

Данный модуль обеспечивает простую, но эффективную форму запроса для доступа приложений к базам данных средствами Borland Delphi. Данный модуль также располагает сервисом Sort By Form (Форма сортировки).

Форма запроса отображает модальное диалоговое окно с компонентом `StringGrid`, содержащим искомые поля, полученные при вызове `DbGrid`. Пользователь может ввести точную величину поиска для любого количества полей и применить функцию `Drag & Drop` для изменения порядка сортировки полей (только тех полей, которые содержат искомые величины, влияющие на сортировку).

Когда пользователь нажимает в диалоговом окне кнопку `OK`, данный модуль модифицирует значение свойства `IndexFieldNames` компонента `DbGrid`, применяет диапазон поиска (точные величины) и обновляет данные.

В случае, если пользователь не указывает ни одну из величин поиска, данный модуль очищает значение свойства `IndexFieldNames` компонента `DbGrid`, очищает диапазон поиска и обновляет данные.

Сервис Sort By Form работает аналогично, за исключением того, что не принимает в расчет величину поиска, введенную пользователем. Последний для

установления порядка сортировки прибегает к функции Drag & Drop и затем нажимает кнопку OK. Данный модуль модифицирует значение свойства IndexFieldNames компонента DbGrid, очищает диапазон поиска и обновляет данные.

Создайте соответствующую форму диалога, используя меню File\New\Other..\Dialogs и выбрав пункт Standard Dialog. Разместите на форме компонент StringGrid (Вы найдете его в палитре компонентов Additional). Установите следующие размеры StringGrid: высота 161 и ширина 305. И, наконец, замените исходный код новой формы (.PAS-файл) данным модулем.

```
unit Db_QBF;
{ Все права защищены. Автор Rick Rutt.
  Данный модуль может без какой-либо оплаты быть использован в программе,
  скопирован или распространен любым человеком и для любой цели, если все копии
  данного модуля сохраняют это авторское уведомление.
  Автор предоставляет разрешение каждому для создания производного кода, если
  каждая производная работа содержит авторское уведомление и строку "Части данной
  работы основываются на Db_QBF.PAS, созданным Rick Rutt." }

interface

uses
  WinTypes, WinProcs, Classes, Graphics, Forms, Controls, Buttons,
  StdCtrls, ExtCtrls, Grids, DBGrids;

{ Следующие две процедуры обеспечивают механизм доступа сервисов данного модуля.
  Кнопка (или пункт меню) вызывают процедуру, передавая ей в качестве аргумента
  DbGrid. (Не забудьте добавить строку "uses Db_QBF;" в секцию реализации модуля
  вызова форм.) Ограничение: компонент DbGrid должен ссылаться на DataSource,
  который, в свою очередь, ссылается на DataSet, работающий с таблицой. Данный
  модуль не поддерживает запрос напрямую к DataSet ввиду отсутствия свойства
  IndexFieldNames. }
procedure QueryByForm(grid: TDbGrid);
procedure SortByForm(grid: TDbGrid);
{ Следующая секция управляется средой Delphi. }
type
  TdlgQBF = class(TForm)
    OKBtn: TBitBtn;
    CancelBtn: TBitBtn;
    HelpBtn: TBitBtn;
    gridQBF: TStringGrid;
    procedure OKBtnClick(Sender: TObject);
    procedure CancelBtnClick(Sender: TObject);
end;
var
  dlgQBF: TdlgQBF;

implementation
{ Следующая секция пишется программистом с помощью среды Delphi. }
uses
  Dialogs, Db, DbTables;
```

```
{$R *.DFM}

const
  qbfRowHeight = 16;
  qbfColWidth = 150;
  qbfFieldLabel = '<<Поле>>';
  qbfValueLabel = '<<Значение>>';
  qbfQueryCaption = 'Запрос для таблицы ';
  qbfSortCaption = 'Порядок сортировки для таблицы ';

var
{ Объявим некоторые элементы управления, участвующие в QBF-диалоге при нажатии
  кнопки OK. }
  CallingGrid: TDbGrid;
  CallingMode: (modeQuery, modeSort);
{ Инициализация формы, обеспечивающей визуализацию работы двух объявленных выше
  процедур }

procedure SetupAndShowForm;
var
  i, j, n: integer;
  tbl: TTable;
  f: TField;
begin
  n := CallingGrid.FieldCount;
  if n <= 0 then begin
 { Вместо вывода сообщений могут генерироваться исключительные ситуации }
    MessageDlg('При обращении к DbGrid, модуль Db_QBF не обнаружил полей',
                mtWarning, [mbOK], 0);
  end else if CallingGrid.DataSource = NIL then begin
    MessageDlg('При обращении к DbGrid, модуль Db_QBF не обнаружил ссылки на
                DataSource', mtWarning, [mbOK], 0);
  end else if CallingGrid.DataSource.DataSet = NIL then begin
    MessageDlg('При обращении к DbGrid, модуль Db_QBF обнаружил подключенный
                DataSource без ссылки на DataSet', mtWarning, [mbOK], 0);
  end else if not (CallingGrid.DataSource.DataSet is TTable) then begin
    MessageDlg('При обращении к DbGrid, модуль Db_QBF обнаружил подключенный
                DataSource с сылкой на DataSet, не являющийся таблицей.',
                mtWarning, [mbOK], 0);
  end else with dlgQBF.gridQBF do begin
{ Данные свойства могут быть изменены и в режиме проектирования }
    DefaultRowHeight := qbfRowHeight;
    Scrollbars := ssVertical;
    ColCount := 2; { Для режима сортировки необходимы две пустые колонки }
{ Данные свойства должны быть установлены во время выполнения программы }
    RowCount := Succ(n);
    Cells[0, 0] := qbfFieldLabel;
    Options := Options + [goRowMoving];
    tbl := TTable(CallingGrid.DataSource.DataSet);
    if CallingMode = modeQuery then begin
      dlgQBF.Caption := qbfQueryCaption + tbl.TableName;
```

```
      Cells[1, 0] := qbfValueLabel;
      Options := Options + [goEditing]; { Разрешаем пользователю ввести значение }
      DefaultColWidth := qbfColWidth;
    end else begin
      dlgQBF.Caption := qbfSortCaption + tbl.TableName;
      Cells[1, 0] := '';   { Ввод "пустышки" для первой, нефункциональной колонки }
      Options := Options - [goEditing];      { Убираем возможность редактирования }
{ Этим трюком мы помещаем две пустые секции над одной колонкой }
      DefaultColWidth := (2 * qbfColWidth);
    end;
    j := 0;                  { Фактическое число полей, показываемое пользователю }
    for i := 1 to n do begin
      f := CallingGrid.Fields[Pred(i)];
    if f.DataType in [ftBLOB, ftBytes, ftGraphic, ftMemo,ftUnknown, ftVarBytes] then
        RowCount := Pred(RowCount)             { Игнорируем неиндексируемые поля }
      else begin
        Inc(j);
        Cells[0, j] := f.FieldName;
        Cells[1, j] := '';                     { Сбрасываем искомую величину }
      end;
    end;
    dlgQBF.HelpBtn.Visible := False;           { Помощь, понятно, отсутствует... }
    dlgQBF.ShowModal;
  end;
end;

procedure QueryByForm(Grid: TDbGrid);
begin
  CallingGrid := Grid;     { Сохраняем для использования при нажатии на кнопку OK }
  CallingMode := modeQuery; SetupAndShowForm;
end;

procedure SortByForm(Grid: TDbGrid);
begin
  CallingGrid := Grid;     { Сохраняем для использования при нажатии на кнопку OK }
  CallingMode := modeSort;
  SetupAndShowForm;
end;

procedure TdlgQBF.CancelBtnClick(Sender: TObject);
begin
{ Просто прячем диалог, не делая никаких изменений в вызывающем Grid'е. }
  dlgQBF.Hide;
end;
procedure TdlgQBF.OKBtnClick(Sender: TObject);
var
  flds, sep, val: string;
  i, n, nfld: integer;
begin
  flds := '';           { Список полей, разделенных ';'. }
  sep := '';            { Разделитель ';' ставится после добавления первого поля. }
```

```
    { Количество полей в списке. }
    nfld := 0;
    with dlgQBF.gridQBF do begin
      n := Pred(RowCount);
      if n > 0 then for i := 1 to n do begin

      { Значение поиска, введенное пользователем (если имеется) }
      val := Cells[1, i];
        if (CallingMode = modeSort) or (val <> '') then begin
          flds := flds + sep + Cells[0, i];
          sep := ';';
          nfld := Succ(nfld);
        end;
      end;
      with CallingGrid.DataSource.DataSet as TTable do begin
        IndexFieldNames := flds;
        if (CallingMode = modeSort) or (flds = '') then begin
          CancelRange;
        end else begin
          SetRangeStart;
          for i := 1 to n do begin
            val := Cells[1, i];
            if val <> '' then begin
              FieldByName(Cells[0, i]).AsString := val;
            end;
          end;

{ Устанавливаем конец диапазона так, чтобы он соответствовал его началу }
          SetRangeEnd;
          for i := 1 to n do begin
            val := Cells[1, i];
            if val <> '' then begin
              FieldByName(Cells[0, i]).AsString := val;
            end;
          end;
          ApplyRange;
        end;
        Refresh;
      end;
    end;
    dlgQBF.Hide;
  end;
  end.
```

Примечание

При работе с этим примером необходимо помнить, что он создавался для Delphi 1, поэтому расположение и наполнение стандартных форм диалога может быть другим.

Изменение размеров DBGrid

На форме расположены: поле редактирования, компонент TQuery, DBGrid *и кнопка. Я заполняю поле редактирования и, при нажатии кнопки, в DBGrid отображается результат запроса. Как изменить размер табличной сетки и ее колонок в зависимости от новых значений полей? Поля, возвращаемые запросом, не заполняют всей ширины сетки, а все мои попытки сделать это из кода терпят крах...*

Можно изменить размер колонки во время выполнения программы, изменяя свойство DisplayWidth соответствующего поля компонента DBGrid:

```
MyTableMyField.DisplayWidth := Length(MyTableMyField.Value);
```

Если вам действительно необходимо вычислить ширину всего DBGrid, используйте следующий код:

```
function NewTextWidth(fntFont: TFont; const sString: OpenString): integer;
var
  fntSave: TFont;
begin
  result := 0;
  fntSave := Application.MainForm.Font;
  Application.MainForm.Font := fntFont;
  try
    result := Application.MainForm.Canvas.TextWidth(sString);
  finally
    Application.MainForm.Font := fntSave;
  end;
end;

{ вычисляем ширину табличной сетки, которую необходимо отобразить без
  горизонтальной полосы прокрутки и без дополнительного пространства между
  последней колонкой и вертикальной полосой прокрутки. Свойство DataSource у
  компонента DBGrid, как и свойство DataSet у DataSource, должны быть назначены
  заранее, но таблица не должна быть открытой. Примечание: полученная ширина
  включает ширину вертикальной полосы прокрутки, полученной на основе базового
  режима отображения. Вычисленная ширина полностью занимает рабочую область
  компонента. }
function iCalcGridWidth(dbg: TDBGrid): integer;
const
  cMEASURE_CHAR = '0';
  iEXTRA_COL_PIX = 4;
  iINDICATOR_WIDE = 11;
var
  i, iColumns, iColWidth, iTitleWidth, iCharWidth: integer;
begin
  iColumns := 0;
  result := GetSystemMetrics(SM_CXVSCROLL);
  iCharWidth := NewTextWidth(dbg.Font, cMEASURE_CHAR);
  with dbg.DataSource.DataSet do
    for i := 0 to FieldCount - 1 do with Fields[i] do
```

```
        if Visible then begin
          iColWidth := iCharWidth * DisplayWidth;
          if dgTitles in dbg.Options then begin
            iTitleWidth := NewTextWidth(dbg.TitleFont, DisplayLabel);
            if iColWidth < iTitleWidth then iColWidth := iTitleWidth;
          end;
          inc(iColumns, 1);
          inc(result, iColWidth + iEXTRA_COL_PIX);
        end;
        if dgIndicator in dbg.Options then begin
          inc(iColumns, 1);
          inc(result, iINDICATOR_WIDE);
        end;
    if dgColLines in dbg.Options then inc(result, iColumns)
    else inc(result, 1);
  end;
```

Необходимо использовать функцию NewTextWidth, а не Canvas.TextWidth компонента DBGrid, поскольку Canvas еще не инициализирован во время вызова iCalcGridWidth.

Перемещение данных из DBGrid

Кто-нибудь пробовал перемещать что-либо из DBGrid *методом Drag & Drop?*

Скопируйте код из данного совета, сохраните его с именем DBGrid.pas и установите компонент в палитру. У вас появится новый компонент EDBGrid с двумя новыми событиями: OnMouseDown и OnMouseUp.

```
unit Dbgrid;

interface

uses
  DBGrids, Controls, Classes;

type
  TEDBGrid = class(TDBGrid)
  private
    FOnMouseDown: TMouseEvent;
    FOnMouseUp: TMouseEvent;
  protected

    procedure MouseDown(Button: TMouseButton; Shift: TShiftState;
                        X, Y: Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState;
                      X, Y: Integer); override;
  published
    property OnMouseDown: TMouseEvent read FOnMouseDown write FOnMouseDown;
    property OnMouseUp: TMouseEvent read FOnMouseUp write FOnMouseUp;
  end;
```

```
procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('Data Controls', [TEDBGrid]);
end;

procedure TEDBGrid.MouseDown(Button: TMouseButton; Shift: TShiftState;
                             X, Y: Integer);
begin
  if Assigned(FOnMouseDown) then FOnMouseDown(Self, Button, Shift, X, Y);
  inherited MouseDown(Button, Shift, X, Y);
end;

procedure TEDBGrid.MouseUp(Button: TMouseButton; Shift: TShiftState;
                           X, Y: Integer);
begin
  if Assigned(FOnMouseUp) then FOnMouseUp(Self, Button, Shift, X, Y);
  inherited MouseUp(Button, Shift, X, Y);
end;

end.
```

DBGrid и клавиши акселерации

Форма содержит поле TEdit*, поле* TComboBox*,* TDBGrid *и несколько кнопок* BitBtn *(в перечисленном порядке идет и обход компонентов клавишей* <Tab>*). Табличная сетка была установлена для выбора целой колонки, и обратной операции при потере фокуса. Некоторым кнопкам были назначены клавиши акселерации путем установки символа «&» в их заголовках. Нажимая* <Alt> *и соответствующую клавишу, ожидаешь определенной реакции, но не тут-то было. Беда приходит, когда табличная сетка получает фокус.*

Все работает превосходно, если вы работаете с клавишами навигации (например, <Up>, <Down>, <PageUp> и т. д.); тем не менее, если использовать клавиши акселерации (не используя клавишу <Alt>), возникают соответствующие клавишам события OnClick.

DBGrid – свойства FixRows и FixCols

Сетки, не предназначенные для работы с данными, имеют свойства FixCols и FixRows, которые не позволяют данным прокручиваться, но эти свойства не были унаследованы TDBGrid.

Обходной путь для TDrawGrid:

```
TDrawGrid(DBGrid1).FixedCols := 2;
```

DBGrid – поддержка одинарного щелчка

Есть какой-либо способ отследить событие одинарного щелчка на компоненте `TDBGrid`*?*

Нет. Вы должны взять объект, поддерживающий данное событие (типа `TStringGrid`) и добавить поддержку данных (см. руководство по созданию компонентов «Component Creation User Guide»), или взять `TDBGrid` и добавить событие `OnMouseDown` (см. главу 4 руководства по созданию компонентов).

Работа с несколькими записями

`DBGrid` не отображает графические поля. Но эту задачу можно решить программным путем. Один из методов заключается в следующем: на форме располагаются несколько компонентов `TTable` со ссылкой на одну и ту же таблицу и несколько компонентов `DBImage`, связанных с компонентами `TTable`. Основная идея заключается в синхронизации позиции курсора, управляемого пользователем. Для этого мы применим компонент `DBNavigator`, который заботится о перемещении в первой таблице, и запишем код для обработчика события навигатора `OnClick` позиционирования указателя записи для других объектов. Для воссоздания примера выполните следующее:

- Расположите на форме три компонента `TTable` с именами `Table1`, `Table2` и `Table3` соответственно;
- Расположите на форме три компонента `DataSources` и свяжите их с `Table1..Table3`;
- Расположите на форме три компонента `DBImage` и свяжите их с `DataSource1..DataSource3`, все они будут иметь одно и то же значение свойства `DataField`;
- Расположите на форме компонент `DBNavigator` и свяжите его с `DataSource1`;
- Создайте процедуру, как показано ниже:

```
procedure InitPosition;
begin
{ Начальная позиция }
  Form1.Table2.Active := True;
  Form1.Table3.Active := True;
  if Form1.Table1.RecordCount > 1 then Form1.Table2.MoveBy(1);
  if Form1.Table2.RecordCount > 2 then Form1.Table3.MoveBy(2);
end;
```

- В обработчике события `Form.OnShow` вызовите `InitPosition`;
- В обработчике события `OnClick` компонента `DBNavigator` создайте следующий код:

```
procedure TForm1.DBNavigator1Click(Sender: TObject; Button: TNavigateBtn);
begin
  case button of
```

```
      nbNext: begin
                if not Table2.EoF then Table2.Next;
                if Table2.EoF then Table2.Active := False;
                if not Table3.EoF then Table3.Next;
                if Table3.EoF then Table3.Active := False;
              end;
    nbPrior: begin
                if not Table2.BoF then Table2.Prior;
                if not Table3.BoF then Table3.Prior;
             end;
    nbFirst: InitPosition;
     nbLast: begin
               Table2.Active := False;
               Table3.Active := False;
             end;
  end;
end;
```

Данный метод отображает на форме три графических поля. Пожалуйста, имейте в виду, что вышеуказанный код не завершен и требует доработки. Кроме того, существуют другие решения.

[News Group]

Примечание

С большим успехом можно использовать компонент `DBCtrlGrid`. *При этом никакой дополнительный код не нужен, да и связка* `Ttable – TDataSource` *нужна всего одна.*

Предохранение от автоматического добавления записи

Как предотвратить автоматическое добавление записей в таблицу. Может быть, предусмотреть какую-то хитрость для создания новой записи в табличной сетке?

Для `DBGridKeyDown` используйте такой код:

```
  ...
begin
  s := 'ASCII-код клавиши ' + IntToStr(Ord(key)) + ' десятичное';

{ ShowMessage(s); }

  s := IntToStr(Ord(key));
end;
  ...
```

Затем в `TTable` сделайте следующее:

```
  ...
begin
  if s <> '45' then raise EAbort.Create('');
```

```
  s := '';
end;
  ...
```

Естественно, переменная s должна быть объявлена глобально.

Перехват события компонента DBGrid OnMouseDown

Как перехватить событие DBGrid OnMouseDown*?*

Необходимо создать класс-наследник TDBGrid и перекрыть процедуру MouseDown.

Небольшой пример:

```
unit MyDBGrid;

interface

uses
  SysUtils, Windows, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Grids, DBGrids;

type
  TMyDBGrid = class(TDBGrid)
  protected
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState;
                        X, Y: Integer); override;
  published
    property OnMouseDown;
end;

procedure Register;

implementation

procedure TMyDBGrid.MouseDown(Button: TMouseButton; Shift: TShiftState;
                              X, Y: Integer);
var
  FOnMouseDown: TMouseEvent;
begin
  inherited MouseDown(Button, Shift, X, Y);
  FOnMouseDown := OnMouseDown;
  if Assigned(FOnMouseDown) then FOnMouseDown(Self, Button, Shift, X, Y);
end;

procedure Register;
begin
  RegisterComponents('Samples', [TMyDBGrid]);
end;

end.
```

Использование клавиши <Enter> как <Tab> в DBGrid

Данный код включает обработку клавиши <Enter> для всего приложения, включая поля и пр. Код для работы с компонентом DBGrid заключен в блок Else. Приведенный код не имитирует поведение клавиши <Tab>, связанное с переходом на следующую запись, когда курсор достигает последней колонки табличной сетки, в нашем случае он перемещается на первую колонку.

```
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
begin
  if Key = #13 then                                  { если это клавиша Enter }
    if not (ActiveControl is TDBGrid) then begin     { если не на TDBGrid }
      Key := #0;                                     { гасим клавишу Enter }
      Perform(WM_NEXTDLGCTL, 0, 0); { перемещаем на следующий элемент управления }
    end else if (ActiveControl is TDBGrid) then      { если это TDBGrid }
      with TDBGrid(ActiveControl) do
        if SelectedIndex < (FieldCount -1) then      { увеличиваем поле }
          SelectedIndex := SelectedIndex +1
        else
          SelectedIndex := 0;
end;
```

Обновление вычисляемых полей в DBGrid

Разместите строчку типа нижеприведенной в конце кода обработчика события OnCalcFields:

```
if DBGrid1.Showing then DBGrid1.Invalidate;
```

[News Group]

DBGrid без вертикальной полосы прокрутки

Для удаления вертикальной полосы прокрутки из компонента TDBGrid необходимо перекрыть метод Paint. Внутри метода Paint следует вызвать процедуру API SetScrollRange, чтобы установить минимальные и максимальные значения полосы прокрутки в ноль (это запретит вывод полосы прокрутки), после чего вызвать родительский метод Paint. Код, приведенный ниже, – новый компонент, названный TNoVertScrollDBGrid, у которого отсутствует вертикальная полоса прокрутки. Вы можете скопировать этот код в файл с именем NEWGRID.PAS и добавить данный компонент в палитру компонентов.

```
unit Newgrid;
interface

uses
  WinTypes, WinProcs, Classes, DBGrids;

type
  TNoVertScrollDBGrid = class(TDBGrid)
```

```
  protected
    procedure Paint; override;
end;

procedure Register;
implementation

procedure TNoVertScrollDBGrid.Paint;
begin
  SetScrollRange(Self.Handle, SB_VERT, 0, 0, False);
  inherited Paint;
end;

procedure Register;
begin
  RegisterComponents('Data Controls', [TNoVertScrollDBGrid]);
end;

end.
```

Примечание

Проблема решена лишь частично. Появляется другая проблема – мелькание `ScrollBar` *при работе с компонентом (перерисовка).*

Многострочный DBGrid

Добавлено дополнительное свойство `LinesPerRow`. Установка значений данного свойства соответственно изменяет высоту строки, в зависимости от текущего шрифта. Текст в ячейках будет переноситься, если значение `LinesPerRow` больше, чем единица. Все это произведение искусства оказалось чрезвычайно полезным и удивительно простым, так что я публикую его здесь в надежде, что оно пригодится кому-нибудь еще. Код простой, но для его понимания необходимо изучение исходного кода VCL.

Небольшая доводка все же нужна (обработка полей BLOB, ошибок и т. д.), но это не сложно.

```
unit Dbmygrid;

interface

uses
  SysUtils, Windows, Messages, Classes, Graphics, Controls, Forms, Dialogs, DB,
  DBTables, StdCtrls, ExtCtrls, Grids, DBGrids;

type
  TMultiLineDBGrid = class(TDBGrid)
  private
```

```
    FLinesPerRow: Integer;
    procedure DrawDataCell(Sender: TObject; const Rect: TRect; Field: TField;
                           State: TGridDrawState);
    procedure LayoutChanged; override;
    procedure SetLinesPerRow(ALinesPerRow: Integer);

  public
    property LinesPerRow: Integer read FLinesPerRow write SetLinesPerRow default 1;
    constructor Create(AOwner: TComponent); override;
end;

procedure Register;
implementation

constructor TMultiLineDBGrid.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  FLinesPerRow := 1;
  OnDrawDataCell := DrawDataCell;
end;

procedure TMultiLineDBGrid.LayOutChanged;
begin
  inherited LayOutChanged;
  DefaultRowHeight := DefaultRowHeight * LinesPerRow;
end;

procedure TMultiLineDBGrid.DrawDataCell(Sender: TObject; const Rect: TRect;
                                        Field: TField; State: TGridDrawState);
var
  Format: Word;
  C: array[0..255] of Char;
  R: TRect;

begin
  if LinesPerRow = 1 then Format := DT_SINGLELINE or DT_LEFT
  else Format := DT_LEFT or DT_WORDBREAK;
  Canvas.FillRect(Rect);
  StrPCopy(C, Field.AsString);
  R := Rect;
  Windows.DrawText(Canvas.Handle, C, StrLen(C), R, Format);
end;

procedure TMultiLineDBGrid.SetLinesPerRow(ALinesPerRow: Integer);
begin
  if ALinesPerRow <> FLinesPerRow then begin
    FLinesPerRow := ALinesPerRow;
    LayoutChanged;
  end;
end;
```

```
procedure Register;
begin
  RegisterComponents('Data Controls', [TMultiLineDBGrid]);
end;

end.
```

[News Group]

Примечание

Компонент нуждается в серьезной доводке, т. к. он часто является источником ошибок, да и визуальные средства компонента оставляют желать лучшего.

DBGrid DefaultDrawDataCell

TDBGrid имеет недокументированный в электронной справке метод DefaultDrawDataCell.

Пример его использования:

```
procedure TForm1.DBGrid1DrawDataCell(Sender: TObject; const Rect: TRect;
                                     Field: TField; State: TGridDrawState);
begin
  with Sender as TDBGrid do begin
    Canvas.Font.Color := clRed;
    DefaultDrawDataCell(Rect, Field, State);
  end;
end;
```

TDBGrid – копирование в буфер обмена

Простая процедура копирования информации из DBGrid в Clipboard может существенно облегчить жизнь при реализации требований экспорта выборок данных во внешние приемники:

```
unit UnGridToClb;

interface
uses
  Windows, SysUtils, Classes, Dialogs, Grids, DBGrids, Db, DBTables, ClipBrd;

procedure CopyGRDToClb(dbg :TDBGrid);
// Копирует DBGrid в буфер обмена, после чего данные отлично переносятся
// как в простой текстовый редактор, так и в Excel
implementation

procedure CopyGRDToClb(dbg: TDBGrid);
var
  bm: TBookMark;
  pch, pch1: PChar;
```

```
  s, s2: string;
  i, j: integer;
begin
  s := '';
  for j := 0 to dbg.Columns.Count - 1 do
    s := s + dbg.Columns.Items[j].Title.Caption +#9 ;
  s := s + #13 + #10;
  if not dbg.DataSource.DataSet.active then begin
    ShowMessage('Нет выборки!!!');
    Exit;
  end;
  try
    dbg.Visible := False;   // Делаем сетку невидимой, чтобы не уходило время
                            // на ее перерисовку при прокрутке DataSet
    bm := dbg.DataSource.DataSet.GetBookmark; // чтобы не потерять текущую запись
    dbg.DataSource.DataSet.First;
    while not dbg.DataSource.DataSet.EOF do begin
      s2 := '';
      for j := 0 to dbg.Columns.Count - 1 do begin
        s2 := s2 + dbg.Columns.Items[j].Field.AsString + #9;
      end;
      s := s + s2 + #13 + #10;
      dbg.DataSource.DataSet.Next;
    end;
// Переключаем клавиатуру в режим кириллицы, иначе - проблемы с кодировкой
    GetMem(pch, 100);
    GetMem(pch1, 100);
    GetKeyboardLayoutName(pch);
    StrCopy(pch1, pch);
    while pch <> '00000419' do begin
      ActivateKeyboardLayout(HKL_NEXT, 0);
      GetKeyboardLayoutName(pch);
      if StrComp(pch, pch1) = 0 then
// Круг замкнулся - нет такого языка '00000419'
        StrCopy(pch, '00000419');
    end;
    Clipboard.AsText := s;                      // Данные - в буфер!!!
    while strComp(pch, pch1) <> 0 do begin      // Возвращаем режим клавиатуры
      ActivateKeyboardLayout(HKL_NEXT, 0);
      GetKeyboardLayoutName(pch);
    end;
    FreeMem(pch);
    FreeMem(pch1);
    dbg.DataSource.DataSet.GotoBookmark(bm);
//     ShowMessage('Данные успешно скопированы в буфер обмена.');
  finally
    dbg.Visible := True;
  end;
end;
end.
```

[Беличенко Б]

DBGrid с номером строки

Кто подскажет, как в DBGrid *можно нумеровать строки?*

Откомпилируйте приведенный ниже код и вы получите новый компонент с нужными свойствами:

```
unit RowGrid;
interface

uses
  Windows, Classes, Grids, DBGrids;

type
  TRowDBGrid = class(TDBGrid)
  public
    property Row;
    property RowCount;
    property VisibleRowCount;
  end;

procedure Register;
implementation

procedure Register;
begin
  RegisterComponents('Data Controls', [TRowDBGrid]);
end;

end.
```

Небольшой демонстратор. Поместите на форму TRowDBGrid, три компонента TEdit, а следующий код – в обработчик события OnDrawDataCell нового компонента:

```
procedure TForm1.RowDBGrid1DrawDataCell(Sender: TObject; const Rect: TRect;
  Field: TField; State: TGridDrawState);
begin
  eb_Row.Text := IntToStr(RowDBGrid1.Row);
  eb_RowCount.Text := IntToStr(RowDBGrid1.RowCount);
  eb_VisibleRowCount.Text := IntToStr(RowDBGrid1.VisibleRowCount);
end;
```

[News Group]

Примечание

eb_Row, eb_RowCount, eb_VisibleRowCount – *имена компонентов* TEdit. *Свойство* RowDBGrid1.DefaultDrawing = False.

Хочется напомнить, что номер записи может играть роль только для формата .DBF *таблиц базы данных. Для таблиц* .DB *и баз данных SQL он не имеет смысла.*

Текстовое содержимое ячейки DBGrid

Не могу найти метод получения текстового значения ячейки сетки, кроме назначения `TField`...

Это можно сделать с помощью простого наследника `TDBGrid`:

```
type
  TMyGrid = class(TDBGrid)
  public
    property InPlaceEditor;
  end;
```

А затем получить доступ к содержимому ячейки приблизительно таким способом:

```
Label1.Caption := DBGrid1.InPlaceEditor.Text;
```

Имейте в виду, что это работает только в случае, когда ячейка находится в состоянии редактирования или вставки.

[News Group]

DBGrid – выбранные строки

Существует одно свойство, не упомянутое в файлах помощи (оплошность, господа программисты из Borland), с именем `SelectedRows`. И вот как его можно использовать:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  i: integer;
begin
  for i := 0 to DBGrid1.SelectedRows.Count - 1 do begin
    Table1.GoToBookmark(TBookmark(DBGrid1.SelectedRows[i]));
    Table1.Delete;
  end;
end;
```

[News Group]

Улучшенный DBGrid

Код улучшенного `TDBGrid`, имеющего свойства `Col`, `Row`, `Canvas` и метод `CellRect`. Это чрезвычайно полезно, если требуется получить выпадающий список на месте редактируемой пользователем ячейки.

```
unit VUBComps;

interface
```

```
uses
  Windows, SysUtils, Messages, Classes, Graphics, Controls, Dialogs,
  Grids, DBGrids, DB;

type
  TDBGridVUB = class(TDBGrid)
  public
    function CellRect(ACol, ARow: Longint): TRect;
    property Canvas;
    property Col;
    property Row;
  end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('VUBudget', [TDBGridVUB]);
end;

function TDBGridVUB.CellRect(ACol, ARow: Longint): TRect;
begin
  Result := inherited CellRect(ACol, ARow);
end;

end.
```

Контроль данных в TDBGrid

Как узнать, что пользователь вводит в TDBGrid*?*

«Увидеть», что набирается в TDBGrid можно, «смотря» на элемент редактирования сетки TInPlaceEdit. Необходимо только убедиться в том, что к моменту использования TInPlaceEdit элемент управления уже создан. Следующая функция покажет данные, редактируемые в колонках сетки:

```
procedure TForm1.DBGrid1KeyUp(Sender: TObject; var Key: Word; Shift: TShiftState);
var
  B: integer;
begin
  for B := 0 to DBGrid1.ControlCount - 1 do
    if DBGrid1.Controls[B] is TInPlaceEdit then
      with DBGrid1.Controls[B] as TInPlaceEdit do
        Label1.Caption := 'Текст = ' + Text;
end;
```

Обновление DBGrid после редактирования отдельной записи в отдельной форме

При закрытии форма самостоятельно не сохраняет данные. Необходимо сохранить изменения или с помощью компонента DBNavigator, или с помощью кода, который при закрытии формы сохраняет данные в основную таблицу.

На стр. 95 «Database Application Developers Guide» (Руководство разработчиков приложений баз данных), поставляемого с Delphi, приведен демонстрационный проект с двумя формами, демонстрирующий хорошую технику при использовании TTable на мастер-форме в качестве набора данных для подчиненной формы.

Одним из решений проблемы может служить связывание компонента DataSource на Form2 с набором данных DataSet на Form1. Это достигается путем добавления следующей строки в обработчик события OnActivate для Form2:

```
MyDataSource.DataSet := Form1.MyTable;
```

Данный метод имеет три преимущества:

- внесенные изменения немедленно отображаются, поскольку вы используете одну и ту же логическую таблицу;
- определение установок для полей таблицы, например, DisplayFormat или EditMask необходимо произвести только один раз в таблице на Form1. Это не нужно делать на каждой форме, которая работает с таблицей;
- это сохраняет ресурсы и повышает производительность, поскольку приложение при работе с таблицей использует только одну сессию. Тем не менее, во время разработки нужно иметь TTable на Form2 для того, чтобы можно было выбрать поля для элементов управления базы данных. После чего TTable можно удалить.

Решение проблемы передачи фокуса DBGrid

Решение проблемы невозможности получения DBGrid фокуса после щелчка по какому-либо элементу управления родительской формы, если DBGrid находится на ее дочерней MDI-форме.

Эта проблема решена в приведенном ниже наследнике TDBGrid, в котором обрабатываются «мышиные» сообщения и выясняется, когда фокус должен быть передан сетке. Наследник создан в виде компонента, который легко устанавливается в Палитру компонентов.

```
unit FixedDBGrid;

interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Grids, DBGrids;
```

```
type
  TFixedDBGrid = class(TDBGrid)
  public
    procedure WMRButtonDown(var Message: TWMRButtonDown); message WM_RBUTTONDOWN;
    procedure WMLButtonDown(var Message: TWMLButtonDown); message WM_LBUTTONDOWN;
  end;

procedure Register;
implementation

procedure TFixedDBGrid.WMRButtonDown(var Message: TWMRButtonDown);
begin
  Windows.SetFocus(handle);
  inherited;
end;

procedure TFixedDBGrid.WMLButtonDown(var Message: TWMLButtonDown);
begin
  Windows.SetFocus(handle);
  inherited;
end;

procedure Register;
begin
  RegisterComponents('Samples', [TFixedDBGrid]);
end;

end.
```

Позиция ячейки в DBGrid

В TCustomGrid определен метод CellRect, который, к сожалению, защищен. Это означает, что данный метод доступен только для TCustomGrid и его наследников. Но все-таки способ вызова данного метода существует, хотя и немного мудреный:

```
type
  TMyDBGrid = class(TDBGrid)
  public
    function CellRect(ACol, ARow: Longint): TRect;
  end;

function TMyDBGrid.CellRect(ACol, ARow: Longint): TRect;
begin
  Result := inherited CellRect(ACol, ARow);
end;
```

Сортировка DBLookupComboBox по вторичному индексу

Одним из способов вывода данных в другом порядке сортировки является использование TQuery и включение в SQL-запрос ключевого слова ORDER BY. После чего вы можете установить этот запрос как DataSource в своем DBLookupComboBox.

Решение

Пусть имеется таблица Customer, содержащая поля Customer_No и Customer_Name, индексированная по Customer_No. Тогда запрос должен содержать в редакторе списка строк для TQuery следующую строку:

```
SELECT Customer_No, Customer_Name
FROM Customer
ORDER BY Customer_Name
```

Значение DBLookupComboBox

Искомое поле можно получить, обратившись к свойству LookUpValue.

```
unit clookup;

interface

uses
  Windows, Messages, Classes, Graphics, Controls, SysUtils, Forms, Dialogs,
  StdCtrls, DB, DBLookup;

type
  TDBJustLookupCombo = class(TDBLookupCombo)
  protected
    function GetLValue: TField;
  public
    property LookUpValue: TField read GetLValue;
  end;
  TDBJustLookupList = class(TDBLookupList)
  protected
    function GetLValue: TField;
  public
    property LookUpValue: TField read GetLValue;
end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('Data Controls', [TDBJustLookupList, TDBJustLookupCombo]);
end;
```

```
function TDBJustLookupCombo.GetLValue: TField;
begin
  Result := LookupSource.DataSet.FieldByName(LookUpField);
end;

function TDBJustLookupList.GetLValue: TField;
begin
  Result := LookupSource.DataSet.FieldByName(LookUpField);
end;
end.
```

Примечание

Компоненты TDBLookupCombo *и* TDBLookupList *оставлены в палитре компонентов только для совместимости, поэтому их крайне нежелательно использовать для новых разработок. Но сама идея этого совета может кому-то потребоваться для работы.*

Две колонки в DBLookupComboBox

Попробуйте использовать что-то подобное следующему:

```
DBLookupCombo1.LookupDisplay := 'Company;City;Country';
```

То есть, для того чтобы выводить более одного поля, отделите имя каждого поля точкой с запятой.

Проблема хранения DBImage

Исходный код компонента DBImage содержит ошибку, поскольку пробует, загружая данные буфера обмена, отыскать CF_PICTURE. А это несовместимо с хранящимися в поле данными.

Исправьте эту ошибку, изменив исходный код таким образом, чтобы данные буфера обмена сначала читались во временный TBitmap, который регистрируется как CF_PICTURE. Временное изображение затем назначьте полю, поскольку оно также работает с изображениями, то и данным поля.

Все это потребует внести изменения в исходный код VCL. Конкретно – в модуль DBCTRLS.PAS. Затем, естественно, это необходимо перекомпилировать и пересобрать:

```
procedure TDBImage.PasteFromClipboard;
var
  ClipBrdBmp: TBitmap;
begin
  ClipBrdBmp := TBitmap.Create;
  if Clipboard.HasFormat(CF_BITMAP) and FDataLink.Edit then begin
    ClipBrdBmp.Assign(Clipboard);
    Picture.Assign(ClipBrdBmp);
  end;
  ClipBrdBmp.Free;
end;
```

[News Group]

Копирование текста DBMemo

Вся хитрость заключается в создании TStringList и перекачивании в него данных для последующей работы. Вот простая процедура без проверок и уведомлений об ошибках, использующая модуль WinCRT:

```
procedure TForm1.Button1Click(Sender: TObjct);
var
  I: integer;
  StringList: TStringList;
begin
  StringList := TStringList.Create;
  try
    StringList.Assign(Table1Memo);
    for I := 0 to StringList.Count - 1 do
      Writeln(StringList[I])
  finally
    StringList.Free
  end
end;
```

Поиск текста в DBMemo

Подключите следующую процедуру к событию OnFind для FindDialog. Единственная неприятность заключается в том, что в DBMemo нельзя получить выделенный текст. Тем не менее, в стандартном Memo такой проблемы нет.

```
procedure TForm1.FindDialog1Find(Sender: TObject);
var
  Buff, P, FT: PChar;
  BuffLen: Word;
begin
  with Sender as TFindDialog do begin
    GetMem(FT, Length(FindText) + 1);
    StrPCopy(FT, FindText);
    BuffLen := DBMemo1.GetTextLen + 1;
    GetMem(Buff, BuffLen);
    DBMemo1.GetTextBuf(Buff, BuffLen);
    P := Buff + DBMemo1.SelStart + DBMemo1.SelLength;
    P := StrPos(P, FT);
    if P = NIL then MessageBeep(0)
    else begin
      DBMemo1.SelStart := P - Buff;
      DBMemo1.SelLength := Length(FindText);
    end;
    FreeMem(FT, Length(FindText) + 1);
    FreeMem(Buff, BuffLen);
  end;
end;
```

Пример KeyDown компонента DBNavigator

Есть решение для создания «горячих» клавиш в DBNavigator. Установите свойство TForm.KeyPreview в True и напишите обработчик события OnKeyDown:

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
const
  KeyBtn: array[TNavigateBtn] of record
          Key: Word;
          Btn: TNavigateBtn;
          end = ((Key: VK_F1; Btn: nbFirst),
                 (Key: VK_F2; Btn: nbPrior),
                 (Key: VK_F3; Btn: nbNext),
                 (Key: VK_F4; Btn: nbLast),
                 (Key: VK_F5; Btn: nbInsert),
                 (Key: VK_F6; Btn: nbDelete),
                 (Key: VK_F7; Btn: nbEdit),
                 (Key: VK_F8; Btn: nbPost),
                 (Key: VK_F9; Btn: nbCancel),
                 (Key: VK_F10; Btn: nbRefresh));
var
  i: TNavigateBtn;
begin
  for i := nbFirst to nbRefresh do
    if KeyBtn[i].Key = Key then begin
      DBNavigator1.BtnClick(KeyBtn[i].Btn);
      Exit;
    end;
end;
```

[Dmitry]

Свойства кнопок DBNavigator

Как выяснить значения свойств кнопок компонента DBNavigator *(enabled/disabled)?*

Управлять видимостью кнопок можно при помощи свойства VisibleButtons. Например:

```
if nbRefresh in DBNavigator1.VisibleButtons then
  ShowMessage('Кнопка Refresh видимая');
```

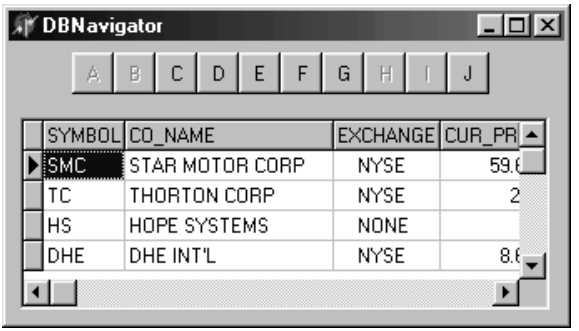
Для того чтобы узнать, активизирована кнопка или нет (enabled/disabled):

```
if DBNavigator1.Controls[Ord(nbFirst)].Enabled then
  ShowMessage('Кнопка First активизирована') ;
```

Вместо nbFirst можно определить другой член TNavigateBtn. Например, nbFirst, nbPrior, nbNext, nbLast, nbInsert, nbDelete, nbEdit, nbPost, nbCancel, nbRefresh.

DBNavigator без пиктограмм

Я хочу создать `DBNavigator`*, который вместо стандартных пиктограмм содержал бы текст.*



В этом поможет следующий код:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  c: shortint;
  s: string;
begin
  s := 'A';
  with DBNavigator1 do
    for c := 0 to ControlCount - 1 do
      if Controls[c] is TNavButton then
        with TNavButton(Controls[c]) do begin
          Glyph := nil;
          Caption := s;
          Inc(s[1]);
        end;
end;
```

[News Group]

Настройки всплывающих подсказок в TDBNavigator

Возможно ли изменение свойства `Hints` *компонента* `TDBNavigator` *во время выполнения программы?*

Это должно работать:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  ix: integer;
begin
  with DBNavigator1 do
    for ix := 0 to ControlCount - 1 do
```

```
      if Controls[ix] is TNavButton then
        with Controls[ix] as TNavButton do
          case index of
            nbFirst: Hint := 'Подсказка для кнопки First';
            nbPrior: Hint := 'Подсказка для кнопки Prior';
             nbNext: Hint := 'Подсказка для кнопки Next';
             nbLast: Hint := '';
           {......}
          end;
end;
```

Выключение кнопок в TDBNavigator

Расширение TDBNavigator. Позволяет разработчику включать и выключать отдельные кнопки посредством методов EnableButton и DisableButton.

```
unit GNav;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, DBCtrls;

type
  TMyNavigator = class(TDBNavigator)
  public
    procedure EnableButton(Btn: TNavigateBtn);
    procedure DisableButton(Btn: TNavigateBtn);
  end;

procedure Register;
implementation

procedure TMyNavigator.EnableButton(Btn: TNavigateBtn);
begin
  Buttons[Btn].Enabled := True;
end;
procedure TMyNavigator.DisableButton(Btn: TNavigateBtn);
begin
  Buttons[Btn].Enabled := False;
end;

procedure Register;
begin
  RegisterComponents('Samples', [TMyNavigator]);
end;
end.
```

Получение индекса компонента в списке родителя

Как найти индекс компонента в родительском списке дочерних элементов управления? Я попытался модифицировать `prjexp.dll`*, но без успеха. У кого-нибудь есть идеи?*

Есть такая функция. Ищет родителя заданного компонента, перебирает список и возвращает индекс искомого компонента. Функция прошла многочисленные тесты и вполне работоспособна.

```
function IndexInParent(vControl: TControl): integer;
var
  ParentControl: TWinControl;
begin
{ делаем "слепок" родителя через базовой класс на предмет доступности }
  ParentControl := TForm(vControl.Parent);
  if (ParentControl <> nil) then begin
    for Result := 0 to ParentControl.ControlCount - 1 do begin
      if (ParentControl.Controls[Result] = vControl) then Exit;
    end;
  end;
  Result := -1;
end;
```

Дублирование компонентов и их потомков во время выполнения приложения

```
unit Unit2;

interface

uses
  SysUtils, Windows, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

function Replicator(C: TComponent): TComponent;
implementation

{ Следующая процедура "клонирует" свойства C1 и записывает их в C2. C1 и C2 должны
  иметь один и тот же тип. Используйте данный метод для компонентов, не имеющих
  метода Assign. }
procedure CloneComponent(C1: TComponent; C2: TComponent);
var
  S: TMemoryStream;
begin
  if C1.ClassType <> C2.ClassType then
    raise EComponentError.Create('Типы объектов не совместимы');
  if C1 is TWinControl then TWinControl(C2).Parent := TWinControl(C1).Parent;
  S := TMemoryStream.Create;       { создаем поток для работы с памятью }
  with S do begin
    WriteComponent(C1);            { пишем свойства C1 в поток }
    Seek(0, 0);                    { перемещаемся в начало потока }
```

```
    ReadComponent(C2);              { читаем свойства из потока в C2 }
    Free;                           { освобождаем поток }
  end;
end;

{ Следующая функция "реплицирует" компонент C и возвращает новый компонент типа и
  со свойствами компонента C. }
function Replicator(C: TComponent): TComponent;
begin
  Result := TComponentClass(C.ClassType).Create(C.Owner);   { создаем компонент }
  CloneComponent(C, Result);                                { клонируем его }
end;

end.
```

[News Group]

Refresh или Repaint?

В файлах справки написано, что `Repaint` *перед отрисовкой не очищает область элемента управления, а как быть, если мне это необходимо? Есть какое-то решение?*

Попробуйте добавить `csOpaque` в `ControlStyle`:

```
ControlStyle := ControlStyle + [csOpaque];
```

Это уведомит VCL о том, что элемент управления полностью закрашивает свою область и нет необходимости в ее очищении.

[News Group]

Имя класса компонента и модуля

Мне необходима функция, которая возвращала бы имя класса компонента и имя модуля, где определен данный класс. Например: `xxx('TPanel')` *возвращала бы* `'ExtCtrls'`

Решение

```
uses TypInfo;

function ObjectsUnit(Obj: TClass): String;
begin
  Result := GetTypeData(PTypeInfo(Obj.ClassInfo))^.UnitName;
end;
```

Пример компонента HotSpot

Пример компонента `HotSpot`, основанного на `TPanel` (небольшая переделка). Управляя событиями `OnMouseDown` и `OnMouseUp`, можно получить эффект «резинового» контура.

```
unit Newpanel;

interface

uses
  Windows, Classes, Controls, StdCtrls, ExtCtrls;

type
  THotSpotClickEvent = procedure(Sender: TObject; Button: TMouseButton;
                                 Shift: TShiftState) of object;
  TNewPanel = class(TPanel)
  private
    fHotSpotClick: THotSpotClickEvent;
    fHotSpot: TRect;
    fInHotSpot: Boolean;
    function GetHotSpotTop: Word;
    function GetHotSpotLeft: Word;
    function GetHotSpotWidth: Word;
    function GetHotSpotHeight: Word;
    procedure SetHotSpotTop(Value: Word);
    procedure SetHotSpotLeft(Value: Word);
    procedure SetHotSpotWidth(Value: Word);
    procedure SetHotSpotHeight(Value: Word);
  protected
    procedure Paint; override;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState;
                        X, Y: Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState;
                      X, Y: Integer); override;
  public
    procedure Click; override;
    property HotSpot : TRect Read fHotSpot Write fHotSpot;
  published
    property hsTop: Word Read GetHotSpotTop Write SetHotSpotTop;
    property hsLeft: Word Read GetHotSpotLeft Write SetHotSpotLeft;
    property hsWidth: Word Read GetHotSpotWidth Write SetHotSpotWidth;
    property hsHeight: Word Read GetHotSpotHeight Write SetHotSpotHeight;
    property OnHotSpot: THotSpotClickEvent Read fHotSpotClick Write fHotSpotClick;
  end;

procedure Register;

implementation

uses
  Graphics;

procedure Register;
begin
  RegisterComponents('Custom', [TNewPanel]);
end;
```

```
procedure TNewPanel.MouseDown(Button: TMouseButton; Shift: TShiftState;
                              X, Y: Integer);
begin
  if PtInRect(fHotSpot, Point(X, Y)) and Assigned(fHotSpotClick) then
    fInHotSpot := True;
    Inherited MouseDown(Button, Shift, X, Y);
end;

procedure TNewPanel.MouseUp(Button: TMouseButton; Shift: TShiftState;
                            X, Y: Integer);
begin
  Inherited MouseUp(Button, Shift, X, Y);
  if fInHotSpot then begin
    if Assigned(fHotSpotClick) then fHotSpotClick(Self, Button, Shift);
    fInHotSpot := False;
  end;
end;

procedure TNewPanel.Click;
begin
  if Not fInHotSpot then Inherited Click;
end;

procedure TNewPanel.Paint;

var
  OldStyle: TPenStyle;

begin
  Inherited Paint;
  if csDesigning in ComponentState then begin
    OldStyle := Canvas.Pen.Style;
    Canvas.Pen.Style := psDash;
    Canvas.Rectangle(HotSpot.Left, HotSpot.Top, HotSpot.Right, HotSpot.Bottom);
    Canvas.Pen.Style := OldStyle;
  end;
end;

procedure TNewPanel.SetHotSpotTop(Value: Word);
begin
  fHotSpot.Bottom := fHotSpot.Bottom + Value - fHotSpot.Top;
  fHotSpot.Top := Value;
  Paint;
end;

procedure TNewPanel.SetHotSpotLeft(Value: Word);
begin
  fHotSpot.Right := fHotSpot.Right + Value - fHotSpot.Left;
  fHotSpot.Left := Value;
  Paint;
end;
```

```
procedure TNewPanel.SetHotSpotWidth(Value: Word);
begin
  fHotSpot.Right := fHotSpot.Left + Value;
  Paint;
end;

procedure TNewPanel.SetHotSpotHeight(Value: Word);
begin
  fHotSpot.Bottom := fHotSpot.Top + Value;
  Paint;
end;

function TNewPanel.GetHotSpotTop: Word;
begin
  Result := fHotSpot.Top
end;

function TNewPanel.GetHotSpotLeft: Word;
begin
  Result := fHotSpot.Left;
end;

function TNewPanel.GetHotSpotWidth: Word;
begin
  Result := fHotSpot.Right - fHotSpot.Left;
end;

function TNewPanel.GetHotSpotHeight: Word;
begin
  Result := fHotSpot.Bottom - fHotSpot.Top;
end;

end.
```

Прозрачный компонент

При выводе прозрачного окна, окна или элементы управления, расположенные под ним, также видны. Если перемещается невидимое окно, при условии, что оно не скрывается/выводится вновь, оно сохраняет «снимок» графической области, которая была расположена под ним в момент вывода (а именно: окна или видимые компоненты).

Взгляните на модуль, приведенный ниже. В нем предлагается решение проблемы невидимого окна.

```
unit Invwin1;

interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls;
```

```
type
  TInvWin = class(TWinControl)
  private
    fOnControl: TControl;
    fInvisible: Boolean;
    procedure WMPaint(Var Message: TWMPaint); message WM_PAINT;
  protected
    procedure CreateParams(var Params: TCreateParams ); override;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState;
                        X, Y: Integer); override;
    procedure SetOnControl(Value: TControl); virtual;
  public
    constructor Create(aOwner: TComponent);
    property OnControl: TControl Read fOnControl Write SetOnControl;
    property Invisible: Boolean Read fInvisible Write fInvisible;
  end;
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    CheckBox1: TCheckBox;
    procedure FormCreate(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure CheckBox1Click(Sender: TObject);
  public
    InvWin: TInvWin;
end;

var
  Form1: TForm1;

implementation
{$R *.DFM}
constructor TInvWin.Create(aOwner: TComponent);
begin
  Inherited Create(aOwner);
  ControlStyle := ControlStyle + [csOpaque];
end;
procedure TInvWin.CreateParams(var Params: TCreateParams);
begin
  Inherited CreateParams(Params);
  Params.ExStyle := Params.ExStyle or WS_EX_TRANSPARENT;
end;

procedure TInvWin.WMPaint(var Message: TWMPaint);
var
  DC: THandle;
```

```
  PS: TPaintStruct;
begin
  if Not Invisible then begin
    if Message.DC = 0 then DC := beginPaint(Handle, PS)
    else DC := Message.DC;
    PatBlt(DC, 0, 0, 5, 5, BLACKNESS);
    PatBlt(DC, Width - 6, 0, 5, 5, BLACKNESS);
    PatBlt(DC, 0, Height - 6, 5, 5, BLACKNESS);
    PatBlt(DC, Width - 6, Height - 6, 5, 5, BLACKNESS);
    if Message.DC = 0 then endPaint(Handle, PS);
  end;
end;

procedure TInvWin.MouseDown(Button: TMouseButton; Shift: TShiftState;
                            X, Y: Integer);
begin
  ShowMessage('MouseDown над невидимым окном');
end;

procedure TInvWin.SetOnControl(Value: TControl);
var
  Rect: TRect;
begin
  if Value <> fOnControl then begin

{ Используйте только WM_SETREDRAW, если окно полностью невидимо }
  if Invisible and (Parent <> Nil) then Parent.Perform(WM_SETREDRAW, 0, 0);
    if fOnControl <> Nil then Visible := False;
      if Value <> Nil then begin
        Rect := Value.BoundsRect;
        InflateRect(Rect, 2, 2);
        BoundsRect := Rect;
      end;
      fOnControl := Value;
      if fOnControl <> Nil then Visible := True;
{ Используйте только WM_SETREDRAW, если окно полностью невидимо }
      if Invisible and (Parent <> Nil) then Parent.Perform (WM_SETREDRAW, 1, 0 );
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  InvWin := TInvWin.Create(Self);
  InvWin.Visible := False;
  InvWin.Parent := Self;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  ShowMessage('MouseClick над Button1');
end;
```

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  ShowMessage('MouseClick над Button2');
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  InvWin.OnControl := Button1;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
  InvWin.OnControl := Button2;
end;

procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  InvWin.OnControl := Nil;
  InvWin.Invisible := CheckBox1.Checked;
end;

end.
```

[News Group]

Создание свойства массива компонентов

На форме имеется около 20 компонентов CheckBox. *Я хотел бы поочередно ссылаться на них как на элементы массива.*

Вот способ использования свойства массива. Здесь применяется стандартная техника создания списка компонентов CheckBox в обработчике события OnCreate, позволяющая затем обращаться к ним именно как к массиву компонентов CheckBox.

Здесь приведено два способа «собирания» CheckBox в одном массиве. Один просто собирает все CheckBox на форме. Это простой список, но вы должны убедиться в том, что порядок размещения в нем компонентов такой, каким вы хотите его видеть в массиве. Это нормально в данном случае. Другой способ перечисляет все компоненты согласно их нумерации, заложенной в их имени, но чтобы им воспользоваться, его необходимо раскомментировать. Существует множество других способов. Пусть данный совет послужит читателю отправной точкой в изучении интересного вопроса оптимизации кода.

```
type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
  private
    FCheckboxes: TList;
    function GetCheckbox(Index: integer): TCheckbox;
```

```
  public
    property Checkboxes[Index: integer]: TCheckbox read GetCheckbox;
end;

var
  Form1: TForm1;

implementation
{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
var
  i: integer;
{  AComponent: TComponent;}
begin
  FCheckboxes := TList.Create;
{ это собирает все CheckBox в порядке очередности }
  for i := 0 to ComponentCount - 1 do
    if Components[ i ] is TCheckbox then FCheckboxes.Add(Components[i]);
{ если они имеют имя CheckboxNN и вы хотите разместить их в массиве в том же
  порядке... }
{ for i := 1 to MaxInt do begin
    AComponent := FindComponent('Checkbox' + IntToStr(i));
    if AComponent = NIL then break;
    FCheckboxes.Add(AComponent);
  end ;}
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  FCheckboxes.Free;
end;

function TForm1.GetCheckbox(Index: integer): TCheckbox;
begin
  Result := TCheckbox(FCheckboxes[Index]);
end;
```

[News Group]

Отображение всплывающих подсказок компонентов

Обычно всплывающие подсказки появляются непосредственно под областью компонента. Как отобразить их немного выше?

```
unit Unit1;

interface
uses
  Windows, SysUtils, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, DB, DBTables, ExtCtrls;
```

```
type
  TForm1 = class(TForm)
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
  public
    procedure DoShowHint(var HintStr: string; var CanShow: Boolean;
                         var HintInfo: THintInfo);
end;

var
  Form1: TForm1;

implementation
{$R *.DFM}

procedure TForm1.DoShowHint(var HintStr: string; var CanShow: Boolean;
                            var HintInfo: THintInfo);
begin
  with HintInfo do
    if HintControl = Button1 then begin
      HintPos.X := Left + Button1.Left;
      HintPos.Y := Top  + (Button1.Top);
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Application.OnShowHint := DoShowHint;
end;

end.
```

[News Group]

Создание компонентов для работы с базами данных

Здесь описывается минимум шагов, необходимых чтобы создать компонент для работы с базами данных, который может отображать данные отдельного поля. Примером такого компонента может служить панель со свойствами `DataSource` и `DataField`, похожая на компонент `TDBText`. Для получения дополнительных примеров обратитесь к руководству по написанию компонентов в разделе «Making a Control Data-Aware» электронной справки Delphi.

Для наилучшего понимания данного раздела надо ознакомиться с механизмом функционирования элементов управления для работы с базами данных и основополагающими принципами создания компонентов, такими как:

- создание компонентов на основе существующих;
- перекрытие конструкторов и деструкторов;
- создание новых свойств;

- чтение и запись значений свойств;
- назначение обработчиков событий.

Основные шаги по созданию компонента, осуществляющего навигацию по данным:

- Создайте или наследуйте компонент, который допускает свое отображение, но не ввод данных. Например, компонент TMemo с установленным в True свойством ReadOnly. В примере, приведенном в данном совете, используется TCustomPanel, который как раз и позволяет себя отображать, но не дает возможности вводить данные.
- Добавьте к своему компоненту datalink object (объект для связи с данными). Данный объект позволяет управлять связью между компонентом и таблицей базы данных.
- Добавьте к компоненту свойства DataField и DataSource.
- Добавьте методы для получения и установления DataField и DataSource.
- Добавьте к компоненту метод DataChange, позволяющий управлять событиями OnDataChange объекта datalink.
- Перекройте конструктор компонента для создания datalink и перехвата метода DataChange.
- Перекройте деструктор компонента для очищения datalink.

Опишем создание TDBPanel:

- Создайте или наследуйте компонент, который допускает свое отображение, но не ввод данных. В качестве отправной точки для нашего примера будем использовать TCustomPanel.
- Выберите соответствующий пункт меню для создания нового компонента (он меняется от версии к версии Delphi), определите TDBPanel как имя класса и TCustomPanel в качестве наследуемого типа. Определите любую страницу Палитры компонентов.
- Добавьте DB и DBTables в список используемых модулей.
- Добавьте datalink объект в секцию private своего компонента. Данный пример отображает данные одного поля, поэтому мы используем TFieldDataLink для обеспечения связи между нашим новым компонентом и DataSource. Имя нового datalink-объекта – FDataLink.

```
private
  FDataLink: TFieldDataLink;
```

- Добавьте к компоненту свойства DataField и DataSource. Мы добавим соответствующий код для методов записи/чтения в последующих шагах.

Примечание

Наш новый компонент будет иметь свойства DataField *и* DataSource*,* FDataLink *также будет иметь собственные свойства* DataField *и* DataSource.

```
published
  property DataField: string read GetDataField write SetDataField;
  property DataSource: TDataSource read GetDataSource write SetDataSource;
```

- Добавьте частные методы для чтения/записи значений свойств DataField, DataSource и свойств DataField и DataSource для FDataLink:

```
private
  FDataLink: TFieldDataLink;
  function GetDataField: String;
  function GetDataSource: TDataSource;
  procedure SetDataField(Const Value: string);
  procedure SetDataSource(Value: TDataSource);
  ...

implementation
  ...

function TDBPanel.GetDataField: String;
begin
  Result := FDataLink.FieldName;
end;

function TDBPanel.GetDataSource: TDataSource;
begin
  Result := FDataLink.DataSource;
end;

procedure TDBPanel.SetDataField(Const Value: string);
begin
  FDataLink.FieldName := Value;
end;

procedure TDBPanel.SetDataSource(Value: TDataSource);
begin
  FDataLink.DataSource := Value;
end;
```

- Добавьте частный метод DataChange, назначая событие объекта datalink OnDataChange. В методе DataChange добавьте код для отображения данных поля актуальной базы данных, связь с которой обеспечивает объект datalink. В нашем примере мы присваиваем значение поля FDataLink заголовку панели:

```
private
...

  procedure DataChange(Sender: TObject);
...

implementation
...
```

```
procedure TDBPanel.DataChange(Sender: TObject);
begin
  if FDataLink.Field = nil then Caption := ''
  else Caption := FDataLink.Field.AsString;
end;
```

- Перекройте метод конструктора компонента Create. При реализации Create создайте объект FDataLink и назначьте частный метод DataChange событию FDataLink OnDataChange:

```
public
  constructor Create(AOwner: TComponent); override;
...
implementation
...
constructor TDBPanel.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  FDataLink := TFieldDataLink.Create;
  FDataLink.OnDataChange := DataChange;
end;
```

- Перекройте метод деструктора компонента Destroy. При реализации Destroy, установите OnDataChange в nil (чтобы избежать GPF) и освободите FDatalink:

```
public
...
  destructor Destroy; override;
...
implementation
...
destructor TDBPanel.Destroy;
begin
  FDataLink.OnDataChange := nil;
  FDataLink.Free;
  inherited Destroy;
end;
```

- Сохраните модуль и установите компонент (см. документацию «Users Guide» и «Component Writers Guide» для получения дополнительной информации по сохранению модулей и установке компонентов).
- Для тестирования функциональности компонента расположите на форме компоненты TTable, TDatasource, TDBNavigator и TDBPanel. Установите TTable DatabaseName и Tablename в 'DBDemos' и 'BioLife', а свойство Active в True. Установите свойство TDatasource Dataset в Table1. Установите TDBNavigator и свойство TDBPanel DataSource в Datasource1. Имя TDBpanel DataField должно быть установлено в 'Common_Name'. Запустите приложение и, используя навигатор и перемещаясь по записям, убедитесь в том, что TDBPanel обнаруживает изменение данных и отображает значение соответствующего поля.

Полный код компонента:

```
unit Mydbp;

interface

uses
  Windows, SysUtils, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, DB, DBTables;

type
  TDBPanel = class(TCustomPanel)

  private
    FDataLink: TFieldDataLink;
    function GetDataField: String;
    function GetDataSource: TDataSource;
    procedure SetDataField(Const Value: string);
    procedure SetDataSource(Value: TDataSource);
    procedure DataChange(Sender: TObject);
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
  published
    property DataField: string read GetDataField write SetDataField;
    property DataSource: TdataSource read GetDataSource write SetDataSource;
  end;

procedure Register;
implementation

procedure Register;
begin
  RegisterComponents('Samples', [TDBPanel]);
end;

function TDBPanel.GetDataField: String;
begin
  Result := FDataLink.FieldName;
end;

function TDBPanel.GetDataSource: TDataSource;
begin
  Result := FDataLink.DataSource;
end;

procedure TDBPanel.SetDataField(Const Value: string);
begin
  FDataLink.FieldName := Value;
end;
```

```
procedure TDBPanel.SetDataSource(Value: TDataSource);
begin
  FDataLink.DataSource := Value;
end;

procedure TDBPanel.DataChange(Sender: TObject);
begin
  if FDataLink.Field = nil then Caption := ''
  else Caption := FDataLink.Field.AsString;
end;

constructor TDBPanel.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  FDataLink := TFieldDataLink.Create;
  FDataLink.OnDataChange := DataChange;
end;

destructor TDBPanel.Destroy;
begin
  FDataLink.Free;
  FDataLink.OnDataChange := nil;
  inherited Destroy;
end;

end.
```

Примечание

Поскольку класс TFieldDataLink *в современных версиях Delphi определен в модуле* DBCtrls.pas*, этот модуль необходимо добавить в секцию* uses.

Позиция курсора в TEdit

Как получить позицию курсора в TEdit*?*

Используйте свойство SelStart.

Файл типа TList

Обратите внимание на методы объекта ReadData и WriteData, предназначенные для записи на диск, и методы SaveToFile и LoadFromFile самого TList. Правильным было бы сделать их более совместимыми. То есть, TList должен был бы восстанавливать/сохранять любой объект с помощью метода ReadData/WriteData.

```
unit Charactr;

interface
```

```
uses
  Graphics, StdCtrls, Classes, Sysutils, Winprocs, Ohmap, ohmstuff;

type
  TMapCharacterList = class(TList)
  private
    FMap: TOverHeadMap;
  public
    destructor Destroy; override;
    procedure RenderVisibleCharacters; virtual;
    procedure Savetofile(const filename: String);
    procedure Loadfromfile(const filename: String);
    procedure Clear;
    property MapDisp: TOverHeadMap read FMap write FMap;
end;
  TFrameStore = class(TList)
    procedure WriteData(Writer: Twriter); virtual;
    procedure ReadData(Reader: TReader); virtual;
    procedure Clear;
  end;
  TMapCharacter = class(TPersistent)
  private
    FName: string;
    FMap: TOverHeadMap;
    FFrame: Integer;
    FFramebm: TBitmap;
    FFrameMask: TBitmap;
    FWorkBuf: TBitmap;
    FFrameStore: TFrameStore;
    FMaskStore: TFrameStore;
    FXpos: Integer;
    FYpos: Integer;
    FZpos: Integer;
    FTransColor:TColor;
    FVisible: Boolean;
    FFastMode: Boolean;
    FIsClone: Boolean;
    FRedrawBackground: Boolean;
    function GetOnScreen: Boolean;
    procedure SetFrame(num: Integer);
    procedure SetVisible(vis: Boolean);
    procedure MakeFrameMask(trColor: TColor);
    procedure MakeFrameMasks; { Для переключения в быстрый режим... }
    procedure ReplaceTransColor(trColor: TColor);
    procedure SetXPos(x: Integer);
    procedure SetYPos(y: Integer);
    procedure SetZPos(z: Integer);
    procedure SetFastMode(fast: Boolean);
  public
    constructor Create(ParentMap: TOverheadmap); virtual;
    destructor Destroy; override;
```

```
    property Name: string read FName write FName;
    property Fastmode: Boolean read FFastMode write SetFastMode;
    property FrameStore: TFrameStore read FFrameStore write FFramestore;
    property MaskStore: TFrameStore read FMaskStore write FMaskStore;
    property Frame: integer read FFrame write SetFrame;
    property Framebm: TBitmap read FFramebm;
    property FrameMask: TBitmap read FFrameMask;
    property TransColor: TColor read FTransColor write FTransColor;
    property Xpos: Integer read FXpos write SetXpos;
    property YPos: Integer read FYpos write SetYpos;
    property ZPos: Integer read FZpos write SetZpos;
    property Map: TOverHeadMap read FMap write FMap;
    property OnScreen: Boolean read GetOnScreen;
    property Visible: Boolean read FVisible write SetVisible;
    property IsClone: Boolean read FIsClone write FIsClone;
    property RedrawBackground: Boolean read FredrawBackground
                                       write FRedrawBackground;
    procedure Render; virtual;
    procedure RenderCharacter(mapcoords: Boolean; cxpos, cypos: Integer;
                              mask, bm, wb: TBitmap); virtual;
    procedure Clone(Source: TMapCharacter); virtual;
    procedure SetCharacterCoords(x, y, z: Integer); virtual;
    procedure WriteData(Writer: Twriter); virtual;
    procedure ReadData(Reader: TReader); virtual;
  end;

implementation
constructor TMapCharacter.Create(ParentMap: TOverheadmap);
begin
  inherited Create;
  FIsClone := False;
  FFramebm := TBitMap.create;
  FFrameMask := TBitmap.Create;
  FWorkbuf := TBitMap.Create;
  if not(FIsClone) then FFrameStore := TFrameStore.Create;
  FTransColor := clBlack;
  FFastMode := False;
  FMap := ParentMap;
end;

destructor TMapCharacter.Destroy;
var
  a, b: Integer;
begin
  FFramemask.free;
  FFramebm.free;
  FWorkBuf.Free;
  if not(FIsClone) then begin
    FFrameStore.Clear;
    FFrameStore.free;
  end;
```

```
  if (MaskStore <> nil) and not(FIsClone) then begin
    MaskStore.Clear;
    MaskStore.Free;
  end;
  inherited Destroy;
end;

{ Данная процедура присваивает себе значения свойств объекта, заданного
  в параметре Source }
procedure TMapCharacter.Clone(Source: TMapCharacter);
begin
  FName := Source.Name;
  FFastMode := Source.FastMode;
  FFrameStore := Source.FrameStore;
  FMaskStore := Source.MaskStore;
  FTransColor := Source.TransColor;
  FMap := Source.Map;
  FVisible := False;
  Frame := Source.Frame;              { Ищем фрейм триггера }
  FIsClone := True;
end;

procedure TMapCharacter.SetXPos(x: Integer);
begin
  Map.Redraw(xpos, ypos, zpos, -1);
  FXpos := x;
  Render;
end;

procedure TMapCharacter.SetYPos(y: Integer);
begin
  Map.Redraw(xpos, ypos, zpos, -1);
  FYPos := y;
  Render;
end;

procedure TMapCharacter.SetZPos(z: Integer);
begin
  Map.Redraw(xpos, ypos, zpos, -1);
  FZpos := z;
  Render;
end;

procedure TMapCharacter.SetCharacterCoords(x, y, z: Integer);
begin
  Map.Redraw(xpos, ypos, zpos, -1);
  Fxpos := x;
  Fypos := y;
  Fzpos := z;
  Render;
end;
```

```
procedure TMapCharacter.SetFrame(num: Integer);
begin
  if (num <= FFrameStore.Count - 1) and (num > -1) then begin
    FFrame := num;
    FFramebm.Assign(TBitmap(FFrameStore.Items[num]));
    if Ffastmode = False then begin
      FFrameMask.Width := FFramebm.Width;
      FFrameMask.Height := FFramebm.Height;
      FWorkBuf.Height := FFramebm.Height;
      FWorkBuf.Width := FFramebm.Width;
      makeframemask(TransColor);
      replacetranscolor(TransColor);
    end else begin
      FWorkBuf.Height := FFramebm.Height;
      FWorkBuf.Width := FFramebm.Width;
      FFrameMask.Assign(TBitmap(FMaskStore.Items[num]));
    end;
  end;
end;

procedure TMapCharacter.MakeFrameMask(trColor: TColor);
var
  testbm1, testbm2: TBitmap;
  trColorInv: TColor;
begin
  testbm1 := TBitmap.Create;
  testbm1.width :=  1;
  testbm1.height := 1;
  testbm2 :=  TBitmap.Create;
  testbm2.width := 1;
  testbm2.height:= 1;
  testbm1.Canvas.Pixels[0, 0]:= trColor;
  testbm2.Canvas.CopyMode:= cmSrcInvert;
  testbm2.Canvas.Draw(0, 0, testbm1);
  trColorInv:= testbm2.Canvas.Pixels[0, 0];
  testbm1.free;
  testbm2.free;
  with FFrameMask.Canvas do begin
    Brush.Color:=  trColorInv;
    BrushCopy(Rect(0, 0, FFrameMask.Width, FFrameMask.Height), FFramebm,
              Rect(0, 0, FFramebm.Width, FFramebm.Height), trColor);
    CopyMode := cmSrcInvert;
    Draw(0, 0, FFramebm);
  end;
end;

procedure TMapCharacter.ReplaceTransColor(trColor: TColor);
begin
  with FFramebm.Canvas do begin
    CopyMode := cmSrcCopy;
    Brush.Color := clBlack;
```

```
    BrushCopy(Rect(0, 0, FFramebm.Width, FFramebm.Height), FFramebm,
              Rect(0, 0, FFramebm.Width, FFramebm.Height), trColor);
  end;
end;

function TMapCharacter.GetOnScreen: Boolean;
var
  dispx, dispy: Integer;
begin
  dispx := Map.width div map.tilexdim;
  dispy := Map.height div map.tileydim;
  if (xpos >= Map.xpos) and (xpos <= map.xpos + dispx) and (ypos >= map.ypos)
      and (ypos >= map.ypos + dispy) then
    result := True;
end;

procedure TMapCharacter.SetVisible(vis: Boolean);
begin
  if vis and OnScreen then Render;
  FVisible := vis;
end;

procedure TMapCharacter.SetFastMode(fast: Boolean);
begin
  if fast <> FFastMode then begin
    if fast = True then begin
      FMaskStore := TFrameStore.Create;
      MakeFrameMasks;
      FFastMode := True;
      frame := 0;
    end else begin
      FMaskStore.Free;
      FFastMode := False;
    end;
  end;
end;

procedure TMapCharacter.MakeFrameMasks;
var
  a: Integer;
  bm: TBitMap;
begin
  if FFrameStore.Count > 0 then begin
    for a := 0 to FFrameStore.Count - 1 do begin
      Frame := a;
      bm := TBitMap.Create;
      bm.Assign(FFrameMask);
      FMaskStore.add(bm);
    end;
  end;
end;
```

```
procedure TMapCharacter.Render;
var
  x, y: Integer;
begin
  if visible and onscreen then
    RenderCharacter(True, xpos, ypos, FFramemask, FFramebm, FWorkbuf);
end;

procedure TMapCharacter.RenderCharacter(mapcoords: Boolean;
                        cxpos, cypos: Integer; mask, bm, wb: TBitmap);
var
  x, y: Integer;
begin
  if map.ready then begin

{ Если пользователь определил это в mapcoords, то в первую очередь перерисовываем
  секцию(и). Если нет, делает это он }
    if mapcoords then begin
      if FRedrawBackground then
        Map.redraw(cxpos, cypos, FMap.zpos, -1);
      wb.Canvas.Draw(0, 0,
                TMapIcon(FMap.Iconset[map.zoomlevel].items[FMap.Map.Iconat(cxpos,
                cypos, Map.zpos)]).image);
      x:= (cxpos - Map.xpos) * FMap.tilexdim;
      y:= (cypos - Map.ypos) * FMap.tileydim;
    end else
      wb.Canvas.Copyrect(rect(0, 0, FMap.tilexdim, FMap.tileydim),
                         FMap.Screenbuffer.Canvas, Rect(x, y, x + FMap.tilexdim,
                         y + FMap.tileydim));
    with wb do begin
      Map.Canvas.CopyMode := cmSrcAnd;
      Map.Canvas.Draw(0, 0, Mask);
      Map.Canvas.CopyMode := cmSrcPaint;
      Map.Canvas.Draw(0, 0, bm);
      Map.Canvas.Copymode:= cmSrcCopy;
    end;
    Map.Canvas.CopyRect(Rect(x, y, x + FMap.tilexdim, y + FMap.tileydim),
                        wb.canvas, Rect(0, 0, FMap.tilexdim, FMap.tileydim));
  end;
end;

procedure TMapCharacter.WriteData(Writer: TWriter);
begin
  with Writer do begin
    WriteListBegin;
    WriteString(FName);
    WriteBoolean(FFastMode);
    WriteInteger(TransColor);
    FFrameStore.WriteData(Writer);
    if FFastMode then
      FMaskStore.WriteData(Writer);
```

```
    WriteListEnd;
  end;
end;

procedure TMapCharacter.ReadData(Reader: TReader);
begin
  with Reader do begin
    ReadListBegin;
    Fname := ReadString;
    FFastMode := ReadBoolean;
    TransColor := ReadInteger;
    FFrameStore.ReadData(Reader);
    if FFastMode then begin
      FMaskStore := TFrameStore.Create;
      FMaskStore.ReadData(Reader);
    end;
    ReadListEnd;
  end;
end;

procedure TMapCharacterList.RenderVisibleCharacters;
var
  a: Integer;
begin
  for a := 0 to count - 1 do
    TMapCharacter(Items[a]).Render;
end;

procedure TMapCharacterList.Clear;
var
  Obj: TObject;
begin
{ Этот код освобождает все ресурсы, присутствующие в списке }
  if Self.Count > 0 then begin
    repeat
      obj := Self.Items[0];
      Obj.Free;
      Self.Remove(Self.Items[0]);
    until Self.Count = 0;
  end;
end;

destructor TMapCharacterList.Destroy;
var
  a: Integer;
begin
  if count > 0 then
    for a := 0 to count - 1 do
      TObject(Items[a]).Free;
  inherited destroy;
end;
```

```
procedure TMapCharacterList.loadfromfile(const filename: string);
var
  i: Integer;
  Reader: Treader;
  Stream: TFileStream;
  obj: TMapCharacter;
begin
  stream:= TFileStream.Create(Filename, fmOpenRead);
  try
    reader := TReader.Create(Stream, $ff);
    try
      with reader do begin
        try
          ReadSignature;
          if ReadInteger <> $6667 then
            Raise EReadError.Create('Не список сиволов.');
        except
            Raise EReadError.Create('Неверный формат файла.');
        end;
        ReadListBegin;
        while not EndofList do begin
          obj := TMapCharacter.create(FMap);
          try
            obj.ReadData(reader);
          except
            obj.free;
            raise EReadError.Create('Ошибка в файле списка символов.');
          end;
          Self.Add(obj);
        end;
        ReadListEnd;
      end;
    finally
      Reader.Free;
    end;
  finally
    Stream.Free;
  end;
end;

procedure TMapCharacterList.SaveToFile(const filename: String);
var
  Stream: TFileStream;
  Writer: TWriter;
  i: Integer;
  obj: TMapCharacter;
begin
  stream := TFileStream.Create(filename, fmCreate or fmOpenWrite);
  try
    writer := TWriter.Create(Stream, $ff);
    try
```

```
      with writer do begin
        WriteSignature; WriteInteger($6667); WriteListBegin;
        for i := 0 to Self.Count - 1 do
          TMapCharacter(Self.Items[i]).WriteData(writer); WriteListEnd;
      end;
    finally
      writer.free;
    end;
  finally
    Stream.Free;
  end;
end;

procedure TFrameStore.WriteData(Writer: TWriter);
var
  mstream: TMemoryStream;
  a, size: Longint;
begin
  mstream := TMemoryStream.Create;
  try
    with writer do begin
      WriteListBegin;
      WriteInteger(count);
      for a := 0 to count - 1 do begin
        TBitmap(Items[a]).SaveToStream(mstream);
        size := mstream.size;
        WriteInteger(size);
        Write(mstream.memory^, size);
        mstream.position := 0;
      end;
      WriteListEnd;
    end;
  finally
    Mstream.free;
  end;
end;

procedure TFrameStore.ReadData(Reader: TReader);
var
  mstream: TMemoryStream;
  a, listcount, size: Longint;
  newframe: TBitMap;
begin
  mstream := TMemoryStream.Create;
  try
    with reader do begin
      ReadListBegin;
      Listcount := ReadInteger;
      for a := 1 to listcount do begin
        size := ReadInteger;
        mstream.setsize(size);
```

```
          read(mstream.Memory^, size);
          newframe := TBitmap.Create;
          newframe.loadfromstream(mstream);
          add(newframe);
        end;
        ReadListEnd;
      end;
    finally
      Mstream.Free;
    end;
  end;

  procedure TFrameStore.Clear;
  var
    Obj: TObject;
  begin
  { Этот код освобождает все ресурсы, присутствующие в списке }
    if Self.Count > 0 then begin
      repeat
        obj := Self.Items[0];
        obj.free;
        Self.remove(Self.items[0]);
      until Self.count = 0;
    end;
  end;
  end.
```

[News Group]

Примечание

Написано для Delphi 1

Сохранение содержимого TreeView

Сохраняем:

```
TreeView1.SaveToFile('Filename');
```

Читаем:

```
TreeView1.LoadFromFile('Filename');
```

Данный метод сохраняет только имена элементов и структуру (при сохранении файл получается текстовым, с отражением структуры). Метод не сохраняет свойство `ImageIndex` и другие. После выполнения `LoadFromFile` необходимо восстановить исходные пиктограммы.

Использование шрифта в TreeView

Как сделать, чтобы в TreeView *текущий узел выделялся другим шрифтом, например, с подчеркиванием и т. д.?*

Решение

```
procedure TForm1.TreeView1CustomDrawItem(Sender: TCustomTreeView; Node: TTreeNode;
                                State: TCustomDrawState; var DefaultDraw: Boolean);
begin
  if cdsSelected in State then
    Sender.Canvas.Font.Style := Sender.Canvas.Font.Style + [fsUnderline];
end;
```

TImage – эффект плавного перехода

Существует ли для этого эффекта какой-либо алгоритм генерации изображений вместо использования кисточки?

Решение

```
procedure WaitAWhile(n: longint);
var
  StartTime: longint;
begin
  StartTime := timeGetTime;
  while timeGetTime < StartTime + n do;
end;

procedure TForm1.Image1Click(Sender: TObject);
var
  BrushBmp, BufferBmp, Buffer2Bmp, ImageBmp, Image2Bmp: TBitmap;
  j, k, row, col: longint;
begin
  row := 0;
  col := 0;
  BrushBmp := TBitmap.Create;
  with BrushBmp do begin
    Monochrome := False;
    Width := 8;
    Height := 8;
  end;
  imageBmp := TBitmap.Create;
  imagebmp. LoadFromFile('c:\huh.bmp');
  image2bmp := TBitmap.Create;
  image2bmp.LoadFromFile('c:\whatsis.bmp');

{ При 256 цветах лучше иметь ту же самую палитру! }
  BufferBmp := TBitmap.Create;
  with BufferBmp do begin
    Height := 200;
    Width := 200;
```

```
    canvas.brush.bitmap := TBitmap.Create;
  end;
  Buffer2Bmp := TBitmap.Create;
  with Buffer2Bmp do begin
    Height := 200;
    Width := 200;
    canvas.brush.bitmap := TBitmap.Create;
  end;
  for k := 1 to 16 do begin
    WaitAWhile(0);            { Для пентиума необходимо добавить задержку }
    for j := 0 to 3 do begin
      row := (row + 5) mod 8;
      col := (col + 1) mod 8;
      if row = 0 then col := (col + 1) mod 8;
      BrushBmp.canvas.Pixels[row, col] := clBlack;
    end;
    with BufferBmp do begin
      canvas.copymode := cmSrcCopy;
      canvas.brush.bitmap.free;
      canvas.brush.style := bsClear;
      canvas.brush.bitmap := TBitmap.Create;
      canvas.brush.bitmap.Assign(BrushBmp);
      canvas.Rectangle(0, 0, 200, 200);
      canvas.CopyMode := cmMergeCopy;
      canvas.copyrect(rect(0, 0, 200, 200), imageBmp.canvas, rect(0, 0, 200, 200));
    end;
    with Buffer2Bmp do begin
      canvas.copymode := cmSrcCopy;
      canvas.brush.bitmap.free;
      canvas.brush.style := bsClear;
      canvas.brush.bitmap := TBitmap.Create;
      canvas.brush.bitmap.Assign(BrushBmp);
      canvas.Rectangle(0, 0, 200, 200);
      canvas.copymode := cmSrcErase;
      canvas.copyrect(rect(0, 0, 200, 200), image2bmp.canvas,
                      rect(0, 0, 200, 200));
    end;
    BufferBmp.Canvas.CopyMode := cmSrcPaint;
    BufferBmp.Canvas.Copyrect(rect(0, 0, 200, 200), Buffer2Bmp.Canvas,
                              rect(0, 0, 200, 200));
    canvas.copymode := cmSrcCopy;
    canvas.copyrect(rect(0, 0, 200, 200), BufferBmp.Canvas, rect(0, 0, 200, 200));
  end;
  BufferBmp.canvas.brush.bitmap.free;
  Buffer2Bmp.canvas.brush.bitmap.free;
  BrushBmp.Free;
  BufferBmp.Free;
  Buffer2Bmp.Free;
  ImageBmp.Free;
  image2Bmp.Free;
end;
```

На быстрых процессорах можно использовать большее количество кисточек. Изменим приведенные выше строки на следующие:

```
  ...
for k := 1 to 64 do begin
  WaitAWhile(50);
  for j := 0 to 0 do
  ...
```

За счет указанной задержки и достигается плавный переход.

Заполняя кисть в другом порядке, можно получить ряд других эффектов, но приведенная выше версия единственная, которая позволяет максимально «приблизиться» к эффекту перехода, но вы можете, скажем, написать:

```
begin
  row := (row + 1) mod 8;
{ col := (col + 1) mod 8; }
  if row = 0 then col := (col + 1) mod 8;
  ...
```

и получить другой своеобразный эффект перехода.

[News Group]

Примечание

Необходимо добавить модуль `MMSystem` *в секцию* `uses` *(для* `TimeGetTime`*).*

TOutline – чтение из файла

Если вы хотите сохранять `TOutline`, обратите внимание на методы `SaveToFile` и `ReadFromFile`. Если же вы хотите создать собственный файл (для хранения данных, связанных с вашим `TOutline`), рекомендуем воспользоваться `TStream` или его потомком – `TFileStream`.

Для этой цели идеально подходит создание записи следующего вида:

```
TSaveNode = record
  Text: String
  Index: Longint;
  Parent: Longint;
  Data: Pointer
end;
```

Это вся информация о `TOutLine`, которую нужно сохранить. Вы можете сохранить ее, пройдя в цикле через все узлы `TOutlineNodes` и записывая их в поток. Для загрузки всего файла читайте запись за записью и используйте метод `TOutline.AddChild`. `TRecord` при этом будет содержать всю необходимую информацию.

TOutline – Drag & Drop

Решение 1

Необходимо перехватывать в TOutline сообщение WM_DropFiles. Для этого надо создать его потомка. Также следует убедиться в том, что дескриптор TOutline.Handle хотя бы раз передавался в качестве параметра функции DragAcceptFiles. Для определения положения мыши в момент перетаскивания используется DragQueryPoint.

Решение 2

Проблема заключается в том, что прежде чем Windows сможет обработать сообщение WM_MouseUp, курсор мыши передвинется дальше. Вот решение этой головоломки:

- Разрешите Windows как можно скорее обработатывать события мыши:

```
OnMouseDown:
BeginDrag(False);
while ... do
  begin
    Application.ProccessMessages; { это позволяет Windows обработать }
                                  { все сообщения за один шаг }
  end;
```

Примечание

Обратите пристальное внимание на то, что при создании цикла, если используется цикл while*, необходимо предусмотреть возможность выхода из него, например, при закрытии приложения или других действиях пользователя, требующих экстренного выхода из тела цикла.*

- Аналогично:

```
OnMouseDown:
BeginDrag(False);
Application.ProccessMessages;
while ... do
  begin
   { единственный шаг обработки }
  end;
```

- Переместите вызов BeginDrag в обработчик события OnMouseMove.

Решение 3

Установите DragMode = dmManual, создайте OnMouseDownHandler, внутри обработчика осуществите вызов BeginDrag(False), который в действительности не начинает перемещение, пока пользователь не переместит объект больше чем на 5 пикселов, так что если пользователь просто щелкнет по компоненту, операция перетаскивания даже не начнется.

Компонент HTML-редактора

Если используется MS Internet Explorer 5, то редактор ваш. Дело в том, что вместе с броузером поставляется DHTML – элемент управления ActiveX, представляющий собой редактор HTML и поддерживающий стили CSS. Находится в C:\Program Files\Common Files\Microsoft Shared\Triedit. В Delphi устанавливается через меню Components/Import ActiveX Control. При распространении своей программы с этим компонентом не забудьте, что помимо копирования на машину клиента всех файлов из приведенного выше каталога нужно, чтобы они были зарегистрированы.

Canvas и освобождение дескрипторов

TCanvas автоматически не вызывает ReleaseDC. При создании холста с WindowDC в качестве дескриптора лучшей идеей будет создание потомка TCanvas (моделированного с TControlCanvas):

```
type
  TWindowCanvas = class(TCanvas)
  private
    FWinControl: TWinControl;
    FDeviceContext: HDC;
    procedure SetWinControl(AControl: TWinControl);
  protected
    procedure CreateHandle; override;
  public
    destructor Destroy; override;
    procedure FreeHandle;
    property WinControl: TWinControl read FWinControl write SetWinControl;
  end;

implementation
destructor TWindowCanvas.Destroy;
begin
  FreeHandle;
  inherited Destroy;
end;

procedure TWindowCanvas.CreateHandle;
begin
  if FControl = nil then inherited CreateHandle
  else begin
    if FDeviceContext = 0 then FDeviceContext := GetWindowDC(WinControl.Handle);
    Handle := FDeviceContext;
  end;
end;

procedure TControlCanvas.FreeHandle;
begin
  if FDeviceContext <> 0 then begin
```

```
      Handle := 0;
      ReleaseDC(WinControl.Handle, FDeviceContext);
      FDeviceContext := 0;
    end;
  end;

  procedure TControlCanvas.SetWinControl(AControl: TWinControl);
  begin
    if FWinControl <> AControl then begin
      FreeHandle;
      FWinControl := AControl;
    end;
  end;
```

Очевидно, необходимо следить за ситуацией и разрушать TWindowCanvas (или освобождать дескриптор) перед тем, как уничтожить элемент управления, связанный с ним. Также, имейте в виду, что дескриптор DeviceContext не освобождается автоматически после обработки каждого сообщения (как это происходит с дескрипторами TControlCanvas). Для освобождения дескриптора следует явно вызвать FreeHandle (или разрушить Canvas). И, наконец, имейте в виду, что WindowCanvas.Handle := 0 не освобождает дескриптор, для его освобождения вы должны вызывать FreeHandle.

Определение свойства объекта

```
TypInfo.GetPropInfo(My_Component.ClassInfo, 'Hint') <> nil
```

Таким образом можно узнать наличие публикуемых свойств. Если это не приведет к положительному результату, обратитесь к FieldAddress. Однако этот метод дает адрес полей, которые перечисляются сразу после объявления класса.

Модифицировать значение можно посредством прямой записи по адресу FieldAddress либо при помощи TypeInfo.

10

Классы

Поиск класса

Во время выполнения приложения можно определить, существует ли класс с именем TLog. Для этого в модуле TLog используйте RegisterClass(TLog) или потомка TLog, затем FindClass('TLog') или FindClass('TLogSubclass') для получения ссылки на класс в вызывающем модуле, обеспечивая тем самым возможность работы с объектами данного класса. Также можно добавить классовый метод, который либо возвращает существующий экземпляр или NIL, либо создает и возвращает новый экземпляр при отсутствии текущего.

Самое необходимое, что нужно сделать, это создать абстрактный, чисто виртуальный базовый класс TLog и подкласс TLogSubclass с необходимыми свойствами. Вызывающему оператору необходимо знать всего лишь о TLog, а не о TLogSubclass, чтобы получить доступ к методам и свойствам последнего.

Тем не менее, технология поиска класса по его имени – не лучшее решение. Вот что предлагается:

```
unit LogUnit;
interface

type
  TLog = class
  public
    constructor Create;
    procedure LogMessage(const Message: string); virtual; abstract;
  end;

var
  Log: TLog;
```

```
implementation

constructor TLog.Create;
begin
  Log := Self;
end;

procedure TidyUp; far;
begin
  Log.Free;
end;

initialization
  AddExitProc(TidyUp);
end.

unit LogImpl;

interface
implementation

uses Log;

type
  TLogImplementation = class(TLog)
  public
    procedure LogMessage(const Message: string); override;
  end;

procedure TLogImplementation.LogMessage(const Message: string);
begin
  { записываем сообщение в журнальный файл }
end ;

initialization
  TLogImplementation.Create;

end.
```

Обратите внимание, что здесь используются «скрытые» данные – класс TLogImplementation объявлен внутри секции реализации модуля LogImpl, поэтому никакой другой модуль их не видит. Фактически, интерфейсная часть пуста. Можно убедиться в этом, изучив Log и увидев NIL в самом начале.

Кроме того, можно иметь ничего не делающий TLog.LogMessage. Затем создать экземпляр TLog в секции инициализации модуля LogUnit.pas и освобождать его перед созданием экземпляра TLogImplementation в LogImpl.pas. Таким образом, для подключения к приложению класса нужно просто добавить к проекту модуль LogImpl.

[News Group]

Создание синего экрана установки

Пример градиентной заливки:

```
procedure TForm1.FormPaint(Sender: TObject);
var
  Row, Ht: Word ;
begin
  Ht := (ClientHeight + 255) div 256;
  for Row := 0 to 255 do
    with Canvas do begin
      Brush.Color := RGB(0, 0, Row) ;
      FillRect(Rect(0, Row * Ht, ClientWidth, (Row + 1) * Ht));
    end;
end;
```

Отображение логотипа при запуске приложения

Решение

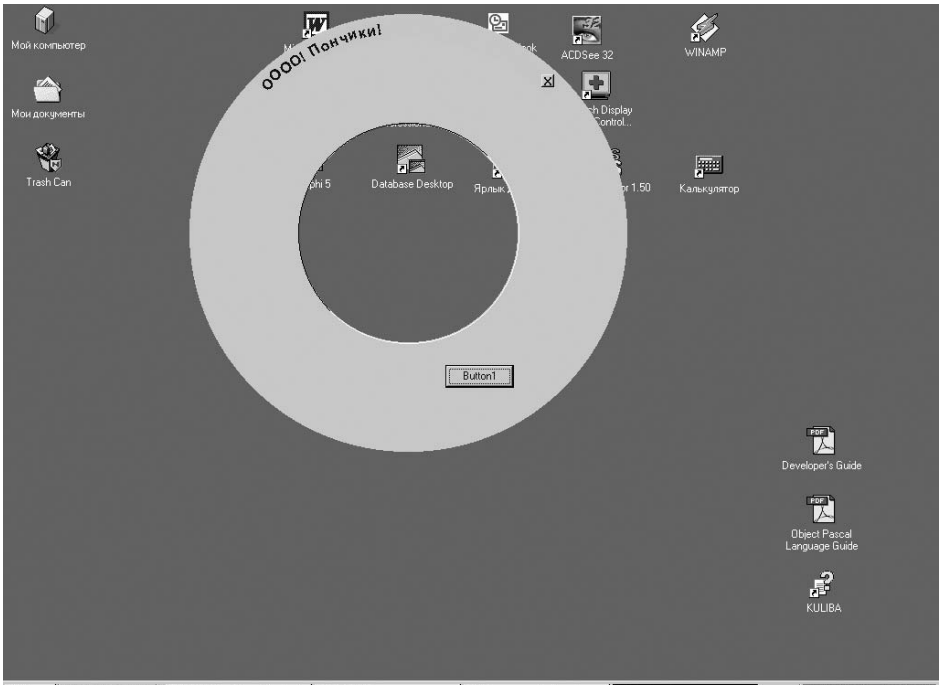
```
program Project1;

uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};
  ULogo in 'ULogo.pas' {LogoForm};

{$R *.RES}

begin
  Application.Initialize; {до этого момента никаких изменений}
  with TLogoForm.Create(Application) do
  try
    Show;
    Update;
    Application.CreateForm(TForm1, Form1);
{ GProgress.AddProgress(1); - здесь можно использовать индикатор выполнения,
  если TGauge или TProgressBar лежат на TLogoForm, если есть еще формы,
  то Application.CreateForm(TForm2, Form2); и т.д.}
  finally
    Free;
  end;
  Application.Run;
end.
```

[Алексей]

Круглый логотип при запуске приложения

Код позволяет вывести окно-кольцо с полукруглой областью заголовка. Сквозь отверстие в форме можно увидеть и запустить щелчком мыши другие

приложения! Создайте новый проект и сохраните главный модуль с именем RGNU.PAS. Используйте следующий код:




```
unit rgnu;
interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Buttons, Menus, StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure FormPaint(Sender: TObject);
  private
    rTitleBar: THandle;
    Center: TPoint;
    CapY: Integer;
    SB1: TSpeedButton;
    RL, RR: Double;
    procedure TitleBar(Act: Boolean);
    procedure WMNCHITTEST(var Msg: TWMNCHitTest); message WM_NCHITTEST;
    procedure WMNCACTIVATE(var Msg: TWMNCACTIVATE); message WM_NCACTIVATE;
    procedure WMSetText(var Msg: TWMSetText); message WM_SETTEXT;
  end;
```

```
var
  Form1: TForm1;

implementation

{$R *.DFM}

const
  TitlColors: ARRAY[Boolean] OF TColor = (clInactiveCaption, clActiveCaption);
  TxtColors: ARRAY[Boolean] OF TColor = (clInactiveCaptionText, clCaptionText);

procedure TForm1.FormCreate(Sender: TObject);
var
  rTemp, rTemp2: THandle;
  Vertices: array[0..2] of TPoint;
  X, Y: integer;
begin
  Caption := 'OOOO! Пончики!';
  BorderStyle := bsNone;           { требуется }
  if Width > Height then Width := Height
  else Height := Width;            { труднее вычислить, если width <> height }
  Center := Point(Width div 2, Height div 2);
  CapY := GetSystemMetrics(SM_CYCAPTION) + 8;
  rTemp := CreateEllipticRgn(0, 0, Width, Height);
  rTemp2 := CreateEllipticRgn((Width div 4), (Height div 4), 3 * (Width div 4),
                              3 * (Height div 4));
  CombineRgn(rTemp, rTemp, rTemp2, RGN_DIFF);
  SetWindowRgn(Handle, rTemp, True);
  DeleteObject(rTemp2);
  rTitleBar := CreateEllipticRgn(4, 4, Width - 4, Height - 4);
  rTemp := CreateEllipticRgn(CapY, CapY, Width - CapY, Height - CapY);
  CombineRgn(rTitleBar, rTitleBar, rTemp, RGN_DIFF);
  Vertices[0] := Point(0, 0);
  Vertices[1] := Point(Width, 0);
  Vertices[2] := Point(Width div 2, Height div 2);
  rTemp := CreatePolygonRgn(Vertices, 3, ALTERNATE);
  CombineRgn(rTitleBar, rTitleBar, rTemp, RGN_AND);
  DeleteObject(rTemp);
  RL := ArcTan(Width / Height);
  RR := -RL + (22 / Center.X);
  X := Center.X - Round((Center.X - 1 - (CapY div 2)) * Sin(RR));
  Y := Center.Y - Round((Center.Y - 1 - (CapY div 2)) * Cos(RR));
  SB1 := TSpeedButton.Create(Self);
  with SB1 do begin
    Parent := Self;
    Left := X;
    Top := Y;
    Width := 14;
    Height := 14;
    OnClick := Button1Click;
    Caption := 'X';
```

```
    Font.Style := [fsBold];
  end;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  Close;
end;

procedure TForm1.WMNCHITTEST(var Msg: TWMNCHitTest);
begin
  Inherited;
  with Msg do
    with ScreenToClient(Point(XPos,YPos)) do
      if PtInRegion(rTitleBar, X, Y)
          and (not PtInRect(SB1.BoundsRect, Point(X,Y))) then
        Result := htCaption;
end;

procedure TForm1.WMNCActivate(var Msg: TWMncActivate);
begin
  Inherited;
  TitleBar(Msg.Active);
end;

procedure TForm1.WMSetText(var Msg: TWMSetText);
begin
  Inherited;
  TitleBar(Active);
end;

procedure TForm1.TitleBar(Act: Boolean);
var
  TF : TLogFont;
  R : Double;
  N, X, Y: Integer;
begin
  if Center.X = 0 then Exit;
  with Canvas do begin
    Brush.Style := bsSolid;
    Brush.Color := TitlColors[Act];
    PaintRgn(Handle, rTitleBar);
    R := RL;
    Brush.Color := TitlColors[Act];
    Font.Name := 'Arial';
    Font.Size := 12;
    Font.Color := TxtColors[Act];
    Font.Style := [fsBold];
    GetObject(Font.Handle, SizeOf(TLogFont), @TF);
    for N := 1 to Length(Caption) do begin
      X := Center.X - Round((Center.X - 6) * Sin(R));
```

```
        Y := Center.Y - Round((Center.Y - 6) * Cos(R));
        TF.lfEscapement := Round(R * 1800 / pi);
        Font.Handle := CreateFontIndirect(TF);
        TextOut(X, Y, Caption[N]);
        R := R - (((TextWidth(Caption[N]))+2) / Center.X);
        if R < RR then Break;
      end;
      Font.Name := 'MS Sans Serif';
      Font.Size := 8;
      Font.Color := clWindowText;
      Font.Style := [];
    end;
  end;

  procedure TForm1.FormPaint(Sender: TObject);
  begin
    with Canvas do begin
      Pen.Color := clBlack;
      Brush.Style := bsClear;
      Pen.Width := 1;
      Pen.Color := clWhite;
      Arc(1, 1, Width - 1, Height - 1, Width, 0, 0, Height);
      Arc((Width div 4) - 1, (Height div 4) - 1, 3 * (Width div 4) + 1,
          3 * (Height div 4) + 1, 0, Height, Width, 0);
      Pen.Color := clBlack;
      Arc(1, 1, Width - 1, Height - 1, 0, Height, Width, 0);
      Arc((Width div 4) - 1, (Height div 4) - 1, 3 * (Width div 4) + 1,
          3 * (Height div 4) + 1, Width, 0, 0, Height);
      TitleBar(Active);
    end;
  end;
  end.
```

Деактивация приложения

Если необходимо что-то сделать, когда приложение теряет фокус, используйте обработчик события `Application.OnDeactivate`. Добавьте следующую строку в обработчик формы `FormCreate`:

```
Application.OnDeactivate := AppDeactivate;
```

Затем создайте следующий метод:

```
procedure Form1.AppDeactivate(Sender: TObject);
begin
  ...
{ здесь ваш код}
  ...
end;
```

[News Group]

Невидимая главная форма

Я пытаюсь создать приложение, помещающее во время запуска пиктограмму в системную область панели задач с надлежащим контекстным меню. Тем не менее, приложение остается видимым в панели задач. Применение `Application.ShowMainForm := False` *оказывается недостаточным. Как сделать главную форму полностью невидимой?*

Маленький код, который справляется с этой проблемой.

```
procedure TMainForm.FormCreate(Sender: TObject);
begin
  Application.OnMinimize := AppMinimize;
  Application.OnRestore := AppMinimize;
  Application.Minimize;
  AppMinimize(@Self);
end;

procedure TMainForm.AppMinimize(Sender: TObject);
begin
  ShowWindow(Application.Handle, SW_HIDE);
end;
```

Приложения без форм

Почему нет? Откройте пункт меню File|New|Other... и выберите Console Application в диалоге New Items. Delphi создаст проект, который все еще представляет собой Windows-программу, но функции `WriteLn`, `ReadLn` уже работают так же, как и в сеансе DOS. Если хотите, удалите модуль SysUtils из списка используемых модулей.

Приложение продолжает оставаться приложением Windows, поэтому оно без проблем может обращаться к функциям Windows API. Для этого необходимо добавить в список используемых модулей модуль Windows.

Окно произвольной формы

Код, приведенный ниже, позволяет сделать форму круглой, без заголовка и рамки окна.

- Создайте новое приложение.
- Перепишите метод формы CreateParams как показано ниже.
- Разместите на форме какое-либо изображение и присвойте свойству `Transparent` значение `True`.
- Установите кнопку `SpeedButton` на форме и создайте метод, закрывающий приложение. (Форма получится круглой, поэтому системные кнопки и системное меню будут недоступны, они останутся за пределами контура формы.)
- Создайте метод, показанный ниже, отвечающий за обработку события `OnCreate`.

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls;

type
  TForm1 = class(TForm)
    Image1: TImage;
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    procedure CreateParams(var Params: TCreateParams); override;
  end;

var
  Form1: TForm1;

implementation
{$R *.DFM}

procedure TForm1.CreateParams(var Params: TCreateParams);
begin
  inherited CreateParams(Params);
{ Здесь убираем заголовки и границы формы }
  Params.Style := Params.Style or WS_POPUP xor WS_DLGFRAME;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  Formrgn: hrgn;
begin
{ очищаем форму }
  Form1.Brush.Style := bsclear;
{ делаем форму круглой }
  GetWindowRgn(Form1.Handle, FormRgn);
  DeleteObject(FormRgn);
  Formrgn:= CreateroundRectRgn(0, 0, Form1.Width, Form1.Width,
                               Form1.Width, Form1.Width);
  SetWindowRgn(Form1.Handle, Formrgn, True);
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  Close;
end;

end.
```

Окно без заголовка

Для создания окна без заголовка с любым стилем контура сделайте следующее:

Добавьте объявление процедуры:

```
procedure CreateParams(var Params: TCreateParams); override;
```

и ее реализацию:

```
procedure TForm1.CreateParams(var Params: TCreateParams);
begin
  inherited CreateParams(Params);
  with Params do Style := (Style OR WS_POPUP) AND NOT WS_DLGFRAME;
end;
```

Установите `BorderStyle` в `bsSizeable`.

Добавление пунктов в системное меню программы

Как добавить свой пункт в системное меню программы?

Пример демонстрирует применение API-функции `AppendMenu` и последующий перехват `WM_SYSCOMMAND` именно с этой целью.

```
type
  TForm1 = class(TForm)
      procedure FormCreate(Sender: TObject);
    private
      procedure WMSysCommand(var Msg: TWMSysCommand); message WM_SYSCOMMAND;
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}

const
  SC_MyMenuItem = WM_USER + 1;

procedure TForm1.FormCreate(Sender: TObject);
begin
  AppendMenu(GetSystemMenu(Handle, FALSE), MF_SEPARATOR, 0, '');
  AppendMenu(GetSystemMenu(Handle, FALSE), MF_STRING, SC_MyMenuItem,
             'My Menu Item');
end;

procedure TForm1.WMSysCommand(var Msg: TWMSysCommand);
begin
  if Msg.CmdType = SC_MyMenuItem then ShowMessage('Got the message')
  else inherited;
end;
```

Создание формы на основе строки

В данном совете рассказывается, как в Delphi можно создать экземпляр формы на основе строки, содержащей имя типа.

В первую очередь следует зарегистрировать данный тип в Delphi. Это делает функция RegisterClass, описанная следующим образом:

```
procedure RegisterClass(AClass: TPersistentClass);
```

AClass – класс TPersistent. Другими словами, класс, который мы регистрируем, в какой-то точке должен наследоваться от TPersistent. Поскольку все элементы управления Delphi, включая формы, отвечают этому требованию, то проблем быть не должно. Но такой способ не пройдет, если регистрируемые классы наследуются непосредственно от TObject.

После регистрации класса можно найти указатель на тип, передавая строку в FindClass. Функция возвратит ссылку на класс, которую можно использовать для создания формы. Небольшой поясняющий пример:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  b: TForm;
  f: TFormClass;
begin
  f := TFormClass(FindClass('TForm2'));
  b := f.Create(Self);
  b.Show;
end;
```

Данный код создаст тип TForm2, который мы зарегистрировали с помощью RegisterClass.

Рассмотрим демонстрационный проект. Создайте новый проект, затем добавьте четыре формы так, чтобы в общей сложности получилось 5. В реальном проекте можно заполнить их необходимыми элементами управления, для данного же примера это не важно.

В первой форме разместите поле редактирования и кнопку. Удалите все формы, кроме главной, из списка AutoCreate. Наконец, скопируйте приведенный ниже код в unit1, он позволит создавать форму по имени типа класса, введенному в поле редактирования.

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm1 = class(TForm)
```

```
    Edit1: TEdit;
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  end;

var
  Form1: TForm1;

implementation
{$R *.DFM}

uses Unit2, Unit3, Unit4, Unit5;

procedure TForm1.FormCreate(Sender: TObject);
begin
  RegisterClass(TForm2);
  RegisterClass(TForm3);
  RegisterClass(TForm4);
  RegisterClass(TForm5);
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  f: TFormClass;
begin
  f := TFormClass(FindClass(Edit1.Text));
  with f.Create(Self) do Show;
end;

end.
```

Форма OnTop

Мне необходимо поместить Delphi-форму действительно поверх других приложений, не просто поверх всех форм приложения (что просто), а постоянно, даже если я работаю, например, с Excel.

Решение

```
with MyForm do
  SetWindowPos(Handle, HWND_TOPMOST, Left, Top, Width, Height,
               SWP_NOACTIVATE or SWP_NOMOVE or SWP_NOSIZE);
```

Примечание

Теоретически, достаточно установить `FormStyle` *в* `fsStayOnTop`*. При необходимости можно дополнительно изменить* `BorderStyle` *и другие свойства. Практически же это соблюдается только для основной формы. Да и совет этот будет работать только для основной формы. См. следующий совет.*

Особенности fsStayOnTop

Почему, если присвоить свойству `FormStyle` *значение* `fsStayOnTop`*, форма так и не остается на самом верху?*

Просто добавьте `Application.RestoreTopMosts` в обработчик события формы `OnPaint`. Это ошибка реализации `fsStayOnTop`.

Примечание

И этот совет работает только для основной формы.

Обработка запроса на максимальное раскрытие окна

Как создать в приложении форму, раскрывающуюся при нажатии кнопки Открыть на весь экран *только вполовину экрана, а не на полный экран.*

Решение 1

Необходимо обработать из формы сообщение `WM_GETMINMAXINFO`. Например, добавьте следующее объявление в защищенную (`protected`) секцию формы:

```
procedure _WM_GETMINMAXINFO(var mmInfo: TWMGETMINMAXINFO);
                            message WM_GetMinMaxInfo;
```

и создайте обработчик этого сообщения следующим образом (`TForm1`, естественно, имя вашей формы):

```
procedure TForm1._WM_GETMINMAXINFO(var mmInfo: TWMGETMINMAXINFO);
begin
// устанавливаем позицию и размер формы при ее максимальном раскрытии:
  with mmInfo.MinMaxInfo^ do begin
    ptmaxposition.x := Screen.Width div 4;
    ptmaxposition.y := Screen.Height div 4;
    ptmaxsize.x := Screen.Width div 2;
    ptmaxsize.y := Screen.Height div 2;
  end;
end;
```

Решение 2

Определите в классе обработчик сообщения:

```
procedure WMGetMinMaxInfo(var Msg: TMessage); message WM_GETMINMAXINFO;
```

и его реализацию:

```
procedure TForm1.WMGetMinMaxInfo(var Msg: TMessage);
begin
  with TWMGetMinMaxInfo(Msg).MinMaxInfo^ do begin
    ptMaxTrackSize := Point(800, 600);         // макс. размеры окна
    ptMinTrackSize := Point(400, 200);         // минимальные
    ptMaxPosition := Point(50, 55);            // позиция при развертывании
```

```
    ptMaxSize := Point(750, 550);              // размер при развертывании
  end;
  Msg.Result := 1;
end;
```

Минимизирование формы при запуске

Мне необходимо при запуске приложения спрятать главную форму, но после того как я установил в главной форме свойство `WindowState` *в* `wsMinimized` *и запустил ее, форма свернулась на рабочем столе Windows 95 вместо положенной панели задач. Кто-нибудь знает, как решить эту проблему?*

Вариант обхода ошибки:

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm1 = class(TForm)
    procedure DoRestore(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  end;

var
  Form1: TForm1;

implementation
{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
// Устанавливаем временный обработчик события восстановления формы приложения
  Application.OnRestore := DoRestore;
  Application.Minimize;
end;

procedure TForm1.DoRestore(Sender: TObject);
begin
  Application.ShowMainForm := True;
// Восстанавливаем приложение
  Perform(wm_SysCommand, sc_Restore, 0);
// Гарантируем правильную перерисовку всех компонентов
  Show;
// Убираем временный обработчик события, чтобы не вызывался в будущем
  Application.OnRestore := nil;
end;
```

```
initialization
// Здесь прячем минимизированную главную форму
  Application.ShowMainForm := False;

end.
```

Чтение флажка Run Minimized

Как заставить Delphi-приложение стартовать в свернутом состоянии, если оно запускается в нормальном режиме, хотя флаг RunMinimized *установлен?*

Объект Delphi `Application` создает скрытое окно приложения, и это оно, а не главная форма, отдает команду на показ приложения в свернутом состоянии. Для того чтобы это обойти, создайте примерно такой обработчик события главной формы `OnCreate`:

```
procedure TForm1.FormCreate(Sender: TObject);
{$IFDEF WIN32}                    { Delphi 2-X.0 (32-бит) }
var
  MyInfo: TStartUpInfo;
{$ENDIF}
begin
{$IFDEF WIN32}                    { Delphi 2-X.0 (32-бит) }
  GetStartUpInfo(MyInfo);
  ShowWindow(Handle, MyInfo.wShowWindow);
{$ENDIF}
{$IFDEF WINDOWS}                  { Delphi 1.0 (16-бит) }
  ShowWindow(Handle, cmdShow);
{$ENDIF}
end;
```

Другими словами, для 16-бит достаточно флага `cmdShow` в `ShowWindow`. Для 32-бит необходимо получить информацию о способе запуска вызовом процедуры `GetStartUpInfo`, которая заполняет запись `TStartUpInfo`, и затем передать `TStartUpInfo.wShowWindow` в `ShowWindow`.

Предотвращение закрытия формы

Следующий код убирает команду закрыть из системного меню и одновременно делает серой кнопку Закрыть в заголовке формы:

```
procedure TForm1.FormCreate(Sender: TObject);
var
  hMenuHandle: HMENU;
  ...
begin
  hMenuHandle := GetSystemMenu(Handle, False);
  if (hMenuHandle <> 0) then DeleteMenu(hMenuHandle, SC_CLOSE, MF_BYCOMMAND);
end;
```

[Nikolaev Igor]

Предотвращение изменения размеров формы

Решение

```
procedure TForm1.CreateParams(var Params: TCreateParams);
begin
  inherited CreateParams(Params);
  Params.Style := Params.Style xor WS_SIZEBOX xor WS_MAXIMIZEBOX;
end;
```

[Николай]

Примечание

Не забывайте добавлять директиву `override` *после описания метода* `CreateParams` *в описании класса* `TForm1`.

Масштабирование окна

Описание функции `PixelsPerInch` в документации не верно, поскольку она имеет отношение к размеру (в пикселах/дюймах) системных шрифтов, а не формы.

Например, Delphi-свойство `PixelsPerInch` по умолчанию обычно содержит значение 96 для всех создаваемых форм, независимо от экранного разрешения системы.

Тем не менее, при изменении разрешения экрана, если установить опцию Large Fonts (большие шрифты) вместо Small Fonts (маленькие шрифты), свойство `PixelPerInch` по умолчанию для любых создаваемых форм будет равно 120 и снова не будет зависеть от разрешения системы.

Причина этого в следующем: Delphi необходимо знать оба значения `PixelsPerInch` для формы в момент ее разработки и во время работы приложения. С помощью этой информации Delphi может правильно масштабировать форму и изменять размер шрифта, с тем чтобы текст помещался на форму.

Приведем пример. Пусть создана форма при установленной опции Small Fonts. По умолчанию `PixelsPerInch` равно 96.

Пользователь пытается запустить приложение в системе, где установлены большие шрифты (выбрана опция Large Fonts). Из-за этого весь текст формы будет непропорционально большим относительно остальной части формы. Свойство `PixelsPerInch` для данного пользователя должно быть равно 120.

Если свойство формы `Scaled` установлено в `False`, Delphi не предпримет никаких мер для компенсации изменения разрешения экрана. Ваш текст при этом, вероятно, будет более крупным, чем во время разработки формы, и не поместится целиком в отведенное для него место.

Если же свойство формы `Scaled` установлено в `True`, Delphi автоматически масштабирует форму с коэффициентом 120/96. При данном способе форма

«вырастает» в соответствии с увеличенным размером шрифтов. После чего выглядит точно так, как вы ее разработали.

Примечание

Масштабирование происходит только в случае, если пользовательское свойство `PixelsPerInch` *отличается от его значения во время разработки. Насколько известно, это может случиться при изменении размера шрифтов во время смены разрешения экрана.*

Лучшее решение состоит в масштабировании формы вручную, при помощи `ScaleBy(...)`.

Другим решением могло бы быть изменение свойства формы `PixelsPerInch` во время выполнения программы, когда форма только «готовится создаться». Эта хитрость заставила бы Delphi думать, что форма была разработана с использованием другого значения `PixelsPerInch`, и что поэтому ее необходимо масштабировать. Например, изменение свойства формы `PixelsPerInch` с 96 на 80 во время ее создания заставить Delphi масштабировать ее с коэффициентом 96/80.

Текущая позиция окна

Текущую позицию можно получить, вызвав функцию Windows API `GetCurrentPositionEx`:

```
procedure TForm1.FormClick(Sender: TObject);
var
  P: TPoint;
begin
  GetCurrentPositionEx(Canvas.Handle, @P);
  Label1.Caption := IntToStr(P.x) + '/' + IntToStr(P.y);
end;
```

[Тугаев Олег]

Сохранение размеров, позиции и состояния окна

Вы, наверное, замечали, что профессионально написанные программы «запоминают» состояние и позицию окон на момент их закрытия? А большинство RAD-приложений это игнорируют? Эту ошибку можно исправить, взяв на вооружение приведенный ниже модуль, позволяющий сохранять позицию, размер и состояние окна.

Поместите `WINRSTOR` в список используемых модулей главной или любой другой формы, состояние, размер и позицию которой хотите сохранить. (Для того чтобы сэкономить время и для восстановления дочерних форм использовать `WinSaveChildren` и `WinRestoreChildren` из главной формы, надо объявить этот модуль только в главной форме).

В MainForm.Create инициализируйте глобальный объект WinRestorer следующим образом (он должен предварительно быть объявлен, но еще не инициализирован):

```
GlobalWinRestorer := TWinRestorer.Create(Application, True, WHATSAVE_ALL);
```

или так:

```
GlobalWinRestorer := TWinRestorer.Create(Application, True,
                                         [location, size, state]);
```

Затем в MainForm.Destroy необходимо разрушить глобальный объект WinRestorer следующим образом:

```
GlobalWinRestorer.Free;
```

Сохрантъ статус формы можно в обработчике события QueryClose или в специально созданной кнопке или пункте меню. Этот пункт обычно создают в меню Файл под именем "&Сохранение рабочей области" и обрабатывают следующим образом:

```
GlobalWinRestorer.SaveChildren(Self, [default]);
```

При закрытии основной формы необходимо сделать следующее:

```
GlobalWinRestorer.SaveWin(Self, WHATSAVE_ALL);
```

Восстановить состояние дочерних форм можно следующим образом:

```
GlobalWinRestorer.RestoreWin(Self, [default]);
```

Но лучше переместить данный код в обработчик события Show главной формы:

```
GlobalWinRestorer.RestoreWin(Self, [default]);
GlobalWinRestorer.RestoreChildren(Self, [default]);
```

Если свойство TForm.Position установлено в poScreenCenter или что-то подобное, данный модуль не поможет. poDesigned работает, как положено. Можно добавить обработку исключения, если вы пытаетесь установить верхнюю или левую позицию при значении формы poScreenCenter, но при этом надо соблюдать осторожность при использовании WinRestoreChildren.

```
unit WinRstor;
interface

uses SysUtils, Forms;

type
{ Восстановитель окон классовых объектов и связанных типов }
  EWinRestorer = class(Exception);
  TWhatSave = (default, size, location, state);
  STWhatSave = set of TWhatSave;
  TWinRestorer = class(TObject)
```

```
  protected
    mIniFile: string;
    mIniSect: string[80];
    mIsInitialized: boolean;
    mDefaultWhat: STWhatSave;
  public
    constructor Create(TheApp: TApplication; LocalDir: boolean;
                       DefaultWhatSave: STWhatSave);
    procedure SaveWin(TheForm: TForm; What: STWhatSave);
    procedure SaveChildren(TheMDIForm: TForm; What: STWhatSave);
    procedure RestoreWin( TheForm: TForm; What: STWhatSave);
    procedure RestoreChildren(TheMDIForm: TForm; What: STWhatSave);
    property IniFileName: string read mIniFile;
  end;
const
  WHATSAVE_ALL = [size, location, state];

var
  GlobalWinRestorer: TWinRestorer;

implementation
uses IniFiles;

constructor TWinRestorer.Create;
var
  fname, path: string[100];
begin
  inherited create;
{ Получаем имя ini-файла }
  if default in DefaultWhatSave then
    raise EWinRestorer.Create('Попытка инициализации параметров с позицией окна'
                          + 'по умолчанию с установленным элементом [default]. '
                          + 'Параметры по умолчанию могут содержать только'
                          + 'установленные элементы - [size, location, state].')
  else mDefaultWhat := DefaultWhatSave;
  fname := ChangeFileExt(ExtractFileName( TheApp.exeName), '.INI');
  if LocalDir then begin           { вычисляем путь и добавляем к нему имя файла }
    path := ExtractFilePath(TheApp.exeName);
    if path[length(path)] <> '\' then path := path + '\';
    fname := path + fname;
  end;
{ заполняем поля объекта }
  mIniFile := fname;
  mIniSect := 'WindowsRestorer';
{ Для порядка напишем некоторое примечание в секцию с именем [WinRestorer Notes] }
end;

procedure TWinRestorer.RestoreWin;
var
  FormNm, SectionNm: string[80];
  ini: TIniFile;
  n, l, t, w, h: integer;           { Left, Top, Width, Height }
```

```
begin
  ini := TIniFile.create(mIniFile);
  try
    SectionNm := mIniSect;
    FormNm := TheForm.classname;
    if default in What then What := mDefaultWhat;
{ При необходимости обновляем состояние окна }
    if state in What then
      n := ini.ReadInteger( SectionNm, FormNm + '_WindowState', 0);
    case n of
        1: TheForm.WindowState := wsMinimized;
        2: TheForm.WindowState := wsNormal;
        3: TheForm.WindowState := wsMaximized;
    end;
{ При необходимости обновляем размеры и позицию. }
    with TheForm do begin
      l := left;
      t := top;
      h := height;
      w := width;
    end;                { Сохраняем текущие значения. }
    if size in What then begin
      w := ini.ReadInteger(SectionNm, FormNm + '_Width', w);
      h := ini.ReadInteger(SectionNm, FormNm + '_Height', h);
    end;
    if location in What then begin
      t := ini.ReadInteger( SectionNm, FormNm + '_Top', t);
      l := ini.ReadInteger( SectionNm, FormNm + '_Left', l);
    end;
    TheForm.SetBounds(l,t,w,h);
  finally
    ini.free;
  end;
end;

procedure TWinRestorer.RestoreChildren;
var
  i: integer;
begin
  if TheMDIForm.formstyle <> fsMDIForm then
    raise EWinRestorer.create('Попытка сохранения размеров дочернего окна для'
                              + ' не MDI окна родителя.')
  else
    for i := 0 to TheMDIForm.MDIChildCount - 1 do
      RestoreWin( TheMDIForm.MDIChildren[i], what);
end;

procedure TWinRestorer.SaveWin;
var
  FormNm, SectionNm: string[80];
  w: STWhatsave;
  ini: TIniFile;
```

```
begin
  ini := TIniFile.create(mIniFile);
  try
    SectionNm := mIniSect;
    FormNm := TheForm.ClassName;
    if default in What then w := mDefaultWhat else w := mDefaultWhat;
    if size in w then begin
      ini.WriteInteger(SectionNm, FormNm + '_Width', TheForm.Width);
      ini.WriteInteger(SectionNm, FormNm + '_Height', TheForm.Height);
    end;
    if location in w then begin
      ini.WriteInteger( SectionNm, FormNm + '_Top', TheForm.Top);
      ini.WriteInteger( SectionNm, FormNm + '_Left', TheForm.Left);
    end;
    if state in w then
      case TheForm.WindowState of
        wsMinimized: ini.WriteInteger(SectionNm, FormNm + '_WindowState', 1);
            wsNormal: ini.WriteInteger(SectionNm, FormNm + '_WindowState', 2);
        wsMaximized: ini.WriteInteger(SectionNm, FormNm + '_WindowState', 3);
      end;
  finally
    ini.free;
  end;
end;

procedure TWinRestorer.SaveChildren;
var
  i: integer;
begin
  if TheMDIForm.Formstyle <> fsMDIForm then
    raise EWinRestorer.Create('Попытка восстановления размеров дочернего окна'
                              + ' для не MDI окна родителя.')
  else
    for i := 0 to TheMDIForm.MDIChildCount - 1 do
      SaveWin(TheMDIForm.MDIChildren[i], what);
end;

end.
```

[News Group]

Определение перемещения формы

Как определить перемещение пользователем главной формы приложения (не изменение ее размеров), кроме как при помощи таймера и проверки значений свойств `Form.Top` *и* `Form.Left`*?*

Решение 1

Объявите процедуру в секции `protected` класса формы:

```
procedure WMMove(var Message: TWMMove); message WM_Move;
```

И ее реализацию:

```
procedure TForm1.WMMove(var Message: TWMMove);
begin
  Label1.Caption := 'X = ' + IntToStr(Message.XPos) + ', Y = '
                    + IntToStr(Message.YPos);
end;
```

Событие генерируется в течение всего времени перемещения. Поэтому необходимо проверять, удерживается ли левая кнопка мыши.

Решение 2

Можно воспользоваться обработчиками следующих системных сообщений:

- WM_WINDOWPOSCHANGING (посылается перед перемещением)
- WM_WINDOWPOSCHANGED (посылается после перемещения)
- WM_MOVE (посылается после перемещения)

[News Group]

Восстановление размера окна

Существует ли какой-либо способ получения координат формы, которые она должна иметь при восстановлении из развернутого состояния?

Используйте API-функцию GetPlacement.

```
procedure TForm1.SetFormPlace(AName: string; AForm: TForm);
var
  s: string;
  Place: TWindowPlacement;
begin
  Place.length := SizeOf(TWindowPlacement);
  if not GetWindowPlacement(AForm.Handle, @Place) then exit;
  with Place do begin
    s := IntToStr(Flags);
    s := AppendS(s, ShowCmd);
    s := AppendS(s, ptMinPosition.X);
    s := AppendS(s, ptMinPosition.Y);
    s := AppendS(s, ptMaxPosition.X);
    s := AppendS(s, ptMaxPosition.Y);
    s := AppendS(s, rcNormalPosition.Left);
    s := AppendS(s, rcNormalPosition.Top);
    s := AppendS(s, rcNormalPosition.Right);
    s := AppendS(s, rcNormalPosition.Bottom);
  end;
  SetString(AName, s);
end;
```

[News Group]

Помещение компонентов VCL в область заголовка

Нужно разместить все необходимые элементы управления в отдельной форме, которая должна отслеживать перемещение и изменение размеров основной формы. Данная форма всегда будет находиться над областью заголовка основной формы.

Нижеприведенный проект включает в себя две формы и выпадающий список (ComboBox). После запуска программы список появляется в области заголовка главной формы.

Два ключевых вопроса:

- организация перехвата сообщения `WM_MOVE` главной формы;
- возвращение фокуса в главную форму после того как пользователь выберет какой-либо элемент управления, способный иметь фокус (например, `TComboBox`, `TButton` и др.).

```
unit Unit1;

interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;
type
  TForm1 = class(TForm)
    procedure FormResize(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormHide(Sender: TObject);
  public
    procedure WMMove(var Msg: TWMMove); message WM_MOVE;
  end;
var
  Form1: TForm1;
implementation
uses Unit2;
{$R *.DFM}

procedure TForm1.FormResize(Sender: TObject);
begin
  with Form2 do begin
{ Заменим магические числа реальной информацией SystemMetrics }
    Width := Form1.Width - 120;
    Top := Form1.Top + GetSystemMetrics(SM_CYFRAME);
    Left := ((Form1.Left + Form1.Width) - Width) - 60;
  end;
end;

procedure TForm1.FormShow(Sender: TObject);
begin
  Form2.Show;
end;
```

```
procedure TForm1.FormHide(Sender: TObject);
begin
  Form2.Hide;
end;

procedure TForm1.WMMove(var Msg: TWMMove);
begin
  inherited;
  if (Visible) then FormResize(Self);
end;

end.
```

Исходный код для псевдозаголовка. Данная форма может содержать любые элементы управления VCL, которые предполагается установить в области заголовка главной формы. По существу, это независимый диалог со следующими свойствами:

```
Caption = ''           { NULL строка }
Height =               { высота области заголовка }
Width =                { высота всех компонентов на форме }
BorderIcons = []       { пусто }
BorderStyle = bsNone
FormStyle = fsStayOnTop
```

Исходный код для Form2:

```
unit Unit2;

interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;
type
  TForm2 = class(TForm)
    ComboBox1: TComboBox;
    procedure FormCreate(Sender: TObject);
    procedure ComboBox1Change(Sender: TObject);
    procedure FormResize(Sender: TObject);
  end;
var
  Form2: TForm2;

implementation
uses Unit1;
{$R *.DFM}

procedure TForm2.FormCreate(Sender: TObject);
begin
  Height := ComboBox1.Height - 1;
  Width := ComboBox1.Width - 1;
end;
```

```
procedure TForm2.ComboBox1Change(Sender: TObject);
begin
  Form1.SetFocus;
end;

procedure TForm2.FormResize(Sender: TObject);
begin
  ComboBox1.Width := Width;
end;

end.
```

Файл проекта (.DPR) довольно простой:

```
program Project1;

uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1},
  Unit2 in 'Unit2.pas' {Form2};
{$R *.RES}
begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm2, Form2);
  Application.Run;
end.
```

Хотя авторы некоторых книг утверждают: «Вы не можете установить компоненты Delphi в заголовок окна, точнее, не существует никакого способа установить их там».

Перемещение формы

Как перетащить форму, удерживая мышью ее клиентскую часть?

Решение 1

Простой компонент, работающий по этому принципу:

```
unit WinDrag;

interface
uses
   Windows, SysUtils, Classes;

type
  TWinDrag = class(TComponent)
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override; function GetDragFlag: boolean;
    procedure SetDragFlag(Status: Boolean);
```

```
  published
    property DragFlag: Boolean read GetDragFlag write SetDragFlag;
end;

procedure Register;
implementation

constructor TWinDrag.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  DragFlag := GetDragFlag;
end;

destructor TWinDrag.Destroy;
begin
  inherited Destroy;
end;

function TWinDrag.GetDragFlag: boolean;
var
  Value: Boolean;
begin
  SystemParametersInfo(SPI_GETDRAGFULLWINDOWS, 0, @Value, 0);
  Result := Value;
end;

procedure TWinDrag.SetDragFlag(Status: Boolean);
begin
  SystemParametersInfo(SPI_SETDRAGFULLWINDOWS, Integer(Status), POINTER(0), 0);
end;
procedure Register;

begin
  RegisterComponents('Samples', [TWinDrag]);
end;
end.
```

Решение 2

Перетаскивать форму можно за любой элемент, который находится на ней. Для этого создайте обработчик(и) OnMouseDown для элементов, за которые хотите перетаскивать форму (и/или для самой формы). Пример для Panel:

```
procedure TForm1.Panel1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
const
  SC_DRAGMOVE : Longint = $F012;
begin
  ReleaseCapture;
  SendMessage(Form1.Handle, WM_SYSCOMMAND, SC_DRAGMOVE, 0);
end;
```

[Matveychuk Sergey]

Решение 3

А так можно передвигать форму, если пользователь «захватил» мышью клиентское пространство:

```
unit Unit1;

interface
uses
  Windows, SysUtils, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
    procedure FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
  end;
var
  Form1: TForm1;
  MX: integer;
  MY: integer;
implementation
{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
begin
  Close;
end;

procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
  if Shift <> [ssLeft] then begin
    MX := X;
    MY := Y;
  end else begin
   Left := Left + X - MX;
   Top := Top + Y - MY;
  end;
end;

end.
```

[Ситников Митрий]

Решение 4

Выберите элемент управления или саму форму и поместите следующий текст в его обработчик события `OnMouseDown` (данный пример дан только для формы):

```
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
                               Shift: TShiftState; X, Y: Integer);
```

```
begin
  ReleaseCapture;
  Perform(WM_SYSCOMMAND, SC_MOVE + 2, 0);
end;
```

Решение 5

Простейший путь. Пусть окно «думает», что его перемещают за заголовок. Сделайте это с помощью системного сообщения WM_NCHitTest.

```
type
  TForm1 = class(TForm)
  public
    procedure WMNCHitTest(var M: TWMNCHitTest); message WM_NCHitTest;
  end;

var
  Form1: TForm1;

implementation
{$R *.DFM}

procedure TForm1.WMNCHitTest(var M: TWMNCHitTest);
begin
  inherited;                    // вызвали наследованный дескриптор сообщения,
  if M.Result = htClient then   // щелкнув в области окна?
    M.Result := htCaption;      // если так, то мы заставили Windows «думать»,
                                // что щелчок был сделан на заголовке окна.
end;
```

Помещение формы в поток

Delphi имеет в своем распоряжении функцию, позволяющую сделать это:

```
procedure WriteComponentResFile(const FileName: string; Instance: TComponent);
```

Заполните имя файла, в котором будете сохранять компонент, и читайте его затем следующей функцией:

```
function ReadComponentResFile(const FileName: string;
                              Instance: TComponent): TComponent;
```

Рисуем на рамке окна

Можно ли рисовать на рамке окна?

Создавайте обработчик для WM_NCPAINT. Приведенный ниже пример рисует вокруг формы красную рамку шириной в один пиксел.

```
type
  TForm1 = class(TForm)
    private
```

```
      procedure WMNCPaint(var Msg: TWMNCPaint); message WM_NCPAINT;
end;

var
  Form1: TForm1;

implementation
{$R *.DFM}

procedure TForm1.WMNCPaint(var Msg: TWMNCPaint);
var
  dc: hDc;
  Pen: hPen;
  OldPen: hPen;
  OldBrush: hBrush;
begin
  inherited;
  dc := GetWindowDC(Handle);
  msg.Result := 1;
  Pen := CreatePen(PS_SOLID, 1, RGB(255, 0, 0));
  OldPen := SelectObject(dc, Pen);
  OldBrush := SelectObject(dc, GetStockObject(NULL_BRUSH));
  Rectangle(dc, 0, 0, Form1.Width, Form1.Height);
  SelectObject(dc, OldBrush);
  SelectObject(dc, OldPen);
  DeleteObject(Pen);
  ReleaseDC(Handle, Canvas.Handle);
end;
```

Вызов функций из различных дочерних MDI-окон

Решение

```
TChild(ActiveMDIChild).SomeMethod;
```

Динамическое создание формы

В обычной ситуации динамические экземпляры форм (или дочерних форм в нашем случае) создаются примерно таким образом:

```
frmNewForm := TNewForm.Create(Owner);
```

Объявляя `frmNewForm` как глобальную переменную, можно использовать следующий код с привлечением глобальных переменных:

```
if frmNewForm = NIL then
  frmNewForm := TNewForm.Create(Owner);
frmNewForm.Show;
```

Найдите ссылку на frmNewForm, которая сохраняется в списке компонентов родителя при вызове TForm.Create(Owner). В большинстве случаев ссылка, а точнее указатель будет собственностью массива компонентов главной формы.

Если нет главной формы, которую можно было бы использовать в качестве родителя подформы, то воспользуйтесь методом TAppllication CreateForm(TForm, reference). Позже, когда понадобится получить ссылку на подформу, нужно будет пройти в цикле по свойству-массиву Components вашего объекта Application. Найдите форму по ее имени и получите ссылку.

Одно важное замечание: при разрушении объекта необходимо устанавливать указатель на объект в Nil. В противном случае, в следующий раз можете получить GPF.

```
frmNewForm.Release;
frmNewForm := nil;
```

Примечание

Если при создании формы вместо 'Application' *в методе* Create *указать* Self*, форма не будет являться компонентом* Application*, и в дальнейшем ее нельзя будет найти, перебирая в цикле компоненты объекта* Application.

Вместо создания цикла, перебирающего все компоненты Application.Components*, мы могли бы воспользоваться* Application.FindComponent*, но только в том случае, если после создания формы ей было назначено имя (например,* SubForm.Name := 'SubForm';*).* FindComponent *ищет компоненты в массиве* Components *по их имени. Но найти форму таким образом можно только после того, как ей назначено имя.*

Используя для проверки существования формы что-то типа 'if SubForm = nil'*, учтите, что это может не работать, если при освобождении формы ссылка на нее не установлена в nil. Delphi не делает это автоматически.*

Если в качестве родителя формы определен Application, то для ее поиска (чтобы убедиться, что она еще существует) необходимо перебрать все компоненты объекта Application.

Чтобы сделать это, можно выбрать один из двух способов:

```
Application.CreateForm(TSubForm, SubForm);
```

или

```
SubForm := TSubForm.Create(Application);
```

А затем проверить существование формы, перебирая в цикле компоненты:

```
for i := 0 to Application.ComponentCount - 1 do begin
  if Application.Components[i] = SubForm then ... { форма существует };
```

[News Group]

Создание формы небольшой ширины

Добавьте следующее определение в секцию intarface создаваемой формы:

```
procedure WMGetMinMaxInfo(var Message: TWMGetMinMaxInfo); message WM_GETMINMAXINFO;
```

и следующий код в секцию implementaion:

```
procedure TForm1.WMGetMinMaxInfo(var Message: TWMGetMinMaxInfo);
begin
  with Message do begin
    MinMaxInfo^.ptMinTrackSize.x := 20;
  end;
end;
```

Пример использования из приложения:

```
procedure TForm1.Form1OnCreate(Sender: TObject);
begin
  Width := 20;
end;
```

[News Group]

Управление разворачиванием формы

Небольшая программа, позволяющая управлять размером формы при ее разворачивании. Она сделана таким образом, что устраняет мигание, наблюдаемое при изменении ее размера в обработчике события OnResize. В основном этот код работает с сообщением WM_GetMinMaxInfo. Чтобы использовать данный стиль формы вместо стандартного стиля, принятого в Delphi, скомпилируйте следующий файл с паскалевским кодом в DCU-файл. Затем замените стандартную TForm новой TMaxForm и добавьте maxform в список используемых модулей. А теперь сам код описываемого модуля:

```
unit MaxformEx;

interface

uses
  WinTypes, WinProcs, SysUtils, Messages, Classes, Graphics, Controls, Forms,
  Dialogs;

type
  TMaxForm = class(TForm)
  private
    fmh, fmw, fml, fmt: word;
    procedure mymax(var m: TWMGETMINMAXINFO); message wm_getminmaxinfo;
  public
    constructor create(AOwner: TComponent); override;
  published
```

```
    property maxheight: word read mh write mh;
    property maxwidth: word read mw write mw;
    property maxleft: word read ml write ml;
    property maxtop: word read mt write mt;
end;

implementation

procedure TMaxForm.mymax(var m: TWMGETMINMAXINFO);
begin
  m.minmaxinfo^.ptmaxsize.x := fmw;
  m.minmaxinfo^.ptmaxsize.y := fmh;
  m.minmaxinfo^.ptmaxposition.x := fml;
  m.minmaxinfo^.ptmaxposition.y := fmt;
end;

constructor TMaxForm.create(Aowner: TComponent);
begin
  fmw := screen.width;
  fmh := screen.height;
  fmt := 0;
  fml := 0;
  inherited create(Aowner);
end;

end.
```

[News Group]

Примечание

Пример приведен для Delphi 1.

Закрытие модальной формы

Решение 1

```
procedure TForm1.AppDeactivate(Sender: TObject);
var
  hw: HWnd;
  CurTask: THandle;
  WndStyle: Longint;
begin
  CurTask := GetWindowTask(handle);
  hw := GetWindow(GetDesktopWindow, GW_CHILD);
  while GetWindowTask(hw) <> CurTask do
    hw := GetWindow(hw, GW_HWNDNEXT);
  while (hw <> handle) and (GetWindowTask(hw) = CurTask) do begin
    PostMessage(hw, WM_Close, 0, 0);
    hw := GetWindow(hw, GW_HWNDNEXT);
  end;
end;
```

Решение 2

Обработчик FormActivate вызывается как часть кода, выполняющегося при перемещении фокуса на новую форму. Поэтому Delphi подавляет изменения фокуса во время работы данного обработчика. Если вы когда-нибудь пробовали трассировать изменения фокуса с помощью Windows API, то знаете, что изменение фокуса во время его смены приводит к его бесконечному зацикливанию в Windows.

Решение должно быть простым. Пошлите форме, которую хотите закрыть, в конце обработчика OnActivate сообщение WM_CLOSE. Это работает, поскольку Windows поместит сообщение в очередь лишь после того, как будет выполнен обработчик OnActivate и, следовательно, завершится процедура изменения фокуса.

Пример программы. Одна форма с кнопкой, которая активизирует диалог About, после чего он немедленно закрывается. Пример не очень полезен, но цель его – показать принцип.

```
...
implementation
uses About;
procedure TForm1.Button1Click(Sender: TObject);
begin
  AboutBox.ShowModal;
end;
...
unit About;
...
implementation
uses
  Messages;
procedure TAboutBox.FormActivate(Sender: TObject);
begin
  PostMessage(Handle, WM_CLOSE, 0, 0);
end;
```

Модальные формы и минимизация

Если на главной форме в событии Application.OnMinimize запоминать последнюю активную форму LastActiveForm = Screen.ActiveForm, а в событии Application.OnRestore выполнить ShowWindow(LastActiveForm. Handle, SW_NORMAL), то она и восстановится как положено.

Модальные диалоги для всей системы

Ознакомьтесь с API-функциями SetSysModalWindow и BringWindowToTop.

Если вам необходим простой диалог с небольшим количеством текста, дополнительной пиктограммой и некоторыми кнопками, используйте MessageBox из Windows с MB_SYSTEMMODAL в параметре TextType.

Обратитесь к функции Windows API SetSysModalWindow(). Код, приведенный ниже, демонстрирует технологию работы с этой функцией. В любой момент времени может быть возможен только один модальный системный диалог, дескриптор которого возвращается функцией SetSysModalWindow(). Необходимо запомнить возвращаемую функцией величину, для того чтобы завершить показ диалога таким образом. Это должно выглядеть примерно так:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  x: word;
begin
  x := SetSysModalWindow(AboutBox.Handle);
  AboutBox.ShowModal;
  SetSysModalWindow(x);
end;
```

Примечание

Этот пример написан для Win16, но функция SetSysModalWindow *поддерживается Delphi только для совместимости с Win16 и в Win32 не работает, т. к. Windows не поддерживает более модальных диалогов.*

Сворачивание окон приложения

Как свернуть все запущенные окна?

Решение

```
{$APPTYPE CONSOLE}
program Minimize;
uses
  Windows, Messages;

var
  Count: integer;

function EnumProc(WinHandle: HWnd; Param: LongInt): boolean; stdcall;
begin
  if (GetParent (WinHandle) = 0) and (not IsIconic (WinHandle))
      and (IsWindowVisible (WinHandle)) then begin
    PostMessage(WinHandle, WM_SYSCOMMAND, SC_MINIMIZE, 0);
    Inc(Count);
  end;
  EnumProc := True;
end;
begin
  Count := 0;
  EnumWindows(@EnumProc, 0);
  Writeln('Minimized ', Count, ' windows');
end.
```

[News Group]

Динамическое создание/закрытие формы

Когда форма невидима, ее освобождение приводит к освобождению ресурсов. Это нужно делать, если форма динамически создается во время выполнения приложения. Пользуйтесь методом `Release`, а не `Free`.

Попробуйте следующий код:

```
unit Unit1;

interface

uses
  Windows, SysUtils, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  public
    Form2: TForm;
end;

var
  Form1: TForm1;
implementation
{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
begin
  if Form2 <> nil then begin
    Form2.Release;
    Form2 := nil;
  end;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  if Form2 = nil then begin
    Form2 := TForm.Create(Application);
    Form2.Show;
  end;
end;

end.
```

В данной ситуации следует отказаться от:

```
begin
  Application.Create(myForm);
```

```
  Application.Run;
end.
```

и воспользоваться следующим кодом:

```
begin
  myForm := TmyForm.Create(Application);
{ вставьте сюда нужный вам код }
  myForm.Show;                    { или myForm.ShowModal }
{ вставьте сюда нужный вам код }
  myForm.Hide;                    { если вы хотите скрыть ее }
{ вставьте сюда нужный вам код }
  myForm.Free;
end.
```

Можно применять методы формы Create и Free когда угодно и сколько угодно раз.

Не забывайте освобождать ресурсы всех форм перед завершением приложения!

[News Group]

Заполнение изображением MDI-формы

Кто-нибудь знает, как поместить в MDI-форму изображение и заполнить им всю форму (tile)?

Решение 1

```
procedure TMDIChild.OnPaint(Sender: TObject);

  procedure Tile(c: TCanvas; b: TBitMap);
  var
    x, y, h, w, i, j: integer;
  begin
    h := b.Height;
    w := b.Width;
    y := 0;
    with c.ClipRect do begin
      i := bottom - top - 1;        // высота
      j := right - left - 1;        // ширина
    end;
    while y < i do begin
      x := 0;
      while x < j do begin
        c.draw(x, y, b);
        inc(x, w);
      end;
      inc(y, h);
    end;
  end;
```

```
begin
  if Sender is TForm then Tile(TForm(Sender).Canvas, bmap);
end;
```

Примечание

Необходимо позаботиться, чтобы на MDI-форме не было компонентов с `Align = alClient`*, иначе не удастся увидеть рисунок.* `bmap` *– глобальная переменная (*`TBitMap`*), содержащая рисунок. Рисунок можно загрузить в обработчике* `OnCreate` *формы, а освободить ресурсы в* `OnDestroy`*. Для правильной отрисовки необходимо добавить вызов* `OnPaint(Sender);` *в обработчики* `OnFormPaint` *и* `OnFormResize`*.*

Решение 2

```
  ...
  private

    OutCanvas: TCanvas;
    OldWinProc, NewWinProc: Pointer;

    procedure NewWinProcedure(var Msg: TMessage);
  ...

procedure TMainForm.FormCreate(Sender: TObject);
begin
  NewWinProc := MakeObjectInstance(NewWinProcedure);
  OldWinProc := Pointer(SetWindowLong(ClientHandle, gwl_WndProc,
                        Cardinal(NewWinProc)));
  OutCanvas := TCanvas.Create;
end;

procedure TMainForm.NewWinProcedure (var Msg: TMessage);
var
  BmpWidth, BmpHeight: Integer;
  I, J: Integer;
begin
  Msg.Result := CallWindowProc(OldWinProc, ClientHandle,
                               Msg.Msg, Msg.wParam, Msg.lParam);
  if Msg.Msg = wm_EraseBkgnd then begin
    BmpWidth := MainForm.Image1.Width;
    BmpHeight := MainForm.Image1.Height;
    if (BmpWidth <> 0) and (BmpHeight <> 0) then begin
      OutCanvas.Handle := Msg.wParam;
      for I := 0 to MainForm.ClientWidth div BmpWidth do
        for J := 0 to MainForm.ClientHeight div BmpHeight do
          OutCanvas.Draw(I * BmpWidth, J * BmpHeight,
                         MainForm.Image1.Picture.Graphic);
     end;
  end;

end;
```

```
procedure TMainForm.FormDestroy(Sender: TObject);
begin
  OutCanvas.Free;
end;
```

[News Group]

Удаление заголовка дочерней MDI-формы

Решение 1

Мы смогли удалить область заголовка дочерней MDI-формы, сделав следующее:

```
type
  TMDIChild = class(TForm)
  ...

    procedure CreateParams(var Params: TCreateParams); override;
  ...
  end;

procedure TMDIChild.CreateParams(var Params: TCreateParams);
begin
  inherited CreateParams(Params);
  Params.Style := Params.Style and not WS_OVERLAPPEDWINDOW or WS_BORDER;
end;
```

Решение 2

В дочерних MDI-формах выставление в свойстве `BorderStyle` флага `bsNone` не убирает заголовок (об этом упоминается в файлах справки). Попробуйте сделать так:

```
procedure TMDIChildForm.CreateParams(var Params: TCreateParams);
begin
  Inherited CreateParams(Params);
  Params.Style := Params.Style and (not WS_CAPTION);
end;
```

Проблема закрытия дочернего MDI-окна

Не пытайтесь разрушить форму из нее самой. Присвоение параметру `Action` значения `caFree` в обработчике события формы `OnClose` заставит родительское окно само уничтожить дочернюю форму.

Для предотвращения закрытия формы необходимо обрабатывать событие `OnCloseQuery` (например, в момент редактирования таблицы или для корректного сохранения вновь введенных значений на дочерней MDI-форме).

Родительское MDI-окно должно иметь пункт меню для возможности закрытия активного в текущий момент дочернего окна. Примерный код, обрабатывающий выбор данного пункта меню:

```
ActiveMDIChild.Close;
```

Попробуйте следующее:

```
procedure TFrmServers.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action := caFree;
end;

procedure TFrmServers.FormDestroy(Sender: TObject);
begin
  Table1.Close;
end;

procedure TFrmServers.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
  if Table1.State in [dsEdit, dsInsert] then begin

// Предупреждаем пользователя о возможной потере редактируемых данных.
  if not UserSaysOk then CanClose := False; // При нажатии кнопки OK закрываем окно.
  end;
end;
```

Скрытие дочерних MDI-форм

Помогите разобраться с проблемой скрытия дочерних MDI-форм в приложении MDI. В идеале я не хотел бы, чтобы по умолчанию форма была минимизирована в родительской. Хотелось бы отображать в последней только пиктограмму.

Попытайтесь переписать `WM_PAINT` (в вашем случае может понадобиться `WM_NCPAINT`), проверьте `IsIconized`, и создайте для скрытой формы красивую пиктограмму.

Создание главной формы по условию

Существует ли в Delphi возможность создавать главную форму по условию? Я хочу использовать условие `IF` *(в зависимости от передаваемого параметра) для того, чтобы определить, какая форма будет главной при старте приложения.*

Хитрость здесь заключается в том, что мы предоставляем компилятору весь необходимый для создания форм код, но не допускаем его выполнения (`IF FALSE THEN`), при этом компилятор не ругается, а мы тем временем (во время выполнения приложения) выбираем и создаем главную форму. Ниже в качестве примера приведен измененный .DPR-файл, который при старте случайным образом выбирает из двух форм главную:

```
  ...
begin
  Application.Initialize;
  if false then begin
    Application.CreateForm(TForm1, Form1);
    Application.CreateForm(TForm2, Form2);
  end;
  Randomize;
  if Random < 0.5 then Application.CreateForm(TForm1, Form1)
  else Application.CreateForm(TForm2, Form2);
  Application.Run;
end.
```

[News Group]

Мерцание формы

Как осуществить рисование в окне без мерцания и без создания виртуального изображения в памяти? WM_SETREDRAW *здесь поможет?*

Попробуйте этот код. Даже если некоторые компоненты имеют пару BeginUpdate/EndUpdate, то для таких компонентов, как TTreeView, интенсивное рисование может послужить причиной перемещения полосы прокрутки и появления других эффектов. В таких ситуациях вместо дескриптора элемента управления используйте родительский дескриптор.

```
procedure BeginScreenUpdate(hwnd: THandle);
begin
  if (hwnd = 0) then hwnd := Application.MainForm.Handle;
  SendMessage(hwnd, WM_SETREDRAW, 0, 0);
end;

procedure EndScreenUpdate(hwnd: THandle; erase: Boolean);
begin
  if (hwnd = 0) then hwnd := Application.MainForm.Handle;
  SendMessage(hwnd, WM_SETREDRAW, 1, 0);
  RedrawWindow(hwnd, nil, 0, DW_FRAME + RDW_INVALIDATE + RDW_ALLCHILDREN
               + RDW_NOINTERNALPAINT );
  if (erase) then Windows.InvalidateRect(hwnd, nil, True);
end;
```

[News Group]

Слияние меню дочернего и главного окна

Как создать MDI-приложение, в котором способны сливаться не только меню дочернего и главного окна, но и панели инструментов?

Предлагаем решение с использованием CoolBar:

```
procedure TMainForm.SetBands(AControls: array of TWinControl;
                             ABreaks: array of boolean);
```

```
var
  i: integer;
begin
  with CoolBar do begin
    for i := 0 to High(AControls) do begin
      if Bands.Count = Succ(i) then TCoolBand.Create(Bands);
      with Bands[Succ(i)] do begin
        if Assigned(Control) then Control.Hide;
        MinHeight := AControls[i].Height;
        Break := ABreaks[i];
        Control := AControls[i];
        Control.Show;
        Visible := True;
      end;
    end;
    for i := High(AControls) + 2 to Pred(Bands.Count) do Bands[i].Free
  end;
end;
```

и

```
procedure TMsgForm.FormActivate(Sender: TObject);
begin
  MainForm.SetBands([ToolBar], [False]);
end;
```

Оба массива должны быть равны по длине. `CoolBar.Bands[0]` должен существовать всегда. На нем размещаются «глобальные» кнопки. `CoolBar[1]` тоже можно создать во время разработки с `Break = False` и перетащить поближе к началу (к левому верхнему углу) формы. При `CoolBar.AutoSize = True` не исключено «мигание», так что можно добавить:

```
AutoSize := False;
try
  ...
finally
AutoSize := True;
```

Прямой вызов метода Hint

Строка с комментарием <<<< позиционирует подсказку ниже элемента управления. Это может быть изменено, если по какой-то причине потребуется другое расположение окна с подсказкой.

```
function RevealHint (Control: TControl): THintWindow;
{ Демонстрирует всплывающую подсказку для определенного элемента управления
  (Control), возвращает ссылку на hint-объект, поэтому в дальнейшем подсказка может
  быть спрятана вызовом RemoveHint }
var
  ShortHint: string;
  AShortHint: array[0..255] of Char;
  HintPos: TPoint;
```

```
  HintBox: TRect;
begin
{ Создаем окно }
  Result := THintWindow.Create(Control);

{ Получаем первую часть подсказки до '|' }
  ShortHint := GetShortHint(Control.Hint);

{ Вычисляем месторасположение и размер окна подсказки }
  HintPos := Control.ClientOrigin;
  Inc(HintPos.Y, Control.Height + 6);             <<<< Смотри примечание
  HintBox := Bounds(0, 0, Screen.Width, 0);
  DrawText(Result.Canvas.Handle, StrPCopy(AShortHint, ShortHint), -1, HintBox,
           DT_CALCRECT or DT_LEFT or DT_WORDBREAK or DT_NOPREFIX);
  OffsetRect(HintBox, HintPos.X, HintPos.Y);
  Inc(HintBox.Right, 6);
  Inc(HintBox.Bottom, 2);

{ Теперь показываем окно: }
  Result.ActivateHint(HintBox, ShortHint);
end; {RevealHint}

procedure RemoveHint(var Hint: THintWindow);
{ Освобождаем дескриптор окна всплывающей подсказки }
begin
  Hint.ReleaseHandle;
  Hint.Free;
  Hint := nil;
end;
```

«Устойчивые» всплывающие подсказки

На компоненте `TTabbedNotebook` *у меня есть множество компонентов* `TEdit`. *Я изменяю цвет компонентов* `TEdit` *на желтый и назначаю свойству* `Hint` *компонента строчку предупреждения, если поле редактирования содержит неверные данные.*

Поведение окна со всплывающей подсказкой (hintwindow) позволяет делать его видимым только тогда, когда курсор мыши находится в области элемента управления. Но мой заказчик хочет видеть подсказки все время, пока поле редактирования имеет фокус. Я не знаю, как изменить поведение всплывающей подсказки, заданное по умолчанию. Я знаю, что это возможно, но кто мне подскажет как?

Ниже приведен модуль, содержащий новый тип `TFocusHintWindow`. Когда вы «просите» `TFocusHintWindow` появиться, он появляется ниже элемента управления, имеющего фокус. Для отображения и скрытия достаточно следующих команд:

```
FocusHintWindow.Showing := True;
FocusHintWindow.Showing := False;
```

Пример применения содержится в комментариях к модулю.

```
// Вот пример использования TFocusHintWindow. Данный пример выводит
// всплывающую подсказку ниже любого TEdit, имеющего фокус. В противном случае
// выводится стандартная подсказка Windows.

unit Unit1;

interface

uses
  Windows, SysUtils, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, FHintWin;

type
  TForm1 = class(TForm)
      procedure FormCreate(Sender: TObject);
    private
      FocusHintWindow: TFocusHintWindow;
      procedure AppIdle(Sender: TObject; var Done: Boolean);
      procedure AppShowHint(var HintStr: string; var CanShow: Boolean;
                            var HintInfo: THintInfo);
end;

implementation
procedure TForm1.FormCreate(Sender: TObject);
begin
  Application.OnIdle := AppIdle;
  Application.OnShowHint := AppShowHint;
  FocusHintWindow := TFocusHintWindow.Create(Self);
end;

procedure TForm1.AppIdle(Sender: TObject; var Done: Boolean);
begin
  FocusHintWindow.Showing := Screen.ActiveControl is TEdit;
end;

procedure TForm1.AppShowHint(var HintStr: string; var CanShow: Boolean;
                             var HintInfo: THintInfo);
begin
  CanShow := not FocusHintWindow.Showing;
end;

end.

unit FHintWin;
interface

uses
  SysUtils, WinTypes, WinProcs, Classes, Controls, Forms;

type
  TFocusHintWindow = class(THintWindow)
```

```
    private
      FShowing: Boolean;
      HintControl: TControl;
    protected
      procedure SetShowing(Value: Boolean);
      function CalcHintRect(Hint: string): TRect;
      procedure Appear;
      procedure Disappear;
    public
      property Showing: Boolean read FShowing write SetShowing;
  end;

implementation

function TFocusHintWindow.CalcHintRect(Hint: string): TRect;
var
  Buffer: array[Byte] of Char;
begin
  Result := Bounds(0, 0, Screen.Width, 0);
  DrawText(Canvas.Handle, StrPCopy(Buffer, Hint), -1, Result,
           DT_CALCRECT or DT_LEFT or DT_WORDBREAK or DT_NOPREFIX);
  with HintControl, ClientOrigin do OffsetRect(Result, X, Y + Height + 6);
  Inc(Result.Right, 6);
  Inc(Result.Bottom, 2);
end;

procedure TFocusHintWindow.Appear;
var
  Hint: string;
  HintRect: TRect;
begin
  if (Screen.ActiveControl = HintControl) then Exit;
  HintControl := Screen.ActiveControl;
  Hint := GetShortHint(HintControl.Hint);
  HintRect := CalcHintRect(Hint);
  ActivateHint(HintRect, Hint);
  FShowing := True;
end;

procedure TFocusHintWindow.Disappear;
begin
  HintControl := nil;
  ShowWindow(Handle, SW_HIDE);
  FShowing := False;
end;

procedure TFocusHintWindow.SetShowing(Value: Boolean);
begin
  if Value then Appear else Disappear;
end;
end.
```

Создание Hint-окна

Как создать собственное Hint-окно?

Решение

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Timer1.Enabled := false;
  Panel1.Visible := false;
  Panel1.BevelInner := bvNone;
  Panel1.BevelOuter := bvNone;
  Panel1.BorderStyle := bsSingle;
  Panel1.Color := clWhite;
  Button1.Hint := 'Hint test';
end;

procedure TForm1.ShowAHint(x: integer; y: integer; Caption: string;
                           Duration: LongInt);
var
  dc: hdc;
  OldFont: hFont;
  pt: TSize;
  p: pChar;
begin
  if Timer1.Enabled <> false then Timer1.Enabled := false;
  Timer1.Enabled := false;
  if Panel1.Visible <> false then Panel1.Visible := false;
  if Caption = '' then exit;
  Panel1.Caption := caption;
{ Get the width of the caption string }
  GetMem(p, Length(Panel1.Caption) + 1);
  StrPCopy(p, Panel1.Caption);
  dc := GetDc(Panel1.Handle);
  OldFont := SelectObject(dc, Panel1.Font.Handle);
  GetTextExtentPoint32(dc, p, Length(Panel1.Caption), pt);
  SelectObject(dc, OldFont);
  ReleaseDc(Panel1.Handle, Dc);
  FreeMem(p, Length(Panel1.Caption) + 1);
{ Position and show the panel }
  Panel1.Left := x;
  Panel1.Top := y;
  Panel1.Width := pt.cx + 6;
  Panel1.Height := pt.cy + 2;
  Panel1.Visible := true;
{ Fire off the timer to hide the panel }
  Timer1.Interval := Duration;
  Timer1.Enabled := true;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  if Panel1.Visible <> false then
```

```
  Panel1.Visible := false;
  Timer1.Enabled := false;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
{ Let the button repaint }
  Application.ProcessMessages;
  ShowAHint(Button1.Left, Button1.Top + Button1.Height + 6, Button1.Hint, 2000);
end;
```

Канва от THandle

Мне необходимо создать метафайл Windows. Delphi непосредственно это не поддерживает, поэтому я создаю новый метафайл при помощи функций Windows API. При создании метафайла мне возвращается его `THandle`*, являющийся дескриптором контекста устройства Windows. Как мне в Delphi использовать возвращаемый* `THandle` *для получения или создания канвы (Canvas) для рисования?*

Решение

```
unit Metaform;
interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Buttons, ExtCtrls;

type
  TForm1 = class(TForm)
    Panel1: TPanel;
    BitBtn1: TBitBtn;
    Image1: TImage;
    procedure BitBtn1Click(Sender: TObject);
  end;
var
  Form1: TForm1;

implementation
{$R *.DFM}
type
  TMetafileCanvas = class(TCanvas)
  private
    FClipboardHandle: THandle;
    FMetafileHandle: HMetafile;
    FRect: TRect;
  protected
    procedure CreateHandle; override;
    function GetMetafileHandle: HMetafile;
  public
    constructor Create;
```

```
    destructor Destroy; override;
    property Rect: TRect read FRect write FRect;
    property MetafileHandle: HMetafile read GetMetafileHandle;
  end;

constructor TMetafileCanvas.Create;
begin
  inherited Create;
  FClipboardHandle := GlobalAlloc(GMEM_SHARE or GMEM_ZEROINIT,
                                  SizeOf(TMetafilePict));
end;

destructor TMetafileCanvas.Destroy;
begin
  DeleteMetafile(CloseMetafile(Handle));
  if Bool(FClipboardHandle) then GlobalFree(FClipboardHandle);
  if Bool(FMetafileHandle) then DeleteMetafile(FMetafileHandle);
  inherited Destroy;
end;
procedure TMetafileCanvas.CreateHandle;
var
  MetafileDC: HDC;
begin
{ Создаем в памяти DC метафайла }
  MetafileDC := CreateMetaFile(nil);
  if Bool(MetafileDC) then begin
{ Совмещаем верхний левый угол отображаемого прямоугольника с левым верхним углом
  контекста устройства. Создаем границу шириной 10 логических единиц вокруг
  изображения. }
    with FRect do SetWindowOrg(MetafileDC, Left - 10, Top - 10);
{ Устанавливаем размер изображения с бордюром шириной 10 логических единиц }
    with FRect do SetWindowExt(MetafileDC, Right - Left + 20, Bottom - Top + 20);
{ Задаем корректное содержание данному метафайлу. }
    if Bool(FMetafileHandle) then begin
      PlayMetafile(MetafileDC, FMetafileHandle);
    end;
  end;
  Handle := MetafileDC;
end;

function TMetafileCanvas.GetMetafileHandle: HMetafile;
var
  MetafilePict: PMetafilePict;
  IC: HDC;
  ExtRect: TRect;
begin
  if Bool(FMetafileHandle) then DeleteMetafile(FMetafileHandle);
  FMetafileHandle := CloseMetafile(Handle);
  Handle := 0;
{ Подготавливаем метафайл для показа в буфере обмена. }
  MetafilePict := GlobalLock(FClipboardHandle);
```

```
  MetafilePict^.mm := mm_AnIsoTropic;
  IC := CreateIC('DISPLAY', nil, nil, nil);
  SetMapMode(IC, mm_HiMetric);
  ExtRect := FRect;
  DPtoLP(IC, ExtRect, 2);
  DeleteDC(IC);
  MetafilePict^.xExt := ExtRect.Right - ExtRect.Left;
  MetafilePict^.yExt := ExtRect.Top - ExtRect.Bottom;
  MetafilePict^.HMF :=  FMetafileHandle;
  GlobalUnlock(FClipboardHandle);
{ Передаем дескриптор в качестве результата выполнения функции. }
  Result := FClipboardHandle;
end;

procedure TForm1.BitBtn1Click(Sender: TObject);
var
  MetafileCanvas: TMetafileCanvas;
begin
  MetafileCanvas := TMetafileCanvas.Create;
  MetafileCanvas.Rect := Rect(0, 0, 500, 500);
  MetafileCanvas.Ellipse(10, 10, 400, 400);
  Image1.Picture.Metafile.LoadFromClipboardFormat(cf_MetafilePict,
                 MetafileCanvas.MetafileHandle, 0);
  MetafileCanvas.Free;
end;
end.
```

Примечание

Этот пример написан для Win16, поэтому для адаптации под Win32 необходимо заменить вызовы всех функций, этой платформой не поддерживаемых.

Изменение цвета

Как перевести цвет в соответствующий цвет тени?

Пример показывает, как это осуществить, при помощи метода, который применяется в черно-белом телевидении при обработке цветной картинки.

```
function RgbToGray(RGBColor: TColor): TColor;
var
  Gray: byte;
begin
  Gray := Round((0.30 * GetRValue(RGBColor)) + (0.59 * GetGValue(RGBColor)) +
                (0.11 * GetBValue(RGBColor )));
  Result := RGB(Gray, Gray, Gray);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Shape1.Brush.Color := RGB(255, 64, 64);
  Shape2.Brush.Color := RgbToGray(Shape1.Brush.Color);
end;
```

Прозрачные формы и изображения

Код, рисующий одно изображение с прозрачными областями на другом.

Данная процедура рисует исходное изображение на целевом, получая информацию об областях исходного изображения, которые должны остаться в области целевого изображения, имеющей прозрачный цвет.

- `t` – целевой холст для рисования;
- `x, y` – позиция целевого изображения, где должно быть наложено исходное;
- `s` – исходное изображение;
- `TrCol` – цвет, определяющий прозрачность в исходном изображении.

Примечание

Не забывайте обновлять (repaint) целевое изображение.

```
procedure DrawTransparent(t: TCanvas; x,y: Integer; s: TBitmap; TrCol: TColor);
var
  bmpXOR, bmpAND, bmpINVAND, bmpTarget: TBitmap; oldcol: Longint;
begin
  try
    bmpAND := TBitmap.Create;
    bmpAND.Width := s.Width;
    bmpAND.Height := s.Height;
    bmpAND.Monochrome := True;
    oldcol := SetBkColor(s.Canvas.Handle, ColorToRGB(TrCol));
    BitBlt(bmpAND.Canvas.Handle, 0, 0, s.Width,s.Height, s.Canvas.Handle,
           0,0, SRCCOPY);
    SetBkColor(s.Canvas.Handle, oldcol);
    bmpINVAND := TBitmap.Create;
    bmpINVAND.Width := s.Width;
    bmpINVAND.Height := s.Height;
    bmpINVAND.Monochrome := True;
    BitBlt(bmpINVAND.Canvas.Handle, 0, 0, s.Width, s.Height, bmpAND.Canvas.Handle,
           0, 0, NOTSRCCOPY);
    bmpXOR := TBitmap.Create;
    bmpXOR.Width := s.Width;
    bmpXOR.Height := s.Height;
    BitBlt(bmpXOR.Canvas.Handle, 0, 0, s.Width,s.Height, s.Canvas.Handle,
           0, 0, SRCCOPY);
    BitBlt(bmpXOR.Canvas.Handle, 0, 0, s.Width,s.Height, bmpINVAND.Canvas.Handle,
           0, 0, SRCAND);
    bmpTarget := TBitmap.Create;
    bmpTarget.Width := s.Width;
    bmpTarget.Height := s.Height;
    BitBlt(bmpTarget.Canvas.Handle, 0, 0, s.Width,s.Height, t.Handle,
           x, y, SRCCOPY);
    BitBlt(bmpTarget.Canvas.Handle, 0, 0, s.Width,s.Height, bmpAND.Canvas.Handle,
           0, 0, SRCAND);
```

```
      BitBlt(bmpTarget.Canvas.Handle, 0, 0, s.Width,s.Height, bmpXOR.Canvas.Handle,
             0, 0, SRCINVERT);
      BitBlt(t.Handle, x, y, s.Width, s.Height, bmpTarget.Canvas.Handle,
             0, 0, SRCCOPY);
    finally
      bmpXOR.Free;
      bmpAND.Free;
      bmpINVAND.Free;
      bmpTarget.Free;
    end;
  end;
```

Использование пиктограммы в качестве глифа

Решение

```
unit Unit1;
interface
uses
  Windows, SysUtils, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, ExtCtrls;
type
  TForm1 = class(TForm)
    BitBtn1: TBitBtn;
    Button1: TButton;
    Bevel1: TBevel;
    BitBtn2: TBitBtn;
    Label1: TLabel;
    Label2: TLabel;
    SpeedButton1: TSpeedButton;
    procedure BitBtn2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.BitBtn2Click(Sender: TObject);
begin
  close;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  Image1: TImage;
begin
  try
```

```
    Image1 := TImage.Create(Self);     { Создаем временно }
    Image1.Picture.LoadFromFile('c:\Program Files\Common Files\
                                Borland Shared\Images\Icons\Earth.ico');
    with BitBtn1.Glyph do begin        { Сначала работаем с BitBtn }
      Width := 32;                     { Высота и ширина стандартной пиктограммы }
      Height := 32;
      Canvas.Brush.Color := clBtnFace; { Проверяем цвет кнопки }
      Canvas.Rectangle(0, 0, 32, 32);  { Объявляем область изображения }
      Canvas.Draw(0, 0, Image1.Picture.Icon);
    end;
    with SpeedButton1.Glyph do begin  { Теперь делаем то же самое для SpeedButton }
      Width := 32;
      Height := 32;
      Canvas.Brush.Color := clBtnFace;
      Canvas.Rectangle(0, 0, 32, 32);
      Canvas.Draw(0, 0, Application.Icon);
    end;

  finally
    Image1.Free;
    Button1.Enabled := False;       { Убедимся в том, что процесс запущен однажды }
  end;
end;

end.
```

Использование Parser

Несколько процедур для модуля Parser, создающих документацию Delphi-компонентов напрямую из исходного кода.

TDelphiUnitParser – подкласс со специфическими методами, обеспечивающими синтаксический разбор секции Interface модуля Delphi. Можете не включать текст этого наследника TParser в свой код – это просто иллюстрация возможного применения некоторых свойств и методов TParser.

```
function TDelphiUnitParser.CheckSectionBreak: Boolean;
begin
  with Parser do begin
    Result := (Token = toSymbol) and (Compare('Var') or Compare('Const')
            or Compare('Type') or Compare('Implementation') or Compare('Procedure')
            or Compare('Function'));
  end;
end;

procedure TDelphiUnitParser.ParseParameterList;
begin
  with Parser do begin
{ пропускаем '(' }
    NextToken;
    while Token <> ')' do NextToken;
```

```
    NextToken ;
  end;
end;

procedure TDelphiUnitParser.ParseRecord;
begin
  with Parser do begin
{ пропускаем 'record' }
    NextToken;
    while (Token <> toSymbol) or not Compare('End') do begin
      if Token = 'Record' then ParseRecord else NextToken;
    end;
  end;
end;

procedure TDelphiUnitParser.ParseDeclaration;
begin
  with Parser do begin
    while Token <> ';' do begin
      if Token = '(' then ParseParameterList
      else if (Token = toSymbol) and Compare('Record') then ParseRecord
      else NextToken;
    end;
  end;
end;

procedure TDelphiUnitParser.ParseConst;
var
  AString: String ;
  AStart: PChar ;
  EndOfConsts: Boolean ;
begin
  with Parser do begin
    NextToken;
    repeat
      if Token <> toSymbol then ErrorStr('Неопознанный идентификатор');
      AString := TokenString;
      AStart := FSourcePtr;
      NextToken;
      if not(Token in ['=', ':']) then ErrorStr('''=''' or ''':''ожидалось');
      ParseDeclaration;
      FindToken(';');
    until CheckSectionBreak;
  end;
end;
```

[News Group]

Пример использования Parser

С интересом прочли ваш совет про TParser. *Но нужен еще совет. Нам необходимо вычислить математическое выражение с применением стандартных механизмов (приоритеты, порядок и т. д.). Это возможно с помощью* TParser*? Или придется все делать «ручками»?*

На самом деле TParser не является синтаксическим анализатором, скорее это лексический анализатор или сканер. Или, другими словами, входной поток признаков (tokens) в ASCII-коде. Довольно не трудно использовать эти «признаки» для разбора выражения при помощи простой рекурсии. Сделав это, вы сможете произвести разбор математического выражения практически любой сложности. TParser вам поможет, но рекурсивный анализатор придется создавать ручками.

Рассмотрим простой пример:

```
(23.34 + 21.21) * 2.92 - 12.21 * sin (180) * - 1
```

Пример синтаксического анализатора выражений, наследника TParser, который может разобрать вышеуказанное выражение. Он использует следующие определения для выражений:

```
Expr   ::= Term + Expr | Term - Expr | Term
Term   ::= Factor * Term | Factor / Term | Factor
Factor ::= + Item | - Item | Item
Item   ::= ( Expr ) | Fn( Expr ) | Number
Fn     ::= Sin | Cos
Number ::= floating point literal number ( плавающая точка литерала числа )
```

Примечание

TParser *имеет ошибку в подпрограмме разбора плавающего числа. Любое сочетание символов со знаками* '+' *или* '-' *воспринимается как часть плавающего числа, поскольку* 1e+3 – *корректное выражение.*

Естественно, это должно быть правильным только в совокупности с символом 'e'. Поэтому необходимо убедиться, что перед символами '+' и '-' имеется хотя бы один пробел, как показано в нашем выражении. Это можно исправить (если у вас есть исходный код VCL), редактируя функцию TParser.NextToken.

Скопируйте поочередно три приведенных ниже файла и вставьте их в окно редактора Delphi. Самый простой способ – закройте все открытые проекты и создайте новый модуль. Выделите весь текст, сгенерированный Delphi, и вставьте текст модуля ExpParse. Сохраните его под именем ExpParse.pas. Затем создайте другой модуль, перенесите в него EvalForm.pas и также сохраните. Снова закройте файл и создайте новый модуль. Вставьте в него EvalForm.dfm и сохраните, выбрав меню Save as и отметив в списке тип файла DFM. Затем создайте новый проект, удалите форму, созданную по умолчанию и добавьте файл EvalForm.pas.

```
unit ExpParse;

interface

uses Classes;

type
  TExpressionParser = class(TParser)
    protected
      function SkipToken(Value: char): boolean;
      function EvalItem: double; virtual;
      function EvalFactor: double; virtual;
      function EvalTerm: double; virtual;
    public
      function EvalExpr: double;
  end;

implementation
uses SysUtils;

function TExpressionParser.SkipToken(Value: char): boolean;
begin
{ возвращаем истину, если текущий признак Value, и если так, то получаем следующий
  признак }
  Result := Token = Value;
  if Result then NextToken;
end;

function TExpressionParser.EvalItem: double;
var
  Expr: double;
  Fn: integer;
begin
  case Token of
    toInteger: Result := TokenInt;
      toFloat: Result := TokenFloat;
          '(': begin
                 NextToken;
                 Result := EvalExpr;
                 CheckToken(')');
               end;
     toSymbol: begin
                 if CompareText(TokenString, 'SIN') = 0 then Fn := 1
                 else if CompareText(TokenString, 'COS') = 0 then Fn := 2
                 else Raise EParserError.CreateFmt('Неизвестный элемент "%s"',
                                                   [TokenString]);
                 NextToken;
                 CheckToken('(');
                 NextToken;
                 Expr := EvalExpr;
                 CheckToken(')');
```

```
                case Fn of
                    1: Result := SIN(Expr);
                    2: Result := COS(Expr);
                end;
              end;
        else
          Raise EParserError.CreateFmt('Неожидаемый символ "%s"', [Token]);
  end;
  NextToken;
end;

function TExpressionParser.EvalFactor: double;
begin
  case Token of
    '+': begin
           NextToken;
           Result := EvalItem;
         end;
    '-': begin
           NextToken;
           Result := -EvalItem;
         end;
    else
      Result := EvalItem;
  end;
end;

function TExpressionParser.EvalTerm: double;
var
  AToken: char;
begin
  Result := EvalFactor;
  if SkipToken('*') then Result := Result * EvalTerm
  else if SkipToken('/') then Result := Result / EvalTerm;
end;

function TExpressionParser.EvalExpr: double;
begin
  Result := EvalTerm;
  if SkipToken('+') then Result := Result + EvalExpr
  else if SkipToken('-') then Result := Result - EvalExpr;
end;
end.
unit EvalForm;

interface
uses
  Windows, SysUtils, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExpParse;
type
  TForm1 = class(TForm)
```

```
    Edit1: TEdit;
    Label1: TLabel;
    Button1: TButton;
    Label2: TLabel;
    procedure Button1Click(Sender: TObject);
  end;

var
  Form1: TForm1;

implementation
{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
var
  s: string;
  MemStream: TMemoryStream;
  ExpressionParser: TExpressionParser;
begin
  s := Edit1.Text;
{ создаем поток для работы с памятью, содержащий текст, - TParser может разбирать
  выражения из потока}
  MemStream := TMemoryStream.Create;
  try
    MemStream.SetSize(Length(s));
    MemStream.WriteBuffer(s[1], Length(s));
    MemStream.Position := 0;
{ создаем анализатор выражения, используя поток }
    ExpressionParser := TExpressionParser.Create(MemStream);
    try
      Label2.Caption := Format('Результат = %g', [ExpressionParser.EvalExpr]);
    finally
      ExpressionParser.Free;
    end;
  finally
    MemStream.Free;
  end;
end;
end.
```

Модуль EvalForm.dfm:

```
object Form1: TForm1
  Left = 216
  Top = 102
  Width = 433
  Height = 300
  Caption = 'Form1'
  Font.Color = clWindowText
  Font.Height = -13
  Font.Name = 'System'
  Font.Style = []
```

```
  PixelsPerInch = 96
  TextHeight = 16
  object Label1: TLabel
    Left = 8
    Top = 8
    Width = 74
    Height = 16
    Caption = 'Выражение'
  end
  object Label2: TLabel
    Left = 120
    Top = 72
    Width = 297
    Height = 16
    AutoSize = False
    Caption = 'Результат='
  end
  object Edit1: TEdit
    Left = 8
    Top = 27
    Width = 409
    Height = 24
    TabOrder = 0
    Text = '(23.34 + 21.21) * 2.92 - 12.21 * sin (180) * - 1'
  end
  object Button1: TButton
    Left = 8
    Top = 64
    Width = 89
    Height = 33
    Caption = 'Оценка'
    Default = True
    TabOrder = 1
    OnClick = Button1Click
  end
end
```

Примечание

Пример написан с ошибками распознавания вещественных чисел, но идея заслуживает внимания. В общем, желательно иметь анализатор, который умел бы работать без разделительных пробелов, но, и встретившись с пробелом, не «пасовал» бы перед ним.

Преобразование PChar в StringList

Допустим, у нас имеется P типа PChar, содержащий символы перевода строки #13 или #13#10 и TStringList с именем TS, для копирования P в TS достаточно следующей команды:

```
TS.SetText(P);
```

[News Group]

Создание списка StringList с объектами

Как создать TStringList*, содержащий в строке имя объекта, и сам объект* TStringList*?*

Компонент TStringList имеет возможность хранить для каждой строки свой указатель: см. свойство Objects. Чтобы понять принцип работы с указателями, обратитесь к описанию метода TStringList.AddObject в электронной справке. Пример работы с методом:

```
StringList1.AddObject('Имя списка', TStringList.Create);
```

Предупреждение

Delphi не удаляет эти объекты. Вы должны позаботиться об этом сами.

Доступ к связанному StringList можно получить, назначая его переменной TStringList:

```
TempStringList := TStringList(StringList1.Objects[index]);
```

Приведенный ниже код правильно компилируется, выполняется и демонстрирует все вышесказанное. Form1 имеет только один компонент Label.

```
unit Unit1;

interface

uses
  Windows, SysUtils, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;
type
  TForm1 = class(TForm)
      Label1: TLabel;
      procedure FormCreate(Sender: TObject);
      procedure FormDestroy(Sender: TObject);
    public
      StringList1, TempStringList: TStringList;
  end;

var
  Form1: TForm1;
implementation
{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
  StringList1 := TStringList.Create;
  StringList1.AddObject('имя', TSTringList.Create);
  TempStringList := TStringList(StringList1.Objects[0]);
  TempStringList.Add('Привет');
  Label1.Caption := TempStringList[0];
```

```
end;

procedure TForm1.FormDestroy(Sender: TObject);
var
  i: Longint;
begin
  for i := 0 to StringList1.Count - 1 do begin
    TempStringList := TStringList(StringList1.Objects[i]);
    TempStringList.Free;
  end;
end;

end.
```

StringList, владеющий объектами

Решение 1

В этом случае нужно освобождать объекты `StringList` точно так же, как мы это делали без него. `StringList.Free` просто очистит список и строки, но не связанные с ними объекты. Дело в том, что вы могли бы иметь указатели на объекты, например, в двух `StringList`, и вы могли бы захотеть освободить в одном список строк без освобождения объектов, содержащихся в другом.

Можно создать свой собственный класс, автоматизирующий процедуру освобождения объектов. Образец такого класса:

```
TOwnerStringList = class(TStringList)

private
  FOwnsObjects: Boolean;
public
  constructor Create(AOwnsObjects: Boolean); override;
  destructor Destroy; override;
end;

constructor TOwnerStringList.Create(AOwnsObjects: Boolean);
begin
  inherited Create;
  FOwnsObjects := AOwnsObjects;
end;
destructor TOwnerStringList.Destroy;
var
  I: Integer;
begin
  if FOwnsObjects then
    for I := 0 to MyStrLst.Count - 1 do
      TObject(Objects[I]).Free;
  inherited Destroy;
end;
```

Теперь, когда у нас есть взаимнооднозначное соответствие между списком строк и имеющимися объектами, для создания нового списка, который уничтожал бы объекты при освобождении списка строк, нам необходимо выполнить команду `MyStringList := TOwnerStringList.Create(True)`. Для того чтобы список вел себя как обычно, передавайте в методе `Create` в качестве параметра `False` или просто используйте нормальный `TStringList`.

Решение 2

`TStrings` – абстрактный базовый класс, используемый многими визуальными компонентами, такими как, например, `TListBox`. Все, что вы хотите, имеется в `TStringList`, а если нужно следить только за объектами, то вместо него используйте `TList`. Для добавления элемента в конец списка применяется метод `Add`, для добавления элемента в определенное место списка – метод `Insert`. Для получения строки из списка используется свойство `Items`.

Обратите внимание, что счет начинается с нуля, поэтому последний элемент имеет порядковый номер `Count-1`.

Строка удаляется посредством метода `Delete`. Для нахождения строки в списке используется `IndexOf`. Можно сделать так, чтобы `TStringList` хранил список в алфавитном порядке. Чтобы сделать это, прежде чем добавить что-то к списку, установите свойство `Sorted` в `True`. С помощью `TStringList` может также организовать хранение для каждой строки ссылки на объект; осуществить это можно с помощью `AddObject` и свойства `Objects`. Компонент `TList` делает вышесказанное, но без строк. Для создания `TStringList` сделайте следующее:

```
procedure MakeList;
var
  aList: TStringList;
begin
  aList := TStringList.Create;
  aList.Sorted := true;                    { по выбору }
  aList.Duplicates := dupIgnore;           { или dupAccept, или dupError }
  aList.Add('Строка 1');
  aList.Add('Строка 2');
  Edit1.Text := aList.Items[0];            { Edit1 теперь содержит 'Строка 1' }
  aList.Delete(0);
  aList.Free;
end;
```

StringList и потоки

Возможно ли выполнение `StringList.SaveToStream`*, если еще не завершен процесс* `WriteComponent`*? Можно ли для создания кода, сохраняющего каждую строку, воспользоваться* `Stream.WriteStr` *или надо сохранять длину в байтах плюс сами символы?*

Вы можете записывать строки, если определите, что список строк вместо `DefineBinaryProperty` должен использовать `DefineProperty`. Чтение и запись

должны выполняться соответственно с помощью методов TReader и TWriter. Записывать можно приблизительно так:

```
Writer.WriteListBegin;
  for i := 0 to TheStringList.Count - 1 do
    Writer.WriteString(TheStringList[i]);
Writer.WriteListEnd;
```

Для чтения можно воспользоваться следующим кодом:

```
Reader.ReadListBegin;
  while not Reader.EndOfList do
    TheStringList.Add(Reader.ReadString);
Reader.ReadListEnd;
```

[News Group]

Запись строки в поток с помощью TWriter/TReader

Что нужно сделать, чтобы с помощью TWriter/TReader записать строку в поток?

Автор этого совета до сих пор для простоты использует TMemoryStream. Ключевыми являются вызовы Read/WriteListBegin и Read/WriteListEnd. Без них будет получено исключение.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  sWrite, sRead: string[25];
  MyStream: TMemoryStream;
  MyWriter: TWriter;
  MyReader: TReader;
begin
  MyStream := TMemoryStream.Create;
  MyStream.SetSize(4096);
  MyWriter := TWriter.Create(MyStream, 4096);
  sWrite := 'sWriteContents';
  MyWriter.WriteListBegin;
  MyWriter.WriteString(sWrite);
  MyWriter.WriteListEnd;
  MyWriter.Free;
  MyStream.Seek(0, 0);
  MyReader := TReader.Create(MyStream, 4096);
  MyReader.ReadListBegin;
  sRead := MyReader.ReadString;
  MyReader.ReadListEnd;
  MyReader.Free;
  Label1.Caption := sRead;
  MyStream.Free;
end;
```

Встроенные форматы буфера обмена

Где бы раздобыть список встроенных в Windows 9x форматов буфера обмена и соответствующие им номера?

Эту информацию можно получить посредством следующей процедуры:

```
procedure TForm1.Button1Click(Sender: TObject);

var
  buf: Array [0..60] of Char;
  n: Integer;
  fmt: Word;
  name: String[30];

begin
  MemFormats.Clear;
  for n := 0 to Clipboard.FormatCount-1 do begin
    fmt := Clipboard.Formats[n];
    if GetClipboardFormatName(fmt, buf, Pred(Sizeof(buf))) <> 0 then
      MemFormats.Lines.Add(StrPas(buf))

    else begin
      case fmt of
                  1: name := 'CF_TEXT';
                  2: name := 'CF_BITMAP';
                  3: name := 'CF_METAFILEPICT';
                  4: name := 'CF_SYLK';
                  5: name := 'CF_DIF';
                  6: name := 'CF_TIFF';
                  7: name := 'CF_OEMTEXT';
                  8: name := 'CF_DIB';
                  9: name := 'CF_PALETTE';
                 10: name := 'CF_PENDATA';
                 11: name := 'CF_RIFF';
                 12: name := 'CF_WAVE';
                 13: name := 'CF_UNICODETEXT';
                 14: name := 'CF_ENHMETAFILE';
                 15: name := 'CF_HDROP(Win 95)';
                 16: name := 'CF_LOCALE(Win 95)';
                 17: name := 'CF_MAX(Win 95)';
               $0080: name := 'CF_OWNERDISPLAY';
               $0081: name := 'CF_DSPTEXT';
               $0082: name := 'CF_DSPBITMAP';
               $0083: name := 'CF_DSPMETAFILEPICT';
               $008E: name := 'CF_DSPENHMETAFILE';
        $0200..$02FF: name := 'частный формат';
        $0300..$03FF: name := 'Объект GDI';
        else
          name := 'неизвестный формат';
      end;
```

```
        MemFormats.Lines.Add( name );
      end;
    end;

end;
```

[News Group]

Примечание

В uses *дописываем* Clipbrd. MemFormats – Memo *для отображения результатов работы процедуры.*

Копирование в буфер обмена

Как в приложении выполнить копирование в буфер обмена?

Решение

```
procedure CopyButtonClick(Sender: TObject);
begin
  if ActiveControl is TMemo then TMemo(ActiveControl).CopyToClipboard;
  if ActiveControl is TDBMemo then TDBMemo(ActiveControl).CopyToClipboard;
  if ActiveControl is TEdit then TEdit(ActiveControl).CopyToClipboard;
  if ActiveControl is TDBedit then TDBedit(ActiveControl).CopyToClipboard;
end;

procedure PasteButtonClick(Sender: TObject);
begin
  if ActiveControl is TMemo then TMemo(ActiveControl).PasteFromClipboard;
  if ActiveControl is TDBMemo then TDBMemo(ActiveControl).PasteFromClipboard;
  if ActiveControl is TEdit then TEdit(ActiveControl).PasteFromClipboard;
  if ActiveControl is TDBedit then TDBedit(ActiveControl).PasteFromClipboard;
end;
```

Просмотр буфера обмена

Решение 1

Пример на основе простого модуля-класса, осуществляющего просмотр буфера обмена.

```
unit ClipboardViewer;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;
type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
```

```
    procedure FormDestroy(Sender: TObject);
  private
    FNextViewerHandle: THandle;
    procedure WMDrawClipboard(var message: TMessage); message WM_DRAWCLIPBOARD;
    procedure WMChangeCBCHain(var message: TMessage); message WM_CHANGECBCHAIN;
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
// Проверяем работоспособность функции. При невозможности просмотра буфера обмена
// функция возвратит значение Nil.
  FNextViewerHandle := SetClipboardViewer(Handle);
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  ChangeClipboardChain(Handle, FNextViewerHandle);   // Восстанавливаем цепочки.
end;

procedure TForm1.WMDrawClipboard(var message: TMessage);
begin
  message.Result := SendMessage(WM_DRAWCLIPBOARD, FNextViewerHandle, 0, 0);
end;

procedure TForm1.WMChangeCBCHain(var message: TMessage);
begin
// Вызывается при любом изменении цепочек буфера обмена.
  if message.wParam = FNextViewerHandle then begin

// Удаляем следующую цепочку просмотра. Корректируем внутреннюю переменную.
    FNextViewerHandle := message.lParam;

// Возвращаем 0, чтобы указать, что сообщение было обработано
    message.Result := 0;
  end else begin

// Передаем сообщение следующему окну в цепочке.
    message.Result := SendMessage(FNextViewerHandle, WM_CHANGECBCHAIN,
                                  message.wParam, message.lParam);
  end;
end;

end.
```

Решение 2

Вот участок кода программы, отображающий текст буфера обмена. Расположите компонент Memo на главной форме нового проекта, присвойте свойству Align значение alClient, добавьте необходимые частные поля и методы и создайте их реализацию следующим образом:

```
  ...
  private
    PrevHwnd: Hwnd;

    procedure WMChangeCBChain(var Msg: TWMChangeCBChain); message WM_CHANGECBCHAIN;

    procedure WMDrawClipboard(var Msg: TWMDrawClipboard); message WM_DRAWCLIPBOARD;
  ...
procedure TForm1.WMChangeCBChain(var Msg: TWMChangeCBChain);
begin
  if PrevHWnd = Msg.Remove then PrevHWnd := Msg.Next;
  if Msg.Remove <> Handle then
   SendMessage(PrevHWnd, WM_CHANGECBCHAIN, Msg.Remove, Msg.Next);
end;

procedure TForm1.WMDrawClipboard(var Msg: TWMDrawClipboard);
var
  P: PChar;
  H: THandle;
  Len: DWORD;
begin
  SendMessage(PrevHWnd, WM_DRAWCLIPBOARD, 0, 0);
  if Clipboard.HasFormat(CF_TEXT) then begin
    H := Clipboard.GetAsHandle(CF_TEXT);
    Len := GlobalSize(H) + 1;
    P := GlobalLock(H);
    Memo1.SetTextBuf(P);
    GlobalUnlock(H);
  end;
  Msg.Result := 0;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  PrevHwnd := SetClipboardViewer(Handle);
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  ChangeClipboardChain(Handle, PrevHwnd);
end;
```

[News Group]

Копирование большого файла в буфер обмена

Общее решение, которое будет работать, даже если размер файла превышает 64 Кбайт:

```
function _hread(FileHandle: word; BufPtr: pointer; ByteCount: longint): longint;
                far; external 'KERNEL32' index 349;

procedure CopyFileToClipboard(Const fname: String);
var
  hmem, hFile: THandle;
  size: LongInt;
  p: Pointer;
begin
  hFile := FileOpen(fname, fmOpenRead);
  try
    size := FileSeek(hFile, 0, 2);
    FileSeek(hfile, 0, 0);
    if size > 0 then begin
      hmem := GlobalAlloc(GHND, size);
      if hMem <> 0 then begin
        p := GlobalLock(hMem);
        if p <> Nil then begin
          _hread(hFile, p, size);
          GlobalUnlock(hMem);
          Clipboard.SetAsHandle(CF_TEXT, hMem);
        end else GlobalFree(hMem);
      end;
    end;
  finally
    FileClose(hFile);
  end;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  fname: String;
begin
  if OpenDialog1.Execute then begin
    fname := OpenDialog1.Filename;
    CopyFileToClipboard(fname);
  end;
end;
```

[News Group]

Примечание

Добавляем в секцию uses *модуль* Clipbrd.

Буфер обмена и потоки

Как удобнее работать с буфером обмена? Как с последовательностью байт?

Используйте потоки.

```
unit ClipStrm;

interface uses Classes, Windows;

type
  TClipboardStream = class(TStream)

  private
    FMemory: pointer;
    FSize: longint;
    FPosition: longint;
    FFormat: word;
  public
    constructor Create(fmt: word);
    destructor Destroy; override;
    function Read(var Buffer; Count: Longint): Longint; override;
    function Write(const Buffer; Count: Longint): Longint; override;
    function Seek(Offset: Longint; Origin: Word): Longint; override;
  end;

implementation

uses
  SysUtils;

constructor TClipboardStream.Create(fmt: word);
var
  tmp: pointer;
  FHandle: THandle;
begin
  FFormat := fmt;
  OpenClipboard(0);
  FHandle := GetClipboardData(FFormat);
  FSize := GlobalSize(FHandle);
  FMemory := AllocMem(FSize);
  tmp := GlobalLock(FHandle);
  MoveMemory(FMemory, tmp, FSize);
  GlobalUnlock(FHandle);
  FPosition := 0;
  CloseClipboard;
end;

destructor TClipboardStream.Destroy;
begin
  FreeMem(FMemory);
end;
```

```
function TClipboardStream.Read(var Buffer; Count: longint): longint;
begin
  if FPosition + Count > FSize then Result := FSize - FPosition
  else Result := Count;
  MoveMemory(@Buffer, PChar(FMemory) + FPosition, Result);
  Inc(FPosition, Result);
end;

function TClipboardStream.Write(const Buffer; Count: longint): longint;
var
  FHandle: HGlobal;
  tmp: pointer;
begin
  ReallocMem(FMemory, FPosition + Count);
  MoveMemory(PChar(FMemory) + FPosition, @Buffer, Count);
  FPosition := FPosition + Count;
  FSize := FPosition;
  FHandle := GlobalAlloc(GMEM_MOVEABLE or GMEM_SHARE or GMEM_ZEROINIT, FSize);
  try
    tmp := GlobalLock(FHandle);
      try
        MoveMemory(tmp, FMemory, FSize);
        OpenClipboard(0);
        SetClipboardData(FFormat, FHandle);
      finally
        GlobalUnlock(FHandle);
      end;
      CloseClipboard;
  except
    GlobalFree(FHandle);
  end;

  Result := Count;
end;

function TClipboardStream.Seek(Offset: Longint; Origin: Word): Longint;
begin
  case Origin of
    0: FPosition := Offset;
    1: Inc(FPosition, Offset);
    2: FPosition := FSize + Offset;
  end;
  Result := FPosition;
end;

end.
```

[Mahotkin Alexey]

Поддержка команд Cut, Copy, Paste

Решение 1

Используйте следующие процедуры. Вызывайте их при выборе соответствующих пунктов меню. Это будет работать со всеми «редактируемыми» элементами управления. Но для TTree нужно использовать специальные сообщения редактирования.

```
procedure TForm1.CopyClick(Sender: TObject);
var
  Mes: TWMCopy;
begin
  Mes.Msg := WM_COPY;
  Screen.ActiveControl.Dispatch(Mes);
end;

procedure TForm1.CutClick(Sender: TObject);
var
  Mes: TWMCut;
begin
  Mes.Msg := WM_CUT;
  Screen.ActiveControl.Dispatch(Mes);
end;

procedure TForm1.PasteClick(Sender: TObject);
var
  Mes: TWMPaste;
begin
  Mes.Msg := WM_PASTE;
  Screen.ActiveControl.Dispatch(Mes);
end;

procedure TForm1.UndoClick(Sender: TObject);
var
  Mes: TWMUndo;
begin
  Mes.Msg := WM_UNDO;
  Screen.ActiveControl.Dispatch(Mes);
end;
```

[Shejchenko Andrij]

Решение 2

Свойство формы ActiveControl позволяет получить ссылку на активный в данный момент элемент управления. Но не все элементы управления могут работать с буфером обмена. Если хотите работать только с компонентами Edit и Memo, то самый простой метод для CopyToClipboard:

```
ActiveControl.Perform(WM_COPY, 0, 0);
```

Для PasteFromClipboard:

```
ActiveControl.Perform(WM_PASTE, 0, 0);
```

Если элемент управления «не понимает» посланных сообщений, то это никак не скажется на его работе, он просто проигнорирует их.

Другим способом является проверка типа во время выполнения приложения:

```
if ActiveControl is TCustomEdit then
  TCustomEdit(ActiveControl).CopyToClipboard;
```

[News Group]

Решение 3

Реализация команд Cut, Copy и Paste средствами WinAPI:

```
SendMessage(GetFocus, WM_CUT, 0, 0);
SendMessage(GetFocus, WM_COPY, 0, 0);
SendMessage(GetFocus, WM_PASTE, 0, 0);
```

[News Group]

Решение 4

Есть два шага, положенных в основу работы с буфером обмена. Во-первых, нужно знать, какие пункты меню Правка должны быть в данный момент активизированы. Во-вторых, необходимо работать с тем элементом управления, который в данный момент выбран.

```
procedure TForm1.Edit1Click(Sender: TObject);
begin
  if ActiveControl is TCustomEdit then begin
    with TCustomEdit(ActiveControl) do begin
      Cut1.Enabled := SelLength > 0;
      Copy1.Enabled := SelLength > 0;
      Paste1.Enabled := ClipBoard.HasFormat(CF_TEXT);
    end;
  end else begin
    Cut1.Enabled := False;
    Copy1.Enabled := False;
    Paste1.Enabled := False;
  end;
end;

procedure TForm1.Cut1Click(Sender: TObject);
begin
  if ActiveControl is TDBEdit then
    with TDBEdit(ActiveControl).DataSource.DataSet do Edit;
  TCustomEdit(ActiveControl).CutToClipboard;
  if ActiveControl is TDBEdit then
    with TDBEdit(ActiveControl).DataSource.DataSet do Post;
end;
```

```
procedure TForm1.Copy1Click(Sender: TObject);
begin
  TCustomEdit(ActiveControl).CopyToClipboard;
end;

procedure TForm1.Paste1Click(Sender: TObject);
begin
  if ActiveControl is TDBEdit then
    with TDBEdit(ActiveControl).DataSource.DataSet do Edit;
  TCustomEdit(ActiveControl).PasteFromClipboard;
  if ActiveControl is TDBEdit then
    with TDBEdit(ActiveControl).DataSource.DataSet do Post;
end;
```

Edit1 – меню редактирования верхнего уровня. Если по нему щелкают, то прежде чем меню «вывалится» вниз, необходимо проверить, принадлежит ли текущий активный элемент управления некоторым типам редактирования. Если это условие выполняется, активизируются пункты меню Вырезать и Копировать, и если есть текст в буфере обмена, то и пункт Вставить. Если нет, то все три пункта будут недоступны.

Для копирования содержимого элемента редактирования достаточно просто вызвать CopyToClipboard; это не проблема. Для вырезания и вставки необходимо «изменить» содержимое активного элемента редактирования – если это DBEdit, необходимо перейти в режим редактирования и после манипуляций с данными буфера обмена сохранить измененные данные.

[News Group]

Копирование формы в буфер обмена

Как скопировать в буфер обмена форму в виде графического изображения?

Решение

```
uses Clipbrd;
procedure TForm1.Kopieren1Click(Sender: TObject);
var
  bitmap: TBitmap;
begin
  bitmap := TBitmap.Create;
  bitmap.Width := ClientWidth;
  bitmap.Height := ClientHeight;
  try
    with bitmap.Canvas do
      CopyRect(ClientRect, Canvas, ClientRect);
      Clipboard.Assign(bitmap);
  finally
    bitmap.Free;
  end;
end;
```

Индикатор хода выполнения в консольном приложении

Данный unit содержит описание класса для отображения в консольном окне хода выполнения какой-либо длительной операции.

Период, с которым будет производиться вывод на экран, задается функцией SetPause(), в качестве параметра в нее передается период (в секундах) вывода сообщений на экран. Сообщение может содержать ключевые слова:

- MIN – минимальное значение прогресса;
- MAX – максимальное значение прогресса;
- CURRENT – текущая позиция;
- PROGRESS – прогресс (процент завершения);
- TIME – время, в течение которого работает программа;
- LEFTTIME – сколько приблизительно осталось работать;
- TOTALTIME – общее время работы процесса.

Все ключевые слова должны предваряться escape-символом, который по умолчанию равен '#', но его можно менять функцией SetEsc().

Формат вывода процентов задается переменной ProgressFmt (см. справку Delphi по функции FormatFloat). По умолчанию формат равен '00.00', т. е. процент завершения выводится с точностью до второго знака.

Текущее значение параметра, характеризующего ход выполнения (см. переменную i в примере) задается функцией SetCurr. Если значение этого параметра меняется при каждой итерации на единицу, можно использовать функцию Inc (не системную, а принадлежащую классу TConsoleProgress).

Функция SetCurr возвращает код нажатой клавиши, если пользователем была нажата клавиша, код которой определяется параметром WaitKeys. Это массив из кодов клавиш, нажатие которых необходимо «отлавливать». Добавление в список дополнительных кодов клавиш осуществляется функцией AddWaitKey, удаление из списка – DelWaitKey.

Пример использования (в консольном приложении):

```
program Project1;

{$AppType Console}

uses
  SysUtils,
  progress in 'Progress.pas';

const
  g = 10000000;

var
  p: TConsoleProgress;
  i: Integer;
```

```
begin
  p := TConsoleProgress.Create(0, g, 1, 'Минимум = #MIN, Максимум = #max,
                               Позиция = #CURRENT, Прогресс = #ProGreSs%');
  p.AddWaitKey($1B);
  for i := 0 to g do
    if p.SetCurr(i) <> 0 then WriteLn('Escape key is pressed!');
  p.Free;
end.
```

Собственно unit:

```
unit Progress;

interface

uses
  SysUtils, Windows;
const
  DEF_MAX_STR_LEN = 79;                // Максимальное число символов в строке
  W_Num = 7;                           // Общее число обрабатываемых команд
  W: array [0..W_Num - 1] of String = ('MIN', 'MAX', 'CURRENT', 'PROGRESS',
                                       'TIME', 'LEFTTIME', 'TOTALTIME');
type
  TConsoleProgress = class
    private
      LastTime: TDateTime;
      Current: Integer;      // текущее значение
      Pause: TDateTime;      // пауза между отображением прогресса
      StartTime: TDateTime;  // время начала тестирования
      ProgressFmt: String;   // формат команды FormatFloat для вывода процентов
      ParamCount: Integer;   // сколько параметров в строке Text
      TextParts: array of String;  // кусочки Text
      ParamNums: array of Integer; // номера команд в порядке появления
      WaitKeys: array of Word;     // коды клавиш, нажатие на которые заставит
                                   // функции SetCurr и Inc вернуть значение false;
      isForce: Boolean;            // форсировать вывод на экран, не дожидаясь
                                   // завершения интервала ожидания
      Con_Hnd: THandle;            // дескриптор консольного окна
      CCI: TConsoleCursorInfo;     // информация о курсоре
      function StrToOEM(str: String): String;   // конвертор строк в OEM
      function KeyHook: Word;
      procedure CursorON;          // включение курсора
      procedure CursorOFF;         // выключение курсора
    public
      Esc: Char;            // Escape-символ, с которого начинаются служебные слова
      Min, Max: Integer;    // минимальное и максимальное значения
      TimeFmt, LeftTimeFmt, TotalTimeFmt: String; // форматы вывода времени работы
                                                  // и оставшегося времени
      MaxStrLength: Integer;      // максимальная длина строки вывода на экран
      ClearOnComplete: Boolean;   // очищать ли строку вывода при достижении 100%
      NewLineOnComplete: Boolean; // ставить перевод строки при достижении 100%
```

```
      isKeyPressed: Boolean;      // была ли нажата в процессе работы хотя бы одна
                                  // клавиша
      PercentDone: Real;          // процент выполнения процесса
      constructor Create; overload;  // конструктор без параметров (по умолчанию)
      constructor Create(Max: Integer; Text: String); overload;
      constructor Create(Min, Max: Integer; Pause: Real; Text: String); overload;
      procedure Init;                // инициализация, обнуление всех счетчиков
      procedure SetPause(Pause: Real);
      procedure SetText(Text: String);
      procedure AddWaitKey(KeyCode: Word);
      procedure DelWaitKey(KeyCode: Word);
      function Inc(Step: Integer): Word;
      function SetCurr(CurrentPos: Integer): Word;
      procedure ShowP;
      procedure ForceShow;           // форсировать вывод на экран
end;

implementation

{ TConsoleProgress }

constructor TConsoleProgress.Create;
begin
  Min := 0;
  Max := 0;
  Pause := EncodeTime(0, 0, 1, 0);
  Esc := '#';
  SetText('Done #PROGRESS%');
  ProgressFmt := '00.00';
  TimeFmt := 'h:nn:ss';
  LeftTimeFmt := 'h:nn:ss';
  TotalTimeFmt := 'h:nn:ss';
  LastTime := Now;
  MaxStrLength := DEF_MAX_STR_LEN;
  GetConsoleCursorInfo(Con_Hnd, CCI);
  Init;
end;

constructor TConsoleProgress.Create(Max: Integer; Text: String);
begin
  Min := 0;
  Self.Max := Max;
  Pause := EncodeTime(0, 0, 1, 0);
  Esc := '#';
  SetText(Text);
  ProgressFmt := '00.00';
  TimeFmt := 'h:nn:ss';
  LeftTimeFmt := 'h:nn:ss';
  TotalTimeFmt := 'h:nn:ss';
  LastTime := Now;
  MaxStrLength := DEF_MAX_STR_LEN;
```

```
  Init;
end;
constructor TConsoleProgress.Create(Min, Max: Integer; Pause: Real; Text: String);
begin
  Self.Min := Min;
  Self.Max := Max;
  SetPause(Pause);
  Esc := '#';
  SetText(Text);
  ProgressFmt := '00.00';
  TimeFmt := 'h:nn:ss';
  LeftTimeFmt := 'h:nn:ss';
  TotalTimeFmt := 'h:nn:ss';
  LastTime := Now;
  MaxStrLength := DEF_MAX_STR_LEN;
  Init;
end;

procedure TConsoleProgress.DelWaitKey(KeyCode: Word);
var
  i, j: Integer;
begin
  i := 0;
  while (i < Length(WaitKeys)) and (WaitKeys[i] <> KeyCode) do
    System.Inc(i);
  if i < Length(WaitKeys) then begin
    for j := i + 1 to Length(WaitKeys) - 1 do
    WaitKeys[j - 1] := WaitKeys[j];
    SetLength(WaitKeys, Length(WaitKeys) - 1);
  end;
end;

function TConsoleProgress.Inc(Step: Integer): Word;
begin
  System.Inc(Current, Step);
  ShowP;
  Result := KeyHook;
end;

procedure TConsoleProgress.Init;
begin
  Current := Min;
  StartTime := Now;
  isKeyPressed := false;
  Con_Hnd := GetStdHandle(STD_OUTPUT_HANDLE);
  GetConsoleCursorInfo(Con_Hnd,CCI);
  CursorOFF;
end;

function TConsoleProgress.SetCurr(CurrentPos: Integer): Word;
begin
  Current := CurrentPos;
```

```
  ShowP;
  Result := KeyHook;
end;

procedure TConsoleProgress.SetPause(Pause: Real);
begin
  Self.Pause := Pause/24/60/60;
end;

procedure TConsoleProgress.SetText(Text: String);
var
  i, p: Integer;
begin
  Text := StrToOEM(Text + Char($0D));
  ParamCount := 0;
  SetLength(TextParts, 1);
  p := Pos(Esc, Text);
  if p = 0 then p := Length(Text) - 1;
  TextParts[0] := Copy(Text, 1, p - 1);
  while p <= Length(Text) do begin
    if Text[p] = Esc then begin
      i := 0;
      while (i < W_Num) and (UpperCase(Copy(Text, p + 1,
                             Length(W[i]))) <> W[i]) do
        System.Inc(i);
      if i < W_Num then begin
        System.Inc(ParamCount);
        SetLength(TextParts, ParamCount + 1);
        SetLength(ParamNums, ParamCount);
        ParamNums[ParamCount - 1] := i;
        System.Inc(p, Length(W[i]) + 1);
      end;
    end;
    TextParts[ParamCount] := TextParts[ParamCount] + Text[p];
    System.Inc(p);
  end;
end;

procedure TConsoleProgress.ShowP;
var
  i: Integer;
  c: String;
begin
  if ((Now - LastTime) < Pause) and (Current < Max) and not isForce then Exit;
  c := TextParts[0];
  for i := 0 to ParamCount - 1 do begin
    case ParamNums[i] of
        0:                      // Команда MIN
             c := c + IntToStr(Min);
        1:                      // Команда MAX
             c := c + IntToStr(Max);
```

```
        2:                      // Команда CURRENT
             c := c + IntToStr(Current);
        3:   begin              // Команда PROGRESS
               PercentDone := (Current - Min) / (Max - Min) * 100;
               if (Max - Min) <> 0 then
                 c := c + FormatFloat(ProgressFmt, PercentDone);
             end;
        4:                      // Команда TIME
             c := c + FormatDateTime(TimeFmt, Now - StartTime);
        5:                      // Команда LEFTTIME
             if (Current - Min) > 0 then
               c := c + FormatDateTime(LeftTimeFmt, (Now - StartTime) *
                    (Max - Current) / (Current - Min));
        6:                      // Команда TOTALTIME
             if (Current - Min) > 0 then
               c := c + FormatDateTime(TotalTimeFmt,
                    (Now - StartTime) * (Max - Min) / (Current - Min));
    end;
    c := c + TextParts[i + 1];
  end;
  if Length(c) > MaxStrLength then SetLength(c, MaxStrLength);
  write(c);
  if (Current = Max) then begin
    if ClearOnComplete then begin
      write(StringOfChar(' ', DEF_MAX_STR_LEN));
      if not NewLineOnComplete then Write(Char($0D));
    end;
    if NewLineOnComplete then WriteLn;
    CursorON;
  end;
  LastTime := Now;
end;

function TConsoleProgress.StrToOEM(str: String): String;
begin
  Result := str;
  CharToOEM(PChar(Result), PChar(Result));
end;

procedure TConsoleProgress.AddWaitKey(KeyCode: Word);
begin
  SetLength(WaitKeys, Length(WaitKeys) + 1);
  WaitKeys[Length(WaitKeys) - 1] := KeyCode;
end;

function TConsoleProgress.KeyHook: Word;
var
  i: Integer;
  ks: TKeyboardState;
begin
  Result := 0;
```

```
  for i := 0 to Length(WaitKeys) - 1 do
    if (GetKeyState(WaitKeys[i]) and $80) = $80 then begin
      Result := WaitKeys[i];
      ks[Result] := 0;
      SetKeyboardState(ks);
      isKeyPressed := true;
      Exit;
    end;
end;

procedure TConsoleProgress.ForceShow;
begin
  isForce := true;
  ShowP;
end;

procedure TConsoleProgress.CursorOFF;
begin
  CCI.bVisible := false;
  SetConsoleCursorInfo(Con_Hnd, CCI);
end;

procedure TConsoleProgress.CursorON;
begin
  CCI.bVisible := true;
  SetConsoleCursorInfo(Con_Hnd, CCI);
end;

end.
```

[Kostin Slava]

Высокоточный таймер

Решение

```
unit HRTimer;

interface

uses Windows;

type
  THRTimer = Class(TObject)
    constructor Create;
    function StartTimer: Boolean;
    function ReadTimer: Double;
  private
    StartTime: Double;
    ClockRate: Double;
  public
```

```
    Exists: Boolean;
  End;
implementation

constructor THRTimer.Create;
var
  QW: TLargeInteger;
begin
  Inherited Create;
  Exists := QueryPerformanceFrequency(QW);
  ClockRate := QW.QuadPart;
end;

function THRTimer.StartTimer: Boolean;
var
  QW: TLargeInteger;
begin
  Result := QueryPerformanceCounter(QW);
  StartTime := QW.QuadPart;
end;

function THRTimer.ReadTimer : Double;
var
  ET: TLargeInteger;
begin
  QueryPerformanceCounter(ET);
  Result := 1000.0 * (ET.QuadPart - StartTime)/ClockRate;
end;

end.
```

Примечание

Вызовы `xx.QuadPart` *необходимо изменить на* `TULargeInteger(xx).QuadPart`*, т. к. определения типов изменились со времени написания данного совета.*

Информация о DataLink

`TFieldDatalink` – производный класс от `TDataLink`, являющийся базовым классом для объекта, используемого компонентами для работы с базами данных и осуществляющий функцию связи с набором данных `DataSet` (`TTable` или `TQuery`). `DataLink` используется `DataSet` для информирования всех компонентов баз данных об изменении записи, о необходимости обновления записи перед помещением ее в базу данных, о том, что `DataSet` сменила свое состояние на активное или неактивное, и т. д. И наоборот, `DataLink` используется компонентами баз данных для обновления `DataSet`, например, его статуса.

`DataSet` может быть связан с несколькими источниками данных `DataSource`, каждый `DataSource` может быть связан с несколькими `DataLink`, и каждый `DataLink` может быть связан с единственным компонентом базы данных.

В большинстве случаев отдельный компонент использует только один DataLink, тем не менее, имеются компоненты, такие как, например, DBLookupList или DBLookupCombo, использующие два DataLink. В этих элементах управления первый DataLink предназначен для чтения данных из LookUp DataSet, второй DataLink – для записи этих данных (при их изменении) во второй имеющийся DataSet. Каждый DataSet поддерживает связанный с ним список DataSource и, аналогично этому, каждый DataSource поддерживает список связанных с ним DataLink.

В момент, когда DataSet должен уведомить компоненты базы данных о наступлении какого-то события, например, при изменении пользователем какой-либо записи, он рассылает это сообщение всем DataSource, находящимся в его списке. Каждый DataSource затем повторяет этот процесс и рассылает сообщение всем DataLink, находящимся в его списке. Другими словами, связь не зависит от элемента управления, при этом логика программирования должна отслеживать передачу сообщения каждому элементу управления, пользующемуся услугами DataLink, и изолировать только те события, на которые элементу необходимо отреагировать. Связывая компонент с набором данных другим способом, мы не получим в свое распоряжение столько управляющих функций, гибкости и мониторинга, сколько даст нам один DataSet, соединенный с помощью DataLink.

Кроме функции поддержки коммуникационного канала между DataSet и компонентами базы данных, DataLink также обеспечивает управление буфером для каждого компонента. Большинству элементов управления, таких как, например, TDBEdit, отображающий только отдельно взятую запись, буферизация не требуется, тем не менее, таким компонентам, как, например, TDBGrid и TDBLookupList, отображающим множество записей, буферизация нужна. Физически DataLink данные не буферизует, эта функция выполняется DataSet. Вместо этого DataLink поддерживает виртуальный буфер, который, в сущности, представляет собой небольшое «окно» в физический буфер DataSet. Размер этого виртуального буфера может быть установлен с помощью свойства DataLinks BufferCount, а количество записей, реально в нем хранимых, с помощью свойства RecordCount.

[News Group]

11

Интернет

UUE-кодирование

Как выполнить UUE-кодирование?

Решение

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Button1: TButton;
    Edit2: TEdit;
    procedure Edit1DblClick(Sender: TObject);
    procedure Edit2DblClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}
```

```
const
// эта константа используется при выделении памяти
  ssz = (High(Cardinal) - $F) div SizeOf(byte);

// эта константа используется функцией toanysys
  p: string = '0123456789ABCDEF';

// выбор входного файла
procedure TForm1.Edit1DblClick(Sender: TObject);
begin
  if OpenDialog1.Execute then Edit1.Text := OpenDialog1.Filename;
end;

// выбор выходного (UUE) файла
procedure TForm1.Edit2DblClick(Sender: TObject);
begin
  if SaveDialog1.Execute then Edit2.Text := SaveDialog1.Filename;
end;

// выделение подстроки
function mid(s: string; fromc, toc: byte): string;
var
  s1: string;
  i: byte;
begin
  s1 := '';
  for i := fromc to toc do
    s1 := s1 + s[i];
  mid := s1;
end;

// перевод числа (a) из десятичной системы в другую с основанием (r)
function ToAnySys(a, r: byte): string;
var
  s, k: string;
  n, m, i: byte;
begin
  s := '';
  m := 1;
  while m <> 0 do begin
    m := a div r;
    n := a - m * r + 1;
    k := p[n];
    s := k + s;
    a := m;
  end;
// добавляет незначащие нули
  for i := 1 to 8 - Length(s) do
    s := '0' + s;
  ToAnySys := s;
end;
```

```
// перевод 6-разрядного числа из двоичной системы в десятичную. Двоичное число
// подставляется в виде строки символов
function FromBin(s: string): byte;
var
  i, e, b: byte;
begin
  b := 0;
  for i := 1 to 6 do begin
    e := 1 shl (6 - i);
    if s[i] = '1' then b := b + e;
  end;
  FromBin := b;
end;

// непосредственно кодирование
type
  Tcoola = array [1..1] of byte;
  Pcoola = ^Tcoola;

procedure TForm1.Button1Click(Sender: TObject);
var
  inf: file of byte;
  ouf: textfile;
  uue: pcoola;
  b: array[1..4] of byte;
  bin, t: string;
  szf, oum, szl, szh, sxl, sxh, i, j: longint;
begin
{$I-}
  AssignFile(inf, Edit1.Text);        // входной файл
  Reset(inf);
  szf := FileSize(inf);
  szh := (szf * 8) div 6;
  if szf * 8 - szh * 6 = 0 then szl := 0 else szl := 1;
  GetMem(uue, szh+szl);               // выделение памяти
  oum := 1;
  while not(EoF(inf)) do begin
    b[1] := 0;
    b[2] := 0;
    b[3] := 0;
    b[4] := 0;
// чтение должно быть сделано посложнее, дабы избежать "read beyond end of file"
    Read(inf, b[1], b[2], b[3]);
// читаем 3 байта из входного файла и формируем "двоичную" строку
    bin := ToAnySys(b[1], 2) + ToAnySys(b[2], 2) + ToAnySys(b[3], 2);
// разбиваем строку на куски по 6 бит и добавляем 32
    t := Mid(bin, 19, 24);
    b[4] := FromBin(t) + 32;
    t := Mid(bin, 13, 18);
    b[3] := FromBin(t) + 32;
    t := Mid(bin, 07, 12);
```

```
    b[2] := FromBin(t) + 32;
    t := Mid(bin, 01, 06);
    b[1] := FromBin(t) + 32;
// записываем полученные байты во временный массив
    uue[oum] := b[1];
    oum := oum + 1;
    uue[oum] := b[2];
    oum := oum + 1;
    uue[oum] := b[3];
    oum := oum + 1;
    uue[oum] := b[4];
    oum := oum + 1;
  end;
  CloseFile(inf);                  // входной файл больше не нужен - закрываем его
// формируем выходной файл
  AssignFile(ouf, Edit2.Text);
  Rewrite(ouf);
  oum := 1;
  sxh := (szh + szl) div 60;               // число строк в UUE-файле
  sxl := (szh + szl) - sxh * 60;
// заголовок UUE-файла
  Writeln(ouf, 'begin 644 ' + ExtractFilename(Edit1.Text));
// записываем строки в файл
  for i :=1 to sxh do begin
    Write(ouf, 'M');
// 'M' значит, что в строке 60 символов
    for j := 1 to 60 do begin
      Write(ouf, Chr(uue[oum]));
      oum := oum + 1;
    end;
    Writeln(ouf);
  end;
// записываем последнюю строку, которая обычно короче 60 символов
  sxh := (sxl * 6) div 8;
  Write(ouf, Chr(sxh + 32));
  for i := 1 to sxl do begin
    Write(ouf, Chr(uue[oum]));
    oum := oum + 1;
  end;
// "добиваем" строку незначащими символами
  for i := sxl + 1 to 60 do Write(ouf, ```);
// записываем последние строки файла
  Writeln(ouf);
  Writeln(ouf, ```);
  Writeln(ouf, 'end');
  Closefile(ouf);
  FreeMem(uue, szh + szl);                  // освобождаем память
  ShowMessage('DONE.');                     // Готово
end;
end.
```

[Dubarev Sergei]

Проблемы ISAPI в Delphi 3

Почему ISAPI-ориентированные библиотеки, созданные в Delphi 3, не могут обрабатывать несколько соединений?

Wizard по созданию ISAPI DLL в Delphi 3 создает полностью безопасную многопоточную библиотеку, но не выставляет флаг, говорящий приложению, что данная библиотека в этом отношении безопасна. Это легко исправить, просто добавив строчку:

```
IsMultiThread := true;
```

[Nomadic]

Dialer

Как из программы выполнить соединение с провайдером?

Решение

(Модуль Ras.pas можно найти на http://www.torry.ru):

```
// connection - имя учетной записи
function DialProvider(connection: string): boolean;
var
  pars: TRasDialParams;
  hRas: ThrasConn;
  r: integer;
begin
  hRas := 0;
  StrPCopy(pars.szEntryName, connection);          // имя учетной записи
  pars.szPhoneNumber := '';                        // номер телефона - по умолчанию
  pars.szcallbacknumber := '';                     // callback нам не нужен
  pars.szUserName := '';                           // логин - по умолчанию
  pars.szPassWord := '';                           // пароль - по умолчанию
  pars.szDomain := '';                             // аналогично с доменом
  pars.dwSize := SizeOf(TRasDialParams);           // вычисляем размер записи
  r := RasDial(nil, nil, pars, 0, nil, hRas);      // звоним
  if r <> 0 then begin                             // если что-то не получилось, то
    rasHangUp(hRas);                               // сбрасываем соединение
    result := false;                               // функция теперь вернет false
  end else
    result := true;                                // а если все ок, то true.
end;
```

[Nomadic]

Проверка URL

Как выполнить проверку URL?

Данная функция позволяет проверить существование определенного адреса (URL) в Интернете. Естественно, она может пригодиться вебмастерам, у кото-

рых на сайте много ссылок и которым необходимо с определенной периодичностью эти ссылки проверять.

URL может быть как с префиксом 'http://', так и без него – эта функция добавляет префикс 'http://', если он отсутствует (необходимо для функции internetOpenUrl, которая также поддерживает 'FTP://' и 'gopher://').

Эта функция проверяет только два возвращаемых кода – '200' (OK) и '302' (перенаправление), но можно заставить ее проверять и другие коды. Для этого достаточно модифицировать строчку "result := ...".

```
uses
  WinInet;

function CheckUrl(url: string): boolean;
var
  hSession, hfile, hRequest: hInternet;
  dwindex, dwcodelen :dword;
  dwcode: array[1..20] of char;
  res: pchar;
begin
  if Pos('http://', LowerCase(url)) = 0 then url := 'http://' + url;
  Result := false;
  hSession := InternetOpen('InetURL:/1.0', INTERNET_OPEN_TYPE_PRECONFIG,
                           nil, nil, 0);
  if Assigned(hSession) then begin
    hfile := InternetOpenUrl(hSession, PChar(url), nil, 0,
                             INTERNET_FLAG_RELOAD, 0);
    dwIndex := 0;
    dwCodeLen := 10;
    HttpQueryInfo(hfile, HTTP_QUERY_STATUS_CODE, @dwcode, dwcodeLen, dwIndex);
    res := PChar(@dwcode);
    result := (res = '200') or (res = '302');
    if Assigned(hfile) then InternetCloseHandle(hfile);
    InternetCloseHandle(hSession);
  end;
end;
```

[The_Sprite]

Проверка соединения с провайдером

Как определить, что компьютер подключен к Интернету?

Решение

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if TCP1.LocalIp = '0.0.0.0' then ShowMessage('Компьютер не подключен!');
end;
```

TCLIENTSOCKET и TSERVERSOCKET

Почему сокету невозможно передать более чем 8 Кбайт данных?

Поскольку IP-слой «режет» поток данных на куски размером 8 Кбайт, разработчик должен явно включить в начало потока целочисленную длину, сообщающую на приемную сторону размер ожидаемых данных. Для нее очень важным является количество пакетов по 8 Кбайт, которые необходимо принять.

Поскольку Socket-компоненты являются простыми обертками для WinSock, а не протоколами с информационными заголовками (как было указано выше), разработчик должен сам побеспокоиться о «заполнении» их данными. Для этого необходимо:

- заполнить источник пакетами с целочисленной длиной;
- обеспечить получение целевым хостом пакетов с целочисленной длиной;
- подсчитать количество принятых байт и сопоставить с длиной пакета;
- проанализировать выражение `TotalBytesReceived <> LengthInteger`, и если это неравенство выполняется, то следующий пакет является продолжением, в противном случае далее в потоке ожидать очередной пакет.

Пакеты обрабатываются протоколами HTTP и FTP на основании информации заголовка (например, `'Content\Length: '`).

Примечание

Реализация может быть изменена разработчиком, поэтому приведены общие рекомендации.

Работа с cookies

Цель данного совета не в том, чтобы объяснить, что такое cookies и как их использовать. Предполагается, что это вы уже знаете, а хотите узнать, как с cookies работать из приложения, создаваемого для веб-сервера на Delphi 3.

Объекты `TWebRequest` и `TWebResponse`, поставляемые с Delphi 3, имеют свойства, обеспечивающие работу с cookies. `TWebRequest` имеет свойства `Cookie` и `CookieFields`, позволяющие приложению веб-сервера читать заголовки cookies, посылаемых как часть HTTP-запроса. Объект `TWebResponse` имеет свойство `Cookies`, благодаря которому приложение веб-сервера может размещать cookies на клиентской машине. Они передаются в заголовке как часть HTTP-запроса. Обычно это устанавливается с помощью метода `SetCookieField`.

Отвечая на HTTP-запрос, сервер посылает документ с заголовком и «содержательной» секцией. Delphi обеспечивает возможность добавлять заголовок cookie через свойство `TWebResponse.Cookies`. Но лучшим способом является применение метода `SetCookieField`. Приведенный ниже `TWebActionItem` демонст-

рирует технику использования метода SetCookieField для возвращения cookie, требующегося броузеру. В данном примере в качестве домена выступает localhost. Для работы в реальных условиях замените эту строку строкой с именем вашего домена. Третий параметр – косая черта. Это означает, что cookie будет посылаться из броузера со всеми запросами, относящимися к данному домену. Полное описание данных параметров можно обнаружить в документе RFC 2109.

```
procedure TWebModule1.WebModule1WebActionItem1Action(Sender: TObject;
               Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
var
  MyCookies: TStringList;
begin
  MyCookies := TStringList.Create;
  with MyCookies do begin
    Add('Name=Frank Borland');
    Add('Address=100 Borland Way');
    Add('Product=Delphi');
  end;
  with Response do begin
    SetCookieField(MyCookies, 'localhost', '/', (Now + 1), False);
    Content := 'cookie вставлен';
  end;
end;
```

При выполнении запроса, когда HTTP-сервер запрашивает броузер клиента, имена всех cookies пакуются и включаются в заголовок в ответ на запрос HTTP. В Delphi 3 это можно сделать для приложения веб-сервера двумя путями. Первый способ – включить имя как строку через свойство Cookie параметра TWebRequest у TWebActionItem. Второй основан на том, что имя доступно как свойство TStrings с именем CookieFields. CookieFields – разобранное содержимое заголовка cookie из сообщения HTTP-запроса.

В следующих примерах извлекается имя значения и возвращается клиенту в виде пустой страницы HTML. В первом из них cookie возвращается в виде единственной строки, во втором каждая пара имя-значение возвращается в отдельной строке.

```
procedure TWebModule1.WebModule1WebActionItem2Action(Sender: TObject;
               Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
begin
  Response.Content := '' + Request.Cookie + '';
end;

procedure TWebModule1.WebModule1WebActionItem3Action(Sender: TObject;
               Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
var
  i: integer;
begin
  Response.Content := '';
  for i := 0 to (Request.CookieFields.Count - 1) do begin
```

```
      Response.Content := Response.Content + ' ' + Request.CookieFields[i] + ' ';
    end;
    Response.Content := Response.Content + '';
  end;
```

Объект DocInput

DocInput – объект из пакета Internet Solutions Pack фирмы NetManage, поставляемый с Delphi 2.01. Он описывает входную информацию для документа, передаваемого элементу управления. Все элементы управления для работы с Интернетом, содержащиеся в данном пакете, имеют доступ к объекту через соответствующее свойство, могут хранить в нем документы и передавать их от одного элемента управления другому. Объект DocInput имеет следующие свойства: BytesTotal, BytesTransferred, DocLink, FileName, Headers, PushStreamMode, State и Suspended.

BytesTotal – счетчик общего количества байт передаваемого элемента. Значение по умолчанию и начальное значение равны нулю. Тип данных – long. Это свойство времени выполнения и доступно только для чтения. Значение данного свойства получается из свойства заголовка «content-length» (длина содержимого). По этому значению элемент управления определяет размер (объема) передаваемой информации. С его помощью также возможно управление буфером, который вы используете для «сборки» данных после их передачи.

BytesTranferred представляет собой свойство, передаваемое приложению при наступлении события OnDocInput. Это свойство времени выполнения, доступно только для чтения и имеет тип long. При начале новой передачи значение свойства обнуляется. Обновляется в начале события OnDocInput. Значение данного свойства отражает величину последней передачи, когда другие передачи не осуществлялись. Свойство BytesTransferred может использоваться для показа индикатора хода выполнения или для проверки соответствия фактически переданного размера ожидаемому.

DocLink сообщает получающему элементу управления о том, что источник не будет посылать документ через поток данных или входной файл. Оно ссылается на свойство DocOutput.DocLink, которое становится источником при передаче данных. Это свойство для чтения и записи (read/write) и доступно только во время выполнения программы. Свойство DocLink представляет собой строковый тип, по умолчанию имеющий значение ''. Если данному свойству присваивается значение, отличное от '', свойство FileName автоматически устанавливается в ''. Данное свойство используется для определения источника, являющегося интернет-компонентом с указывающим на объект свойством DocOutput.DocLink, т. е. они используются попарно.

FileName является свойством для чтения и записи (read/write), доступно только во время выполнения и имеет строковый тип. Значение должно быть правильным именем файла и по умолчанию равно ''. Данное свойство может быть установлено при его передаче в качестве аргумента объекту DocInput.

Если значению данного свойства присваивается величина, отличная от '', свойство DocLink автоматически устанавливается в ''.

Свойство Headers доступно во время выполнения и предназначено только для чтения. Headers – коллекция элементов DocHeader, которые определяют передаваемый документ. Содержимое свойства Headers должно быть изменено перед вызовом метода GetDoc. Каждый DocHeader представляет собой MultiPurpose Internet Mail Extension (MIME). MIME является механизмом для определения и описания формата тела сообщения Интернет (Internet Message Bodies). (Для получения дополнительной информации см. документ RFC 1341.) Заголовки (headers) зависят от используемого протокола, но существуют два заголовка, которые от протокола не зависят:

- content-type (тип содержимого) указывает спецификацию MIME для следующего за заголовком документа. Примером является «text/plain»;
- content-length (размер содержимого) указывает размер документа в байтах.

State является свойством времени выполнения, доступно только для чтения и имеет перечислимый тип DocStateConstants. Значение по умолчанию – icDocNone. Свойство State элемента управления обновляется каждый раз при наступлении события DocInput.

Свойство Suspended доступно только для чтения во время выполнения и имеет логический тип. Устанавливается вызовом метода Suspend. При установке значения, равного True, передача приостанавливается.

Свойство для чтения и записи (read/write) PushStream доступно только во время выполнения и имеет логический тип. Значение по умолчанию – False. Если свойству FileName или DocLink присваивается значение, отличное от '', то свойство PushStream становится недоступным.

Объект DocInput имеет четыре метода: GetData, PushStream, SetData и Suspend.

GetData сообщает объекту DocInput об извлечении текущего блока данных в момент наступления события DocOutput. Данный метод может быть вызван только в течение события OnDocInput и только когда свойство State установлено в icDocData(3). При использовании свойства FileName или DocLink данный метод позволяет исследовать данные во время их передачи.

PushStream может быть вызван, только если PushStreamMode установлен в True и когда данные доступны. PushStream устанавливает свойство State на основе следующего шага передачи документа и активизирует в нужный момент событие DocInput. Затем происходит возврат до следующего вызова PushStream. Перед PushStream должен быть вызван метод SetData.

SetData определяет следующий буфер передаваемых данных при наступлении события DocInput. Вызывается в течение события DocInput или перед SendDoc. Если метод используется перед вызовом SendDoc, он может служить альтернативой передаче параметров InputData в InputData. Тип должен быть определен как variant.

Suspend передает форме команду suspend(true) или suspend(false). Если метод с параметром True был вызван дважды, то для продолжения передачи его необходимо дважды вызвать с параметром False.

Код примера показывает, как можно использовать объект DocInput. Полный проект, содержащий данный код, можно найти в подкаталоге \demos на CD-ROM с Delphi 2.01. Данный проект с именем SimpMail.dpr представляет собой хороший пример работы со свойством объекта Headers. Также показано соответствующее использование события DocInput и свойства State.

```
{ Очистка и новое заполнение заголовков MIME с помощью свойства компонента
  DocInput.
  Может также использоваться отдельный OLE-объект DocInput. Для получения полной
  информации о типах MIME см. документ RFC1521/1522. }
procedure TMainForm.CreateHeaders;
begin
  with SMTP1 do begin
    DocInput.Headers.Clear;
    DocInput.Headers.Add('To', eTo.Text);
    DocInput.Headers.Add('From', eHomeAddr.Text);
    DocInput.Headers.Add('CC', eCC.Text);
    DocInput.Headers.Add('Subject', eSubject.Text);
    DocInput.Headers.Add('Message-Id', Format('%s_%s_%s', [Application.Title,
                         DateTimeToStr(Now), eHomeAddr.Text]));
    DocInput.Headers.Add('Content-Type', 'TEXT/PLAIN charset=US-ASCII');
  end;
end;

{ Посылаем простое почтовое сообщение }
procedure TMainForm.SendMessage;
begin
  CreateHeaders;
  with SMTP1 do
    SendDoc(NoParam, DocInput.Headers, reMessageText.Text, '', '');
end;

{ Посылаем файл, расположенный на диске. Оставляем пустым параметр SendDoc
  InputData и определяем имя файла для InputFile для посылки содержимого файла,
  расположенного на диске. Для осуществления собственного кодирования (Base64,
  UUEncode и др.) вы можете использовать событие DocInput и методы GetData }
procedure TMainForm.SendFile(Filename: string);
begin
  CreateHeaders;
  with SMTP1 do begin
    DocInput.Filename := FileName;
    SendDoc(NoParam, DocInput.Headers, NoParam, DocInput.FileName, '');
  end;
end;

{ Событие DocInput возникает при каждом изменении состояния DocInput во время
  передачи почтового сообщения. DocInput хранит всю информацию о текущей передаче,
```

```
    включая заголовки, количество переданных байт и сами данные сообщения. Хотя в этом
    примере и не показано, но перед отправкой каждого блока для кодирования данных
    вы можете вызвать метод DocInput SetData, если DocInput.State = icDocData. }

  procedure TMainForm.SMTP1DocInput(Sender: TObject; const DocInput: Variant);
  begin
    case DocInput.State of
      icDocBegin:    SMTPStatus.SimpleText := 'Начало передачи документа';
      icDocHeaders:  SMTPStatus.SimpleText := 'Посылаем заголовки';
      icDocData:     if DocInput.BytesTotal > 0 then
                       SMTPStatus.SimpleText :=
                          Format('Послано данных: %d из %d байт (%d%%)',
                                 [Trunc(DocInput.BytesTransferred),
                                  Trunc(DocInput.BytesTotal),
                                  Trunc(DocInput.BytesTransferred /
                                  DocInput.BytesTotal * 100)])
                     else
                       SMTPStatus.SimpleText := 'Посылка...';
      icDocEnd:      if SMTPError then
                       SMTPStatus.SimpleText := 'Передача прервана'
                     else
                       SMTPStatus.SimpleText :=
                          Format('Почта послана %s (%d байт данных)',
                          [eTo.Text, Trunc(DocInput.BytesTransferred)]);
    end;
    SMTPStatus.Update;
  end;
```

Объект DocOutput

`DocOutput` – объект из пакета Internet Solutions Pack фирмы NetManage, поставляемого с Delphi 2.01. Он описывает выходную информацию передаваемого документа. Все элементы управления, имеющие свойство `DocOutput`, используют этот тип. Он также является объектом, на который указывает событие `DocOutput`. Объект `DocOutput` имеет следующие свойства: `BytesTotal`, `BytesTransferred`, `DocLink`, `FileName`, `Headers`, `PushStreamMode`, `State` и `Suspend`.

Описание свойств можно найти в предыдущем разделе, посвященном объекту `DocInput`.

Объект `DocOutput` имеет три метода: `GetData`, `SetData` и `Suspend`. Эти методы также были рассмотрены в предыдущем разделе.

Приведенный ниже код взят из демонстрационного проекта, расположенного в подкаталоге \demos\internet Delphi 2.01. Данный проект с именем `HTTPDemo.dpr` представляет собой пример использования свойств объекта `BytesTransferred` и `state`. Также показано применение различных типов данных, являющихся новыми для Delphi 2.01. Эти типы данных важны для работы с OLE, и пользователи Delphi должны о них узнать как можно скорее, если они хотят применять технологию OLE в своих приложениях.

```
procedure TForm1.HTTP1DocOutput(Sender: TObject; const DocOutput: Variant);
var
  S: String;
  i: integer;
  MsgNo, Header: String;
  Parser: TSimpleHTMLParser;
  ALine: String;
begin
  Statusbar1.Panels[2].Text := Format('Байт: %s', [DocOutput.BytesTransferred]);
  case DocOutput.State of
    icDocBegin:    begin
                     Memo1.Lines.Clear;
                     Data := '';
                   end;
    icDocData:     begin
                     DocOutput.GetData(S, VT_BSTR);
                     Data := Data + S;
                   end;
    icDocEnd:      begin
                   { Теперь удаляем все теги HTML и отображаем текст }
                     Parser := TSimpleHTMLParser.Create(Data);
                     ALine := '';
                     while Parser.FToken <> etEnd do begin
                       case Parser.FToken of
                         etHTMLTag:   begin
                                        if Parser.TokenHTMLTagIs('BR') then
                                          ALine := ALine + #13#10;
                                        if Parser.TokenHTMLTagIs('P') then
                                          ALine := ALine + #13#10#13#10;
                                      end;
                         etSymbol:
                                      ALine := ALine + ' ' + Parser.FTokenString;
                         etLineEnd:   begin
                                        Memo1.Lines.Add(ALine);
                                        ALine := '';
                                      end;
                       end;
                       Parser.NextToken;
                     end;
                     Memo1.Lines.Add(ALine);
                     Memo1.SelStart := 0;
                     SendMessage(Memo1.Handle, EM_ScrollCaret, 0, 0);
                   end;
  end;
  Refresh;
end;
```

Захват текущего URL в MS IE

Пример показывает, как найти окно Internet Explorer и захватить из него текущий URL, находящийся в поле адреса IE. В исходном листинге приведены простые функции API Win32 на Delphi.

```
function GetText(WindowHandle: hwnd): string;
var
  txtLength: integer;
  buffer: string;
begin
  txtLength := SendMessage(WindowHandle, WM_GETTEXTLENGTH, 0, 0);
  txtLength := txtLength + 1;
  SetLength(buffer, txtLength);
  SendMessage(WindowHandle, wm_gettext, txtLength, longint(@buffer[1]));
  result := buffer;
end;

function GetURL: string;
var
  ie, toolbar, combo, comboboxex, edit, worker, toolbarwindow: hwnd;
begin
  ie := FindWindow(pchar('IEFrame'), nil);
  worker := FindWindowEx(ie, 0, 'WorkerA', nil);
  toolbar := FindWindowEx(worker, 0, 'rebarwindow32', nil);
  comboboxex := FindWindowEx(toolbar, 0, 'comboboxex32', nil);
  combo := FindWindowEx(comboboxex, 0, 'ComboBox', nil);
  edit := FindWindowEx(combo, 0, 'Edit', nil);
  toolbarwindow := FindWindowEx(comboboxex, 0, 'toolbarwindow32', nil);
  result := GetText(edit);
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  ShowMessage(GetURL);
end;
```

[The_Sprite]

IP-адрес и имя хоста

Как получить IP-адрес или имя хоста на компьютере клиента?

Используйте WinSocks API, описанный в модуле WinSocks.pas.

```
uses
  WinSock;

procedure GetHostInfo(var Name, Address: string);
var
  WSAData: TWSAData;
```

```
  HostEnt: PHostEnt;
begin
{ no error checking...}
  WSAStartup(2, WSAData);
  SetLength(Name, 255);
  GetHostName(PChar(Name), 255);
  SetLength(Name, StrLen(PChar(Name)));
  HostEnt := GetHostByName(PChar(Name));
  with HostEnt^ do
    Address := Format('%d.%d.%d.%d', [Byte(h_addr^[0]), Byte(h_addr^[1]),
                      Byte(h_addr^[2]), Byte(h_addr^[3])]);
  WSACleanup;
end;
```

Кроме того, можно просто перенести TCP-компонент с вкладки Internet на форму и использовать его свойства.

```
Memo1.Lines.Add(TCP1.LocalHostName);
Memo1.Lines.Add(TCP1.LocalIp);
```

Обработка ошибок WinSock

Для работы любого из нижеследующих методов обработки исключительной ситуации необходимо, чтобы VCL знала о возникновении самой ошибки. Если вызов WinSock ничего не возвращает или не обеспечивает информацией потомка TCustomWinSocket, который его вызвал, то механизм выполнения данного условия не существует.

Обработчик исключения OnError

При организации обработки исключительных ситуаций в потомке TCustomWinSocket для обработки исключения OnError должен использоваться метод, который обрабатывает ограниченный набор условий, поскольку механизм, предусмотренный WinSock для событий-уведомителей, реагирует только на ограниченный список условий. Чтобы быть извещенным об исключительной ситуации, возникшей при работе WinSock, TCustomWinSocket регистрирует пользовательское сообщение CM_SocketMessage, посылаемое компоненту, а обработчик сообщения CMSocketMessage генерирует исключительную ситуацию. Сообщение регистрируется WinSock вызовом API WSASyncSelect. WSASyncSelect – запрос для уведомления о событиях чтения из сокета, записи, соединения, закрытия и приема. Если исключительная ситуация не вызвана событием чтения, записи, соединения, закрытия или приема либо CM_SocketMessage по любой причине не был послан WinSock, то обработчик события не сработает.

Использование:

```
procedure TChatForm.ClientSocketError(Sender: TObject; Socket: TCustomWinSocket;
                                      ErrorEvent: TErrorEvent; var ErrorCode: Integer);
const
  ErrorEvents: array[eeGeneral..eeAccept] of string = (
```

```
                'eeGeneral',
                'eeSend',
                'eeReceive',
                'eeConnect',
                'eeDisconnect',
                'eeAccept'
               );
begin
  ListBox1.Items.Add('ClientSocketError. TErrorEvent: ' + ErrorEvents[ErrorEvent] +
                     ' ErrorCode: ' + IntToStr(ErrorCode));
  ErrorCode := 0;     // исключение не генерируем
end;
```

Определение:

```
procedure TCustomWinSocket.CMSocketMessage(var Message: TCMSocketMessage);

  function CheckError: Boolean;
  var
    ErrorEvent: TErrorEvent;
    ErrorCode: Integer;
  begin
    if Message.SelectError <> 0 then begin
      Result := False;
      ErrorCode := Message.SelectError;
      case Message.SelectEvent of
        FD_CONNECT:    ErrorEvent := eeConnect;
        FD_CLOSE:      ErrorEvent := eeDisconnect;
        FD_READ:       ErrorEvent := eeReceive;
        FD_WRITE:      ErrorEvent := eeSend;
        FD_ACCEPT:     ErrorEvent := eeAccept;
      else
        ErrorEvent := eeGeneral;
      end;
      Error(Self, ErrorEvent, ErrorCode);
      if ErrorCode <> 0 then
        raise ESocketError.CreateFmt(sASyncSocketError, [ErrorCode]);
    end else
      Result := True;
  end;
begin
  with Message do
    if CheckError then
      case SelectEvent of
        FD_CONNECT:   Connect(Socket);
        FD_CLOSE:     Disconnect(Socket);
        FD_READ:      Read(Socket);
        FD_WRITE:     Write(Socket);
        FD_ACCEPT:    Accept(Socket);
      end;
end;
```

Обработка исключений средствами Object Pascal

Можно также поместить специфичный вызов в конструкцию try..except или создать свой обработчик исключительных ситуаций на уровне приложения. Чтобы это работало, компонент должен иметь возможность получить уведомление о возникновении исключительной ситуации, а чтобы исключение можно было перехватить, его необходимо сгенерировать.

Пример обработчика исключительных ситуаций на уровне приложения:

```
TChatForm = class(TForm)
  public

    procedure AppException(Sender: TObject; E: Exception);
end;

implementation

procedure TChatForm.AppException(Sender: TObject; E: Exception);
begin
  ListBox1.Items.Add('AppException: ' + E.Message);
end;

procedure TChatForm.FormCreate(Sender: TObject);
begin
  Application.OnException := AppException;
end;
```

Пример конструкции Try..Except:

```
with ClientSocket do
  try
    Active := True;
  except
    on E: Exception do
      ListBox1.Items.Add('В течение Try..except: ' + E.Message);
  end;
end;
```

SocketErrorProc

В случае вызовов, которые для проверки возвращаемого WSAGetLastError результата обычно используют функцию CheckSocketResult, ошибки могут быть обработаны программистом, определившим функцию, устанавливающую SocketErrorProc.

Применение:

```
interface

procedure MySocketError(ErrorCode: Integer);

implementation
```

```
procedure MySocketError(ErrorCode: Integer);
begin
  ShowMessage('MySocketError: ' + IntToStr(ErrorCode));
end;

procedure TChatForm.FormCreate(Sender: TObject);
begin
  SocketErrorProc := MySocketError;
end;
```

Определение:

```
function CheckSocketResult(ResultCode: Integer; const Op: string): Integer;
begin
  if ResultCode <> 0 then begin
    Result := WSAGetLastError;
    if Result <> WSAEWOULDBLOCK then
      if Assigned(SocketErrorProc) then
        SocketErrorProc(Result)
      else
        raise ESocketError.CreateFmt(sWindowsSocketError, [SysErrorMessage(Result),
                                     Result, Op]);
  end else
    Result := 0;
end;
```

Подсказка для SocketErrorProc:

```
unit
  ScktComp;
```

Процедура SocketErrorProc обрабатывает сообщения об ошибках, которые были получены от соединения с сокетом Windows.

```
threadvar SocketErrorProc: procedure (ErrorCode: Integer);
```

Описание:

Назначаем значение SocketErrorProc для обработки сообщения об ошибке от вызова Windows Sockets API, прежде чем компонент возбудит исключение. Установка SocketErrorProc предотвращает генерирование исключительной ситуации socket-компонентом. SocketErrorProc – локальная переменная потока. Это делает возможной обработку ошибки, возникающей при вызовах Windows Sockets API, осуществляемых в пределах единственного потока выполнения.

12

OLE

Получение данных из Program Manager через DDE

Установите соединение DDEClientConv с сервером, затем установите DDETopic в 'ProgMan' и для сервера, и для клиента. Вызовите RequestData и передайте 'Groups' как элемент (Item); обратно вы получите список имен групп. Вызовите RequestData с одним из имен групп, и вы получите детальную информацию о группе. Вероятно, дальше вы захотите передать полученные данные в ListBox, т. к. это позволяет сразу увидеть результат и понять, как его можно обработать, например:

```
  ...
var
  P: PChar;
  ...
  P := DdeClientConv1.RequestData('Groups');
  ListBox1.Items.SetText(P);
  StrDispose(P);
  ...
  with ListBox1 do
    P := DdeClientConv1.RequestData(Items[ItemIndex]);
  ListBox2.Items.SetText(P);
  StrDispose(P);
  ...
```

[News Group]

Управление Program Manager в Windows 95 с помощью DDE

Для управления программными группами в Program Manager с помощью DDE был использован следующий модуль:

```
unit Pm;

interface

uses
  SysUtils, Classes, DdeMan;

type
  EProgManError = class(Exception);

  TProgMan = class(TComponent)
  private
    FDdeClientConv: TDdeClientConv;
    procedure InitDDEConversation;
    function ExecMacroString(Macro: String): Boolean;
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    procedure CreateGroup(GroupName: String; ShowGroup: Boolean);
    procedure DeleteGroup(GroupName: String);
    procedure DeleteItem(ItemName: String);
    procedure AddItem(CmdLine, ItemName: String);
  end;

implementation

const
{ DDE макростроки для Program Manager }
  SDDECreateGroup = '[CreateGroup(%s)]';
  SDDEShowGroup = '[ShowGroup(%s, 1)]';
  SDDEDeleteGroup = '[DeleteGroup(%s)]';
  SDDEDeleteItem = '[DeleteItem(%s)]';
  SDDEAddItem = '[AddItem(%s, "%s", %s)]';

constructor TProgMan.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  InitDDEConversation;
end;

destructor TProgMan.Destroy;
begin
  if Assigned(FDDEClientConv) then FDdeClientConv.CloseLink;
  inherited Destroy;
end;
```

```
function TProgMan.ExecMacroString(Macro: String): Boolean;
begin
  Result := FDdeClientConv.ExecuteMacro(PChar(Macro), False);
end;

procedure TProgMan.InitDDEConversation;
begin
  FDdeClientConv := TDdeClientConv.Create(Self);
  if not FDdeClientConv.SetLink('PROGMAN', 'PROGMAN') then
    raise EProgManError.Create('Не могу установить DDE Link');
end;

procedure TProgMan.CreateGroup(GroupName: String; ShowGroup:Boolean);
begin
{ Удаляем группу, если она существует }
  ExecMacroString(Format(SDDEDeleteGroup, [GroupName]));
  if not ExecMacroString(Format(SDDECreateGroup, [GroupName])) then
    raise EProgManError.Create('Не могу создать группу ' + GroupName);
  if ShowGroup then
    if not ExecMacroString(Format(SDDEShowGroup, [GroupName])) then
      raise EProgManError.Create('Не могу показать группу ' + GroupName);
end;

procedure TProgMan.DeleteGroup(GroupName: String);
begin
  if not ExecMacroString(Format(SDDEDeleteGroup, [GroupName])) then
    raise EProgManError.Create('Не могу удалить группу ' + GroupName);
end;

procedure TProgMan.DeleteItem(ItemName: String);
begin
  if not ExecMacroString(Format(SDDEDeleteGroup, [ItemName])) then
    raise EProgManError.Create('Не могу удалить элемент ' + ItemName);
end;

Procedure TProgMan.AddItem(CmdLine, ItemName: String);
var
  P: PChar;
  PSize: Word;
begin
  PSize := StrLen(SDDEAddItem) + (Length(CmdLine) * 2) + Length(ItemName) + 1;
  GetMem(P, PSize);
  try
    StrFmt(P, SDDEAddItem, [CmdLine, ItemName, CmdLine]);
    if not FDdeClientConv.ExecuteMacro(P, False) then
      raise EProgManError.Create('Не могу добавить элемент ' + ItemName);
  finally
    FreeMem(P, PSize);
  end;
end;
end.
```

Добавление группы в Program Manager

Как добавить группу в Program Manager?

Решение

```
interface

procedure CreateGroup;

implementation

{ Для установки группы в Program Manager используем компонент TProgMan }
procedure TSetupForm.CreateGroup;
var
  ItemList: TStringList;
  GroupName: String;
  ItemName: String;
  i: word;
begin
{ Получаем из INI-файла строку GroupName }
  GroupName := IniFile.ReadString('General', 'PMGroup', '');
{ Если один есть, устанавливаем группу }
  if GroupName <> '' then begin
    ItemList := TStringList.Create;

    try
{ читаем элементы для установки }
      IniFile.ReadSectionValues('PMGroup', ItemList);
      with TProgMan.Create(Self) do
        try
          CreateGroup(GroupName);
          for i := 0 to ItemList.Count - 1 do begin
{ получаем имя файла }
            ItemName := Copy(ItemList.Strings[i], 1,
                             Pos('=', ItemList.Strings[i]) - 1);
{ прибавляем путь к имени файла и добавляем элемент }
            AddItem(GetTarget(ItemList.Values[ItemName][1]) + ItemName, ItemName);
          end;
        finally
          Free;
        end;
    finally
      ItemList.Free;
    end;
  end;
end;
```

Примечание

Этот совет, как и предыдущие, характерен для Delphi 1 и Windows 3.1x.

DDE – передача текста

Пример для работы с Excel:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  P: PChar;
begin
  P := 'Пример передачи текста';
  DDEClientConv1.SetLink('Excel', 'Книга1');
  try
    DDEClientConv1.OpenLink;
    DDEClientItem1.DDEItem:= 'R1C1';
    DDEClientConv1.PokeData(DDEClientItem1.DDEItem, P);
  finally
    DDEClientConv1.CloseLink;
  end;
end;
```

Как видите, свойство `DDEItem` определяется сервером. Если ваш сервер представляет собой приложение Delphi, то `DDEItem` – имя `DDEServerItem`. Возможно, не стоит связываться с отладкой DDE-программ. Все же лучшим способом является синхронизация, позволяющая понять при отладке правильность действий.

Примечание

Напомним, что для работы с этим примером необходимы компоненты `DDEClientConv` *и* `DDEClientItem` *на форме и* `DDEClientItem1.DDEConv = DDEClientConv1`. *Предварительно запустите Excel.*

COM

COM (Component Object Model, модель компонентных объектов) – это стандарт, описывающий, как должны работать интерфейсы класса (или объекта), включая такие вопросы, как, например, работа с памятью или многопоточностью, и каким образом приложения могут использовать компоненты, созданные в стандарте COM. COM представляет собой стандарт, независимый от языка программирования и аппаратного окружения. Первоначально стандарт был разработан, принят и опубликован фирмами Microsoft и IBM, но пока не существует технической причины для того, чтобы он не поддерживался, например, и на Sun Sparc20 под управлением операционной системы Solaris.

Отличительные признаки и основные механизмы функционирования COM-объекта:

- У объекта имеется глобальный уникальный идентификатор (globally unique) – сокращенно GUID или CLSID (Class Identifier) – 128-битное целое число, которое, по существу, должно гарантировать уникаль-

ность COM-объекта (или интерфейса в случае CLSID) на всех компьютерах нашей планеты. Для того чтобы воспользоваться COM-объектом, клиенту необходимо знать его GUID.

- Если клиенту известен GUID COM-объекта, то для создания его экземпляра можно воспользоваться стандартным API-вызовом `CoCreateInstance`.
- Объект содержит по крайней мере один интерфейс с именем `IUnknown`. `IUnknown` имеет три метода: `AddRef`, `Release` и `QueryInterface`. Первые два вызываются клиентами для управления счетчиком ссылок, в котором хранится количество используемых ссылок на объект. Как только счетчик ссылок станет равным нулю, объект будет удален из памяти.

Клиент может вызвать `QueryInterface`, для того чтобы определить, поддерживает ли объект специфический интерфейс (интерфейс – группа свойств и методов). Если это так, `QueryInterface` возвращает ссылку на таблицу указателей свойств и методов, поддерживаемых интерфейсом. После этого клиент может вызывать эти методы, используя указатели, получаемые из таблицы.

OLE – «развивающийся стандарт» (который, вероятно, в нашем контексте можно назвать «перемещающий цели»), созданный для предоставления комплекса услуг (компонентам, приложениям и др.) с применением COM-ориентированных объектов.

Автоматизация OLE является отдельным OLE-сервисом, предоставляющим клиенту возможность управления отдельными компонентами. Действительно, это несколько непривычно. Компонент, которым можно управлять с помощью OLE-автоматизации, называется `IDispatch`. `IDispatch` включает методы для определения поддерживаемых интерфейсом методов (извините за каламбур), их имен, типов данных, описание всех параметров и специальный `"dispatch ID"` для каждого метода интерфейса. Если клиент знает имя метода, то с помощью `IDispatch` он может узнать о нем все. Само собой, это займет какое-то время, но это работает. Конечно лучше, если заранее известны (например, во время компиляции) `dispID` и информация о параметрах вызываемого метода для его непосредственного вызова во время выполнения программы посредством метода `IDispatch Invoke`.

Для вызовов всех методов объекта программа использует интерфейс объектов `IDispatch` (и/или библиотеку типов) для определения `dispID` и получения информации о параметрах и при компиляции вставляет эту информацию в вашу программу. Это так называемое «раннее связывание», позволяющее осуществлять максимально быстрые вызовы методов COM-объектов в случае применения автоматизации OLE. Можно также воспользоваться «поздним связыванием», когда программа для определения `dispID`, имени метода и т. д. обращается к `IDispatch` только во время ее выполнения. Это случается при использовании объектной переменной, объявленной «как объект» («As Object»). Позднее связывание достаточно медленное и отчасти ненадежное (поскольку во время компиляции программа не может осуществить проверку параметров), но может иногда применяться ввиду своей гибкости.

OLE–тестер

В данном примере в момент создания формы создается OLE-объект, а после нажатия какой-либо из кнопок вызывается определенная процедура OLE-сервера.

```
unit oletestu;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  public
    ttsesed: variant;
  end;

var
  Form1: TForm1;

implementation
uses ComObj;
{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
  ttsesed := CreateOleObject('ttdewed.ttsesole');
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  ttsesed.OpenEditFile;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  ttsesed.AppShow;
end;

end.
```

Чтение сложных OLE-документов

Не используйте для этого компонент TOleContainer. При создании приложения с возможностью доступа к файлам OLE Structured Storage (структурное хранилище) реально необходима только пара вызовов OLE API и 5–6 вызовов API для доступа к функциям-членам интерфейсов IStorage, IRootStorage и IStream. Есть подозрение, что функция-член TOleContainer IStorage является указателем на корневое хранилище файла Structured Storage, создаваемого функциями-членами ToleContainer LoadFromFile/SaveToFile.

Необходимые функции API:

```
StgOpenStorage
StgIsStorageFile
```

Определения можно найти в OLE 2.01 SDK.

Будьте внимательны, при переносе заголовочных файлов C++ из OLE SDK в Delphi Borland допустил несколько ошибок. Вот правильное определение IStorage:

```
MyIStorage = class(IUnknown)
  function CreateStream(const pwcsName: PChar; grfMode: Longint;
        reserved1: Longint; reserved2: Longint; var ppstm: IStream): HResult;
        virtual; cdecl; export; abstract;

function OpenStream(const pwcsName: PChar; reserved1: Pointer; grfMode: Longint;
         reserved2: Longint; var ppstm: IStream): HResult; virtual; cdecl;
         export; abstract;

function CreateStorage(const pwcsName: PChar; grfMode: Longint; reserved1: Longint;
                       reserved2: Longint; var ppstg: MyIStorage): HResult;
                       virtual; cdecl; export; abstract;

function OpenStorage(const pwcsName: PChar; pstgPriority: MyIStorage;
                     grfMode: Longint; snbExclude: PStr; reserved: Longint;
                     var ppstg: MyIStorage): HResult; virtual; cdecl;
                     export; abstract;

function CopyTo(ciidExclude: Longint; const rgiidExclude: IID;
                var snbExclude: PStr; pstgDest: MyIStorage): HResult; virtual;
                cdecl; export; abstract;

function MoveElementTo(const lpszName: PChar; pstgDest: MyIStorage;
                       const lpszNewName: PChar; grfFlags: Longint): HResult;
                       virtual; cdecl; export; abstract;

function Commit(grfCommitFlags: Longint): HResult; virtual;
                cdecl; export; abstract;
function Revert: HResult; virtual; cdecl; export; abstract;
function EnumElements(reserved1: Longint; reserved2: Pointer; reserved3: Longint;
                      var ppenm: IEnumStatStg): HResult; virtual; cdecl;
                      export; abstract;
```

```
function DestroyElement(const pwcsName: PChar): HResult; virtual; cdecl;
                        export; abstract;

function RenameElement(const pwcsOldName: PChar;
                       const pwcsNewName: PChar): HResult;
                       virtual; cdecl; export; abstract;

function SetElementTimes(const lpszName: PChar; const pctime: TFileTime;
                         const patime: TFileTime; const pmtime: TFileTime): HResult;
                         virtual; cdecl; export; abstract;

function SetClass(const clsid: IID): HResult; virtual; cdecl; export; abstract;
function SetStateBits(grfStateBits: Longint; grfMask: Longint): HResult;
                      virtual; cdecl; export; abstract;
function Stat(var pstatstg: TStatStg; grfStatFlag: Longint): HResult;
              virtual; cdecl; export; abstract;
end;
```

OLE-сервер

Следующий код не так ясен и понятен, но он может оказаться полезным:

```
unit Unit1;

interface

function OLEfunction(x, y, z: integer): integer; cdecl; export;

implementation

function OLEfunction(x, y, z: integer): integer;
begin
end;

procedure buildOLEstructure;
var
  F: pointer;
begin
  F := @OLEfunction;    { Компилируется без проблем ... }
end;

end.
```

Используйте метод, приведенный ниже. Необходимо объявить одну вызывающую функцию для каждой комбинации параметров, которые вы собираетесь передавать. Затем можно в качестве указателя передать функцию, которую требуется вызвать, вызывающей функции. Поясню на примере:

```
library Pcdecl;

function olefunction(a1: pchar; a2: longint; x: integer): integer; cdecl; export;
```

```
begin
end;

function callolefunction(func: pointer; a1: pchar; a2: longint;
                         x: integer): integer; assembler;
asm
  push  x                 // помещаем параметры в обратном порядке
  push  word ptr a2 + 2   // если 32-битная величина передается в этих двух шагах,
                          // то начинаем с самой "высокой" (high) части
  push  word ptr a2
  push  word ptr a1 + 2
  push  word ptr a1
  call  func
  add   sp, 10            // восстанавливаем стек добавлением вытолкнутых байтов.
                          // Обратите внимание на то, что func не была вытолкнута
end;

procedure buildolefunction;
var
  f: pointer;
  reslt: integer;

begin
  f := @olefunction;
{ ... }
  reslt := callolefunction(f, 'Здравствуй, мир', 1000000, 25);
{ ... }
end;

begin
{ ... }
end.
```

Предупреждение

Обращение к методам должно быть немного другим, нежели к функциям.

Интерфейс OLE AutoServer

Я пытаюсь создать внутренний (in-process) OLE-сервер с возможностью обратного вызова (callback). Хочу передавать OLE-объект MS C++ DLL так, чтобы DLL могла быть вызвана из сервера. Проблема в том, что DLL «вылетает», если в качестве сервера выступает Delphi 2.0, но работает в VB 4.0.

Проблема в том, что вы передаете со стороны Delphi `Variant`, но на стороне C++ «ожидают» `IUnknown`. Измените прототип функции Delphi следующим образом:

```
function SmtOleLink(OleCallBack: IUnknown; ...) ...;
```

Для получения доступа к типу IUnknown необходимо добавить "Ole2" к списку используемых модулей. Теперь измените вызов со стороны Delphi:

```
SmtOleLink(VarToInterface(MyObject), 16, 0);
```

Функция VarToInterface (определенная в модуле OleAuto) извлекает указатель IDispatch из Variant (или возбуждает исключение, если Variant не содержит ссылки на объект OLE Automation).

[News Group]

Вызов DLL Delphi из MS Visual C++

Во-первых, создайте в Delphi простую DLL:

```
Library MinMax;

function Min(X, Y: Integer): Integer;
         export;
begin
  if X < Y then Min := X else Min := Y;
end;

function Max(X, Y: Integer): Integer;
         export;
begin
  if X > Y then Max := X else Max := Y;
end;

Exports
  Min index 1,
  Max index 2;

begin

end.
```

Затем для вызова этих функций из вашего C-кода сделайте следующее:

В .DEF-файл добавьте следующие строки:

```
IMPORTS
  Min = MINMAX.Min
  Max = MINMAX.Max
```

Объявите в C-приложении прототипы функций, как показано ниже:

```
int FAR PASCAL Min(int x, y);
int FAR PASCAL Max(int x, y);
```

Теперь из любого места своего приложения вы можете вызвать функции Min и Max.

Проблема использования в DLL чисел с плавающей точкой

Если вы создаете DLL не с помощью Delphi, то избегайте чисел с плавающей точкой в возвращаемом значении. Вместо этого для возвращаемого значения используйте var-параметр (указатель или ссылочный параметр в C++).

Причина кроется в том, что Borland и Microsoft применяют различные способы возврата чисел с плавающей точкой. Borland C++ и Delphi могут использовать один и тот же метод.

Не задавайте одинарную или двойную точность. Они могут быть изменены компилятором. Применяйте тип double.

[News Group]

DLL – убийственная утилита

Метод для удаления DLL из памяти. На форме присутствует одно поле редактирования TEdit с именем EditDLLName, кнопки Ok и Close. Следующий код выполняется при нажатии кнопки Ok:

```
procedure TForm1.OkBtnClick(Sender: TObject);
var
  hDLL: THandle;
  aName: array[0..10] of char;
  FoundDLL: Boolean;
begin
 if EditDLLName.Text = '' then begin
 MessageDlg('Сначала вы должны ввести имя выгружаемой DLL!', mtInformation,
             [mbOk], 0);
    exit;
  end;
  StrPCopy(aName, EditDLLName.Text);
  FoundDLL := False;

  repeat
    hDLL := GetModuleHandle(aName);
    if hDLL = 0 then break;
    FoundDLL := True;
    FreeLibrary(hDLL);
  until False;
  if FoundDLL then
    MessageDlg('Успешно!', mtInformation, [mbOk], 0)
  else
    MessageDlg('DLL не найдена!', mtInformation, [mbOk], 0);
  EditDLLName.Text := '';
end;
```

Импортирование или «обертка» вызовов функций DLL

Существует два метода для импорта и загрузки функций из Dynamic Link Library (DLL). Первый метод, обсуждаемый здесь, называется «неявной» (`Implicit`) загрузкой. Неявная загрузка включает в себя статическую загрузку DLL при запуске программы и получение доступа к функциям через интерфейс Object Pascal. Данный метод должен использоваться в случае, если приложение полностью зависит от загрузки DLL для соответствующего функционирования. Другой метод доступа называется «явной» загрузкой, поскольку DLL загружается динамически по требованию. Этот метод требует дополнительного кодирования и должен применяться, если приложению нужно продолжить работу даже тогда, когда DLL не смогла правильно загрузиться.

Что такое «обертка» функциональных вызовов? Обертка функции или набора функций состоит из объявлений в секции `interface` и кода в секции `implementation` (вместе со связанными константами или типами), которые соответствуют функции или набору функций, импортируемых из DLL. Обертка является простой декларацией в паскалевском модуле, которая обеспечивает точку входа в DLL. В Delphi обертка представляет собой файл модуля, содержащий код Object Pascal. Группа разработчиков Delphi уже создала для вас обертку функций и стандартных элементов управления Windows. Иногда возникает необходимость создания обертки для вызовов функций DLL, но в Delphi это не реализовано в связи с сугубой индивидуальностью каждой DLL и входящих в нее функций. Первым и самым сложным шагом в данном процессе является получение информации о функциях. Одним из лучших источников для получения документации об имеющихся (доступных) в DLL функциях является World Wide Web. Начать поиск можно с MSDN, а если и там ничего не найдено, то можно обратиться к многочисленным поисковым серверам, которые частенько находят нужную информацию. Для получения структуры вызовов функций ищите заголовочные файлы C++ в продуктах типа Borland C++ или MS Visual C++. Соглашения о вызовах и преобразованиях типов обычно способны разрешить конфликты и несовместимость вызовов между C++ и PASCAL. Хороший ресурс по вопросам совместимости между Delphi и C++ расположен на сайте Borland по адресу: *http://www.borland.com/delphi/papers/brick.html*.

Итак, вы нашли необходимый пример или документацию по экспортируемым DLL функциям. Следующим шагом будет создание нового модуля. Интерфейс модуля будет содержать константы и типы, необходимые для вызова отдельных функций DLL, и заголовки самих функций. Данные заголовки функций являются «объектно-паскалевским» интерфейсом, позволяющим другим приложениям Delphi вызывать функции рассматриваемой DLL. Завершив секцию модуля `interface`, принимайтесь за секцию `implementation`. Секция модуля `implementation` содержит объявления импортируемых внешних функций. Эти заголовки не идентичны тем, которые расположены в секции модуля `interface` (и содержат реальные идентификаторы функций плюс другую важную информацию для реализации). Для получения дополни-

тельной информации по этой теме обратитесь к разделу «DLLs: accessing procedures and functions» справки Delphi.

Представим себе, что у нас есть функция с именем BOB в DLL с именем BLODGE.DLL. Ниже приведены подробные и необходимые шаги, где подразумевается, что мы будем использовать неявную загрузку DLL:

- Поиск информации в Интернете показал, что функция BOB должна возвращать логическое значение и требует в качестве аргументов значения типа word и boolean.
- Создайте в Delphi новый файл модуля с именем 'UseBob.pas' (File/New и выберите Unit).
- Следующая строка кода должна располагаться в секции interface нового модуля:

```
function BOB(Fire: Word; Dances: Boolean): Boolean; stdcall;
```

- Следующая строка кода должна располагаться в секции implementation нового модуля:

```
function BOB; external 'BLODGE';
```

- Сохраните модуль с именем UseBob.pas.
- Необходимо убедиться, что UseBob.pas расположен в каталоге текущего проекта или в каталоге, прописанном в путях поиска Delphi.
- Добавьте модуль UseBob к списку uses модуля нового проекта. Теперь функция BOB может быть вызвана из нового проекта подобно любой другой стандартной функции.
- Во время выполнения приложения файл BLODGE.DLL должен находиться в пути текущих переменных окружения процесса.

Для способа, при котором BLODGE.DLL должна быть загружена явно, требуется дополнительное кодирование. Как было подчеркнуто выше, необходимо знание аргументов функций/процедур (и тип результата в случае функции).

Ниже приведен модуль с реализацией вызова функции BOB, инициализируемый при нажатии кнопки:

```
unit UDLLTest;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  end;
```

```
{ типы, которые требуются для работы нашей функции bob }
  TBOB = function(Fire: Word; Dances: Boolean): Boolean; stdcall;

var
  Form1: TForm1;

implementation
{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
var
  BOB: TBOB;
  hDLLInst: THandle;
  IsAlive, IsDancing: Boolean;
  Years: Word;
begin
{ Загружаем и получаем дескриптор нашего BLODGE.DLL }
  hDLLInst := LoadLibrary('BLODGE.DLL');
{ Если загрузка не была успешной, генерируем свое исключение }
  if (hDLLInst <= 0) then
    raise exception.create('[Неудачный вызов LoadLibrary]');
{ Попытаемся получить адрес функции BOB }
  try
    @BOB := GetProcAddress(hDLLInst, 'BOB');
    if not Assigned(BOB) then
      raise Exception.Create('[Неудачный вызов GetProcAddress]');
      Years := 25;
      IsDancing := True;

{ Теперь мы можем выполнить функцию BOB }
      IsAlive := BOB(Years, IsDancing);
    finally
{ Освобождаем дескриптор DLL }
      FreeLibrary(hDLLInst);
  end;
end;

end.
```

Uses в DLL

Нам необходимы по крайней мере два файла – библиотечный файл и файл с исходным кодом. Библиотечный файл mylib.dpr:

```
library MyLib;

uses
  MyCode in 'MYCODE.PAS';
exports MyFunc index 1;
begin
end.
```

Файл с исходным кодом: mycode.pas:

```
unit MyCode;

interface

function MyFunc(MyParam: string): string; export;

implementation

function MyFunc(MyParam: string): string;
begin
  Result := 'Это просто пример!';
end;

end.
```

Функции VER.DLL

Информация в диалоговом окне О программе. Код приведен ниже. Блок StringFileInfo вы можете и не использовать – он необходим для осуществления простейшей проверки, но вместо этого можно получить информацию из корневого блока (для дополнительной информации см. описание структуры TVS_FIXEDFILEINFO в файле справки по API).

```
procedure TAboutBox.FormCreate(Sender: TObject);
var
  VIHandle : LongInt;
  VSize: LongInt;
  VData: Pointer;
  VVers: Pointer;
  Len: Word;
  FileName: String;

const
  Prefix = '\StringFileInfo\040904E4\'; { Предустановленный набор символов U.S. }

function GetVerValue(Value: String): String;
var
  ItemName: String;
begin
  ItemName := Prefix + Value + chr(0);
  Result := '';
  if VerQueryValue(VData, @ItemName[1], VVers, Len) then
    if Len > 0 then begin
      if Len > 255 then
        Len := 255                        { "Обрезаем" любые длинные строки }
        Move(VVers^, Result[1], Len);
        Result[0] := Chr(Len);
    end;
end;
```

```
begin
  FileName := Application.EXEName + chr(0);
  VSize := GetFileVersionInfoSize(@FileName[1], VIHandle);
  if VIHandle <> 0 then begin
    GetMem(VData, VSize);
    try
      if GetFileVersionInfo(@FileName[1], VIHandle, VSize, VData) then begin
{ В этом месте мы получаем значения из блока StringFileInfo, но точно также мы
  могли бы взять значения из корневого блока, используя VerQueryValue }
        ProductName.Caption := GetVerValue('ProductName');
        Version.Caption := GetVerValue('ProductVersion');
        Copyright.Caption := GetVerValue('LegalCopyright');
        Comments.Caption := GetVerValue('FileDescription');
      end;
    finally
      FreeMem(VData, VSize);
    end;
  end;
end;
```

Примечание

Этот совет написан на Delphi 1 для Win16, поэтому желающим его адаптировать для старших версий среды необходимо будет изменить вызовы почти всех функций, т. к. некоторые не только «изменили» типы своих параметров, но и свои имена. Пользуйтесь контекстной подсказкой Delphi, т. к. в справке по Windows API есть ошибки!

13

Часто задаваемые вопросы (FAQ)

Можно ли загрузить BP-программу (или как проект, или как программу) в IDE и скомпилировать ее?

Да, можно, но необходима небольшая квалифицированная подсказка. Если быть более точным, в Delphi можно загрузить BPW-программу. Выберите пункт Open Project меню File или соответствующую кнопку на панели управления и в выпадающем списке типов файлов выберите элемент *.PAS. Это позволит загрузить программы BPW и файлы *.PAS. Не забудьте добавить \DELPHI\SOURCE\RTL70 в пути поиска вашего проекта.

Могу ли я, компилируя BP7-проект в Delphi, использовать интегрированный отладчик? Если да, то как указать Delphi, какой файл является первичным? А если в проекте нет ни одной формы?

Да, вы можете воспользоваться встроенным отладчиком. Первичный файл – файл с расширением *.PAS, который был открыт как проект. Выберите Open Project, измените тип файла с *.DPR на *.PAS и затем выберите программу BP7, которую хотите компилировать.

Могу ли я воспользоваться вторым монитором для контроля выходных данных BP-программы?

Для работы с двумя мониторами необходимо, чтобы один из них был MDA-монитором (старый черно-белый стиль Hercules), а другой – монитором стандарта VGA (или EGA и т. п.). Они должны использовать два адресных пространства: $B000 для MDA и $B800 для VGA. После установки обоих мониторов можно сообщить BP IDE о наличии второго монитора в Options|Environment|StartUp. Монитор по умолчанию, стартующий BP IDE, будет отображать поток выходных данных программы.

Примечание

После установки карты Hercules VGA-карта будет доступна только как 8-битная. Delphi данную характеристику не поддерживает.

Когда я компилирую код с использованием модуля WinPrn, Delphi говорит, что не может найти WinPrn.DCU.

Добавьте \DELPHI\SOURCE\RTL70 к пути поиска вашего проекта.

Почему при закрытии окна модуля (pas) также закрывается и окно с формой?

Delphi должен работать с кодом при манипуляциях с формой. Delphi постоянно следит за тем, что вы делаете, и изменяет код для отражения изменений на форме.

В каком количестве компоненты используют GDI-ресурсы?

Потомки TGraphicControl типа TLabel и TBevel не обращаются к свободным системным ресурсам вообще. Используют их потомки TWinControl типа TEdit и TPanel. Также каждый переключатель (radiobutton) в TRadioGroup является окном, т. е. имеет дескриптор окна.

Как хранится в Delphi код формы?

Формы хранятся в файлах DFM (в двоичном виде), но если открыть один из них (FILE|OPEN FILE) в редакторе IDE как текст, то можно будет редактировать его непосредственно.

Почему простейшие EXE-файлы имеют такой большой размер (одна форма с одной кнопкой занимает 200 Кбайт)?

Delphi VCL основана на RTTI и исключениях. Это уже примерно 120 Кбайт от «пустого» приложения. 200 Кбайт вы получаете за счет добавочной информации или не оптимизированного компилятором кода. Обратите внимание на тот факт, что при добавлении второй кнопки размер вашего EXE-файла составит не 400 Кбайт, а всего 201, т. е. после необходимого кода располагаются обычные данные/код с типичными (ожидаемыми) размерами.

Почему моя программа не может найти ресурсы, упакованные мной в .RES-файл, если .RES-файл имеет то же самое имя, что и модуль формы?

Если имя используемого вами .RES-файла совпадает с именем .DPR-файла, Delphi перезапишет его, создав собственный .RES-файл с именем проекта. Кроме того, проектный RES-файл предназначен только для менеджера проекта Delphi, не редактируйте этот файл и не добавляйте к нему свои ресурсы.

Как поменять ширину и высоту редактора IDE по умолчанию?

Все соответствующие значения находятся в реестре в следующих ключах:

```
HKEY_CURRENT_USER\Software\Borland\Delphi\2.0\Editor
```

```
HKEY_CURRENT_USER\Software\Borland\Delphi\3.0\Editor
HKEY_CURRENT_USER\Software\Borland\Delphi\4.0\Editor
HKEY_CURRENT_USER\Software\Borland\C++Builder\1.0\Editor
```

В этих ключах находятся DefaultHeight и DefaultWidth, определяющие высоту и ширину окна редактора, соответственно.

Как программно поменять обои Windows?

Объяснение не требуется. Посмотрите код:

```
SystemParametersInfo(SPI_SETDESKWALLPAPER, 0, PChar('C:\SOMEPATH\SOME.BMP'),
                     SPIF_SENDWININICHANGE);
```

Как поменять обои в Windows 98/Windows 2000, когда включен Active Desktop?

Да, в данном случае стандартное изменение через SystemParametersInfo не пройдет. Придется использовать IActiveDesktop. Но учтите, что для его применения необходима библиотека SHELL32.DLL версии старше 4.71. В этой главе есть пример, показывающий, как узнавать версию файла. Можете им воспользоваться.

```
uses
  ComObj, // For CreateComObject and Initialization/Finalization of COM
  ShlObj; // For IActiveDesktop

{ The CLASS ID for ActiveDesktop is not defined in ShlObj, while the IID is so we
  define it here. }
const
  CLSID_ActiveDesktop: TGUID = '{75048700-EF1F-11D0-9888-006097DEACF9}';

{ Demonstrate getting the Wallpaper }
procedure TForm1.Button1Click(Sender: TObject);
var
  ActiveDesktop: IActiveDesktop;
  CurrentWallpaper: string;
  CurrentPattern: string;
  WallpaperOptions: TWallpaperOpt;
  tmpBuffer: PWideChar;
begin
// Create the ActiveDesktop COM Object
  ActiveDesktop := CreateComObject(CLSID_ActiveDesktop) as IActiveDesktop;
// We now need to allocate some memory to get the current Wallpaper.
// However, tmpBuffer is a PWideChar which means 2 bytes make
// up one Char. In order to compenstate for the WideChar, we
// allocate enough memory for MAX_PATH*2
  tmpBuffer := AllocMem(MAX_PATH * 2);
  try
    ActiveDesktop.GetWallpaper(tmpBuffer, MAX_PATH * 2, 0);
    CurrentWallpaper := tmpBuffer;
  finally
    FreeMem(tmpBuffer);
  end;
```

```
  if CurrentWallpaper <> '' then
    Label1.Caption := 'Current Wallpaper: ' + CurrentWallpaper
  else
    Label1.Caption := 'No Wallpaper set';
// Now get the current Wallpaper options.
// The second parameter is reserved and must be 0.
  WallpaperOptions.dwSize := SizeOf(WallpaperOptions);
  ActiveDesktop.GetWallpaperOptions(WallpaperOptions, 0);
  case WallpaperOptions.dwStyle of
     WPSTYLE_CENTER:  Label2.Caption := 'Centered';
       WPSTYLE_TILE:  Label2.Caption := 'Tiled';
    WPSTYLE_STRETCH:  Label2.Caption := 'Stretched';
        WPSTYLE_MAX:  Label2.Caption := 'Maxed';
  end;
// Now get the desktop pattern. The pattern is a string of decimals whose bit
// pattern represents a picture. Each decimal represents the on/off state
// of the 8 pixels in that row.
  tmpBuffer := AllocMem(256);
  try
    ActiveDesktop.GetPattern(tmpBuffer, 256, 0);
    CurrentPattern := tmpBuffer;
  finally
    FreeMem(tmpBuffer);
  end;
  if CurrentPattern <> '' then Label3.Caption := CurrentPattern
  else Label3.Caption := 'No Pattern set';
end;

{ Demonstrate setting the wallpaper }
procedure TForm1.Button2Click(Sender: TObject);
var
  ActiveDesktop: IActiveDesktop;
begin
  ActiveDesktop := CreateComObject(CLSID_ActiveDesktop) as IActiveDesktop;
  ActiveDesktop.SetWallpaper('c:\downloads\images\test.bmp', 0);
  ActiveDesktop.ApplyChanges(AD_APPLY_ALL or AD_APPLY_FORCE);
end;
```

Можно ли рисовать непосредственно на рабочем столе компьютера?

Да. Достаточно получить DC посредством функции `GetDc(0)`.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  dc: HDC;
begin
  dc := GetDc(0);
  MoveToEx(Dc, 0, 0, nil);
  LineTo(Dc, 300, 300);
  ReleaseDc(0, Dc);
end;
```

Как обнаружить изменения конфигурации PnP в системе?

Надо перехватить сообщение WM_DEVICECHANGE:

```
type
  TForm1 = class(TForm)
  private
    procedure WMDeviceChange(var Message: TMessage); message WM_DEVICECHANGE;
  end;
var
  Form1: TForm1;

implementation

{$R *.DFM}

const
  DBT_DEVICEARRIVAL = $8000;
  DBT_DEVICEQUERYREMOVE = $8001;
  DBT_DEVICEQUERYREMOVEFAILED = $8002;
  DBT_DEVICEREMOVEPENDING = $8003;
  DBT_DEVICEREMOVECOMPLETE = $8004;
  DBT_DEVICETYPESPECIFIC = $8005;
  DBT_CONFIGCHANGED = $0018;

procedure TForm1.WMDeviceChange(var Message: TMessage);
var
  s: string;
begin
{ Do Something here }
  case Message.wParam of
    DBT_DEVICEARRIVAL:  s := 'A device has been inserted and is now available';
DBT_DEVICEQUERYREMOVE: begin
                        s := 'Permission to remove a device is requested';
                        ShowMessage(s);
                        { True grants permission }
                        Message.Result := integer(true);
                        exit;
                       end;
DBT_DEVICEQUERYREMOVEFAILED:  s := 'Request to remove a device has been canceled';
DBT_DEVICEREMOVEPENDING:  s := 'Device is about to be removed';
DBT_DEVICEREMOVECOMPLETE:  s := 'Device has been removed';
DBT_DEVICETYPESPECIFIC:  s := 'Device-specific event';
    DBT_CONFIGCHANGED:  s := 'Current configuration has changed'
                else
                        s := 'Unknown Device Message';
  end;
  ShowMessage(s);
  inherited;
end;
```

Можно ли определить, что мое приложение запущено под Windows NT?

```
{$IFNDEF WIN32}
const
```

```
  WF_WINNT = $4000;
{$ENDIF}

function IsNT: bool;
{$IFDEF WIN32}
var
  osv: TOSVERSIONINFO;
{$ENDIF}
begin
  result := true;
{$IFDEF WIN32}
  GetVersionEx(osv);
  if osv.dwPlatformId = VER_PLATFORM_WIN32_NT then exit;
{$ELSE}
  if ((GetWinFlags and WF_WINNT) = WF_WINNT ) then exit;
{$ENDIF}
  result := false;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  if IsNt then ShowMessage('Running on NT')
  else ShowMessage('Not Running on NT');
end;
```

Как определить системную директорию и директорию Windows?

Для этого используйте `GetWindowsDirectory()` и `GetSystemDirectory()`.

```
{$IFNDEF WIN32}
  const MAX_PATH = 144;
{$ENDIF}

procedure TForm1.Button1Click(Sender: TObject);
var
  a: Array[0..MAX_PATH] of char;

begin
  GetWindowsDirectory(a, SizeOf(a));
  ShowMessage(StrPas(a));
  GetSystemDirectory(a, SizeOf(a));
  ShowMessage(StrPas(a));
end;
```

Как удалить файл в Корзину?

Обратитесь к `SHFileOperation`.

```
uses ShellAPI;

procedure SendToRecycleBin(FileName: string);
var
  SHF: TSHFileOpStruct;
begin
  with SHF do begin
    Wnd := Application.Handle;
```

```
    wFunc := FO_DELETE;
    pFrom := PChar(FileName);
    fFlags := FOF_SILENT or FOF_ALLOWUNDO;
  end;
  SHFileOperation(SHF);
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  SendToRecycleBin('c:\DownLoad\Test.gif');
end;
```

Из какой части кода программы лучше всего вызвать при ее запуске окно с логотипом?

Наилучшее место для вывода окна с логотипом – в исходном коде проекта после первого Application.FormCreate и перед Application.Run. Этим мы осуществляем создание формы «на лету» и отображение ее до момента фактического запуска приложения.

```
program Project1;

uses
  Forms, Unit1 in 'UNIT1.PAS' {Form1}, Splash;

{$R *.RES}
var
  SplashScreen: TSplashScreen;     { в модуле Splash }
begin
  Application.CreateForm(TForm1, Form1);
  SplashScreen := TSplashScreen.Create(Application);
  try
    SplashScreen.Show;
    SplashScreen.Update; { Обрабатываем сообщения Windows, касающиеся отрисовки }
{ Осуществите остальные CreatForms или другие действия, прежде чем приложение
запустится. Если процедура запуска более или менее продолжительна, то для того,
  чтобы окно реагировало на системные сообщения, необходимо периодически давать
  команду Application.ProcessMessages. }
  finally                { Убедитесь, что окно с логотипом освобождается }
    SplashScreen.Free;
  end;
  Application.Run;
end.
```

Как программно перезагрузить Windows?

Функция ExitWindows неправильно документирована. У нее в наличии три значения первого параметра и второй параметр, используемый для запуска DOS-программ при выходе из Windows: EW_RESTARTWINDOWS, EW_REBOOTSYSTEM, EW_EXITANDEXECAPP.

Пример:

```
ExitWindows(EW_RESTARTWINDOWS, 0);
```

Как передать код выхода (errorlevel) запустившей программе или .bat-файлу?

Выйдите командой `Halt()` с параметром кода выхода:

```
begin
  ...
  Halt(2);
  ...
end.
```

*Как запустить *.lnk?*

```
uses ShellApi;

procedure TForm1.Button1Click(Sender: TObject);
begin
  ShellExecute(0, nil, 'C:\WINDOWS\START MENU\DELPHI\Delphi3.lnk', nil, nil,
               SW_SHOWNORMAL);
end;
```

Как определить, что запущено: среда Delphi или C++ Builder?

Все по-прежнему решается через `FindWindow`. Учтите, что основной класс указанных программ – `TAppBuilder`.

```
if FindWindow('TAppBuilder', Nil) <> 0 then
  ShowMessage('Delphi and (or) C++ Builder is running');
```

Как обойтись без TTimer и наиболее рационально использовать ресурсы системы?

```
procedure Delay(ms: longint);
var
  TheTime: LongInt;
begin
  TheTime := GetTickCount + ms;
  while GetTickCount < TheTime do
    Application.ProcessMessages;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  ShowMessage('Start Test');
  Delay(2000);
  ShowMessage('End Test');
end;
```

Какую информацию показывает Инспектор объектов?

Инспектор объектов (Object Inspector) отображает все опубликованные (`published`) свойства класса/объекта и служит как для просмотра значений свойств, так и для их изменения. Инспектор объектов не показывает методы класса.

Как определить класс компонента, при щелчке по которому появляется контекстное меню?

Обратитесь к свойству PopupComponent компонента PopupMenu для определения компонента, на котором была нажата правая кнопка мыши.

```
procedure TForm1.PopupItem1Click(Sender: TObject);
begin
  Label1.Caption := PopupMenu1.PopupComponent.ClassName;
end;
```

Можно использовать свойство формы ActiveControl, но компонент, вызвавший контекстное меню, не обязательно должен быть активным элементом управления.

Возможно ли создание глобальных переменных в Delphi?

Откройте новый модуль и объявите необходимые глобальные структуры в секции interface (секция implementation должна оставаться пустой). Добавьте имя данного модуля в список USES тех модулей, для которых необходимо получить доступ к объявленным глобальным переменным.

Какие вопросы необходимо учесть при создании приложения, ориентированного на работу при различных разрешениях экрана (масштабирование форм)?

При разработке приложения для различных режимов разрешения надо учитывать следующие рекомендации:

- Заранее, в самом начале этапа разработки, решите для себя – собираетесь ли вы разрешить масштабирование формы или нет. Преимущество запрета масштабирования в том, что вам ничего не нужно менять во время выполнения приложения. Недостаток запрета масштабирования – во время выполнения приложения никаких изменений не происходит (ваша форма может быть слишком малой или слишком большой для работы в некоторых режимах при отсутствии масштабирования).
- Если вы не собираетесь масштабировать форму, установите свойство Scaled в False.
- В противном случае установите свойство формы Scaled в True.
- Установите AutoScroll в False. AutoScroll = True. Это означает «не изменять размер окна формы во время выполнения приложения», что приводит к «плохому виду» формы, если ее содержимое меняет размер.
- Установите шрифты формы в масштабируемые шрифты TrueType типа Arial. MS San Serif также подойдет в качестве альтернативы, только помните, это не TrueType-, а BitMapped-шрифт. Только Arial может правильно изменять свою высоту с дискретностью 1 пиксел.

Примечание

Если используемый шрифт не установлен на машине пользователя, Windows выбирает альтернативный шрифт из данной линейки (семьи) шрифтов. Размеры нового шрифта могут отличаться от размеров оригинального шрифта, что также может вызвать проблемы.

- Установите свойство формы `Position` не в `poDesigned`, а в какое-нибудь другое значение. `poDesigned` всегда показывает форму в первозданном виде. Представьте себе, на что будет похожа форма, разработанная в разрешении 1280×1024, будучи выведенной на экран при разрешении 640×480.
- Не «сцепляйте» на форме элементы управления, оставляйте между ними по крайней мере 4 пиксела, в противном случае при изменении положения границы на 1 пиксел (это происходит при масштабировании) элементы управления наедут друг на друга.
- Для однострочных компонентов `Label`, у которых свойство `Aligned` равно `alLeft` или `alRight`, установите `AutoSize` в `True`. В противном случае установите `AutoSize` в `False`.
- Убедитесь, что компоненты `Label` имеют достаточный запас по ширине (требуется примерно 25%) от длины текущего текста. (При переводе приложения на другие языки необходим примерно 30-процентный запас от текущей ширины текста.) Если свойство `AutoSize` установлено в `False`, убедитесь, что ширины компонента `Label` достаточно для размещения реального текста. Если `AutoSize` равно `True`, убедитесь, что компонент `Label` может вместить (например, на форме) весь текст и есть еще запас, который может пригодиться при смене шрифтов.
- В случае многострочного текста и компонентов `Label` с переносом слов убедитесь, что в нижней части имеется по крайней мере еще одна строчка. Она необходима для того, чтобы не допустить переполнения строки, если размер шрифта увеличится при масштабировании. Не думайте, что если вы работаете с большими шрифтами и переполнение не возникает, то эта проблема снята – кто-нибудь может использовать шрифты с еще большим размером!
- Будьте осторожны при открытии проекта в IDE с другим разрешением. Свойство формы `PixelsPerInch` будет изменено, как только вы откроете форму, и сохранено в `DFM`-файле при сохранении проекта. Лучше всего запускать приложение отдельно от IDE, а редактировать его при одном разрешении. Редактируя формы при различных разрешениях и размерах шрифтов, вы провоцируете «дрейф» компонентов по форме и изменения их размеров.
- Из-за опасности дрейфа компонентов не следует многократно масштабировать форму, как во время разработки, так и во время выполнения приложения. Каждое изменение размеров сопровождается ошибками округления, которые достаточно быстро накапливаются (с тех пор как

координаты стали строго целочисленными). Поскольку при вычислении новых размеров дробная часть отбрасывается, вновь пересчитанные размеры оказываются меньше, а координаты элементов управления – северо-западнее. Если вы решили дать пользователю возможность изменять масштабы форм, начинайте масштабирование с последней загруженной/созданной формы, этим вы уменьшите накапливаемые при масштабировании ошибки.

- Старайтесь не изменять значение свойства формы `PixelsPerInch`.
- В общих словах, нет необходимости разрабатывать формы для всех возможных режимов, перед окончательным релизом приложения следует оценить поведение формы в пограничных режимах – 640×480 с мелкими и крупными шрифтами и при высоком разрешении (также с мелкими и крупными шрифтами). Это должно быть частью регулярных проверок на предмет системной совместимости для ведения так называемой тестирующей контрольной таблицы.
- Обратите пристальное внимание на однострочные компоненты `TMemo` типа `TDBLookupCombo`. Системные многострочные редакторы всегда выводят только целые строки текста – если ширина элемента управления слишком мала для своего шрифта, то `TMemo` вообще ничего не показывает (`TEdit` отображает обрезанный текст). Размер таких компонентов лучше сделать на несколько пикселов больше, чем на несколько пикселов меньше – это позволит определить наличие в компоненте оставшейся части текста.
- Обратите внимание, что масштабирование во время проектирования и во время выполнения программы отличается коэффициентом и зависит от высоты шрифта, а не от экранного разрешения в пикселах. Помните также, что «начало» компонентов будет изменяться в зависимости от масштаба формы, а для их «броуновского» движения также необходимо небольшое пространство.

Как отключить монитор (перевести в «зеленый» режим)?

В списках FAQ на сайте компании Borland почему-то не указано, что приведенный ниже пример не работает под Windows NT/2000, но тем не менее это так.

```
type
  TForm1 = class(TForm)
    Button1: TButton;
    Timer1: TTimer;
    procedure FormCreate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  public
    MonitorOff: bool;
  end;

var
```

```
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
  Timer1.Enabled := false;
  Timer1.Interval := 10000;
  MonitorOff := false;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  if MonitorOff then begin
    MonitorOff := false;
    SendMessage(Application.Handle, wm_SysCommand, SC_MonitorPower, -1);
    Timer1.Enabled := false;
  end;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  MonitorOff := true;
  Timer1.Enabled := true;
  SendMessage(Application.Handle, wm_SysCommand, SC_MonitorPower, 0);
end;
```

Как вывести на форму изображение и текст, не используя для этого визуальные компоненты?

```
procedure TForm1.Button1Click(Sender: TObject);
var
  dc: hdc;
  MemDc: hdc;
  MemBitmap: hBitmap;
  OldMemBitmap: hBitmap;
begin
{ Get the handle to the screen's dc }
  dc := GetDc(0);
{ Create and retrieve a handle to a memory dc based on the screen }
  MemDc := CreateCompatibleDc(dc);
{ Create a bitmap that is compatible with the display. Note: if you pass "MemDc" to
 CreateCompatibleBitmap() instead of "dc", you will get a monochrome bitmap! }
  MemBitmap := CreateCompatibleBitmap(dc, 100, 100);
{ Release the screen dc }
  ReleaseDc(0, dc);
{ Select the bitmap surface into the MemDc remembering the default bitmap }
  OldMemBitmap := SelectObject(MemDc, MemBitmap);
{ Draw on the MemoryDc }
  PatBlt(MemDc, 0, 0, 100, 100, WHITENESS);
  Ellipse(MemDc, 0, 0, 100, 100);
{ Copy the MemDc to the Form Canvas }
```

```
  BitBlt(Form1.Canvas.Handle, 100, 100, 100, 100, MemDc, 0, 0, SRCCOPY);
{ Select the default bitmap back into the memory dc }
  SelectObject(MemDc, OldMemBitmap);
{ You can now use the memory bitmap handle with functions such as GetDiBits() }
{ Delete the Memory Bitmap }
  DeleteObject(MemBitmap);
{ Delete the MemoryDc }
  DeleteDc(MemDc);

end;
```

Как обновить пиктограммы на рабочем столе?

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  SendMessage(FindWindow('Progman', 'Program Manager'), WM_COMMAND, $A065, 0);
end;
```

Как извлечь доли составляющих цветов из данного цвета?

Используйте `GetRValue()`, `GetGValue()` и `GetBValue()`.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Form1.Canvas.Pen.Color := clRed;
  Memo1.Lines.Add('  Red = ' + IntToStr(GetRValue(Form1.Canvas.Pen.Color)));
  Memo1.Lines.Add('Green = ' + IntToStr(GetGValue(Form1.Canvas.Pen.Color)));
  Memo1.Lines.Add(' Blue = ' + IntToStr(GetBValue(Form1.Canvas.Pen.Color)));
end;
```

*Как извлечь пиктограмму из *.EXE или *.DLL?*

Используйте `ExtractIcon()`, передавая ей дескриптор приложения, путь к EXE и номер пиктограммы, которую требуется извлечь.

```
uses ShellAPI;

procedure TForm1.Button1Click(Sender: TObject);
var
  TheIcon: TIcon;
begin
  TheIcon := TIcon.Create;
  TheIcon.Handle := ExtractIcon(hInstance, 'c:\windows\notepad.exe', 0);
{ Do something with the icon }
  Image1.Picture.Icon := TheIcon;
  TheIcon.Free;
end;
```

Как создать мигающую пиктограмму (свернутое окно)?

Обратитесь к функции `FlashWindow()`.

```
var
  Flash: bool;

procedure TForm1.Timer1Timer(Sender: TObject);
```

```
begin
  FlashWindow(Form1.Handle, Flash);
  FlashWindow(Application.Handle, Flash);
  Flash := not Flash;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Flash := False;
end;
```

Как добавить пункт в системное меню программы?

Пример демонстрирует применение API-функции AppendMenu и последующий перехват WM_SYSCOMMAND именно с этой целью.

```
type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
  private
    procedure WMSysCommand(var Msg: TWMSysCommand); message WM_SYSCOMMAND;
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

const
  SC_MyMenuItem = WM_USER + 1;

procedure TForm1.FormCreate(Sender: TObject);
begin
  AppendMenu(GetSystemMenu(Handle, FALSE), MF_SEPARATOR, 0, '');
  AppendMenu(GetSystemMenu(Handle, FALSE), MF_STRING, SC_MyMenuItem, 'My Menu Item');
end;

procedure TForm1.WMSysCommand(var Msg: TWMSysCommand);
begin
  if Msg.CmdType = SC_MyMenuItem then ShowMessage('Got the message')
  else inherited;
end;
```

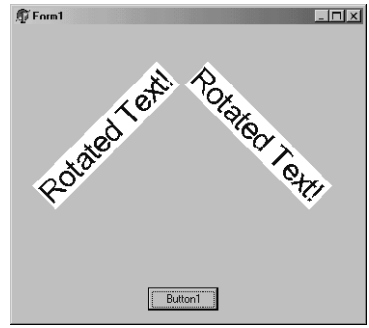
Как создать наклонный шрифт?

Можно создавать шрифты, основанные на уже существующих, но, теоретически, правильно работать будут только шрифты True Type.

Пример создает шрифт, повернутый на 45 градусов.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  lf: TLogFont;
  tf: TFont;
```

```
begin
  with Form1.Canvas do begin
    Font.Name := 'Arial';
    Font.Size := 24;
    tf := TFont.Create;
    tf.Assign(Font);
    GetObject(tf.Handle, SizeOf(lf), @lf);
    lf.lfEscapement := 450;
    lf.lfOrientation := 450;
    tf.Handle := CreateFontIndirect(lf);
    Font.Assign(tf);
    tf.Free;
    TextOut(20, Height div 2, 'Rotated Text!');
  end;
end;
```



Возможны ли изменения задаваемого по умолчанию размера шрифта компонентов, размещенных на форме?

Добавьте форму с установленным по желанию свойством Font в хранилище (Tools|Repository...). Располагаемые на форме компоненты принимают настройки шрифта родительской формы. Кроме того, можно установить новую форму как главную по умолчанию и также поместить ее в хранилище.

Как изменить Font.Style на нормальный после вызова Canvas.Font.Style := [fsBold]? В справке указаны стили ([fsBold], [fsItalic] и др.), но не указан нормальный стиль.

Определите пустое множество для набора стилей:

```
Canvas.Font.Style := [];
```

или исключите стиль жирного шрифта следующим образом:

```
Canvas.Font.Style := Canvas.Font.Style - [fsBold];
```

Можно ли изменить шрифт всплывающих подсказок?

```
type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
  public
    procedure MyShowHint(var HintStr: string; var CanShow: Boolean;
                         var HintInfo: THintInfo);
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.MyShowHint(var HintStr: string; var CanShow: Boolean;
                            var HintInfo: THintInfo);
var
  i: integer;
```

```
begin
  for i := 0 to Application.ComponentCount - 1 do
    if Application.Components[i] is THintWindow then
      with THintWindow(Application.Components[i]).Canvas do begin
        Font.Name := 'Arial';
        Font.Size := 18;
        Font.Style := [fsBold];
        HintInfo.HintColor := clWhite;
      end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Application.OnShowHint := MyShowHint;
end;
```

Как определить длину строки в пикселах для определенного шрифта?

Применяются два метода Canvas – TextHeight и TextWidth. Не забудьте назначить шрифт объекту Canvas перед тем, как что-либо нарисовать или провести измерения.

Все визуальные компоненты имеют свойство Canvas, но по умолчанию оно защищено (protected) для того, чтобы доступ к нему имели только прямые потомки. Но, поскольку вы создаете свой код на основе наследников TForm, то у вас практически всегда имеется доступ к унаследованному свойству формы Canvas. Компонент TPaintBox имеет доступное (public) свойство Canvas для того, чтобы в обработчике события компонента OnPaint вы могли бы что-либо нарисовать, но можно им воспользоваться и для наших целей.

Если компонент не имеет свойства Canvas, то следующая функция поможет возвратить ширину текста, основанного на определенном шрифте:

```
function GetTextWidth(CanvasOwner: TForm; Text: String; TextFont: TFont): Integer;
var
  OldFont: TFont;
begin
  OldFont := TFont.Create;
  try
    OldFont.Assign(CanvasOwner.Font);
    CanvasOwner.Font.Assign(TextFont);
    Result := CanvasOwner.Canvas.TextWidth(Text);
    CanvasOwner.Font.Assign(OldFont);

  finally
    OldFont.Free;
  end;
end;
```

Как вывести диалог выбора директории?

Используйте функцию SHBrowseForFolder()

```
uses
  ShellAPI, ShlObj;
```

```
procedure TForm1.Button1Click(Sender: TObject);
var
  TitleName: string;
  lpItemID: PItemIDList;
  BrowseInfo: TBrowseInfo;
  DisplayName: array[0..MAX_PATH] of char;
  TempPath: array[0..MAX_PATH] of char;
begin
  FillChar(BrowseInfo, SizeOf(TBrowseInfo), #0);
  BrowseInfo.hwndOwner := Form1.Handle;
  BrowseInfo.pszDisplayName := @DisplayName;
  TitleName := 'Please specify a directory';
  BrowseInfo.lpszTitle := PChar(TitleName);
  BrowseInfo.ulFlags := BIF_RETURNONLYFSDIRS;
  lpItemID := SHBrowseForFolder(BrowseInfo);
  if lpItemId <> nil then begin
    SHGetPathFromIDList(lpItemID, TempPath);
    ShowMessage(TempPath);
    GlobalFreePtr(lpItemID);
  end;
end;
```

Как получить серийный номер тома жесткого диска?

Посмотрите пример, в нем есть и еще кое-какая полезная информация.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  VolumeName, FileSystemName: array [0..MAX_PATH - 1] of Char;
  VolumeSerialNo, MaxComponentLength, FileSystemFlags: DWord;
begin
  GetVolumeInformation('C:\', VolumeName, MAX_PATH, @VolumeSerialNo,
                       MaxComponentLength, FileSystemFlags, FileSystemName, MAX_PATH);
  Memo1.Lines.Add('VName = ' + VolumeName);
  Memo1.Lines.Add('SerialNo = $' + IntToHex(VolumeSerialNo, 8));
  Memo1.Lines.Add('CompLen = ' + IntToStr(MaxComponentLength));
  Memo1.Lines.Add('Flags = $' + IntToHex(FileSystemFlags, 4));
  Memo1.Lines.Add('FSName = ' + FileSystemName);
end;
```

Как получить список доступных носителей?

```
procedure TForm1.Button1Click(Sender: TObject);
var
  ld: DWORD;
  i: integer;
begin
  ld := GetLogicalDrives;
  for i := 0 to 25 do begin
    if (ld and (1 shl i)) <> 0 then
      Memo1.Lines.Add(Char(Ord('A') + i) + ':\');
  end;
end;
```

Как распознать тип носителя?

Используем функцию GetDriveType.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  case GetDriveType('C:\') of
                    0:  ShowMessage('The drive type cannot be determined');
                    1:  ShowMessage('The root directory does not exist');
      DRIVE_REMOVABLE:  ShowMessage('The disk can be removed');
          DRIVE_FIXED:  ShowMessage('The disk cannot be removed');
         DRIVE_REMOTE:  ShowMessage('The drive is remote (network) drive');
          DRIVE_CDROM:  ShowMessage('The drive is a CD-ROM drive');
        DRIVE_RAMDISK:  ShowMessage('The drive is a RAM disk');
  end;
end;
```

Как, проверяя готовность носителя, подавить сообщение об ошибке?

Можно использовать SetErrorMode() для предотвращения подобных сообщений.

```
function IsDriveReady(DriveLetter: char): bool;
var
  OldErrorMode: Word;
  OldDirectory: string;
begin
  OldErrorMode := SetErrorMode(SEM_NOOPENFILEERRORBOX);
  GetDir(0, OldDirectory);
{$I-}
  ChDir(DriveLetter + ':\');
{$I+}
  if IoResult <> 0 then Result := False
  else Result := True;
  ChDir(OldDirectory);
  SetErrorMode(OldErrorMode);
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  if not IsDriveReady('A') then ShowMessage('Drive Not Ready')
  else ShowMessage('Drive is Ready');
end;
```

Как получить длинное имя файла по короткому?

Обратитесь к Win32_Find_Data компонента TSearchRec:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  SearchRec: TSearchRec;
  Success: integer;
begin
  Success := SysUtils.FindFirst('C:\window~1.bmk', faAnyFile, SearchRec);
```

```
  if Success = 0 then begin
    ShowMessage(SearchRec.FindData.CFileName);
  end;
  SysUtils.FindClose(SearchRec);
end;
```

Как узнать дату и время последнего доступа к файлу?

Пример получает требуемые сведения. Только учтите, что не все файловые системы поддерживают эту возможность.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  SearchRec: TSearchRec;
  Success: integer;
  DT: TFileTime;
  ST: TSystemTime;
begin
  Success := SysUtils.FindFirst('C:\autoexec.bat', faAnyFile, SearchRec);
  if (Success = 0) and ((SearchRec.FindData.ftLastAccessTime.dwLowDateTime <> 0)
      or (SearchRec.FindData.ftLastAccessTime.dwHighDateTime <> 0)) then begin
    FileTimeToLocalFileTime(SearchRec.FindData.ftLastAccessTime, DT);
    FileTimeToSystemTime(DT, ST);
    Memo1.Lines.Clear;
    Memo1.Lines.Add('AutoExec.Bat was last accessed at:');
    Memo1.Lines.Add('Year := ' + IntToStr(st.wYear));
    Memo1.Lines.Add('Month := ' + IntToStr(st.wMonth));
    Memo1.Lines.Add('DayOfWeek := ' + IntToStr(st.wDayOfWeek));
    Memo1.Lines.Add('Day := ' + IntToStr(st.wDay));
    Memo1.Lines.Add('Hour := ' + IntToStr(st.wHour));
    Memo1.Lines.Add('Minute := ' + IntToStr(st.wMinute));
    Memo1.Lines.Add('Second := ' + IntToStr(st.wSecond));
    Memo1.Lines.Add('Milliseconds := ' + IntToStr(st.wMilliseconds));
  end;
  SysUtils.FindClose(SearchRec);
end;
```

Как вывести стандартное окно копирования файлов?

Используйте `SHFileOperation`.

```
uses ShellAPI;

procedure TForm1.Button1Click(Sender: TObject);
var
  Fo: TSHFileOpStruct;
  buffer: array[0..4096] of char;
  p: pchar;
begin
  FillChar(Buffer, SizeOf(Buffer), #0);
  p := @buffer;
  p := StrECopy(p, 'C:\DownLoad\1.ZIP') + 1;
  p := StrECopy(p, 'C:\DownLoad\2.ZIP') + 1;
  p := StrECopy(p, 'C:\DownLoad\3.ZIP') + 1;
```

```
  StrECopy(p, 'C:\DownLoad\4.ZIP');
  FillChar(Fo, SizeOf(Fo), #0);
  Fo.Wnd := Handle;
  Fo.wFunc := FO_COPY;
  Fo.pFrom := @Buffer;
  Fo.pTo := 'D:\';
  Fo.fFlags := 0;
  if ((SHFileOperation(Fo) <> 0) or (Fo.fAnyOperationsAborted <> false)) then
    ShowMessage('Cancelled')
end;
```

Как определить каталог, из которого запущено приложение?

Следующая функция извлекает путь из свойства ExeName глобального объекта Application:

```
function GetExePath: String;
var
  LastBackSlashPos, Index: Integer;
begin
  Result := Application.ExeName;
  for Index := 1 to length(Result) do
    if Result[Index] = '\' then
      LastBackSlashPos := Index;
{ вычитаем 1 для исключения последней косой черты }
  Result := Copy(Result, 1, LastBackSlashPos - 1);
end;
```

Примечание

Лучше просматривать строку с конца. До последней косой черты оттуда ближе.

Почему при вызове диалогового окна, например, MessageBox() из обработчика события OnExit, мерцающий курсор исчезает при закрытии диалога?

Это стандартное поведение Windows. Форсирование дополнительного изменения фокуса (например, с помощью ShowMessage) во время обработки события, связанного с фокусом (например, OnExit), может вызвать непредсказуемые действия со стороны Windows. Это Windows, которая управляет передачей фокуса, и это Windows, которая может вызвать неадекватную реакцию, если принудительно изменить фокус в середине этого процесса.

Можно ли определить полный путь и имя файла запущенной DLL из самой DLL?

```
procedure ShowDllPath; stdcall;
var
  TheFileName: array[0..MAX_PATH] of char;
begin
  FillChar(TheFileName, SizeOf(TheFileName), #0);
  GetModuleFileName(hInstance, TheFileName, SizeOf(TheFileName));
  MessageBox(0, TheFileName, 'The DLL file name is:', mb_ok);
end;
```

Как получить доступ к файлам, которые пользователь перетаскивает на форму с помощью мыши?

Пример с применением Windows API показывает, как осуществлять такие действия, добавляя имена файлов в Memo.

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm1 = class(TForm)
    Memo1: TMemo;
    procedure FormCreate(Sender: TObject);
  private
    procedure WMDROPFILES(var Message: TWMDROPFILES); message WM_DROPFILES;
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

uses ShellApi;

procedure TForm1.FormCreate(Sender: TObject);
begin
{ Let Windows know we accept dropped files }
  DragAcceptFiles(Form1.Handle, True);
end;

procedure TForm1.WMDROPFILES(var Message: TWMDROPFILES);
var
  NumFiles: longint;
  i: longint;
  buffer: array[0..255] of char;
begin
{ How many files are being dropped }
  NumFiles := DragQueryFile(Message.Drop, $FFFFFFFF, nil, 0);
{ Accept the dropped files }
  for i := 0 to (NumFiles - 1) do begin
    DragQueryFile(Message.Drop, i, @buffer, sizeof(buffer));
    Form1.Memo1.Lines.Add(buffer);
  end;
end;

end.
```

Как при запуске Windows Explorer открыть определенную папку?

Надо ввести в командную строку имя этой самой папки. Программно это делается приблизительно так:

```
uses
  ShellApi;

procedure TForm1.Button1Click(Sender: TObject);
begin
  ShellExecute(0, 'explore', 'C:\WINDOWS', nil, nil, SW_SHOWNORMAL);
end;
```

Как сбросить на диск кэшированную информацию о бинарном файле?

```
procedure TForm1.Button1Click(Sender: TObject);
var
  f: file;
  i: integer;
begin
  i := 10;
  AssignFile(f, 'C:\DownLoad\Test.Bin');
  ReWrite(f, 1);
  BlockWrite(f, i, SizeOf(i));
  FlushFileBuffers(TFileRec(f).Handle);
  CloseFile(f);
end;
```

Как печатать в Delphi без TPrinter?

Пример показывает, как с помощью `PrintDlg()` дать возможность пользователю выбрать принтер и напечатать две страницы.

```
uses CommDlg;

{$IFNDEF WIN32}
const MAX_PATH = 144;
{$ENDIF}

procedure TForm1.Button1Click(Sender: TObject);
var
  Pd: TPrintDlg;
  DocInfo: TDocInfo;
begin
  FillChar(Pd, SizeOf(Pd), #0);
  Pd.lStructSize := SizeOf(Pd);
  Pd.hWndOwner := Form1.Handle;
  Pd.Flags := PD_RETURNDC;
  if PrintDlg(pd) then begin
    FillChar(DocInfo, SizeOf(DocInfo), #0);
    DocInfo.cbSize := SizeOf(DocInfo);
    GetMem(DocInfo.lpszDocName, 32);
    GetMem(DocInfo.lpszOutput, MAX_PATH);
    lStrCpy(DocInfo.lpszDocName, 'My Document');
{ Add this line to print to a file }
    lStrCpy(DocInfo.lpszOutput, 'C:\Download\Test.doc');
    StartDoc(Pd.hDc, DocInfo);
      StartPage(Pd.hDc);
        TextOut(Pd.hDc, 100, 100, 'Page 1', 6);
```

```
        EndPage(Pd.hDc);
        StartPage(Pd.hDc);
          TextOut(Pd.hDc, 100, 100, 'Page 2', 6);
        EndPage(Pd.hDc);
      EndDoc(Pd.hDc);
      FreeMem(DocInfo.lpszDocName, 32);
      FreeMem(DocInfo.lpszOutput, MAX_PATH);
    end;
  end;
```

Как получить поддерживаемое принтером разрешение в пикселах на дюйм?

Для получения величины в пикселах можно воспользоваться API-функцией `GetDeviceCaps()` **(используем модуль** `Printers`**):**

```
VertPixelsPerInch := GetDeviceCaps(Printer.Handle, LogPixelsX);
HorzPixelsPerInch := GetDeviceCaps(Printer.Handle, LogPixelsY);
```

Как «сказать» принтеру, чтобы он печатал не в графическом режиме, а использовал встроенные шрифты?

```
uses Printers;

procedure TForm1.Button1Click(Sender: TObject);
var
  tm: TTextMetric;
  i: integer;
begin
  if PrintDialog1.Execute then begin
    Printer.BeginDoc;
    Printer.Canvas.Font.Handle := GetStockObject(DEVICE_DEFAULT_FONT);
    GetTextMetrics(Printer.Canvas.Handle, tm);
    for i := 1 to 10 do begin
      Printer.Canvas.TextOut(100, i * tm.tmHeight + tm.tmExternalLeading, 'Test');
    end;
    Printer.EndDoc;
  end;
end;
```

Как послать строку raw-формата на принтер?

В Win16 можно вызвать функцию `SpoolFile`, а в Win32 – функцию `WritePrinter`.

Пример показывает, как открыть принтер и послать строку `raw`. Имейте в виду, что необходимо ввести правильное имя принтера и посылать принтеру правильные управляющие коды.

```
uses WinSpool;

procedure WriteRawStringToPrinter(PrinterName: String; S: String);
var
  Handle: THandle;
  N: DWORD;
  DocInfo1: TDocInfo1;
```

```
begin
  if not OpenPrinter(PChar(PrinterName), Handle, nil) then begin
  ShowMessage('Error ' + IntToStr(GetLastError));
  Exit;
  end;
  with DocInfo1 do begin
    pDocName := PChar('test doc');
    pOutputFile := nil;
    pDataType := 'RAW';
  end;
  StartDocPrinter(Handle, 1, @DocInfo1);
  StartPagePrinter(Handle);
  WritePrinter(Handle, PChar(S), Length(S), N);
  EndPagePrinter(Handle);
  EndDocPrinter(Handle);
  ClosePrinter(Handle);
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  WriteRawStringToPrinter('HP', 'Test This');
end;
```

Можно ли программно закрыть лоток CD-ROM?

Используем `mciSendCommand`, передавая `MCI_SET_DOOR_CLOSED` команду.

```
uses MMSystem;

function CloseCD(Drive: char): boolean;
var
  mp: TMediaPlayer;
begin
  result := false;
  Application.ProcessMessages;
  mp := TMediaPlayer.Create(nil);
  mp.Visible := false;
  mp.Parent := Application.MainForm;
  mp.Shareable := true;
  mp.DeviceType := dtCDAudio;
  mp.FileName := Drive + ':';
  mp.Open;
  Application.ProcessMessages;
  mciSendCommand(mp.DeviceID, MCI_SET, MCI_SET_DOOR_CLOSED, 0);
  Application.ProcessMessages;
  mp.Close;
  Application.ProcessMessages;
  mp.free;
  result := true;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  CloseCD('H');
end;
```

Может ли TMediaPlayer в Delphi проигрывать AVI-файлы?

Delphi умеет воспроизводить с помощью TMediaPlayer AVI-файлы, необходимо всего лишь установить тип файла и его имя, разместить компонент на TPanel и вызвать метод Open. Для воспроизведения AVI-файлов необходимо, чтобы в системе был установлен пакет Video for Windows. При выполнении этих условий воспроизведение файлов AVI будет столь же легким, как и воспроизведение файлов WAV. Присвойте типу устройства AutoSelect, поместите компонент на панель, задайте имя AVI-файла и воспроизведите его. Анимация появится в отдельном небольшом окне.

Как воспроизвести AVI в полноэкранном режиме?

Проиграйте его на другой форме, развернутой на весь экран.

```
{ Code for Form 1 }

uses Unit2;

procedure TForm1.Button1Click(Sender: TObject);
begin
  Form2.Show;
  Form2.WindowState := wsMaximized;
  Form2.MediaPlayer1.Notify := false;
  Form2.MediaPlayer1.Display := Form2.Panel1;
  Form2.MediaPlayer1.FileName := 'C:\TheWall\DELCAR2.AVI';
  Form2.MediaPlayer1.Open;
  Form2.MediaPlayer1.DisplayRect := Form2.ClientRect;
  Form2.MediaPlayer1.Play;
end;

{ Code for Form 2 }

procedure TForm2.MediaPlayer1Notify(Sender: TObject);
begin
  if MediaPlayer1.NotifyValue = nvSuccessful then Form2.Close;
end;
```

Определение дорожки CD

Создаем таймер и помещаем данный код в обработчик события OnTimer:

```
uses MMSystem;

procedure TForm1.Timer1Timer(Sender: TObject);
var
  Trk, Min, Sec: Word;
begin
  with MediaPlayer1 do begin
    Trk := MCI_TMSF_TRACK(Position);
    Min := MCI_TMSF_MINUTE(Position);
    Sec := MCI_TMSF_SECOND(Position);
    Label1.Caption := Format('%.2d', [Trk]);
    Label2.Caption := Format('%.2d:%.2d', [Min,Sec]);
  end;
end;
```

Хочу использовать функцию Beep(), но получаю ошибку «Too many parameters».

Все дело в том, что Beep определена и в SysUtils, и в Windows. Посмотрите на пример, наиболее просто разъясняющий применение этих двух функций.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  SysUtils.Beep;
  Windows.Beep(100, 1000);
end;
```

Как поменять приоритет приложения?

```
procedure TForm1.Button1Click(Sender: TObject);
var
  ProcessID: DWORD;
  ProcessHandle: THandle;
  ThreadHandle: THandle;
begin
  ProcessID := GetCurrentProcessID;
  ProcessHandle := OpenProcess(PROCESS_SET_INFORMATION, false, ProcessID);
  SetPriorityClass(ProcessHandle, REALTIME_PRIORITY_CLASS);
  ThreadHandle := GetCurrentThread;
  SetThreadPriority(ThreadHandle, THREAD_PRIORITY_TIME_CRITICAL);
end;
```

Можно ли программно закрыть какое-либо приложение?

Да, надо послать сообщение WM_QUIT его окну.

```
PostMessage(FindWindow(Nil, 'window caption'), WM_QUIT, 0, 0);
```

window caption – заголовок окна, которое нужно закрыть.

Каков порядок наступления событий при создании и показе формы?

При создании формы события наступают в следующем порядке: OnCreate, OnShow, OnPaint, OnActivate, OnResize и снова OnPaint.

Как создать новую форму, которая бы не отбирала фокус у существующей?

```
uses Unit2;

procedure TForm1.Button1Click(Sender: TObject);
begin
  Form2 := TForm2.Create(Application);
  Form2.Visible := False;
  ShowWindow(Form2.Handle, SW_SHOWNA);
end;
```

Почему, если в обработчике формы OnActivate изменить свойство FormStyle, возникает ошибка «Cannot change Visible in OnShow or OnHide» (не могу изменить Visible (видимость) в OnShow или OnHide)?

Свойство FormStyle определяет стиль создаваемого окна и обычно устанавливается в обработчике события OnCreate, тем не менее, вы можете его изменить

и после создания дескриптора окна, только не в обработчиках событий OnActivate, OnShow или OnHide. Проблема заключается в том, что попытка изменить стиль формы предпринимается при возникновении событий OnShow и OnHide.

Как запретить кнопки Закрыть/Открыть системного меню окна?

Смотрим пример. Необходимо учитывать возможность разворачивания окна по двойному щелчку мыши на заголовке окна и т. п., подбирая соответствующий тип окна.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
{ Disable }
  Form1.BorderIcons := Form1.BorderIcons - [biSystemMenu, biMinimize, biMaximize];
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
{ Enable }
  Form1.BorderIcons := Form1.BorderIcons + [biSystemMenu, biMinimize, biMaximize];
end;
```

Можно ли реагировать на разворачивание/сворачивание окна до того, как это произойдет?

Перехватывайте WM_SYSCOMMAND.

```
type
  TForm1 = class(TForm)
  private
    procedure WMSysCommand(var Msg: TWMSysCommand); message WM_SYSCOMMAND;
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.WMSysCommand;
begin
  if (Msg.CmdType = SC_MINIMIZE) or (Msg.CmdType = SC_MAXIMIZE) then MessageBeep(0)
  else inherited;
end;
```

Как предотвратить перемещение или изменение размеров формы?

Перехватывайте WM_WINDOWPOSCHANGING и соответствующие флаги SWP_NOMOVE и SWP_NOSIZE.

```
type
  TForm1 = class(TForm)
  private
    procedure WMPosChange(var Message: TWMWINDOWPOSCHANGING);
                          message WM_WINDOWPOSCHANGING;
  end;
```

```
var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.WMPosChange(var Message: TWMWINDOWPOSCHANGING);
begin
  PWindowPos(TMessage(Message).lParam).Flags :=
      PWindowPos(TMessage(Message).lParam).Flags or SWP_NOMOVE or SWP_NOSIZE;
end;
```

Как отследить окончание изменения размера или перемещения окна?

Если вы попробуете отслеживать WM_SIZE или WM_MOVE, то получите множество таких сообщений во время перемещения окна. Код демонстрирует пример перехвата события WM_EXITSIZEMOVE, которое хоть и документировано только для применения в Windows NT, но работает и в Windows 95. Учтите, что можно использовать WM_ENTERSIZEMOVE для отлавливания начала изменения размера/перемещения.

```
type
  TForm1 = class(TForm)
  public
    procedure WMEXITSIZEMOVE(var Message: TMessage); message WM_EXITSIZEMOVE;
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.WMEXITSIZEMOVE(var Message: TMessage);
begin
  Form1.Caption := 'Finished Moving or Sizing';
end;
```

Как разместить форму в центре экрана?

Используйте вместо оператора / оператор div:

```
MyLeft := (Screen.Left - Form1.Left) div 2;
```

Или присвойте свойству Position формы значение poScreenCenter.

Как получить дескриптор Панели задач (TaskBar)?

Очень просто:

```
hTaskbar := FindWindow('Shell_TrayWnd', Nil);
```

Как спрятать приложение из Панели задач (TaskBar)?

Зайдите в меню View|Code Explorer, добавьте в uses модуль Windows, после Application.Initialize впишите Application.ShowMainForm := False; затем перед Application.Run впишите ShowWindow(Application.Handle, SW_HIDE);.

Ваш проект должен выглядеть приблизительно так:

```
program Project1;

uses
  Windows,
  Forms,
  Unit1 in 'Unit1.pas' {Form1},
  Unit2 in 'Unit2.pas' {Form2};

{$R *.RES}

begin
  Application.Initialize;
  Application.ShowMainForm := False;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm2, Form2);
  ShowWindow(Application.Handle, SW_HIDE);
  Application.Run;
end.
```

В конце каждого initialization модулей, использующих формы, надо вписать ShowWindow(Application.Handle, SW_HIDE).

Как определить размер Рабочего стола за вычетом Панели задач (Desktop-TaskBar)?

```
procedure TForm1.Button1Click(Sender: TObject);
var
  r: TRect;
begin
  SystemParametersInfo(SPI_GETWORKAREA, 0, @r, 0);
  Memo1.Lines.Add(IntToStr(r.Top));
  Memo1.Lines.Add(IntToStr(r.Left));
  Memo1.Lines.Add(IntToStr(r.Bottom));
  Memo1.Lines.Add(IntToStr(r.Right));
end;
```

Как спрятать Панель задач?

Надо найти дескриптор окна Панели задач и передать в функции ShowWindow параметр SW_HIDE.

В примере Панель задач сначала скрывается, потом отображается.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  hTaskBar: THandle;
begin
  hTaskbar := FindWindow('Shell_TrayWnd', Nil);
  ShowWindow(hTaskBar, SW_HIDE);
end;

procedure TForm1.Button2Click(Sender: TObject);
var
  hTaskBar: THandle;
begin
  hTaskbar := FindWindow('Shell_TrayWnd', Nil);
```

```
    ShowWindow(hTaskBar, SW_SHOWNORMAL);
  end;
```

Как создать приложение, которое работало бы в Системной панели (Systray)?

На самом деле приложение не работает в «трее» и даже не сворачивается туда, как думают очень многие, задающие этот вопрос. Хитрость заключается в следующем – приложение скрывается из Панели задач, вместо этого в «трей» (systray) помещается созданная произвольная пиктограмма. Усложняется все это различными разворачиваниями/сворачиваниями окна. Не более.

Пример показывает структуру самой этой процедуры. Для повседневной работы лучше все-таки написать свой компонент либо использовать существующие, например из пакета компонентов RxLIB (rxTrayIcon):

```
{TrayIt.dpr}

program TrayIt;

uses
  Windows,
  Forms,
  TrayIt1 in 'TrayIt1.pas' {Form1};

{$R *.RES}

begin
  Application.Initialize;
  Application.ShowMainForm := False;
  Application.CreateForm(TForm1, Form1);
  ShowWindow(Application.Handle, SW_HIDE);
  Application.Run;
end.

{ TrayIt1.pas }

unit TrayIt1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, Menus,
  ShellAPI, ExtCtrls;

type
  TForm1 = class(TForm)
    PopupMenu1: TPopupMenu;
    Open1: TMenuItem;
    Exit1: TMenuItem;
    Timer1: TTimer;
    procedure FormCreate(Sender: TObject);
    procedure Open1Click(Sender: TObject);
    procedure Exit1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Timer1Timer(Sender: TObject);
  private
    procedure WndProc(var Msg: TMessage); override;
```

```
  public
    IconData : TNotifyIconData;
    IconCount : integer;
end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.WndProc(var Msg: TMessage);
var
  p: TPoint;
begin
  case Msg.Msg of
      WM_USER + 1:
          case Msg.lParam of
              WM_RBUTTONDOWN:  begin
                                 GetCursorPos(p);
                                 PopupMenu1.Popup(p.x, p.y);
                               end;
          end;
  end;
  inherited;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  BorderIcons := [biSystemMenu];
  IconCount := 0;
  IconData.cbSize := SizeOf(IconData);
  IconData.Wnd := Handle;
  IconData.uID := 100;
  IconData.uFlags := NIF_MESSAGE + NIF_ICON + NIF_TIP;
  IconData.uCallbackMessage := WM_USER + 1;
  IconData.hIcon := Application.Icon.Handle;
  StrPCopy(IconData.szTip, Application.Title);
  Shell_NotifyIcon(NIM_ADD, @IconData);
  Timer1.Interval := 1000;
  Timer1.Enabled := true;
end;

procedure TForm1.Open1Click(Sender: TObject);
begin
  Form1.Show;
  ShowWindow(Application.Handle, SW_HIDE);
end;

procedure TForm1.Exit1Click(Sender: TObject);
begin
  Shell_NotifyIcon(NIM_DELETE, @IconData);
  Application.ProcessMessages;
  Application.Terminate;
end;
```

```
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action := caNone;
  Form1.Hide;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  case(IconCount) of
      0:  IconData.hIcon := LoadIcon(0, IDI_APPLICATION);
      1:  IconData.hIcon := LoadIcon(0, IDI_ASTERISK);
      2:  IconData.hIcon := LoadIcon(0, IDI_EXCLAMATION);
      3:  IconData.hIcon := LoadIcon(0, IDI_HAND);
      4:  IconData.hIcon := LoadIcon(0, IDI_QUESTION);
      5:  IconData.hIcon := Application.Icon.Handle;
  end;
  inc(IconCount);
  if IconCount > 5 then IconCount := 0;
  Application.Title := TimeToStr(Now);
  StrPCopy(IconData.szTip, Application.Title);
  Shell_NotifyIcon(NIM_MODIFY, @IconData);
end;

initialisation
  ShowWindow(Application.Handle, SW_HIDE);

end.
```

Как привязать кнопку к нижней границе окна так, чтобы при изменении размеров окна расстояние между ней и нижней границей окна не изменялось?

Создайте `TPanel`, установите свойство `Alignment` в `Bottom` и расположите на ней кнопку. Если необходимо, чтобы панель не была видима на форме, сбросьте флаг `Ctl3d`, установите `bevels` в `none` и `ParentColor` равным `True`.

Примечание

Начиная с Delphi 4 у компонентов есть специальное свойство `Anchors`*, отвечающее за привязку элементов при изменении размера. В данном случае достаточно добавить в множество значений этого свойства значение* `akBottom`.

Как получить дескриптор какого-либо определенного окна и сделать его активным?

В случае если вам известен точный заголовок окна, лучше всего использовать `FindWindow()`. Но, к сожалению, часто приходится искать окно, зная только часть заголовка. Например «Microsoft Word – Some Unknown Doc». В данном случае можно вызвать функцию `EnumWindows()`, перебрать все окна и, просматривая заголовки как строки, сравнивать их со строкой поиска.

Пример:

```
type
  PFindWindowStruct = ^TFindWindowStruct;
  TFindWindowStruct = record
```

```
    Caption: string;
    ClassName: string;
    WindowHandle: THandle;
  end;

function EnumWindowsProc(hWindow: hWnd; lParam: LongInt): Bool
         {$IFDEF Win32} stdcall; {$ELSE}; export; {$ENDIF}
var
  lpBuffer: PChar;
  WindowCaptionFound: bool;
  ClassNameFound: bool;
begin
  GetMem(lpBuffer, 255);
  Result := True;
  WindowCaptionFound := False;
  ClassNameFound := False;
  try
    if GetWindowText(hWindow, lpBuffer, 255) > 0 then
      if Pos(PFindWindowStruct(lParam).Caption, StrPas(lpBuffer)) > 0 then
        WindowCaptionFound := true;
      if PFindWindowStruct(lParam).ClassName = '' then
        ClassNameFound := True
      else if GetClassName(hWindow, lpBuffer, 255) > 0 then
      if Pos(PFindWindowStruct(lParam).ClassName, StrPas(lpBuffer)) > 0 then
          ClassNameFound := True;
      if (WindowCaptionFound and ClassNameFound) then begin
        PFindWindowStruct(lParam).WindowHandle := hWindow;
        Result := False;
        end;
  finally
    FreeMem(lpBuffer, sizeof(lpBuffer^));
  end;
end;

function FindAWindow(Caption: string; ClassName: string): THandle;
var
  WindowInfo: TFindWindowStruct;
begin
  with WindowInfo do begin
    Caption := Caption;
    ClassName := ClassName;
    WindowHandle := 0;
    EnumWindows(@EnumWindowsProc, LongInt(@WindowInfo));
    FindAWindow := WindowHandle;
  end;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  TheWindowHandle: THandle;
begin
  TheWindowHandle := FindAWindow('Netscape - ', '');
  if TheWindowHandle = 0 then ShowMessage('Window Not Found!')
```

```
    else BringWindowToTop(TheWindowHandle);
  end;
```

Можно ли создать прозрачную форму?

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Form1.Brush.Style := bsClear;
  Form1.BorderStyle := bsNone
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  Application.Terminate;
end;
```

Как совершить некоторые действия после того, как форма будет нормально отображена? Похоже, что все события формы (OnCreate, OnPaint и др.) происходят еще до того, как форма становится видимой.

В обработчик события `OnPaint` надо добавить `Visible := True`, после чего можно совершать любые действия.

Как создать «модальную» форму в MDI-приложении? Когда мое приложение вызывает метод ShowModal, генерируется сообщение «Can't Show Modal when Visible is set true» (не могу показать в модальном режиме, поскольку свойство видимости установлено в True). При попытке установить Visible в False я получаю ошибку с сообщением о том, что на дочерней MDI-форме нельзя установить свойство Visible в False.

По технологии первая форма проекта (главная форма) создается со свойством `Visible`, установленным в `True`, а все остальные формы при создании имеют свойство `Visible`, установленное в `False`. С другой стороны, дочерняя MDI-форма не может быть невидимой, поэтому ее свойство `Visible` установлено в `True`. При изменении стиля формы на `fsNormal` свойство `Visible` имеет значение `True`, поэтому в `False` его необходимо установить вручную.

Почему я получаю исключительную ситуацию «EInvalidOperation: Cannot make a visible window modal» (не могу сделать модальное окно видимым), если открываю форму с помощью метода ShowModal? Форма при этом не открывается.

Убедитесь, что свойство формы `Visible` не устанавливается в `True` во время проектирования или во время выполнения программы.

Как сообщить активному дочернему MDI-окну о том, что была нажата кнопка Сохранить на панели управления родительского окна? Вызов TEditForm.Save1Click не работает.

Попробуйте сделать так:

```
with Application.MainForm.ActiveMDIChild as TEditForm do Save1Click(Sender);
```

или так:

```
if ActiveMDIChild is TEditForm then TEditForm(ActiveMDIChild).Save1Click(Sender);
```

Расходует ли Delphi системные ресурсы при открытии и закрытии модальных окон? Например, следующий код уменьшает системные ресурсы при каждом отображении модального диалога:

```
ModalForm1 := TModalForm1.Create(Self);
ModalForm1.ShowModal;
```

В отсутствие обработчика OnClose установите параметр Action в caFree, ваш код создает каждую новую форму вызовом TModalForm1.Create(Self), но предыдущий экземпляр формы не уничтожается. Все предыдущие экземпляры форм ModalForm1 так и остаются в Windows.

Для того чтобы форма освобождала свои ресурсы, можно также использовать метод Free. Вот демонстрация этого метода:

```
  try
    ModalForm1.ShowModal;
{ здесь размещается необходимый код }
  finally
    ModalForm1.Free;
  end;
```

Во время выполнения программы каждое вновь открытое дочернее окно отображается немного ниже и правее предыдущего. Проблема заключается в том, что когда я закрываю какое-либо дочернее окно и открываю новое, оно появляется правее и ниже того, которое я закрыл перед этим, даже если оно было единственным. Какие правила тут действуют?

Так работают дочерние MDI-окна. В этой ситуации VCL не перекрывает поведение Windows, такие правила диктует сама система.

В процедуре FormCreate установите необходимые значения для свойств Top, Left, Width и Height. FormCreate дочерней MDI-формы будет вызвана прежде, чем будет показано само окно.

Как сделать глобальную переменную видимой всем (в том числе скрытым) окнам программы?

Задача решается рассылкой пользовательского сообщения всем окнам массива Screen.Forms:

```
{Code for Unit1}

const
  UM_MyGlobalMessage = WM_USER + 1;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Button1: TButton;
    procedure FormShow(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
```

```
procedure UMMyGlobalMessage(var AMessage: TMessage);
                                message UM_MyGlobalMessage;
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

uses Unit2;

procedure TForm1.FormShow(Sender: TObject);
begin
  Form2.Show;
end;

procedure TForm1.UMMyGlobalMessage(var AMessage: TMessage);
begin
  Label1.Left := AMessage.WParam;
  Label1.Top := AMessage.LParam;
  Form1.Caption := 'Got It!';
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  f: integer;
begin
  for f := 0 to Screen.FormCount - 1 do
    Screen.Forms[f].Perform(UM_MyGlobalMessage, 42, 42);
end;

{ Code for Unit2 }

const
  UM_MyGlobalMessage = WM_USER + 1;

type
  TForm2 = class(TForm)
    Label1: TLabel;
  private
    procedure UMMyGlobalMessage(var AMessage: TMessage);
                                message UM_MyGlobalMessage;
  end;
var
  Form2: TForm2;

implementation

{$R *.DFM}

procedure TForm2.UMMyGlobalMessage(var AMessage: TMessage);
begin
  Label1.Left := AMessage.WParam;
  Label1.Top := AMessage.LParam;
  Form2.Caption := 'Got It!';
end;
```

Как определить, нажаты ли клавиши <Shift>, <Control>, <Alt>?

```
function CtrlDown: Boolean;
var
  State: TKeyboardState;
begin
  GetKeyboardState(State);
  Result := ((State[vk_Control] and 128) <> 0);
end;

function ShiftDown: Boolean;
var
  State: TKeyboardState;
begin
  GetKeyboardState(State);
  Result := ((State[vk_Shift] and 128) <> 0);
end;

function AltDown: Boolean;
var
  State: TKeyboardState;
begin
  GetKeyboardState(State);
  Result := ((State[vk_Menu] and 128) <> 0);
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  if ShiftDown then Form1.Caption := 'Shift'
  else Form1.Caption := '';
end;
```

Поле редактирования автоматически отрабатывает комбинации <Ctrl>+<C>, <Ctrl>+<X>, <Ctrl>+<Y> и <Ctrl>+<Z> соответственно для копирования, вырезания, вставки и отката. Как мне показать эти опции в главном меню и, самое главное, связать с надлежащими действиями?

Попробуйте следующие обработчики событий:

```
procedure TForm1.SpeedButtonCutToClipBoardClick(Sender: TObject);
begin
  if ((ActiveControl) is TCustomEdit) then TEdit(ActiveControl).CutToClipBoard;
end;

procedure TForm1.Delete1Click(Sender: TObject);
begin
  if ((ActiveControl) is TCustomEdit) then TEdit(ActiveControl).ClearSelection;
end;
```

В обоих случаях необходимо определять, какой элемент управления активен в настоящий момент, и вызывать соответствующий случаю метод. Все элементы управления, позволяющие редактировать текст, являются наследниками `TCustomEdit`. Приведенный выше код определяет, является ли активный элемент наследником `TCustomEdit`, и если да, вызывает соответствующий метод.

Существует ли функция прокрутки формы с помощью клавиш? Например, прокрутка вверх и вниз при нажатии <PgUp> и <PgDown> соответственно.

Прокрутка формы осуществляется с помощью изменения свойства Position свойства формы VertScrollbar или HorzScrollbar. Следующий код показывает, как это можно сделать:

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
const
  PageDelta = 10;
begin
  with VertScrollbar do
    if Key = VK_NEXT then Position := Position + PageDelta
    else if Key = VK_PRIOR then Position := Position - PageDelta;
end;
```

Примечание

Этот код может не работать, если активный элемент управления (типа TMemo*) также использует <PgUp> и <PgDn>.*
Свойство KeyPreview *формы устанавливаем в* True.

Почему, когда пользователь нажимает кнопку SpeedButton, на компоненте TEdit не возникает событие OnExit? Существует ли способ заставить генерироваться событие OnExit при нажатии кнопки SpeedButton?

SpeedButton в действительности никогда не получает фокуса. Поэтому активный элемент управления его и не теряет, следовательно, на активном элементе управления событие OnExit не генерируется.

Имеется единственный способ вызвать событие OnExit на активном элементе управления – явно при наступлении события OnClick кнопки SpeedButton. Например:

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
  if ActiveControl is TEdit then
    (ActiveControl as TEdit).OnExit(ActiveControl);
end;
```

Как задать порядок обхода элементов управления клавишей <Tab>?

Выберите на форме необходимые компоненты и воспользуйтесь пунктом меню Edit|Tab Order.

Некоторые компоненты не реагируют на изменения курсора, пока пользователь не передвинет мышь. Можно ли эмулировать перемещение мыши?

Пример демонстрирует возможности управления мышью.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  pt: TPoint;
```

```
begin
  Application.ProcessMessages;
  Screen.Cursor := CrHourglass;
  GetCursorPos(pt);
  SetCursorPos(pt.x + 1, pt.y + 1);
  Application.ProcessMessages;
  SetCursorPos(pt.x - 1, pt.y - 1);
end;
```

Почему некоторые визуальные компоненты типа TPanel и TEdit не имеют свойства Canvas?

Все наследники TCustomControl имеют свойство Canvas, тем не менее, необходим какой-то механизм защиты для того, чтобы другие «художники» не могли рисовать на компоненте. Наследники компонента всегда имеют доступ к защищенным свойствам, которые они наследуют от компонента (как, например, Canvas), но те же пользователи компонента к ним доступа не имеют.

```
type
  TCanvasPanel = class(TPanel)
  public
    property Canvas;
  end;
```

Для того чтобы рисовать на компоненте, не имеющем опубликованного свойства Canvas, попробуйте взять другой компонент, позволяющий рисовать на нем (TPaintBox). Или компоненты-слои, обеспечивающие тот же результат (client-align у TPaintBox внутри TPanel позволяет получить оконтуренную область с возможностью рисования).

Как поместить прозрачный текст на Canvas TBitmap?

```
procedure TForm1.Button1Click(Sender: TObject);
var
  OldBkMode: integer;
begin
  Image1.Picture.Bitmap.Canvas.Font.Color := clBlue;
  OldBkMode := SetBkMode(Image1.Picture.Bitmap.Canvas.Handle, TRANSPARENT);
  Image1.Picture.Bitmap.Canvas.TextOut(10, 10, 'Hello');
  SetBkMode(Image1.Picture.Bitmap.Canvas.Handle, OldBkMode);
end;
```

Как нарисовать линию?

Соединяем точки с координатами 10,10; 100,100 дугой, имеющей радиус 10.

```
procedure DDAProc(x: integer; y: integer; LParam: LongInt)
                 {$IFDEF WIN32} stdcall; {$ELSE}; export; {$ENDIF}
begin
  if x mod 10 = 0 then
    TCanvas(LParam).Ellipse(x - 10, y - 10, x + 10, y + 10);
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  LineDDA(10, 10, 100, 100, @DDAProc, {$IFDEF WIN32} LongInt(Form1.Canvas)
{$ELSE} Form1.Canvas {$ENDIF} );
end;
```

Как сделать границы компонента углубленными (sunken) или выпуклыми (raised)?

Для того чтобы заставить компонент выглядеть чуть утопленным или приподнятым, разместите его на компоненте `TBevel` или `TPanel`, которые имеют свойства соответствующего назначения.

Как проще перемещать компонент во время работы программы?

В примере показано, как перемещать компонент при перетаскивании (нажаты левая кнопка мыши и клавиша <Ctrl>).

```
procedure TForm1.Button1MouseDown(Sender: TObject; Button: TMouseButton;
                                  Shift: TShiftState; X, Y: Integer);
{$IFNDEF WIN32}
var
  pt: TPoint;
{$ENDIF}

begin
  if ssCtrl in Shift then begin
    ReleaseCapture;
    SendMessage(Button1.Handle, WM_SYSCOMMAND, 61458, 0);
{$IFNDEF WIN32}
    GetCursorPos(pt);
    SendMessage(Button1.Handle, WM_LBUTTONUP, MK_CONTROL, Longint(pt));
{$ENDIF}
  end;
end;
```

Можно ли получить доступ к компонентам по их имени (например, "SpeedButton" + IntToStr(i))?

Да, это возможно. Следующий пример использует метод `FindComponent` формы `Form1` для выключения первых 10 компонентов `SpeedButton` по их имени.

```
for i := 1 to 10 do
  with Form1.FindComponent('SpeedButton' + IntToStr(i)) as TSpeedButton do
    Enabled := False;
```

Как в BitBtn разместить текст в нескольких строках?

Смотрим пример, помещающий надпись прямо на кнопку.

```
procedure TForm1.FormCreate(Sender: TObject);
var
  R: TRect;
  Buff: array[0..255] of Char;
```

```
begin
  with BitBtn1 do begin
    Caption := 'A really long caption';
    Glyph.Canvas.Font := Self.Font;
    Glyph.Width := Width - 6;
    Glyph.Height := Height - 6;
    R := Bounds(0, 0, Glyph.Width, 0);
    StrPCopy(Buff, Caption);
    Caption := '';
    DrawText(Glyph.Canvas.Handle, Buff, StrLen(Buff), R,
             DT_CENTER or DT_WORDBREAK or DT_CALCRECT);
    OffsetRect(R, (Glyph.Width - R.Right) div 2, (Glyph.Height - R.Bottom) div 2);
   DrawText(Glyph.Canvas.Handle, Buff, StrLen(Buff), R, DT_CENTER or DT_WORDBREAK);
  end;
end;
```

Как назначить быструю клавишу объекту, не имеющему заголовка (Caption)?

В данном случае можно создать невидимый Label и назначить ему соответствующий обработчик. В приведенном примере по нажатию клавиш <Alt>+<M> фокус будет переведен на Memo1.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Label1.Visible := false;
  Label1.Caption := '&M';
  Label1.FocusControl := Memo1;
end;
```

Что надо сделать, чтобы при вводе пароля в поле редактирования не отображались вводимые символы?

Необходимо определить, какой символ (например *) будет отображаться при вводе пароля, и назначить его свойству Password в компоненте TEdit. Если для этого использовать символ пробела (#32), то символы при вводе отображаться не будут.

Как ограничить в TEdit количество вводимого текста, чтобы он не выходил за пределы объекта?

Ниже приведены два варианта решения данного вопроса. Первый основан на предварительном подсчете количества входящих букв, исходя из количества входящих "W" как наиболее широкой буквы (в русском, наверное, это должна быть буква "Ы"). Второй – на подсчете ширины введенных символов непосредственно во время ввода. Второй способ предпочтительнее, т. к. первый может выдавать неправильные результаты в случае шрифтов разного размера.

```
procedure TForm1.FormCreate(Sender: TObject);
var
  cRect: TRect;
  bm: TBitmap;
  s: string;
```

```
begin
  Windows.GetClientRect(Edit1.Handle, cRect);
  bm := TBitmap.Create;
  bm.Width := cRect.Right;
  bm.Height := cRect.Bottom;
  bm.Canvas.Font := Edit1.Font;
  s := 'W';
  while bm.Canvas.TextWidth(s) < CRect.Right do
    s := s + 'W';
  if length(s) > 1 then begin
    Delete(s, 1, 1);
    Edit1.MaxLength := Length(s);
  end;
end;

{ Alternatively }

procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
var
  cRect: TRect;
  bm: TBitmap;
begin
  if ((Ord(Key) <> VK_TAB) and (Ord(Key) <> VK_RETURN) and (Ord(Key) <> VK_LEFT)
      and (Ord(Key) <> VK_BACK)) then begin
    Windows.GetClientRect(Edit1.Handle, cRect);
    bm := TBitmap.Create;
    bm.Width := cRect.Right;
    bm.Height := cRect.Bottom;
    bm.Canvas.Font := Edit1.Font;
    if bm.Canvas.TextWidth(Edit1.Text + Key) > CRect.Right then begin
      Key := #0;
      MessageBeep($FFFFFFFF);
    end;
    bm.Free;
  end;
end;
```

Как включить режим перезаписывания в TMemo и TEdit?

Эти два компонента не поддерживают режим перезаписывания. Но если немного исхитриться с SetLength, то можно добиться желаемого результата.

```
type
  TForm1 = class(TForm)
    Memo1: TMemo;
    procedure Memo1KeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
    procedure Memo1KeyPress(Sender: TObject; var Key: Char);
  private
    InsertOn : bool;
  end;

var
  Form1: TForm1;

implementation
```

```
{$R *.DFM}

procedure TForm1.Memo1KeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if (Key = VK_INSERT) and (Shift = []) then InsertOn := not InsertOn;
end;

procedure TForm1.Memo1KeyPress(Sender: TObject; var Key: Char);
begin
  if ((Memo1.SelLength = 0) and (not InsertOn)) then Memo1.SelLength := 1;
end;
```

Какой объем текста может содержать Delphi-компонент Memo?

Элементы редактирования Microsoft, интегрированные в Windows и применяемые в компонентах-обертках TEdit и TMemo, имеют ограничение по объему текста, равное 32 Кбайт. Классы-оболочки Delphi имеют специальный механизм, позволяющий каждому элементу управления типа Edit или Memo, размещенному на форме, содержать до 32 Кбайт текста. При нормальной ситуации суммарный предел всех элементов редактирования приложения ограничен 32 Кбайт.

Как вставить содержимое файла в TMemo с текущей позиции?

Используем TMemoryStream для чтения файла, после чего – SetSelTextBuf() для вставки текста.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  TheMStream: TMemoryStream;
  Zero: char;
begin
  TheMStream := TMemoryStream.Create;
  TheMStream.LoadFromFile('C:\AUTOEXEC.BAT');
  TheMStream.Seek(0, soFromEnd);
// Null terminate the buffer!
  Zero := #0;
  TheMStream.Write(Zero, 1);
  TheMStream.Seek(0, soFromBeginning);
  Memo1.SetSelTextBuf(TheMStream.Memory);
  TheMStream.Free;
end;
```

Как установить табуляторы в элементе управления TMemo?

Для установки табулятора в компоненте многострочного редактирования (например, TMemo) пошлите ему сообщение EM_SetTabStops. Массив Tabs указывает на месторасположение табуляторов. Поскольку параметр WParam в SendMessage равен 1, то все табуляторы будут установлены в значение, передаваемое в массиве Tabs. Не забывайте для включения табуляторов устанавливать свойство WantTabs компонента TMemo в True.

```
procedure TForm1.FormCreate(Sender: TObject);
```

```
const
  TabInc: LongInt = 10;
begin
  SendMessage(Memo1.Handle, EM_SetTabStops, 1, Longint(@TabInc));
end;
```

Как отобразить содержимое Memo-поля в DBGrid?

Используйте расположенный ниже код в обработчике события OnDrawDataCell компонента DBGrid. Примечание: перед созданием объекта TMemoField дважды щелкните на компоненте TTable и добавьте Memo-поле.

```
procedure TForm1.DBGrid1DrawDataCell(Sender: TObject; const Rect: TRect;
                                     Field: TField; State: TGridDrawState);
var
  P: array [0..50] of char;      { размер массива - необходимое число символов }
  BS: TBlobStream;               { из Memo-поля }
  S: String;
begin
  if Field is TMemoField then begin
    with (Sender as TDBGrid).Canvas do begin
{ Table1Notes - TMemoField }
      BS := TBlobStream.Create(Table1Notes, bmRead);
      FillChar(P, SizeOf(P), #0);      { строка с терминатором }
      BS.Read(P, 50);                  { читаем 50 символов из memo в BlobStream }
      BS.Free;
      S := StrPas(P);
      while Pos(#13, S) > 0 do         { удаляем символы перевода строки и }
        S[Pos(#13, S)] := ' ';         { пропуска страницы }
      while Pos(#10, S) > 0 do
        S[Pos(#10, S)] := ' ';
      FillRect(Rect);                  { очищаем ячейку }
      TextOut(Rect.Left, Rect.Top, S); { заполняем ячейку данными memo }
    end;
  end;
end;
```

Что надо сделать, чтобы определенное поле в компоненте TDBGrid не выводилось?

Это может быть сделано с помощью простого удаления поля из списка полей Fields Editor или установки свойства поля Visible в False.

Что надо сделать, чтобы компонент другого окна скопировал свое содержимое в буфер обмена Windows?

Отправка сообщения WM_COPY компоненту активного окна заставит его выполнить требуемые действия.

```
SendMessage(Edit1.handle, WM_COPY, 0, 0);
```

Существует ли способ заполнения TListbox или TMemo за один проход?

Для заполнения сразу нескольких строк в компоненте TListBox или TMemo может применяться метод SetText. Методу SetText передается строка с термини-

рующим нулем, где каждая строка сопровождается разделителем, символом возврата каретки – (#13). Пояснительный пример:

```
ListBox1.Items.SetText('aaaaa'#13'bbbbb'#13'ccccc');
```

При этом ListBox отобразит следующий текст:

```
aaaaa
bbbbb
ccccc
```

Примечание

Предпочтительным методом заполнения ListBox *или* Memo *является метод Add их соответствующих свойств* Items *и* Lines.

Существует ли у компонента ListBox свойство, позволяющее выровнять текст по правому краю?

Нет. Попробуйте заполнить строку пробелами.

Как при добавлении новых элементов (ListBoxName.Items.Add('xxx')) делать их видимыми? Поясняю: если общее количество элементов превышает количество видимых в окне, появляется полоса прокрутки, и новые пункты добавляются в нижнюю, невидимую часть списка. Как переместить список «наверх», чтобы был виден последний введенный пункт.

Попробуйте установить:

```
ListBoxName.Items.Add('xxx');
ListBoxName.ItemIndex := (ListBoxName.Items.Count - 1);
```

Как отменить выбор одного или нескольких элементов в компоненте ListBox или ComboBox?

```
ListBox1.ItemIndex := -1;
```

Почему, если в компоненте ComboBox свойству Style присвоить значение csDropDownList, то следующая строка ничего не делает?

```
ComboBox1.Text := 'Здесь происходит что-то странное!';
```

Это стандартное поведение выпадающего списка Windows при заданном стиле DropDownList. Идея этого стиля в том, что выводимый текст должен совпадать с одним из элементов списка. В этом случае правильным будет выбор одного из элементов:

```
ComboBox1.ItemIndex := 0;
{ предположим ComboBox1.Items[0] = 'Здесь происходит что-то странное!' }
```

Как перехватывать сообщения о прокрутке TScrollBar?

Пример передвигает второй ScrollBar компонента ScrollBox на такое же количество единиц, на которое пользователь передвинет первый.

```
type
{$IFDEF WIN32}
  WParameter = LongInt;
{$ELSE}
  WParameter = Word;
{$ENDIF}
  LParameter = LongInt;

{ Declare a variable to hold the window procedure we are replacing }
var
  OldWindowProc: Pointer;

function NewWindowProc(WindowHandle: hWnd; TheMessage: WParameter;
                       ParamW: WParameter; ParamL: LParameter): LongInt
{$IFDEF WIN32} stdcall; {$ELSE}; export; {$ENDIF}
var
  TheRangeMin: integer;
  TheRangeMax: integer;
  TheRange: integer;
begin
  if TheMessage = WM_VSCROLL then begin
{ Get the min and max range of the horizontal scroll box }
    GetScrollRange(WindowHandle, SB_HORZ, TheRangeMin, TheRangeMax);
{ Get the vertical scroll box position }
    TheRange := GetScrollPos(WindowHandle, SB_VERT);
{ Make sure we wont exceed the range }
    if TheRange < TheRangeMin then TheRange := TheRangeMin
    else if TheRange > TheRangeMax then TheRange := TheRangeMax;
{ Set the horizontal scroll bar }
    SetScrollPos(WindowHandle, SB_HORZ, TheRange, true);
  end;
  if TheMessage = WM_HSCROLL then begin
{ Get the min and max range of the horizontal scroll box }
    GetScrollRange(WindowHandle, SB_VERT, TheRangeMin, TheRangeMax);
{ Get the horizontal scroll box position }
    TheRange := GetScrollPos(WindowHandle, SB_HORZ);

{ Make sure we wont exceed the range }
    if TheRange < TheRangeMin then TheRange := TheRangeMin
    else if TheRange > TheRangeMax then TheRange := TheRangeMax;
{ Set the vertical scroll bar }
    SetScrollPos(WindowHandle, SB_VERT, TheRange, true);
  end;
{ Call the old Window procedure to allow processing of the message. }
  NewWindowProc := CallWindowProc(OldWindowProc, WindowHandle, TheMessage,
                                  ParamW, ParamL);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
{ Set the new window procedure for the control and remember the old window procedure }
  OldWindowProc := Pointer(SetWindowLong(ScrollBox1.Handle, GWL_WNDPROC,
                           LongInt(@NewWindowProc)));
end;
```

```
procedure TForm1.FormDestroy(Sender: TObject);
begin
{ Set the window procedure back to the old window procedure. }
  SetWindowLong(ScrollBox1.Handle, GWL_WNDPROC, LongInt(OldWindowProc));

end;
```

Где найти исходный код страничных компонентов (например, TTabbedNotebook)?

Исходный код VCL не содержит код "Tab"-компонентов по юридическим причинам.

Примечание

Зарегистрированные владельцы исходного кода Delphi RTL могут запросить исходный код TTabSet *и* TTabbedNotebook *у подразделения Borland Corporate Affairs. Инструкция находится в файле readme исходного кода RTL.*

Почему TTabbedNotebook использует так много системных ресурсов, если в отдельный момент видна только одна его страница?

Даже если отображается только одна страница, все остальные страницы со всеми надлежащими компонентами также уже созданы, расходуя при этом системные ресурсы. Есть решение, альтернативное страницам NoteBook, – используйте для каждой страницы отдельные формы, которые создаются и уничтожаются в зависимости от щелчка пользователя на соответствующей вкладке. Необходимо задать параметры создания каждой дочерней формы, как показано ниже:

```
  ...
  private
procedure CreateParams(VAR Params: TCreateParams); override;
  ...
procedure TForm2.CreateParams(VAR Params: TCreateParams);
begin
  Inherited CreateParams(Params);
  with Params do begin
    WndParent := Application.MainForm.Handle;
    Style := (Style OR WS_CHILD) AND NOT (WS_POPUP);
  end;
end;
```

Свойство дочерней формы BorderStyle должно быть установлено в bsNone. В главной форме создайте закрытое поле данных с типом TForm. Инициализируйте его при наступлении события OnActivate (не OnCreate). Теперь при каждом щелчке по вкладке «смены страниц» освобождаем текущую дочернюю форму и создаем новую. Например, при наступлении события OnActivate:

```
Child := TForm2.Create(Self);
with Child do begin
  Parent := Self;
  Align := alClient;
```

```
    Visible := True;
  end;
```

Создавая дочернюю страницу при щелчках на вкладках, делайте это так, как показано выше. Естественно, надо будет использовать главную форму для хранения данных о состоянии элементов управления дочернего окна, поскольку при его освобождении данные теряются.

Существует ли в природе компонент коммерческий или свободный, позволяющий создавать вертикальные (на левой или правой части страницы) вкладки в Блокноте.

Такую возможность имеет продукт фирмы TurboPower Orpheus. Для получения дополнительной информации посетите сервер компании TURBOPOWER.

Как добавить акселераторы ко вкладкам PageControl (TabSheets)?

Пример демонстрирует перехват сообщения `CM_DIALOGCHAR`:

```
type
  TForm1 = class(TForm)
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    TabSheet2: TTabSheet;
    TabSheet3: TTabSheet;
  private
    procedure CMDialogChar(var Msg: TCMDialogChar); message CM_DIALOGCHAR;
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.CMDialogChar(var Msg: TCMDialogChar);
var
  i: Integer;
begin
  with PageControl1 do begin
  if Enabled then
    for i := 0 to PageControl1.PageCount - 1 do
      if ((IsAccel(Msg.CharCode, Pages[i].Caption))
          and (Pages[i].TabVisible)) then begin
        Msg.Result := 1;
        ActivePage := Pages[i];
        exit;
      end;
  end;
  inherited;
end;
```

Как обойти ошибку Stream Read Error?

Попробуйте удалить `*.DFM`-файл и перекомпилировать проект.

Каково лучшее решение для создания группы RadioGroup и размещения в ней кнопок RadioButton? Кажется, можно просто создать компонент RadioGroup и «набросать» туда несколько элементов RadioButton. Или лучше создать RadioGroup, задать значение свойству Items, тем самым определив заголовки элементов (RadioButton) и разместив их в группе?

Если вы собираетесь работать с RadioGroup, то логичным будет для создания RadioButton воспользоваться списком строк Items. При этом RadioButton не должны быть объединены в группу, именно поэтому к элементу RadioGroup нельзя обратиться как к отдельному компоненту RadioButton.

Как получить имя пользователя, зарегистрировавшегося в системе?

```
function GetCurrentUserName: string;
var
  Len: Cardinal; { This will have to be Integer, not cardinal, in Delphi 3. }
begin
{ arbitrary length to allocate for username string, plus one for null terminator }
  Len := 255;
  SetLength(Result, Len - 1);              { set the length }
  if GetUserName(PChar(Result), Len) then  { get the username }
    SetLength(Result, Len - 1)             { set the exact length if it succeeded }
  else begin
    RaiseLastWin32Error;                   { raise exception if it failed }
  end;
end;
```

Где получить помощь по работе с ReportSmith, InterBase и связке SQL Links/ODBC?

Зайдите на борландовский форум разработчиков (BDEVTOOLS). В нем присутствуют группы, посвященные работе с ReportSmith, InterBase, Borland Database Engine, SQL Links/ODBC connectivity и др.

Как избавиться от заставки ReportSmith при запуске своего отчета?

Добавьте следующую строку в секцию [RS_RunTime] файла RS_RUN.INI, это позволит избежать появления логотипа при запуске ReportSmith:

```
ShowAboutBox = 0
```

При попытке запустить приложение для работы с базой данных из-под Delphi я получаю исключение EDatabaseError с сообщением «An error occurred while attempting to initialize the Borland Database Engine (Error $2C09)» (ошибка при инициализации BDE).

Добавьте SHARE.EXE в файл AUTOEXEC.BAT или DEVICE=VSHARE.386 в секцию [386Enh] файла SYSTEM.INI и перезагрузитесь.

Что значит ошибка IDAPI $2C08?

«Не могу загрузить IDAPI01.DLL». Убедитесь, что в вашем файле WIN.INI в секции, указанной ниже, в переменной DLLPATH указан корректный путь к IDAPI:

```
[IDAPI]
DLLPATH=C:\IDAPI
CONFIGFILE01=C:\IDAPI\IDAPI.CFG
```

Где найти список и описание функций BDE и типов данных?

`BDE.INT` в вашем каталоге DELPHI5\DOC\ содержит список функций BDE, ожидаемые параметры, возвращаемые значения и их краткое описание. Для вызова функций BDE необходимо наличие модуля BDE в списке USES. Также может потребоваться BDECONST. Для получения более подробной информации по функциям IDAPI закажите руководство «Database Engine User's guide» в службе поддержки пользователей.

Вы можете дать определение IDAPI?

IDAPI – Integrated Database Application Program Interface (Интегрированный программный интерфейс приложений баз данных). BDE обеспечивает доступ к множеству источников данных. Интерфейс IDAPI – это API для BDE. Он включает в себя все функции, необходимые для получения доступа, манипуляции данными и т. п. Delphi, dBASE for Windows и Paradox for Windows посредством этих функций получают доступ к данным. Пользователи также могут обращаться к ним в своих программах. С приобретением BDE вы также получаете всю необходимую документацию. В ней содержится перечень всех доступных функций с описанием того, что они делают. Если посмотреть исходные файлы Delphi, то можно увидить, как используются там эти функции. Названия функций имеют префикс `"Dbi"` (например, `DbiCreateTable`).

SQL Links – это коллекция драйверов, обеспечивающих подключение к удаленным серверам баз данных.

Можно ли программным путем добавить псевдоним к файлу IDAPI.CFG?

Функция BDE, позволяющая сделать это, называется `DbiAddAlias()`. Спецификация доступна в Section 6 (Database) библиотеки, файл `AddAlias.txt`. Также в Section 6 (Database) библиотеки доступен компонент `AliasManager`. С его помощью возможно создание, удаление и редактирование псевдонимов.

IDAPI необходим для доступа к данным в Delphi? Может быть, как-то «упрятать» IDAPI внутрь Delphi EXE-файла, чтобы не устанавливать IDAPI на каждом компьютере пользователя вдобавок к моей программе?

IDAPI необходим для доступа к данным в Delphi. Delphi поставляется с дискетой, содержащей установку IDAPI.

При редактировании записи я получаю сообщение DBEngine, гласящее: «Multiple records found but only one expected» (обнаружены многочисленные записи, в то время как ожидалась одна). Что это значит?

Возможно, вам придется создать уникальный индекс в таблице для того, чтобы каждая строка могла быть однозначно идентифицирована. Для этого необходимо добавить в таблицу колонку и заполнить ее уникальными значениями.

Как определить номер записи набора данных?

Если набор данных связан с таблицей Paradox или dBASE, то для определения номера записи достаточно пары вызовов BDE (как показано ниже). BDE не поддерживает нумерацию записей для набора данных, базирующегося на таблицах SQL, поэтому, если ваш сервер сам поддерживает нумерацию записей, то для выяснения способа ее получения необходимо обратиться к его документации.

Приведенная ниже функция в качестве параметра принимает любой компонент, производный от `TDataset` (например, `TTable`, `TQuery`, `TStoredProc`) и возвращает номер текущей записи (больший чем ноль), если это таблица Paradox или dBASE. В противном случае функция возвращает ноль.

Примечание

Для таблиц dBASE функция всегда возвращает физический номер записи. Так, если набор данных – `TQuery`, или в наборе данных установлен диапазон (фильтр), то возвращаемый номер записи не обязательно должен относиться к заданному набору данных, а может отражать физическое расположение записи в таблице dBASE.

```
uses DbiProcs, DbiTypes, DBConsts;

function RecordNumber(Dataset: TDataset): Longint;
var
  CursorProps: CurProps;
  RecordProps: RECProps;
begin
{ Возвращает 0, если набор данных связан не с Paradox или dBASE }
  Result := 0;
  with Dataset do begin
{ набор данных активен? }
    if State = dsInactive then DBError(SDataSetClosed);
{ Данный вызов необходим для получения курсора iSeqNums }
    Check(DbiGetCursorProps(Handle, CursorProps));
{ Синхронизируем курсор BDE с курсором Dataset }
    UpdateCursorPos;
{ Заполняем RecordProps свойствами текущей записи }
    Check(DbiGetRecord(Handle, dbiNOLOCK, nil, @RecordProps));
{ какого типа наш набор данных? }
    case CursorProps.iSeqNums of
       0:  Result := RecordProps.iPhyRecNum;      { dBASE }
       1:  Result := RecordProps.iSeqNum;         { Paradox }
    end;
  end;
end;
```

Примечание

Пример для Delphi 1 (Win16). Для адаптации под Win32 необходимо заменить вызовы функций BDE, имена модулей, в которых они содержатся, и т. д.

Как узнать, что текущая запись в наборе данных изменилась?

Проверьте в обработчике события OnDataChanged компонента DataSource значение свойства State. В случае изменения текущей записи свойство State будет иметь значение dsBrowse. Следующий пример выводит информационное окно при каждом изменении записи в источнике данных MyDataSource:

```
procedure TMyForm.MyDataSourceDataChange(Sender: TObject; Field: TField);
begin
  if (Sender as TDataSource).State = dsBrowse then
     ShowMessage('Позиция записи изменилась');
end;
```

Может ли BDE API или другие доступные DLL паковать таблицы dBASE?

Функция BDE для упаковки таблиц dBASE называется DbiPackTable().

Добавьте следующие модули в секцию USES: DBITYPES, DBIPROCS и DBIERRS. Затем функцию можно вызвать следующим образом:

```
DBIPackTable(Table1.DbHandle, Table1.Handle, 'TABLENAME.DBF', szDBASE, TRUE);
```

Примечание

Таблица должна быть открыта в монопольном режиме. Для современных версий Delphi необходимо указать модуль BDE.

Как посмотреть записи dBASE, помеченные для удаления?

В обработчике события AfterOpen вызовите нижеприведенную функцию. При вызове функции укажите в качестве аргументов TTable и True/False для отображения/скрытия удаленных записей.

```
uses bde;

procedure SetDelete(oTable: TTable; Value: Boolean);
var
  rslt: DBIResult;
  szErrMsg: DBIMSG;
begin
  try
    oTable.DisableControls;
    try
      rslt := DbiSetProp(hDBIObj(oTable.Handle), curSOFTDELETEON, LongInt(Value));
      if rslt <> DBIERR_NONE then begin
        DbiGetErrorString(rslt, szErrMsg);
        raise Exception.Create(StrPas(szErrMsg));
      end;
    except
      on E: EDBEngineError do ShowMessage(E.Message);
      on E: Exception do ShowMessage(E.Message);
    end;
  finally
    oTable.Refresh;
    oTable.EnableControls;
```

```
  end;
end;

procedure TForm1.Table1AfterOpen(DataSet: TDataset);
begin
  SetDelete(Table1, True);
end;
```

Как в табличной сетке создать колонку, в которую будут заноситься записи из таблицы dBASE, помеченные для удаления?

Создайте вычисляемые поля, затем в обработчике события таблицы `OnCalcField` замените вычисляемые поля, которые вы создали, следующим образом:

```
procedure TForm1.Table1CalcFields(DataSet: TDataset);
var
  RCProps: RecProps;
  Result: DBIResult;
begin
  Result := DbiGetRecord(Table1.Handle, dbiNoLock, Nil, @RCProps);
  if RCProps.bDeleteFlag then Table1Del.Value := 'X'
  else Table1Del.Value := '';
end;
```

Примечание

Сначала следует вызвать функцию `SetDelete(TTable, TRUE)`*, описанную в предыдущем вопросе.*

Почему при создании таблицы с помощью метода CreateTable компонента TTable поля создаются правильно, но индексы не создаются, даже если сделать следующее:

```
NewTable.IndexDefs.Assign(Table1.IndexDefs);
```

Вы правильно передали определение индекса в `NewTable`, но, тем не менее, свойство `IndexDefs Table1` следует обновить, для этого существует метод `Update`:

```
with NewTable do begin
  Active := False;
  DatabaseName := 'DBDEMOS';
  TableName := 'Temp';
  TableType := ttParadox;
  FieldDefs.Assign(Table1.FieldDefs);
  Table1.IndexDefs.Update;                    { В первую очередь обновляем }
  IndexDefs.Assign(Table1.IndexDefs);
  CreateTable;
end;
```

Почему я получаю сообщение «Index out of range» (индекс за пределами диапазона), когда использую Table.FindNearest и Table.FindKey в таблице dBASE с выражением индекса (expression index)?

`FindKey` и `FindNearest` не хотят работать с выражениями индексов dBASE. Для нормальной работы с выражениями индексов dBASE вызывайте методы класса `TTable GoToKey` и `GotoNearest`.

Может ли BDE API или другие доступные DLL воссоздавать разрушенные индексы (как, например, TUTILITY.EXE, поставляемая с Pdoxwin)?

Функция BDE для воссоздания индексов называется `DbiRegenIndexes()`.

Добавьте следующие модули в секцию `USES`: `DBITYPES`, `DBIPROCS` и `DBIERRS`. Затем функцию можно вызвать следующим образом:

```
DBIRegenIndexes(Table1.Handle);
```

Примечание

Таблица должна быть открыта в монопольном режиме, и индекс должен уже существовать.

Почему нельзя использовать опцию ixUnique при создании индекса таблицы Paradox с помощью метода AddIndex компонента TTable?

Опции индекса, которыми «руководствуется» метод `AddIndex` компонента `TTable`, чувствительны к типу таблицы. Для примера, опция `ixUnique` работает с таблицами dBASE, но не с таблицами Paradox. Ниже показано, какие опции могут быть применены к разным таблицам (dBASE или Paradox).

Опции индекса	dBASE	Paradox
`ixUnique`	*	
`ixDescending`	*	*
`ixNonMaintained`	*	*
`ixPrimary`		*
`ixCaseInsensitive`		*

Как программным путем создать таблицу Paradox с автоприращиваемым (Auto Increment) типом поля? Я вызываю TTable.CreateTable, но TFieldType не поддерживает этот тип полей.

Используйте `TQuery` или SQL-запрос `CREATE TABLE`. Например:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  with Query1 do begin
    DatabaseName := 'DBDemos';
    with SQL do begin
      Clear;
      Add('CREATE TABLE "PDoxTbl.db" (ID AUTOINC,');
      Add('Name CHAR(255),');
      Add('PRIMARY KEY(ID))');
      ExecSQL;
      Clear;
      Add('CREATE INDEX ByName ON "PDoxTbl.db" (Name)');
      ExecSQL;
    end;
  end;
end;
```

Есть ли в Delphi эквивалент Paradox TCursor?

Компонент TTable.

В чем разница между SetKey/GoToKey и FindKey? Как определить, когда следует использовать одно вместо другого?

Функционально разницы нет никакой. SetKey плюс некоторые возможности для работы с IndexFields, плюс GotoKey эквивалентны FindKey. Если работа с IndexFields для вас сложна, программируйте с помощью SetKey.

Как воспользоваться функцией Locate в неиндексированном поле?

Ниже приводится код функции, которую можно разместить в своем модуле и вызывать следующим образом:

```
Locate(Table1, Table1LName, 'Beman');
```

Table1 – компонент Table, Table1Lname – TField, поле добавляемое вами с помощью редактора полей (двойной щелчок на компоненте Table), и 'Beman' – имя, которое требуется найти.

```
function Locate(const oTable: TTable; const oField: TField;
                const sValue: String): Boolean;
var
  bmPos: TBookMark;
  bFound: Boolean;
begin
  Result := False;
  bFound := False;
  if not oTable.Active then Exit;
  if oTable.FieldDefs.IndexOf(oField.FieldName) < 0 then Exit;
  bmPos := oTable.GetBookMark;
  with oTable do begin
    DisableControls;
    First;
    while not EOF do
      if oField.AsString = sValue then begin
        Result := True;
        bFound := True;
        Break;
     end else Next;
  end;
  if (Not bFound) then oTable.GotoBookMark(bmPos);
  oTable.FreeBookMark(bmPos);
  oTable.EnableControls;
end;
```

Примечание

Зачем писать функцию, если можно воспользоваться методом Locate TTable. *Можно искать сразу по нескольким полям без индекса, с дополнительными опциями:*

```
with CustTable do
  Locate('Company;Contact;Phone', VarArrayOf(['Sight Diver', 'P', '408-431-1000']),
         [loPartialKey]);
```

Как предотвратить появление диалогового окна с запросом пароля, открывая таблицу, защищенную паролем?

Обеспечьте объект Session паролем, который необходимо ввести для открытия таблицы:

```
Session.AddPassword ('PASSWORD');
```

При закрытии таблицы надо удалить пароль с помощью команды RemovePassword('PASSWORD') или удалить все текущие пароли с помощью RemoveAllPasswords.

Примечание

Все вышесказанное относится только к таблицам Paradox.

У меня имеются TQuery и TDataSource. Свойство Query1.SQL имеет значение SELECT * FROM dbo.AnyTable, где dbo – база данных на моем SQL-сервере. Устанавливая свойство Active в TRUE, я получаю следующую ошибку: «Token not found. Token :dbo. line number:1». В чем дело?

Если свойство RequestLive установлено в True, тогда имя базы данных и таблицы необходимо указывать в кавычках:

```
SELECT * FROM "dbo.table"
```

Если свойство RequestLive установлено в False, кавычки не нужны:

```
SELECT * FROM dbo.table
```

Как создать маску в компоненте TDBEdit?

Маски редактирования применимы к полям таблицы (компоненты TField), а не к управляющим элементам управления для работы с базами данных. Дважды щелкните по пиктограмме TTable и добавьте необходимые поля таблицы. Если поле выделено, его свойства появляются в Инспекторе объектов, включая редактор маски. Связывание поля с TDBEdit позволяет решить эту проблему.

Компонент TDBGrid имеет события OnMouseDown, OnMouseUp и OnMouseMove?

События существуют, но они не опубликованы. Можно создать простой наследник TDBGrid и опубликовать их.

Как узнать, какая запись и какое поле TDBGrid в данный момент являются текущими?

Метод, следящий за текущими колонкой и строкой. Приведенный ниже код в методе DBGrid1DrawDataCell обновляет переменные Col и Row (которые не должны быть локальными для метода) при каждой отрисовке табличной сетки. Используя данный код, можно всегда с помощью переменных Col и Row узнать номер текущей колонки и строки соответственно.

```
var
  Col, Row: Integer;
```

```
procedure TForm1.DBGrid1DrawDataCell(Sender: TObject; const Rect: TRect;
                                     Field: TField; State: TGridDrawState);
var
  RowHeight: Integer;
begin
  if gdFocused in State then begin
    RowHeight := Rect.Bottom - Rect.Top;
    Row := (Rect.Top div RowHeight) - 1;
    Col := Field.Index;
  end;
end;
```

Примечание

Номера колонки и строки в пределах области отображения.

Как выделить текущую строку в TDBGrid?

В свойстве TDBGrid.Options выставьте флаг dgRowSelect.

Как изменить цвет ячейки TDBGrid?

Создайте следующий обработчик события компонента TDBGrid OnDrawDataCell:

```
procedure TForm1.DBGrid1DrawDataCell(Sender: TObject; const Rect: TRect;
                                     Field: TField; State: TGridDrawState);
begin
  if gdFocused in State then
    with (Sender as TDBGrid).Canvas do begin
      Brush.Color := clRed;
      FillRect(Rect);
      TextOut(Rect.Left, Rect.Top, Field.AsString);
    end;
end;
```

Установите отрисовку по умолчанию в True. Только таким способом можно нарисовать отдельную ячейку. Если же DefaultDrawing установлено в False, придется самостоятельно отрисовать все ячейки, используя свойства Canvas.

Как определить реальный размер BLOB-поля, хранящегося в таблице?

Здесь приведена функция GetBlobSize, возвращающая размер заданного BLOB, Memo или графического поля. Пример вызова функции:

```
function GetBlobSize(Field: TBlobField): Longint;
begin
  with TBlobStream.Create(Field, bmRead) do
    try
      Result := Seek(0, 2);
    finally
      Free;
   end;
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
begin
{ Поле редактирования Edit1 будет показывать размер memo-поля с именем Notes. }
  Edit1.Text := IntToStr(GetBlobSize(Notes));
end;
```

Какие версии Informix (Online, I-NET) поддерживает SQL Links?

SQL Links прекрасно работает со всеми версиями Informix sever, включая версию 5.01. Тем не менее, она не совместима с новыми версиями их клиентов (клиент Windows версии 5.0). В случае клиентов следует пользоваться клиентом версии 4.2 (для DOS).

Мой опыт обращения к данным Microsoft Access через Delphi-компонент TTable успешным не назовешь. Используя Tquery, я смог получить для работы только представление с флажком «только для чтения», но представление с возможностью чтения/записи получить не удалось. После диалога авторизации я получил сообщение об исключительной ситуации типа «Passthrough SQL connection must be shared».

Используйте Database Engine Configuration для изменения опции SQLPASSTHRU MODE в псевдониме, связанном с вашей базой данных Access, из пустого значения по умолчанию на SHARED AUTOCOMMIT (без кавычек).

На машине установлен и функционирует Quattro Pro 6.0 и IDAPI для работы в сети. Установив Delphi и новый IDAPI поверх сетевого IDAPI и запуская Quattro Pro с другой машины, я получаю сообщение о том, что невозможно загрузить драйвер языка (Language Driver).

Добавьте в секцию [Borland Language Drivers] файла WIN.INI строчку, содержащую путь к каталогу \IDAPI\LANGDRV. Для примера:

```
[Borland Language Drivers]
LDPATH=C:\IDAPI\LANGDRV
```

Возможно ли хранение целочисленной величины вместе со строкой в объекте списка TString или каком-либо свойстве?

Да, но это потребует некоторых преобразований типов. Компонент TString вместе с массивом строк имеет массив объектов, который может использоваться с целью хранения целочисленных данных: типом данных для массива объектов служит TObject. В сущности, он хранит значение указателя размером 4 байта. Поэтому для хранения в нем целочисленной величины необходимо приведение типа значения. Для примера: следующий код добавляет строку и целое число 100 к свойству Items (объект TString) компонента ListBox:

```
ListBox1.Items.AddObject('Текстовая строка', TObject(100));
```

Для получения значения сделайте следующее:

```
Result := LongInt(ListBox1.Items.Objects[0]);
```

Здесь подразумевается, что Result имеет тип Longint, и значение хранится в объекте с индексом 0.

Примечание

Все-таки подобное использование объекта не является традиционным решением вопроса. Поэтому такой код должен быть хорошо прокомментирован.

Если надо хранить более одного значения, создайте новый, производный от TObject класс и сделайте необходимые объявления:

```
type
  ManyValues = class(TObject)
    Value1: Integer;
    Value2: Integer;
  end;
```

Как в Delphi осуществить обработку ошибок времени выполнения?

При возникновении ошибки времени выполнения Delphi генерирует исключение (или, правильнее сказать, создает объект исключительной ситуации). Если отмечена соответствующая опция в настройках среды, исключения будут перехватываться во время выполнения программы из-под Delphi, и Delphi будет перемещать курсор к строке, вызвавшей исключительную ситуацию. Тем не менее, программа при этом не завершает свою работу. Напротив, при возникновении исключительной ситуации программа не сможет окончить свою работу в автоматическом режиме, что связано с необходимостью поработать с программой в «предсмертном» режиме и найти источник ошибки.

Существует ли простой способ перехватывать исключения в событиях элементов управления?

Создайте метод формы для перехвата исключений. Данный метод вызывает метод приложения (объект Application) OnException. В нем необходимо производить проверку на необходимые исключения, например EDatabaseError. Откройте файл справки и посмотрите информацию о событии OnException. Там ясно и достаточно подробно описан процесс создания обработчика данного события. Например:

```
procedure TForm1.MyExcept(Sender: TObject; E: Exception);
begin
  if E is EDatabaseError then
    MessageDlg('Перехвачено исключение', mtInformation, [mbOk], 0)
  else
{ если эта не та ошибка, которую вы ищете, передайте объект исключительной
                                                          ситуации дальше }
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Application.OnException := MyExcept;
```

```
{ Мы только что назначили обработчик события OnException }
end;
```

Как гарантированно очистить экран в консольном приложении?

Надо просто использовать `GetConsoleScreenBufferInfo()` для ввода нескольких пустых строк.

```
program Project1;

{$APPTYPE CONSOLE}

uses Windows;

{$R *.RES}

var
  sbi: TConsoleScreenBufferInfo;
  i: integer;
begin
  Writeln('A Console Applicaiton');
  Writeln('Press Enter To Clear The Screen');
  GetConsoleScreenBufferInfo(GetStdHandle(STD_OUTPUT_HANDLE), sbi);
  Readln;
  GetConsoleScreenBufferInfo(GetStdHandle(STD_OUTPUT_HANDLE), sbi);
  for i := 0 to sbi.dwSize.y do Writeln;
  Writeln('Press Enter To End');
  Readln;
end.
```

Моя программа выполняет длительный расчет. В это время компьютер не реагирует на нажатия клавиш и не прорисовывает изменяющиеся компоненты на форме. Что делать?

Вставить в длинном цикле `Application.ProcessMessages`, чтобы дать возможность системе обрабатывать сообщения.

Можно ли определить изменение времени другим приложением?

Код демонстрирует это. Учтите, что приложение, меняющее время, должно передать сообщение `WM_TIMECHANGE` всем другим приложениям.

```
type
  TForm1 = class(TForm)
  private
    procedure WMTIMECHANGE(var Message: TWMTIMECHANGE); message WM_TIMECHANGE;
  end;
var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.WMTIMECHANGE(var Message: TWMTIMECHANGE);
```

```
begin
  Form1.Caption := 'Time Changed';
end;
```

Как перехватить сообщения от неклиентской части моей формы, такой, например, как заголовок?

Пример показывает, как отследить перемещения мыши на неклиентской части формы.

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm1 = class(TForm)
  private
    procedure WMNCMOUSEMOVE(var Message: TMessage); message WM_NCMOUSEMOVE;
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.WMNCMOUSEMOVE(var Message: TMessage);
var
  s: string;
begin
  case Message.wParam of
          HTERROR:  s := 'HTERROR';
    HTTRANSPARENT:  s := 'HTTRANSPARENT';
        HTNOWHERE:  s := 'HTNOWHERE';
         HTCLIENT:  s := 'HTCLIENT';
        HTCAPTION:  s := 'HTCAPTION';
        HTSYSMENU:  s := 'HTSYSMENU';
           HTSIZE:  s := 'HTSIZE';
           HTMENU:  s := 'HTMENU';
        HTHSCROLL:  s := 'HTHSCROLL';
        HTVSCROLL:  s := 'HTVSCROLL';
      HTMINBUTTON:  s := 'HTMINBUTTON';
      HTMAXBUTTON:  s := 'HTMAXBUTTON';
           HTLEFT:  s := 'HTLEFT';
          HTRIGHT:  s := 'HTRIGHT';
            HTTOP:  s := 'HTTOP';
        HTTOPLEFT:  s := 'HTTOPLEFT';
       HTTOPRIGHT:  s := 'HTTOPRIGHT';
         HTBOTTOM:  s := 'HTBOTTOM';
     HTBOTTOMLEFT:  s := 'HTBOTTOMLEFT';
```

```
      HTBOTTOMRIGHT:  s := 'HTBOTTOMRIGHT';
           HTBORDER:  s := 'HTBORDER';
           HTOBJECT:  s := 'HTOBJECT';
            HTCLOSE:  s := 'HTCLOSE';
             HTHELP:  s := 'HTHELP';
                else  s := '';
  end;
  Form1.Caption := s;
  Message.Result := 0;
end;

end.
```

Как отобразить число, разделив каждые три цифры запятыми?

```
procedure TForm1.Button1Click(Sender: TObject);
var
  i: integer;
begin
  i := 12345678;
  Memo1.Lines.Add(FormatFloat('#,', i));
end;
```

Как перевести полярные величины в линейные (радианы в градусы)?

```
uses math;

procedure TForm1.Button1Click(Sender: TObject);
var
  Angle, x, y, Distance, Radians: Double;
begin
  Distance := 100;
  Angle := 270;
  Radians := DegToRad(Angle);
  x := Round(Distance * Cos(Radians));
  y := Round(Distance * Sin(Radians));
  ShowMessage(FloatToStr(x) + ' ' + FloatToStr(y));
end;
```

Есть ли в Delphi компонент, поддерживающий последовательные коммуникации (порты)?

Нет. Тем не менее, существуют библиотеки для работы с последовательными портами и компоненты третьих фирм, как, например, TurboPower, SaxComm и др.

Как запустить используемый по умолчанию веб-броузер с каким-либо определенным URL?

Используйте `HlinkNavigateString`, описанную в модуле `UrlMon`.

Пример вызова этой функции:

```
HlinkNavigateString(Nil, 'http://www.borland.com');
```

При запуске с ActiveForm рекомендуется следующая форма запуска:

```
HlinkNavigateString(ComObject, 'http://www.borland.com');
```

Второй вариант:

```
uses ShellAPI;

procedure TForm1.Button1Click(Sender: TObject);
begin
  ShellExecute(Form1.Handle, nil, 'http://www.borland.com', nil, nil, SW_SHOWNORMAL);
end;
```

Существует ли в Delphi денежный компонент?

Нет, но существует компонент редактирования денежных сумм в секции VCL форума Delphi под именем CURREDIT.ZIP.

Два числа с плавающей запятой неправильно сравниваются! Задаю if d1 = d2 ..., или if d1<> d2, а результат иногда неверный. В чем дело? Это ошибка Delphi?

Нет. Просто, в отличие от целочисленных, IEEE-числа с плавающей запятой являются приблизительными значениями, и вы не должны использовать оператор = или <> для сравнения двух чисел такого типа. Вместо этого надо вычесть из одного числа другое и сравнить разницу с очень малой величиной.

Например:

```
if Abs(d1-d2) < 0.00001 then ShowMessage('D1 и D2 равны');
```

Как грамотно осуществить выравнивание чисел с десятичной точкой?

```
  0.50
 10.55
245.98
```

В случае TDBEdits можно добавить поля к форме и задать соответствующие значения свойствам Alignment и DisplayFormat.

Хочу перевести дату «December 6, 1969» в формат TDateTime, но StrToDate не предоставляет такой возможности. Как поступить?

```
procedure TForm1.Button1Click(Sender: TObject);
var
  D1, D2, D3: TDateTime;
begin
  D1 := VarToDateTime('Декабрь 6, 1969');
  D2 := VarToDateTime('6-Апр-1998');
  D3 := VarToDateTime('1998-Сен-6');
  ShowMessage(DateToStr(D1) + ' ' + DateToStr(D2) + ' ' + DateToStr(D3));
end;
```

Примечание

Формат задаваемых дат зависит от локализации Windows.

Как представить строку из 0 и 1 в числовом виде?

Пример переводит строку в longint.

```
function BinStringToLongInt(BinString: string): longint;
var
  i: integer;
  Num: longint;
begin
  Num := 0;
  for i := 1 to Length(BinString) do
    if BinString[i] = '1' then Num := (Num shl 1) + 1
    else Num := (Num shl 1);
  Result := Num;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  Memo1.Lines.Add(IntToStr(BinStringToLongInt('11111111')));
end;
```

Возможно ли создание аналога массива элементов управления, применяемого в Visual Basic? Например, я хочу иметь группу кнопок с общим обработчиком события, с помощью которого можно было бы определить нажатую кнопку или ее порядковый номер. В Visual Basic это можно было сделать с помощью индекса управляющего массива (Control Array).

Один из способов решения задачи состоит в присваивании полю Tag уникального числа, создании общего обработчика события для всех кнопок и проверки содержимого поля Tag передаваемого объекта Sender. Назначьте всем кнопкам группы один и тот же обработчик события OnClick. Он должен выглядеть приблизительно так:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  case TButton(Sender).Tag of
    1:  ShowMessage('Нажата первая кнопка');
    2:  ShowMessage('Нажата вторая кнопка');
    3:  ShowMessage('Нажата третья кнопка');
  end;
end;
```

Перечень авторов

Astafiev	*A.Astafiev@ftc.ru*
Bogachev	*bogachov@a-teleport.com*
Briculski	*bserge@airport.md*
Den is Com	*slaavgum@uninet.ee*
Dmitry	*dimon@diogen.nstu.nsk.su*
Debarred Sergei	*ha@inbox.ru*
Fyodorov Oleg	*ofyodorov@mtu-net.ru*
Good_mag	*good_mag@chat.ru*
Jungle Vitaliy	*skymetrics@inbox.ru*
Kordyum Alexandr	*avk@ksk-market.com.ua*
Kostin Slava	*sk1978@mailru.com*
Kravchenko Denis	*denis.k@azovstal.com.ua*
Mahotkin Alexey	*alexm@hsys.msk.ru*
Matveychuk Sergey	*sem@ciam.ru*
Nikolaev Igor	*spritesoft@mail.ru*
Novikov Alex	*mixerinc@chat.ru*
Pastushenko Andrey	*alisa@mog-pod.vinnitsa.com*
Pavel Stont	*pavel_stont@mail.ru*
Pincuk Vasili	*pvasili@geocities.com*
Polyanskiy Alex	*AlexPol@ufimb.kiev.ua*
Popov Igor	*igp@ukrpost.net*
Sadovnikoff	*morph@cci.lg.ua*
Semenushkin Igor	*sin@uc.jinr.ru*
Shejchenko Andrij	*andrij@dep01.niiit.kiev.ua*
SottNick	*sottnick@mail.ru*
Streblechenko Dmitriy	*dmitrys@phyast.la.asu.edu*
Subfire	*subfire@mail.ru*
The_Sprite	*the_sprite@mail.ru*
Trubachev Pavel	*ptrub@nvtb.ru*
Virtualik	*virty1k@mail.ru*
Vozny Alexander	*vozny@unicom.cv.ua*
VS	*shvetadvipa@mtu-net.ru*
YoungHacker	*younghac@nsys.by*
Zolotorenki Pavlo	*pvz@mail.univ.kiev.ua*

Аркадий	*ark@on-line.jar.ru*
Александр	*suhorukov@arzi.akc.ru*
Алексей	*msalex@tomcat.ru*
Беличенко Борис	*Belichenko@VVS.Viaduk.net*
Варавва Алексей	*alexv@vse.karelia.ru*
Василенко Игорь	*vasilenko@nursat.kz*
Васильев Николай	*N.Vasiliev@apatit.com*
Виталий	*vit@everest.kaluga.ru*
Волосенков Владимир	*Uno@mail.ru*
Галимарзанов Фанис	*inrus51@poikc.bashnet.ru*
Громов Роман	*GromovRV@mail.ru*
Гуменюк Максим	*max1gu@fiberia.com*
Демский Александр	*demsky@mail.ru*
Дьяченко Сергей	*sd@arzamas.nnov.ru*
Еремеев Виталий	*vit@everest.kaluga.ru*
Ермолаев Александр	*swift@glazov.udm.net*
Иваненко Фёдор	*theodor_iv@mail.ru*
Иванов Андрей	*lynxhome@mail.ru*
Каширин Дмитрий	*kashirin@yahoo.com*
Клюкач Олег	*Olaf72KOG@Yahoo.com*
Кондратюк Виталий	*vit@mo.msk.ru*
Кулабухов Олег	*www.olegon.com*
Куприн Александр	*ru_classic@mail.ru*
Лагонский Сергей	*lsn@tranzit.donetsk.ua*
Лихолетов Алексей	*alicho@go.ru*
Марковский Михаил	*mrkvsky@chem.kubsu.ru*
Николай	*nikolay@pi.net.ua*
Пангин Дмитрий	*pangin@mail.ru*
Рыбант Владимир	*newlife@intbel.ru*
Сахаров Сергей	*ssa_sss@mail.ru*
Ситников Митрий	*740561@kmr.kuzbass.net*
Слабко Дмитрий	*dima@integral.spb.su*
Степанов Павел	*mercury@gin.global-one.ru*
Тарасов Николай	*tarasov_nick@mail.ru*
Тимошенко Александр	*sherhan@ok.net.ua*
Тугаев Олег	*knsprog@krintel.ru*
Чумак Михаил	*chuka@mail.ru*
Шпанер Михаил	*mshp@mail.ru*
Щаматис Сергей	*Serg@mak.scs.ru*