Игорь Сафронов

Visual Basic
в задачах и примерах

Санкт-Петербург
«БХВ-Петербург»
2006

УДК 681.3.06
ББК 32.973.26-018.1Visual Basic
С12

Сафронов И. К.

С12 Visual Basic в задачах и примерах. — СПб.: БХВ-Петербург, 2006. — 400 с.: ил.

ISBN 5-94157-495-9

В книге рассмотрены возможности языка Visual Basic на основе авторских задач и примеров. Описывается история языков семейства Basic, применение Visual Basic к реализации линейных, разветвляющихся и циклических алгоритмов, работа с подпрограммами и файлами, мультимедийные возможности языка при оформлении созданных приложений, написание простых игр. Каждая из рассматриваемых тем предваряется коротким теоретическим вступлением, поясняющим приведенные примеры и задачи. В конце книги дан справочник по языку и решения избранных задач.

Для учащихся 8—11 классов, студентов первых курсов и преподавателей школ и вузов

УДК 681.3.06
ББК 32.973.26-018.1Visual Basic

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Людмила Еремеевская*
Зав. редакцией	*Григорий Добин*
Редактор	*Елена Сухогузова*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 15.11.05.
Формат 60×90 1/16. Печать офсетная. Усл. печ. л. 25.
Тираж 5000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-495-9

Оглавление

Предисловие
Приятного аппетита!

"Работа программиста и шамана имеет много общего — оба бормочут непонятные слова, совершают непонятные действия и не могут объяснить, как оно работает".

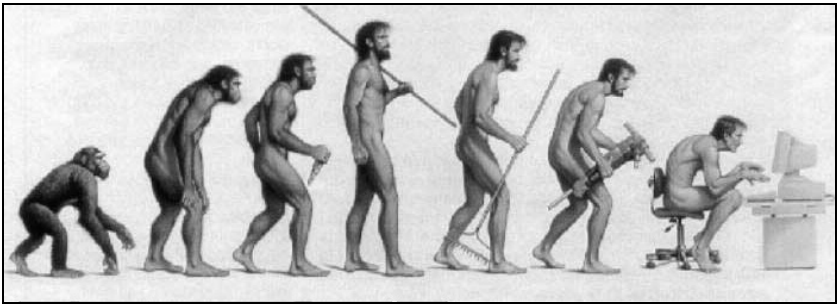
Анекдот на тему

Для пробуждения аппетита к чтению и работе с этой книгой хочу ознакомить вас с некоторыми интересными подробностями языка Visual Basic (VB) и соображениями о его полезности. Хотелось бы, чтобы эта книга ассоциировалась с вкусным и полезным обедом, каждое блюдо которого было бы насыщено витаминами и радостью от творческих побед.

В качестве эпиграфа одна интересная цитата из журнала "Наука и жизнь" (№11, 2003):

> "Способности к работе с компьютером обнаружили ученые у высших приматов, в частности, у бабуинов. В ходе исследований, проводившихся в Стэмфордской зоологической школе (США), выяснилось, что, работая с компьютером, обезьяны могут тестировать программное обеспечение и даже заниматься программированием. Правда, у них возникают трудности, когда по ходу работы приходится обращаться к меню, содержащему более двух уровней.
>
> Однако ученым удалось разрешить эту проблему. Оказалось, что если "многоходовые" действия ведут к некоей заветной картинке, самец бабуина способен осознать и запомнить до семи уровней в меню. Попутно во время работы выяснился забавный факт: как только бабуин научается нажимать нужные клавиши или пользоваться сложными меню, его социальный статус среди сородичей резко возрастает.
>
> После простейшего курса по работе с Windows большинство "студентов" освоило язык программирования **Visual Basic 3.0**.

В результате обезьяны смогли самостоятельно менять программные настройки и даже редактировать параметры атрибутов файлов. Однако освоить язык программирования Java не смог ни один из приматов".



Вперед, к Visual Basic!

Введение

Зачем я привел эту цитату, и, вообще, зачем я взялся за эту книгу?

В свое время я написал книгу "Бейсик в примерах и задачах". Мне говорили, что Бейсик уже умер, он никому не нужен, но я как действующий ☺ преподаватель информатики знал истинное положение вещей при обучении детей в школе и настоял на своем. В результате книга состоялась, несколько раз допечатывалась, и сейчас готовится ее второе издание. Бейсик, на мой взгляд, идеальное средство для ознакомления с алгоритмизацией, и я вижу, что никакие программные пакеты или готовые игры не заменят продукта, созданного своими руками. В глазах юного программиста читается гордость за содеянное, оно демонстрируется друзьям и родственникам, а тем остается только завидовать, потому что программирование — это творчество.

Недавно я стал замечать, что ряд студентов, моих бывших учеников, обращается ко мне с просьбой помочь с программированием на Visual Basic. То есть наметился переход к преподаванию в вузах этого замечательного языка программирования. И это тоже подтолкнуло меня к написанию данного задачника.

Ну, а кроме того, мне всегда хотелось написать книгу для начинающих программистов, книгу с человеческим лицом, да так, чтобы не оттолкнуть их первыми же сильно умными и непонятными словами. Поэтому принцип этого задачника будет следующий: я привожу подробно разобранный пример и прошу вас затем решить ряд однотипных с этим примером задач. В конце книги будет приведен краткий справочник по языку, а дальнейшее, конечно, будет зависеть только от вас. Ну, а если возникнут трудности, меня всегда можно будет найти по адресу **old_matros@mail.ru**

(после old стоит знак подчеркивания, а не пробел). Ну а тем, кто уже знаком с любой из обычных версий языка Basic, будет намного проще — все повторяется.

Кроме того, отдельное спасибо хочу сказать всем, кто помог мне в создании этой книги: любимой жене — за то, что не мешала, сыну и дочери — за то, что они есть, и моему издательству БХВ — за долготерпение.

Немного истории

Язык Basic был разработан преподавателями Дартмутского колледжа Джоном Кемени и Томасом Куртцом в 1964 году как средство обучения и работы начинающих программистов. (Дартмутский колледж в штате Нью-Гампшир, США, был создан в середине XVIII века, это одно из старейший высших заведений Америки.) Предназначение этого языка определено в самом его названии, которое является аббревиатурой слов Beginner's All-purpose Symbolic Instruction Code ("многоцелевой язык символических инструкций для начинающих") и при этом в дословном переводе означает "базовый". Тут создатели языка видимо провели историческую параллель с миссионерами-англичанами, которые несли христианские ценности во все новые и новые британские колонии, а средством общения сделали Basic English ("базовый английский"), включивший в себя 300 самых распространенных и простых для усвоения туземцами слов английского языка.

Замечание

Раньше языки программирования писались обязательно строчными буквами — BASIC, FORTRAN, COBOL. В 1990 году Международная организация стандартов приняла решения, что они пишутся как обычные имена собственные (прописной является только первая буква).

Однако парадокс заключается в том, что, будучи действительно весьма простым средством программирования, совершенно непригодным в те времена для решения серьезных задач, Basic представлял собой качественно новую технологию создания программ в режиме интерактивного диалога между разработчиком

и компьютером. То есть представлял собой прообраз современных систем программирования. Другое дело, что решение подобной задачи на технике тех лет было возможно только за счет максимального упрощения языка программирования и использования транслятора типа "интерпретатор".

Замечание

Интерпретатор — это такой синхронный программный "переводчик" алгоритмического языка на язык машинный. Его достоинство — получение результата выполнения программы сразу же (до первой ошибки ☺). Недостаток — увы, медлительность.

Резкое развитие систем на основе Basic началось с появлением в начале 80-х годов XX века персональных компьютеров, производительность и популярность которых растет вот уже двадцать лет невиданными темпами.

QuickBasic против TurboBasic

В конце 80-х годов насчитывалось около десятка систем Basic различных фирм-разработчиков. Однако главная борьба шла между QuickBasic (компания Microsoft) и TurboBasic (Borland). Вообще-то, конкуренция между этими двумя разработчиками средств программирования шла по целому спектру языков: Basic, Pascal и C. И результатом ее в 1989 году стало неявное мировое соглашение, когда Microsoft отказалась от дальнейшей поддержки Pascal, а Borland — Basic.

Тогда многие комментаторы язвительно замечали, что Microsoft отказалась от Pascal в пользу Basic исключительно из-за личных пристрастий основателя и руководителя корпорации Билла Гейтса. Действительно, разработка в 1975 году интерпретатора Basic для микроЭВМ Altair 8800 была первым проектом двадцатилетних Билла Гейтса и Пола Аллена, только что основавших фирму Microsoft (в тот момент они были единственными сотрудниками новой компании). С тех пор президент Microsoft постоянно участвовал в стратегии разработок Basic-систем корпорации и до сих пор, перечисляя свои титулы, Билл Гейтс довольно часто добавляет "Basic-программист". Но "отцом Visual Basic" считается Алан Купер — независимый программист, ко-

торый в конце 80-х годов разработал, а затем и продал фирме Microsoft прототип механизма визуального проектирования форм для Basic.

Однако победа QuickBasic определялась чисто технологическими причинами. В этой системе была удачно реализована схема смешанного использования традиционных Basic-технологий и классических методов создания сложных программных систем. Отметим, что с 1990 года усеченный вариант QuickBasic под названием QBasic был включен в состав MS-DOS.

Замечание

Многие современные пользователи ошибочно думают, что QuickBasic и QBasic — это одно и то же.

Эпоха Visual Basic

В начале 90-х годов Microsoft начала активную борьбу за продвижение в массы своей новой операционной системы Windows (против своей же, но более уже устаревающей MS-DOS). Но, как известно, пользователи работают не с операционной системой (ОС), а с программами, которые работают в ее среде. Поэтому скорость смены платформы в основном определяется темпами появления соответствующих прикладных программ.

Однако смена ОС представляет серьезную проблему и для программистов, так как им нужно было осваивать новую технологию разработки программ. В тот момент бытующим (и в значительной степени, совершенно справедливым) мнением было то, что Windows предъявляет более высокие требования к квалификации программиста.

В 1991 году под лозунгом "теперь и начинающие программисты могут легко создавать приложения для Windows" появилась первая версия нового инструментального средства Microsoft Visual Basic. В тот момент Microsoft достаточно скромно оценивала возможности этой системы, ориентируя ее, прежде всего, на категорию начинающих и непрофессиональных программистов. Основной задачей тогда было выпустить на рынок простой и удобный инструмент разработки в то время еще довольно новой

среде Windows, программирование в которой представляло проблему и для опытных специалистов.

Действительно, Visual Basic 1.0 в тот момент был больше похож не на рабочий инструмент, а на действующий макет будущей среды разработки. Его принципиальное новшество заключалось в реализации идей событийно-управляемого и визуального программирования в среде Windows, которые весьма радикально отличались от классических схем разработки программ. По общему признанию Visual Basic стал родоначальником нового поколения инструментов, называемых сегодня средствами быстрой разработки программ (Rapid Application Development, RAD). Сегодня эта идеология считается привычной, но тогда она казалась совершенно необычной и создавала серьезные проблемы (в том числе чисто психологического плана) для программистов "старых времен".

Тем не менее, число Visual Basic-пользователей росло, причем во многом за счет огромной популярности ее предшественника — QuickBasic. При этом Visual Basic быстро "мужал", усиливаясь за счет как развития среды программирования, так и включения профессиональных элементов языка и проблемно-ориентированных средств. И к моменту выпуска в 1995 году Visual Basic 4.0 эта система была уже признанным и одним из самых распространенных инструментов создания широкого класса приложений. В настоящее время используется версия Visual Basic 6.0, появление версии 7.0 ожидается в ближайшем будущем.

Visual Basic for Applications

В начале 90-х годов наметилась отчетливая тенденция включения в приложения, предназначенные для конечного пользователя, средств внутреннего программирования, которые должны были решать задачи настройки и адаптации этих пакетов для конкретных условий их применения.

В конце 1993 года Microsoft объявила о намерении создать на основе Visual Basic новую универсальную систему программирования для прикладных программ, которая получила название Visual Basic for Applications (VBA, Visual Basic для приложений). Естественно, реализацию этого проекта она начала с собственных офисных пакетов.

Первый вариант VBA 1.0 появился в составе MS Office 4.0, но лишь в программах Excel 4.0 и Project 6.0. В других же приложениях — Word 6.0 и Access 2.0 — были собственные варианты Basic. Более того, VBA 1.0 довольно сильно отличался (причем имея ряд существенных преимуществ) от используемой тогда универсальной системы Visual Basic 3.0.

Качественный перелом наступил в конце 1996 года с выпуском MS Office 97, в котором была реализована единая среда программирования VBA 5.0, включенная в программы Word, Excel и PowerPoint. Более того, VBA 5.0 использовала тот же самый языковый механизм и среду разработки, что и универсальная система Visual Basic 5.0. В состав MS Office 2000 вошла соответственно версия VBA 6.0, которая используется в шести программах — Word, Excel, PowerPoint, Access, Outlook, Frontpage.

В результате последние три года Microsoft позиционирует свой пакет MS Office не просто как набор прикладных программ, а как комплексную платформу для создания бизнес-приложений, решающих широкий круг специализированных задач пользователей. Именно этим объясняется появление в его составе специального выпуска для разработчиков приложений — Developer Edition.

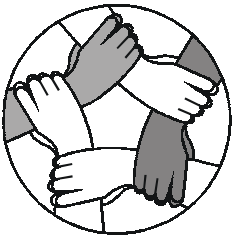
Одновременно VBA активно продвигает в качестве отраслевого стандарта для управления программируемыми приложениями, объявив о возможности его лицензирования. Сегодня уже более ста ведущих мировых фирм-разработчиков прикладных программ (среди них есть и российские) приобрели на него лицензии и включают его в состав своих программных продуктов.

Однако нужно подчеркнуть, что несмотря на доминирование VB и VBA на рынке программных средств, этот инструмент не является единственным примером использования языка Basic. В частности, в конце 90-х годов значительное распространение получила разработка компании Sax Software — механизм Sax Basic Engine, своеобразный "облегченный" вариант Visual Basic 6.0, успешно используемый многими разработчиками для автоматизации своих приложений.

Не начать ли с "Васика"?

Из сказанного ранее можно сделать следующий вывод. Освоение механизма программирования на VBA, реализованного в офисном приложении, которое установлено на вашем компьютере, откроет вам возможность использования полученных знаний и навыков при работе с десятками и сотнями других программ, в том числе и тех, которых пока еще нет на свете. Начав с составления простейших макрокоманд, при желании можно в рамках одного инструментария стать профессионалом, разрабатывающим программные системы любой сложности. Не говоря уже о том, что после освоения технологии разработки приложений смена инструментария не будет составлять серьезных проблем.

Десять лет назад во всем мире было не более двух миллионов программистов. Сегодня их насчитывается около пятнадцати миллионов, из них не менее 70 процентов используют в качестве хотя бы одно из инструментов VB или VBA.



Глава 1

Начинаем наш обед. Первое

"Откушав" первое блюдо, вы узнаете о том, где взять Visual Basic и как начать работу с первым проектом, узнаете о форме и инструментах, научитесь линейно программировать и красиво оформлять ваши проекты.

1.1. Где взять и как запустить

Ну, где в нашей стране берут программы (?) — все по-разному. Одни покупают, другие берут у друзей, а некоторые так ☺.

Я лично купил в магазине на Невском проспекте. Установил на компьютер, а ярлык поместил на рабочий стол. Он выглядит вот так (рис. 1.1).

Рис. 1.1. Ярлык для Visual Basic 6.0

Теперь щелкаем по ярлыку (кто-то два раза, а кто-то и один — как настроено!), и загружается редактор языка. Я надеюсь, дорогой читатель, что у вас уже есть какие-то навыки программирования, хотя бы в обычном QBasic, и поэтому я буду меньше

останавливаться на алгоритмических конструкциях (они, собственно, остались теми же самыми), а больше буду уделять времени реализации алгоритмов в редакторе Visual Basic. Как говорил один мой хороший знакомый программист: "Язык лишь средство. Самое трудное разработать алгоритм".

Итак, окно редактора после запуска выглядит следующим образом (рис. 1.2).

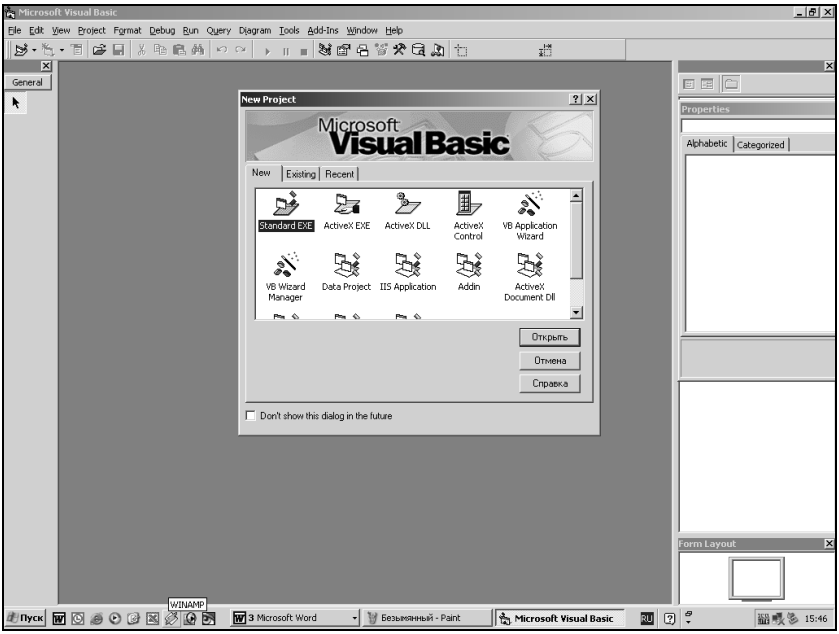


Рис. 1.2. Окно редактора

Окно как окно — все вроде бы пока стандартно для приложений Windows: заголовок, меню, панели инструментов Все подписано по-английски, но это полезно, всякий программист должен знать несколько слов на этом языке (типа, "wait" или "game over" ☺).

Для своего первого проекта из предложенных вариантов выберем Standard EXE (рис. 1.3).

Рис. 1.3. На подступах к новому проекту

1.2. Почему проект?

Раньше мы писали программы, а то, что мы будем делать здесь, будет называться проектом. Почему так? Когда мы переводим наш алгоритм на языки, например, QBasic, Pascal, C++ и другие, то в результате получаем текст программы, которую, в свою очередь, можем запустить и полюбоваться на ее работу. В Visual Basic при воплощении вашего алгоритма в жизнь необходимо создать несколько взаимосвязанных частей — одну или несколько экранных форм и один или несколько программных модулей. Поэтому эта большая работа будет носить гордое название "Проект" и сохраняться в файле с расширением vbp (Visual Basic Project).

1.3. Первый проект "«Зенит» — чемпион!"

Примечание

Чтобы не обижать болельщиков других команд, сразу скажу, что вы можете делать проект "«Локомотив» — чемпион!", "«ЦСКА» — чемпион!" или даже "«Спартак» — чемпион!" ☺

В этом разделе мы рассмотрим следующие темы:

- создание, сохранение, запуск, остановка проекта. Создание из проекта Exe-приложения;
- инструмент **Label**, **Image**, **CommandButton**;
- свойства `Caption`, `Left`, `Top`, `Width`, `Height`, `Font`, `BackColor`, `ForeColor`, `Alignment`, `Scretch`;
- если пропала панель инструментов **Standard**;
- если пропало окно свойств **Properties**.

После запуска Standard EXE мы получаем такую интересную картинку (рис. 1.4).

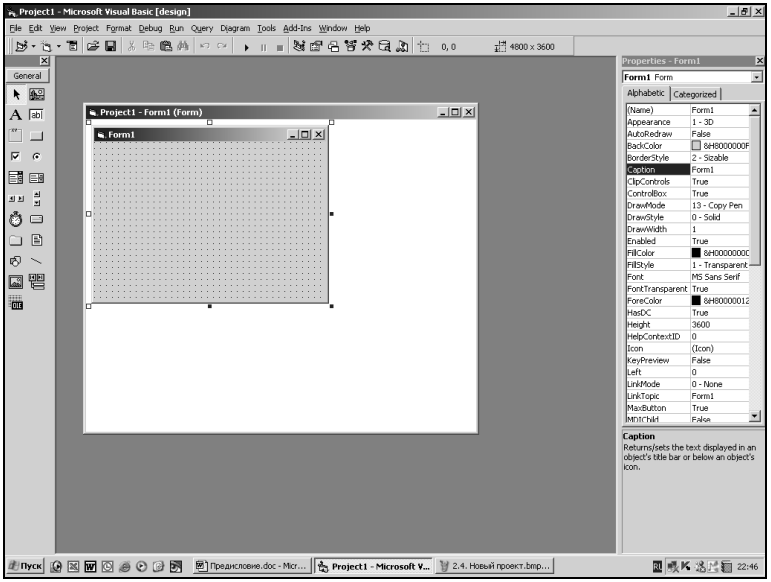


Рис. 1.4. Новый проект

В общем, те, кто хоть что-нибудь уже делал в приложениях Windows, ничего принципиально нового не увидят. Заголовок окна **Project1-Microsoft Visual Basic [design]**, ниже — строка меню (**File**, **Edit**, **View** и т. д.), еще ниже — панель инструментов **Standard** (если вы вдруг ее случайно куда-то перетащили, то восстановить можно из меню **View | Toolbars | Standard**), слева — набор базовых компонентов **General**, справа — окно свойств объекта **Properties-Form1** (в нашем случае они относятся к форме **Form1**), ну, а по центру — непосредственно сама форма, которую мы и будем проектировать.

Замечание

Если случайно закрыли окно свойств, то следуйте **View | Property Pages** или нажмите клавишу <F4>.

Кроме того, в окне свойств есть две вкладки: первая — **Alphabetic**, где свойства расположены по алфавиту, вторая — **Categorized**, где свойства сгруппированы по категориям.

Итак, первый проект до безобразия элементарен. Он будет запускать никак не оформленную форму, на которой будет находиться надпись "«Зенит» — чемпион!" (или любая другая, какая вам больше понравится). Закрываться наша форма после запуска будет как обычное окно Windows. Итак, за работу!

1.3.1. Инструмент Label

Возьмем на панели **General** инструмент **Label** (выглядит он так ), который позволяет выводить на форме любой текст, и растянем на форме прямоугольник. Должно получиться так (рис. 1.5).

Теперь в окне свойств **Properties-Label1** (следите только, чтобы нужный объект на форме был выделен) найдем свойство `Caption` и изменим его, написав вместо "Label1" — "«Зенит» — чемпион!".

Теперь форма должна выглядеть вот так (рис. 1.6).

Все! Первый проект готов. Аплодисменты в студию! Но, чтобы не пропал наш скорбный труд, надо обязательно перед запуском проекта его сохранить.

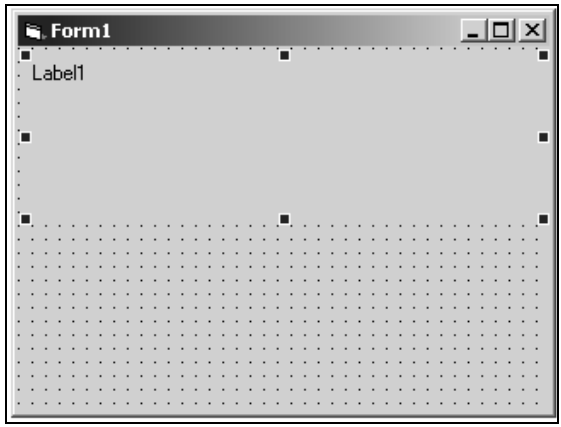


Рис. 1.5. Метка на форме

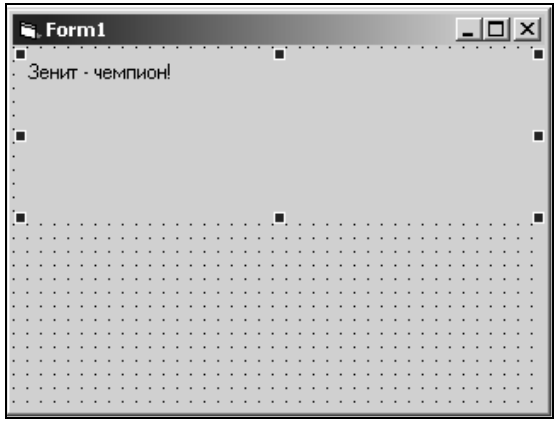


Рис. 1.6. "Зенит" — чемпион!

1.3.2. Сохранение проекта

В меню **File** выбираем команду **Save project** или нажимаем на стандартной панели инструментов кнопку — **Save**. В появившемся окне выбираем свою папку, создаем в ней папку для нашего первого проекта (так давайте и назовем ее — Первый проект) и сохраним туда нашу форму (Form1.frm) и проект (Project1.vbp).

Теперь можно запускать.

1.3.3. Запуск проекта

Запускается проект клавишей <F5> или из меню **Run | Start** или нажатием на стандартной панели инструментов кнопки ▶ (правда, похоже, как на плеере ☺).

И вот наш первый проект в работе. Ничего, что он пока такой серенький и невзрачный, как гадкий утенок. Пройдет совсем немного времени, и он будет красив и достоин тех слов, которые на нем красуются!

Остановим выполнение проекта. Лучше это делать через меню **Run | End** или нажатием на стандартной панели инструментов кнопки ■.

1.3.4. Создание Exe-приложения

Прежде чем мы займемся красотой нашего проекта, предлагаю ознакомиться с возможностью создания из него полноценного Exe-приложения, которое сможет запускаться безо всякого участия оболочки Visual Basic.

Выберем в меню команду **File | Make Project1.exe** В появившемся окне сохранения выберем нужную папку (в нашем случае пусть будет все та же папка Первый проект) и дадим имя файлу (пусть будет zenit), после чего нажмем кнопку **OK**.

1.3.5. Запуск Exe-приложения

Теперь давайте закроем Visual Basic, найдем и откроем папку Первый проект и двойным щелчком запустим файл zenit.exe. Вы видите, что наша форма загружается и работает безо всякого внешнего воздействия ☺.

Теперь закроем форму и снова загрузим Visual Basic. Откроем уже существующий проект. Для этого выполним команду из меню **File | Open Project** или нажмем на стандартной панели инструментов кнопку 📂 — **Open**, выберем там нужную папку и файл Project1.vbp.

Замечание

Если вдруг стало отсутствовать окно обзора Проекта, то выберите в меню **View | Project Explorer** или давите клавиши <Ctrl>+<R>.

Займемся свойствами формы. Выделите ее. Почему у нее в заголовке написано безликое Form1? Непорядок! Изменим свойство `Caption` с Form1 на Футбол. Уже лучше.

Теперь займемся размерами объектов.

1.3.6. Установка положения и размеров объекта

Размеры объекта задаются свойствами `Width` (Ширина) и `Height` (Высота). По умолчанию, эти величины задаются в специальных единицах — твипах. *Один твип* — это 1/1440 логического дюйма. *Логический дюйм* — это такое расстояние на экранной форме, которое при печати на принтере будет равным 1 дюйму (1 дюйм = 2,54 см). Вы можете видеть на экранной форме сетку — ряды точек. Сетка помогает точно размещать на форме объекты. По умолчанию расстояние между соседними точками составляет 120 твипов.

Положение формы и объектов на ней тоже определяется в твипах. На стандартной панели инструментов справа вы можете видеть две пары чисел, которые называются *индикатором положения и размеров* выделенного объекта (рис. 1.7).

Рис. 1.7. Индикатор положения и размеров

Первая пара чисел указывает в данном случае положение формы относительно левого верхнего угла экрана (по горизонтали форма после запуска будет отстоять на 0 твипов, по вертикали от того же угла тоже на 0 твипов).

Вторая пара чисел задает размеры формы в твипах по горизонтали (в нашем случае 4800) и вертикали (3600).

Таким образом, положение формы на экране после запуска можно регулировать свойствами `Left` и `Top`, а размеры, соответственно, `Width` и `Height`.

Для приблизительного, на глазок, изменения положения формы можно воспользоваться окном, вызываемым через меню **View** |

Form Layout Window, которое появится в районе **Properties**. На нем вы будете видеть экран и схематичное положение формы, которое мышью сможете подвинуть туда, куда хотите. Числа на индикаторе положения тут же изменятся.

Размеры формы также легко меняются мышью, как, впрочем, и размеры любого окна Windows.

Теперь об объектах, расположенных на форме. Когда они выделены, то индикатор положения показывает первой парой чисел положение объекта относительно верхнего левого угла формы, второй парой — размеры объекта.

Точно выставить эти позиции можно через те же свойства, что и для формы.

Давайте вернемся к нашему проекту и установим следующие свойства для формы и объекта Label (табл. 1.1).

Таблица 1.1. *Свойства объектов Form и Label*

Свойство, Объект	`Left`	`Top`	`Width`	`Height`
Экранная форма	3000	3000	6000	4000
Label	240	120	5415	1215

Замечание

Если вдруг захотелось измерять размеры объектов не в твипах, а в чем-либо еще, то находите для формы свойство `ScaleMode` и выбирайте из вариантов, представленных на рис. 1.8 (числа на индикаторе положения и размеров будут представлены в этой же размерности).

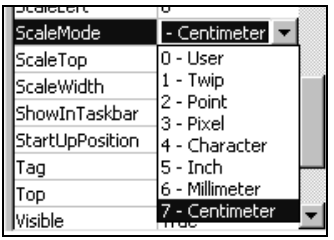


Рис. 1.8. ScaleMode

1.3.7. Цвета и шрифты объектов

Цвета... Как-то все в серых тонах, для чемпиона не пафосно! Используем свойство `BackColor`. Воспользуйтесь им, выбрав закладку **System** (Системные цвета) или **Palette** (Выбор из палитры). Выберите подходящие на ваш взгляд цвета для формы и объекта Label.

Шрифт для нашего гордого текста в объекте Label — это, соответственно, свойства `Font` (шрифт, начертание, размер) и `ForeColor` (цвет шрифта). Установите для нашего проекта: цвет шрифта — белый, шрифт — MS Sans Serif, начертание — жирный, размер — 24.

Осталось выравнивание текста внутри нашего объекта. Это будет свойство `Alignment` — выберите Center.

Запустим проект и увидим, как хорошо он расположился на экране и как он прекрасно выглядит ☺ (рис. 1.9)!

Сохраним его под тем же именем.

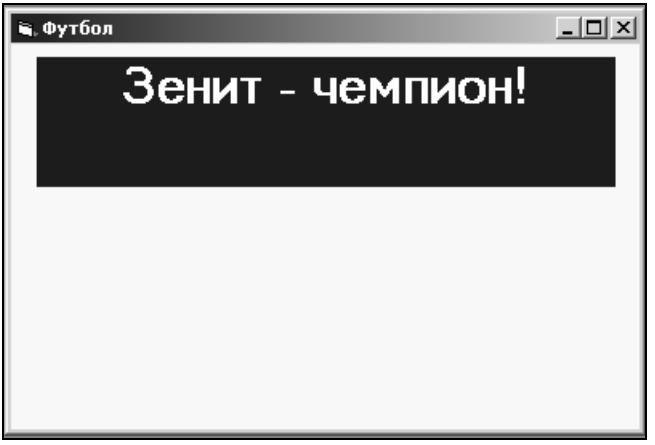


Рис. 1.9. Слегка оформленный объект

1.3.8. Картинки на форме

Продолжаем наводить красоту. Хочется приукрасить все это дело картинкой. Сразу вопрос: "Где взять?" Можно из библиотеки

стандартных картинок или с дисков с ClipArt'ами, можно из Интернета, а можно и самим нарисовать. Важно, чтобы рисунок был у вас на диске в виде файла.

Я бы сделал так: взял бы инструмент **Image** с панели **General** и растянул бы в правом нижнем углу формы прямоугольник для вставки подходящей картинки. Установил бы свойства: `Left` — 240; `Top` — 1440; `Width` — 1455; `Height` — 2055.

И новое для нас свойство — `Scretch`. Установите для него значение True. В этом случае картинка растянется по отведенному ей месту.

Замечание

Правда, в таком случае могут быть нарушены ее пропорции!

После установки свойств можно войти в свойство `Picture` и найти там файл с вашей картинкой. У меня получилось вот так (рис. 1.10).

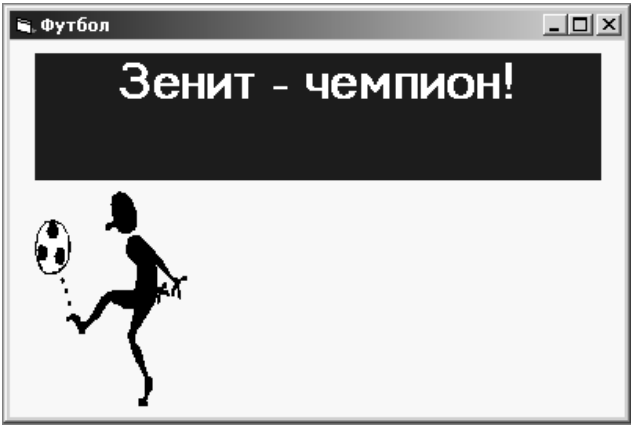


Рис. 1.10. Проект с картинкой

1.3.9. Командные кнопки на форме и в проекте

Теперь пора сделать проект хоть немножко управляемым. То есть от оформления формы мы переходим непосредственно

к ее программированию. И вот тут-то и пригодится знание старого доброго Quick или любого другого Basic'a.

Сейчас наша форма закрывается как обычное окно Windows, но мы будем закрывать ее с помощью специальной командной кнопки.

Итак, разместим на форме CommandButton командную кнопку  со свойствами положения и размеров: `Left` — 2040; `Top` — 2880; `Width` — 1815; `Height` — 615.

Выберем теперь какой-нибудь подходящий `BackColor`. Ой, цвет кнопки не изменился! Паника в проекте! Это все потому, что не установлено свойство `Style` — Graphical. Теперь все нормально, надеюсь?

Примечание

Я долго искал и в справке, и во всевозможной литературе, и в Интернете, и у своих коллег информацию об изменении цвета шрифта на командной кнопке. Не нашел ответа. Может быть, кто-то из вас найдет, сообщите, пожалуйста, а то ведь автору плохо спится ☺.

Установите размер шрифта — 12 (Font), поменяйте Caption на слово "Закрыть".

Вот так у вас получилось (рис 1.11)?

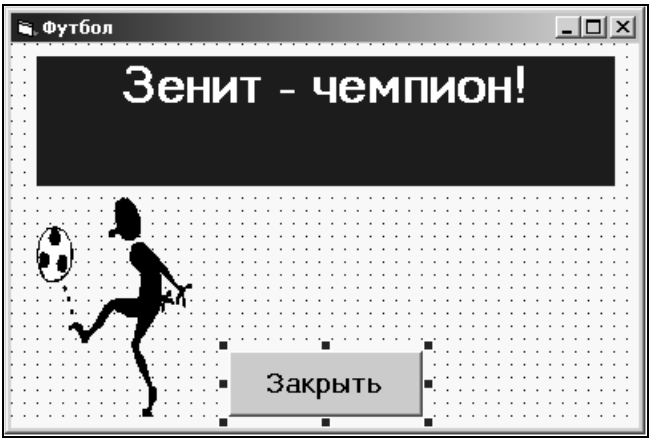


Рис. 1.11. Форма с командной кнопкой

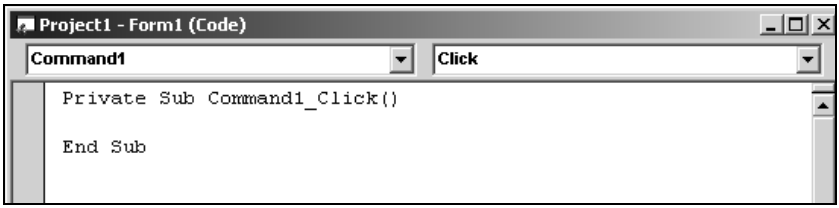


Рис. 1.12. Окно программного кода

Рис. 1.13. Процедура закрытия формы

А теперь напишем для этой кнопки наш первый командный код. Дважды щелкнув по кнопке "Закрыть", мы попадаем в окно **Project — Form1 (Code)** (рис. 1.12).

Как вы видите, все наши описания событий, происходящих в проекте, будут представляться в виде процедур, начало и окончание которых нам любезно предоставляется. В данном случае нам надо описать, что будет происходить с формой после ее запуска по щелчку мышью по кнопке "Закрыть". Мы хотим, чтобы в этом случае форма просто закрывалась. Это описывается очень короткой, но емкой командой `End`, которую мы и впишем в предоставленное нам место (рис. 1.13).

Запустим проект и щелкнем по кнопке "Закрыть". Ой, закрылось! И все это вашими замечательными руками и головой!

Разместим на форме еще одну командную кнопку (повыше предыдущей) со свойствами: `Left` — 2040; `Top` — 2800; `Width` — 1815; `Height` — 615; `Caption` — Показать; `Style` — Graphical и каким-нибудь `BackColor` (рис. 1.14).

Мы хотим теперь, чтобы после запуска проекта на форме были бы только командные кнопки и все, а потом, по нажатию кнопки "Показать", увидели бы нашу надпись и картинку (необычайно красивые!).

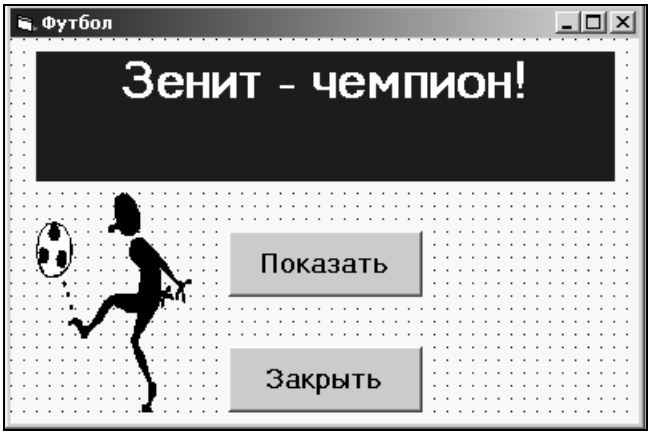


Рис. 1.14. Форма с двумя командными кнопками

Во-первых, чтобы текст и картинка изначально не были видны, предварительно их выделив, установите для них свойство `Visible` — False.

Теперь щелкните дважды по кнопке "Показать" и напишите следующий командный код (рис. 1.15).

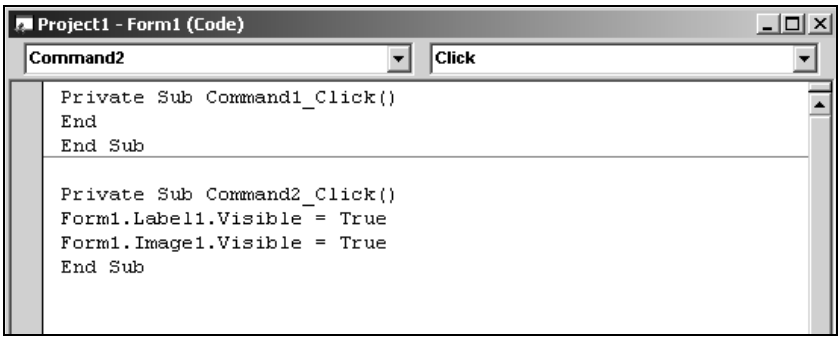


Рис. 1.15. Процедура показа

Обратите внимание, что для того чтобы увидеть тот или иной объект, мы его указываем следующим образом: сначала `Name` (имя) формы Form1, затем через точку `Name` (имя) объекта Label1, затем через точку его свойство `Visible`, которое мы этой командой и делаем `True`.

Замечание

Так как форма в нашем проекте одна, то можно было бы не указывать в этих командах имя формы.

Запустите теперь проект. Ничего не видно. А теперь щелкните по кнопке "Показать" — победа человеческого разума!

Не забудьте сохранить эту работу и показывать ее всем с гордостью!

А теперь порешаем задачи.

Замечание

Нумерация задач сквозная по всей книге.

1. Создайте и оформите проект, аналогичный приведенному в примере со второй картинкой, расположенной справа от командных кнопок (рис. 1.16).

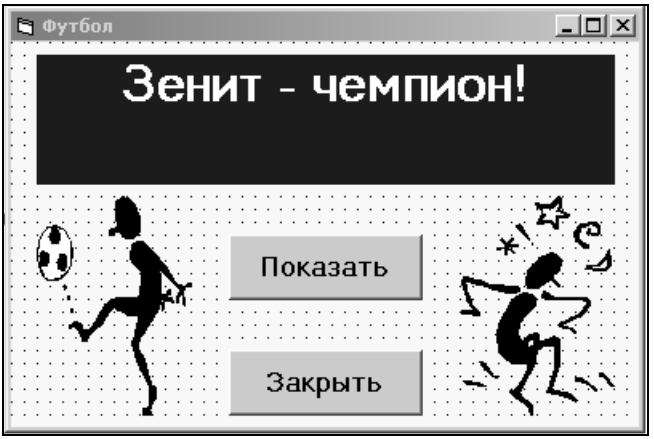


Рис. 1.16. Проект с двумя картинками

2. Создайте и оформите проект, показывающий по нажатию одной командной кнопки животное с надписью (например, "Это — лев"), а по нажатию другой командной кнопки — другое животное с надписью (рис. 1.17).

Рис. 1.17. Мини-зоопарк

3. Создайте и оформите проект, по нажатию командной кнопки показывающий знак дорожного движения и комментарий о его назначении (рис. 1.18).

Рис. 1.18. Знаки дорожного движения

4. Создайте и оформите проект, выводящий какой-нибудь из неправильных английских глаголов, а затем, по нажатию командных кнопок, соответствующие формы этого глагола (рис. 1.19).

Рис. 1.19. Неправильный глагол

5. Нарисуйте шесть позиций игрального кубика с выпавшими цифрами 1, 2, 3, 4, 5 и 6 соответственно, а затем создайте и оформите проект, выводящий соответствующие картинки по нажатию командных кнопок 1, 2, 3, 4, 5 или 6 (рис. 1.20).

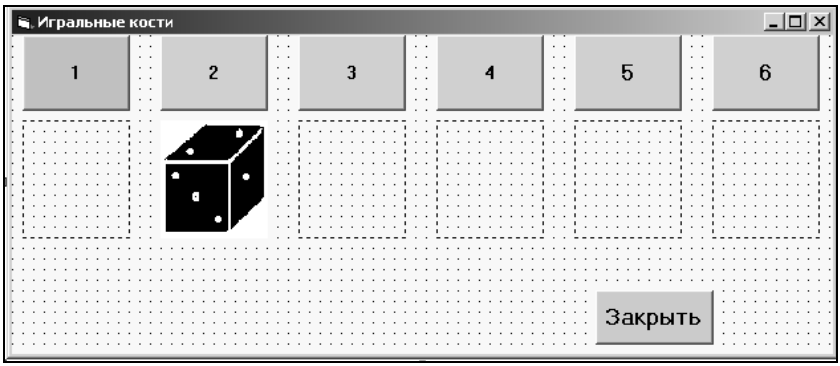


Рис. 1.20. Игральные кости

1.4. Линейное программирование

Слово *"компьютер"*, не устаю я напоминать юным программистам, происходит от слова "compute" — вычислять, а информация, обрабатываемая этим самым компьютером — сплошь *циф-*

ровая. Следовательно, необходимо научиться с помощью Visual Basic вычислять и обрабатывать эту самую цифровую информацию.

В этом разделе, как это ни прискорбно, будет много математики (рис. 1.21). Просьба подготовиться!

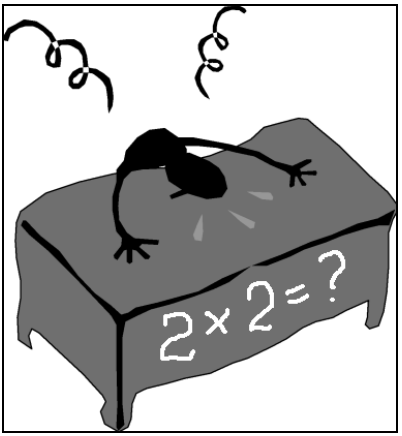


Рис. 1.21. Будет много математики!

Сначала пойдут простые вычисления. И все же, чтобы было чем вычислять, сделаем проект "Калькулятор". Да, предвижу ваши возражения: и на мобильнике есть, и в Windows в Стандартных программах, да где его только нет! Но мы же его сделаем своими руками, да еще можем навороченных функций каких-нибудь навставлять, которых на простом калькуляторе-то и нет!

1.4.1. Проект "Калькулятор"

На открывшейся форме (сделайте ее побольше, не мельчите — глазки надо жалеть ☺) сначала подготовим три текстовых поля (инструмент **TextBox**): два — для ввода чисел, над которыми будут производиться арифметические действия, и еще одно — для вывода результата вычислений. Инструмент **TextBox** на панели **General** выглядит так: ab|.

По умолчанию **TextBox** несет в себе сообщение text1, которое и будет присутствовать на экране, если мы не позаботимся его удалить. Выбираем свойство `Text` и все оттуда удаляем.

После чего сделайте на форме шесть командных кнопок: четыре — для арифметических действий, пятую — для очистки текстовых полей и еще одну — для закрытия формы.

Должно получиться примерно так, как на рис. 1.22. (Конечно, оформление очень желательно! Все любят красивое! А вы ведь уже умеете...)

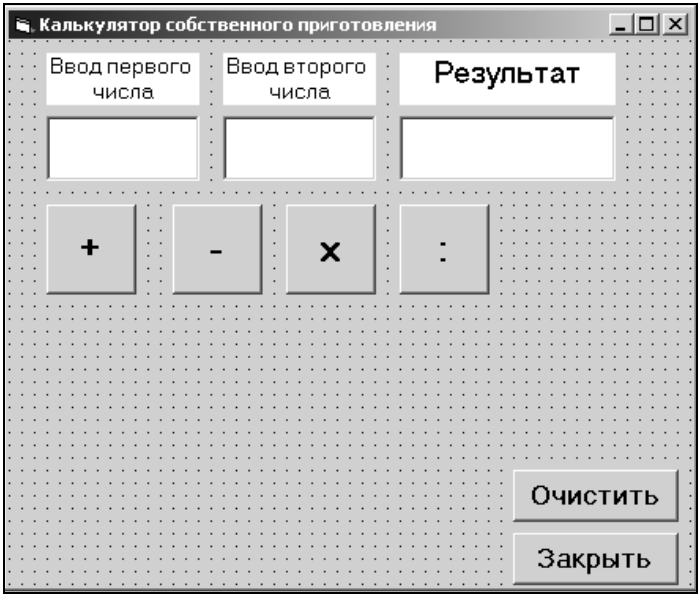


Рис. 1.22. Форма проекта "Калькулятор"

А теперь пишем командный код (пока только для кнопки "+"):

```
Private Sub Command1_Click()
Text3.Text = Text1.Text + Text2.Text
End Sub
```

Для кнопки "Закрыть" пишем заветное слово `End`, а для кнопки "Очистить" такой код (справа от знака равенства — две пары кавычек):

```
Private Sub Command5_Click()
Text1.Text = ""
Text2.Text = ""
```

```
Text3.Text = ""
End Sub
```

Теперь давайте запустим проект для проверки сложения. Вводим первое число, скажем, 1, а второе, например, 7. Нажимаем кнопочку "+". Получить должны 8, а получаем о, ужас! 17! Как же так?

Объясняю. То, что вводится в TextBox, после запуска проекта по умолчанию компьютером воспринимается как строка символов. Знак плюс (+) для символов — это действие *конкатенации* или склейки строк, вот он, родной, нам и приклеил к единице семь, получилась строка семнадцать (не число опять же). Как же быть? Надо как-то компьютеру объяснить, что мы с числами работаем. Для этого существует *функция перевода из текстовой переменной в числовую* — `Val` (происходит от англ. слова "value"). Исправляем наш программный код следующим образом:

```
Private Sub Command1_Click()
Text3 = Val(Text1) + Val(Text2)
End Sub
```

Для вычитания, умножения и деления пишем аналогичные процедуры. Арифметические знаки меняются: для вычитания используется дефис (–), для умножения — звездочка (*), для деления — косая черта (/).

Ну что же, вот калькулятор и готов. Запускаем, проверяем. Радуемся!

Можете сделать из него самостоятельное exe-приложение — это уже серьезно!!!

1.4.2. Вывод картинки в качестве фона формы

Если вдруг хочется сделать очень красиво (а ведь хочется, правда?), то в качестве фона формы можно использовать почти любой (в форматах bmp, wmf, jpg) графический файл, имеющийся у вас на компьютере. Для этого, по крайней мере, есть два способа.

- Непосредственно для выделенной формы выбрать свойство `Picture` и загрузить нужную картинку. Сложность здесь заключается в ручной подгонке размеров формы и картинки.

- Растянуть сначала на всю форму инструмент **Image**, выбрать затем свойство `Scretch` — True, затем свойство `Picture`, нужную картинку. Свойство `Scretch` растягивает картинку на всю рамку Image. Есть, правда, опасность нарушения пропорций, — ну, попробуйте с этим справиться самостоятельно!

Попробуйте теперь на нашу форму калькулятора установить рисунок. Может получиться примерно как на рис. 1.23.

Рис. 1.23. Форма с фоновым рисунком

1.4.3. Имена объектов формы

В нашем проекте задействованы шесть командных кнопок. Когда мы пишем для них процедуры, то каждая из них начинается словами `Private Sub Command1_Click` — это для первой командной кнопки, которая отвечает за сложение. Когда таких кнопок много, то немудрено запутаться, какая кнопка за что отвечает, поэтому прежде чем писать для них процедуры, рекомендуется давать осмысленные имена не только кнопкам, но

и другим объектам формы. Достигается это изменением свойства `Name` соответствующих объектов, например, Command1 можно было бы изменить на CmdPlus, Command2 — на CmdMinus и т. д. Соответственно и в процедуре будет уже вот так:

```
Private Sub CmdPlus_Click()
```

Здесь сокращение `Cmd` выступает в роли префикса, указывая всем видящим, что это не Label и не TextBox, а CommandButton.

Есть специальное соглашение, называемое *венгерским*, которое унифицирует сокращения для всех объектов Visual Basic. Я приведу для основных (табл. 1.2). "Старайтесь соблюдать конвенцию", — как говорили дети лейтенанта Шмидта ☺.

Таблица 1.2. *Принятые сокращения для объектов Visual Basic*

Название	Префикс	Пример
3D Panel	`pnl`	`pnlGroup`
Check box	`chk`	`chkReadOnly`
Combo box	`cbo`	`cboEnglish`
Command button	`cmd`	`cmdExit`
Data	`dat`	`datBiblio`
Directory list box	`dir`	`dirSource`
Drive list box	`drv`	`drvTarget`
File list box	`fil`	`filSource`
Form	`frm`	`frmEntry`
Frame	`fra`	`fraLanguage`
Horizontal scroll bar	`hsb`	`hsbVolume`
Image	`img`	`imgIcon`
Label	`lbl`	`lblHelpMessage`
Line	`lin`	`linVertical`
OLE container	`ole`	`oleWorksheet`
Option button	`opt`	`optGender`
Picture box	`pic`	`picVGA`

***Таблица 1.2** (окончание)*

Название	Префикс	Пример
Shape	`shp`	`ShpCircle`
Slider	`sld`	`sldScale`
Text box	`txt`	`txtLastName`
Timer	`tmr`	`tmrAlarm`
Vertical scroll bar	`vsb`	`vsbRate`

1.4.4. Функции для вычислений

Вообще, в Visual Basic, кроме четырех арифметических действий, можно использовать еще несколько полезных функций, которые представлены в следующей таблице (табл. 1.3).

***Таблица 1.3.** Функции для вычислений*

Словесное описание	В алгебре	В Visual Basic	Пример	Чему будет равно Z
Возведение в степень	x^y	`x^y`	`Z=5^3`	125
Извлечение квадратного корня	$\sqrt{x}$	`Sqr(x)`	`Z=Sqr(81)`	9
Извлечение корня n-степени	$\sqrt[n]{x}$	`x^(1/n)`	`Z=8^(1/3)`	2
Нахождение целочисленного остатка от деления	вручную ☺	`X mod Y`	`Z=23 mod 5`	3
Нахождение целочисленного частного от деления	вручную ☺	`X \ Y`	`Z=23 \5`	4
Модуль (абсолютная величина)	| X |	`Abs (X)`	`Z=Abs(-45)`	45
Синус	sin X	`Sin(x)`	`Z=sin(1.5)`	0.997...
Косинус	cos X	`Cos(x)`	`Z=Cos(2)`	0.416...

Таблица 1.3 *(окончание)*

Словесное описание	В алгебре	В Visual Basic	Пример	Чему будет равно Z
Тангенс	tg X	`Tan(x)`	`Z=Tan(0.5)`	0.546...
Арктангенс числа	arctg X	`Atn(X)`	`Z=Atn(0.5)`	0.463...
Целая часть числа	нет	`Fix(x)`	`Z=Fix(-2.9)`	–2
Целая часть числа	нет	`Int(x)`	`Z=Int(-3.1)`	–4
Перевод десятичного числа в шестнадцатеричное	нет	`Hex(x)`	`Z=Hex(543)`	21F
Перевод десятичного числа в восьмеричное	нет	`Oct(x)`	`Z=Oct(543)`	1037
Знак числа	нет	`Sgn(x)`	`Z=Sgn(-1.5)`	–1
Экспонента	e^x	`Exp(x)`	`Z=Exp(0.37)`	1.44...
Натуральный логарифм	ln X	`Log(X)`	`Z=Log(2.7)`	0.99...

Рис. 1.24. Тригонометрический калькулятор

Теперь предлагаю выполнить самостоятельные задания.

6. Спроектируйте и воплотите в жизнь красивый многофункциональный калькулятор, с использованием вышеописанных функций.
7. Спроектируйте тригонометрический калькулятор, где углы можно было бы вводить в градусной форме, а уж в программном коде калькулятор переводил бы их в радианы и вычислял значения (рис. 1.24).

1.4.5. Переменные и типы данных

Сначала о переменных. Надеюсь, вы уже понимаете, о чем идет речь. А если нет, напомню, что *переменная* — это временное хранилище для данных в вашем проекте, имеющее уникальное имя, по которому программа может отыскать эти данные в огромных просторах ОЗУ (оперативного запоминающего устройства) вашего ПК (персонального компьютера) ☺.

Имя переменной должно отвечать нескольким условиям:

- может начинаться только с латинской буквы;
- может состоять из латинских букв, цифр, знаков подчеркивания;
- имя не должно содержать пробел, точку, запятую и знаки !, %, &, #, $, @;
- не должно быть более 256 символов;
- не может быть ключевым словом Visual Basic;
- желательно должно быть описательным, то есть наиболее полно отражать содержимое переменной. Например, имя переменной MotherBirthDay гораздо более информативно, чем просто Mother или Birth или, тем более, Day;
- по соглашению, если имя составлено из нескольких слов, то каждое такое слово должно начинаться с заглавной буквы, например ZenitBestTeam ☺.

Ну и еще одно, прежде чем перейдем к вычислениям. Дело в том, что по умолчанию Visual Basic относит переменные к типу Variant. Хорошо это или плохо? С одной стороны, хорошо, потому что нам не надо об этом думать и голову свою, и так пере-

груженную, ломать. С другой стороны, как мы увидим чуть позже, тип Variant очень неэкономичный и, освободив свою голову, мы можем забить голову нашему ПК, который, в свою очередь, в лучшем случае медленнее станет работать, а в худшем зависнет (ну это правда, серьезную программу надо написать ☺).

Итак, какие типы переменных возможны, указано в табл. 1.4.

Таблица 1.4. *Типы переменных*

Наименование	Описание	Размер (в байтах)	Пример использования
Boolean (логический тип)	Только два значения: True или False	2	`Dim Flag As Boolean` `Flag=False`
Byte (байтовый)	Положительные числа без десятичных точек (Целые в диапазоне от 0 до 255)	1	`Dim NumMonth As Byte` `November=11`
Currency (Денежный)	Денежные значения от –$922337203685477,5808 до $922337203685477,5807. Четыре знака после запятой обеспечивают правильное округление.	8	`Dim Debet@` `Debet@=75.89`
Date/Time (Дата/время)	Значения даты и времени. Дата может находиться в диапазоне от 1 января 100 года до 31 декабря 9999 года.	8	`Dim NewYear As Date` `NwYear=#01/01/2005`
Double (числами с плавающей точкой двойной точности)	Значения в диапазоне от –1,79769313486232D+308 до 1,79769313486232D+308	8	`Dim E#` `E#=2.718`
Integer (целый)	Целочисленные значения в диапазоне от –32768 до +32767	2	`Dim Men%` `Men%=1000`

Таблица 1.4 *(окончание)*

Наименование	Описание	Размер (в байтах)	Пример использования
Long (длинное целое)	Целочисленные значения в диапазоне от –2147483648 до +2147483647 4 байта	4	`Dim China&` `China&=1 000 000`
Single (числа с плавающей точкой одинарной точности)	Численные значения в диапазоне от –3,402823E+38 до +3,402823E+38	4	`Dim Price!` `Price!=9999.99`
String (строковый)	Строки, состоящие из 0-654000 алфавитно-цифровых символов	1 байт на один символ	`Dim Cat$` `Cat$="Barsik"`
Variant (общий)	Для всех типов данных	16	`Dim Pan` `Pan=234.56`

Итак, желательно тип используемых в программе переменных описывать до их использования. Это способствует эффективности исполнения программы и экономии памяти компьютера.

Объявление переменных чаще начинается со слова `Dim`, после которого перечисляются через запятую имена переменных и после слова `As` — их тип. Например, `Dim First, Second As Integer`.

Замечание

Некоторые ошибочно думают, что если они объявили переменные следующим образом:

```
Dim First, Second, Third As Single
```

то все три переменные будут типа Single. Это в корне неверно! Переменные First и Second будут типа Variant и только переменная Third будет объявленного типа Single.

Правильно было бы объявить так:

```
Dim First As Single, Second As Single,
Third As Single
```

или так:

```
Dim First As Single
Dim Second As Single
Dim Third As Single
```

Переменные могут быть объявлены и с использованием специальных символов (см. табл. 3), например, `Dim First%, Second%`, но тогда и в программе надо их использовать с указанием этих символов.

Замечание

Если вы хотите, чтобы программа контролировала правильность написания используемых переменных и указывала на ошибки, можно в начале программы указать два заветных слова `Option Explicit` или в меню **Tools | Options** установить флажок в пункте **RequareVariable Declaration**.

Кроме того, сейчас стало признаком хорошего тона указывать тип переменных непосредственно в их имени. Вот список применяемых для этого сокращений (табл. 1.5).

Таблица 1.5. *Сокращения для типов данных*

Тип данных	Префикс (сокращение)	Пример
Boolean	`bln`	`blnPravda`
Byte	`byt`	`bytNota`
Currency	`cur`	`curPrice`
Date	`dtm`	`dtmPobeda`
Double	`dbl`	`dblMulti`
Integer	`int`	`intKolvo`
Long	`lng`	`lngZerno`
Single	`sng`	`sngWidth`
String	`str`	`strName`
Variant	`vnt`	`vntPan`

Ну что же, с учетом всего вышеперечисленного повычисляем.

Проект "Количество прожитых дней"

Проект предусматривает введение сегодняшней даты, даты рождения человека и выводит в окне количество прожитых им дней.

На форме разместим три TextBox (два — для ввода дат, одно — для вывода результата) и две командных кнопки (одну для расчета, другую для выхода). На рис. 1.25 приведена форма данного проекта.

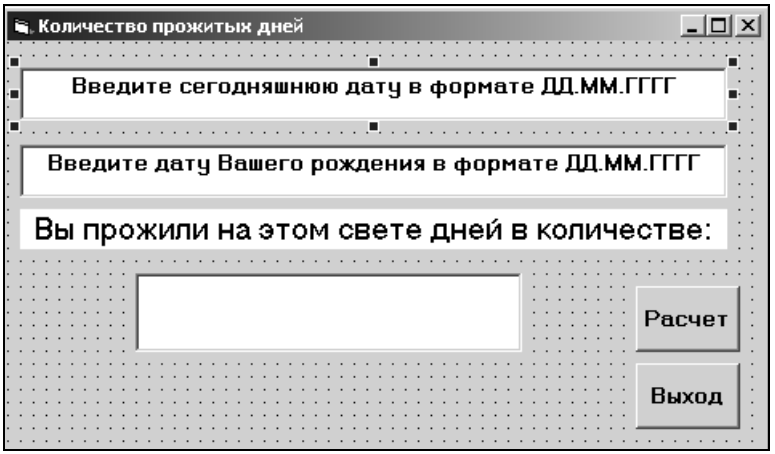


Рис. 1.25. Форма проекта "Количество прожитых дней"

Программный код будет выглядеть так: `dtmDate1` и `dtmDate2` — переменные, в которые поступают исходные даты; `dtmDays` — переменная, в которую записывается результат; TextBox'ы переименованы соответственно в txtDate1, txtDate2 и txtResult (в соответсвии с соглашением об именах см. табл. 1.2). Функция `Cdate` переводит текстовые значения дат в формат Дата.

```
Option Explicit
Dim dtmDate1 As Date
Dim dtmDate2 As Date
Dim dtmDays As Long
Private Sub cmdCompute_Click()
dtmDate1 = CDate(txtDate1.Text)
dtmDate2 = CDate(txtDate2.Text)
```

```
dtmDays = dtmDate1 - dtmDate2
txtResult.Text = dtmDays
End Sub
Private Sub cmdExit_Click()
End
End Sub
```

Результат работы программы приведен на рис. 1.26. Ох, скоро у кого-то очень круглый юбилей!

Рис. 1.26. Результат работы проекта "Количество прожитых дней"

Ну что ж, если вопросов нет, то переходим к очередным заданиям.

8. Разработать проект, запрашивающий дату рождения пользователя и рассчитывающий, в какой день им было прожито 5000 дней и в какой будет прожито 10 000 дней. Усложнить задачу, выполнив запрос круглого числа прожитых дней, которое хочет узнать пользователь.
9. Разработать проект, запрашивающий у пользователя год его рождения и год рождения его мамы, выдающий результат о возрасте мамы в момент рождения пользователя.
10. Разработать проект, запрашивающий у пользователя имя в TextBox и в том же TextBox приветствующий его (рис. 1.27, 1.28). При всей кажущейся простоте задания необходимо

сделать вывод приветствия правильно (используя конкатенацию или склейку строк.

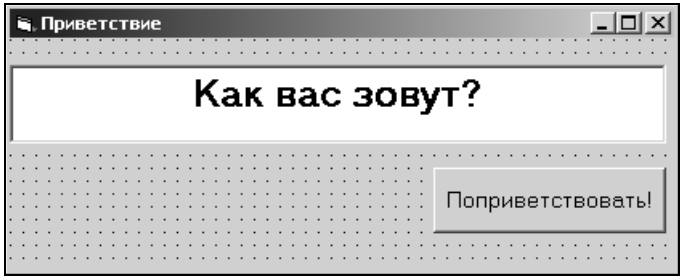


Рис. 1.27. Запрос имени

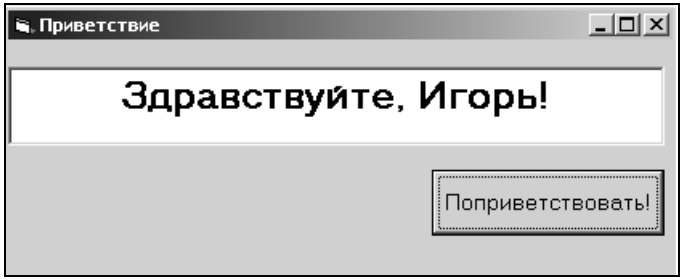


Рис. 1.28. Приветствие

Если совсем никак, вот программный код:

```
Option Explicit
Dim Name As String
Private Sub Command1_Click()
Name = Text1.Text
Text1.Text = "Здравствуйте, " + N + "!"
End Sub
```

Примечание

Здесь знак плюс (+) как раз и осуществляет склейку трех строковых переменных.

А вот в следующем аналогичном задании подсказывать не буду.

11. Разработать и оформить проект, запрашивающий и выводящий в одном окне длину одного катета прямоугольного треугольника, в другом окне — длину второго катета, в третьем — выводящий длину гипотенузы. Все это делается по нажатию только одной командной кнопки.

12. Разработайте проект, запрашивающий три стороны треугольника a, b, c и вычисляющий его площадь по формуле Герона:

$$S = \sqrt{p(p-a)(p-b)(p-c)},$$

где $p = (a+b+c)/2$ — полупериметр.

13. Разработайте проект, в двух текстовых окнах которого запрашиваются два разных целых числа, а по нажатию командной кнопки они меняются местами. Подсказка: надо использовать третью, вспомогательную переменную.

Замечание

Конечно, можно просто показать значения чисел наоборот, без обмена из содержимого. Но это – моветон ☺.

14. Разработайте и оформите проект, запрашивающий высоту дома h (в метрах), ускорение свободного падения g и вычисляющий время падения кирпича t (в секундах) с крыши этого дома по формуле:

$$t = \sqrt{\frac{2h}{g}}.$$

Желательно вставить в проект рисунок, изображающий дом и кирпич ☺.

15. Расстояние до ближайшей к Земле звезды Альфа Центавра — 4,3 световых года. Световой год — это расстояние, которое проходит свет за год. Скорость света принять 300 000 км/с. Скорость земного звездолета 100 км/с. За сколько лет звездолет долетит до звезды? Подсказка: чтобы не было переполнения при вычислениях, лучше задайте правильный тип данных для расстояния до звезды в км, а еще лучше сначала выведите формулу для количества лет и увидите, что она станет очень простой.

16. Известна теория биоритмов. С момента рождения жизнь человека подчиняется трем синусоидальным биоритмам. Физический цикл — 23 дня, эмоциональный — 28 дней и интеллектуальный — 33 дня. Первая половина каждого цикла — положительная, вторая — отрицательная. При переходе от положительной к отрицательной фазе в каждом цикле есть так называемый критический день. В физическом цикле — 12-й день, в эмоциональном — 15-й день, в интеллектуальном — 17-й день. Когда два и более цикла находятся в отрицательной фазе или в критических днях, то в такие дни повышена вероятность физических недомоганий и травм, эмоциональных срывов и ссор, замедленности интеллектуальных процессов и реакции. В Японии, например, в крупных фирмах для каждого работника составлен график и в "плохие" дни их не допускают до работы, дают отдыхать, потому что оплата больничного или брак в работе обойдутся фирме дороже. Теперь к делу. Опираясь на результат предыдущего задания и используя операцию нахождения целочисленного остатка, рассчитайте, каков для вас сегодняшний день по всем трем циклам.

17. Запросите у пользователя валютный курс на сегодняшний день, затем имеющуюся у него рублевую сумму и рассчитайте, сколько долларов и сколько евро он может купить на эти деньги.

18. Дискета 3,5" вмещает 1,44 Мбайт. Рукопись содержит *X* страниц текста. На каждой странице *Y* строк по *Z* символов в каждой. Сколько дискет потребуется для записи рукописи? *X*, *Y* и *Z* запрашиваются.

19. Документ содержит текст из *X* строк по *Y* символов в каждой и точечную черно-белую фотографию 10×15 см. Каждый квадратный сантиметр содержит 600 точек, любая точка описывается 4-мя битами. Каков общий информационный объем документа в Кбайтах?

20. Игра "Zavr In The Sky" требует для установки на жесткий диск *А* мегабайт свободного места. На жестком диске сейчас *Б* мегабайт свободного места. Сколько флэш-карт по *В* мегабайт понадобится, чтобы освободить недостающее пространство? (Данные *А*, *Б* и *В* запрашиваются.)

21. Жесткий диск пуст и имеет объем свободного пространства *Г* гигабайт. а) Сколько книг, каждая из которых состоит из *Д* страниц, на каждой странице *Е* строк, в каждой строке *Ж* символов, можно записать на такой жесткий диск? б) Если учесть, что каждая такая книга 3 см толщиной, то какой высоты в метрах будет стопка, если все их сложить друг на друга?

22. Скорость передачи данных по локальной сети *С* миллионов бит в секунду. Ученик качал игру *Т* минут. а) Сколько это гигабайт и сколько дискет по 1,5 Мбайта можно заполнить таким объемом информации? б) Сколько денег (в рублях) придется заплатить ученику за трафик, если первый 1 Гбайт не оплачивается, а все, что сверх его — по *У* копеек за 1 Мбайт.

23. Запросите у пользователя длину ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.

24. Есть притча о шахматах, где выигравший запросил у могущественного правителя, чтобы ему был выплачен выигрыш зерном пшеницы по следующим правилам: на первую клетку шахматной доски положить одно зерно, на вторую — два зерна, на третью — четыре, на четвертую — восемь и т. д., иными словами, на каждую последующую в два раза больше зерен, чем на предыдущую. Сколько же зерен должен был бы получить выигравший?

Предупреждение

На каком-то этапе выполнения этого задания возможно переполнение памяти. Подумайте, как обойти это препятствие.

25. Кстати, для оценки предыдущего результата еще одно задание. В России ежегодно собирают около 90 млн. тонн зерновых. Масса одного зерна около 5 граммов. Сколько зерен в таком урожае и сколько лет пришлось бы расплачиваться России по условиям предыдущего задания?

26. Продав квартиру, вы получили 52 000$ и положили их в банк. Банк начисляет 1% в первый месяц, а каждый следующий — тоже 1%, но уже с получившейся суммы. Сколько денег будет в банке на вашем счету через год? (Нестабильностью нашей экономической системы пренебрегаем.)

27. Допустим, вы получили наследство 1000 000$ и хотите красиво пожить. После долгих раздумий вы решаете, что будете скромно жить на 800$ в месяц. На сколько лет вам хватит наследства?
28. Пушка стреляет под углом 30° к линии горизонта. Начальная скорость 500 м/с. Какова будет дальность полета снаряда? (Формулу вспомните из курса физики. Ускорение свободного падения принять равным 9,81 м/с^2.)
29. Разработать проект, определяющий количество цифр во введенном целом сначала четырехзначном, а затем пятизначном, положительном числе и выводящий последовательно эти самые цифры с комментариями. Для трехзначного числа привожу пример (рис. 1.29). (Подсказка — воспользуйтесь функциями деления нацело и нахождения целочисленного остатка.)

Рис. 1.29. Цифры числа

Прежде чем дать вам еще несколько расчетных задач, давайте поговорим о графических примитивах, которые могут быть ис-

пользованы в Visual Basic для украшения наших замечательных форм.

1.4.6. Графические примитивы

Самым простым способом является использование инструментов панели **General**. Приведем пример.

Проект "Российский флаг в ночи"

Возьмем стандартную форму, увеличим ее максимально в пределах возможностей вашего монитора и зададим следующие свойства для формы: `Caption` — Российский флаг в ночи; `BackColor` — черный.

Теперь возьмем на панели инструментов **General** инструмент **Line** —  и проведем вертикальную линию на нашей форме (своего рода флагшток).

Черное на черном обычно видно плохо, поэтому зададим для линии свойство `BorderColor` — белый и `BorderWidth` — 2.

Затем возьмем инструмент **Shape** —  и от верхнего окончания линий вправо и вниз сделаем прямоугольник. Компьютер пока не знает, что будет в нем, и мы должны ему об этом сообщить, выбрав свойства: `Shape` — Rectangle; `FillStyle` — Solid (сплошная заливка); `FillColor` — белый; `BorderColor` — белый.

У нас должен получиться белый прямоугольник с белыми же границами.

Аналогичные действия необходимо проделать еще с двумя полосами российского флага.

Примечание

Для тех, кто не помнит: наш флаг описывается фактически тем же словом, что и изучаемый нами язык — БеСиК — бело-сине-красный☺.

И вот он реет над нашей формой (рис. 1.30). Виват, Россия!

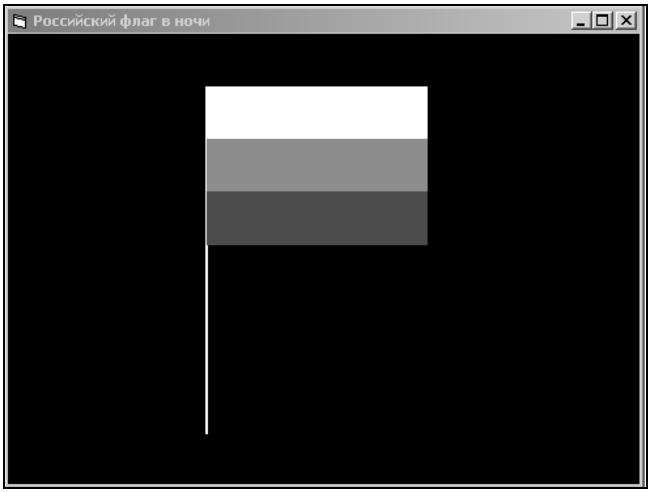


Рис. 1.30. Российский флаг в ночи

Вам предлагается самостоятельно разобраться с еще возможными Shape'ами, их свойствами и возможностями. Для этого неплохо бы выполнить еще пару-тройку заданий.

30. Изобразить на форме идиллическую картинку: травушка, на травушке стоит дом, в небе светит солнце (рис. 1.31).

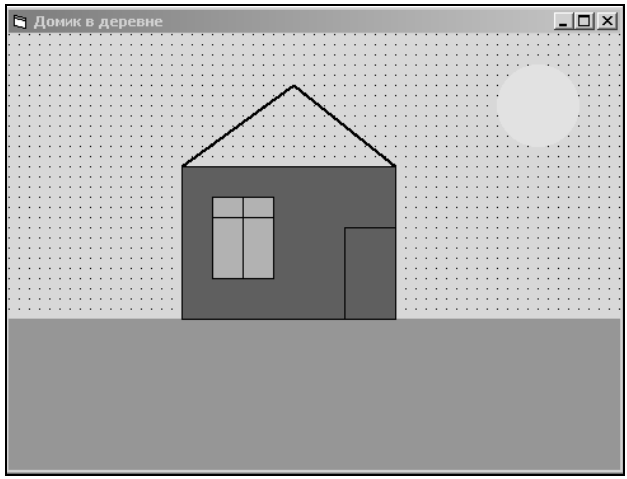


Рис. 1.31. Домик в деревне днем

31. Картинка остается почти прежней, только наступает ночь, и солнце превращается в месяц. Подсказка: месяц получается наложением двух `Shapes` — круга и овала (рис. 1.32).

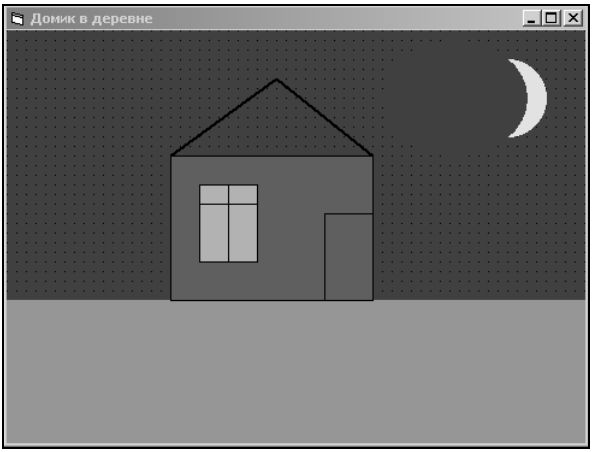


Рис. 1.32. Домик в деревне ночью

32. Ну, а героиню нашу народную, нечто похожее на Масяню, слабо изобразить? Напоминаю, как она выглядит (рис. 1.33). Кто хочет — рисует Хрюнделя ☺.

Рис. 1.33. Масяня

Программируемые графические примитивы

В предыдущем разделе мы пользовались графическими примитивами, что называется, на глазок. Теперь мы попробуем действовать холодно и расчетливо, хирургически точно.

Для этого есть программируемые примитивы — ведь любое компьютерное изображение по сути — это набор разноцветных экранных точек (пикселов), а стало быть, потрудившись, мы сможем любое изображение воспроизвести. Для облегчения нашей деятельности в Visual Basic предусмотрены не только точки, но и линии, и прямоугольники, и окружности, и дуги. Кроме того, существует возможность все это подписать (табл. 1.6).

Таблица 1.6. *Графические примитивы*

Наименование	Синтаксис и комментарии
Точка	`object.pset (X, Y), C` `X, Y` — координаты точки, `C` — цвет. Если `object` не указан, то построение осуществляется на форме
Окружность	`object.circle(X, Y), R, C, A, B` `X, Y` — координаты центра в выбранной системе координат, `R` — радиус, `C` — цвет, `A,B` — углы дуги в радианах. Если `object` не указан, то построение осуществляется на форме. Если углы не указаны, то строится полная окружность. Если углы указаны, то строится дуга против часовой стрелки от угла A к углу B. Сведения о построении эллипса (овала) содержатся в задаче 53
Отрезок линии	`object.Line(X1,Y1)-(X2,Y2), C` `X1,Y1` — координаты точки начала отрезка, `X2,Y2` — координаты точки окончания отрезка
Прямоугольник со сторонами, параллельными экрану	`object.Line(X1,Y1)-(X2,Y2), C, B` `X1,Y1` — координаты точки начала диагонали прямоугольника, `X2,Y2` — координаты точки окончания диагонали прямоугольника, `C` — цвет, `B` (от англ. "Box" — коробка) заставляет программу достроить прямоугольник по его диагонали (диагональ может быть взята любая из двух возможных). Сама диагональ при этом не строится

Таблица 1.6 *(окончание)*

Наименование	Синтаксис и комментарии

Прямоугольник со сторонами, параллельными экрану, закрашенный	`object.Line(X1,Y1)-(X2,Y2), C, BF`
	`X1,Y1` — координаты точки начала диагонали прямоугольника, `X2,Y2` — координаты точки окончания диагонали прямоугольника, `C` — цвет,
	`BF` (от англ. "Box Full" — полная коробка) заставляет программу достроить прямоугольник по его диагонали и закрасить его цветом `C`

Очистка нарисованного	`object.Cls` Очищает объект от текста и графики, созданных в нем программно
Возвращение цвета точки с указанными координатами	`object.Point(X,Y)`
Вывод сообщения в указанном месте объекта	`object.Print [output]` В качестве `output` может быть строковое или числовое выражение. Чтобы вывести сообщения в задуманном месте, можно действовать двумя способами: 1) Вывод осуществляется от последней поставленной графической точки. Поэтому можно непосредственно перед выводом поставить эту самую точку там, где будет располагаться левый верхний угол первого символа (цвет самой точки должен совпадать с цветом фона, чтобы ее видно даже не было!) 2) Воспользоваться свойствами объекта, в который выводится сообщение: `CurrentX` — горизонтальная координата начала вывода; `CurrentY` — вертикальная координата начала вывода; `Font` — размер и шрифт выводимого; `FontTransparent` — прозрачность текста; `ForeColor` — цвет

Замечание

Закрашенный круг получается, если перед рисованием окружности указать свойство `[объект].FillStyle=0` (0 — сплошная заливка, а есть еще всякие варианты, посмотрите сами в таблице свойств) и затем свойство `FillColor=<собственно цвет>` (про цвета см. чуть ниже).

Сектор получится, если перед дугами поставить знак минус (–), а закрашенный сектор — аналогично закрашенному кругу.

Почти во всех вышеперечисленных методах рисования присутствуют координаты. Откуда их взять? Лучше всего задать самому.

Задание координатной сетки объекта

Это осуществляется оператором `Scale`, в котором задаются левый верхний и правый нижний углы объекта в наших условных единицах. Это называется *масштабированием*.

Проект "Координатные оси"

Допустим, мы хотим, чтобы для построения графика какой-либо функции на экране присутствовали хорошо нам знакомые оси абсцисс и ординат и начало координат лежало в начале формы (рис. 1.34).

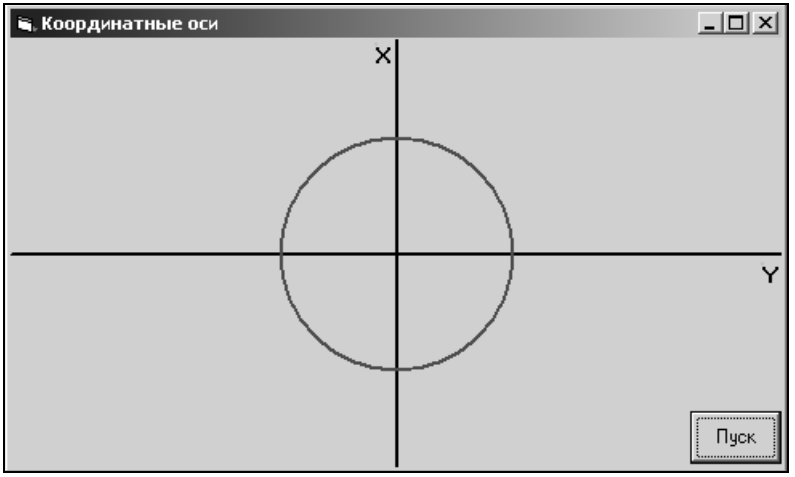


Рис. 1.34. Координатные оси

Сначала мы задаемся общими размерами прямоугольника, в котором и будут находиться наши оси. Допустим, ширина прямоугольника — 20 условных единиц, а высота — 14 условных единиц. Тогда если центр будет иметь координаты (0, 0), то левый верхний угол будет иметь координаты (−10, 7), а координаты правого нижнего угла будут (10, −7).

Программный код построения осей будет выглядеть следующим образом:

```
Private Sub Command1_Click()
'Масштабирование
Scale (-10, 7)-(10, -7)
'Задание толщины рисуемых объектов
DrawWidth = 2 'Если DrawWidth=1, то толщина - 1 пиксел
'Рисование оси ординат
Line (0, -7)-(0, 7)
'Рисование оси абсцисс
Line (-10, 0)-(10, 0)
'Подпись оси ординат
PSet (9.5, -0.3), QBColor(7)
Print "Y"
'Подпись оси абсцисс
PSet (-0.5, 6.9), QBColor(7)
Print "X"
'Рисование красной окружности радиуса 3
Circle (0, 0), 3, vbRed
End Sub
```

Примечание

Знаком *апостроф* (') начинаются строки комментариев (на клавиатуре он находится там, где русская буква Э). Такие строки не исполняются, а служат пояснениями к программному коду.

1.4.7. О цветах

При оформлении форм и других, уже использовавшихся объектов мы уже применяли цветовое оформление, используя свойства `BackColor`, `ForeColor`, `FillStyle`, `FillColor` etc. Но Visual Basic

позволяет оперировать с количеством оттенков, равным 256^3. (Не поленитесь, кстати, использовать наш проект Калькулятор для подсчета количества вариантов.)

Программно цвет можно задать тремя способами.

- Используя константы цветов (табл. 1.7). В этом случае цвет указывается непосредственно константным словом.

Таблица 1.7. *Константы цветов*

Константа	Цвет
`vbBlack`	Черный
`vbWhite`	Белый
`vbRed`	Красный
`vbBlue`	Синий
`vbGreen`	Зеленый
`vbYellow`	Желтый
`vbCyan`	Голубой
`vbMagenta`	Фиолетовый

- Используя старый добрый набор цветов QBasic (табл. 1.8). В этом случае цвет указывается так: qbColor (номер цвета).

Таблица 1.8. *Цвета функции QBColor*

Номер цвета	Цвет
0	Черный
1	Синий
2	Зеленый
3	Бирюзовый
4	Красный
5	Темно-красный
6	Коричневый
7	Светло-серый

Таблица 1.8 *(окончание)*

Номер цвета	Цвет
8	Серый
9	Голубой
10	Светло-зеленый
11	Светло-бирюзовый
12	Светло-красный
13	Фиолетовый
14	Желтый
15	Белый

- С помощью функции `RGB` (Red-Green-Blue). Есть три основных цвета, остальные получаются смешением этих трех. Значение каждого из цветов меняется от 0 до 255.

  Например, цвет `RGB (0, 0, 255)` будет чисто синим, а черный получится `RGB (0, 0, 0)`, а серый, например, `RGB (192, 192, 192)`. Поэкспериментируйте — и все получится!

  Замечание

  Если цвет не указан, то рисование производится по умолчанию черным цветом.

Для рисования можно использовать непосредственно форму, но чаще используется инструмент **PictureBox** —  , который, как и инструмент **Image,** позволяет внутри себя не только размещать картинки, но и рисовать программно, используя вышеописанные способы.

А теперь перейдем к выполнению заданий. Сначала без графики.

33. Разработайте проект, в котором пользователь мог ввести название города-юбиляра, год его основания, текущий год. После нажатия командной кнопки должно выйти поздравление: "Поздравляем жителей города N с X-летием!!!"
34. Посчитайте среднее арифметическое трех натуральных чисел.

35. Разработайте проект, который находит квадратный корень произведения двух вещественных чисел одинакового знака.
36. Разработайте проект, который вычисляет сумму двух введенных чисел типа Integer и переводит ее в шестнадцатеричную систему.

А теперь с графическим оформлением наиболее подходящим способом.

37. Вычислите диагональ квадрата со стороной A (A — целое число от 2 до 10 условных единиц). Нарисуйте на форме квадрат в соответствии с введенной стороной, подпишите длину стороны и длину диагонали после вычисления. Чтобы был понятен уровень моих притязаний к оформлению нижеследующих задач, я представляю решение этой задачи (рис. 1.35).

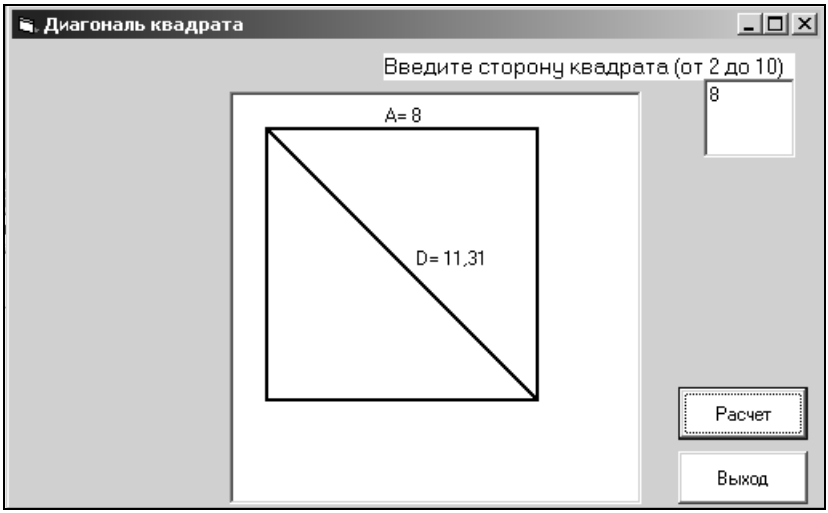


Рис. 1.35. Диагональ квадрата

Замечание

В этом примере и во многих нижеследующих у меня в командных строках (ввиду ограничений книжной страницы) будут встречаться переносы. Они допустимы в VB. Обозначаются они символом подчеркивания, обязательно после пробела. В тексте программы

переносы тоже можно использовать — для наглядности программного кода. Если же при механическом переписывании моего примера на ваш компьютер перенос окажется в середине командной строки – это будет расценено компилятором как ошибка — в таких случаях переносы надо удалять!

Программный код:

```
Option Explicit
' Описание переменной A (стороны квадрата) типа 'Byte
Dim A As Byte
'Командный код для нажатия кнопки Расчет
Private Sub Command1_Click()
'Масштабирование (квадрат 10х10)
Picture1.Scale (0, 0)-(12, 12)
'Занесение значения длины стороны квадрата в
'переменную A
A = Val(Text1.Text)
'Установка ширины рисования 2
Picture1.DrawWidth = 2
'Рисование прямоугольника с заданной длиной стороны
'цвет пропущен - поэтому две запятых
Picture1.Line (1, 1)-(A + 1, A + 1), , B
'Рисование диагонали
Picture1.Line (1, 1)-(A + 1, A + 1)
'Подпись стороны квадрата буквой A с ее значением
Picture1.PSet ((1 + A) / 2, 0.3), vbWhite
Picture1.Print "A=";A
'Подпись диагонали квадрата вычисленным значением
Picture1.PSet ((Sqr(2 * A * A) - 1) / 2, (1+A)  / 2) _
, vbWhite
'D - значение диагонали, вычисление гипотенузы
'сопровождается умножением на сто с отбрасыванием
'дробной части и делением на сто для получения двух
'знаков после запятой
Picture1.Print "D=";Fix(Sqr(2 * A * A) * 100) / 100
End Sub
'Процедура выхода из программы
Private Sub Command2_Click()
End
End Sub
```

38. Запрашивается радиус круга (от 10 до 100 пикселов). Напишите программу, которая вычисляет площадь этого круга. Круг нарисован в зависимости от введенного радиуса, внутри выводится вычисленная площадь с точностью до сотых.
39. Запрашивается радиус окружности (от 10 до 100 пикселов). Напишите программу, которая вычисляет длину этой окружности. Окружность нарисована в зависимости от введенного радиуса, внутри нее выводится вычисленная длина с точностью до тысячных.
40. Запрашиваются диагонали ромба. Создайте проект, вычисляющий площадь ромба. Ромб изображается, диагонали подписываются, а площадь выводится под ним.
41. Разработайте проект, который находит площадь равнобедренной трапеции по ее основаниям и высоте. Трапеция должна быть нарисована, исходные данные подписаны, а площадь выведена внутри.
42. Вычислите объем цилиндра с радиусом основания *R* и высотой *H*. Цилиндр должен быть нарисован.
43. Определите координату середины отрезка (*X, Y*), если известны координаты концов отрезка: (*X1, Y1*) и (*X2, Y2*). Нарисовать отрезок, вывести все координаты.
44. Даны декартовы координаты вершин треугольника (в плоскости). Разработайте проект, вычисляющий площадь и периметр этого треугольника. Треугольник должен присутствовать на форме в нарисованном виде, вычисленные длины сторон подписаны, под треугольником необходимо вывести его площадь и периметр с точностью до сотых.
45. Определите расстояние, пройденное физическим телом за время *T*, если тело движется с постоянным ускорением *A* и имеет в начальный момент времени скорость *V*. Нарисовать это расстояние в виде толстого горизонтального отрезка, его длину подписать.

Примечание

Автор интересуется — возможно ли такое масштабирование PictureBox, которое позволяло бы выводить геометрические фигуры на объекте при очень больших разбросах значений? Скажем,

сторона квадрата может меняться от 2 до 4000. Мне кажется, что возможно... ☺

Ну и немножко еще порисуем.

Замечание

Казалось бы, зачем так сложно рисовать вручную, ведь можно нарисовать в более продвинутых специальных программах, а затем вставить готовое изображение. Но, как выясняется при ближайшем рассмотрении, программное изображение рисуется в Visual Basic гораздо быстрее, чем вставляется. Поэтому кое-что изобразим вручную. Мало того, что это красиво, так еще и очень полезно (в смысле развития пространственного воображения и изощренности программной мысли ☺).

46. Изобразите корабль под Андреевским флагом (рис. 1.36).

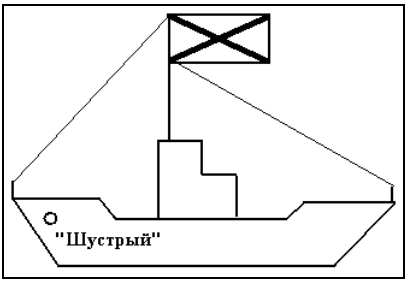


Рис. 1.36. Линкор "Шустрый"

47. Японский флаг и швейцарский на одной форме с подписями и появлением по щелчку мышью (рис. 1.37).

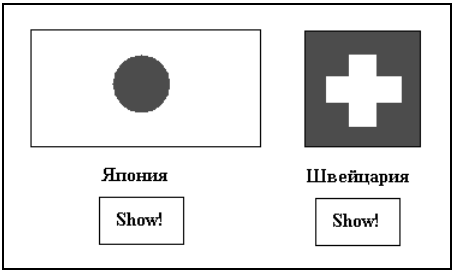


Рис. 1.37. Восток и Запад

48. Смайлик улыбающийся и смайлик грустящий (рис. 1.38).

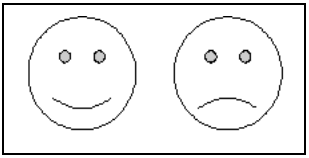


Рис. 1.38. Вроде зебры жизнь, вроде зебры...

49. Правильную пятиконечную звезду. Тут, кажется, без знания синусов-косинусов не обойтись ☹ (рис. 1.39).

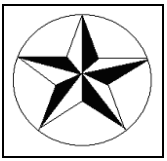


Рис. 1.39. Звезда

50. Мишень (рис. 1.40).

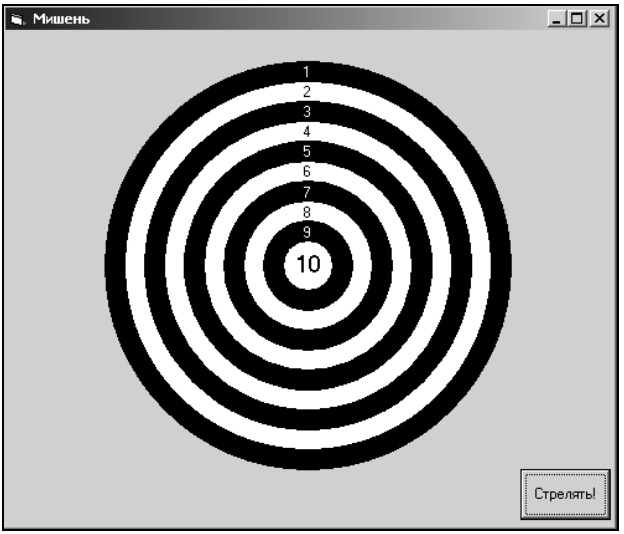


Рис. 1.40. Мишень

51. Угостите меня мороженым (рис. 1.41). Вспоминаю детство золотое, походы с мамой в мороженицы и шарики этого вкуснейшего для детей (и не только ☺) лакомства в специальных вазочках, называемых креманки. Вот этого изображения собственно моего детства я от вас и прошу. Впрочем, есть проблема. Оказывается, нет в VB графического оператора заливки замкнутого контура, а мне очень хотелось бы, чтобы все было цветным и веселым! Так и быть, покажу сейчас (уж больно сильна ностальгия), как я изощрился в этом примере. Наверное, есть более простой способ, но мы же не ищем легких путей А вы сделайте по-своему, а если никак, то хотя бы выполните следующий пример, по-моему ☺.

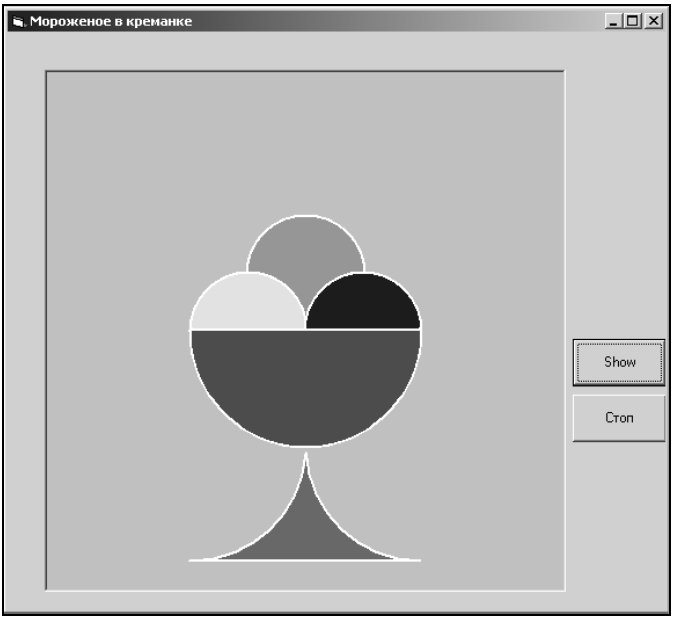


Рис. 1.41. Мороженое

Сначала зададим для формы свойство `ScaleMode` — Pixel, растянем на ней PictureBox, зададим то же свойство `ScaleMode` — Pixel, зададим размеры `Width=450` и `Height=450`. Увеличим форму соразмерно PictureBox. Расположим на форме 2 командные кнопки.

Программный код для проекта "Мороженое":

```
Private Sub Command1_Click()
'Задаем цвет формы (светло-серенький такой)
Picture1.BackColor = QBColor(7)
'Задаем условную шкалу в пикселах, с координатами '0,0
'в центре PictureBox
Picture1.Scale (-225, 225)-(225, -225)
'Задаем удвоенную толщину линий
Picture1.DrawWidth = 2
'Рисуем прямоугольник на месте "ножки" нашей креманки
'(пробел со знаком подчеркивания в конце строки - знак
'переноса в Visual Basic)
Picture1.Line (-100, -200)-(100, -100), _
vbMagenta, BF
'Задаем тип и цвет заливки для двух кругов, которые
'затрут часть прямоугольника и оставят нам закрашенную
'"ножку" креманки
Picture1.FillStyle = 0
Picture1.FillColor = QBColor(7)
Picture1.Circle (-100, -100), 100, QBColor(7)
Picture1.Circle (100, -100), 100, QBColor(7)
'Рисуем контур "ножки" креманки
Picture1.Line (-100, -200)-(100, -200), vbWhite
Picture1.Circle (-100, -100), 100, vbWhite, 4.71, 0
Picture1.Circle (100, -100), 100, vbWhite, _
3.14, 4.71
'Рисуем "чашу" креманки
Picture1.FillColor = vbYellow
Picture1.Circle (0, 0), 100, vbWhite, -3.14, -6.28
'Рисуем шарики мороженого, начиная с самого верхнего
Picture1.FillColor = vbBlue
Picture1.Circle (0, 50), 50, vbWhite

Picture1.FillColor = vbRed
Picture1.Circle (-50, 0), 50, vbWhite, -6.28, -3.14

Picture1.FillColor = vbGreen
```

```
Picture1.Circle (50, 0), 50, vbWhite, -6.28, -3.14
'Вывод надписи с заданием свойств начертания, размера и цвета
Picture1.PSet (-150, 175), QBColor(7)
Picture1.FontBold = True
Picture1.FontSize = 24
Picture1.ForeColor = RGB(255, 0, 0)
Picture1.Print "ВКУС ДЕТСТВА…"
End Sub

'Процедура выхода
Private Sub Command2_Click()
End
End Sub
```

52. Любовь... Что может быть прекраснее на свете? Проект "Иван + Марья = ?" (рис. 1.42). Если ваш изощренный ум еще не готов самостоятельно решить задачу, воспользуйтесь предыдущим примером.

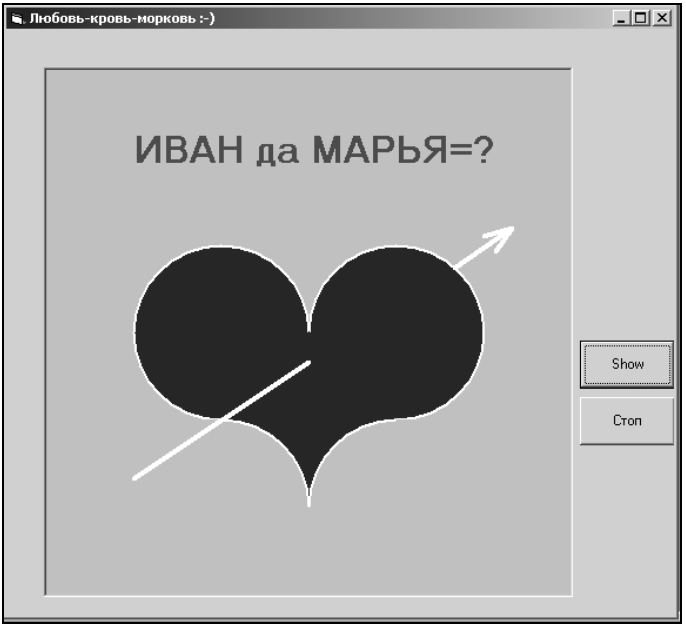


Рис. 1.42. Что же это такое?

53. Страна эллипсов. Для этого проекта надо запомнить, что для построения эллипса (или овала, как многие его называют), надо к оператору построения окружности добавить коэффициент сжатия k. Если $0 < k < 1$, то получится горизонтально сжатая окружность, а если $k > 1$, то вертикально. Команда `Circle(100,150),50,vbRed,,,.5`

построит нам такой эллипс (рис. 1.43).

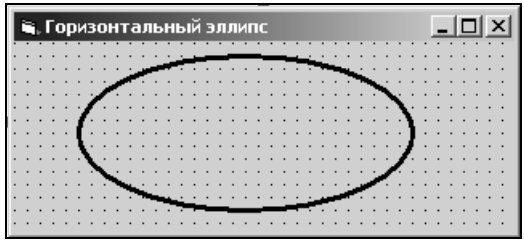


Рис. 1.43. Горизонтальный эллипс

А команда `Circle(100,150),50,vbRed,,,2` — вертикальный соответственно (рис. 1.44).

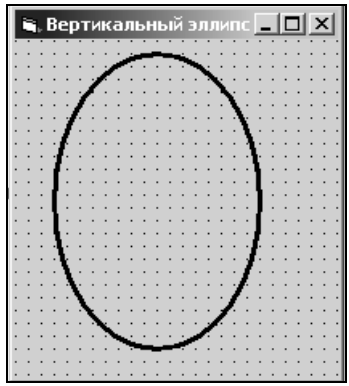


Рис. 1.44. Вертикальный эллипс

Дуги эллипсов строятся аналогично дугам окружности, именно для углов дуг пропущено место после цвета.

Теперь вам предстоит самостоятельно построить следующее изображение (рис. 1.45).

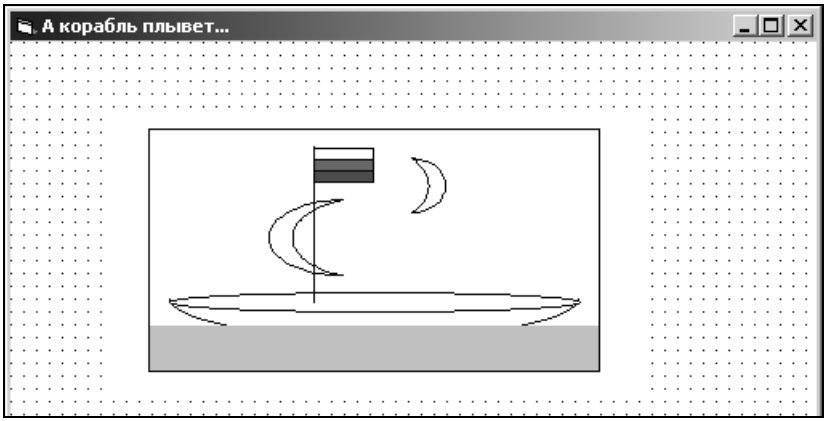


Рис. 1.45. А корабль плывет...

54. "Невозможные" фигуры. Мне очень нравится творчество голландского графика Эшера, пример его работы приведен на рис. 1.46 в стиле импоссибилизма ("невозможности"). Откуда поступает вода для водопада?

Рис. 1.46. "Водопад" М. К. Эшер, 1961 г.

Шведский архитектор Рутерсвард продолжил это направление. Большинство из его фигур могут существовать только на плоскости. Попробуем сначала вместе воспроизвести одну из них (рис. 1.47).

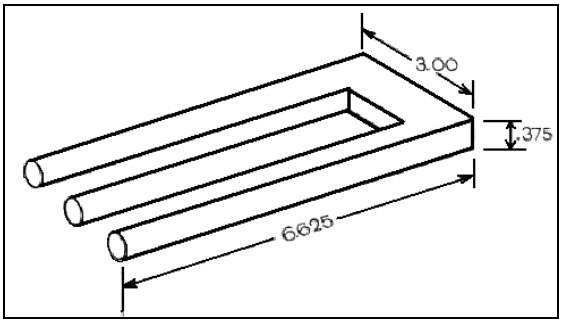


Рис. 1.47. Три или два?

А теперь попробуйте еще две невозможных фигуры (рис. 1.48, 1.49).

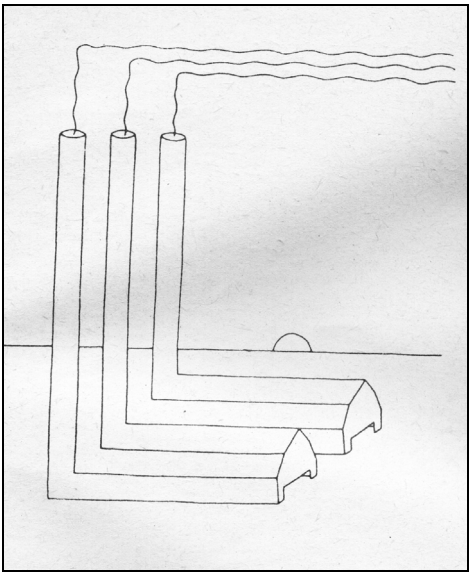


Рис. 1.48. Завод

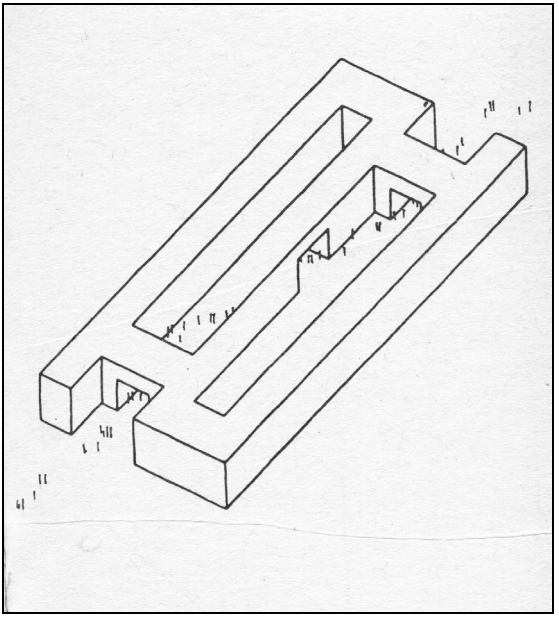


Рис. 1.49. Во дворе

1.4.8. Построение диаграмм и графиков

Вот уж где могут пригодиться графические примитивы, так это при построении диаграмм, иллюстрирующих различные достижения или сравнительные характеристики. Прежде чем дать задания, приведем два примера, где построение диаграммы предваряется расчетами.

Пример 1. Население земного шара по континентам. (Европа — 800 млн. чел., Азия — 4000 млн. чел., Африка — 700 млн. чел., Северная Америка — 500 млн. чел., Южная Америка — 400 млн. чел., Австралия — 40 млн. чел.) Выглядеть диаграмма должна вот так (рис. 1.50).

А строится при помощи следующего командного кода.

Программный код:

```
Private Sub Command1_Click()
' Задание цвета PictureBox
```

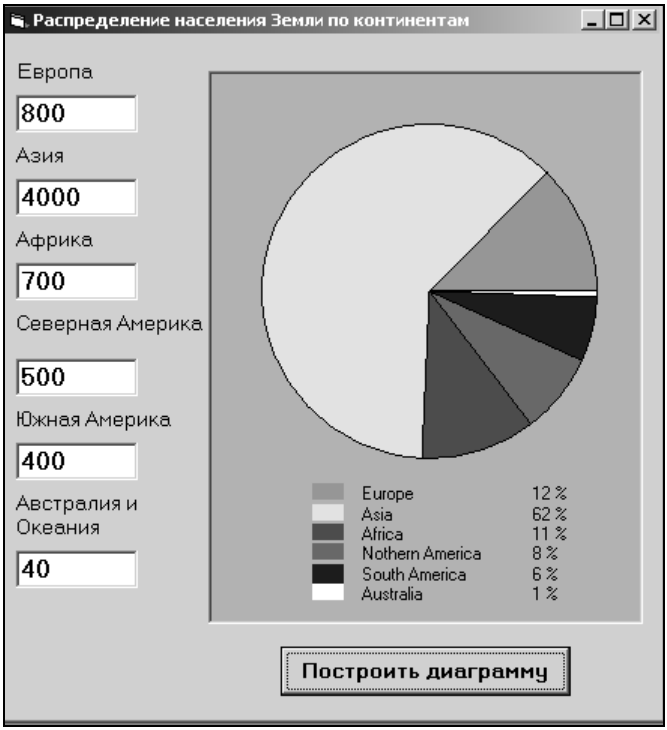


Рис. 1.50. Распределение населения земного шара по континентам

```
Picture1.BackColor = vbCyan
' Считывание данных из TextBox
X1 = CInt(Text1.Text)
X2 = CInt(Text2.Text)
X3 = CInt(Text3.Text)
X4 = CInt(Text4.Text)
X5 = CInt(Text5.Text)
X6 = CInt(Text6.Text)
' Вычисление общего населения Земли
z = X1 + X2 + X3 + X4 + X5 + X6
' Вычисление углов для построения круговой диаграммы
a1 = X1 * 360 / z
a2 = a1 + X2 * 360 / z
```

```
a3 = a2 + X3 * 360 / z
a4 = a3 + X4 * 360 / z
a5 = a4 + X5 * 360 / z
a6 = a5 + X6 * 360 / z
' Вычисление процентного соотношения населения
' соответствующего континента к общему населению Земли
p1 = Round(X1 / z * 100)
p2 = Round(X2 / z * 100)
p3 = Round(X3 / z * 100)
p4 = Round(X4 / z * 100)
p5 = Round(X5 / z * 100)
p6 = Round(X6 / z * 100)
' Построение соответствующих секторов
Picture1.FillStyle = 0
Picture1.FillColor = vbGreen
Picture1.Circle (Picture1.Width / 2, _
Picture1.Width / 2), (Picture1.Width - 1000) _
/ 2, , -6.28, -(a1 * 3.14 / 180)

Picture1.FillColor = vbYellow
Picture1.Circle (Picture1.Width / 2, _
Picture1.Width / 2), (Picture1.Width - 1000) _
/ 2, , -(a1 * 3.14 / 180), -(a2 * 3.14 / 180)

Picture1.FillColor = vbRed
Picture1.Circle (Picture1.Width / 2, _
Picture1.Width / 2), (Picture1.Width - 1000) _
/ 2, , -(a2 * 3.14 / 180), -(a3 * 3.14 / 180)

Picture1.FillColor = vbMagenta
Picture1.Circle (Picture1.Width / 2, _
Picture1.Width / 2), (Picture1.Width - 1000) _
/ 2, , -(a3 * 3.14 / 180), -(a4 * 3.14 / 180)

Picture1.FillColor = vbBlue
```

```
Picture1.Circle (Picture1.Width / 2, _
Picture1.Width / 2), (Picture1.Width - 1000) _
/ 2, , -(a4 * 3.14 / 180), -(a5 * 3.14 / 180)

Picture1.FillColor = vbWhite
Picture1.Circle (Picture1.Width / 2, _
Picture1.Width / 2), (Picture1.Width - 1000) _
/ 2, , -(a5 * 3.14 / 180), -(a6 * 3.14 / 180)

' Вывод "легенды" диаграммы с соответствующими
' значениями процентных соотношений
Picture1.Line (1000, 4100)-(1300, 4250), _
vbGreen, BF
Picture1.CurrentX = 1500
Picture1.CurrentY = 4100
Picture1.Print "Europe", , p1; "%"

Picture1.Line (1000, 4300)-(1300, 4450), _
vbYellow, BF
Picture1.CurrentX = 1500
Picture1.CurrentY = 4300
Picture1.Print "Asia", , p2; "%"

Picture1.Line (1000, 4500)-(1300, 4650), vbRed, BF
Picture1.CurrentX = 1500
Picture1.CurrentY = 4500
Picture1.Print "Africa", , p3; "%"

Picture1.Line (1000, 4700)-(1300, 4850), _
vbMagenta, BF
Picture1.CurrentX = 1500
Picture1.CurrentY = 4700
Picture1.Print "Nothern America", p4; "%"

Picture1.Line (1000, 4900)-(1300, 5150), vbBlue, BF
```

```
Picture1.CurrentX = 1500
Picture1.CurrentY = 4900
Picture1.Print "South America", p5; "%"

Picture1.Line (1000, 5100)-(1300, 5250), _
vbWhite, BF
Picture1.CurrentX = 1500
Picture1.CurrentY = 5100
Picture1.Print "Australia", p6; "%"

End Sub
```

Пример 2. Построить столбиковую диаграмму для отображения процентных соотношений оценок за контрольную работу по вашей любимой физике ☺.

Форма может выглядеть примерно так (рис. 1.51).

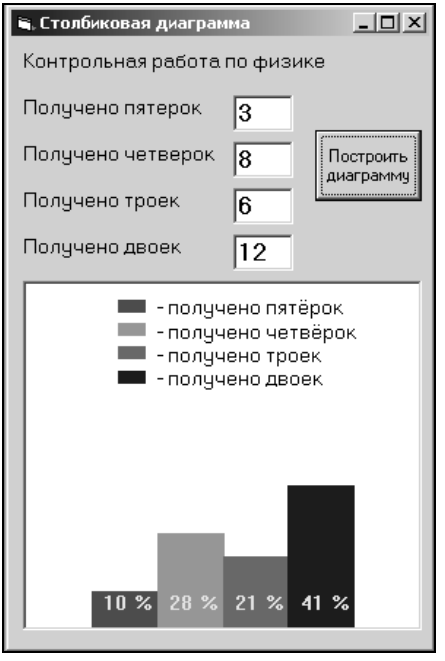


Рис. 1.51. Диаграмма оценок по физике

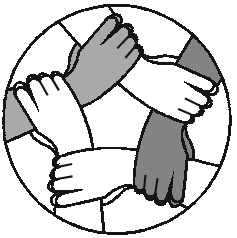
А программный код так:

```
Private Sub Command1_Click()
' Считывание исходных данных из TextBox
X1 = CInt(Text1.Text)
X2 = CInt(Text2.Text)
X3 = CInt(Text3.Text)
X4 = CInt(Text4.Text)
' Нахождение общего количества оценок
z = CInt(X1 + X2 + X3 + X4)
' Задание шкалы PictureBox в зависимости от z
Picture1.Scale (0, z)-(6, 0)
' Построение столбиков
Picture1.Line (1, 0)-(2, X1), vbRed, BF
Picture1.Line (2, 0)-(3, X2), vbGreen, BF
Picture1.Line (3, 0)-(4, X3), vbMagenta, BF
Picture1.Line (4, 0)-(5, X4), vbBlue, BF
' Задание параметров шрифта для вывода процентных
' соотношений
Picture1.ForeColor = vbYellow
Picture1.FontBold = True
Picture1.FontSize = 10
' Вывод на столбиках процентных соотношений
Picture1.CurrentX = 1.1
Picture1.CurrentY = 3
Picture1.Print Round(X1 * 100 / z); "%"

Picture1.CurrentX = 2.1
Picture1.CurrentY = 3
Picture1.Print Round(X2 * 100 / z); "%"

Picture1.CurrentX = 3.1
Picture1.CurrentY = 3
Picture1.Print Round(X3 * 100 / z); "%"

Picture1.CurrentX = 4.1
Picture1.CurrentY = 3
Picture1.Print Round(X4 * 100 / z); "%"
```

```
' Вывод "легенды" диаграммы
Picture1.Line (1.4, z - 1.3)-(1.8, z - 2.3), vbRed, BF
Picture1.Line (1.4, z - 3.3)-(1.8, z - 4.3) , _
vbGreen, BF
Picture1.Line (1.4, z - 5.3)-(1.8, z - 6.3), _
vbMagenta, BF
Picture1.Line (1.4, z - 7.3)-(1.8, z - 8.3), _
vbBlue, BF
' Задание новых параметров шрифта
Picture1.ForeColor = vbBlack
Picture1.FontBold = False
' Вывод текстовых подписей "легенды"
Picture1.CurrentX = 2: Picture1.CurrentY = z - 1
Picture1.Print "- получено пятерок"

Picture1.CurrentX = 2: Picture1.CurrentY = z - 3
Picture1.Print "- получено четверок "

Picture1.CurrentX = 2: Picture1.CurrentY = z - 5
Picture1.Print "- получено троек"

Picture1.CurrentX = 2: Picture1.CurrentY = z - 7
Picture1.Print "- получено двоек"

End Sub
```

А теперь несколько заданий на построение диаграмм.

55. Построить круговую диаграмму процентного соотношения богатых, людей среднего класса, бедных и людей, живущих за чертой бедности в России (данные найти самостоятельно).
56. Построить круговую диаграмму процентного распределения бюджета своей семьи за месяц (данные взять у родителей).
57. Построить столбиковую диаграмму распределения по росту учащихся вашего класса по следующей классификации: выше 175 см, от 170 до 175 см, от 165 до 170 см, от 160 до 165 см, ниже 160 см.
58. Построить столбиковую диаграмму, отражающую распределение своих оценок за месяц (данные взять в дневнике).



Глава 2

Обед продолжается… Второе

На второе у нас в меню алгоритмы разветвляющиеся, выбирающие и циклические, анимация, массивы, строковые переменные, работа с файлами и процедурами.

Все то, о чем мы говорили в первой части нашей книги, были цветочки, подготовительная стадия. Здесь мы будем штурмовать высоты программирования, но только терпенье и труд к сим вершинам приведут ☺. Обедаем дальше...

2.1. Алгоритмы выбирающие и разветвляющиеся…

Сначала повыбираем. Вот, например, получили вы в школе отметку (вариантов не густо, как в детском стишке, — 1, 2, 3, 4, 5 и иди-ка ты гулять ☺). А пусть бы компьютер нам это в отметки перевел: 1 — "плохо", 2 — "неудовлетворительно", 3 — "удовлетворительно", 4 — "хорошо", 5 — "отлично", да еще хорошо бы прокомментировал как-нибудь, да по имени бы обратился, ласково так ☺.

Делать это мы будем при помощи оператора `Select Case`, который работает таким образом, что в зависимости от значения переменной можно исполнить один из блоков кода. Общий вид данной конструкции выглядит следующим образом:

```
Select Case переменная
     Case первое значение переменной
```

```
          <список операторов 1>
     Case второе значение переменной
          <список операторов 2>

     Case N-ное значение переменной
          <список операторов N>
     Case Else
          <список операторов N+1>
End Select
```

Ниже приводится командный код для проекта "Оценки и отметки". Его форма может выглядеть примерно вот так (рис. 2.1).

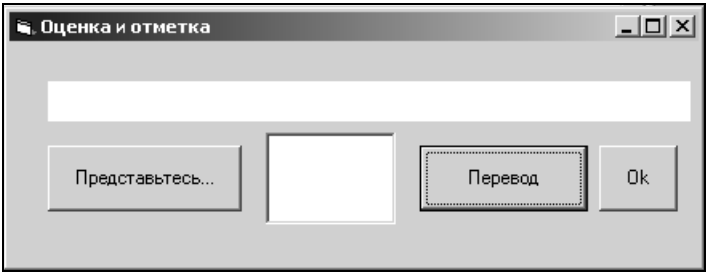


Рис. 2.1. Оценки и отметки

```
Option Explicit
Dim n As String 'Переменная n для имени пользователя
Dim x As Byte 'Переменная x для оценки

'Процедура запроса имени пользователя и вывод
'приветствия в объекте формы Label (по клику
'командной кнопки Представьтесь)
Private Sub Command1_Click()
n = InputBox("Как вас зовут, уважаемый?", "Окно ввода имени")
Label1.Caption = n + ", что Вы сегодня получили в школе?"
End Sub

'Процедура считывания из TextBox оценки и
'комментирования ее при помощи Select Case (по
```

```
'клику командной кнопки Перевод)
Private Sub Command2_Click()
x = Val(Text1.Text)
Select Case x
Case 1
Text1.Text = "УПС!"
Label1.Caption = "Плохо! Я достаю из широких штанин... _
РЕМЕНЬ, негодник!"
Case 2
Text1.Text = "ОПС!"
Label1.Caption = "Неудовлетворительно! Сегодня - без чата! "
Case 3
Text1.Text = "УДВЛ!"
Label1.Caption = "Ну что, " + n + ", удовлетворил школу?"
Case 4
Text1.Text = "ХОР!"
Label1.Caption = "А четверка - это очень, очень хорошо! "
Case 5
Text1.Text = "WOW!"
Label1.Caption = "Ну, " + n + ", отлично! Весь в отца!"
Case Else
Text1.Text = "???!"
Label1.Caption = "Ты что, " + n + ", не знаешь, какие _
оценки бывают?"
End Select
End Sub

'Процедура выхода из проекта
Private Sub Command3_Click()
End
End Sub
```

Обратите внимание, что:

- в первой процедуре используется команда `InputBox`, которая обеспечивает вывод окна запроса для ввода имени;

- во второй процедуре в используемом операторе `Select Case` рассматривается пять значений оценок с выводом соответствующих комментариев. Если же введенная оценка не входит в интервал от 1 до 5, то исполняются команды, прописанные после ключевого слова `Else`;
- при выведении комментариев используется операция конкатенации (склейки) строк для того, чтобы в выводимой фразе задействовать имя пользователя.

Последовательно окна после запуска выглядят так (рис. 2.2).

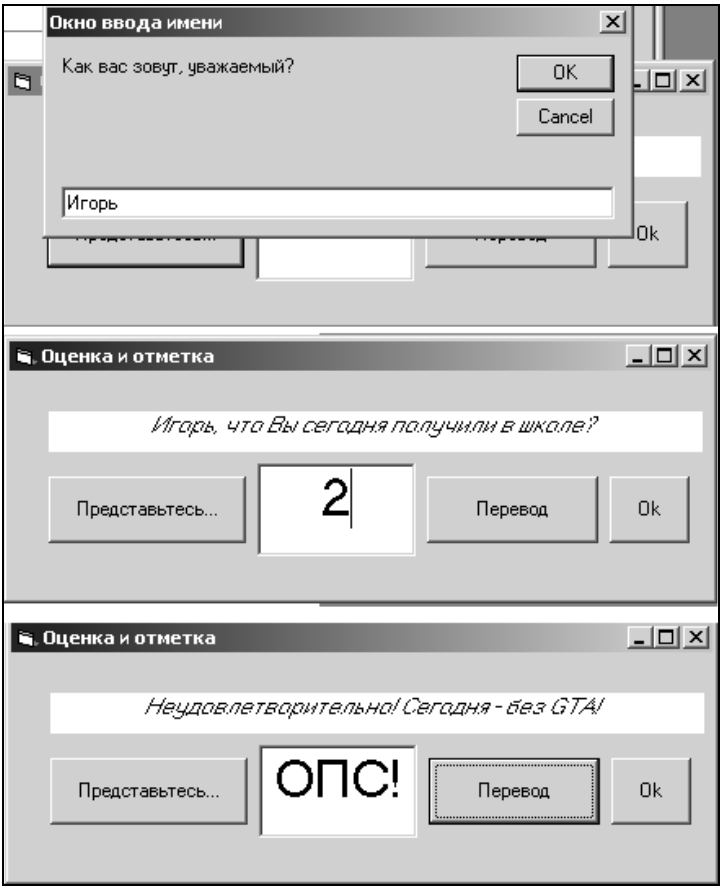


Рис. 2.2. Оценки и отметки в действии

В операторе `Select Case` возможно использование операций сравнения >, <, =, <>, >=, <=. В таком случае возможно задание интервала значений переменной и действий для них. Например, фрагмент кода, определяющий, положительную оценку получил наш герой или нет (оценка считывается из TextBox):

```
x = Val(Text1.Text)
Select Case x
'Рассматриваются значения x<3
Case Is <3
Text2.Text = "Оценка плоха. Придется пересдать ☹"
'Рассматриваются значения x=4 и x=5
Case 4 To 5
Text2.Text = "Нормально. Тема зачтена ☺"
End Select
```

Работа проекта представлена на рис. 2.3.

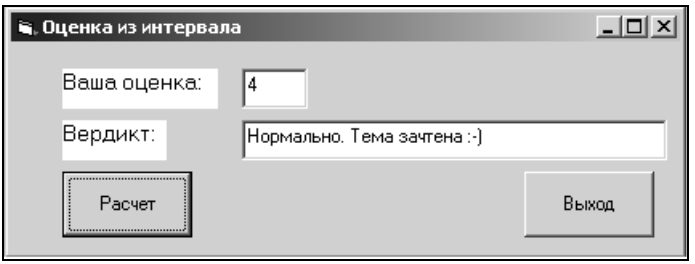


Рис. 2.3. Оценка из интервала

Для закрепления выполните следующие задания.

59. Вспомнив проект "Количество прожитых дней", доработаем его, сообщив пользователю, в какой день недели он родился. Для этого будет использована функция `WeekDay` (Дата), которая возвращает цифру от 1 до 7 (1 — воскресенье, 2 — понедельник, ..., 7 — суббота).

    Пример командного кода с использованием функции `WeekDay` (допустим, что в объекте TextBox введена дата рождения пользователя):

    ```
    x=WeekDay(Text1.Text)
    Select Case x
    ```

```
        Case 1
        Text1.Text= n + ", вы родились в воскресенье!"
        Case 2
        Text1.Text= n + ", вы родились в понедельник!"

End Select
```

Форма проекта представлена на рис. 2.4.

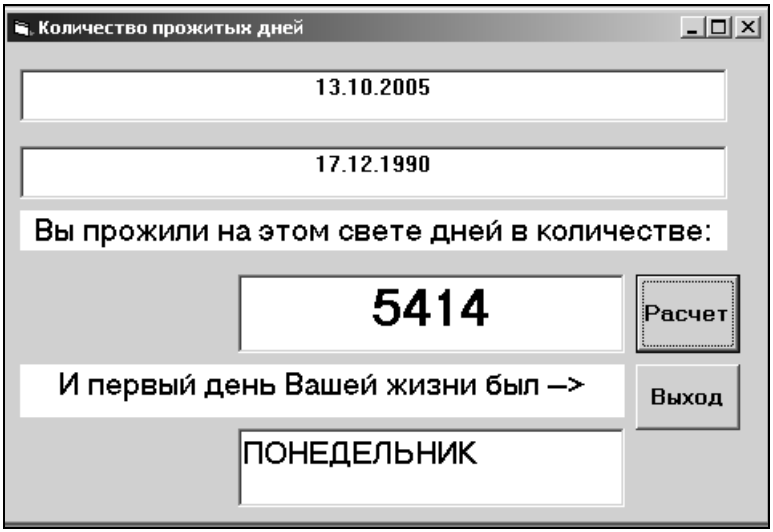


Рис. 2.4. Видно, в понедельник их мама родила...

60. По введенному номеру месяца определить его название и время года и загрузить соответствующую картинку (рис. 2.5). Надо использовать два `Select Case`: один для определения названия месяца, другой — для времени года.

Замечание

Для загрузки готовой картинки в ходе работы проекта используется команда: `[Object].Picture=LoadPicture("C:\...\picture.bmp")`.

61. Разработайте проект, который по знаку арифметической операции выводит ее название.

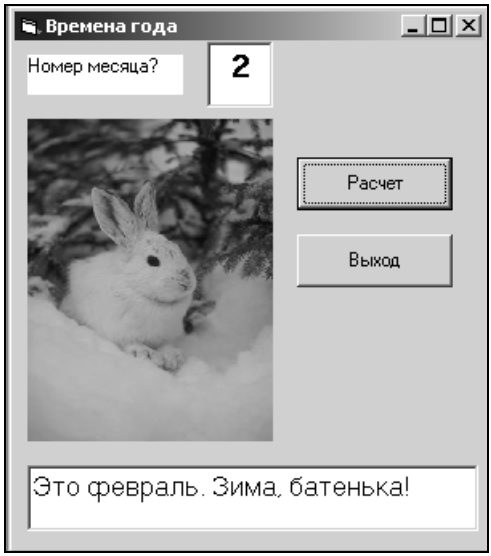


Рис. 2.5. Времена года

62. Разработайте проект, который определяет по заданному числу месяца и по дню недели первого числа этого месяца день недели для заданного числа. (Пример: первое число — вторник, тогда 17 — четверг, задали число 17.)
63. В восточных календарях принят 60-летний цикл, состоящий, в свою очередь, из пяти 12-летних подциклов. Подциклы обозначались цветом:

    0 — зеленый;

    1 — красный;

    2 — желтый;

    3 — белый;

    4 — черный.

    Внутри каждого подцикла годы носили названия животных:

    0 — свинья (или кабан);

    1 — крыса;

    2 — бык;

    3 — тигр;

4 — кролик (заяц или кот);

5 — дракон;

6 — змея;

7 — лошадь;

8 — овца (баран или коза);

9 — обезьяна;

10 — петух;

11 — собака.

Создайте проект с использованием операторов выбора, запрашивающий номер года нашей эры и печатающий его название по восточному календарю. К каждому выведенному году загрузите картинку с изображением соответствующего животного. Шрифт, которым выводится название года, должен быть цвета этого года на контрастном фоне.

Для расчета есть формулы:

```
NumberColor = ((9910-Year) mod 60) \ 12
NumberAnimal = (Year-3) mod 12
```

Для проверки: 1966 г. — год красной лошади, 1984 г. — зеленой крысы. Форма проекта представлена на рис. 2.6.

Рис. 2.6. Восточный календарь

64. Усложненный вариант предыдущего задания. Теперь требуется определить не только название введенного с клавиатуры года, но и год наступления "светлого будущего", до которого вы, безусловно, доживете! Делаем это поэтапно. Приведу примерный словесный алгоритм. После введения исходного четырехзначного года с клавиатуры вам необходимо выделить составляющие его цифры и записать их в переменные `A1`, `A2`, `A3` и `A4`. (Если в номере года есть нули, то в соответствующие переменные записать 4.) Затем необходимо взять от них синусы по модулю. Из полученных значений выделить по две цифры после запятой и сложить их. Если сумма больше либо равна 10, то сложить еще раз. Должно получиться 4 цифры, из которых надо составить четырехзначное число. Это и будет номер года наступления светлого будущего. Осталось определить его название по восточному календарю и вывести результаты на экран. Для проверки: при исходном годе 1986 годом светлого будущего станет 3589.

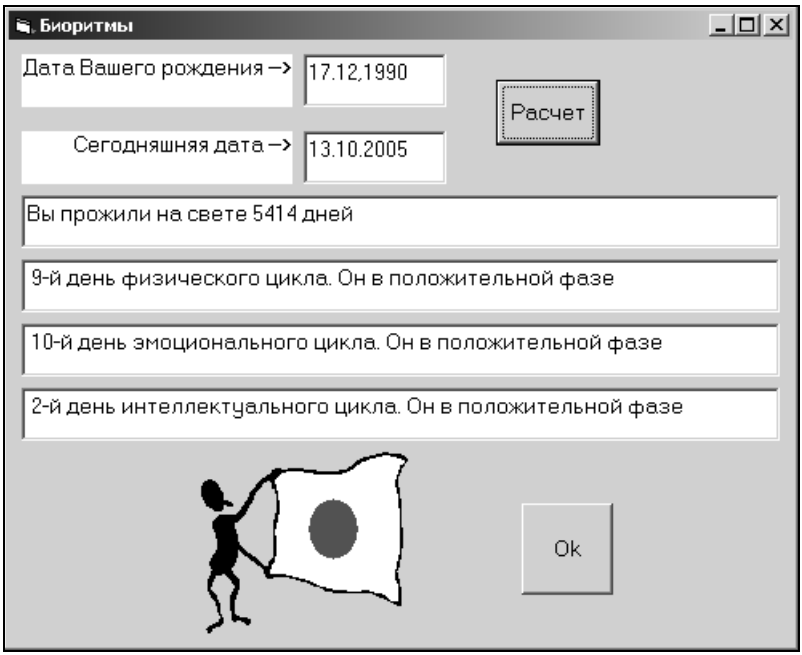


Рис. 2.7. Биоритмы once more

Вспомним проект "Биоритмы" (задание 16). Теперь, зная `Select Case`, мы можем не только указать, какой день того или иного цикла идет у пользователя, но и, по крайней мере, подсказать ему, в каком периоде (положительном или отрицательном) они находятся. По-моему, это уже просто для вас, таких опытных специалистов.

А форма проекта может выглядеть примерно так (рис. 2.7). У этого человека просто все супер! Надеюсь, у вас тоже ☺?

2.2. Использование для выбора переключателей

Попробуем непосредственно на форме разместить переключатели и посмотреть, как они работают. Пусть форма имеет вид, представленный на рис. 2.8.

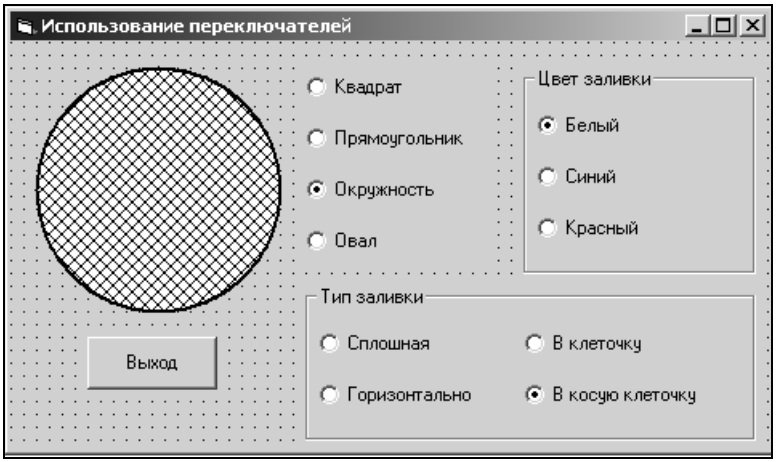


Рис. 2.8. Переключатели

Надо разместить Shape для фигур, четыре OptionButton для типа фигуры. Потом растянуть рамку Frame для цвета и поместить там три OptionButton, еще Frame для типа заливки с четырьмя OptionButton и, наконец, CommandButton для выхода.

Задать объектам формы соответствующие свойства `Caption` и размеры и написать командный код для каждого переключателя. Он будет выглядеть так:

```
'Задание типа фигур. Переключатели 1-4
Private Sub Option1_Click()
Shape1.Shape = 1
End Sub
Private Sub Option2_Click()
Shape1.Shape = 0
End Sub
Private Sub Option3_Click()
Shape1.Shape = 3
End Sub
Private Sub Option4_Click()
Shape1.Shape = 2
End Sub
'Задание цвета фигур
Private Sub Option5_Click()
Shape1.BackColor = vbWhite
End Sub
Private Sub Option6_Click()
Shape1.BackColor = vbBlue
End Sub
Private Sub Option7_Click()
Shape1.BackColor = vbRed
End Sub
'Задание типа заливки
Private Sub Option8_Click()
Shape1.FillStyle = 0
End Sub
Private Sub Option9_Click()
Shape1.FillStyle = 2
End Sub
Private Sub Option10_Click()
Shape1.FillStyle = 6
```

```
End Sub
Private Sub Option11_Click()
Shape1.FillStyle = 7
End Sub
'Процедура выхода
Private Sub Command1_Click()
End
End Sub
```

А теперь попробуйте выполнить следующие два задания.

65. Используя проект задания 5, измените его таким образом, чтобы было 6 переключателей, а на месте Image появлялся бы соответствующий кубик (рис. 2.9).

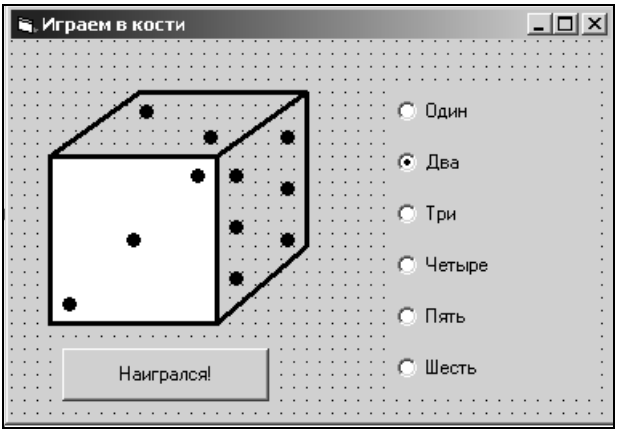


Рис. 2.9. Играем в кости

66. Используя проект задания 48 про смайлики, разработайте новый, в который добавьте еще один смайлик — нейтральный, и расположите их в виде светофора. Расположите на форме три переключателя, дайте им соответствующие свойства `Caption` — грустный, нейтральный, веселый (рис. 2.10).

    Чтобы одновременно не появлялись все три рожицы, используйте управление свойством `Visible`, например, программный код для веселой рожицы:

```
Private Sub Option3_Click()
```

```
    Image1.Visible = False
    Image3.Visible = False
    Image3.Visible = True
    Image3.BorderStyle = 1
    Image3.Picture = LoadPicture("c:\pics\v_smile.bmp")
    End Sub
```

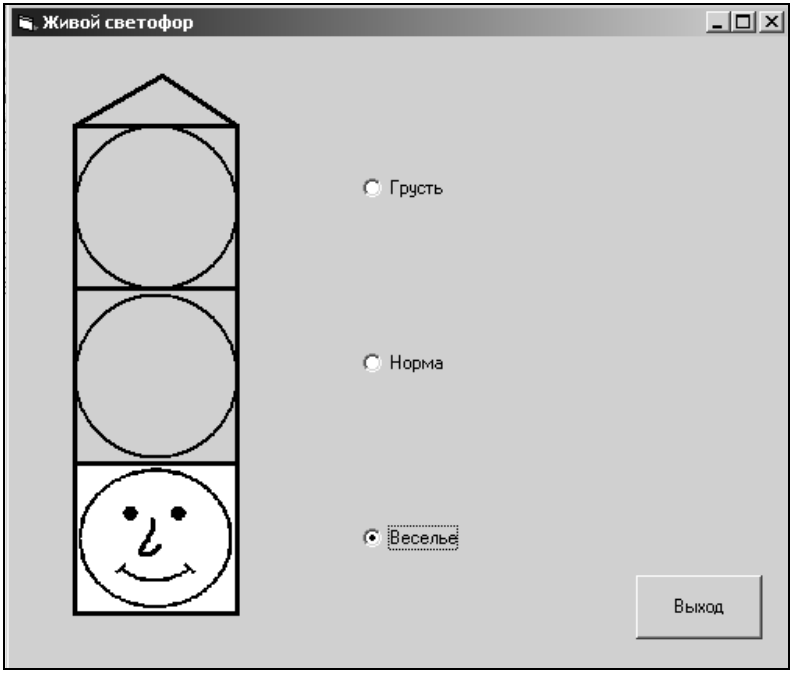


Рис. 2.10. Живой светофор

2.3. Ветвления при помощи условного оператора *If*...

Часто необходимо, чтобы часть программы выполнялась бы только при выполнении определенных условий. Решение данной проблемы заключается в использовании специальных конструкций, использующих операторы ветвления. Подробно рассмотрим данные конструкции.

2.3.1. *If … Then … End If*

Общий вид данной конструкции выглядит следующим образом:

```
If <логическое выражение> Then <список операторов>
End If
```

`<логическое выражение>` — это простое или сложное условие, или логическая константа (true или false).

Простое условие имеет следующий вид:

`<выражение1><операция сравнения><выражение2>`.

Возможны следующие операции сравнения:

`a>b, a<b, a=b, a>=b, a<=b, a<>b`

Сложное условие состоит из простых условий, соединенных логическими операциями `AND` или `OR`.

Например: `(a<b) AND (c>=d)`.

Алгоритм выполнения данной конструкции:

- Вычисляется значение логического выражения.
- Если значение логического выражения true, то выполняется список операторов.
- Если значение логического выражения false, то ничего не выполняется.

> **Замечание**
>
> Операторы `If` и `Then` обязательно должны находиться на одной строке.

Пример использования:

```
x = InputBox ("Введите число от 1 до 20")
If x>10 Then
   a = x/10
End if
Print a
```

2.3.2. *If … Then … Else … End If*

Данная конструкция позволяет создавать дополнительную ветвь условного перехода. Общий вид данной конструкции выглядит следующим образом:

```
If <логическое выражение> Then
<список операторов1>
Else
<список операторов2>
End If
```

Алгоритм выполнения данной конструкции:

- Вычисляется значение логического выражения.
- Если значение логического выражения true, то выполняется список операторов 1.
- Если значение логического выражения false, то выполняется список операторов 2.

Пример использования:

```
x = InputBox ("Введите число от 1 до 20")
If x>10 Then
   a = x/10
Else
a = x*5
End if
Print a
```

2.3.3. *If … Then … ElseIf … End If*

Данная конструкция позволяет организовывать несколько вложенных друг в друга операторов `If`. Общий вид данной конструкции выглядит следующим образом:

```
If <логическое выражение1> Then
<список операторов1>
ElseIf <логическое выражение2> Then
<список операторов2>

ElseIf <логическое выражениеN> Then
```

```
<список операторовN>
End If
```

Алгоритм выполнения данной конструкции:

- Вычисляется значение логического выражения 1.
- Если значение логического выражения 1 — true, то выполняется список операторов 1.
- Если значение логического выражения 1 — false, то вычисляется значение логического выражения 2.
- Если значение логического выражения 2 — true, то выполняется список операторов 2.
- Если значение логического выражения 2 — false, то вычисляется значение логического выражения 3.
- Если значение логического выражения N — true, то выполняется список операторов N.
- Если значение логического выражения N — false, то ничего не происходит.

Замечание

Операторы `If` и `Then` обязательно должны находиться на одной строке, так же как и операторы `ElseIf` и `Then`.

Пример использования:

```
a = InputBox ("Введите целое число A")
If a=1 Then
   b=10
   c= b + 20
ElseIf a=2 Then
   B = 20
   C = b + 30
ElseIf a=3 Then
   B = 30
   C = b + 40
End if
Print C
```

Кстати, определите из последнего примера, что будет напечатано, если $a = 2$, и что, если $a = 5$.

Напишите программы, которые в зависимости от введенного числа либо вычисляют функцию, либо выдают сообщение, что функция не определена.

67. $y = \frac{1}{x}$.

68. $y = \sqrt{x^2 - 1}$.

69. Напишите программу для вычисления функции:

$$y = \begin{cases} \frac{\cos x}{x}, & x > 0 \\ x \sin x, & x \le 0 \end{cases}.$$

70. Напишите программу, определяющую четность или нечетность введенного с клавиатуры целого числа.

71. Напишите программу, находящую меньшее из двух, введенных с клавиатуры чисел.

72. По четырехзначному номеру года, запрошенному с клавиатуры, определите номер столетия (например, для 1492 г. — ответ XV век, для 1812 г. — XIX век). Учесть, что началом века считается первый, а не нулевой, год. (То есть 2000-й год из астрономии — последний год XX века).

73. Запишите в виде одного условного оператора указанные действия:

- известно, что из трех чисел $A1$, $A2$ и $A3$ одно отлично от других, равных между собой. Присвоить номер этого числа переменной `N`;
- $y = \begin{cases} \cos^2 x, & 0 < x < 2 \\ 1 - \sin^2 x \end{cases}$.

74. Напишите программу, запрашивающую у пользователя три разных целых положительных числа и находящую сумму двух наименьших из них.

75. В задании 12 мы уже вычисляли площадь треугольника по формуле Герона. Теперь усложним нашу задачу. С клавиатуры запрашиваются целые числа a, b и c. Программа проверяет, можно ли, представив, что эти числа означают длины сторон, составить из них треугольник, затем рисует его на экране и вычисляет его площадь. Если треугольник с такими сторонами не существует, то на экране появляется соответствующее сообщение и картинка.

    Проект может выглядеть примерно так, как на рис. 2.11 или рис. 2.12.

76. В стене существует квадратное отверстие $N \times N$ см. Имеется кирпич с измерениями A, B и C. Определить, пройдет он в отверстие или нет, если подавать его можно только параллельно стенкам отверстия. Решение задания хорошо бы сопроводить рисунком отверстия и кирпича (рис. 2.13).

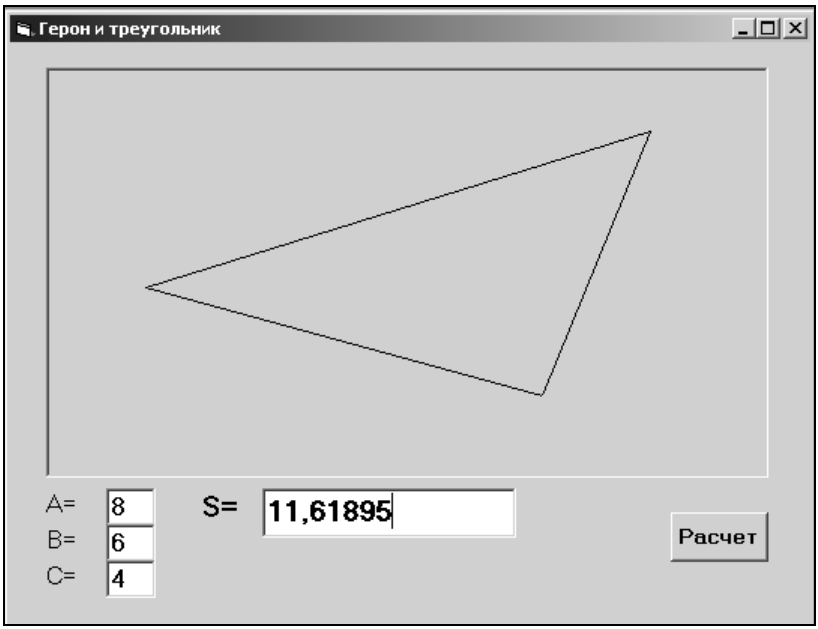


Рис. 2.11. Вот такой вот треугольник...

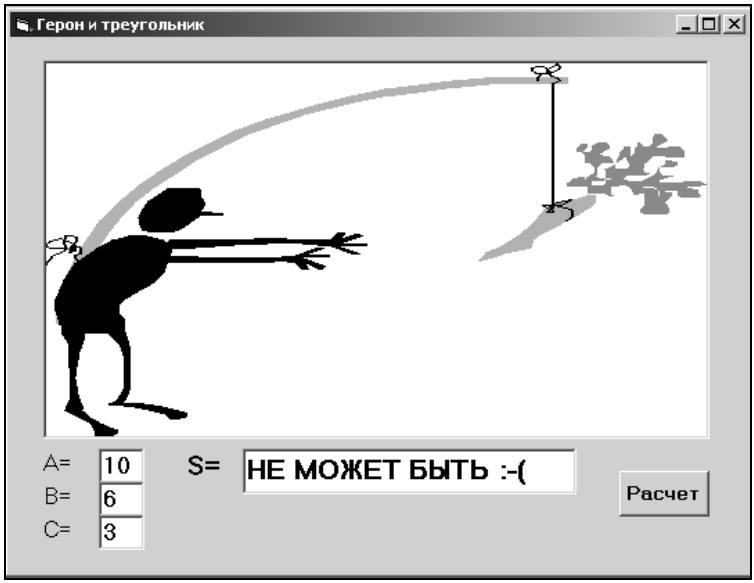


Рис. 2.12. А вот и нет такого треугольника!

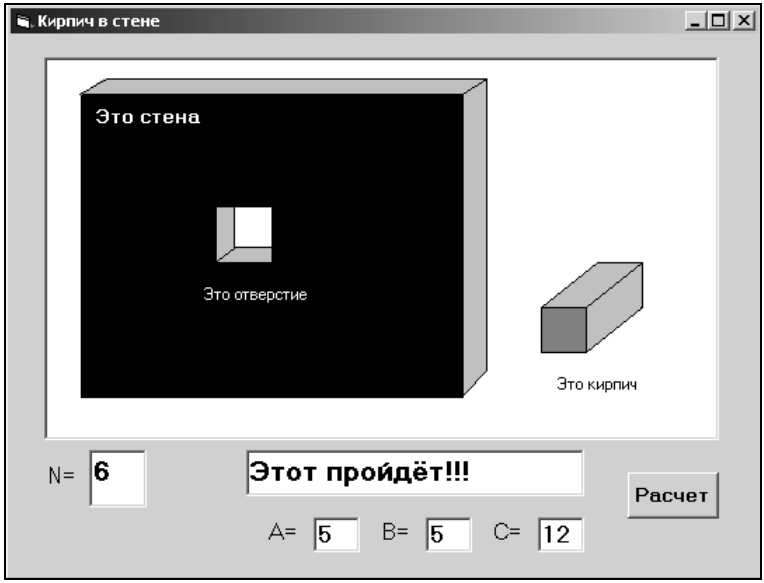


Рис. 2.13. Кирпич в стене

77. По введенным с клавиатуры коэффициентам квадратного уравнения A, B и C найдите его корни. Рассмотрите шесть возможных вариантов:

    $A = B = C = 0$, корней бесчисленное множество (X — любое);

    $A = B = 0$, $C \neq 0$, уравнение не имеет корней;

    $A = 0$, $B \neq 0$, $C \neq 0$, вырожденное квадратное уравнение, имеется один корень (формулу вычисления корня найдите сами);

    $D < 0$, где D — дискриминант, который предварительно надо вычислить; уравнение не имеет вещественных корней;

    $D = 0$, уравнение имеет два одинаковых корня (вывести их значения);

    $D > 0$, уравнение имеет два различных вещественных корня (вычислить и вывести их значения).

    Для проверки правильности работы программы предлагается шесть тестовых вариантов исходных данных:

    `A=B=C=0`;

    `A=B=0, C=1`;

    `A=0, B=3, C=6` (должно получиться `X=-2`);

    `A=5, B=3, C=2`;

    `A=1, B=2, C=1` (должно получиться `X1=X2=-1`);

    `A=2, B=5, C=2` (должно получиться `X1=-2, X2=-8`).

    В проекте предусмотрите запрос трех коэффициентов, вывод уравнения в привычном алгебраическом виде и вывод результата.

78. Осуществите запрос трех целых различных чисел с клавиатуры. Выведите на экран наибольшее и наименьшее.

79. Осуществите запрос с клавиатуры у тренера олимпийской сборной России по марафону о результатах в часах, минутах и секундах трех победителей чемпионата России. Если какие-то два результата различаются менее чем на 5 сек., выведите сообщение "Вот так шла борьба за _____ медаль". В сообщении должно быть указано достоинство медали (золотая, серебряная), за которую шла борьба. В противном

случае, вычислить среднюю скорость победителя в км/час, если принять длину марафонской дистанции 42 км 195 м.

80. А сейчас мы попробуем сделать пока не очень красивый, но очень простой вариант телевизионной игры "Кто хочет стать миллионером!". Придумайте пять любых вопросов, и к каждому из них четыре варианта ответов. Теперь я попробую словесно описать алгоритм, а вы — перевести его на Visual Basic. Итак, запрашиваем у игрока имя и узнаем, желает ли он играть. Если не желает, прощаемся, если желает — приветствуем и предлагаем первый вопрос с вариантами ответов. Запрашиваем у игрока с клавиатуры, какой вариант он выбирает. В случае правильного ответа начисляем ему сто очков и переходим ко второму вопросу. Если ответ неверен, то выражаем сожаление и прощаемся. Первый вопрос — 100 очков, второй — 200, третий — 300, четвертый — 500, пятый — 1000. Если игрок правильно отвечает на все пять вопросов, то поздравляем его и заканчиваем программу.

81. Каждую пятницу члены Клуба толстяков выстраиваются в определенном порядке и взвешиваются. Напишите программу, которая хранит данные взвешивания 10-ти членов Клуба за прошлую неделю. Затем программа запрашивает новые данные взвешивания и для каждого члена Клуба либо выводит поздравление в случае похудения, либо величину прибавки веса с сожалением.

82. Определите и выведите на экран номер квадранта, в котором расположена точка $A(x, y)$, x и y — заданные целые числа.

83. Составьте программу, которая определяет, принадлежит ли точка $M(x, y)$ кругу с центром в точке $Z(a, b)$ и радиусом, равным r.

84. Составьте программу, которая определяет, принадлежит ли точка $N(x, y, z)$ шару с центром в точке $Z(a, b, c)$ и радиусом r.

85. Составьте программу, которая определяет, принадлежит ли точка $M(x, y)$ окружности с центром в точке $Z(a, b)$ и радиусом r.

86. Составьте программу, которая определяет, принадлежит ли точка $N(x, y, z)$ сфере с центром в точке $Z(a, b, c)$ и радиусом r.

87. Определите, какая из двух фигур (круг или квадрат) имеет большую площадь.
88. Известно, что сторона квадрата равна *a*, радиус круга *r*. Выведите на экран название и значение площади большей фигуры.

2.4. Случайные числа

Для многих задач программирования необходимы случайные числа. Особенно это касается всяческих игр и компьютерных моделей, просчитывающих сложные реальные природные процессы.

Для получения их на компьютере используется оператор `RANDOMIZE`, посредством которого мы доводим до компьютера, что будем пользоваться случайными числами, получаемыми от показаний встроенного таймера, имеющегося в каждом компьютере.

Затем можно приступать к "изготовлению" случайных чисел. Делается это примерно так:

```
RANDOMIZE
X=RND
PRINT X
```

По исполнении этого программного кода на форме появится некое дробное число в пределах от 0 до 1, например: 0,2067843 или 0,9216529.

Действительно, вероятность того, что такие числа повторятся, крайне мала. Но часто надо имитировать появление случайных целых чисел, например, при моделировании броска монеты "Орел-решка" или игральной кости — от 1 до 6.

Тут на помощь придет оператор `Fix`, который любезно (или безжалостно ☺) отбросит от любой дроби дробную часть.

Вот пример для монеты (рис. 2.14). В программном коде вы можете видеть, что для получения только нуля (означающего решку) или единицы (означающей орла) используется оператор

```
x = Fix (Rnd * 2)
```

Сначала полученная дробь умножается на 2 (таким образом минимальное число может быть 0,0000002, а максимальное — 1,9999998. При отбрасывании дробной части и остаются только нули или единицы, чего мы, собственно, и хотели добиться.

Программный код для монеты:

```
Option Explicit
Dim x As Single
Private Sub Command1_Click()
Randomize
FontSize = 14
x = Fix(Rnd * 2)
If x = 0 Then
Shape1.Visible = True
Line (800, 1100)-(1800, 1400), vbYellow, BF
PSet (800, 1100), vbYellow
Print "РЕШКА"
Else
Shape1.Visible = True
Line (800, 1100)-(1800, 1400), vbYellow, BF
PSet (800, 1100), vbYellow
Print "ОРЕЛ"
End If
End Sub
```

Рис. 2.14. Орел или решка...

Попробуйте самостоятельно написать оператор получения случайных чисел в предложенных ниже диапазонах.

89. От 1 до 10.
90. От −5 до +5.
91. От 10 до 20.
92. От 50 до 100.
93. От −35 до 65.

Замечание

Если вы смогли выполнить предыдущие пять заданий, то обратите внимание на закономерность и выведите общее правило для подобных задач.

94. Создайте проект, показывающий на форме игральную кость со случайным числом точек на верхней грани — от 1 до 6.
95. "Угадайка". Программа задумывает случайное число от 1 до 10, не выводя его на экран. Человек должен угадать его за три попытки. В каждой попытке компьютер выводит сообщение о том, больше его число или меньше. В случае отгадывания выводится поздравление, иначе — сожаление и загаданное число. Все сообщения пользователю выводятся с обращением по имени, запрошенному в начале с клавиатуры (рис. 2.15).

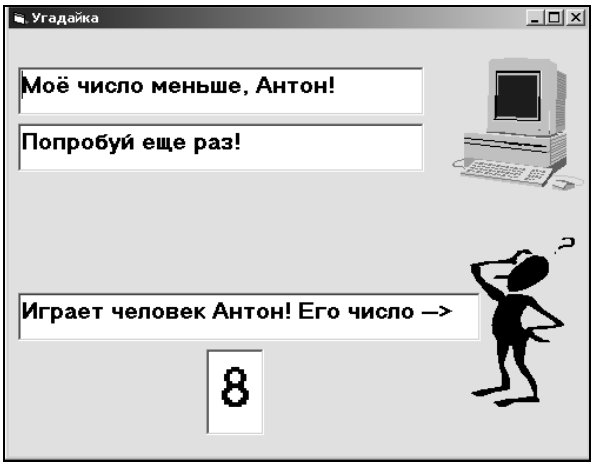


Рис. 2.15. Угадайка

96. Усложните предыдущее задание, предоставив человеку еще одну попытку, и, в случае угадывания, выведите на экран количество попыток.

2.5. Алгоритмы циклические

Довольно часто в программировании приходится, особенно при вычислениях, повторять одни и те же действия либо заданное количество раз, либо до наступления какого-либо события. Это достигается при помощи операторов цикла, которых в Visual Basic несколько разновидностей.

Рассмотрим сначала их теоретически.

Цикл состоит из *оператора цикла* и *тела цикла*. Оператор цикла — это его управляющая конструкция. Она определяет, сколько раз должны выполниться операторы, записанные в тело цикла, либо при каких условиях тело цикла должно повториться еще раз. Тип цикла определяется его оператором. Иными словами, по оператору часто определяют тип цикла.

2.5.1. Цикл *For...Next*

Данная конструкция служит для повтора тела цикла заданное число раз.

Общий вид данной конструкции выглядит следующим образом:

```
'заголовок цикла
For <переменная>=<начало> to <конец> [Step <шаг>]
   <оператор>
      ...    'тело цикла
   [<оператор>]
Next
```

`<переменная>` — переменная (параметр) цикла целого типа;

`<начало>` — начальное значение параметра цикла;

`<конец>` — конечное значение параметра цикла;

`<оператор>` — оператор тела цикла;

<шаг> — шаг цикла, то есть то значение, на которое увеличивается параметр цикла при каждом повторе. Шаг цикла по умолчанию равен 1.

Число выполнений цикла можно определить по следующей формуле:

```
(<конечное значение>-<начальное значение>)/<шаг> +1
```

Пример использования цикла For…Next.

Задача. Вывести на экран квадраты первых десяти натуральных чисел.

Программный код:

```
Option Explicit
Dim I As Integer
Private Sub Command1_Click()
FontSize = 14
For I = 1 To 10
Print I ^ 2 ;
Next
End Sub
```

Результат запуска виден на рис. 2.16.

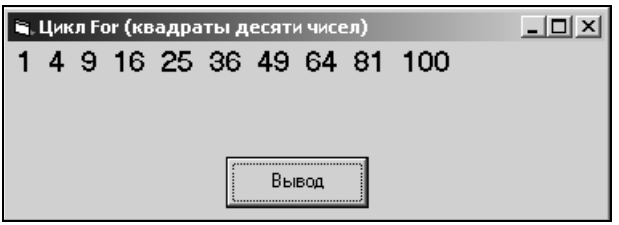


Рис. 2.16. Квадраты десяти чисел

Замечание

Обращаю внимание, что в цикле For…Next переменная, стоящая в заголовке цикла, должна быть обязательно целого типа. Кроме того, если выводящий на экран результаты исполнения программы оператор PRINT заканчивается точкой с запятой, то результаты будут выводиться в строчку, иначе — в столбик.

Еще один пример. Рисование концентрических окружностей (на манер мишени из первой части).

Программный код приведен ниже. Результат изображен на рис. 2.17.

```
Option Explicit
Dim r As Integer
Private Sub Command1_Click()
Picture1.Scale (0, 100)-(100, 0)
For r = 5 To 50 Step 10
Picture1.Circle (50, 50), r
Next
End Sub
```

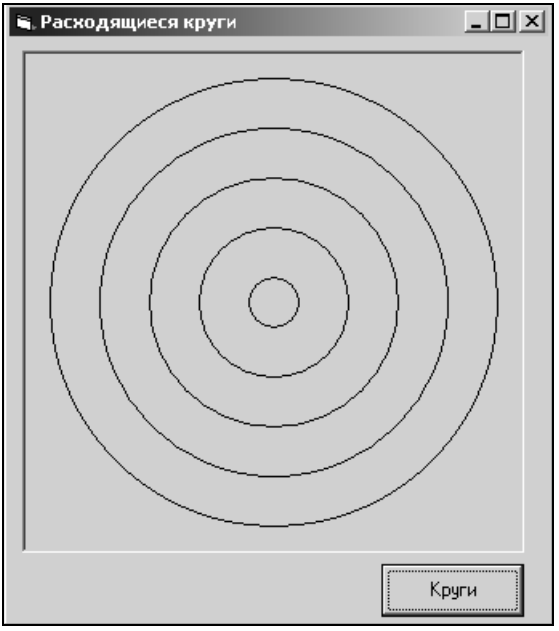


Рис. 2.17. Расходящиеся круги-1

Управление количеством окружностей легко производится уменьшением-увеличением шага. При уменьшении вдвое получаем вдвое больше окружностей (рис. 2.18).

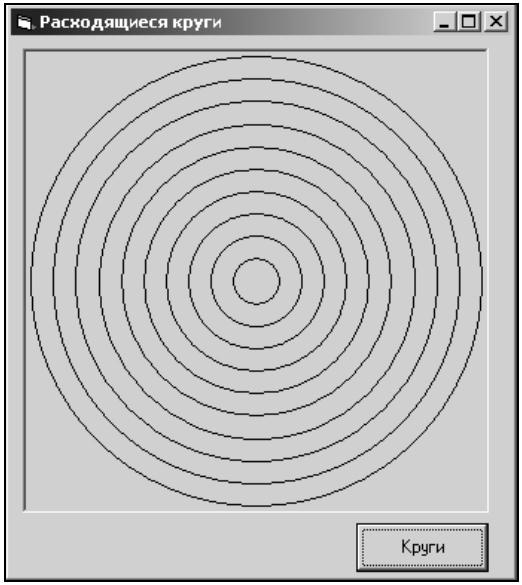


Рис. 2.18. Расходящиеся круги-2

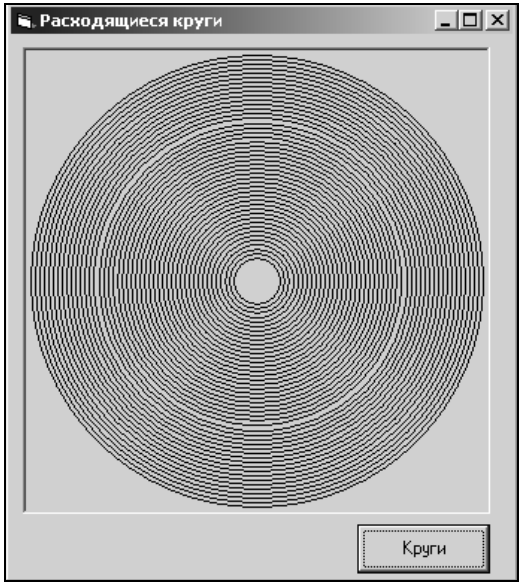


Рис. 2.19. Расходящиеся круги-3

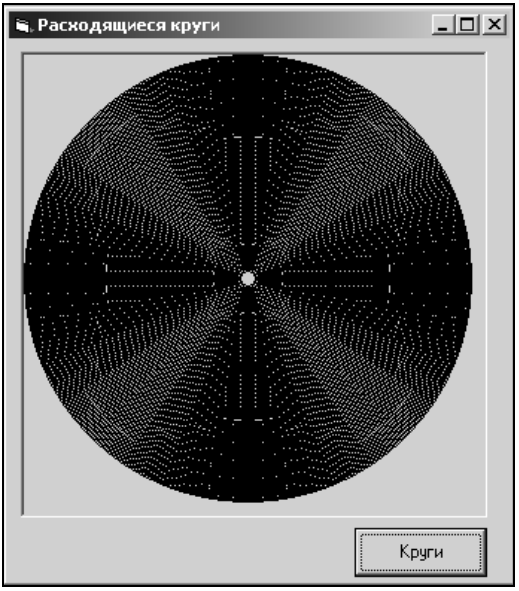


Рис. 2.20. Поверхность компакт-диска

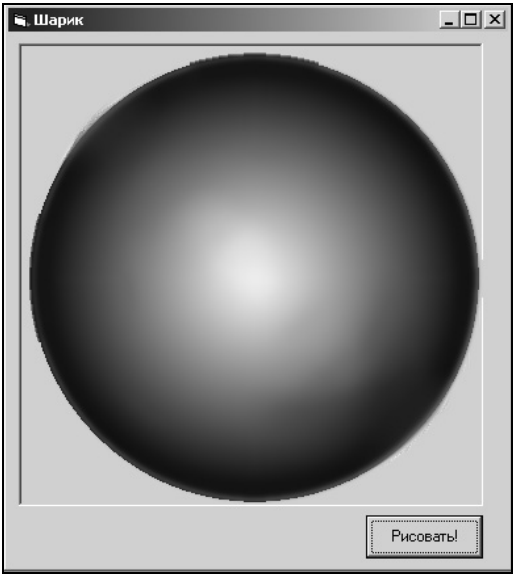


Рис. 2.21. Шарик

При уменьшении шага до единицы получаем нечто, отдаленно напоминающее поверхность компакт-диска (рис. 2.19).

Если бы при выборе шкалы оставить размер PictureBox в пикселях, то результат был бы лучше (рис. 2.20) — попробуйте!

Если же постепенно менять при выводе каждой окружности ее цвет от черного до белого, то получится изображение шара (рис. 2.21) — довольно неплохое, правда!?

2.5.2. Построение графиков при помощи цикла *For...Next*

При помощи рассмотренной разновидности циклов удобно строить графики функций по точкам. Пример такого построения для квадратичной функции $y = ax^2 + bx + c$ приведен на рис. 2.22.

Программный код:

```
Option Explicit
Dim a As Single, b As Single, c As Single
Dim x As Single, y As Single
Dim i As Integer
Private Sub Command1_Click()
Picture1.Scale (-5, 8)-(5, -8)
'Считывание коэффициентов уравнения
a = Text1.Text
b = Text2.Text
c = Text3.Text
'Ось абсцисс
Picture1.Line (-5, 0)-(5, 0)
For i = -5 To 5
Picture1.PSet (i, 0)
Picture1.Print i
Next
'Ось ординат
Picture1.Line (0, -8)-(0, 8)
For i = -8 To 8
Picture1.PSet (0, i)
```

```
Picture1.Print i
Next
'Цикл построения графика по точкам
For x = -5 To 5 Step 0.005
y = a * x * x + b * x + c
Picture1.PSet (x, y)
Next
End Sub
```

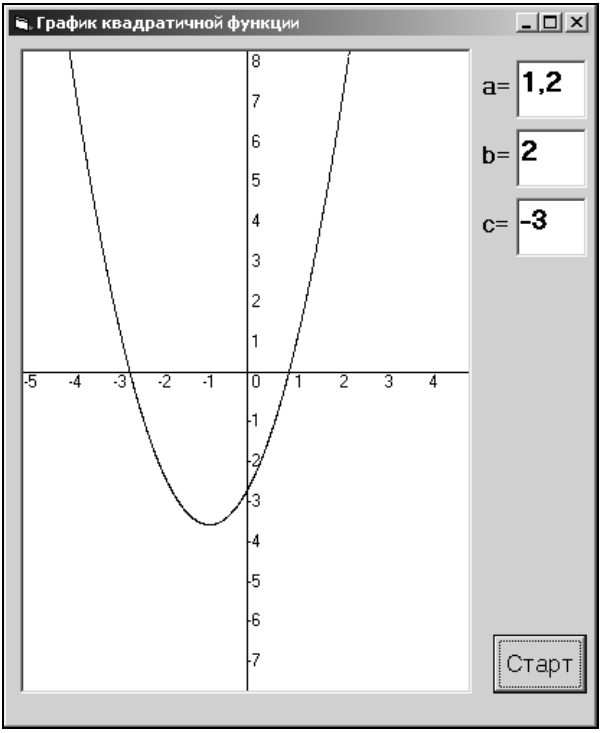


Рис. 2.22. График квадратичной функции

Замечание

Обратите внимание на шаг в построении графика функции. В нашем случае он равен 0,005. Попробуйте его изменять в сторону уменьшения и увеличения и сделайте правильные выводы ☺!

Итак, следующие самостоятельные задания.

97. Постройте график синуса.
98. Постройте график гиперболы $y = \frac{1}{x}$ (запись в программе: `y = 1/x`).
99. Постройте график корня квадратного.
100. Вспомните задание про исследование квадратного уравнения (задание 70) и дополните его построение графика функции (в случае таковой возможности).
101. А теперь попробуем построить поверхности вращения, образующиеся вращением графиков вокруг осей координат.
102. Нарисуйте поверхность (рис. 2.23), образованную вращением вокруг оси *Y*, графика функции $y = x^2$ (запись в программе: `Y = X * X` или `Y = X^2`).

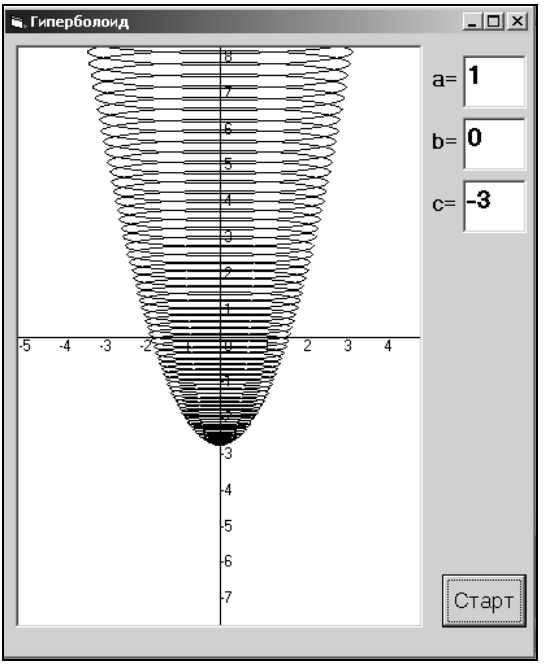


Рис. 2.23. Гиперболоид

Замечание

Попробуйте строить график не точками, а эллипсами, радиус которых... Вот именно ☺!

Изобразите на экране поверхность, образованную вращением вокруг оси X графиков представленных ниже функций.

103. $y = \dfrac{1}{(1+x^2)}$ (запись в программе: `Y = 1/(1 + X ^ 2)`).

104. $y = \dfrac{1}{x}$ (запись в программе: `Y = 1/X)`.

Ну что, двигаемся дальше? Попробуем совместить полученные знания о случайных числах с циклом `For…Next` в графике. Изобразим звездное небо (рис. 2.24).

Рис. 2.24. Звездное небо

Делаем на форме PictureBox размером, скажем, 7500 на 5000 твипов. Размещаем командную кнопочку и пишем такой программный код:

```
Option Explicit
Dim i As Integer, x As Integer, y As Integer, n As Integer
Private Sub Command1_Click()
Randomize
For i = 1 To 1000
x = Fix(Rnd * 7500)
y = Fix(Rnd * 5000)
n = Fix(Rnd * 3) + 1
Picture1.DrawWidth = n
Picture1.PSet (x, y), vbWhite
Next
End Sub
```

Пояснения к программному коду:

- переменная `i` (счетчик цикла) — отсчитывает тысячу звезд (можно больше или меньше);
- переменные `x` и `y` — случайные координаты звезды на PictureBox;
- переменная `n` — меняется случайным образом от 1 до 3 и заведует толщиной точки (от этого-то наши звездочки такие разные).

Далее — простор для фантазии — случайные цвета, окружности etc.

А теперь самостоятельно выполните следующие задания.

105. Изобразите тысячу случайных отрезков на вашей форме.
106. А теперь тысячу прямоугольников (лучше окантованных линией извне).
107. Ну а теперь — случайные окружности и эллипсы.
108. Давайте устроим в центре экрана маленький взрыв. Он будет изображен сотней отрезков разноцветных прямых, сходящихся в центре вашей формы. Длина их пусть лежит случайным образом в пределах от 500 до 1500 твипов (рис. 2.25).

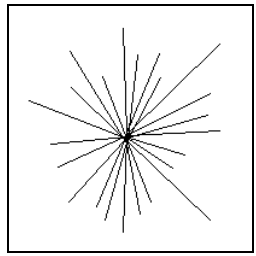


Рис. 2.25. Маленький взрыв

Теперь познакомимся с другими вариантами организации циклических алгоритмов в Visual Basic, приведем несколько примеров их использования и много-много задач, чтобы жизнь не была такой скучной ☺. Мы же хотим выработать в себе алгоритмическое мышление!

2.5.3. Цикл *While ... Wend*

Этот тип цикла служит для того, чтобы повторять тело цикла заранее неизвестное количество раз. Количество повторений определяет ситуация, возникающая во время выполнения тела цикла.

Общий вид данной конструкции выглядит следующим образом:

```
While <логическое выражение>  'заголовок цикла
     <оператор>
      ...                     'тело цикла
     [<оператор>]
      ...
Wend
```

`<логическое выражение>` — это простое или сложное условие, или логическая константа (true или false).

Пока `<логическое выражение>` возвращает true, тело цикла выполняется, а как только `<логическое выражение>` возвратит false, то работа продолжится со следующего оператора за служебным словом `Wend`. Естественно, если в процессе работы программы условие никогда не станет ложным, то цикл будет повторяться бесконечно, то есть программа зависнет. Поэтому надо обязательно предусмотреть возможность выхода из цикла.

Пример использования цикла `While … Wend`.

Задача. С клавиатуры вводятся целые положительные числа. Пока числа меньше 100, программа вычисляет квадратный корень от введенного числа (рис. 2.26). Как только введенное число оказывается больше либо равно 100, программа заканчивает работу с выводом соответствующего сообщения (рис. 2.27).

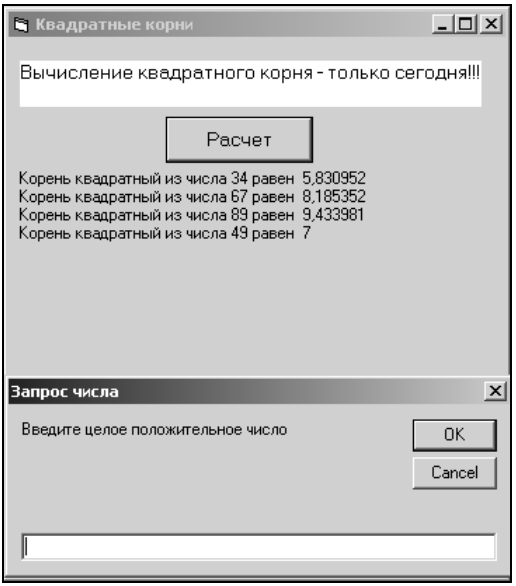


Рис. 2.26. Вычисление квадратного корня

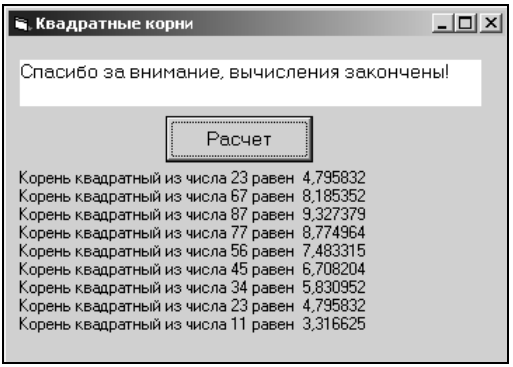


Рис. 2.27. Окончание вычислений

Программный код:

```
Option Explicit
Dim z As Integer
Dim s As Single
Private Sub Command1_Click()
CurrentY = 1400
z = InputBox("Введите целое положительное число", _
"Запрос числа")
While z < 100
s = Sqr(z)
CurrentX = 100
Print "Корень квадратный из числа "; z; _
" равен "; s
z = InputBox("Введите целое положительное число", _
"Запрос числа")
Wend
CurrentX = 100
Label1.Caption = "Спасибо за внимание, вычисления закончены!"
End Sub
```

Примечание

Ввод числа осуществляется с помощью оператора `InputBox`. `CurrentX` отвечает за координату *X* вывода результатов вычислений, `CurrentY` — за координату *Y*. Сообщение об окончании вычислений поступает в `Label1`. Еще было бы неплохо в программный код вставить проверку на положительность вводимого числа.

2.5.4. Цикл *Do While... Loop*

Этот тип цикла служит для того, чтобы пока выполняется условие, повторять тело цикла (проверка условия в начале цикла).

Общий вид данной конструкции выглядит следующим образом:

```
Do While <логическое выражение>   'заголовок цикла
     <оператор>
      ...                         'тело цикла
```

```
     [<оператор>]
      ...
Loop
```

<логическое выражение> — это простое или сложное условие, или логическая константа (true или false).

Пока <логическое выражение> возвращает true, тело цикла выполняется, а как только <логическое выражение> возвратит false, то работа продолжится со следующего оператора за служебным словом Loop.

Пример использования цикла Do While…Loop для той же задачи вычисления квадратного корня (*см. раздел "Цикл WhileWend"*).

Фрагмент программного кода:

```
z = InputBox("Введите целое положительное число", "Запрос
числа")
Do While z < 100
s = Sqr(z)
CurrentX = 100
Print "Корень квадратный из числа "; z; " равен "; s
z = InputBox("Введите целое положительное число", "Запрос числа")
Loop
```

2.5.5. Цикл *Do... Loop While*

Этот тип цикла служит для того, чтобы повторять тело цикла, пока выполняется условие (проверка условия в конце цикла).

Общий вид данной конструкции выглядит следующим образом:

```
Do
   <оператор>
   ...         'тело цикла
   [<оператор>]
   ...
Loop While <логическое выражение>
```

<логическое выражение> — это простое или сложное условие, или логическая константа (true или false).

Вначале выполняется тело цикла, расположенное после ключевого слова Do, а затем проверяется <логическое выражение>.

Пока `<логическое выражение>` возвращает true, тело цикла выполняется, а как только `<логическое выражение>` возвратит false, то работа продолжится со следующего оператора после `Loop While <логическое выражение>`.

Пример использования цикла `Do … Loop While` для все той же задачи вычисления квадратного корня (*см. раздел "Цикл While Wend"*).

Фрагмент программного кода:

```
z = InputBox("Введите целое положительное число", "Запрос числа")
Do
s = Sqr(z)
CurrentX = 100
Print "Корень квадратный из числа "; z; " равен "; s
z = InputBox("Введите целое положительное число", "Запрос числа")
Loop While z < 100
```

2.5.6. Цикл *Do Until... Loop*

Этот тип цикла служит для того, чтобы пока условие не выполняется, повторять тело цикла (проверка условия содержится в начале цикла).

Общий вид данной конструкции выглядит следующим образом:

```
Do Until <логическое выражение>   'заголовок цикла
   <оператор>
   ...                            'тело цикла
   [<оператор>]
   ...
Loop
```

`<логическое выражение>` — это простое или сложное условие, или логическая константа (true или false).

Пока `<логическое выражение>` возвращает false, тело цикла выполняется, а как только `<логическое выражение>` возвратит true, то работа продолжится со следующего оператора за служебным словом `Loop`.

Пример использования цикла `Do Until…Loop` для все той же задачи вычисления квадратного корня (*см. раздел "Цикл While Wend"*).

Фрагмент программного кода:

```
z = InputBox("Введите целое положительное число", _
"Запрос числа")
Do Until z>=100
s = Sqr(z)
CurrentX = 100
Print "Корень квадратный из числа "; z; _
" равен "; s
z = InputBox("Введите целое положительное число", _
"Запрос числа")
Loop
```

2.5.7. Цикл *Do... Loop Until*

Этот тип цикла служит для того, чтобы повторять тело цикла, пока условие не выполняется (проверка условия содержится в конце цикла).

Общий вид данной конструкции выглядит следующим образом:

```
Do
   <оператор>
   ...        'тело цикла
   [<оператор>]
   ...
Loop Until <логическое выражение>
```

`<логическое выражение>` — это простое или сложное условие, или логическая константа (true или false).

Вначале выполняется тело цикла, а затем проверяется `<логическое выражение>`. Пока `<логическое выражение>` возвращает false, тело цикла выполняется, а как только `<логическое выражение>` возвратит true, то работа продолжится со следующего оператора после `Loop Until <логическое выражение>`.

Пример использования цикла `Do…Loop Until` для все той же задачи вычисления квадратного корня (*см. раздел "Цикл While Wend"*).

Фрагмент программного кода:

```
z = InputBox("Введите целое положительное число", _
"Запрос числа")
Do
s = Sqr(z)
CurrentX = 100
Print "Корень квадратный из числа "; z; _
" равен "; s
z = InputBox("Введите целое положительное число", _
"Запрос числа")
Loop Until z >= 100
```

2.5.8. Основные правила выбора типа цикла

Существуют определенные правила выбора типа цикла. Приведем основные из них:

- если вам заранее известно число повторений тела цикла, лучше всего использовать оператор цикла `For`;
- если вам заранее не известно число повторений тела цикла и если окончание цикла зависит от выполнения некоторого условия, лучше использовать конструкции `While…Wend`, `Do While…Loop` или `Do Until…Loop`;
- если необходимо, чтобы цикл всегда выполнялся хотя бы один раз, то используйте конструкции `Do…Loop While` или `Do…Loop Until`.

Ну а теперь предлагаю выполнить много-много самостоятельных заданий. Попробуйте решить каждую с использованием разных типов циклов.

109. Изобразите на одной форме решетку (рис. 2.28), переход (рис. 2.29), глаз (рис. 2.30), взрыв (рис. 2.31) и шарик, надувающийся из центра PictureBox окружностями случайных цветов гаммы RGB (рис. 2.32).

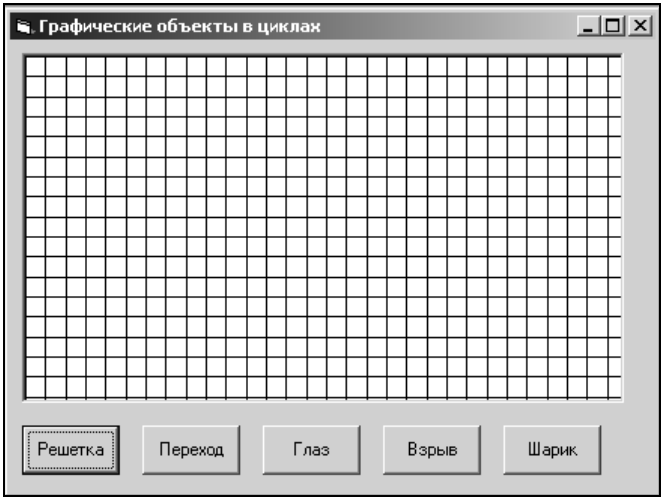


Рис. 2.28. Решетка

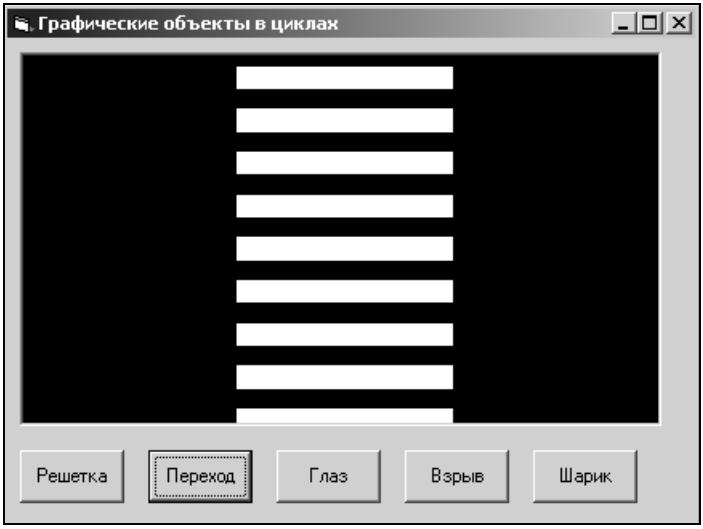


Рис. 2.29. Переход

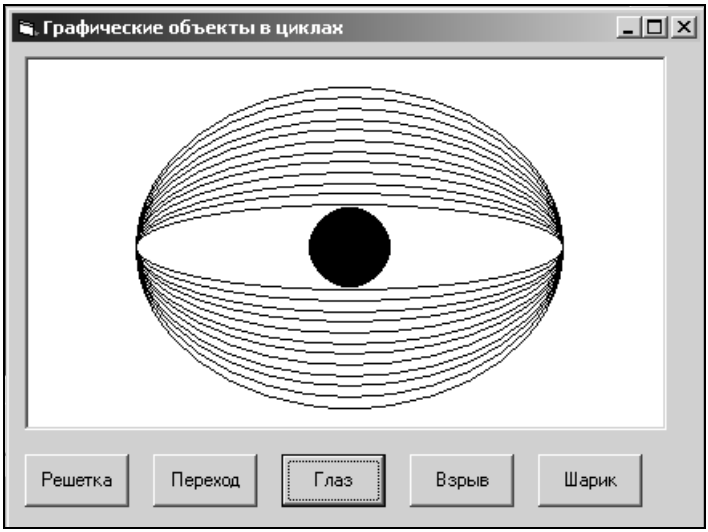


Рис. 2.30. Глаз

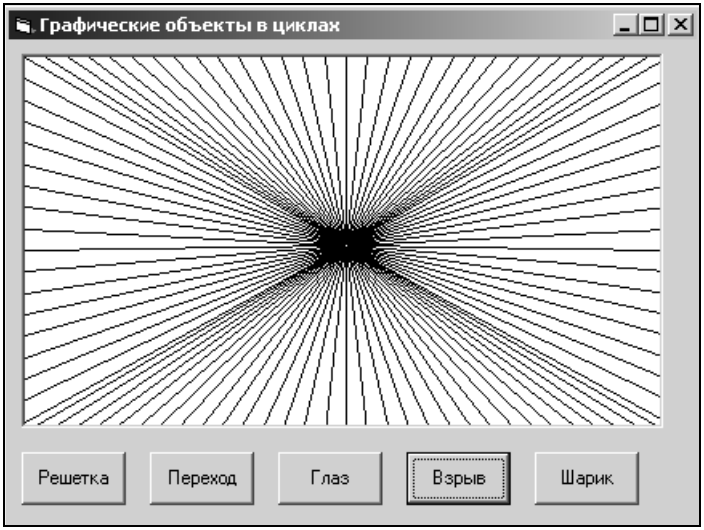


Рис. 2.31. Взрыв

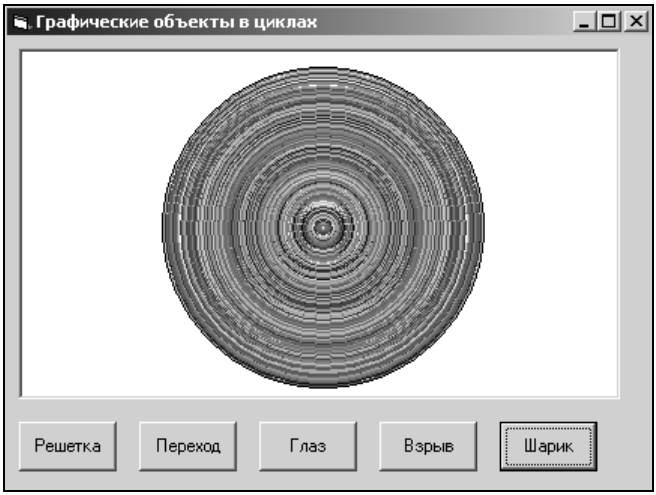


Рис. 2.32. Разноцветный шарик

110. Используя циклы с несколькими зависимыми переменными, изобразите на одной форме рупор (рис. 2.33), четыре рупора (рис. 2.34), лестницу (рис. 2.35), пирамиду — вид сверху (рис. 2.36) и пирамиду — вид сбоку (рис. 2.37).

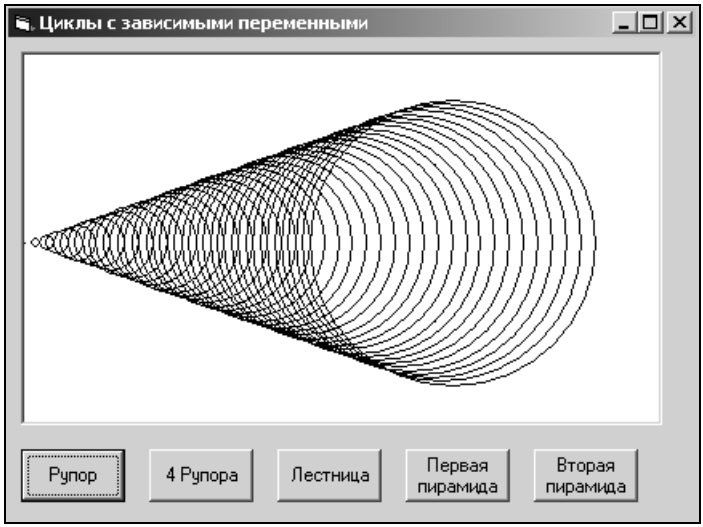


Рис. 2.33. Рупор

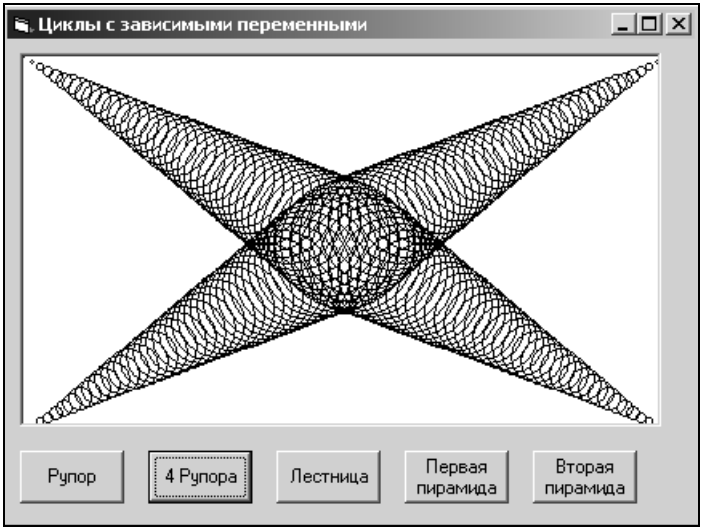


Рис. 2.34. Четыре рупора

Рис. 2.35. Лестница

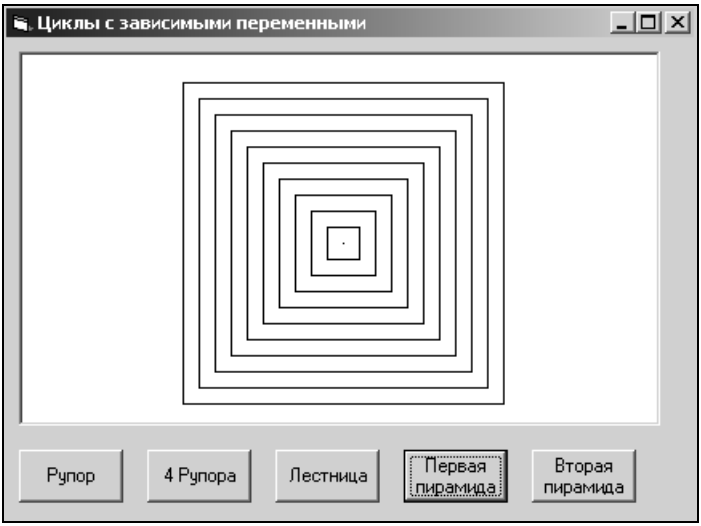


Рис. 2.36. Пирамида — вид сверху

Рис. 2.37. Пирамида — вид сбоку

111. Выведите на экран в строку все числа первой сотни, оканчивающиеся на пять.

112. Определите значение переменной `F` после выполнения следующих операторов:

```
F=1: N=1
FOR I=2 TO N
F=F+1/I
NEXT
```

113. Напишите программу, запрашивающую возраст пользователя, а затем печатающую текст: "Да ты крут!" — по числу прожитых лет. Обратите внимание, что здесь в теле цикла не будет использоваться параметр. Такое тоже возможно.

114. С клавиатуры запрашивается любая цифра от 2 до 9, а затем компьютер печатает таблицу умножения на эту цифру.

115. Напишите программу, выводящую на экран степени числа 2 от 2 до 10 включительно.

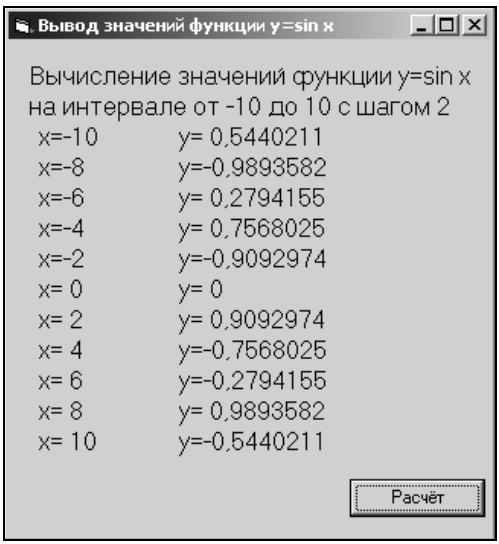


Рис. 2.38. Вывод значений функции y = sin x

116. Вычислить значения следующих функций на заданных интервалах, с указанным шагом изменения аргумента и выводом на форму в табличном виде (рис. 2.38):

а) $y = \sin x$, $x \in [-10,10]$, с шагом 2;

б) $y = \sqrt{x}$, $x \in [200,100]$, с шагом -10;

в) $y = x\dfrac{x}{4-x}$, $x \in [3,10]$, с шагом 0,5;

г) $y = x^3 - 16x$, $x \in [-8,8]$, с шагом 0,8.

117. Вычислить значение $n!$.

118. Определить, существуют ли такие четыре последовательных натуральных числа, сумма квадратов которых равна сумме квадратов трех следующих натуральных чисел.

119. Написать программу вычисления значения S при вводимых с клавиатуры x и n:

$$S = x + 2x^2 + 3x^3 + 4x^4 + ... + nx^n .$$

120. Вычислить наибольший общий делитель натуральных чисел A и B.

121. Вычислить $P = \left(1 - \dfrac{1}{2}\right)\left(1 - \dfrac{1}{3}\right)...\left(1 - \dfrac{1}{n}\right)$, $n > 2$.

122. Подсчитать k — количество цифр в десятичной записи целого неотрицательного числа N.

123. Логической переменной `T` присвоить значение true или false в зависимости от того, является натуральное число k степенью 3 или нет.

124. Вычислить S=1!+2!+3!+...+n! (n > 1).

125. Числа Фибоначчи f_n определяются формулами $f_o = f_1 = 1$; $f_n = f_{n-1} + f_{n-2}$ при n = 2, 3, Написать программу, которая по введенному натуральному числу n находит n-ое число Фибоначчи.

126. Найти первое число Фибоначчи, большее m (m > 1).

127. Вычислить s — сумму всех чисел Фибоначчи, которые не превосходят 1001.

128. Логической переменной `K` присвоить значение true, если целое N (N > 1) — простое число, и значение false в противном случае.

129. Дано целое n > 2. Напечатать все простые числа из диапазона [2, n].
130. Найти сумму цифр заданного натурального числа.
131. Определить число, получаемое выписыванием в обратном порядке цифр заданного натурального числа.
132. Определить, является ли заданное натуральное число палиндромом, т. е. таким, десятичная запись которого читается одинаково слева направо и справа налево.
133. В каких двузначных числах удвоенная сумма цифр равна их произведению? (Для проверки — 36, 44, 63.)
134. Найти двузначное число, равное утроенному произведению его цифр. (Для проверки — 15, 24.)
135. Найти двузначное число, обладающее тем свойством, что куб суммы его цифр равен квадрату самого числа. (Для проверки — 27.)
136. Найти все трехзначные числа, представимые в виде сумм факториалов своих цифр. (Для проверки — 145.)
137. Найти все двузначные числа, сумма квадратов цифр которых делится на 17. (Для проверки — 14, 28, 29, 35, 41, 53, 67, 76, 82, 92.)
138. Найти все трехзначные числа, которые можно представить разностью между квадратом числа, образованного первыми двумя цифрами и квадратом третьей цифры. (Для проверки — 100, 147.)
139. Найти все трехзначные числа, средняя цифра которых равна сумме двух крайних.
140. Найти все трехзначные числа, сумма цифр которых равна данному целому числу.
141. Вывести все трехзначные числа, в десятичной записи которых нет одинаковых цифр. Операции деления не использовать!
142. Найти все делители числа 1234.
143. Найти все двузначные числа, сумма цифр которых не меняется при умножении числа на 2, 3, 4, 5, 6, 7, 8, 9.

144. Даны целое число a и натуральное число n. Вычислить $P = a(a+1)...(a+n-1)$.

145. Найти первую степень числа 3, превышающую данное целое число K.

146. Проверить, содержит ли квадрат данного натурального числа n цифру 3 в своей записи.

147. Привести дробь вида a/b (b не равно 0) к несократимому виду.

148. Найти среднее арифметическое последовательности целых чисел произвольной длины.

149. Сколько существует натуральных чисел n, меньших 1000, для которых $2^n - n$ делится на 7.

150. Дана последовательность из N целых чисел. Напишите программу, которая определяет, какое число встретится раньше, положительное или отрицательное.

151. Дано натуральное число n. Вычислить произведение первых n сомножителей и распечатать:

$$P = \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{5}{6} \cdots$$

152. Вычислить сумму и распечатать для данного натурального n:

$$S = \sum_{1}^{n} \frac{(-1)^{i+1}}{i(i+1)}.$$

153. Вычислить сумму и распечатать для данного натурального n:

$$S = \sum_{1}^{n} \frac{(-1)^{i+1}}{i! i^2}.$$

154. Вычислить сумму и распечатать для данного натурального n:

$$S = \sum_{1}^{n} \frac{(-1)^{i+1}}{i(i+1)(i+2)}.$$

155. Используя вложенные циклы, изобразить шахматную доску (рис. 2.39) и лоскутный ковер (рис. 2.40).

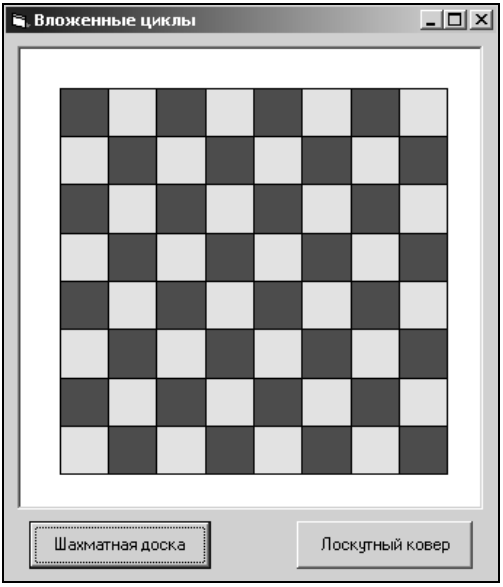


Рис. 2.39. Шахматная доска

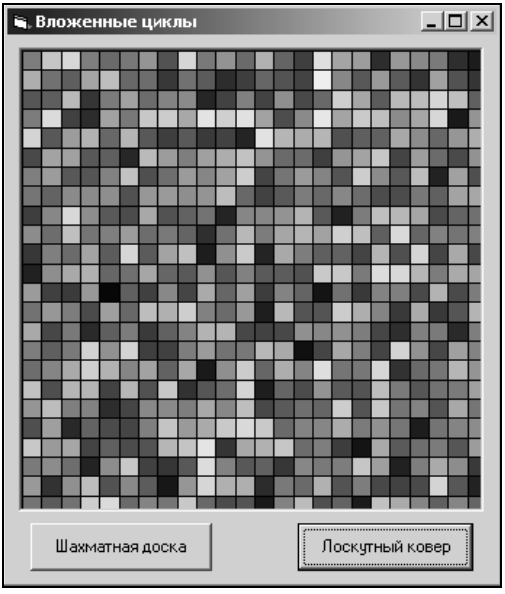


Рис. 2.40. Лоскутный ковер

На шахматной доске хорошо бы потом еще пронумеровать клетки от 1 до 64, а в лоскутном ковре использовать случайные цвета из палитры RGB.

156. Напишите программу, выводящую на экран таблицу умножения от 2 до 10 в следующем виде (рис. 2.41).

	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

Рис. 2.41. Таблица умножения

157. Вычисление числа π при помощи метода Монте-Карло. Это задание носит прикладной характер и позволяет опытным путем вычислить число. Да, безусловно, практически все из вас знают это число с точностью по крайней мере до двух знаков. Но предлагаемый метод очень хорош. Называется он методом Монте-Карло. Монте-Карло — европейская столица игорного бизнеса, а значит, там владычествует Его Величество Случай. Вот мы и попробуем поставить его себе на службу.

Сначала забудьте, чему равно π, и послушайте теорию вопроса. Представьте себе окружность радиусом $R=1$, вписанную в квадрат. Из этого следует, что сторона квадрата равна $2R$, а его площадь $SK=(2R)^2=4R^2$. Площадь круга $SO=\pi R^2$. Далее берем и равномерно посыпаем квадрат песком. Затем нанимаем бригаду рабочих, которые считают, сколько песчинок всего ($N1$) и сколько из них попало в круг ($N2$). Потом составляется простая пропорция — площадь квадрата так относится к площади круга, как общее количество песчинок к количеству песчинок, попавших в круг.

$$\frac{SK}{SO}=\frac{N1}{N2}\Rightarrow\frac{4\times R^2}{\pi\times R^2}=\frac{N1}{N2}\Rightarrow\pi=\frac{4\times N2}{N1}.$$

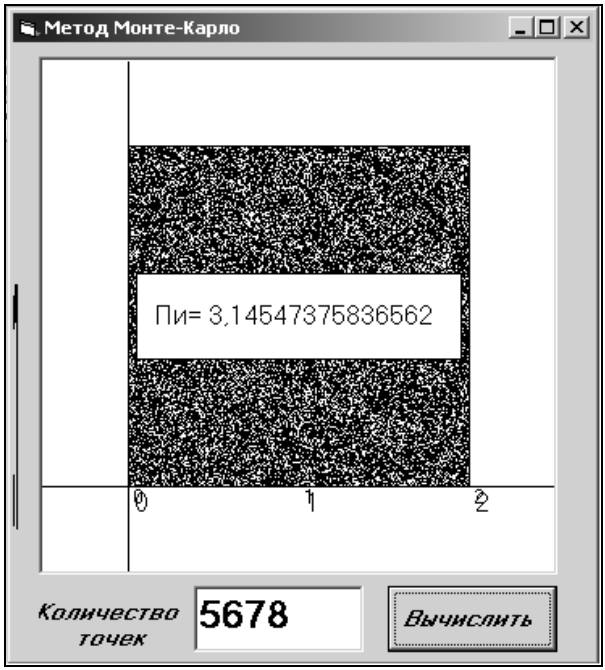


Рис. 2.42. Метод Монте-Карло и число

Отсюда сенсационный вывод: радиус окружности не имеет никакого значения, она должна быть лишь вписана в квадрат (рис. 2.42).

Но где ж мы найдем песок, а главное тех, кто все это будет считать? Поэтому поставим компьютерный эксперимент. Нарисуем квадрат и впишем в него окружность. Координаты опорных точек (если сами рисовали) знаем. Уравнение окружности $X^2 + Y^2 = R^2$ тоже знаем. Задаем цикл до 1000, в котором случайным образом определяем координаты "песчинок" так, чтобы они лежали внутри квадрата. Тут же проверяем условие, а не попала ли "песчинка" в круг (используя уравнение окружности), и если попала, подсчитываем их количество. Кроме того, рисуем их на экране разными цветами (попавшие и не попавшие). По окончании цикла подсчитываем и выводим на экран число π. Понятно,

что чем больше количество "песчинок", тем более точным будет результат. Для того чтобы знать, когда закончится эксперимент, рекомендуется выводить на экран счетчик "песчинок" (как мы делали с хронометром). Но все-таки, экспериментировать с миллионом "песчинок" не надо — замучаетесь ждать сами, да и компьютер, хотя и железный, но все же живой.

158. Вычислить число π по формуле Грегори.

В 1671 году Джеймс Грегори установил, что:

$$\theta = tg\theta - \frac{1}{3}tg^3\theta + \frac{1}{5}tg^5\theta - \frac{1}{7}tg^7\theta + \frac{1}{9}tg^9\theta \pm ...$$

Этот результат позволил Лейбницу получить очень простое выражение для *PI*, а именно:

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} \pm ...$$

или, после умножения на 4:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} \pm ...$$

Просуммируйте этот ряд, и вы получите число *PI*.

159. Вычислить число π по формуле Гаусса:

$$\frac{\pi}{4} = 12\arctan\frac{1}{18} + 8\arctan\frac{1}{57} - 5\arctan\frac{1}{239}.$$

Расчет арктангенса происходит по формуле Тейлора, а именно:

$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - ...$$

2.5.9. Анимация

Сначала пример движения геометрического объекта. "Мечущаяся точка". Точка движется за счет изменения ее координат и отражается от сторон экрана под углом 45° (рис. 2.43).

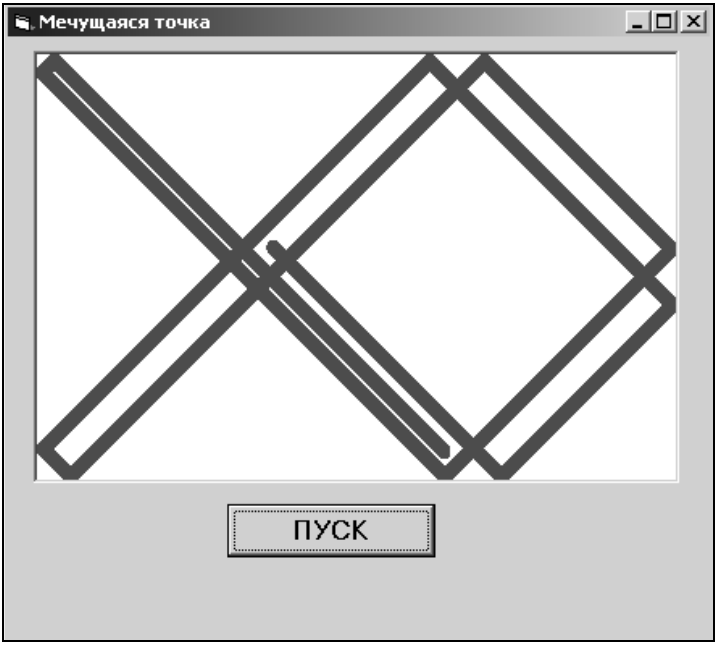


Рис. 2.43. Мечущаяся точка

Программный код:

```
Option Explicit
'x,y - координаты точки
'dx, dy - приращения координат
'n - счетчик шагов
'i - счетчик цикла замедления
Dim x AS Integer, y As Integer, dx As Integer
Dim dy As Integer, n As Integer,   i As Integer
Private Sub Command1_Click()
x = 2183: y = 1777       'начальные координаты
Picture1.DrawWidth = 8   'толщина точки
n = 1
dx = 1: dy = 1
While n < 25000
'Проверка условий выхода за пределы PictureBox
```

```
If x = 50 Or x = 5900 Then dx = -dx
If y = 50 Or y = 3900 Then dy = -dy
Picture1.PSet (x, y), vbRed
'Цикл замедления - счет про себя до 5000
For i = 1 To 5000: Next
'Стирание точки - если убрать апостроф, то точка
'не будет оставлять след
'Picture1.PSet (x, y), vbWhite
'Изменение координат
x = x + dx: y = y + dy
n = n + 1
Wend
End Sub
```

Попробуйте теперь убрать апостроф со стирания точки и посмотрите на эту муху, мечущуюся по экрану ☺.

А сейчас выполните следующие задания.

160. Дорисуйте внутри PictureBox из предыдущего задания прямоугольник. Пусть наша "муха" и его определяет как препятствие (рис. 2.44).

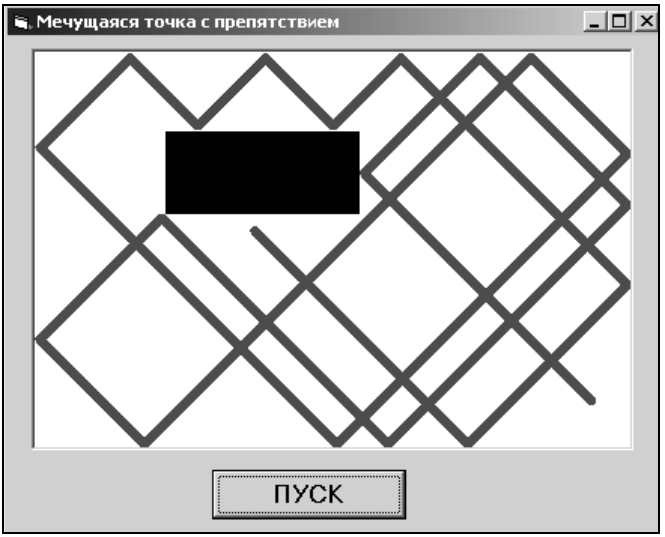


Рис. 2.44. С препятствием

161. Нарисуйте нечто вроде графина и заставьте "муху" летать внутри него, не нарушая целостности стенок. При возможном вылете из "графина" пусть она летает в пределах PictureBox (рис. 2.45).

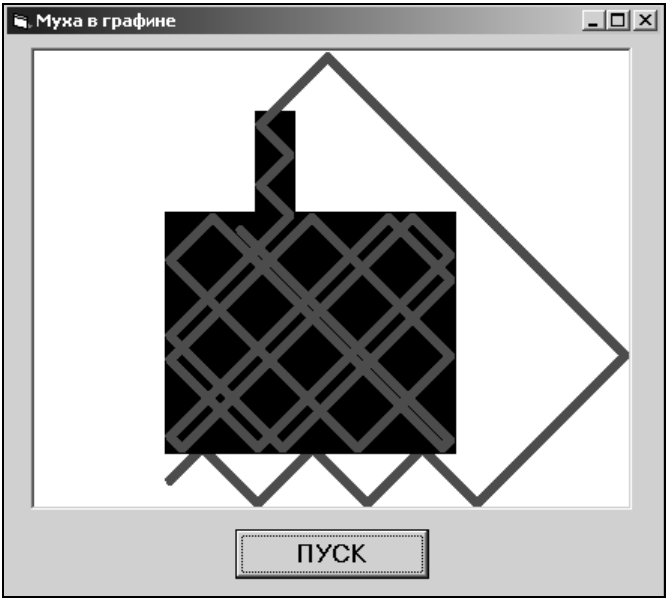


Рис. 2.45. "Муха" в "графине"

162. Теперь тоже стандартный пример геометрической анимации — движение по кругу (рис. 2.46). Измените программу так, чтобы Марс двигался в обратную сторону, против часовой стрелки. А еще попробуйте вращать две планеты, а в идеале — все девять Солнечной системы.

```
'Проект Вращение Марса вокруг Солнца
Option Explicit
Dim n, r As Integer
Dim i As Long
Private Sub Command1_Click()
'Количество оборотов
r = 10
```

Рис. 2.46. Вращение Марса вокруг Солнца

```
'Масштаб
Picture1.Scale (-10, 10)-(10, -10)
'Анимация
For n = 1 To 360 * r
'орбита Марса
Picture1.Circle (0, 0), 8, vbYellow
'Солнце
Picture1.FillStyle = 0
Picture1.FillColor = vbYellow
Picture1.Circle (0, 0), 2, vbYellow
'рисование Марса
Picture1.FillColor = vbRed
Picture1.Circle (8 * Sin(6.28 / 360 * n), _
8 * Cos(6.28 / 360 * n)), 1, vbRed
'Задержка стирания
For I = 1 To 100000:Next
'стирание Марса
```

```
Picture1.FillColor = vbBlue
Picture1.Circle (8 * Sin(6.28 / 360 * n), _
8 * Cos(6.28 / 360 * n)), 1, vbBlue
Next n
End Sub
```

163. Следующий не менее стандартный пример — стрелочные часы (рис. 2.47). Измените программу, расположив цифры в противоположном порядке и заставив стрелки крутиться в обратную сторону. Время, назад!

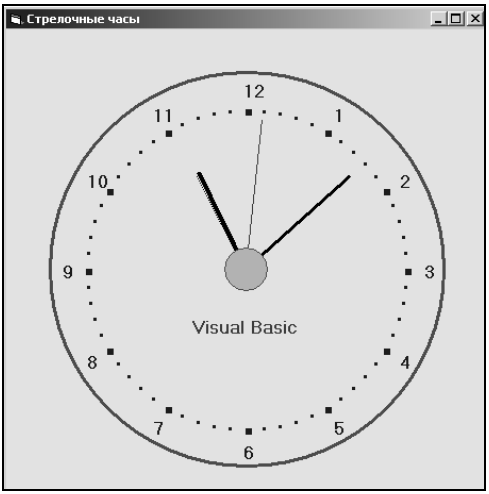


Рис. 2.47. Стрелочные часы

Перед написанием программного кода разместите на форме Timer.

```
Option Explicit
Const r = 150
Const grad = 0.0174532
Dim x0 As Integer, y0 As Integer, hr As Integer
Dim mn As Integer, sc As Integer
'Задание формы
Private Sub Form_Initialize()
Form1.Height =(Form1.Height- Form1.ScaleHeight) + _
```

```
(r + 5) * 2 * Screen.TwipsPerPixelY
Form1.Width =(Form1.Width- Form1.ScaleWidth) + _
(r + 5) * 2 * Screen.TwipsPerPixelX
x0 = r + 75: y0 = r + 75
hr = 90 - Hour(Time) * 30 - (Minute(Time) / 12) * 6
mn = 90 - Minute(Time) * 6
sc = 90 - Second(Time) * 6
Timer1.Interval = 1000
Timer1.Enabled = True
Form1.ScaleMode = 3
Form1.BackColor = vbYellow
End Sub
'Рисование линии из центра циферблата
Sub Strelka(x0, y0, a, s As Integer)
Dim x1, y1 As Integer
x1 = Round(x0 + s * Cos(a * grad))
y1 = Round(y0 - s * Sin(a * grad))
Line (x0, y0)-(x1, y1)
End Sub
'Рисование стрелок и их движение
Sub Strelki()
Form1.DrawWidth = 3
Form1.ForeColor = Form1.BackColor
Call Strelka(x0, y0, hr, r - 50)
Call Strelka(x0, y0, mn, r - 20)
Call Strelka(x0, y0, sc, r - 8)
hr = 90 - Hour(Time) * 30 - (Minute(Time) / 12) * 6
mn = 90 - Minute(Time) * 6
sc = 90 - Second(Time) * 6
Form1.DrawWidth = 4
Form1.ForeColor = RGB(0, 0, 0)
Call Strelka(x0, y0, hr, r - 50)
Form1.DrawWidth = 3
Call Strelka(x0, y0, mn, r - 20)
Form1.DrawWidth = 1
```

```
Form1.ForeColor = RGB(200, 0, 0)
Call Strelka(x0, y0, sc, r - 8)
PSet (175, 270), vbYellow
Print "Visual Basic"
End Sub
'Рисование циферблата
Private Sub Form_Paint()
Dim x1, y1, a, h As Integer
a = 0
h = 3
While a < 360
x1 = Round(x0 + r * Cos(a * grad))
y1 = Round(y0 - r * Sin(a * grad))
If a Mod 30 = 0 Then
Line (x1, y1)-(x1 + 5, y1 + 5), vbBlue, BF
CurrentX=x0+Round((r+20)*Cos(2*a * 3.14 / 360)) - 7
CurrentY=y0-Round((r+20)*Sin(2*a * 3.14 / 360)) - 7
ForeColor = vbBlue
FontSize = 12
Print h
h = h - 1
If h = 0 Then h = 12
Else
Line (x1, y1)-(x1 + 2, y1 + 2), vbBlue, BF
End If
a = a + 6
Wend
'Вызов процедур движения стрелок
Call Strelki
End Sub
Private Sub Timer1_Timer()
Call Strelki
End Sub
```

164. Вот еще один хороший пример анимации (рис. 2.48). Измените программу, укрупнив снег. Введите другой текст, увеличьте форму.

Рис. 2.48. Падающий снег

Подключите модуль Snow_module (меню **Project | Add Module**) и пропишите там следующий код:

```
Option Explicit
Type xParticle
    X As Integer
    Y As Integer
    oldX As Integer
    oldY As Integer
    iStopped As Integer
End Type
Global Const MAXP = 400
Global Const PSIZE = 1
Global Snow(0 To MAXP) As xParticle
```

А теперь основной программный код:

```
Dim bRUN As Boolean
```

```
Dim fMouseDown_X As Single
Dim fMouseDown_Y As Single
Dim bMOUSE_DOWN As Boolean
```

```
Private Sub Form_Load()

    Randomize

    Me.ScaleMode = vbPixels
    Me.DrawWidth = PSIZE
```

```
    Me.BackColor = vbBlack

    Dim i As Integer
    For i = 0 To MAXP
        Snow(i).X = CInt(Int(Me.ScaleWidth * Rnd))
        Snow(i).Y = CInt(Int(Me.ScaleHeight * Rnd))
    Next i

    bRUN = True
    Timer1.Enabled = True

    'Вывод текста в заданном формате
    Const sTEXT = "Снег кружится и летает"
    Me.ForeColor = vbRed
    Me.CurrentX = Me.ScaleWidth / 2 - _
TextWidth(sTEXT) / 2
    Me.CurrentY = Me.ScaleHeight / 2 - _
TextHeight(sTEXT) / 2
    Me.Print sTEXT

    Me.ForeColor = vbWhite
End Sub
Sub DrawSnow()
    Dim i As Integer

    Dim newX As Integer
    Dim newY As Integer
    Timer1.Enabled = False
    Do While bRUN

    For i = 0 To MAXP
     Me.PSet (Snow(i).oldX, Snow(i).oldY), vbBlack
     Me.PSet (Snow(i).X, Snow(i).Y)
    Next i

    For i = 0 To MAXP
```

```
        Snow(i).oldX = Snow(i).X
        Snow(i).oldY = Snow(i).Y
        newX = Snow(i).X + Int(2 * Rnd)
        newX = newX - Int(2 * Rnd)

        'Не разрешаем нашему снегу улетать прочь
        If newX < 0 Then newX = 0
        If newX >= Me.ScaleWidth Then newX = _
Me.ScaleWidth - 1

        newY = Snow(i).Y + 1

        If Me.Point(newX, newY) = vbBlack Then
             Snow(i).Y = newY
             Snow(i).X = newX
        Else
        If Snow(i).iStopped = 10 Then
       'Движение в соответствии с правилами снега ☺
  If Me.Point(Snow(i).X + 1, Snow(i).Y + 1) = _ vbBlack Then
Snow(i).X = Snow(i).X + 1
Snow(i).Y = Snow(i).Y + 1
Snow(i).iStopped = 0
  ElseIf Me.Point(Snow(i).X - 1, Snow(i).Y + 1) = _ vbBlack Then
Snow(i).X = Snow(i).X - 1
Snow(i).Y = Snow(i).Y + 1
Snow(i).iStopped = 0
  Else
   newParticle (i)
End If

  Else
      Snow(i).iStopped = Snow(i).iStopped + 1
  End If
End If
```

```
 If (Snow(i).Y) >= Me.ScaleHeight Then
   newParticle (i)
```

```
 End If
Next i
    DoEvents
Loop

End Sub

Sub newParticle(i As Integer)
    Snow(i).X = CInt(Int(Me.ScaleWidth * Rnd))
    Snow(i).Y = 0
    Snow(i).oldX = 0
    Snow(i).oldY = 0
    Snow(i).iStopped = 0
End Sub

Private Sub Form_MouseDown(Button As Integer,
Shift _ As Integer, X As Single, Y As Single)
    Me.PSet (X, Y)
    bMOUSE_DOWN = True
    fMouseDown_X = X
    fMouseDown_Y = Y
End Sub

Private Sub Form_MouseMove(Button As Integer,
Shift _ As Integer, X As Single, Y As Single)
    If bMOUSE_DOWN Then
        Dim oldDW As Long
        Dim oldFC As Long
        oldDW = Me.DrawWidth
        oldFC = Me.ForeColor
        Me.DrawWidth = 3
        Me.ForeColor = vbRed
        Me.Line (fMouseDown_X, fMouseDown_Y)-(X, Y)
        fMouseDown_X = X
        fMouseDown_Y = Y
        Me.DrawWidth = oldDW
```

```
            Me.ForeColor = oldFC
        End If
    End Sub

    Private Sub Form_MouseUp(Button As Integer, Shift _
    As Integer, X As Single, Y As Single)
        bMOUSE_DOWN = False
    End Sub

    Private Sub Form_QueryUnload(Cancel As Integer, _
    UnloadMode As Integer)
        bRUN = False
    End Sub

    Private Sub Timer1_Timer()
        DrawSnow
    End Sub
```

165. Анимация с использованием метода `Move` (рис. 2.49). По нажатию кнопки "По газам!" машина передвигается вперед, а по нажатию кнопки "Задний ход" — возвращается на место. Измените программу, заставив машину не мгновенно перемещаться на новое место, а плавно переезжать, и так же плавно возвращаться на место.

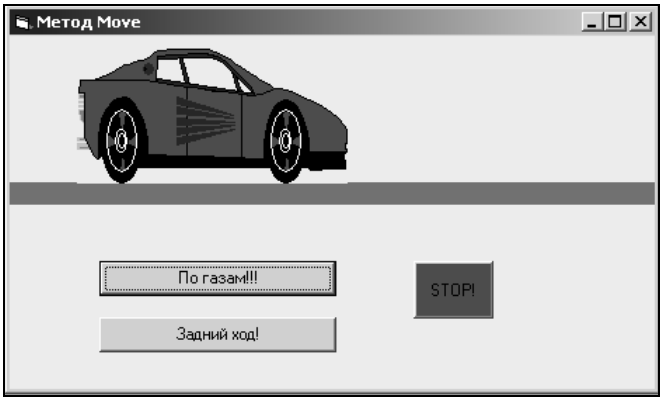


Рис. 2.49. Полный вперед!

Программный код:

```
Dim x, y, z As Integer

Private Sub Command1_Click()
End
End Sub

Private Sub Form_Load()
x = Sp1.Left
y = Sp1.Top
z = 2500
End Sub

Private Sub Cmd1_Click()
Sp1.Move x + z, y
Cmd2.SetFocus
End Sub

Private Sub Cmd2_Click()
Sp1.Move x, y
Cmd1.SetFocus
End Sub
```

Замечание

В этом задании использовался фокус для командных кнопок. В режиме выполнения на форме один из объектов является текущим или, по-другому, имеет фокус. Элемент, у которого есть фокус, доступен для ввода с клавиатуры, на кнопке появляется штриховая рамка, ее можно нажать, используя клавишу <Enter>. Если объект получает фокус, происходит событие `GotFocus`, если теряет — `LostFocus`. В программе для передачи фокуса используется метод `SetFocus`.

166. Попробуйте себя в роли бога Перуна. Пусть по щелчку на форме начинает лить косой дождь, по повторному щелчку дождь прекращается (рис. 2.50). Измените программу, чтобы дождь шел теперь слева направо. Еще попробуйте при-

бавлять к координатам в операторе `Line` не постоянные числа, а случайные. Получился дождь с ураганным ветром?

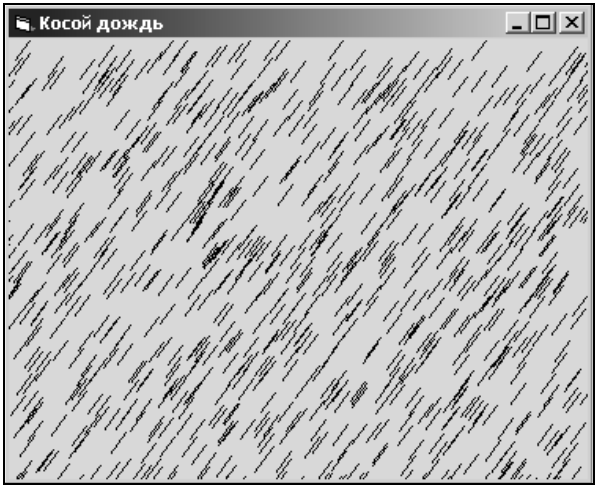


Рис. 2.50. Косой дождь

Программный код:

```
Dim B As Boolean
Private Sub Form_Load()
B = False
End Sub

Private Sub Form_Click()
Dim c As Long
Dim i As Long
Randomize
B = Not B
c = 0
Do While B And c < 5000
x = Rnd * 5640 'ScaleWidth
y = Rnd * 4690 'ScaleHeigth
Line (x, y)-(x - 150, y + 200)
For i = 1 To 30000: Next
```

```
'Оператор DoEvents позволяет узнать о повторном событии 'Click на форме
DoEvents
c = c + 1
Loop
End Sub
```

167. Увеличение и уменьшение размеров объектов в процессе выполнения программы.

Visual Basic предоставляет возможность управлять свойствами `Height` (Высота) и `Width` (Ширина) для большинства объектов формы. Пример на рис. 2.51.

Рис. 2.51. Увеличение размеров картинки

Поместите на форму Image (Изображение), а в нем разместите Picture (Картинку), предварительно установив свойство `Scretch` — True. Дважды щелкнув по Image на форме, запишем следующий программный код:

```
Private Sub Image1_Click()
Image1.Height = Image1.Height + 500
Image1.Width = Image1.Width + 500
End Sub
```

Дополните форму еще одной командной кнопкой, позволяющей уменьшать изображение.

168. Можно причислить к анимации и следующий пример, в котором использование инструментов **HScrollBar** и **VScrollBar** позволяют рисовать и стирать дугу эллипса с заданным коэффициентом сжатия (рис. 2.52).

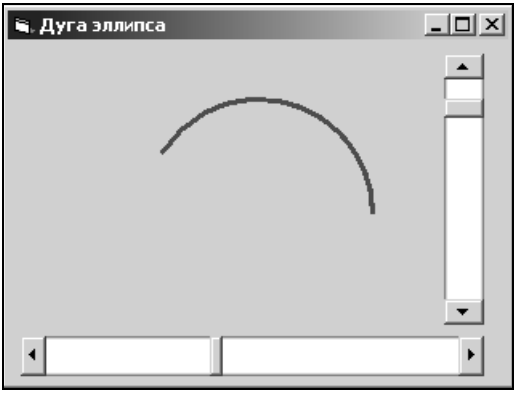


Рис. 2.52. Дуга эллипса

Командный код:

```
Const pi = 3.14159
Dim aa, aac As Single
Dim x0, y0, r, k As Single
Private Sub Form_Load()
x0 = ScaleWidth / 2
y0 = ScaleHeight / 2
r = ScaleHeight / 3
End Sub
'Задание вертикальным ползунком коэффициента сжатия
Private Sub VSB1_Change()
k = vsb1.Value
End Sub
'Процедура рисования-стирания дуги
Private Sub HSB1_Change()
Dim aa As Single
DrawWidth = 3
```

```
'Определение угла по горизонтальному ползунку
aa = HSB1.Value * pi / 180
Circle (x0, y0), r, vbRed, 0, aa, k
If aac > aa Then
Circle (x0, y0), r, BackColor, aa, aac, k
End If
aac = aa
End Sub
```

Попробуйте проделать подобную анимацию с рисованием простых разномасштабных фигур, например, квадратов, а затем чего-нибудь более сложного, например, домиков ☺.

169. Еще простой пример анимации с изменением координат движущегося объекта. Условное название "Солнышко" (рис. 2.53). Оно движется по форме слева направо.

Рис. 2.53. Солнышко

Командный код:

```
Private Sub Command1_Click()
'Установка цвета фона
BackColor = vbCyan
'Установка типа и цвета заливки для солнышка
FillStyle = 0
```

```
FillColor = vbYellow
'Цикл имитации движения солнца по горизонтали
'рисование и тут же рисование цветом фона
For x = 0 To ScaleWidth
Circle (x, 700), 200, vbYellow
Circle (x, 700), 200, vbCyan
'Вложенный пустой цикл для организации замедления
'(счет в уме ☺)
For i = 1 To 20000: Next
Next
End Sub
```

Ваша задача заключается в том, чтобы сначала заставить солнышко двигаться по перевернутой параболе (имитируя настоящий восход и заход — вспоминаем построение графиков!), а затем сделать так, чтобы после захода солнца наступала ночь и справа налево по той же траектории на уже черном фоне двигалась белая луна.

170. А теперь солнышко уже двигается не просто по однородному фону, заданному заранее свойством `BackColor`, а по неоднородному небу загруженной картинки (рис. 2.54).

Рис. 2.54. Движение солнышка по фону загруженной картинки

Для поставленной задачи нам пригодится оператор `Point`, возвращающий цвет в палитре RGB точки с указанными координатами. Синтаксис его таков:

```
object.point(x,y)
```

где `object` — объект, в котором используется оператор, а `x` и `y` — координаты точки в объекте.

Тогда наш программный код для солнышка примет следующий вид (на форме предварительно надо разместить PictureBox):

```
Private Sub Command1_Click()
'Загрузка в PictureBox картинки с пейзажем
Picture1.Picture = LoadPicture("C:\mtn.jpg")
'Задание начальных параметров заливки солнышка
Picture1.FillStyle = 0
Picture1.FillColor = vbYellow
For x = 0 To Picture1.Width + 500
Picture1.Circle (x, 500), 200, vbYellow
'Ниже в качестве цвета "стирания" солнышка в ходе
'движения берется цвет над ним с использованием
'оператора Point
Picture1.Circle (x, 500), 200, Picture1.Point _
(x, 290)
For i = 1 To 20000: Next
Next
End Sub
```

171. Анимация с заменой фаз изображения.

Довольно часто, как и в мультипликации, движение на экране достигается частой сменой фаз изображения. (Такие еще детские блокнотики есть с нарисованными на разных страницах картинками — когда быстро листаешь, кажется, что нарисованное движется ☺). Попробуем это сделать.

Сначала, безусловно, нужно прорисовать (или взять откуда-нибудь) картинки, изображающие что-либо в движении. Хорошо бы они были одинакового размера. Вот, например, такие, как на рис. 2.55.

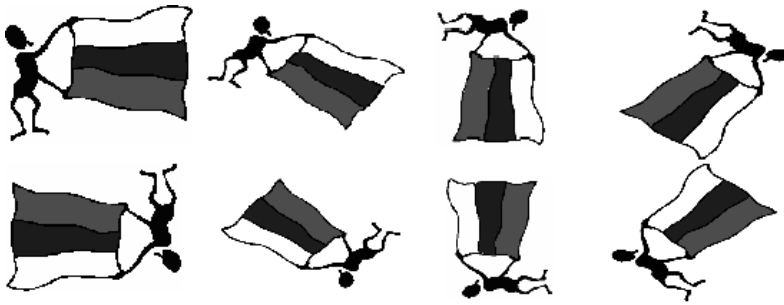


Рис. 2.55. Человечки с флагом

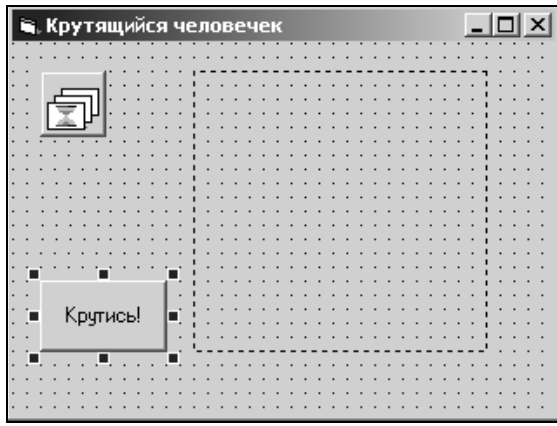


Рис. 2.56. Форма для крутящихся человечков

Теперь подготовим форму. Примерно как на рис. 2.56.

Пунктиром на форме показан объект Image, а вот объект с тремя кадрами или конвертиками (как кому нравится) — это ImageList. Обычно его на панели **General** нет. Он выполняет роль контейнера для заготовленных заранее картинок. Чтобы он появился, необходимо выбрать команду **Components** из меню **Project**, найти в списке управляющих элементов Microsoft Windows Common Controls 6.0, отметить его и нажать кнопку **Применить**. Панель **General** существенно расширится и станет выглядеть вот так (рис. 2.57).

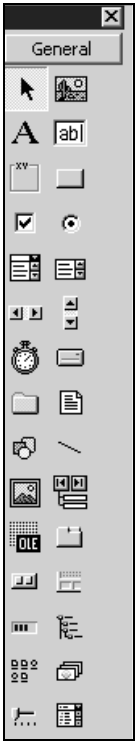


Рис. 2.57. Расширенная панель **General**

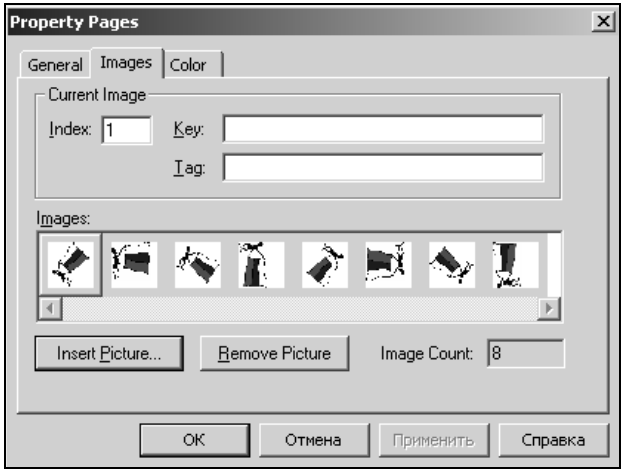


Рис. 2.58. Вкладка **Images**

Помещаем теперь нужный элемент на нашу форму (он не будет виден в режиме выполнения). Выберите в **Properties** свойство `Custom` и войдите в диалог **Property Pages**. На вкладке **Images** (рис. 2.58) нажмите кнопку **Insert Picture**.

Загрузите ваши заранее заготовленные картинки. Напишите программный код, запустите получившийся проект и творите, выдумывайте, пробуйте!

```
Private Sub Command1_Click()
i = 1
For i = 1 To 8
Image1.Picture = ImageList1.ListImages(i).Picture
```

```
For y = 1 To 9000000: Next
Image1.Refresh
Next
End Sub
```

172. На основе предыдущего задания легко делается анимация матросика с семафорными флажками. Можно заставить его передавать какое-нибудь сообщение, например, "Visual Basic forever!". Для слов "Visual Basic" запись семафорной азбукой выглядит так (рис. 2.59).

Рис. 2.59. "Visual Basic" семафорной азбукой

Семафорную азбуку можно найти, например, на следующем сайте:

http://randewy.narod.ru

И пусть матросик передает свое сообщение!

2.6. Массивы

Мы подобрались к одной из самых сложных, на мой взгляд, тем в программировании для начинающих. Именно из-за массивов я остался на второй год в институте (потому что тогда в школах еще этим не занимались). Теперь, когда я объясняю эту тему своим ученикам, то стараюсь сделать это как можно более доходчиво, пусть не совсем научными терминами, но понятно, поскольку без представления, что такое массив, дорога в программирование будет закрыта.

Но и без умных определений не обойтись.

Итак, *массив* — это набор однотипных данных (чисел, символов, слов), имеющий имя и последовательную нумерацию его элементов. Например, список фамилий учеников вашего класса — массив, численные данные о среднесуточной температуре за месяц — массив, буквы русского алфавита — это тоже массив.

Как физически массив представляется в компьютере, я рассказывать не буду, а как формально — это надо себе представлять.

2.6.1. Описание массива

Если мы знаем, что в программе предстоит работать с большим объемом каких-то данных, то мы должны этот массив в программе объявить с помощью уже известного вам оператора `DIM`, после которого указывается имя массива, а потом в скобках следует так называемая размерность массива, т. е. указывается начальный и конечный индексы (номера) его элементов. После скобок мы должны объявить тип данных массива. Например, пусть в группе четыре человека. Массив — это фамилии учеников. Мы тогда должны записать так:

```
Dim Fam (1 To 4) As String
```

В этом случае компьютер в памяти отводит некую область из четырех ячеек, которую называет `Fam`. Кроме того, эти ячейки нумеруются натуральными числами, начиная с 1. Я всегда подобную процедуру, да и сам массив, сравниваю с улицей одноэтажных домов в деревне или маленьком городке. Построили на улице четыре дома, назвали улицу `Fam` (имя массива дается по тем же правилам, что и имя переменной), пронумеровали дома и заселили туда жильцов (рис. 2.60).

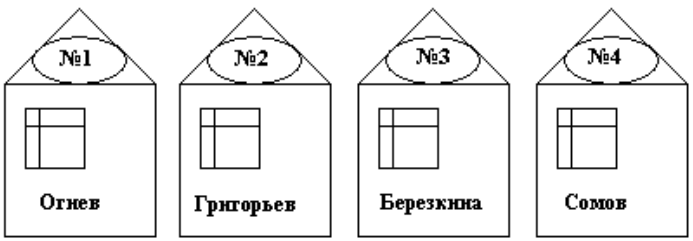


Рис. 2.60. Одномерный массив Fam

Из этого следует, что:

- у массива есть *имя*, которое дает ему программист;
- у массива есть *тип*, который определяется типом данных, его составляющих;
- у массива есть *размер*, т. е. количество составляющих его элементов;
- у массива есть *сквозная последовательная индексация* составляющих его элементов;
- у каждого элемента массива есть *значение* (в нашем случае это фамилия).

Замечание

Оператор `DIM` для каждого конкретного массива должен задаваться только один раз в программе до первого к нему обращения.

Продолжая аналогию с улицей одноэтажных домов, что надо сделать, чтобы обратиться к какому-либо конкретному жильцу? Знать его адрес!

Предположим, мы хотим потревожить господина Григорьева — указываем его адрес — `Fam(2)`, т. е. название улицы и дом. Зачастую начинающие программисты путают индекс элемента массива (его номер) и значение элемента массива, т. е. "кто-кто в тереме живет". Итак, еще раз: индекс или номер элемента массива — величина постоянная, а значение (как и жильцы в доме) с легкостью может меняться.

В начале мы рассмотрим одномерные массивы — такие, в которых адрес элемента массива определяется только одним индексом (номером "дома").

Замечание

На самом деле, нумерация ячеек ("домиков") в Visual Basic начинается с нуля, но с единицы нам привычнее и удобнее, поэтому нулевой "домик" мы пропускаем. Возможен более законный вариант — обязать Бейсик нумеровать "домики" с единицы оператором `Option Base 1`.

2.6.2. Заполнение одномерных массивов и вывод их значений на экранную форму

Первая задача, встающая перед программистом, заполнить массив "жильцами". Для этого в Бейсике существует несколько способов, которые мы и рассмотрим. Кроме того, для контроля правильности заполнения, лучше бы сразу выводить массив на экран, чтобы потом можно было проверить правильность решения поставленной задачи.

Для всех случаев мы рассмотрим один и тот же пример — заполнить массив *N* целыми числами, каждое из которых не более 100. Мы не берем конкретное значение *N*, чтобы вы понимали: программа не должна зависеть от исходных данных. Сейчас мы хотим создать массив из 10 чисел, а в следующий раз — из 1000. Программа останется той же.

Заполнение одномерного массива с клавиатуры

Рассмотрим следующий пример (рис. 2.61).

```
Dim Fam(1 To 4) As String * 20
Cls
For i = 1 To 4
```

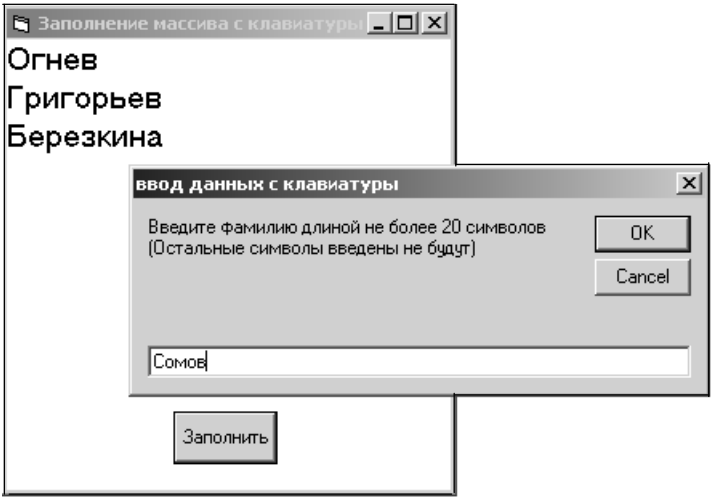


Рис. 2.61. Заполнение массива с клавиатуры

```
Fam(i) = InputBox("Введите фамилию длиной не _
более 20 символов" & _
"(Остальные символы введены не будут)", _
"ввод данных с клавиатуры")
Print Fam(i)
Next i
```

Программа требует некоторых пояснений. Сначала объявляем строковый массив из четырех элементов. Знак умножения на 20 ограничивает длину вводимых фамилий двадцатью символами (очень полезная возможность). Вторая команда традиционна — очистка экрана. Далее идет цикл, в котором от 1 до 4 программа последовательно запрашивает у пользователя ввод очередного элемента массива и записывает его значение по указанному адресу `Fam(i)`. Последний оператор цикла выводит значения массива на форму в столбик.

Заполнение массива случайными числами

Так как чаще мы решаем учебные задачи, то конкретные числовые или строковые значения элементов массива нас мало интересуют. Важно, чтобы программа работала правильно. Поэтому часто массив заполняется случайными значениями.

Рис. 2.62. Заполнение массива случайными числами

В следующем примере массив из десяти элементов заполняется случайными числами от 1 до 100, которые выводятся на форму в строку (рис. 2.62).

```
Dim M(1 To 10) As Integer
Cls
Randomize
```

```
For i = 1 To 10
M(i) = Fix(Rnd(1) * 100) + 1
Print M(i);
Next I
```

Заполнение массива при помощи стандартных функций

Мы вычисляем какую-либо функцию, например синус, и значения этой функции заносим в массив. Пусть дана функция $y = \sin x$, где x меняется от 1 до 10 с шагом 0,5. Здесь мы должны сначала решить для себя, а сколько будет значений вычислено. Мы это уже делали, но повторение — мать учения.

$$N = \frac{(Хнач. - Хкон.)}{шаг} + 1.$$

В нашем случае:

$$N = \frac{(10 - 1)}{0{,}5} + 1 = 19.$$

Тогда программа будет выглядеть так:

```
Cls
Dim M(1 To 19)
i = 1
For X = 1 To 10 Step 0.5
  M(i) = Sin(X)
  Print "X="; X, "Sin(X)="; M(i)
  i = i + 1
Next X
```

Здесь нам уже приходится вручную наращивать индекс `I`, т. к. параметром цикла в данном случае является `X`. Результаты выполнения программы можно видеть на рис. 2.63.

После заполнения массива настает пора его обрабатывать. Чаще всего приходится обрабатывать все элементы массива, поэтому действия выполняются *в цикле*.

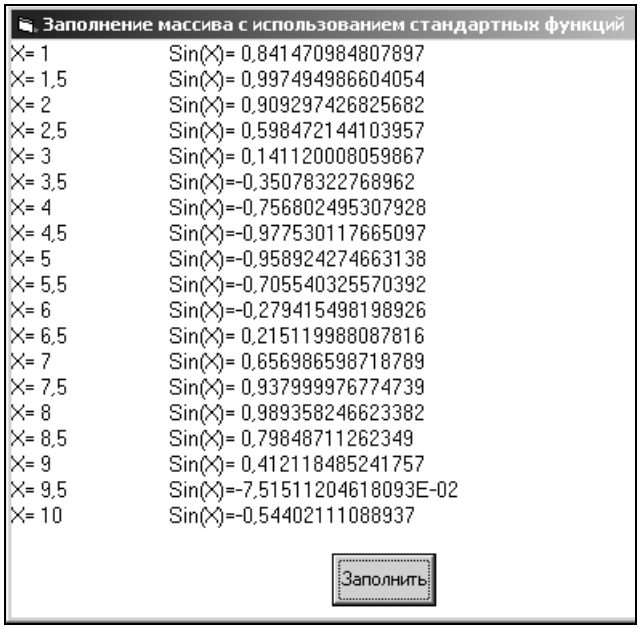


Рис. 2.63. Заполнение массива с использованием стандартных функций

Изменение значений элементов массива ведется с помощью оператора присваивания, например:

```
M (1) = 13
M (3) = 12
M (3) = SQR (M (1)+M (3))
Print M (3)
```

Вопрос. Какое значение будет выведено на форму? Чему равны значения `M (1)` и `M (3)`?

Прежде всего, попрактикуемся в заполнении массивов и выводе на экран не только численных значений элементов массива, но и графической их интерпретации.

173. Заполните массив десятью случайными целыми числами, каждое их которых лежит в пределах от 50 до 200, и выведите на экран их численные значения, а также графическое представление в виде вертикальных закрашенных прямоугольников шириной 30 пикселов и высотой, соответствующей их

значению. Нижние стороны прямоугольников лежат на линии с координатой $Y = 300$, левой стороне первого прямоугольника соответствует координата $X = 100$ (рис. 2.64).

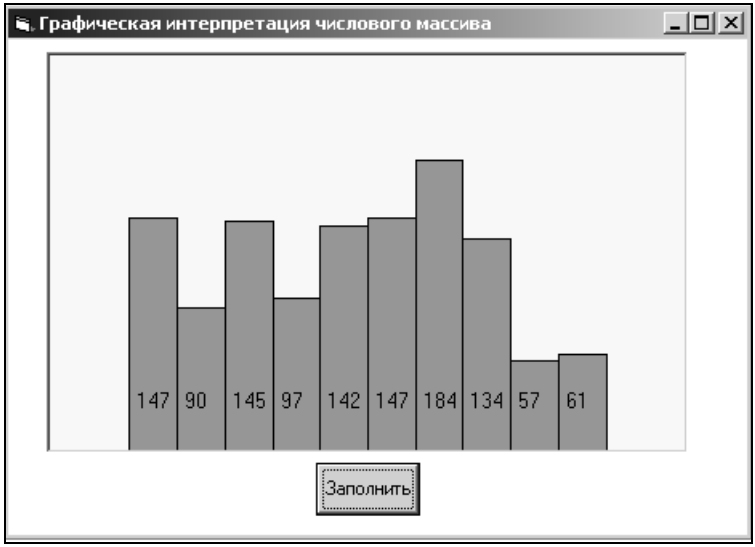


Рис. 2.64. Графическая интерпретация массива

174. Заполните массив десятью случайными целыми числами, каждое их которых лежит в пределах от 5 до 35, и выведите на экран их численные значения, а также графическое представление в виде закрашенных соприкасающихся кругов, радиусы которых равны значениям элементов массива (рис. 2.65).

175. Напишите программу вычисления среднего арифметического следующих вводимых с клавиатуры десяти чисел: 31, 19, 52, 65, 6, 8, 13, 16, 97, 33.

176. Напишите программу вычисления выражения:

$$\frac{(x_1 - S)^2 + (x_2 - S)^2 + \ldots + (x_N - S)^2}{N},$$

где x_i — числа из предыдущего задания; N — их количество; S — среднее арифметическое элементов массива x.

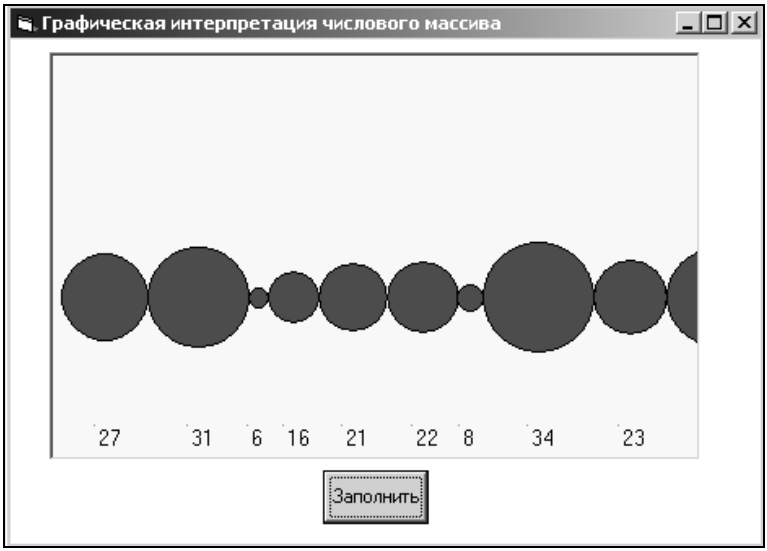


Рис. 2.65. А теперь числа в виде кружочков

177. Найти сумму 1-го, 4-го, 9-го, 16-го и так далее, включая 81-й элемент массива, состоящего из 100 целых случайных чисел, каждое из которых лежит в пределах от 2 до 22.

178. Замените в массиве из 10 случайных целых чисел, каждое из которых лежит в пределах от 1 до 10, все четные элементы нулями и выведите полученный массив на форму.

179. Массив состоит из 60 случайных двузначных целых чисел. Выведите их на экран в обратном порядке по 6 чисел в строке (рис. 2.66).

180. В массиве содержатся 10 букв — С, Ф, О, И, К, Л, О, И, Л, Н. Выведите на экран слово, образованное буквами с четными индексами, и слово, образованное буквами с нечетными индексами.

181. Массив состоит из 20 целых положительных и отрицательных чисел, модуль каждого из которых в пределах от 2 до 12. Выведите на экран сначала отрицательные, а затем положительные числа. Определите, модуль суммы каких чисел больше — положительных или отрицательных.

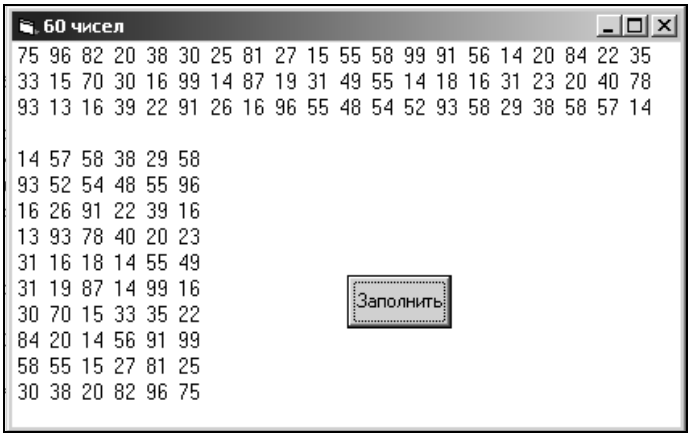


Рис. 2.66. 60 чисел: было и стало

182. Найдите максимальный и минимальный элементы массива из 10 случайных целых двузначных чисел и разность между ними. Представьте графическую столбиковую интерпретацию этого массива, выделив максимальный элемент красным, а минимальный — зеленым цветом. Остальные прямоугольники должны быть желтого цвета.

Теперь задания с несколькими массивами.

183. Даны два массива, заполненные каждый десятью случайными целыми числами, каждое из которых от 1 до 9 включительно. Сложите массивы поэлементно, результаты запишите в третий массив. На экран вывести все три массива.

184. Даны три массива с одинаковым количеством элементов 5, в которых содержатся стороны треугольников A_i, B_i и C_i. Определите периметр P_i и площадь S_i каждого треугольника по формуле Герона.

185. Найдите скалярное произведение двух массивов A и B, состоящих из 5 элементов каждый, которые содержат случайные числа от 2 до 9 включительно. Воспользуйтесь формулой:

`P=A1*B1+A2*B2+ ... +AN*BN`, где `N` — размер массива.

186. Найдите соотношение `SX/SY`, где `SX` и `SY` — средние арифметические значения массивов X и Y, соответственно.

(Массивы из 10 элементов содержат случайные двузначные целые числа.)

187. Определите объем каждого из 10 цилиндров, для которых заданы радиусы оснований R_i (случайные целые числа от 5 до 25 см) и высоты H_i (случайные целые числа от 10 до 30 см).

188. Заданы 10 пар координат X_i, Y_i одних точек на плоскости и 10 пар координат A_i, B_i других точек на плоскости. Вычислите попарно расстояния между точками по формуле:

$$S_i = \sqrt{(X_i - A_i)^2 + (Y_i - B_i)^2} .$$

Занесите эти расстояния в массив S. Проиллюстрируйте задачу графически (соответствующими отрезками на экране). Выделите разными цветами наибольшую и наименьшую длину.

189. Дан одномерный массив W из 10 случайных целых чисел, каждое из которых лежит в пределах от 1 до 100. Получите новый массив R, где каждый элемент создается из массива W делением соответствующего элемента на его индекс.

190. Вычислите и представьте в виде массива последовательность первых 20 чисел Фибоначчи, если $X_1 = 1$, $X_2 = 2$, а каждый последующий элемент равен сумме двух предыдущих.

191. Имеются данные о среднесуточной температуре в течение февраля 2000 г. по Санкт-Петербургу. Вычислите среднюю температуру февраля, наибольшую и наименьшую температуры. Постройте линейный график изменения среднесуточной температуры.

192. Школьники неохотно носят одежду, отличающуюся по цвету от одежды одноклассников. Напишите программу, выбирающую 2 цвета (для мальчиков и для девочек) из 12 возможных цветов, которые сегодня будут носить все. Выбор производится случайным образом и сопровождается выводом на экран прямоугольников соответствующих цветов, внутри одного из которых написано "Сегодня этот цвет для мальчиков", а внутри другого "Сегодня этот цвет для девочек".

193. В фирме "Green Beavis Ltd" работают семь сборщиков компьютеров. Для того чтобы повысить производительность их труда, в конце недели обрабатывают сведения о количестве

компьютеров, собранных каждым из них ежедневно. Напишите программу, которая выдаст на экран следующие данные:

- наибольшее количество компьютеров, собранных одним служащим за неделю;
- среднее количество собранных за день компьютеров;
- лучший результат за один день;
- номер служащего, показавшего этот результат, и день, в который он был достигнут.

194. Дан массив *X*, состоящий из 100 целых случайных чисел, каждое из которых лежит в пределах от 3 до 13. С клавиатуры вводится целое число *N*, также лежащее в этих пределах. Определите количество элементов массива, равных числу *N*.

195. Даны два числовых массива *X* и *Y* с количеством элементов 10 и 20, соответственно. Получите массив *Z* из 30 элементов, составленный добавлением массива *X* в конец массива *Y*.

196. Дан массив из 20 случайных целых чисел, каждое из которых лежит в пределах от 10 до 50. Найдите максимальное число и его номер, а если таких чисел несколько, то подсчитайте, сколько их.

197. Дан массив из 20 случайных целых чисел, каждое из которых лежит в пределах от 10 до 50. Определите среднее арифметическое элементов массива, а также значение элемента, ближайшего к среднему, и его номер.

198. Дан массив среднемесячных температур за год. Определите, в каком месяце была самая высокая температура, а в каком самая низкая, а также среднесезонные температуры.

199. У автора этой книги была копилка, в которой было 1000 российских монеток достоинством в 1, 2, 5, 10 и 50 копеек. Задайте массив *M*(1000) случайным образом из этого набора, а затем подсчитайте, сколько в копилке было пятачков и полтинников и какова общая сумма накопленного.

200. Дан массив из десяти целых двузначных случайных чисел. Найдите сумму трех максимальных из них.

201. Заполните массив *R*(25) случайными целыми двузначными числами так, чтобы числа не повторялись.

202. Программа "Пожиратели звезд". Когда мы говорили о циклах и построении графиков функций, то писали программу о движении "звездолета", который, пролетая по траектории заданной тригонометрической функции, уничтожал планету, находящуюся от него в опасной близости. Теперь, со знанием массивов, мы можем написать более красивую и более сложную программу. Сначала заполним экран тысячью разноцветных звезд, изображенных либо точками, либо окружностями радиусом 1. Координаты всех звезд запоминаются в массивах *X*(10000) и *Y*(10000). Затем по траектории $y = 2\sin\frac{x}{2} + \frac{1}{2}\cos 2x$ начинает двигаться кортеж. Впереди сторожевой звездолет (закрашенный круг радиусом 2), проверяющий, не находится ли какая-нибудь звезда в опасной близости от армады (*S*<=20), и уничтожающий ее в таком случае (звезда заменяется точно такой же, но цветом фона). Позади движутся три крейсерских звездолета (закрашенные круги радиусом 5). В результате выполнения программы на экране должен быть проложен коридор по траектории функции с шириной 60 экранных точек (рис. 2.67).

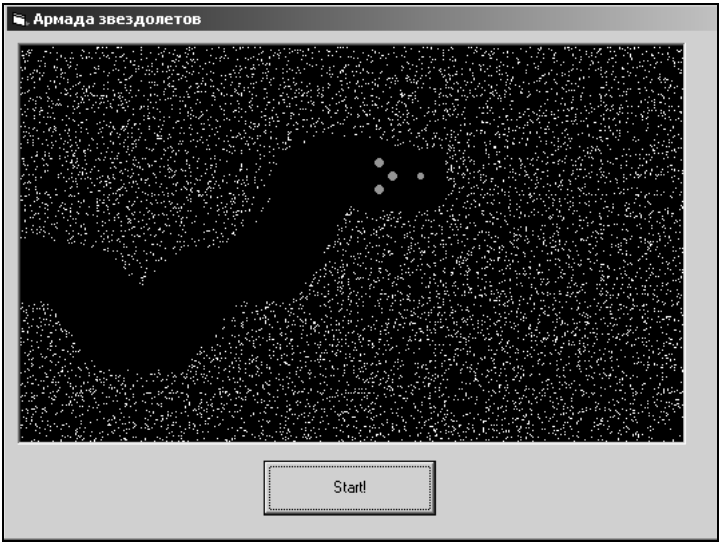


Рис. 2.67. Пожиратели звезд...

203. Случайным образом сформировать вектор $B(4)$, элементы которого являются двузначными числами и делятся на 3.

204. Дана последовательность чисел Фибоначчи, определяемая соотношениями: $u[1] = 1$, $u[2] = 1$, $u[n] = u[n-1] + u[n-2]$, $n > 2$. Проверьте на нескольких примерах, будет ли $u[5k]$ ($k = 1, 2, ...$) делиться на 5.

205. Найти k простых чисел Фибоначчи.

206. Для заданного целого числа m найти среди первых k чисел Фибоначчи хотя бы одно, делящееся на m.

207. Найти среди первых 20 чисел Фибоначчи все четные числа.

208. *Модой массива* называется число M, которое встречается в массиве наиболее часто. Если в массиве имеется несколько наиболее часто встречающихся элементов и число их вхождений совпадает, то считается, что массив не имеет моды. Напишите программу, которая либо вычисляет моду массива, либо устанавливает, что последний ее не имеет.

2.6.3. Простейшие сортировки

Одной из основных операций, производимых над массивами, являются *операции сортировки* или *упорядочивания элементов массива* по какому-либо признаку: чаще по возрастанию или убыванию — для чисел и по алфавиту — для символов и строк.

Сортировок придумано множество, и, говорят, тому, кто придумает новый эффективный метод сортировки, сразу будет вручена Нобелевская премия.

С древних времен, однако, до нас дошли два самых простых (но, конечно, не самых эффективных) способа сортировки, которые мы здесь и рассмотрим.

Сортировка выбором

Допустим, дан числовой массив из N элементов. Надо отсортировать его по возрастанию.

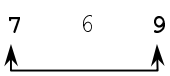
Суть способа в следующем. Находим наибольший элемент в массиве и меняем его местами с последним. Уменьшаем количество рассматриваемых элементов на 1 (т. к. последний элемент уже на своем месте). Повторяем операцию для уменьшенного на единицу массива. И так — $N - 1$ раз.

Пусть дан массив из пяти элементов:

```
8   4   9   6   7
```

Рассмотрим процесс упорядочивания по шагам.

- 8 4 **7** 6 **9**



- **6** 4 7 **8** (9)

- 6 4 7 (8) (9)

  На этом шаге третий элемент поменялся сам с собой.

- **4** **6** (7) (8) (9)

Для закрепления выполните следующие задания.

209. Напишите программу для упорядочивания массива по возрастанию методом выбора. Измените программу так, чтобы она упорядочивала массив по убыванию.

210. Дан массив из 13 чисел. Расположите числа по возрастанию. Введите с клавиатуры число M так, чтобы оно вошло в массив и получившийся массив также был бы упорядочен по возрастанию.

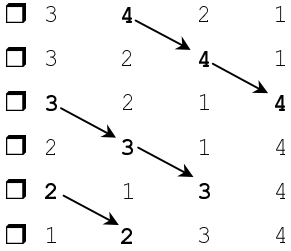
211. Дан массив из двадцати английских слов. Упорядочите его по убыванию — от Z до A.

Метод обмена или "пузырька"

Название данного метода часто вызывает нездоровый смех у молодежи, хотя никакого тайного смысла у этого метода нет. Просто он выполняется таким образом, что максимальное число после каждого шага сортировки как бы всплывает в конец массива, на свое заслуженное место. Заключается метод в следующем. Программа, начиная с первых элементов массивов, сравнивает эти элементы попарно и, в случае, если они расположены не по возрастанию, меняет их местами. В результате $(N-1)^2$ перестановок (в случае самого плохого расположения элементов массива — все элементы по убыванию) массив окажется упорядочен по возрастанию. Например, есть массив из четырех элементов:

```
4   3   2   1
```

Рассмотрим по шагам метод "пузырька" для этого массива.



И опять выполним самостоятельные задания.

212. Напишите программу, реализующую метод "пузырька". Для уменьшения количества ненужных сравнений может служить счетчик, подсчитывающий количество обменов за один полный

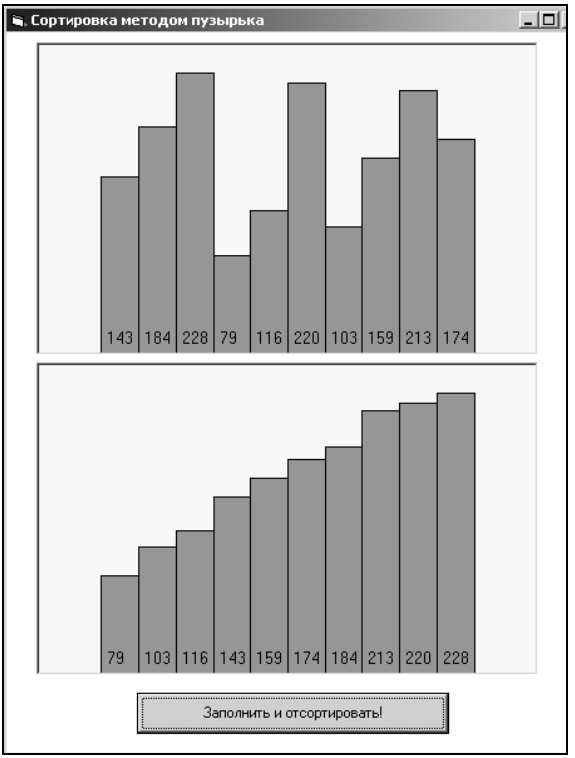


Рис. 2.68. Было случайно. Стало упорядоченно

пробег вдоль всего массива. Как только его значение станет равно нулю (т. е. ни одного нового обмена не будет произведено), это будет означать, что массив упорядочен. Для наглядности оформите исходный и получившийся массив в виде столбиковых интерпретаций друг под другом (рис. 2.68).

213. Дан массив букв, составляющих английский алфавит, но размещенных не по порядку. Напишите программу, преобразующую этот массив в алфавит английского языка.

214. Дан массив из 13 четырехбуквенных русских слов (существительных нарицательных), в единственном числе, в именительном падеже. Упорядочить их по алфавиту.

2.6.4. Двумерные массивы

Что такое *двумерный массив*? Это такой набор однотипных данных, местоположение каждого элемента которого определяется не одним индексом, а двумя. Например, для тех, кто с детства играл в "морской бой", не будет открытием, что каждая клеточка игрового поля обозначается двумя символами — буквой и цифрой, например, А5 — "мимо", И10 — "попал", Ж7 — "убит". Только в Бейсике принято в качестве индексов использовать все же целые числа. Жизненный пример применения двумерных массивов — билеты в кино или театр, имеющие для каждого зрителя две координаты — ряд и место.

Примечание

Кстати, маленькое лирическое отступление. Откуда пошла традиция указывать на билетах ряд и место? Оказывается из Франции. Когда дворяне при шпагах и гордом характере приходили в театр, имея просто билет без указания места, то нередко возникали смертоубийственные стычки из-за этих самых мест. Королю это надоело, и он обратился за помощью к ученому Декарту, который и предложил систему — "ряд—место". Эта система в дальнейшем трансформировалась в привычную нам декартову систему координат — ось *X*, ось *Y*.

Описываются подобные массивы в Бейсике тем же оператором `DIM`, после которого в скобках указываются две размерности

массива — количество строк и количество столбцов. Например, массив 5×3 объявим так:

```
DIM X(1 To 5,1 To 3)
```

X (1, 1)	X (1, 2)	X (1, 3)	X (1, 4)	X (1, 5)
X (2, 1)	X (2, 2)	X (2, 3)	X (2, 4)	X (2, 5)
X (3, 1)	X (3, 2)	X (3, 3)	X (3, 4)	X (1, 5)

Заполнение двумерных массивов и вывод их на экран

В обработке двумерных массивов есть своя специфика — использование вложенных циклов.

Заполним двумерный массив $X(3, 5)$ целыми случайными числами, лежащими в интервале от 1 до 20, и выведем массив на экран в виде таблицы.

```
CLS : RANDOMIZE
DIM X(1 To 3,1 To 5)
FOR I =1 TO 3
  FOR J=1 TO 5
    X(I, J)=INT(RND(1)*20)+1
    ? X(I, J);
  NEXT J
  ?
NEXT I
```

На что надо обратить внимание? Для начала, при выводе массива во внутреннем цикле после оператора `PRINT` стоит точка с запятой. Это дает возможность отображать массив построчно. А оператор `PRINT` без параметров, указанный после внутреннего цикла, позволяет после вывода каждой строки элементов массива переводить курсор на новую строчку.

А теперь выполните следующие задания.

215. Найдите в массиве из приведенного выше примера максимальный и минимальный элемент. При выводе массива на экран выделите их красным цветом.

216. Дан двумерный массив 5×5. Определите сумму элементов каждой строки и ту строку, в которой сумма элементов максимальна.

217. Дан массив *A* (2, 10). В первом столбце содержатся координаты *X* точек плоскости экрана, а во втором столбце — координаты *Y* тех же точек. Определите количество точек, попадающих в правую нижнюю четверть экрана, выведите их на экран, а искомые точки выделите другим цветом.

218. Определите наименьший элемент в массиве *X* (10, 10). Выделите его другим цветом.

219. Дан массив *W$*(5, 4), в котором каждая строка состоит из четырех символов, составляющих английское слово. Отсортируйте массив таким образом, чтобы слова были расположены по алфавиту.

220. В массиве *R* (5×5) поменяйте местами первую и последнюю строки.

221. В массиве *R* (5×5) замените элементы, стоящие ниже главной диагонали, нулями.

222. В массиве *R* (5×5) замените элементы главной диагонали нулями.

223. В массиве *R* (5×5) вычислите сумму элементов главной диагонали.

224. В массиве *R* (5×5) упорядочьте строки по возрастанию элементов главной диагонали.

225. Определите, является ли заданный массив 3×3 магическим квадратом, т. е. таким, суммы элементов которого в строках, столбцах и главных диагоналях равны между собой.

226. Выведите на экран номера строк массива 5×5, сумма элементов которых четна.

227. Выведите на экран изображение Андреевского флага, если у данного массива 5×5 суммы элементов диагоналей равны, и флаг Японии — в обратном случае (рис. 2.69, 2.70).

228. Одномерный массив из *N* элементов свернуть по спирали в квадратную матрицу размерностью корень квадратный из *N* по следующему образцу.

Исходный массив *S*1 (16) состоит из следующих элементов:

3, 5, 9, 7, 12, 34, 21, 13. 6, 89, 54, 66, 2, 10, 99, 55.

Создайте массив *S*2 (4, 4), вид которого представлен на рис. 2.71.

4	6	8	3	**2**
6	**9**	5	**3**	4
6	0	**1**	8	7
5	**8**	2	**3**	5
7	6	0	3	**4**

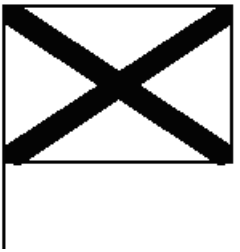


Рис. 2.69. Андреевский флаг (суммы элементов диагоналей равны)

4	6	8	3	**2**
6	**8**	5	**3**	4
6	0	**1**	8	7
5	**8**	2	**3**	5
7	6	0	3	**4**

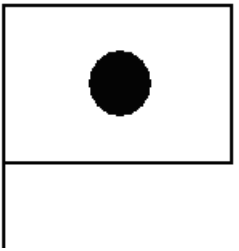


Рис. 2.70. Флаг Японии (суммы элементов диагоналей не равны)

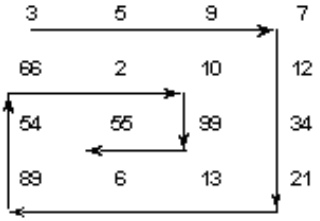


Рис. 2.71. Преобразованный в спираль одномерный массив

229. Сформируйте двумерный массив 7 на 7 по следующему правилу: элементы главной диагонали равны 1, а все ос-

тальные элементы нулевые. Распечатайте полученную матрицу.

230. Дан двухмерный массив $A(m, n)$. Напишите программу построения одномерного массива $B(m)$, элементы которого соответственно равны суммам элементов строк.

231. Транспонируйте (т. е. поменяйте строки и столбцы местами) произвольный двумерный "квадратный" массив. Дополнительные массивы не используйте.

232. Напишите программу замены элементов двухмерного массива, находящихся в четной строке и четном столбце на число 100.

233. Дан массив $A(N, N)$. Напишите программу, которая прибавляла бы к каждому элементу данной строки элемент, принадлежащий этой строке и главной диагонали.

234. Из одномерного массива длиной 8 сформируйте двухмерный массив так, чтобы первая строка нового массива содержала четные по номеру элементы исходного массива, а вторая нечетные.

235. Напишите программу, находящую в двухмерном массиве номера строк с наибольшей суммой элементов.

2.7. Работа со строковыми переменными

2.7.1. Символы и строки

Когда мы говорили о дружественном интерфейсе, то упоминали о так называемых строковых переменных. В таких переменных могут содержаться как отдельные символы, так и их последовательности длиной до 255 символов. К ним в Visual Basic применимы специальные операции, о которых мы здесь и расскажем.

Для начала надо немного отвлечься, чтобы сообщить о том, что каждый символ, представленный на клавиатуре, для компьютера переводится в *числовой код*. Эти коды объединены в стандартную международную таблицу кодов ASCII. Коды с 0 по 32 не имеют изображения на экране и служат для функций управления (пробел, клавиши управления курсором и т. д.). Далее следуют знаки

препинания, цифры, строчные и прописные буквы латинского алфавита и другие символы, которые вы можете найти на клавиатуре. Всего их 128. А еще 128 кодов (от 129 до 255) служат для расширения возможностей клавиатуры, например для генерации национальных символов — в нашем случае для кириллицы. Учить их наизусть ни в коем случае не надо — они есть в таблице. Но если ее не окажется под рукой, то вы должны определить код любого символа, используя специальные функции `ASC` и `CHR`.

2.7.2. Функции *ASC* и *CHR*

Функция `ASC` определит нам код ASCII для первого символа этой строковой переменной и имеет следующую форму записи:

```
ASC(строковая_переменная)
```

Например:

```
N=ASC("F")
print "Код заглавной буквы F —"; N
```

В результате получим:

```
Код заглавной буквы F — 70
```

Еще пример.

```
X="YAHOO"
N=ASC(X)
? N
```

В результате получим код первого символа, входящего в слово "YAHOO", т. е. "Y", который равен 89.

> **Замечание**
>
> Следует помнить, что коды заглавных и строчных букв — разные. Кроме того, если мы напрямую указываем в функции `ASC` символ или текст, то он берется в кавычки (первый пример), а если это строковая переменная, то без кавычек (второй пример).

Функция `CHR` определит нам символ, код которого указан в скобках. Форма записи функции:

```
CHR(код)
```

Например:

```
Cls
a:
n = InputBox ("Введите любой код от 33 до 128")
If N < 33 OR N >=128 Then
Print "Обратите внимание на границы для кода"
Goto a
End If
Print "Символ с кодом "; N; "- это"; Chr(N)
```

Обратите внимание на оформление программы. Сначала выполняется очистка экрана. Затем — запрос кода. Если он введен не в требуемых пределах, то программа возвращает человека к запросу — простейший, но очень полезный способ помочь пользователю.

Если все ясно, то переходите к самостоятельным заданиям.

236. Опробуйте представленную выше программу и узнайте, что за символы скрываются под кодами 33, 66, 99, 100, 128.

237. Примените функции `ASC` и `CHR` к примеру простейшей шифровки информации, когда символы вводятся побуквенно, а программа определяет их код, добавляет к ним 1 и выводит на экран вместо введенного символа символ с новым получившимся кодом. Слово для тестовой проверки такой программы — "CAT", после его побуквенного введения должно получиться "DBU".

238. Напишите программу-дешифратор для предыдущего задания. Тестовая проверка: из слова "DBU" должно получиться слово "CAT".

    Эти программы грамотно работают для первых стандартизированных 128 кодов. Чтобы правильно работать, например, с русским текстом, надо знать коды строчных и прописных букв кириллицы, которые скрываются в интервале от 129 до 255. Поэтому еще одно задание.

239. Напишите программу, выводящую на экран символы, скрывающиеся за кодами 129—255. Распечатайте или выпишите коды строчных и прописных букв кириллицы.

Но всякий раз вводить текст побуквенно — большая морока. Нельзя ли как-нибудь в Visual Basuc обрабатывать слова и строки? Конечно, можно. Для этого существуют специальные функции.

2.7.3. Функция *LEN*

Следующая функция — `LEN`. Она определяет длину введенной или существующей в переменной строковой переменной в символах. Синтаксис:

```
LEN(строковая_переменная)
```

Например,

```
Cls
F=InputBox ("Введите Вашу фамилию")
N=Len(F)
Print "В вашей фамилии "; N; "букв"
```

Представленная программа выясняет количество букв во введенной пользователем фамилии. Причем обратите внимание, что функция `LEN` учитывает не только буквы, но и символы, т. е. она распознает и пробелы, и знаки препинания, и цифры, содержащиеся во введенном тексте. Например:

```
Cls
F = InputBox ("Введите Ваш адрес")
N=Len(F)
Print "В вашем адресе "; N; "символов"
```

А теперь самостоятельное задание.

240. Определите с помощью предыдущего примера, сколько символов будет в следующем адресе:

     191110, Россия, Санкт-Петербург, Чкаловский пр., 78–33.

2.7.4. Функции *LEFT*, *RIGHT* и *MID*

Для получения фрагмента строки (или значения строковой переменной) применяются специальные функции.

Функция `Left` выделяет из введенной строковой переменной `N` символов *слева*:

```
Left (строковая_переменная, N)
```

Рассмотрим пример.

```
Cls
F = "ГАЗОНОКОСИЛЬЩИК"
L = Left (F, 5)
Print L
```

На экране появится слово "ГАЗОН", т. е. первые пять символов слева исходной строковой переменной.

Функция `Right` вырезает из введенной строковой переменной `N` символов *справа*:

```
Right (строковая_переменная, N)
```

Например:

```
Cls
F = "ГАЗОНОКОСИЛЬЩИК"
R = Right (F, 9)
Print R
```

На экране появится слово "КОСИЛЬЩИК", т. е. первые девять символов справа исходной строковой переменной.

Наконец, функция `MID` извлекает `N2` символов, начиная с символа с номером `N1` исходной строковой переменной:

```
Mid (строковая_переменная, N1, N2)
```

Например,

```
Cls
F = "ГАЗОНОКОСИЛЬЩИК"
M = Mid (F, 7, 4)
Print M
```

На экране появится слово "КОСИ", т. е. четыре символа, начиная с седьмого исходной строковой переменной.

Используя эти функции, для начала можно с их помощью поиграть в игру "Наборщик", когда из букв, составляющих какое-либо слово, нужно составить другие слова. Для этого предназначена функция конкатенации или, по-простому, слияния, которая записывается просто знаком +. Например:

```
Cls
F = "ГАЗОНОКОСИЛЬЩИК"
W1 = Mid (F, 4, 2)+Right (F, 7)
```

```
W2 = Mid (F, 4, 2)+Left (F, 2)
W3 = Mid (F, 9, 1)+MIid (F, 7, 2)+Mid (F, 11, 2)+Mid (F,7,2)
Print W1
Print  W2
Print W3
```

Выполните упражнения.

241. Определите, какие слова получатся в результате выполнения приведенной выше программы?

242. Напишите программу, которая выдаст на экран пять слов максимальной длины из слова "ЭЛЕКТРИЧЕСТВО". Побеждает тот, у кого сумма букв во всех словах наибольшая.

В качестве очередного примера приведем задачу подсчета слов во введенном тексте. Как известно, для компьютера словом является последовательность символов, заключенная в пробелы с двух сторон. Подчеркиваю, это не обязательно слово, в привычном для нас понимании, а любой набор символов, например, 45ро9? или ВАР56+УР47. Поэтому, в простейшем случае, подсчет количества слов во введенном тексте сводится к подсчету количества пробелов и добавлении к полученному значению единицы. (Почему так? Очень просто: слов два, а пробел между ними один; слов три — а пробелов два и т. д.) Получаем программу.

```
Cls
W = InputBox ("Введите текст телеграммы")
N = Len (W) : K=0
FOR I=1 TO N
  P = Mid (W, I, 1)
  IF P="  " Then K=K+1
Next
Print "В вашей телеграмме — "; K+1; "слов"
```

Программа работает очень просто. Она определяет длину текста в символах и заносит это число в переменную `N`. Затем устанавливает счетчику пробелов `K` нулевое значение. После чего в цикле вырезает из текста последовательно по одному символу и проверяет, а не является ли он пробелом. Если это так, то увеличивает счетчик пробелов `K` на единицу, а если нет — берет следующий символ. По завершении

цикла в переменной k хранится количество пробелов в тексте, и мы выводим ответ о количестве слов на экран, добавляя к k еще единичку.

243. Используя пример с подсчетом слов в телеграмме, напишите программу, имитирующую отделение связи с очень хорошим обслуживанием. Программа должна выяснить имя клиента и в дальнейшем обращаться к нему только по имени. Запрашивается также регион, куда посылается телеграмма. Их три: Россия (коэффициент 1), страны СНГ (стоимость одного слова умножается на 2) и дальнее зарубежье (стоимость одного слова умножается на 5). По России стоимость одного слова составляет 3 руб. 50 коп. (причем неважно, какой длины слово). Затем у клиента запрашивается текст телеграммы и денежная сумма, определяется количество слов, стоимость телеграммы. Если денег ровно столько, сколько надо, его благодарят и прощаются. Если больше, чем надо, то ему предлагают сдачу и прощаются. Если меньше, то просят добавить необходимую сумму, а затем, после расчета, с клиентом прощаются. А для пущей красоты я обычно прошу нарисовать окошко телеграфа, в прорезях которого и происходит диалог компьютера с пользователем (рис. 2.72).

Рис. 2.72. Телеграф

2.7.5. Сравнение строковых переменных

Над строковыми переменными тоже можно производить операции сравнения. Большей будет являться та переменная, которая начинается с символов, более близких к концу алфавита, т. е. имеющих больший код, а если символы совпадают, то более длинное слово. Строковые переменные считаются идентичными,

если они полностью тождественны. Если они отличаются пробелами в начале или конце, то они уже не идентичны!

Например:

```
"DOG" > "CAT",
"ELEFANT" < "MOUSE"
"TIGER2" > "TIGER1"
"M16  " > "M16"
```

Выполните следующие задания.

244. Напишите программу, проверяющую, является ли введенное слово или фраза палиндромом, т. е. читающимся слева направо и справа налево одинаково (например, шалаш, казак, а роза упала на лапу Азора). Программа сообщает "Да, это палиндром" или "Нет, это не палиндром" и выводит на экран введенный текст в варианте слева направо и справа налево. Здесь необходим цикл посимвольного чтения от *N* (длины текста) до 1.

245. Напишите программу, подсчитывающую количество слогов во введенном слове.

246. С клавиатуры вводятся 10 слов. Напишите программу, которая:

- напечатает все слова из списка, отличные от слова "SUN";
- напечатает слово, ближайшее к началу алфавита в списке (считаем, что все слова различны; выполнять аналогично поиску минимального из ряда чисел);
- слово, составленное из последних символов всех слов списка;
- все слова из списка, содержащие три буквы.

247. В тексте, содержащем между словами от 1 до 3 пробелов, оставить только по одному.

248. Подсчитать, сколько раз входит каждый символ в данную строку.

249. Написать программу шифратор и дешифратор, ставящую в соответствие русским символам соответствующие латинские и наоборот (аналог так называемого транслита).

2.7.6. Преобразование строчных и прописных букв

Если ваш текст напечатан строчными буквами, вы хотите заменить его прописными или наоборот, не надо заново его набирать. Для этого есть две функции:

- `UCASE (строковая_переменная)` — преобразует все буквы строки в прописные.
- `LCASE (строковая_переменная)` — преобразует все буквы строки в строчные.

Пример:

```
CLS
N="I have 50"
R=" рублей"
Print N;R
N1=Ucase(N)
R1=Ucase(R)
Print N1, R1
```

Результатом работы программы будет следующее:

```
I have 50 рублей
I HAVE 50 РУБЛЕЙ
```

Функции эти очень полезны, когда мы просим пользователя ввести один из возможных ответов, например "YES" или "NO", или просто "Y" или "N", а пользователь, естественно, может ввести ответ как строчными, так и прописными буквами. В таком случае, с помощью функций `UCASE` или `LCASE` сначала надо привести ответ к требуемому виду, а потом проверять условие. Например:

```
N=InputBox("Будете еще играть? (Y/N))
IF Ucase(N)="N" Then Print "До свидания"
```

2.7.7 Функция определения вхождения подстроки

Допустим, мы хотим найти в тексте какое-либо слово. Нам на помощь приходит функция `INSTR`, имеющая следующий синтаксис:

```
INSTR(N, F, R)
```

где `N` — позиция, с которой вы хотите начинать поиск (необязательный параметр), `F` — строковая переменная, в которой будет производиться поиск, `R` — подстрока, поиск которой осуществляется. В случае отсутствия этой позиции поиск начнется с первого символа строковой переменной. Функция `INSTR` укажет нам номер позиции, с которой начинается вхождение искомой подстроки, или 0 в следующих случаях:

- подстрока не найдена;
- значение `N` больше длины исходной строковой переменной;
- длина строковой переменной нулевая.

Если подстрока пустая, то результатом будет `N`, а при его отсутствии — 1. Поиск прекращается с первым нахождением подстроки. Например:

```
Cls
F="Свиноводство, овцеводство, пчеловодство"
R="чело"
N=Instr (F, R)
IF N <> 0 THEN
Print "Слово 'чело' в данной строке есть и начинается _ с "; N; "позиции"
```

Результатом выполнения программы будет фраза:

```
Слово "чело" в исходной строке есть и начинается с 29 позиции.
```

Переходим к самостоятельным упражнениям.

250. Напишите программу для вычеркивания из данного слова всех букв "K" и "G".
251. Напишите программу для вычеркивания в данном слове всех букв, стоящих на нечетных местах после буквы "а".
252. Напишите программу для вычеркивания из данного слова всех букв "р", перед которыми стоит буква "а".
253. Напишите программу для вычеркивания из данного слова каждой третьей буквы.
254. Вычеркните из данного слова все буквы "с" и "л", стоящие на нечетных местах.
255. Напишите программу, проверяющую, все ли буквы данного слова одинаковы.

256. Напишите программу, выясняющую, можно ли из букв данного слова *N* составить слово *M*.

257. Напишите программу для проверки, есть ли в данном слове буква "в". Если есть, то найдите номер первой из них.

258. Напишите программу, выясняющую, есть ли в данном слове буква "к", и если есть, то замените все буквы "а" в этом слове на "с".

259. Напишите программу, проверяющую, все ли буквы данного слова, стоящие на четных местах, одинаковы.

260. Определите, есть ли в данных словах *N* и *M* одинаковые буквы.

261. Выясните, есть ли в данном слове буква "в", стоящая на нечетном месте.

262. Определите, имеются ли в данном слове две одинаковые буквы, идущие подряд.

263. Заменить в исходном тексте "photo, graph, philophon, cophe" все сочетания "ph" на символ "f".

264. Подсчитать, сколько раз слвоосочетание ph входит в исходный текст предыдущего задания.

265. Подсчитать, по сколько раз входит каждая буква русского алфавита в следующую фразу:

"Санкт-Петербург отметил свое трехсотлетие в обстановке великого подъема настроения всего населения города и теплого отношения со стороны других стран и народов!".

266. Вычислить сумму кодов всех символов, входящих в выражение "Я уже кое-что понимаю в Visual Basic!".

267. С клавиатуры вводится некое двузначное число. Вывести его на форму в словесной записи. Например 87 — "восемьдесят семь".

268. Усложните предыдущую задачу — пусть это будет трехзначное число.

269. Теперь давайте еще более усложним нашу задачу, добившись того, что вводиться и записываться затем в словесной форме будет любое число, длиной не более 12 цифр.

270. С клавиатуры вводится число в римской записи. Записать его в цифровой десятичной и в словесной формах. Напри-

мер, вводим MDCXXIV — получаем 1624 — тысяча шестьсот двадцать четыре.

271. С клавиатуры вводится дата сегодняшнего дня. Разработать и реализовать алгоритм, выводящий дату завтрашнего дня. (Обратите внимание, что месяцы имеют различное количество дней, что есть високосные годы и всякое такое ☺). Усложните задачу таким образом, чтобы программа выводила дату, отстоящую от введенной на N дней.

272. Пусть с клавиатуры вводится четырехзначное число, означающее количество копеек. Представьте его в словесной форме, описывающей количество рублей и копеек, например:

    2142 — "Двадцать один рубль сорок две копейки".

    Обратите внимание на падежи!

273. Дан непустой текст, в который входят только цифры и буквы. Определить, удовлетворяет ли он следующим свойствам:

    - текст является записью десятичного числа, кратного пяти;
    - текст начинается с ненулевой цифры, за которой следуют только буквы, и их количество равно числовому значению этой цифры;
    - текст состоит только из цифр;
    - текст состоит только из букв;
    - сумма числовых значений цифр, входящих в текст, равна длине текста.

2.8. Программирование при помощи процедур и функций

2.8.1. Процедуры

В практике программирования часто бывает необходимо выполнять одни и те же вычисления, но при различных исходных данных. Чтобы исключить повторение одинаковых записей и сделать тем самым программу проще и понятнее, можно выделить эти повторяющиеся вычисления в самостоятельную часть программы, которая может быть использована многократно по

мере необходимости. Такая автономная часть программы, реализующая определенный алгоритм и допускающая обращение к ней из различных частей общей программы, называется *процедурой*. Любая процедура содержит заголовок и раздел операторов. По сути, процедура очень похожа на программу. Синтаксис объявления процедуры:

```
  Sub MyProc(Prm1, Prm2, Prm3 … PrmN)
     [Operator1]
     [Operator2]
     [Operator3]…
     [OperatorN]
End Sub
```

`MyProc` — это имя создаваемой процедуры.

`Operator1`, `Operator2`, `Operator3`, `OperatorN` — операторы, используемые в процедуре.

Именование процедуры должно проходить по определенным правилам. В скобках за именем процедуры следуют формальные параметры, от которых будет зависеть результат выполнения процедуры.

Формальные параметры — это наименования переменных, через которые передается информация из основной программы или другой процедуры в процедуру.

Говоря о процедурах и функциях, следует отметить, что переменные, используемые в программе, могут быть *локальными* и *глобальными*. Локальные переменные (объявленные только в процедуре или функции) существуют только во время выполнения процедуры или функции. Глобальные переменные (объявленные в самой программе) распространяются, в том числе и на процедуры и функции. Такие переменные существуют, пока программа выполняется.

Для того чтобы "запустить" процедуру в работу, необходимо к ней обратиться (ее вызвать). Вызов процедуры производится следующим образом:

```
MyProc Prm1, Prm2, Prm3 … PrmN
```

или

```
call MyProc(Prm1, Prm2, Prm3 … PrmN)
```

Замечание

Список фактических параметров может отсутствовать.

Соответствие между фактическими и формальными параметрами должно быть следующим.

- Количество фактических параметров должно быть равно количеству формальных параметров.
- Соответствующие фактические и формальные параметры должны совпадать по порядку следования и по типу. Соответствующие параметры не обязательно должны быть одинаково обозначены (имя формального параметра может быть не таким, как у фактического).

Выполнение оператора вызова процедуры состоит в следующем:

- все формальные параметры заменяются соответствующими фактическими;
- создается так называемый динамический экземпляр процедуры, который и выполняется;
- после выполнения процедуры происходит передача управления в основную программу, т. е. начинает выполняться оператор, следующий за оператором вызова процедуры.

Пример использования процедуры в программе (без параметров).

Задача. Вывести на форму значение выражения: `(4+5+6)*200/2`, используя процедуру `Res`.

Текст программы:

```
Sub Res
   Print ((4+5+6)*200/2)
End Sub

Res
```

Замечание

Объявлять процедуру вы можете в любой части программы (в начале, в середине, в конце).

Рассмотрим пример использования процедуры в программе (с параметрами).

Задача. Ввести значения трех переменных при помощи функции `Vvod` и распечатать значение введенных переменных.

Текст программы:

```
option explicit
dim a, b, c     'Описание глобальных переменных
Sub Vvod(x)     'процедура ввода значений переменных,
                x - формальный параметр
    x=InputBox("Введите значение переменной:" _ ,"Окно ввода")
End Sub
Vvod a      'Обращение к процедуре vvod, a - фактический па-
раметр
Vvod b      'Обращение к процедуре vvod, b - фактический па-
раметр
Vvod c      'Обращение к процедуре vvod, c - фактический па-
раметр

'Вывод введенных значений переменных на форму
Print "Вы ввели три переменных: a;b;c
```

2.8.2. Функции

Функция — это процедура, возвращающая результат. Поэтому ее надо применять, когда вызывающая программа должна вернуть только один результат. Оформляется аналогично процедуре. Отличительные особенности функции: она имеет только один результат выполнения. Результат обозначается именем функции и возвращается (передается) в основную программу. Функция объявляется следующим образом:

```
Function MyFunc (Prm1, Prm2, Prm3 … PrmN)
     [Operator1
     [Operator2]
     [Operator3]
     …
     MyFunc =result
     [OperatorN]
End Function
```

`Function` и `End Function` — это служебные слова, означающие начало и конец объявления процедуры.

`MyFunc` — это имя создаваемой функции.

`Prm1`, `Prm2`, `Prm3 … PrmN` — формальные параметры.

`Operator1`, `Operator2`, `Operator3`, `OperatorN` — операторы, используемые в процедуре.

`MyFunc = result` — обязательный оператор (в теле функции ее возвращаемое значение обязательно должно быть присвоено переменной с именем функции).

Для вызова функции достаточно указать ее имя (с фактическими параметрами) в любом выражении. Отметим, что имя функции можно использовать в арифметических выражениях и других командах.

Вызов функции производится следующим образом:

- *без присваивания:*

```
MyFunc Prm1, Prm2, Prm3 … PrmN
```

- *с присваиванием:*

```
x=MyFunc (Prm1, Prm2, Prm3 … PrmN)
```

Замечание

Внутри тела процедуры или функции можно объявлять новые переменные при помощи ключевого слова `Dim`.

Пример использования функции в программе (без параметров).

Задача. Вывести на экран значение выражения: `(4+5+6)*200/4`, используя функцию `summa`.

Текст программы:

```
Function Res
   Print ((4+5+6)*200/4)
End Function
Res
```

Рассмотрим следующий пример.

Задача. Вывести на экран значение пяти введенных переменных, а также их удвоенную величину.

```
Sub Res(i,x)
  Print i;"-е введенное число: ;x; ", _
удвоенное число: ";2*x
```

```
End Sub
dim a, i
For i=1 to 5
  a=InputBox ("Введите число: ", _
"Окно ввода числа: ")
  Res i,a
Next
```

Пример совместного использования функции и процедуры.

Задача. Определить расстояние, пройденное физическим телом, зная исходные время, скорость и ускорение.

Текст программы:

```
Dim v, t, a

Function Rasst(x, y, z)
 Rasst = x * y + z * y * y / 2
End Function

Sub Vvod(param, x)
  x = InputBox("Введите значение параметра" _
& param, "Окно ввода" & param, "0")
End Sub

Private Sub Command1_Click()
End Sub
Print "Задача:"
Print "Определить расстояние, пройденное _ физическим телом"
Print "за время t, со скоростью v, с ускорением a"
Vvod "скорость", v
Vvod "время", t
Vvod "ускорение", a
Print "Тело прошло расстояние"; Rasst(v, t, a)
End Sub
```

Пример демонстрации стиля программирования, который называется *процедурным программированием.*

```
Dim a, b


Sub Vvod(x)
```

```
     'Ввод значения переменной
     x = InputBox("Введите переменную: ", "Окно ввода пере-
менной: ")
End Sub

Sub change(x, y)
     Dim z
     'обмен значениями двух переменных a и b
     z = x
     x = y
     y = z
End Sub

Sub output(x, y)
     'Вывод переменных
     Print "Переменные после обмена значениями: a = "; x; ",
b = "; y
End Sub

Private Sub Command1_Click()
'Процедурный стиль программирования состоит
'просто в последовательном вызове процедур
Vvod a
Vvod b
change a, b
output a, b
End Sub
```

А теперь самостоятельные задания.

274. P принимает значения $(n!+4)^3$, если $n \geq 5$, и значения $\sin(n!)$, если $n < 5$. Напишите программу вычисления P.

275. Даны натуральные числа m и n. Вычислите $\max(\min(m,n),\min(2,6))$.

276. По вещественному значению x вычислите значение функции $sh(x) * tg(x+1) - tg^2(2 + sh(x-1))$.

277. Опишите функцию `Stepen (x, n)`, зависящую от вещественного x и натурального n и вычисляющую (посредством умножения) величину x^n. Используйте ее для вычисления значения выражения $2{,}4^k + (a+1)^{-5}$.

278. Даны три числа. Определите их наибольший общий делитель.

279. Даны отрезки a, b, c, d. Для каждой тройки этих отрезков, из которых можно построить треугольник, найдите площадь данного треугольника. Определите процедуру, определяющую площадь треугольника со сторонами x, y и z, если такой треугольник существует.

280. Даны координаты вершин двух треугольников. Определите, какой из них имеет большую площадь.

281. Найдите наименьшее общее кратное четырех заданных натуральных чисел.

282. Дано натуральное число n. Выясните, является ли оно полным квадратом. Определите функцию, позволяющую распознавать полные квадраты.

283. Дано натуральное число n. Выясните, является ли оно степенью пятерки. Определите функцию, позволяющую распознавать степени пятерки.

284. Дано натуральное число n. Выясните, является ли оно простым. Определите функцию, позволяющую распознавать простые числа.

285. Даны три натуральных числа. Определите их наибольший общий делитель.

286. Числа Фибоначчи $U0$, $U1$, $U2$ определяются следующим образом:

$U0 = 1$, $U1 = 2$, $Un = Un - 1 + Un - 2$ ($n = 2, 3, \ldots$). Напишите программу вычисления Un для данного неотрицательного целого n. Воспользуйтесь функцией.

287. Для каждого двумерного массива $X(3, 4)$, $Y(5, 3)$, $Z(4, 6)$ определите номер строки с максимальной суммой положительных элементов.

288. Напишите программу, предлагающую пользователю меню из десяти функций и строящую по ним графики этих функций в зависимости от выбора пользователя.

289. Напишите программу для младших школьников, проверяющую знание ими таблицы умножения от 2 до 12. Учащемуся задаются 5 примеров перемножения случайных чисел в заданном интервале. Оценкой является количество правильных ответов. Используйте подпрограмму для печати замечаний в ответ на каждый результат, вводимый пользователем. За правильный ответ замечание должно быть поощрительным, за неправильный — сожалеющим. Чтобы сделать опрос более интересным, необходимо заготовить по десять замечаний для правильных и неправильных ответов и выбирать их случайным образом, обращаясь при этом к пользователю по имени, запрошенному в начале программы. Сделайте красочную заставку, легкое музыкальное сопровождение тоже не будет лишним.

290. Напишите программу, отображающую цветное кольцо. Используя ее в качестве подпрограммы, нарисуйте олимпийский флаг.

291. Напишите программы, находящие минимальное и максимальное значения из трех чисел X, Y и Z, введенных с клавиатуры. Используя их в качестве подпрограмм, напишите программу, вычисляющую значение следующей функции:

$$S = \frac{\max(x, y, z) - 2^x \min(x, y, z)}{\sin 2 + \dfrac{\max(x, y, z)}{\min(x, y, z)}}.$$

292. Даны два одномерных массива из 20 элементов каждый. Элементом является случайное целое двузначное число. Напишите программу с использованием подпрограммы, которая изменяет исходный массив путем деления четных чисел на их индексы. Используя эту подпрограмму, определите, в каком из массивов было произведено больше замен.

293. Даны длины a, b и c сторон треугольника. Найти медианы треугольника, сторонами которого являются медианы исходного треугольника.

Медиана, проведенная к стороне a вычисляется по формуле: $m = 0{,}5\sqrt{2b^2 + 2c^2 - a^2}$.

294. Даны две квадратные вещественные матрицы. Напечатать квадрат максимального элемента той из них, в которой наименьший след (сумма диагональных элементов), считая, что такая матрица одна.

2.9. Рекурсия

Рекурсия (самоповторение) — это действие, возвращающееся к "самому себе". Существует два вида рекурсии:

- *прямая* рекурсия (процедура или функция вызывает саму себя);
- *косвенная* рекурсия означает, что одна процедура или функция вызывает другую процедуру или функцию, а это в свою очередь прямо или косвенно приводит к вызову первоначальной процедуры или функции.

Рекурсию следует использовать только тогда, когда задача легко поддается рекурсивному решению. Важно отметить, что любая задача, которая решена рекурсивно, может быть решена и без рекурсии.

С понятием "рекурсия" тесно связано понятие *"рекуррентная последовательность"*, вычисление *n*-го члена которой производится с помощью рекурсии. Определим это понятие.

Числовая последовательность $\{x_k\}$ называется *рекуррентной последовательностью*, если

$$\begin{cases} x_0 = a_0, x_1 = a_1, x_{p-1} = a_{p-1} \\ x_k = f(k, x_{k-1}, x_{k-2}, ..., x_{k-p}), \end{cases} \text{ где } k = p,\ p = 1.$$

Приведем несколько примеров.

Пример вычисления значения факториала введенного натурального числа *n*.

```
Function RecFact(n)
'Рекурсивное вычисление факториала
  If n = 0 Then
```

```
       RecFact = 1
      Else
       RecFact = n * RecFact(n - 1)
  End If
End Function

Function NonRecFact(n)
'Вычисление факториала при помощи цикла
 Dim P
 P = 1
  While (n > 1)
    P = n * P
    n = n - 1
  Wend
  NonRecFact = P
End Function

Private Sub Command1_Click()
Dim n
n = CInt(InputBox("Введите натуральное число", _ "Вычисление
факториала, натурального n:", "0"))
Number = n
Print ": "; _ RecFact(n)
Print "Нерекурсивно вычисленный факториал : "; _
NonRecFact(n)
End Sub
```

Пример нахождения наибольшего общего делителя (НОД) двух целых чисел.

```
FUNCTION RecNod(n,m)
'Рекурсивное вычисление НОД
  If n=m Then
       RecNod=n
      ElseIf n>m Then
            RecNod=RecNod(n-m, m)
```

```
          Else
           RecNod=RecNod(n, m-n)
  End If
End FUNCTION

FUNCTION NonRecNod(n,m)
'Вычисление НОД с помощью цикла
  While (n<>m)
   If n>m Then
        n=n-m
       Else
        m=m-n
   End If
  WEnd
  NonRecNod=n
End FUNCTION

Private Sub Command1_Click()
Dim n, m
n=CLng(InputBox("Введите натуральное число n", _ "Вычисление
НОД: ","0"))
m=CLng(InputBox("Введите натуральное число m", _ "Вычисление
НОД: ","0"))
Print "Рекурсивно вычисленный НОД: " _ RecNod(abs(n),abs(m))
Print "Нерекурсивно вычисленный НОД:" _ NonRec-
Nod(abs(n),abs(m))
End Sub
```

Пример. При помощи рекурсивной функции найти и вывести на экран k-й элемент последовательности Фибоначчи. Нумерация данных чисел начинается с 0.

```
FUNCTION Fib(k)
If k=0 or k=1 Then
       Fib=1
    Else
       Fib=fib(k-1)+Fib(k-2)
```

```
End If
End FUNCTION

Private Sub Command1_Click()
dim k
k=Cint(InputBox("Введите k: ", "Вычисление к-го _ элемента
посл-ти Фибоначчи: "))
Print k;"-й элемент последовательности Фибоначчи: _ "; Fib(k)
End Sub
```

Пример. Вычисление значения следующего выражения, используя рекурсию: sqr(1+(n+1)*sgr(1+(n+2)*sqr(1+(n+3)*sqr(1+... ,

где n — натуральное число, k — количество корней.

```
FUNCTION Kor(i, n)
If i=k Then
     Kor=sqr(1+(n+k))
   Else
     Kor=sqr(1+(n+i)*Kor(i+1,n))
End If
End FUNCTION

Private Sub Command1_Click()
Dim n,k,i
n=CInt(InputBox("Введите натуральное число n: ", _ "Ввод па-
раметров:","1"))
k=CInt(InputBox("Введите количество корней k: ", _ "Ввод па-
раметров:","1"))
i=1
Print "Значение вычисленного корня: ";Kor(i,n))
End Sub
```

295. Дано натуральное число n. Вычислите $(2n)!$ и $2n!$; при этом используйте рекурсивную функцию вычисления факториала.

296. Переделайте программу `Fib.vbs`, так чтобы она находила k-й член последовательности Фибоначчи, но последовательность начиналась бы не с 1, а с 0:

     0, 1, 1, 2, 3, 5, 8, 13, 21, ...

297. Проверьте, будет ли k-й член последовательности Фибоначчи делиться на 5.

298. Опишите рекурсивную функцию Stepen (x,n), формальными параметрами которой являются вещественная переменная x и натуральная переменная n, вычисляющую величину x^n следующим образом:

$$x^n = \begin{cases} 1, & если\ \ n = 0 \\ x \cdot x^{n-1}, & если\ \ n \neq 0 \end{cases}$$

299. Напишите рекурсивную процедуру, при выполнении которой на экран будет выводиться отрезок натурального ряда чисел.

300. Напишите программу вычисления первого числа Фибоначчи, большего m (m>1), включающую рекурсивную функцию, которая основана на непосредственном использовании соотношения $u_n = u_{n-1} + u_{n-2}$.

301. Числа Фибоначчи второго порядка u_0, u_1, u_2,... определяются следующим образом:

$$\begin{cases} u_0 = 1 \\ u_1 = 2, u_2 = 3 \\ u_n = u_{n-1} + u_{n-2} + u_{n-3}, n = 3,4,... \end{cases}$$

Напишите программу вычисления u_n для данного неотрицательного целого n. Используйте рекурсивную функцию.

302. Вычислите, используя рекурсию:

а) $\sqrt{3 + \sqrt{3 + \sqrt{3 + ... + \sqrt{3}}}}$;

б) $\sqrt{1 + 2\sqrt{1 + 3\sqrt{1 + 4\sqrt{...}}}}$.

303. Дано 5 различных натуральных чисел. Напечатать все перестановки этих чисел.

304. Имеется 10 населенных пунктов, перенумерованных от 1 до 10. Некоторые пары пунктов соединены дорогами. Определить, можно ли попасть по этим дорогам их 1-ого пункта в n-ый.

Информация о дорогах задается в виде последовательности пар чисел *I* и *j* (*i*<*j*), указывающих, что *i*-ый и *j*-ый пункты соединены дорогой. Признак конца последовательности — пара нулей.

Дать на форме графическую интерпретацию задачи.

2.10. Работа с файлами

Сейчас, когда в ваших головах и руках уже есть практически все необходимые инструменты для написания сложных программ, осталось немножко поднапрячься и узнать, что хранить исходные данные для больших программ очень удобно в виде отдельных файлов. Тогда при запуске программ нам не надо будет каждый раз эти данные вводить, а при получении результата мы сможем сохранять данные в файловом виде и делать с ними что хотим ☺.

Для работы с файлами Visual Basic обладает немалыми средствами.

Но сначала проинформируем читателя, что файлы, с которыми мы будем работать, делятся на два типа:

- ❒ файлы последовательного доступа;
- ❒ файлы произвольного доступа.

2.10.1. Файлы последовательного доступа

Файлы последовательного доступа можно сравнить с музыкальными записями на аудиокассете — для поиска нужной песни приходится перематывать кассету и последовательно ее прослушивать. Зато они (эти файлы) очень просты, записываются в виде простого текстового файла и могут обрабатываться любым текстовым редактором.

Операторы, предназначенные для работы с файлами последовательного доступа, позволяют нам:

- ❒ открыть файл для записи в него или для чтения уже имеющейся в нем информации;
- ❒ записать в открытый файл новую информацию из программы;

- извлечь данные из открытого файла и обработать их в программе;
- закрыть файл после завершения работы с ним.

Теперь рассмотрим эти операции подробнее.

Команда открытия файла

```
Open ИмяФайла For РежимРаботы As ДескрипторФайла
```

`ИмяФайла` — это полный путь к файлу на диске, взятый в кавычки. Если указывается только одно имя, то файл должен находиться в папке проекта.

`РежимРаботы` может принимать одно из трех значений:

- `Output` — для записи данных (если информация в файле уже сеть, то она в таком случае будет стерта);
- `Append` — для добавления информации в конец файла;
- `Input` — для чтения из файла данных.

`ДескрипторФайла` — любое целое число от 1 до 511. Оно служит идентификатором файла в программе.

Команда закрытия файла

```
Close # [Список Дескрипторов]
```

`Список дескрипторов` — это записанные через запятую идентификаторы тех файлов, которые подлежат закрытию. Если список отсутствует, то будут закрыты все открытые файлы.

Запись в файл

Данные записываются в файл двумя способами:

- способ `Write`;
- способ `Print`.

В обоих способах запись данных в файл производится текстовыми строками.

Текстовая строка — это последовательность символов, заканчивающаяся знаком перехода на новую строку или знаком возврата каретки (коды ASCII 13 и 10).

Текстовый файл — это последовательность текстовых строк.

Операторы записи выглядят так:

```
Write # Дескриптор файла, [Список значений]
Print # Дескриптор файла, [Список значений]
```

Список значений — это значения или переменные, записанные через разделитель. Если список значений отсутствует, то в файл будет записана пустая строка.

Работа операторов `Write` и `Print` различна.

Оператор Write

Разделителем в списке значений является запятая. Список значений просматривается последовательно, и элементы списка записываются в одну текстовую строку файла через запятую. Элементы типа `string` заключаются в кавычки. После записи последнего элемента записывается символ перехода на новую строку.

Оператор *Print*

Разделителем является точка с запятой либо запятая. От этого зависит, как значения будут записаны в текстовую строку файла.

В случае точки с запятой значения будут записываться подряд, без промежутков между ними.

В случае запятой, значения будут записываться в 14-символьные зоны вывода.

Кроме того, в списке значений оператора `Print` могут присутствовать функции:

- `Spc(n)` — для вставки n пробелов между значениями в текстовой строке;
- `Tab(n)` — для указания номера n позиции для записи следующего значения.

На следующем примере посмотрите различия в полученных файлах при записи двумя различными способами.

Запишем в файл следующие сведения об автомобилях: марка автомобиля, его цвет и год выпуска.

Сначала воспользуемся способом `Print`. Имя файла, в который будем записывать информацию, назовем car.txt

```
Dim Marka As String, Tsvet As String, God As _ integer
Open "car.txt" For Output As #1
For i = 1 To 5
Marka = InputBox ("Ввод", "Введите марку _ автомобиля")
Tsvet = InputBox ("Ввод", "Введите цвет _ автомобиля")
God = InputBox ("Ввод", "Введите год выпуска _ автомобиля")
Print #1, Tab(5);Marka;Spc(4);Tsvet;Spc(3);God
Next
Close #1
```

А теперь воспользуемся способом `Write`.

```
Dim Marka As String, Tsvet As String, God As _ integer
Open "car.txt" For Output As #1
For i = 1 To 5
Marka = InputBox ("Ввод", "Введите марку _ автомобиля")
Tsvet = InputBox ("Ввод", "Введите цвет _ автомобиля")
God = InputBox ("Ввод", "Введите год выпуска _ автомобиля")
Write #1, Marka,Tsvet,God
Next
Close #1
```

Откроем эти файлы для чтения, выведем их содержимое на форму и найдем отличия.

Чтение из файла можно производить тремя способами:

- оператором `Input`;
- оператором `Line Input`;
- с помощью функции `Input`.

Рассмотрим их на примерах. Но прежде познакомимся с функцией `EOF` (End Of File). Единственным ее аргументом является дескриптор файла. Функция может принимать только два значения — true или false.

- С помощью оператора `Input`. В этом примере будем выводить данные в `ListBox` (рис. 2.73).

```
Dim Marka As String, Tsvet As String, God As _ integer
Private Sub Command1_Click()
```

```
Open "car.txt" For Input As #1
Do Until EOF(1)
Input #1, Marka, Tsvet, God
List1.AddItem Marka
List1.AddItem Tsvet
List1.AddItem God
Loop
Close #1
End Sub
```

Рис. 2.73. Вывод данных из файла с помощью Input

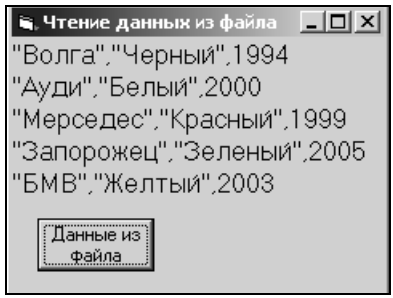


Рис. 2.74. Вывод данных с помощью Line Input

- Чтение с помощью `Line Input`. Отличие состоит в том, что в данном случае выводится сразу целая текстовая строка файла (рис. 2.74).

```
Dim TxtStroka As String
Private Sub Command1_Click()
```

```
Open "car.txt" For Input As #1
Do Until EOF(1)
Line Input #1, Txtstroka
Print TxtStroka
Loop
Close #1
End Sub
```

Это был файл, записанный с помощью способа `Write`. Теперь прочитаем файл, записанный с помощью способа `Print` (рис. 2.75).

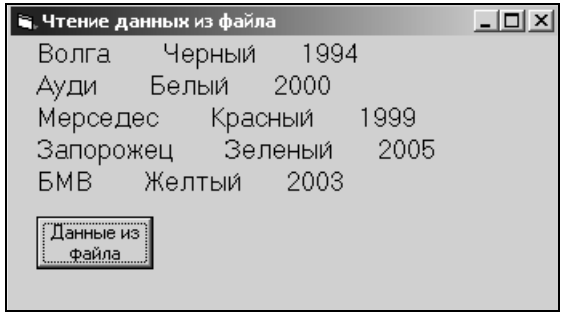


Рис. 2.75. Чтение из файла с помощью Line Input файла, записанного способом Print

Найдите отличия рис. 2.74 от рис. 2.75 и найдите разницу в способах `Write` и `Print`.

Замечание

Способ `Write` удобно применять в тех случаях, когда создаваемый файл будет в дальнейшем использоваться как входной для других программ, а `Print` — когда надо редактировать содержимое файла.

- Чтение с помощью функции `Input`. Производится обычно для чтения всего содержимого файла и помещения его на экранную форму в объект TextBox, которому предварительно задаются свойства `MultiLine` — True и `ScrollBars` — Vertical (рис. 2.76).

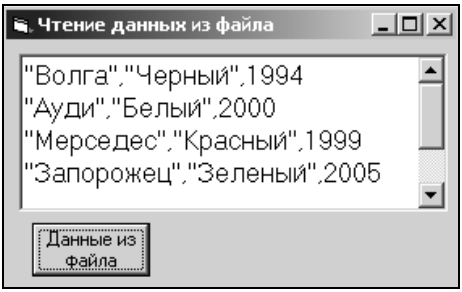


Рис. 2.76. Чтение из файла с помощью функции Input

Используется функция `LOF` (Lehgth Of File), определяющая размер файла в символах.

```
Dim Kolvosymb As Integer
Private Sub Command1_Click()
Open "car.txt" For Input As #1
KolvoSymb=LOF(1)
Text1.Text = Input (KolvoSymb, #1)
Close #1
End Sub
```

2.10.2. Файлы произвольного доступа как базы данных

Файлы произвольного доступа — это провозвестники файлов баз данных. Мы рассмотрим самый простой случай, когда файл содержит одну таблицу. Таблицу сведений о все тех же автомобилях. О каждом автомобиле есть набор сведений. Это будут:

- марка (длина до 20 символов);
- цвет (длина до 12 символов);
- год выпуска (длина 6 символов);
- пробег (длина до 11 символов);
- цена (длина до 11 символов).

Каждое сведение из составляющих набор называется *поле* (всего у нас пять полей), набор всех полей по одному автомобилю называется *запись*. Из таких записей и состоит примитивная

База данных, которую мы создадим. В нашей мини-базе данных будут храниться сведения о 10-ти автомобилях. О каждом автомобиле пять сведений — итого 50. Каждая запись состоит из 60 символов — итого 3000 символов.

Такие таблицы хранят в файлах произвольного доступа.

Главное их отличие от уже рассмотренных нами файлов последовательного доступа состоит в том, что здесь строки имеют фиксированную длину и порядковый номер, то есть каждая запись обладает уникальным (неповторяющимся) номером. Это позволяет быстро находить нужную запись по ее номеру, а также производить над ними операции сортировки, выборки, редактирования.

Для начала нам надо создать свой, пользовательский тип данных:

```
Private Type AboutCars
  Marka As String*20
  Tsvet As String*12
  God As Integer
  Probeg As Long
  Tsena as Long
End Type
```

Далее объявляем переменную `Cars` и определим длину каждой записи.

```
Dim Cars As AboutCars, n As Integer
N=Len(Cars)
```

Откроем файл произвольного доступа:

```
Open "cars.txt" For Random As #1 Len=n
```

В случае файлов произвольного доступа не делается разницы открытия файла для записи или для чтения, они открываются в одном режиме, определяемом ключевым словом `Random`.

Для записи в файл используется оператор

```
Put #ДескрипторФайла, НомерЗаписи, ИмяПеременной
```

Для чтения из файла — оператор

```
Get #ДескрипторФайла, НомерЗаписи, ИмяПеременной,
```

где ИмяПеременной — имя переменной пользовательского типа, значением которой является запись.

Пример заполнения с клавиатуры файла произвольного доступа cars.txt:

```
Private Sub Command1_Click()
Dim Cars As AboutCars, n As Integer
n = Len(Cars)
Open "cars.txt" For Random As #1 Len = n
For i = 1 To n
Cars.Marka =InputBox("Введите название марки _ автомоби-
ля","Ввод данных автомобиля № "&i)
Cars.Tsvet =InputBox("Введите цвет _ автомобиля","Ввод данных
автомобиля № "&i)
Cars.God =InputBox("Введите год выпуска _ автомобиля","Ввод
данных автомобиля № "&i)
Cars.Probeg =InputBox("Введите пробег _ автомобиля","Ввод
данных автомобиля № "&i)
Cars.Tsena =InputBox("Введите цену _ автомобиля","Ввод данных
автомобиля № "&i)
Put #1, i, Cars
Next
Close #1
End Sub
```

А теперь сделаем из уже существующего файла две так называемых выборки — одну по марке и пробегу (рис. 2.77), другую — по марке, году выпуска и цене (рис. 2.78).

```
Private Sub Command2_Click()
Dim Cars As AboutCars, n As Integer
n = Len(Cars)
Open "cars.txt" For Random As #1 Len = n
For i = 1 To 5
Get #1, i, Cars
Print Trim(Cars.Marka), Trim(Cars.Probeg)
Next
Close #1
End Sub
Private Sub Command3_Click()
Dim Cars As AboutCars, n As Integer
```

```
n = Len(Cars)
Open "cars.txt" For Random As #1 Len = n
For i = 1 To 5
Get #1, i, Cars
Print Trim(Cars.Marka), Trim(Cars.God), Trim(Cars.Tsena)
Next
Close #1
End Sub
```

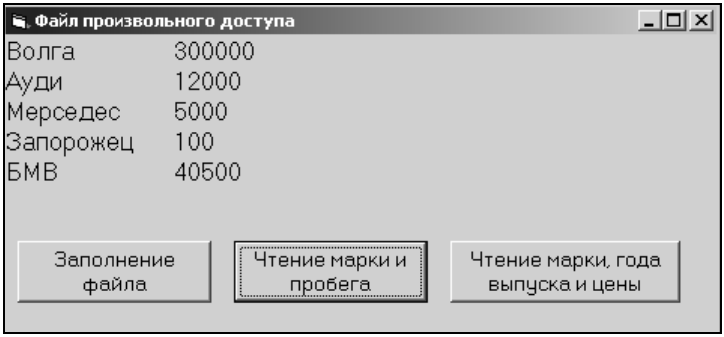


Рис. 2.77. Выборка из файла по марке и пробегу

Рис. 2.78. Выборка из файла по марке, году выпуска и цене

Новая функция `Trim` удаляет лишние пробелы в начале и конце строки символов.

Попробуйте сначала сделать вывод без этой функции, а затем с ней. Сделайте выводы.

Кроме того, можно использовать еще три оператора.

- Переименование файла

```
Name СтароеИмя As НовоеИмя
```

- Копирование файла

```
FileCopy ИмяКопируемогоФайла, ПутьКСоздаваемомуФайлу
```

- Удаление файла

```
Kill ИмяФайла
```

Переходим к выполнению заданий.

305. Школе необходим последовательный файл для учета выпускников.

- Создайте последовательный файл для канцелярии по учету выпускников. Храните в нем фамилию, имя, год выпуска, любимый вид спорта и нынешний род занятий выпускника. Для образца составьте файл на десять человек.
- Воспользуйтесь этим файлом и напечатайте приглашения на очередной домашний матч "Зенита" тем выпускникам, которые назвали футбол своим любимым видом спорта.

306. Компьютерная фирма ведет файл со сведениями о двадцати своих сотрудниках.

- Создайте последовательный файл, содержащий имя и адрес каждого сотрудника (с указанием улицы, дома, квартиры и почтового индекса).
- По содержимому файла напечатайте почтовые адреса для рассылки чеков еженедельной заработной платы.

307. Гидрометцентр ведет статистику выпадения снега по регионам, для каждого из которых заведен последовательный файл. Во всех файлах присутствуют три элемента данных: имя метеоролога, название региона, количество выпавшего за зиму снега в мм.

- Напишите программу ввода данных; заполните файлы для трех регионов.
- Просмотрите все три файла и подсчитайте средний уровень снежных осадков по трем областям. Результат выведите на экран.

308. Налоговая инспекция поощряет налогоплательщиков, вносящих подоходный налог до истечения апрельского контрольного срока, делая им скидку.

- Создайте файл, в котором содержались бы имена, сведения о сроках уплаты и размере налога для каждого налогоплательщика (ограничьтесь группой из шести человек).
- Пусть ваша программа читает файл и делает скидку в 10% для тех, кто уплатил налог досрочно, а также выводит на экран их имена и размер скидки в рублях.

309. Фабрика игрушек ведет учет фирм розничной торговли, сбывающих ее продукцию. Файл контрагентов содержит названия этих фирм, сведения об их местоположении и индекс кредитоспособности: низкая или высокая.

- Напишите программу, которая создала бы последовательный файл контрагентов.
- Напишите программу, которая создала бы два последовательных файла с именами good.dat и bad.dat соответственно для фирм с высокой и низкой кредитоспособностью.
- Пусть ваша программа спрашивает у бухгалтера, какой из двух списков ему представить, а затем выдает названия фирм и их местоположение из соответствующего файла.

310. Предположим, адвокат Михаил Бурщевский с помощью компьютера ведет учет своих клиентов и их дел (табл. 2.1).

- Напишите программу, которая позволяла бы ему вводить в последовательный файл следующие сведения: имя клиента, обвинение, исход дела.

Таблица 2.1. *Исходные данные задачи*

Имя клиента	Обвинение	Исход дела
Сердюков	Клевета	Выиграно
Прохоров	Оскорбление	Проиграно
Мицкевич	Поджог	?????
Максимова	Взлом	Выиграно
Лерман	Взятка	Проиграно

- Мицкевич "из огня попадает в полымя". Напишите программу, которая заменяла бы неопределенное решение суда на "Проиграно".
- Напечатайте обновленный файл.

311. Хоккейные команды "Черные ястребы" и "Красные крылья" хранят в последовательных файлах имена всех своих двенадцати нападающих, число заброшенных ими шайб, сделанных голевых передач и заработанное штрафное время.
    - Создайте файлы black.dat и red.dat, содержащие информацию о каждой из двух команд.
    - Ваша программа по данным, извлеченным из этих файлов, должна создавать новый файл allstars.dat, в котором содержались бы имя, команда и сумма очков (голы и передачи) для шести лучших игроков обеих команд. Пусть имена и показатели результативности хоккеистов выводятся на экран.
312. Имена и адреса всех, кто обращается за информацией в фирму, попадают в список рекламной рассылки.
    - Создайте основной файл master.dat из десяти записей в качестве списка рассылки и меньший файл family.dat из пяти записей для вновь обратившихся с запросами в фирму. Добавьте данные из второго файла в конец первого.
    - Напишите программу, которая случайным образом выбирала бы из основного файла одну запись и посылала бы адресату письмо с уведомлением о выигрыше приза.
313. Инспектор колледжа ведет файл академических занятий студентов.
    - Создайте последовательный файл и заполните его фамилиями, названиями академических курсов и оценочным коэффициентом студентов. Воспользуйтесь данными, перечисленными в табл. 2.2.
    - Выберите "умных" студентов, т. е. тех, кто имеет оценку выше 88, и запишите сведения о них в файл best.dat. Пусть программа помогает инспектору формировать на основе этого файла группы углубленного обучения. По названию курса она должна выдавать список "умных" студентов, зачисленных в такую группу.

Таблица 2.2. *Исходные данные*

Фамилия студента	Курс	Оценочный коэффициент
Югов	Программирование	78
Северов	Японский язык	91
Западов	Психология	56
Востоков	Психология	45
Зюйдов	Корейский язык	89
Вестов	Программирование	66
Полюсов	Психология	90

314. Дешифровальщик. Возьмите в качестве исходного файла последовательного доступа страницу текста вашего любимого писателя. Проведите так называемый частотный анализ, то есть установите, сколько раз каждая буква алфавита встречается в тексте (рис. 2.79).

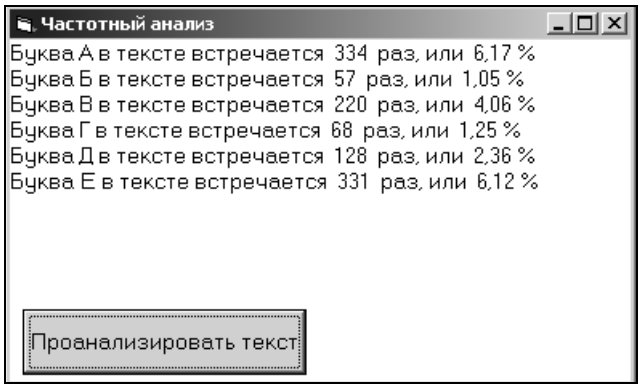


Рис. 2.79. Частотный анализ текста для букв А, Б, В, Г, Д и Е

Для частотного анализа рекомендуется создать файл, содержащий буквы русского алфавита.

Затем попросите своего друга зашифровать какой-нибудь текст так, чтобы каждая буква была заменена какой-нибудь

другой или числом (это, конечно, тоже лучше сделать программно). Используя частотный анализ, попробуйте расшифровать текст.

315. Попробуйте сделать поцесс дешифрации максимально автоматическим — может быть, у вас получится эвристический анализ ☺!

2.11. Создание меню

2.11.1. Создание меню в режиме редактирования

Меню команд формы с помощью редактора меню, который вызывается из меню **Tools** командой **Menu Editor** (рис. 2.80).

Рис. 2.80. Menu Editor (Окно редактора меню)

Свойство `Caption` — это текст заголовка меню, в нашем случае — Игра. Символ "&" перед именем файла определяет клави-

шу доступа (для пункта меню Игра это будет комбинация клавиш <Alt>+<И>).

Так как при создании пунктов меню через Редактор меню каждый из таких пунктов нужно программировать, то надо для каждого пункта меню задать имя в свойтве `Name`. Такие имена принято начинать с буквосочетания mnu и включать ссылку на элемент верхнего уровня.

Кнопки со стрелками влево/вправо определяют подчиненность пунктов меню.

В нашем случае:

```
mnuGame
    mnuGameNew
    mnuGameExit
```

После того как вы создадите все пункты меню и нажмете **OK**, наступит пора программировать события `Click` для каждого пункта.

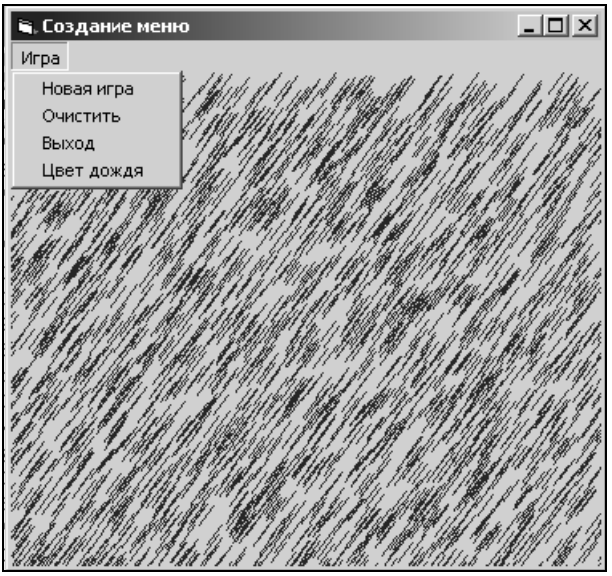


Рис. 2.81. Меню для дождя

Проще всего для пункта Выход — достаточно написать `End` после вызова редактора кода:

```
Private Sub mnuGameExit_Click()
End
End Sub
```

Остальные, впрочем, ненамного сложнее. Вспомним задание 166 про косой дождь. Заставьте его литься по команде Новая игра, пусть меняется его цвет, пусть после дождя можно будет очисть экран (рис. 2.81).

2.11.2. Меню с использованием диалогов

Сначала, тем не менее, создадим простой текстовый редактор и постепенно будем усложнять его, дополняя пунктами меню и панелями инструментов.

Сначала разместим на форме TextBox. Свойство `MultiLine` установим True.

Теперь создадим меню Файл, содержащее команды **Создать**, **Открыть**, **Закрыть**, **Сохранить**, **Сохранить как** и **Выход** (рис. 2.82). И еще пункт меню **Правка** с подменю **Вырезать**, **Копировать**, **Вставить**, **Удалить** (рис. 2.83).

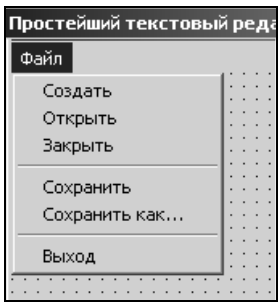


Рис. 2.82. Меню **Файл**

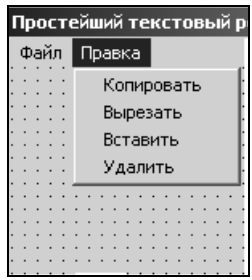


Рис. 2.83. Меню **Правка**

Теперь надо для каждого пункта меню описать событийную процедуру. Сначала в режиме конструирования щелчком мыши создадим пустые заготовки для каждого пункта меню.

Опишем процедуру **Копировать**:

```
Private Sub mnuEditCopy_Click()
Clipboard.Clear 'очистка буфера обмена
'помещение выделенного фрагмента текста в буфер _ обмена
Clipboard.SetText Form1.Text1.SelText
End Sub
```

Опишем процедуру **Вырезать**:

```
Private Sub mnuEditCut_Click()
Clipboard.Clear ' очистка буфера обмена
' помещение выделенного фрагмента текста в буфер _ обмена
Clipboard.SetText Form1.Text1.SelText
'Удаление выделенного текста
Form1.Text1.SelText =""
End Sub
```

Опишем процедуру **Вставить**:

```
Private Sub mnuEditPaste_Click()
'вставка фрагмента текста из буфера обмена
Form1.Text1.SelText = Clipboard.GetText()
End Sub
И, наконец, процедура Удалить.
Private Sub mnuEditDel_Click()
'удаление выделенного фрагмента текста
Form1.Text1.SelText = ""
End Sub
```

Теперь запустите проект, введите в текстовое окно какой-нибудь текст и поредактируйте его с использованием меню **Правка**.

Но для того, чтобы каждый раз не ползать в меню, можно (и нужно!) создать панели инструментов с кнопками, облегчающими нам жизнь!

Эта панель инструментов (**ToolBar**) не входит в стандартный набор **ToolBox**, поэтому нам придется его подключить. Пойдем в меню **Project-Components** и поставим флажок около пункта **Microsoft Windows Common Dialog Control 6.0 (SP6)** (рис. 2.84).

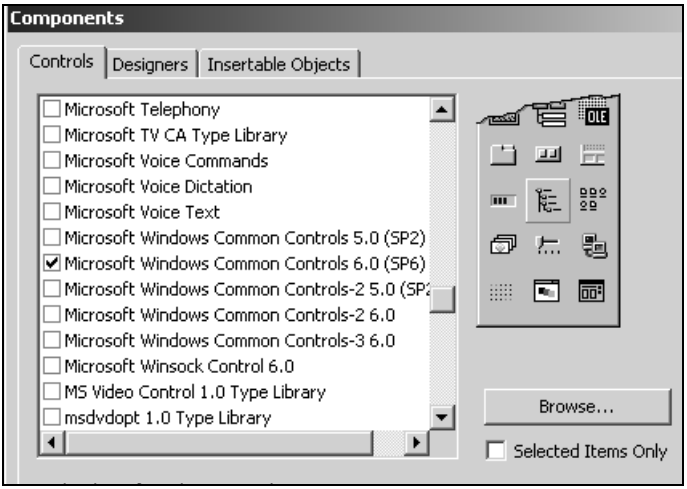


Рис. 2.84. Установка панели инструментов ToolBar

Наш стандартный **ToolBox** пополнился новыми инструментами. Поместим инструмент **ToolBar** на форму. Активизируем его свойство `Custom` так, чтобы появилось диалоговое окно **Property Pages**. Там выберем вкладку **Buttons**. Щелкнем по кнопке **Insert Button**. В поле **Key:** надо ввести имя первой кнопки панели инструментов Copy.

После этого надо повторить процедуру для кнопок **Cut**, **Paste** и **Del**.

Теперь осталось написать событийную процедуру для панели инструментов Правка.

```
Private Sub ToolBar1_ButtonClick(ByVal Button As _
MSComctLib.Button)
Select Case Button.Key
Case Is = "Copy"
Clipboard.Clear
Clipboard.SetText Form1.Text1.SelText
Case Is = "Cut"
Clipboard.Clear
Clipboard.SetText Form1.Text1.SelText
Form1.Text1.SelText = ""
Case Is = "Paste"
```

```
Form1.Text1.SelText = Clipboard.GetText()
Case Is = "Del"
Form1.Text1.SelText = ""
End Select
End Sub
```

Кнопки у нас уже есть, но они какие-то одинаково серые (рис. 2.85).

Надо бы нанести на них подходящие изображение. Начнем! Поместим на форму управляющий элемент ImageList. Также активизируем свойство `Custom` и в появившемся диалоговом окне выберем вкладку **Images**, там щелкнем по кнопке **Insert Picture** и выберем графический файл Copy.bmp для размещения первой кнопки **Copy** (рис. 2.86).

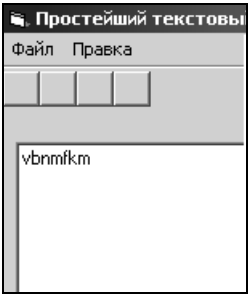


Рис. 2.85. Серые кнопки

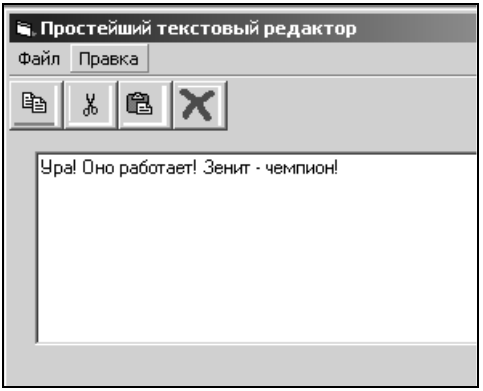


Рис. 2.86. Вот они, кнопочки...

Аналогично выберем остальные графические файлы.

После этого для элемета ToolBar1 активизируем свойство `Custom`, на вкладке **General** в окне **ImageList** выбираем элемент ImageList1 (рис. 2.87).

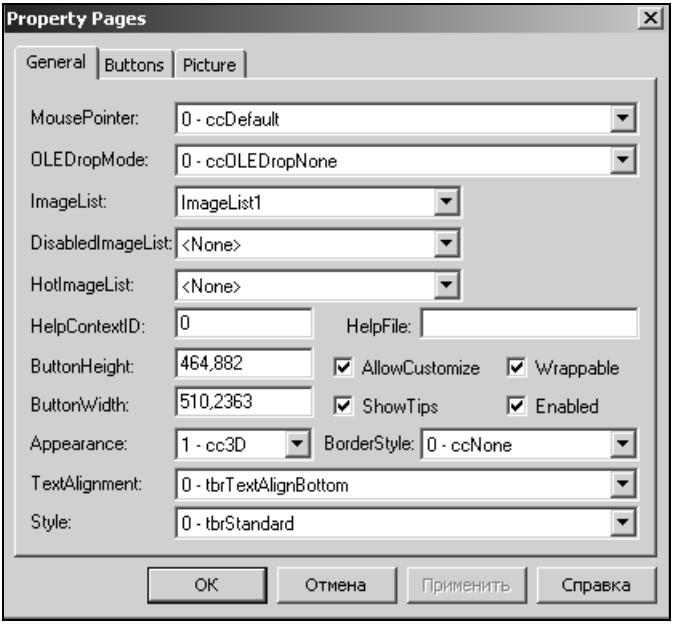


Рис. 2.87. Синхронизация кнопок

На вкладке **Buttons** выставляем значения Index и Image (они должны совпадать). Теперь запустите проект и поработайте. Работает?

Все, казалось бы, хорошо, но нет предела совершенству. Попробуем улучшить наш текстовый редактор. Пусть он еще работает с файлами и форматирует текст. В качестве поля для документа будем использовать вместо простого TextBox элемент RichTextBox. Для этого надо войти в меню **Projects Components**. Отметить там в списке флажком Microsoft RichTextBox Control 6.0. Панель инструментов **ToolBox** дополнится новым элементом. Давайте удалим TextBox с формы и на его месте разместим RichTextBox. Чтобы не менять в проекте имя, дадим новому полю имя Text1.

Для реализации операций над файлами нам нужны еще элементы Общего диалога, позволяющие использовать стандартные диалоговые панели Windows.

Сначала надо добавить инструмент **Common Dialog** в набор **ToolBox**.

В меню **Project** вашего проекта выберите пункт **Components**. В открывшемся списке **Controls** найдите и поставьте флажок около пункта Microsoft Common Dialog Control 6.0, после чего нажмите кнопку **OK**. На **Toolbox** появится новый элемент.

Разместите его на форме, памятуя о том, что в процессе работы приложения он невидим. Дайте ему имя CDlg1.

Теперь для каждого пункта меню Файл необходимо написать событийные процедуры.

Активизируем свойство `Custom` элемента CDlg1. В появившемся окне **Property Pages** выбрать вкладку **open/Save As** и в поле **DialogTitle** ввести имя для окна, которое будет возникать при открытии файла (рис. 2.88).

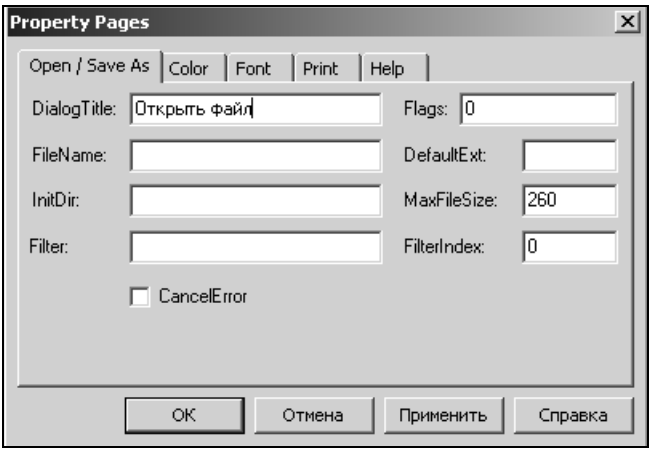


Рис. 2.88. Открытие файла

Теперь напишем программный код с использованием методов `Show/Open` и `LoadFile`.

```
Private Sub mnuFileOpen
CDlg1.ShowOpen
```

```
Text1.LoadFile CdlgFileName
End Sub
```

Пропишем код для команды **Сохранить**:

```
Private Sub mnuFileSave_Click()
CDlg1.ShowSave
If CDlg1.FileName = "" Then
mnuFileSaveAs_Click()
Else
Text1.SaveFile CDlg1.FileName
End If
End Sub
```

А теперь для **Сохранить как**:

```
Private Sub mnuFileSaveAs_Click()
CDlg1.ShowSave
Text1.SaveFile CDlg1.FileName
End Sub
```

Попробуйте теперь набрать что-либо в проекте. Затем сохранить файл, открыть его и сохранить под новым именем.

Теперь создадим процедуру форматирования шрифта.

Для этого сначала создайте структуру меню:

```
Формат
    Шрифт
    Цвет
        Шрифт
        Фон
```

После этого выберем свойство `Custom` объекта CDlg1 и затем вкладку **Font**. В полях **FontName**, **FontSize**, **Min** и **Max** следует ввести параметры шрифта, которые будут использоваться по умолчанию. В поле **Flags** установить значение 2, определяющее тип шрифтов (рис. 2.89).

Событийная процедура для форматирования шрифта:

```
Private Sub mnuFont_Click()
CDlg1.ShowFont
Text1.Font.Size = CDlg1.FontSize
```

```
Text1.Font.Name = CDlg1.FontName
Text1.Font.Bold = CDlg1.FontBold
Text1.Font.Italic = CDlg1.FontItalic
Text1.Font.Underline = CDlg1.FontUnderline
End Sub
```

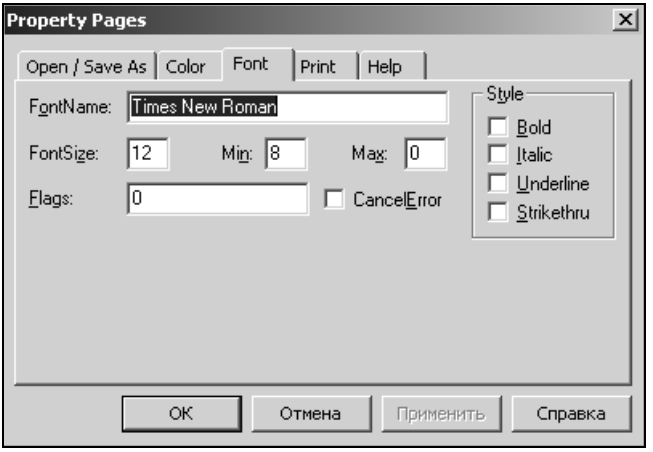


Рис. 2.89. Параметры шрифта по умолчанию

Запустите проект и проверьте работу меню **Формат | Шрифт**.

Теперь разберемся с цветом. Выберем свойство `Custom` объекта CDlg1, а затем вкладку **Color**. В поле **Color** ставим 0 (черный цвет), а в поле **Flags** значение 2.

Программный код для цвета шрифта.

```
Private Sub mnuFontColor_Click()
CDlg1.ShowColor
Text1.SelColor = CDlg1.Color
End Sub
```

Программный код для цвета фона.

```
Private Sub mnuBackColor_Click()
CDlg1.ShowColor
Text1.BackColor = CDlg1.Color
End Sub
```

Вот у нас и получился довольно неплохой текстовый редактор.

А теперь самостоятельно выполните еще два задания.

316. Усовершенствовать текстовый редактор панелью инструментов для работы с файлом — **Открыть**, **Сохранить** и **Выход**.

317. Дополнить текстовый редактор процедурами поиска и замены фрагментов текста.

2.12. Объект управления *TabStrip*

Объект **TabStrip** (Полоса вкладок) позволяет на одной экранной форме разместить несколько страниц-вкладок. Примеры таких форм вы могли видеть, например, в Word при выборе параметров форматирования шрифта или настроек абзаца.

Традиционно пойдем в меню **Projects Components** и отметим в списке флагом Microsoft Windows Common Control 6.0 (SP 6). В **ToolBox** появятся новые интсрументы. Найдем там **TabStrip** и растянем с его помощью на форме объект **Полоса вкладок** (рис. 2.90).

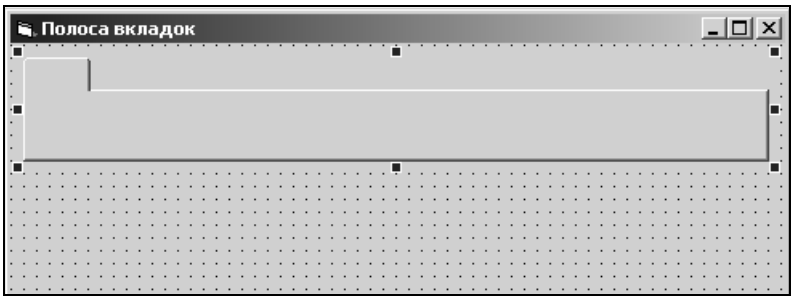


Рис. 2.90. Полоса вкладок

Активизируем свойство `Custom` для открытия окна **Property Pages**. Пусть наши закладки будут носить названия легендарных российских рок-групп. На вкладке **Tabs** в поле **Caption** введем название группы и нажмем **Insert Tab** и так далее. Лишние вкладки после введения всех названий удалим кнопкой **Remove Tab**.

После этого разместим на **TabStrip** TextBox, куда занесем сведения о лидере группы. Ниже, вне **TabStrip**, разместим еще четыре Text-

Box с записями о лидерах остальных групп. Имена всем TextBox присвоим одинаковые — Text. Таким образом создастся массив.

Теперь напишем программный код, который будет размещать все TextBox по образцу первого и даст возможность открывать вкладки, не перекрывая друг друга.

```
Dim I As Integer
Private Sub Form_Load()
For I = 1 To 4
Text(I).Top = Text(0).Top
Text(I).Left = Text(0).Left
Text(I).Height = Text(0).Height
Text(I).Width = Text(0).Width
Text(I).FontSize = Text(0).FontSize
Next
Text(0).ZOrder 0
End Sub

Private Sub TabStrip1_Click()
I = TabStrip1.SelectedItem.Index
Text(I - 1).ZOrder 0
End Sub
```

Метод `ZOrder` со значением, равным 0, обсепечит неперекрываемость объектов друг другом.

Получилось очень даже неплохо (рис. 2.91).

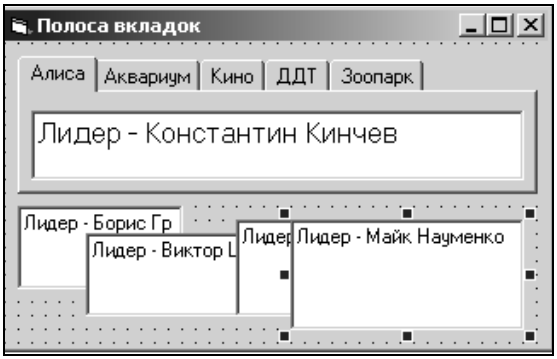


Рис. 2.91. Полоса вкладок

2.13. Объект *Status Bar*

Объект **Status Bar** (Строка состояния) очень полезен, так как показывает пользователю то, что происходит в работающем проекте.

Обычно строка состояния располагается внизу экранной формы. Строка может состоять из нескольких панелей.

Инструмент **Status Bar** появляется оттуда же, откуда и **TabStrip**. Берем его и растягиваем на форме.

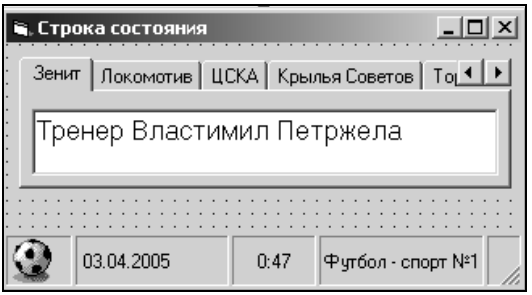


Рис. 2.92. Строка состояния

Активизируем свойство `Custom`, в открывшемся окне **Property Pages** выбираем вкладку **Panels** и задаем там необходимые свойства так, чтобы получилось, как на рис. 2.92.

Рисунок вставляется из имеющихся на вашем компьютере файлов с расширением ico.

2.14. Объект *Progress Bar*

Если программные действия выполняются долго, то, чтобы пользователю было понятно, что процесс все-таки идет, в программный код помещается строка, управляющая элементом **ProgressBar** (Строка процесса). На экранную форму объект **ProgressBar** размещается аналогично **TabStrip** и **StatusBar**. Вы должны установить три свойства этого объекта: `Min`, `Max` и `Value`. `Min` и `Max` в окне **Properties** объекта, а свойство `Value` прописываем в программном коде.

Покажем работу **ProgressBar** на примере цикла `For`, внутри которого происходит много действий, выполняющихся довольно долго (рис. 2.93).

```
For i=1 to 1000000
ProgressBar1.Value=i/1000
Next
```

Вполните следующие задания.

318. Создайте проект "Зоопарк", размещающий на одной форме десять вкладок с изображениями разных животных и краткими сведениями о них.

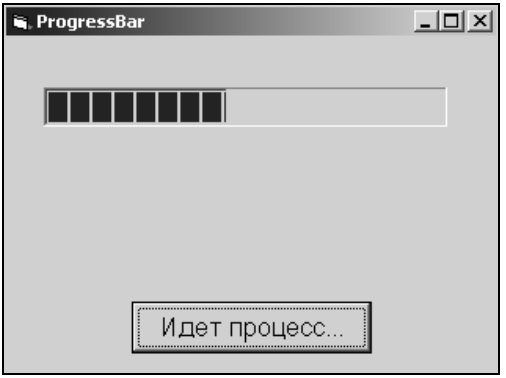


Рис. 2.93. Процесс идет...

319. Дополните проект из задания 301 строкой состояния, содержащей панели даты, иконки и текстового поля, изменяющего текст в зависимсоти от выбранной пользователем вкладки.

320. Дополните проект "Пожиратели звезд" (задание 202) строкой выполнения процесса.

2.15. Мультимедиа

Visual Basic поддерживает основные форматы мультимедийных файлов, таких как: avi, mpeg, mid, wav. Для управления устройствами мультимедиа в Visual Basic применяется специальный интерфейс MCI (Multimedia Control Interface).

Рассмотрим пару примеров его использования.

2.15.1. Проигрыватель WAV файлов

Создадим новый проект, назовем который WAV_player (меню **Project | Project1 | Properties**).

Свойству `Caption` формы придадим такой же заголовок.

Присоединим к проекту набор компонентов Microsoft Multimedia Control 6.0 (как всегда — из меню **Project Components** или щелкнув правой кнопкой мыши по **ToolBox** и выбрав **Components**).

Добавим элемент **CommonDialog** так, как мы делали в *разделе 2.11.*

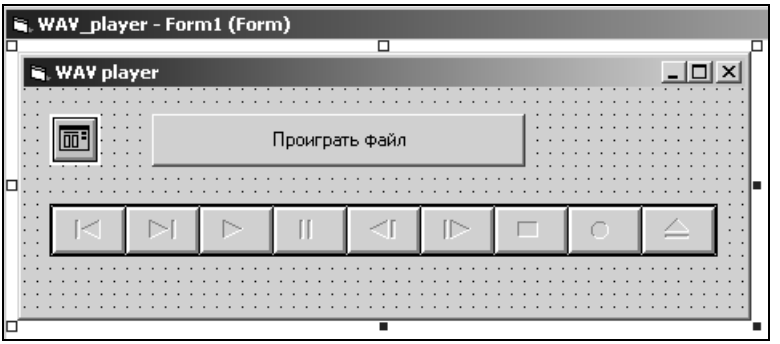


Рис. 2.94. Форма для WAV-проигрывателя

Добавим на форму элементы **MMControl** и **ConmmonDialog** (которую назовем CDlg1). Наконец добавим на форму **CommandButton** (дав ей имя CmdFind). Должно получиться, как на рис. 2.94.

Напишем теперь следующий программный код:

```
Private Sub CmdFind_Click()
     CDlg1.ShowOpen
     MMControl1.FileName = CDlg1.FileName
     MMControl1.Command = "open"
End Sub

Private Sub Form_Load()
     MMControl1.Notify = False
```

```
     MMControl1.Wait = True
     MMControl1.Shareable = False
     MMControl1.DeviceType = "WaveAudio"
End Sub
Private Sub Form_Unload(Cancel As Integer)
     MMControl1.Command = "Close"
End Sub
```

Запустите проект, нажмите кнопку **Проиграть файл**, найдите в открывшемся окне любой wav-файл на вашем компьютере и программируйте под музыку!

2.15.2. Проигрыватель AVI-файлов

При помощи предыдущего проекта можно воспроизводить и видеофайлы в формате AVI. Для этого всего-навсего потребуется назначить другой тип устройства для элемента управления MMControl1.

Изменим командный код на следующий:

```
Private Sub CmdFind_Click()
     CDlg1.ShowOpen
     MMControl1.FileName = CDlg1.FileName
     MMControl1.Command = "open"
End Sub

Private Sub Form_Load()
     MMControl1.Notify = False
     MMControl1.Wait = True
     MMControl1.Shareable = False
     MMControl1.DeviceType = "AVIVideo"
End Sub
Private Sub Form_Unload(Cancel As Integer)
     MMControl1.Command = "Close"
End Sub
```

Посмотрите, как "оно" работает (рис. 2.95).

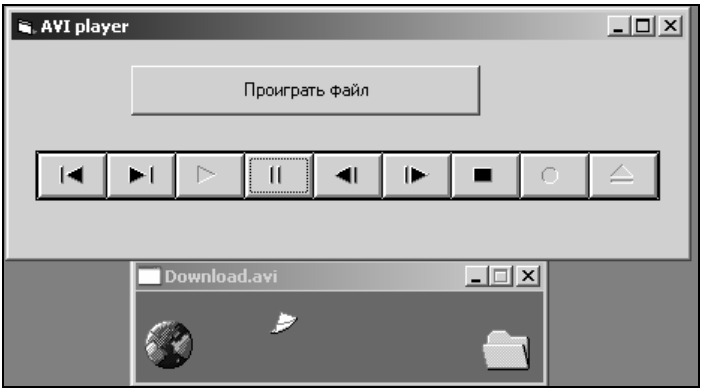


Рис. 2.95. AVI-проигрыватель

Изображение из файла выводится в отдельное окно, но ведь иногда хочется прямо на форме ☺. Как быть? Для этого в проект надо добавить объект для вывода изображения и назначить направление вывода в этот объект. Попробуем в качестве такого объекта использовать PictureBox.

Добавим на форму объект PictureBox (переименуем его в P1). Изменим предыдущий командный код следующим образом:

```
Private Sub CmdFind_Click()
     CDlg1.ShowOpen
     MMControl1.FileName = CDlg1.FileName
     MMControl1.Command = "open"
     MMControl1.hWndDisplay=P1.hWnd
     Form1.P1.SetFocus
End Sub

Private Sub Form_Load()
     MMControl1.Notify = False
     MMControl1.Wait = True
     MMControl1.Shareable = False
     MMControl1.DeviceType = "WaveAudio"

     MMControl1.DeviceType = "AVIVideo"
```

```
End Sub
Private Sub Form_Unload(Cancel As Integer)
     MMControl1.Command = "Close"
End Sub
```

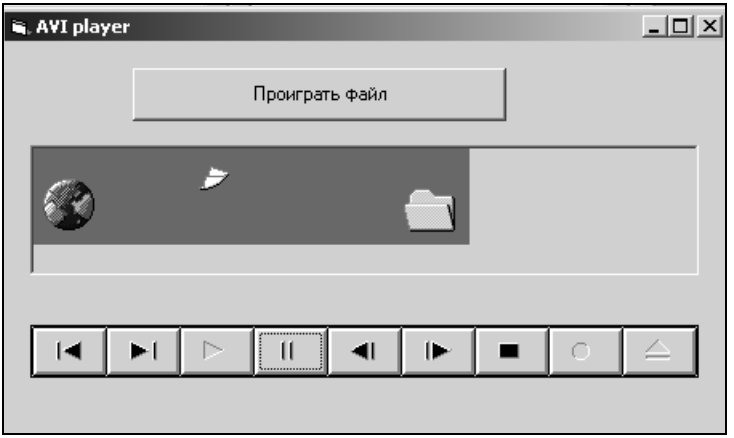


Рис. 2.96. AVI-проигрыватель с выводом изображения на форму

Запустите проект и убедитесь, что он выглядит, как на рис. 2.96.

2.15.3. Просмотр видеофайлов при помощи элемента управления Animation

Вот еще один способ просмотра avi-файлов.

Нам понадобится разместить на форме элемент Animation (дадим ему свойство `Name` – Anim1), который появится после выбора в меню **Projects Components** пункта Microsoft Windows Common Control-2 6.0. Ну, и кроме того — CommandDialog (**Projects Components** пункт Microsoft Common Dialog Control 6.0). Дадим ему имя CDlg1. Разместим на форме также две командных кнопки: **Начать просмотр** и **Закончить** (назовем их соответственно cmdBegin и cmdEnd). Форма должна выглядеть примерно, как на рис. 2.97.

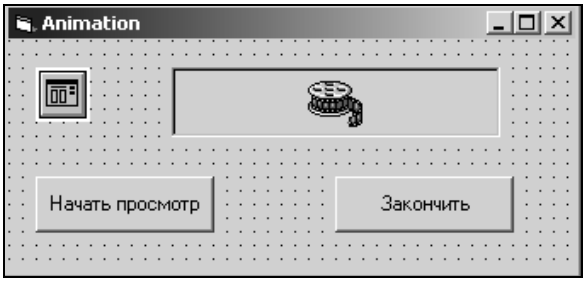


Рис. 2.97. Форма с элементом Animation

Напишем командный код:

```
Private Sub CmdBegin_Click()
     CDlg1.Filter = "avi.files(*.avi)|*.avi"
     CDlg1.ShowOpen
     Anim1.Open CDlg1.FileName
     Anim1.Play
End Sub
Private Sub CmdEnd_Click()
     Anim1.Stop
End Sub
Private Sub Form_Unload(Cancel As Integer)
     Anim1.Close
End Sub
```

Запустим проект и увидим (рис. 2.98).

Рис. 2.98. Animation в действии

2.16. Создание тестовых систем для игровых форм контроля знаний

Я уже много лет занимаюсь интеллектуальными играми типа "Что? Где? Когда?" и "Брэйн-ринг". На больших турнирах, где присутствует большое количество знатоков, в почете интеллектуальные всяческие развлечения. Одно из них —"эрудит-лото" — приписывается Борису Бурде. Оно произошло от известной системы тестирования знаний — вопрос и четыре варианта ответов. Проводится эрудит-лото обычно на карточках. Вопросов имеется от 8 до 10. Реализуем подобную систему в Visual Basic на примере одного тематического эрудит-лото, посвященного парадоксальным, но не отмененным по сей день законам различных штатов США. Вы, конечно, можете использовать эту систему для разработки тестов контроля знаний учащихся по различным предметам и областям знаний.

Проект будет состоять из трех форм.

Замечание

Сразу оговорюсь, что проект не идеален, более того, после его осуществления я даю своим учащимся целый урок на его тестирование, на выявление так называемых багов (от англ. Bug – жучок) – логических ошибок в программе, неувязок в переходах и т. д. Порядка 10 штук можно найти. Но найти мало – надо еще и исправить!

Первая форма (frmBegin) представлена на рис. 2.99.

На форме разместим два объекта Label (Label1 и Label2): TextBox (`txtName`) — для имени участника и CommandButton (`cmdRead`) для перехода к чтению инструкции. Приукрасим форму фоновым рисунком и каким-нибудь оживляющим рисунком на тему.

Добавим в проект вторую форму (меню **Project Add Form**). Назовем ее frmInstr. Выглядеть она будет вот так (рис. 2.100).

На ней размещен один большой Label со свойством `Caption`, содержащим, собственно, текст короткой инструкции и

CommandButton (`cmdInstr`), которая позволит перейти непосредственно к тестированию.

Добавим третью форму (`frmTest`), необходимую для тестирования (рис. 2.101).

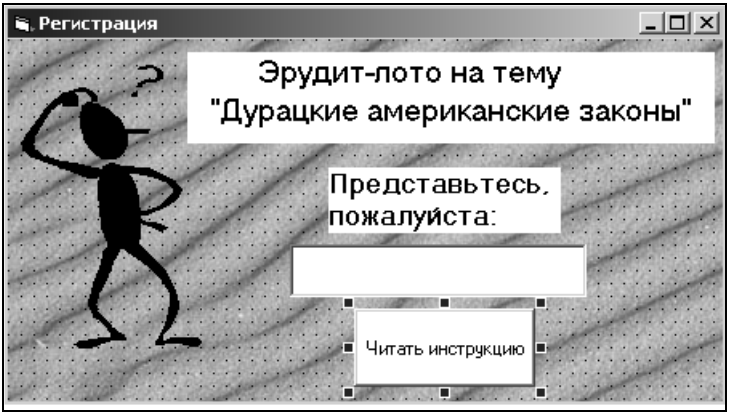


Рис. 2.99. Форма регистрации участника

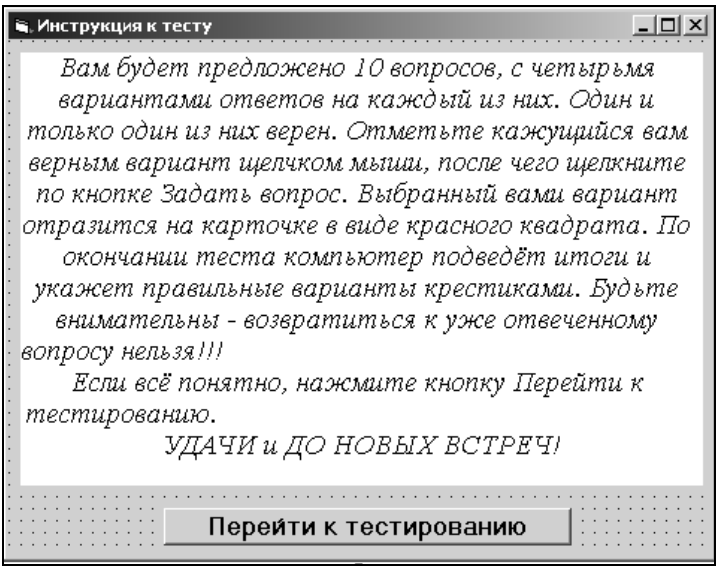


Рис. 2.100. Форма с инструкцией

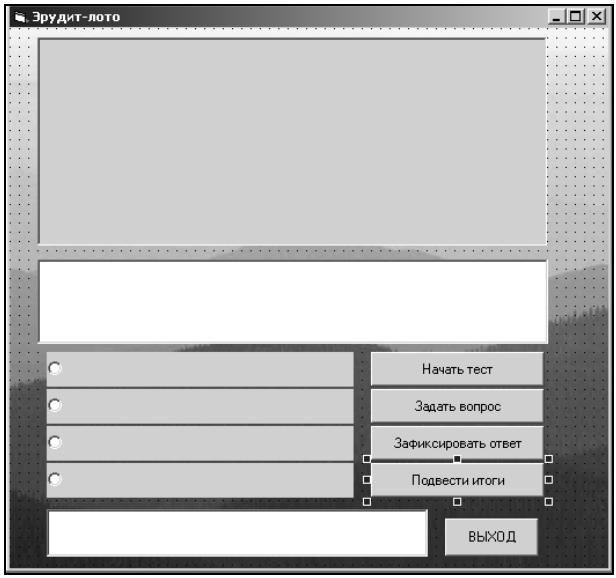


Рис. 2.101. Форма тестирования

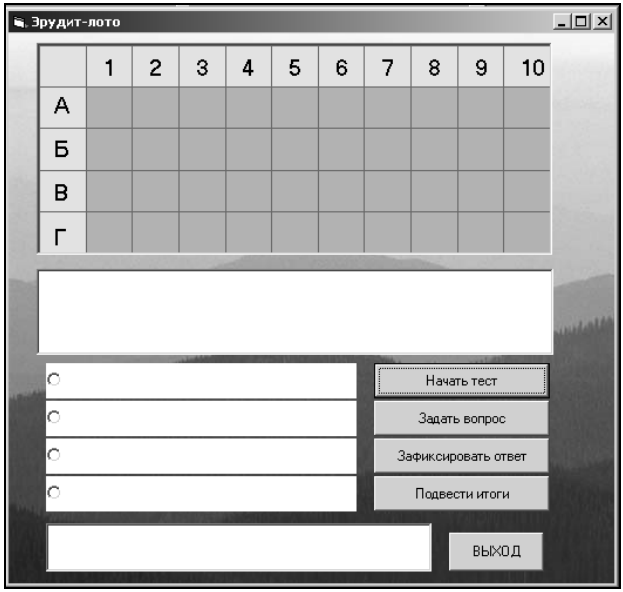


Рис. 2.102. Форма после начала тестирования

Сверху на ней расположен PictureBox (`P1`) для размещения изображения карточки эрудит-лото. Ниже — TextBox (`Text1`) для вывода текста вопроса. Слева (в нижней части формы) — четыре OptionButton (`Op1`, `Op2`, `Op3`, `Op4`) для вывода вариантов ответов. Справа от них размещены командные кнопки: **Начать тест** (`cmdBegin`) предназначенная для прорисовки карточки и подготовки к работе, **Задать вопрос** (`cmdQuest`) — для вывода очередного вопроса и вариантов ответов, **Зафиксировать ответ** (`cmdFix`) — для фиксации ответа (после нажатия этой кнопки нельзя исправить предыдущий ответ), **Подвести итоги** (`cmdItog`) — для подведения итогов теста и **Выход** (`cmdEnd`) — для окончания работы. Внизу помещен PictureBox (`P2`), который будет выводить сообщения об итогах теста.

После нажатия командной кнопки **Начать тест** форма примет следующий вид (рис. 2.102).

Теперь необходимо подготовить текстовый файл произвольного доступа с текстом всех вопросов, вариантов ответов и указанием на единственный в каждом вопросе правильный вариант ответа.

Вот текст для файла моего примера.

1. В штате Мэн мужчинам запрещено щекотать подбородок женщины:

   А) гусиным пером

   Б) лебединым пером

   В) куриным пером

   Г) тупым топором

2. В штате Кентукки запрещено жениться на:

   А) дочери своей жены

   Б) бабушке своей жены

   В) сестре своей жены

   Г) внучатой племяннице жены младшего брата

3. В том же Кентукки женщине запрещено разгуливать вдоль хайвея в купальнике только, если она не вооружена:

   А) винчестером

   Б) плетью

В) бейсбольной битой

Г) кухонным ножом

4. В Лос-Анджелесе мужчина вправе наказать жену кожаным ремнем, ширина которого не должна превышать:

   А) 3,5 дюймов

   Б) 3 дюймов

   В) 2,5 дюймов

   Г) 2 дюймов

5. В графстве Лоутон, штат Оклахома, категорически возбраняется садиться на колени мужчин, если на них не лежит:

   А) подушка

   Б) одеяло

   В) книга

   Г) газета

6. В Бостоне владелец гостиницы на законном основании мог выселить постояльца, если тот не принял ванну в течение:

   А) трех дней

   Б) недели

   В) месяца

   Г) года

7. В городе Лесингтон (штат Кентукки) запрещено класть мороженое:

   А) в капюшон

   Б) в нагрудный карман рубашки

   В) в задний карман брюк

   Г) во внутренний карман смокинга

8. В штате Нью-Йорк, проезжая в трамвае, нельзя охотиться на:

   А) кошек

   Б) собак

   В) волнистых попугайчиков

   Г) кроликов

9. В штате Флорида нельзя привязывать к пожарному гидранту домашнего:

   А) крокодила

   Б) слона

   В) страуса

   Г) койота

10. В столице Америки Вашингтоне запрещено бить быка по:

    А) рогам

    Б) морде

    В) передним ногам

    Г) задним ногам

Файл подготовим с помощью отдельного проекта. Его программный код:

```
Private Type AboutQ
  quest As String * 255
  answera As String * 50
  answerb As String * 50
  answerc As String * 50
  answerd As String * 50
  answert As String * 1
End Type
Private Sub Command1_Click()
Dim Q As AboutQ, n As Integer
  n = Len(Q)
  Open "C:\q.txt" For Random As #1 Len = n
  For i = 1 To 10
Q.quest = InputBox("Введите вопрос", _
"Ввод данных  " & i)
Q.answera = InputBox("Введите ответ А ", _
"Ввод данных по вопросу № " & i)
Q.answerb = InputBox("Введите ответ Б ", _
"Ввод данных по вопросу № " & i)
```

```
Q.answerc = InputBox("Введите ответ В ", _
"Ввод данных по вопросу № " & i)
Q.answerd = InputBox("Введите ответ Г ", _
"Ввод данных по вопросу № " & i)
Q.answert = InputBox("Введите букву правильного _
 ответа ", "Ввод данных по вопросу № " & i)
Put #1, i, Q
Next
Close #1
End Sub
```

Данные записываются в файл q.txt, откуда и будут считываться во время тестирования.

Теперь программный код для основного проекта:

```
Dim I As Integer
Dim Im As String
' Объявление массива ответов, данных пользователем
Dim a(1 To 10) As String
' Объявление массива правильных ответов
Dim t(1 To 10) As String
' Переменная, накапливающая количество правильных ответов
Dim k As Integer
' Описание пользовательского типа данных
Private Type AboutQ
  quest As String * 255
  answera As String * 50
  answerb As String * 50
  answerc As String * 50
  answerd As String * 50
  answert As String * 1
End Type
' Загрузка формы регистрации
Private Sub Form_Load()
Load FrmBegin
FrmBegin.Visible = True
```

```
Im = FrmBegin.TxtName.Text
End Sub

'Процедура чтения вопроса
Sub vopros(I As Integer)
Dim Q As AboutQ, n As Integer
n = Len(Q)
Open "C:\q.txt" For Random As #1 Len = n
If EOF(1) Then End
Get #1, I, Q
Text1.Text = Trim(Q.quest)
Op1.Caption = Trim(Q.answera)
Op2.Caption = Trim(Q.answerb)
Op3.Caption = Trim(Q.answerc)
Op4.Caption = Trim(Q.answerd)
t(I) = Q.answert
Close #1
End Sub
'Вызов очередного вопроса с вариантами ответов
Private Sub CmdQuest_Click()
I = I + 1
Call vopros(I)
' Запрещение нажатия кнопки Задать вопрос до выбора и фиксации
' ответа
Cmdquest.Enabled = False
End Sub

'Прорисовка формы теста (карточки эрудит-лото)
Private Sub CmdBegin_Click()
P1.BackColor = vbCyan
P1.Scale (0, 180)-(440, 0)
P1.Line (0, 144)-(440, 180), vbYellow, BF
P1.Line (0, 0)-(40, 180), vbYellow, BF

For x = 0 To 400 Step 40
P1.Line (x, 0)-(x, 180), vbMagenta
```

```
Next
For y = 0 To 180 Step 36
P1.Line (0, y)-(440, y), vbMagenta
Next
P1.FontSize = 14
x = 50
For I = 1 To 10
P1.PSet (x, 173), vbYellow
P1.Print I
x = x + 40
Next
P1.PSet (13, 140), vbYellow: P1.Print "À"
P1.PSet (13, 102), vbYellow: P1.Print "Á"
P1.PSet (13, 64), vbYellow: P1.Print "Â"
P1.PSet (13, 26), vbYellow: P1.Print "Ã"
Op1.BackColor = vbWhite
Op2.BackColor = vbWhite
Op3.BackColor = vbWhite
Op4.BackColor = vbWhite
I = 0: k = 0
P2.Print Im
End Sub
'Фиксирование ответа пользователя
Private Sub CmdFix_Click()
  Call zakraska(I)
  If a(I) = t(I) Then k = k + 1
'  разрешение нажатия кнопки Задать вопрос
  Cmdquest.Enabled = True
End Sub
'Выбор ответа пользователя
Private Sub Op1_Click()
 a(I) = "À"
End Sub
Private Sub Op2_Click()
 a(I) = "Á"
End Sub
```

```
Private Sub Op3_Click()
 a(I) = "Â"
End Sub
Private Sub Op4_Click()
 a(I) = "Ã"
End Sub
'Процедура указания на карточке ответа пользователя
Sub zakraska(I As Integer)
'Ответ А
If a(I) = "А" Then
P1.Line (I * 40, 108)-(I * 40 + 40, 144), vbRed, BF
End If
'Ответ Б
If a(I) = "Б" Then
P1.Line (I * 40, 72)-(I * 40 + 40, 108), vbRed, BF
End If
'Ответ В
If a(I) = "В" Then
P1.Line (I * 40, 36)-(I * 40 + 40, 72), vbRed, BF
End If
'Ответ Г
If a(I) = "Г" Then
P1.Line (I * 40, 0)-(I * 40 + 40, 36), vbRed, BF
End If
End Sub
'Указание на карточке правильных ответов
Sub Prav(I As Integer)
  P1.DrawWidth = 3
For I = 1 To 2
'Ответ А
If t(I) = "А" Then
  P1.Line (I * 40, 108)-(I * 40 + 40, 144)
  P1.Line (I * 40, 144)-(I * 40 + 40, 108)
End If
'Ответ Б
If t(I) = "Б" Then
```

```
  P1.Line (I * 40, 72)-(I * 40 + 40, 108)
  P1.Line (I * 40, 108)-(I * 40 + 40, 72)
End If
'Ответ Б
If t(I) = "Б" Then
  P1.Line (I * 40, 36)-(I * 40 + 40, 72)
  P1.Line (I * 40, 72)-(I * 40 + 40, 36)
End If
'Ответ Г
    If t(I) = "Г" Then
    P1.Line (I * 40, 0)-(I * 40 + 40, 36)
    P1.Line (I * 40, 36)-(I * 40 + 40, 0)
    End If
   Next
  P1.DrawWidth = 1
End Sub
'Подведение итогов
Private Sub CmdItog_Click()
  P2.FontSize = 12
  Call Prav(I)
  P2.Print "Количество ваших правильных ответов"; k
End Sub
' Процедура выхода
Private Sub CmdEnd_Click()
  End
End Sub
```

После запуска проекта может получиться примерно, как на рис. 2.103.

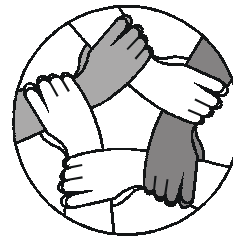
А теперь самостоятельные задания.

321. Разработайте свои эрудит-лото на интересующие вас темы.
322. Дополните проект "Эрудит-лото" возможностью выбора из нескольких тем.
323. Дополните проект "Эрудит-лото" большим набором разнообразных высказываний по ходу тестирования, обращенных к пользователю.

324. Разработайте возможность вывода на печать теста и результатов пользователя.

325. Дополните проект "Эрудит-лото" мультимедийными эффектами — сначала просто звуковыми файлами, а потом (чем черт не шутит), может, и видеофайлами, да так, чтобы мы видели автора (или авторов) "Эрудит-лото", задающего вопрос.

Рис. 2.103. Игра "Эрудит-лото"



Глава 3

Десерт

Ну, а на десерт полагается что-нибудь вкусненькое. И я с удовольствием представляю вам разнообразные исходники всяких интересных программок, которыми вы можете воспользоваться и, надеюсь, улучшить их и разработать кучу своих. Вперед!

3.1. Как написать игрушку на VB

Прежде чем вы начнете писать игру, воспользовавшись нижеприведенной статьей, я бы хотел выразить огромную признательность за предоставленные материалы Михаила Эскина (**http://mik-seite.narod.ru**).

Итак, игра от Михаила Эскина!

Каждый программист в своей жизни написал хотя бы одну игрушку. Не являются исключением из этого правила и те, кто пишет на Visual Basic. Другой разговор, что более ранние версии Visual Basic были более громоздкими и достаточно медленными. Относительно медлительными были и сами компьютеры. С годами постепенно положение изменилось. И хотя Visual Basic стал еще более громоздким, однако скорость работы, особенно с графикой, выросла значительно.

В данном разделе я хочу показать, что на обычном Visual Basic 6 можно писать интересные приложения. А для тех, кто посвятит себя в дальнейшем программированию игр, это будет начальным трамплином. Итак, шаг за шагом мы создадим компьютерную игру. Для тех, кто спешит, есть готовый листинг.

После каждого шага, нажав клавишу <F5>, вы сможете увидеть те изменения, которые Вы внесли в программу. Для того чтобы случайно не потерять свои данные, рекомендую вам производить сохранения проекта после каждого шага.

Шаг 1. Самый главный. Необходимо определить, что же мы хотим "представить миру". Именно на этапе планирования и определяется будущий вид, структура и взаимодействие различных частей программы.

Те, кто постарше, наверно помнят то время, когда персональные компьютеры были недосягаемой мечтой. Именно тогда появились в России первые игровые автоматы, на которых за 15 копеек можно было попускать торпеды в корабли, поучаствовать в скачках, ну и, конечно, порулить. Отдавая дань памяти тем временам, я решил предложить вам написать совместно игровую программу AutoRoad. Она будет состоять из двух форм (основной, содержащей игровое поле, и вспомогательной для вывода пользовательских настроек), модуля, а также ActivX Control'a. Хочу сразу же обратить ваше внимание на то, что в ряде случаев, для ускорения обработки графики, мы будем обращаться к API-функциям. Всем известно (а кому не известно — "мотайте на ус"), что API-функции работают только с пикселами. Поэтому на основной форме (где будет находиться игровое поле) и сама форма и все элементы, расположенные на ней, должны будут иметь свойство `ScaleMode` = 3 (pixel).

Шаг 2. Создайте новый проект StandartEXE и дайте ему имя AutoRoad. Для формы измените следующие параметры:

`Name` — frmMain,

`Caption` — AutoRoad,

`Icon` — иконку можете вставить мою или какая вам больше понравится,

`ScaleMode` — 3 – Pixel,

`StartupPosition` — 2 – CenterScreen.

Разместите на форме 2 PicturesBox. Первый с параметрами:

`Name` — picRoad,

`AutoRedraw` — True,

`BorderStyle` — 0 – None,

`Height` — 461,

`Left` — 8,

`ScaleMode` — 3 – Pixel,

`Top` — 0,

`Width` — 331.

Второй PictureBox с параметрами:

`Name` — picSrc,

`AutoRedraw` — True,

`AutoSize` — True,

`BorderStyle` — 0 – None,

`Left` — 368,

`Picture` — загрузите Auto.bmp,

`ScaleMode` — 3 – Pixel,

`Top` — 364,

`Visible` — False.

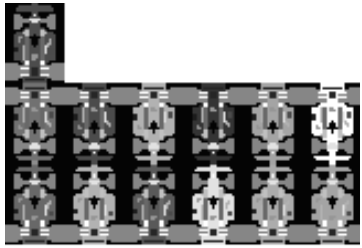


Рис. 3.1. Машинки

Замечание

На случай создания своих картинок, Auto.bmp построена следующим образом (рис. 3.1): в первом ряду — управляемое игроком авто, во втором — 6 авто, двигающихся навстречу, в третьем — 6 авто по ходу движения с основным автомобилем. Размеры каждого авто 23 пиксела в ширину и 31 пиксел в высоту. Если вы захотите нарисовать автомобили других размеров, необходимо будет сделать соответствующие изменения в кодах.

Лирическое отступление 1

Если вы впоследствии переименуете проект и даже сохраните его под другим именем, все равно Visual Basic будет предлагать компилировать его в exe-файл только с первоначальным именем. Закройте проект, а затем в каком-либо текстовом редакторе откройте файл вашего проекта с расширением vbp, и в строке ExeName32 впишите то название, какое вы хотите увидеть.

Шаг 3. Пошли дальше. Нарисуем дорогу. В `frmMain` в разделе деклараций запишем процедуру, которая будет рисовать игровое поле:

```
Public Sub DrawRoad()
picRoad.Line (0, 0)-(picRoad.Width * 0.05, picRoad.Height), _
vbGreen, BF
picRoad.Line (picRoad.Width * 0.95, 0)- (picRoad.Width, _
picRoad.Height), vbGreen, BF
picRoad.Line (picRoad.Width * 0.05, 0)- (picRoad.Width _
* 0.95, picRoad.Height), vbBlack, BF
picRoad.Line (picRoad.Width * 0.5, 0)- (picRoad.Width _
* 0.5,picRoad.Height), vbYellow, BF
End Sub
```

Первая и вторая строки рисуют зеленый бордюр по краям, третья строка — непосредственно саму дорогу, и наконец четвертая — желтую разделительную линию.

Шаг 4. Займемся теперь нашим авто. Для копирования его на игровое поле будем использовать API-функцию `StretchBlt`.

Вначале добавим к программе модуль Project IAdd Module, и назовем его `Name = mdlAuto`. В раздел деклараций модуля запишем объявление данной функции:

```
Public Declare Function StretchBlt Lib "gdi32" ( _
ByVal hdc As Long, ByVal x As Long, _
ByVal y As Long, _
ByVal nWidth As Long, ByVal nHeight As Long, _
ByVal hSrcDC As Long, ByVal xSrc As Long, _
ByVal ySrc As Long, _
ByVal nSrcWidth As Long, ByVal nSrcHeight _
As Long, ByVal dwRop As Long) As Long
```

а также константу, необходимую для последнего параметра:

```
Public Const SRCCOPY = &HCC0020
```

Коротко пройдем по параметрам данной функции: первый требует указания, куда мы будем копировать; со второго по пятый — координаты левого верхнего угла, а также ширину и высоту получаемой картинки; шестой параметр указывает, откуда мы эту картинку берем; с седьмого по десятый — координаты левого верхнего угла вырезаемой картинки, а также ее ширину и высоту; последний параметр говорит о типе проводимой операции (в нашем случае копировании).

Теперь определимся с координатами нашего авто. Отслеживать их нам поможет тип `POINTAPI`, который мы также объявим в разделе деклараций модуля:

```
Public Type POINTAPI
    x As Long
    y As Long
End Type
```

Наконец, объявим переменную для координат нашего авто:

```
Public Auto As POINTAPI
```

На этом закончим предварительный этап копирования нашего авто и перейдем в форму.

Лирическое отступление 2

Вместо API-функции `StretchBlt` можно использовать встроенную в Visual Basic функцию — `PaintPicture`, которая не просто повторяет ее, но и целиком и полностью на нее опирается.

Шаг 4а. В декларациях формы запишем процедуру:

```
Public Sub DrawAuto()
    StretchBlt picRoad.hdc, Auto.x, Auto.y, _
        23, 31, picSrc.hdc, 0, 0, 23, 31, SRCCOPY
End Sub
```

Здесь данная функция вырезает из общей картинки первый автомобиль и копирует его на дорогу.

Объединим процедуры перерисовки дороги и нашего авто в единую процедуру, к которой мы в дальнейшем будем неоднократно обращаться:

```
Public Sub RedrawPic()
    DrawRoad
    DrawAuto
End Sub
```

Теперь, чтобы это все заработало, вернемся в модуль и напишем запускающую процедуру:

```
Public Sub Main()
  With frmMain
    .Show
    .RedrawPic
  End With
End Sub
```

Замечание

Не забудьте сделать процедуру `Main` стартовой при запуске проекта. Для этого в меню **Project** | **AutoRoad Properties...** в выпадающем списке **Startup Object** выберите Sub Main и нажмите **OK**.

Запустим проект (клавиша <F5>). На данном этапе должна быть дорога и ее в верхнем левом углу — наш автомобиль. Как говорится, все хорошо, но ему там не место. Проецирование автомобиля в верхний левый угол произошло потому, что по умолчанию тип `POINTAPI` дает начальное значение для `x` и `y` равным нулю. Давайте сразу же исправим это. Наше авто должно располагаться приблизительно посередине левой полосы дороги. Допишем две строки в процедуру `Main`, инициализирующие начальные координаты авто:

```
Public Sub Main()
  With frmMain
      Auto.x = .picRoad.Width * 0.75
      Auto.y = .picRoad.Height * 0.5
      .Show
      .RedrawPic
  End With
End Sub
```

Еще раз запустим проект и посмотрим, все ли правильно получилось у нас.

Шаг 5. Переходим к перемещению нашего авто. В нашем случае, автомобиль должен уметь перемещаться вправо и влево. Для этого используем событие `KeyDown`, так как оно позволяет обращаться к конкретным клавишам клавиатуры.

```
Private Sub picRoad_KeyDown(KeyCode As Integer, Shift As _
Integer)
Select Case KeyCode
Case vbKeyLeft
    Auto.x = Auto.x - 2
Case vbKeyRight
    Auto.x = Auto.x + 2
End Select
    RedrawPic
End Sub
```

При нажатии левой клавиши происходит смещение координаты `x` на 2 пиксела влево, на правую клавишу — на 2 пиксела вправо. После чего перерисовываем всю картинку с новыми координатами для авто. Одно маленькое неудобство — при длительном нажатии наше авто "уезжает" за край картинки. Ограничим его перемещения, учитывая также зеленый бордюр:

```
Private Sub picRoad_KeyDown(KeyCode As Integer, Shift As _
Integer)
Select Case KeyCode
Case vbKeyLeft
    If Auto.x <= picRoad.Width * 0.05 Then Exit Sub
    Auto.x = Auto.x - 2
Case vbKeyRight
    If Auto.x >= picRoad.Width * 0.95 - 23 Then Exit Sub
    Auto.x = Auto.x + 2
End Select
    RedrawPic
End Sub
```

Итак, на данном этапе мы добились прорисовки дороги, размещения на нем автомобиля и перемещения его вправо и влево.

Поэтому сохраним проект и можем переходить к следующему шагу, позволяющему отобразить другие авто (встречные и попутные).

Шаг 6. В принципе различие между отображением нашего и других авто только одно: наш автомобиль — один, а других — много. Поэтому воспользуемся теми же принципами прорисовки других авто, что и раньше. Единственное исключение сделаем для упрощения просчета координат в этой массе автомобилей: объявим массив. В модуле сделаем объявление:

```
Public OtherAuto(0 To 11) As POINTAPI ' от 0 до 5 - навстречу,
от 6 до 11 - по ходу
```

В форме напишем функцию, описывающую прорисовку одного "другого" авто:

```
Public Sub DrawOtherAuto(Num As Integer)
 Select Case Num
 Case 0 To 5
 StretchBlt picRoad.hdc, OtherAuto(Num).x, OtherAuto(Num).y,
_
    23, 31, picSrc.hdc, Num * 23, 31, 23, 31, SRCCOPY
    Case 6 To 11
    StretchBlt picRoad.hdc, OtherAuto(Num).x,
OtherAuto(Num).y, _
        23, 31, picSrc.hdc, (Num - 6) * 23, 62, 23, 31,
SRCCOPY
    End Select
End Sub
```

Принцип построения процедуры такой же, как и у нашего авто. Различие в вырезании из общей картинки: авто вырезается под номером `Num` (при ширине 23 пиксела), начиная с соответствующей строки (в зависимости от направления движения с 31 или 62 пиксела).

В процедуру `RedrawPic` внесем изменения, для отображения других авто. В итоге она должна выглядеть так:

```
Public Sub RedrawPic()
Dim i%
    DrawRoad
```

```
    DrawAuto
    For i = 0 To 11
        DrawOtherAuto (i)
    Next
End Sub
```

В запускающей процедуре `Main` также сделаем изменения, которые отобразят начальные координаты других машин:

```
Public Sub Main()
Dim i%
With frmMain
   Auto.x = .picRoad.Width * 0.75
   Auto.y = .picRoad.Height * 0.5
   For i = 0 To 5
     'добавляется по 2 пиксела к 23 для расстояния между
      машинами
     OtherAuto(i).x = (i * 25) + (.picRoad.Width * 0.05)
     OtherAuto(i).y = 0
   Next
   For i = 6 To 11
     'добавляется по 2 пиксела к 23 для расстояния между
      машинами
     OtherAuto(i).x = ((i - 6) * 25) + (.picRoad.Width * 0.5)
     OtherAuto(i).y = .picRoad.Height - 31
   Next
   .Show
   .RedrawPic
End With
End Sub
```

Для машин от 0 до 5 (двигающихся навстречу) координата `x` будет равна порядковому номеру авто плюс его ширина (23 пиксела). Но чтобы машины не "слипались" друг с другом бортами, добавим на ширину еще по 2 пиксела. Плюс учитываем ширину зеленого бордюра. Для попутных авто (от 6 до 11) — все то же самое, только расчет будет идти от середины дороги. Координату `y` временно приравняем к тем цифрам, чтобы встречные авто располагались сверху картинки, а попутные — снизу. Такая

симметрия нехарактерна для жизни, поэтому чуть позже мы к этому вернемся и исправим.

Шаг 7. Машины отображены, теперь самое время заставить их двигаться. Желая придать естественность движению на дороге, эту часть мы будем неоднократно изменять, поэтому вместо сразу конечного результата, я покажу, каким путем мы подошли к нему.

Сначала внесем изменения в форму.

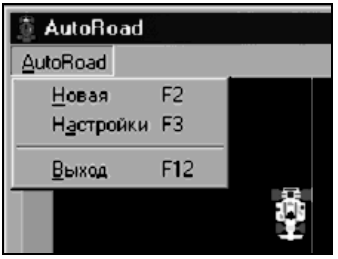


Рис. 3.2. Настройки меню

Итак, откроем форму frmMain и расположим на ней таймер со следующими параметрами (рис. 3.2.):

`Name` — tmrMove,

`Enabled` — False,

`Interval` — 100.

Добавим меню:

`Caption` — "&AutoRoad",

`Name` — mnuAutoRoad.

И подменю:

`Caption` — "&Новая",

`Name` — mnuNew,

`Shortcut` — F2;

`Caption` — "Н&астройки",

`Name` — mnuOptions,

`Shortcut` — F3;

`Caption` — "-",

`Name` — mnuSep;

`Caption` — "&Выход",

`Name` — mnuExit,

`Shortcut` — = F12.

Лирическое отступление 3

В редакторе меню для Visual Basic почему-то не предусмотрена клавиша <F10>, которой мы привыкли закрывать приложения, начиная еще с DOS'а. Вообще-то добавить клавишу <F10> не сложно, но получается более громоздко. А делается это так: на форме в окне свойств устанавливаем свойство `KeyPreview` — True; далее записываем процедуру:

```
Private Sub Form_KeyDown(KeyCode As Integer, Shift As
Integer)
    Select Case KeyCode
    Case vbKeyF10
     'здесь мы записываем, какое меню обрабатываем, на-
пример
      mnuExit_Click
    End Select
End Sub
```

Ну и в самом событии меню пишем, что же мы хотели все-таки сделать.

Чтобы избежать данных громоздкостей, я и остановился на клавише <F12> для данной учебной программы.

Шаг 8. Сначала опишем действия меню. Меню **Настройки** займемся чуть позже, когда будем проектировать соответствующую форму.

```
Private Sub mnuExit_Click()
    Unload Me
End Sub
Private Sub mnuNew_Click()
    tmrMove.Enabled = True
End Sub
```

Меню **Exit** — просто выгружает форму, а меню **Новая игра** — запускает таймер движения авто.

В событии таймера опишем движения авто:

```
Private Sub tmrMove_Timer()
Dim i%
For i = 0 To 5
    OtherAuto(i).y = OtherAuto(i).y + 2
Next
For i = 6 To 11
    OtherAuto(i).y = OtherAuto(i).y - 2
Next
RedrawPic
End Sub
```

`y + 2` или `y - 2` — это смещение на 2 пиксела по вертикали, относительно нынешней координаты.

Учитывая, что перерисовка картинки у нас происходит достаточно часто, согласно таймеру, каждые 0,1 сек., то в процедуре движения нашего авто мы эту перерисовку можем опустить. Удалите из `picRoad_KeyDown` последнюю строку (`RedrawPic`).

Запустим и посмотрим: авто двинулись как на параде! Проехали за край и исчезли, а нам хотелось бы повторяемости. Поэтому добавим отслеживание координаты у каждого авто и при полном исчезновении за краем картинки дороги прорисовываем его на противоположной стороне:

```
Private Sub tmrMove_Timer()
Dim i%
For i = 0 To 5
    If OtherAuto(i).y >= picRoad.Height Then
        OtherAuto(i).y = -31
    ElseIf OtherAuto(i).y <= -31 Then
        OtherAuto(i).y = picRoad.Height
    End If
    OtherAuto(i).y = OtherAuto(i).y + 2
Next
For i = 6 To 11
```

```
    If OtherAuto(i).y >= picRoad.Height Then
        OtherAuto(i).y = -31
    ElseIf OtherAuto(i).y <= -31 Then
        OtherAuto(i).y = picRoad.Height
    End If
    OtherAuto(i).y = OtherAuto(i).y - 2
Next
RedrawPic
End Sub
```

Попробуем запустить еще раз. Авто "строем" доезжают до края дороги и появляются с противоположной стороны. Уже лучше! Но по-прежнему неестественно. Некоторые, наверное, замечали, что на магистрали автомобили, желающие ехать быстрее, располагаются ближе к осевой линии. Давайте и мы внесем эти изменения в движение нашего транспорта, и смещение каждого авто будет не на 2 пиксела, а на 2 плюс порядковый номер авто.

```
Private Sub tmrMove_Timer()
Dim i%
For i = 0 To 5
    If OtherAuto(i).y >= picRoad.Height Then
        OtherAuto(i).y = -31
    ElseIf OtherAuto(i).y <= -31 Then
        OtherAuto(i).y = picRoad.Height
    End If
    OtherAuto(i).y = OtherAuto(i).y + 2 + i
Next
For i = 6 To 11
    If OtherAuto(i).y >= picRoad.Height Then
        OtherAuto(i).y = -31
    ElseIf OtherAuto(i).y <= -31 Then
        OtherAuto(i).y = picRoad.Height
    End If
    OtherAuto(i).y = OtherAuto(i).y - 2 - (13 - i)
```

```
Next
RedrawPic
End Sub
```

Запустим проект и понаблюдаем за движением авто.

Шаг 9. Движение стало более "осмысленным". Немного смущает стартовое положение авто (в одну линию). Давайте поправим это. Для генерации случайных чисел в Visual Basic существует функция `Randomize`. Вот ею мы и воспользуемся для определения "случайного" положения авто по вертикали при загрузке программы. Внесем изменения в процедуру `Main`, расположенную в модуле:

```
Public Sub Main()
Dim i%
With frmMain
  Auto.x = .picRoad.Width * 0.75
  Auto.y = .picRoad.Height * 0.5
  Randomize
  For i = 0 To 5
     OtherAuto(i).x = (i * 25) + (.picRoad.Width * 0.05)
     OtherAuto(i).y = Int((.picRoad.Height + 1) * Rnd)
  Next
  For i = 6 To 11
  OtherAuto(i).x = ((i - 6) * 25) + (.picRoad.Width * _
0.5)
  OtherAuto(i).y = Int((.picRoad.Height + 1) * Rnd)
  Next
   .Show
   .RedrawPic
End With
End Sub
```

Теперь при открытии формы и встречные и попутные авто располагаются в случайном порядке.

Шаг 10. Автомобили едут, согласно предписанию. И только наше авто может перемещаться вправо и влево с нулевой скоростью движения вперед. Пора заняться скоростными качествами нашего авто.

Откроем форму и добавим два лейбла. Первый:

`Name` — lblCaption,

`Caption` —"Скорость:",

`Font.Bold` — True,

`Font.Size` — 10,

`ForeColor` — &H00FF0000&,

`Height` — 16,

`Index` — 0,

`Left` — 348,

`Top` — 64,

`Width` — 74.

И второй:

`Name` — lblSpeed,

`AutoSize` — True,

`Caption` — "0",

`Font.Bold` — True,

`Font.Size` — 10,

`ForeColor` — &H00FF0000&,

`Height` — 16,

`Left` — 428,

`Top` — 64.

Добавим новую переменную в раздел деклараций модуля:

```
Public Speed As Integer
```

В форме в процедуру `picRoad_KeyDown` внесем дополнения для нажатий на клавиши вверх-вниз. Для клавиши "вниз" сразу же исключим отрицательные значения скоростей. Для клавиши "вверх" — ограничение максимальной скорости мы введем позже. После изменения скорости отобразим это в соответствую-

щем лейбле, выбрав чисто эмпирически коэффициент для приближения к действительности (я думаю, что таким коэффициентом может быть, например, 20).

```
Private Sub picRoad_KeyDown(KeyCode As Integer, Shift As _
Integer)
Select Case KeyCode
Case vbKeyUp
    Speed = Speed + 1
    lblSpeed.Caption = Speed * 20
Case vbKeyDown
    Speed = Speed - 1
    If Speed < 0 Then Speed = 0
    lblSpeed.Caption = Speed * 20
End Select
End Sub
```

Если сейчас запустить проект, то мы увидим изменения, происходящие в отображении скорости (в лейбле), но никак это не отобразится на самом движении автомобилей. Поэтому, заканчивая шаг 10, сделаем "видимым" наше изменение скорости. То есть попутные авто с увеличением скорости должны двигаться относительно нас медленнее, а встречные на ту же самую скорость — быстрее. Для этого добавим значение переменной `Speed` к скорости перемещения всех авто.

```
Private Sub tmrMove_Timer()
Dim i%
For i = 0 To 5
    If OtherAuto(i).y >= picRoad.Height Then
        OtherAuto(i).y = -31
    ElseIf OtherAuto(i).y <= -31 Then
        OtherAuto(i).y = picRoad.Height
    End If
    OtherAuto(i).y = OtherAuto(i).y + 2 + i + Speed
Next
For i = 6 To 11
    If OtherAuto(i).y >= picRoad.Height Then
```

```
        OtherAuto(i).y = -31
    ElseIf OtherAuto(i).y <= -31 Then
        OtherAuto(i).y = picRoad.Height
    End If
    OtherAuto(i).y = OtherAuto(i).y - 2 - (13 - i) + Speed
Next
RedrawPic
End Sub
```

Фух-х-х! Полдела сделано ☺.

Шаг 11. Теперь ответственный момент. Мы должны отреагировать на столкновение автомобилей. В нашей игрушке это должно произойти, когда любая точка нашего авто пересечется с любой точкой любого другого авто. Запишем эту процедуру:

```
Private Sub Crash()
Dim i%
For i = 0 To 11
'при совпадении координат
    'по вертикали
    If ((Auto.y + 31 >= OtherAuto(i).y) And _
        (Auto.y + 31 <= OtherAuto(i).y + 31)) Or _
        ((Auto.y >= OtherAuto(i).y) And _
        (Auto.y <= OtherAuto(i).y + 31)) Then
        '+ по горизонтали
        If ((Auto.x + 23 >= OtherAuto(i).x) And _
            (Auto.x + 23 <= OtherAuto(i).x + 23)) Or _
            ((Auto.x >= OtherAuto(i).x) And _
            (Auto.x <= OtherAuto(i).x + 23)) Then
            ' т.е. при столкновении - обнуляем скорость нашего
            ' авто
            Speed = 0
            lblSpeed.Caption = "0"
        End If
    End If
Next
End Sub
```

Данная процедура должна проверяться постоянно при движении всех авто — т. е. в процедуре таймера, непосредственно перед перерисовкой всех объектов:

```
Private Sub tmrMove_Timer()
...
    Crash
    RedrawPic
End Sub
```

Теперь при запуске проекта, в период столкновения скорость нашего авто сбрасывается до нуля.

Шаг 12. Продолжаем улучшать внешний вид нашей игры. Добавим еще 2 лейбла для отображения пройденного нашим автомобилем пути. Первый с параметрами:

`Name` — lblCaption,

`Caption` — "Пройденный путь:",

`Font.Bold` — True,

`Font.Size` — 10,

`ForeColor` — &H000000FF&,

`Height` — 16,

`Index` — 1,

`Left` — 348,

`Top` — 84,

`Width` — 136.

И второй:

`Name` — lblPath,

`AutoSize` — True,

`Caption` — ="0",

`Font.Bold` — True,

`Font.Size` — 10,

`ForeColor` — &H000000FF&,

`Height` — 16,

`Left` — 488,

`Top` — 84.

В событие таймера внесем изменения пройденного расстояния. Если верить учебнику математики за 3 класс, то расстояние высчитывается по формуле $S = V \cdot T$. И скорость и интервал времени у нас известны.

```
Private Sub tmrMove_Timer()
...
  Crash
  lblPath.Caption = lblPath.Caption + _ (lblSpeed.Caption * _
0.01)
  RedrawPic
End Sub
```

Теперь, запустив проект, мы увидим, что в зависимости от скорости расстояние нарастает, соответственно, быстрее или медленнее.

Шаг 13. Любая игра имеет свои ограничения. Чаще всего это бывает ограничением по времени. Не будем оригинальными и ограничим пользователя, допустим, 1 минутой. Отсчет времени должен видеть пользователь, поэтому на форме frmMain расположим еще 2 лейбла и новый таймер.

Первый лейбл:

`Name` — lblCaption,

`Caption` — "Время игры:",

`Font.Bold` — True,

`Font.Size` — 10,

`ForeColor` — &H00000000&,

`Height` — 16,

`Index` — 2,

`Left` — 348,

`Top` — 104,

`Width` — 90.

Второй лейбл:

`Name` — lblTime,

`AutoSize` — True,

`Caption` — "0:01:00",

`Font.Bold` — True,

`Font.Size` — 10,

`ForeColor` — &H00000000&,

`Height` — 16,

`Left` — 444,

`Top` — 104.

Таймер:

`Name` — tmrTime,

`Enabled` — False,

`Interval` — 1000.

Добавим в процедуру запуска новой игры инициализацию второго таймера.

```
Private Sub mnuNew_Click()
    tmrMove.Enabled = True
    tmrTime.Enabled = True
End Sub
```

А в процедуру работы самого таймера добавим обратный отсчет времени и остановку игры по его истечении, а также обнуление переменной `Speed`.

```
Private Sub tmrTime_Timer()
lblTime.Caption = CDate(CDate(lblTime.Caption) - _
CDate("0:00:01"))
If lblTime.Caption = "0:00:00" Then
    tmrMove.Enabled = False
    tmrTime.Enabled = False
    MsgBox "Пройденное расстояние - " & lblPath.Caption _
        & " км.", vbInformation + vbOKOnly, "Игра закончена"
    lblSpeed.Caption = "0"
```

```
    lblPath.Caption = "0"
    lblTime.Caption = "0:01:00"
    Speed = 0
End If
```

Итак, в черновике игра сделана. У нас есть автомобиль, которым мы управляем (скорость и направление), встречные и попутные машины, двигающиеся с различной скоростью, столкновение с которыми ведет к остановке нашего автомобиля. Мы можем отслеживать скорость и пройденный путь, а также зафиксировать остановку игры через определенный интервал времени. В таком случае напрашивается вопрос: "Чего будет недостаточно пользователю в данной игрушке?" Ответ один: ощущения, что он сам может устанавливать правила игры. Давайте позволим ему тешиться этой мыслью и предоставим "свободу" в том объеме, в котором мы сами пожелаем. Итак ...

Шаг 14. Собираем форму "Настройки". Вызываем меню: **Project Add Form** и выбираем обычную форму. Ее параметры:

`Name` — frmOption,

`BorderStyle` — 1 – Fixed Single,

`Caption` — "Настройки",

`Height` — 1815,

`Icon` — устанавливаем ту же, что и на основную форму,

`StartupPosition` — 1 – CenterOwner,

`Width` — 4860.

NB! В данной форме мы не будем работать с графикой и API-функцией, поэтому нет смысла изменять свойство `ScaleMode`. В этой форме мы будем работать с твипами.

Разместим на форме 4 лейбла для надписей:

`Name` — lblCaption,

`Index` — 0; 1; 2; 3,

`Caption` — "Максимальная скорость"; "Время"; "Количество попутных авто"; "Количество встречных авто",

`Height` — 195,

`Left` — 120,

Top — 120; 420; 720; 1020,

Width — 2295.

Добавим два текстовых поля:

Name — txtMaxSpeed,

Height — 285,

Left — 2520,

MaxLength — 3,

Text — "300",

Top — 60,

Width — 735;

Name — txtTime,

Height — 285,

Left — 2520,

MaxLength — 7,

Text — "0:01:00",

Top — 360,

Width — 735.

И два выпадающих списка:

Name — cboNumAuto,

Index — 0; 1,

Left — 2520,

List — 0|1|2|3|4|5|6,

Style — 2 – Dropdown List,

Top — 660; 960,

Width — 795.

Ну и, наконец, две кнопки:

Name — cmdOKCancel,

Index — 0; 1,

Caption — "OK"; "Отмена",

Height — 215,

`Left` — 3480,

`Top` — 540; 960,

`Width` — 1155.

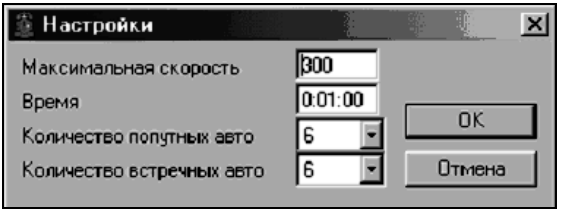


Рис. 3.3. Настройки игры

В форме frmMain сделаем вызов новой формы:

```
Private Sub mnuOptions_Click()
    frmOption.Show vbModal
End Sub
```

Теперь можно попробовать запустить проект.

Шаг 15. Где можно хранить настройки? Вариантов несколько, но наиболее часто встречающиеся — это либо в реестре, либо в отдельном файле (например *.ini). В нашем случае давайте будем сохранять в реестре. В Visual Basic для работы с реестром существует 4 встроенные функции: GetSetting, SaveSetting, DeleteSetting и GetAllSettings. У них существует один недостаток: они работают только с одной ветвью реестра — HKEY_CURRENT_USER\SOFTWARE\VB and VBA Program Settings. В принципе, мы не будем хранить никакой секретной информации типа пароля, серийного номера и т. п., поэтому нас устроят и эти функции (а вернее, первые две, которые считывают и записывают данные из реестра). Единственное, что мы сделаем — это "обертку" для них, чтобы нам было удобнее пользоваться. Так как обращаться к реестру мы будем из обеих форм, то нашу функцию мы должны разместить, естественно, в модуле.

Для начала опишем две нумерованные константы в разделе деклараций модуля.

```
Public Enum constTypeOptions
    GetOption = 0
```

```
    SaveOption = 1
End Enum
Public Enum constKeyOptions
    MaxSpeed = 0
    Time = 1
    AutoUp = 2
    AutoDown = 3
End Enum
```

Замечание

NB! Нумерованные константы могут принимать только целочисленные значения.

Ну а теперь и саму функцию-обертку:

```
Public Function Options(ByVal TypeOptions As _
constTypeOptions, ByVal Key As constKeyOptions, _
Optional Setting As String) As String
Select Case TypeOptions
Case GetOption 'считывание
    Options = GetSetting(App.EXEName, "Options", Key)
Case SaveOption 'запись
    SaveSetting App.EXEName, "Options", Key, Setting
End Select
End Function
```

Теперь давайте попробуем запустить эту функцию в работу. В самый первый запуск нашей программы мы должны сделать запись в реестр со значениями по умолчанию. В дальнейшем, через окно настроек мы будем устанавливать свои параметры. В процедуру `Main`, в самое начало впишем проверку на наличие записей о нашей программе в реестре.

```
Public Sub Main()
Dim i%
If Options(GetOption, Time) = vbNullString Then
    Options SaveOption, AutoDown, 6
    Options SaveOption, AutoUp, 6
    Options SaveOption, MaxSpeed, 300
```

```
    Options SaveOption, Time, "0:01:00"
End If
...
End Sub
```

Запустим проект и сразу же его закроем. Зайдем в реестр: **Пуск | Выполнить**; в текстовом поле наберем "regedit" и нажмем **ОК**. Откроем ветку HKEY_CURRENT_USER\SOFTWARE\VB and VBA Program Settings. Опустимся ниже в наш проект: AutoRoad\Options. Если все сделано правильно, то в правом окне вы должны увидеть следующие записи:

(По умолчанию) — (Значение не присвоено)

0 — "300"

1 — "0:01:00"

2 — "6"

3 — "6"

То есть наши параметры по умолчанию были записаны.

Шаг 16. Перейдем в форму frmMain и в событии загрузки формы запишем считывание в лейбл времени.

```
Private Sub Form_Load()
    lblTime.Caption = Options(GetOption, Time)
End Sub
```

Поставим ограничение максимальной скорости нашего авто.

```
Private Sub picRoad_KeyDown(KeyCode As Integer, Shift As _
Integer)
Select Case KeyCode
...
Case vbKeyUp
    Speed = Speed + 1
    If Speed * 20 > Options(GetOption, MaxSpeed) Then _
        Speed = Options(GetOption, MaxSpeed) / 20
    lblSpeed.Caption = Speed * 20
...
End Select
End Sub
```

В событии таймера, следящего за временем игры, сделаем исправление для лейбла времени на окончание игры.

```
Private Sub tmrTime_Timer()
lblTime.Caption = CDate(CDate(lblTime.Caption) -
CDate("0:00:01"))
If lblTime.Caption = "0:00:00" Then
...lblTime.Caption = Options(GetOption, Time)
   Speed = 0
End If
End Sub
```

Пока на этом закончим и перейдем к следующему шагу.

Шаг 17. Форма frmOptions. Эта форма должна при открытии считать данные реестра и заполнить соответствующие поля, а при выходе сохранить изменения в реестре.

```
Private Sub Form_Load()
    txtTime.Text = Options(GetOption, Time)
    txtMaxSpeed.Text = Options(GetOption, MaxSpeed)
    cboNumAuto(0).ListIndex = Options(GetOption, AutoUp)
    cboNumAuto(1).ListIndex = Options(GetOption, AutoDown)
End Sub
Private Sub cmdOKCancel_Click(Index As Integer)
Select Case Index
Case 0 'если OK - сохраняемся в реестре
    Options SaveOption, Time, txtTime.Text
    Options SaveOption, MaxSpeed, txtMaxSpeed.Text
    Options SaveOption, AutoUp, CStr(cboNumAuto(0).ListIndex)
    Options SaveOption, AutoDown,
CStr(cboNumAuto(1).ListIndex)
End Select
'закрытие формы
Unload Me
End Sub
```

В случае если нажимается кнопка **Отмена**, то происходит просто выгрузка формы без сохранения данных в реестр.

Теперь можно немного поиграть с окном настроек, выискивая недостатки.

Шаг 18. Недостатков оказалось не так много, но они все-таки есть. Если в ComboBox'ах мы особенно-то развернуться пользователю не даем (можно только выбирать из предложенного), то в текстовых полях — пиши чего хочешь. На стадии рисования формы мы ограничили количество вводимых знаков в текстовые поля (свойство `MaxLenght`), теперь давайте займемся ограничением ввода тех знаков, которые мы хотим разрешить вводить пользователю. Очень удобным для этого является событие `KeyPress`, которое использует ASCII-коды каждого символа. Мы проверяем вводимый символ и, если он допустим, — ничего не делаем, а если нет — не выводим его и выдаем звуковой сигнал.

```
Private Sub txtMaxSpeed_KeyPress(KeyAscii As Integer)
Select Case KeyAscii
Case 8, 48 To 57 'backspace + цифры от 0 до 9
Case Else
    Beep
    KeyAscii = 0
End Select
End Sub
Private Sub txtTime_KeyPress(KeyAscii As Integer)
Select Case KeyAscii
Case 8, 48 To 57, 58 'backspace + цифры от 0 до 9 + двоеточие
Case Else
    Beep
    KeyAscii = 0
End Select
End Sub
```

Теперь проблем с вводом цифр нет, и поле Максимальной скорости работает идеально, а вот с полем времени проблемы еще остались. Например, мы можем ввести такую запись ":::7755". Давайте заставим пользователя правильно ввести время:

```
Private Sub cmdOKCancel_Click(Index As Integer)
Select Case Index
Case 0 'если ОК - сохраняемся в реестре
```

```
    'проверка на правильность введения времени
    If Not IsDate(txtTime.Text) Then
        MsgBox "В поле не время!", vbExclamation, "Ошибка!"
        Exit Sub
    End If
    Options SaveOption, Time, txtTime.Text
    Options SaveOption, MaxSpeed, txtMaxSpeed.Text
    Options SaveOption, AutoUp, CStr(cboNumAuto(0).ListIndex)
    Options SaveOption, AutoDown,
CStr(cboNumAuto(1).ListIndex)
End Select
'закрытие формы
Unload Me
End Sub
```

Осталось изменить параметры в форме frmMain при выгрузке формы frmOptions

```
Private Sub Form_Unload(Cancel As Integer)
'передача данных в основную форму
    frmMain.lblTime.Caption = Options(GetOption, Time)
End Sub
```

Количеством встречных и попутных авто мы займемся чуть-чуть позже.

Шаг 19. Итак, это "чуть-чуть позже" наступило. Для начала выведем из процедуры `Main` в модуле часть кода, для определения случайного положения машин в отдельную процедуру и назовем ее `RndAuto`. А циклы `For ... Next` привяжем к данным о количестве машин, хранящихся в реестре. В исправленном виде обе процедуры должны выглядеть так:

```
Public Sub Main()
If Options(GetOption, Time) = vbNullString Then
    Options SaveOption, AutoDown, 6
    Options SaveOption, AutoUp, 6
    Options SaveOption, MaxSpeed, 300
    Options SaveOption, Time, "0:01:00"
End If
```

```
With frmMain
    Auto.x = .picRoad.Width * 0.75
    RndAuto
    .Show
    .RedrawPic
End With
End Sub
Public Sub RndAuto()
Dim i%
With frmMain
    Randomize
    For i = 0 To Options(GetOption, AutoDown) - 1
      OtherAuto(i).x = (i * 25) + (.picRoad.Width * 0.05)
      OtherAuto(i).y = Int((.picRoad.Height + 1) * Rnd)
    Next
    For i = 6 To Options(GetOption, AutoUp) + 5
      OtherAuto(i).x = ((i - 6) * 25) + (.picRoad.Width *
0.5)
      OtherAuto(i).y = Int((.picRoad.Height + 1) * Rnd)
    Next
End With
End Sub
```

В форме **Настройки** на выходе добавляем процедуру `RndAuto` и заставляем основную форму перерисоваться — `frmMain.RedrawPic`.

```
Private Sub Form_Unload(Cancel As Integer)
    With frmMain
        .lblTime.Caption = Options(GetOption, Time)
        RndAuto
        .RedrawPic
    End With
End Sub
```

Осталось немного: сделать исправления в цикле `RedrawPic`, относительно настроек реестра.

```
Public Sub RedrawPic()
Dim i%
    DrawRoad
```

```
    DrawAuto
    For i = 0 To Options(GetOption, AutoDown) - 1
        DrawOtherAuto i
    Next
    For i = 6 To Options(GetOption, AutoUp) + 5
        DrawOtherAuto i
    Next
End Sub
```

И то же самое в циклах процедуры `tmrMove_Timer`:

```
Private Sub tmrMove_Timer()
Dim i%
For i = 0 To Options(GetOption, AutoDown) - 1
...
Next
For i = 6 To Options(GetOption, AutoUp) + 5
...
Next
...
End Sub
```

Собственно говоря, на этом написание логики нашей игры можно считать законченной.

Шаг 20. При желании, можно немного украсить нашу игру. Давайте создадим ActiveX Control, который будет эмулировать работу спидометра.

NB! ActiveX Control'ы могут существовать в 2-х видах: как отдельный файл (для многоразового использования в различных программах) и как составная часть конкретной программы.

Мы воспользуемся встроенным ActivX Control'ом. Меню **Project | Add User Control** добавит его в ваш проект. Установим некоторые его свойства:

`Name` — Speedometr,

`AutoRedraw` — True,

`BorderStyle` — 1 – Fixed Single,

`DrawStyle` — 0 – Solid,

`FillColor` — &H00FFFFC0&,

`FillStyle` — 0 – Solid,

`ToolboxBitmap` — вставьте любую картинку (bmp) 15×16 пикселов, какую хотите увидеть на панели инструментов.

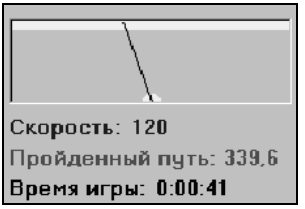


Рис. 3.4. Спидометр

Далее воспользуемся мастером создания ActivX Control'ов. Для этого выберите меню **Add-Ins | Add-In Manager...** В открывшемся окне выберите VB6 ActiveX Ctrl Interface Wizard, а в **CheckBox**'е "Loaded/Unloaded" поставьте галочку и нажмите кнопку **OK**. Теперь через меню **Add-Ins | ActiveX Control Interface Wizard** вызовем непосредственно сам мастер и воспользуемся его пошаговыми услугами.

Step 1. В правом списке оставим только свойство `BackColor`.

Step 2. С помощью кнопки **New** добавим свои свойства: `Min`, `Max`, `Value`, `ArrowColor`, `ConturColor`, `ArrowWidth`.

Step 3. Здесь мы сделаем привязку некоторых свойств к свойствам самого UserControl'а:

```
BackColor – UserControl.BackColor
ConturColor – UserControl.FillColor
ArrowWidth – UserControl.DrawWidth
```

Step 4. Для оставшихся свойств выберем значения (`Public Name — Data Type – Default Value`):

```
ArrowColor - OLE_COLOR - 0
Max - Long - 100
Min - Long - 0
Value - Long - 0
```

Нажмем кнопку **OK**. Мастер сгенерировал нам код.

Сделаем маленькие дополнения, чтобы наш ActiveX Control нормально изображал спидометр. Вначале очистим от предыдущей графики. Следующие две строки служат в качестве украшений: они рисуют светлую полоску сверху и полукруг внизу в центре. А вот последняя строка — основная. Она рисует стрелку спидометра (прямая линия) с началом — внизу в центре и концом — вверху в точке, зависящей от значения `Value` (пересчет производится в единицы, относительно ширины Control'a).

```
Private Sub Draw()
Cls
Line (0, 0)-(ScaleWidth, ScaleHeight * 0.1), ConturColor, BF
Circle (ScaleWidth * 0.5, ScaleHeight), ScaleHeight * 0.1, _
ConturColor
Line (ScaleWidth * 0.5, ScaleHeight)-(ScaleWidth * _
(m_Value - m_Min)/ _(m_Max - m_Min), 0), m_ArrowColor
End Sub
```

Теперь, чтобы эта процедура заработала, ее необходимо вставить в каждое `Property Let` нашего Control'a, примерно так:

```
Public Property Let Value(ByVal New_Value As Long)
    m_Value = New_Value
    PropertyChanged "Value"
    Draw
End Property
```

и добавить процедуру, инициализирующую графику при запуске Control'a:

```
Private Sub UserControl_Show()
    Draw
End Sub
```

Control готов.

Шаг 21. Привязываем Control к форме. В frmMain внесем изменения в события `Form_Load`, `picRoad_KeyDown`, `Crash`, `tmrTime_Timer`, чтобы привязать к изменениям, происходящим с переменной `Speed` и дублирующим изменения `lblSpeed`, только в графическом варианте:

```
Private Sub Form_Load()
    lblTime.Caption = Options(GetOption, Time)
```

```
    Speedometr1.Max = Options(GetOption, MaxSpeed)
End Sub
Private Sub picRoad_KeyDown(KeyCode As Integer, _
Shift As Integer)
Select Case KeyCode
...
Case vbKeyUp
    Speed = Speed + 1
    If Speed * 20 > Options(GetOption, MaxSpeed) Then _
        Speed = Options(GetOption, MaxSpeed) / 20
    lblSpeed.Caption = Speed * 20
    Speedometr1.Value = Speed * 20
Case vbKeyDown
    Speed = Speed - 1
    If Speed < 0 Then Speed = 0
    lblSpeed.Caption = Speed * 20
    Speedometr1.Value = Speed * 20
End Select
End Sub
Private Sub Crash()
...
            Speed = 0
            lblSpeed.Caption = "0"
            Speedometr1.Value = 0
...
End Sub
Private Sub tmrTime_Timer()
...
    lblTime.Caption = Options(GetOption, Time)
    Speedometr1.Value = 0
    Speed = 0
End If
End Sub
```

В frmOption в событие `Form_Unload` также внесем изменения, которые характеризуют изменения в Control'e, а именно его максимальное значение:

```
Private Sub Form_Unload(Cancel As Integer)
    With frmMain
        .lblTime.Caption = Options(GetOption, Time)
        .Speedometr1.Max = Options(GetOption, MaxSpeed)
        RndAuto
        .RedrawPic
    End With
End Sub
```

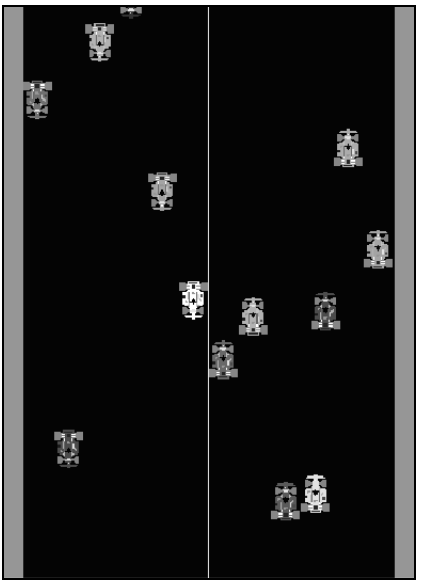


Рис. 3.5. Игра в действии

Таким образом, за 21 шаг сделана вполне функциональная игрушка. У каждого, кто программирует, вполне возможно, появятся свои идеи, как улучшить данное приложение, чтобы оно стало еще интереснее для пользователя. Что это будет: улучшение графики или увеличение возможностей движения авто — решать вам.

3.2. Листинг программного кода к части "Как написать игрушку на Visual Basic"

3.2.1. Модуль: mdlAuto

```
Option Explicit
Public Declare Function StretchBlt Lib "gdi32" ( _
    ByVal hdc As Long, _
    ByVal x As Long, _
    ByVal y As Long, _
    ByVal nWidth As Long, _
    ByVal nHeight As Long, _
    ByVal hSrcDC As Long, _
    ByVal xSrc As Long, _
    ByVal ySrc As Long, _
    ByVal nSrcWidth As Long, _
    ByVal nSrcHeight As Long, _
    ByVal dwRop As Long) _
As Long
Public Type POINTAPI
    x As Long
    y As Long
End Type
Public Enum constTypeOptions
    GetOption = 0
    SaveOption = 1
End Enum
Public Enum constKeyOptions
    MaxSpeed = 0
    Time = 1
    AutoUp = 2
    AutoDown = 3
End Enum
```

```
Public Const SRCCOPY = &HCC0020
Public Auto As POINTAPI
Public OtherAuto(0 To 11) As POINTAPI ' от 0 до 5 -
навстречу,
                                      ' от 6 до 11 - по ходу
Public Speed As Integer
Public Sub Main()
'проверка на самое первое открытие игры,
'в случае необходимости пишем в реестр
If Options(GetOption, Time) = vbNullString Then
    Options SaveOption, AutoDown, 6
    Options SaveOption, AutoUp, 6
    Options SaveOption, MaxSpeed, 300
    Options SaveOption, Time, "0:01:00"
End If
With frmMain
    'позиционируем наше авто
    Auto.x = .picRoad.Width * 0.75
    Auto.y = .picRoad.Height * 0.5
    'просчитываем координаты других авто
    RndAuto
    .Show
    .RedrawPic
End With
End Sub
Public Sub RndAuto()
Dim i%
With frmMain
    Randomize
    For i = 0 To Options(GetOption, AutoDown) - 1
'для встречных добавляется по 2 пиксела к 23 для расстояния
'между машинами
OtherAuto(i).x = (i * 25) + (.picRoad.Width * 0.05)
OtherAuto(i).y = Int((.picRoad.Height + 1) * Rnd)
    Next
```

```
    For i = 6 To Options(GetOption, AutoUp) + 5
'для попутных добавляется по 2 пиксела к 23 для расстояния
'между машинами
      OtherAuto(i).x = ((i - 6) * 25) + (.picRoad.Width *
0.5)
      OtherAuto(i).y = Int((.picRoad.Height + 1) * Rnd)
    Next
End With
End Sub
'упрощение доступа к реестру
Public Function Options(ByVal TypeOptions As _
constTypeOptions, ByVal Key As constKeyOptions, Optional _
Setting As String) As String
Select Case TypeOptions
Case GetOption 'считывание
    Options = GetSetting(App.EXEName, "Options", Key)
Case SaveOption 'запись
    SaveSetting App.EXEName, "Options", Key, Setting
End Select
End Function
```

3.2.2. Форма: frmMain

```
Option Explicit
Private Sub Form_Load()
    'обновляем значения, согласно сохраненному в реестре
    lblTime.Caption = Options(GetOption, Time)
    Speedometr1.Max = Options(GetOption, MaxSpeed)
End Sub
Private Sub mnuExit_Click()
    Unload Me
End Sub
Private Sub mnuNew_Click()
    'новая игра, запускаем таймеры
    tmrMove.Enabled = True
    tmrTime.Enabled = True
```

```
End Sub
Private Sub mnuOptions_Click()
'запуск формы настроек
    frmOption.Show vbModal
End Sub
Private Sub picRoad_KeyDown(KeyCode As Integer, _
Shift As Integer)
Select Case KeyCode
Case vbKeyLeft 'смещение влево
    If Auto.x <= picRoad.Width * 0.05 Then Exit Sub
    Auto.x = Auto.x - 2
Case vbKeyRight 'смещение вправо
    If Auto.x >= picRoad.Width * 0.95 - 23 Then Exit Sub
    Auto.x = Auto.x + 2
Case vbKeyUp 'увеличение скорости
    Speed = Speed + 1
    If Speed * 20 > Options(GetOption, MaxSpeed) Then _
        Speed = Options(GetOption, MaxSpeed) / 20
    lblSpeed.Caption = Speed * 20
    Speedometr1.Value = Speed * 20
Case vbKeyDown 'уменьшение скорости
    Speed = Speed - 1
    If Speed < 0 Then Speed = 0
    lblSpeed.Caption = Speed * 20
    Speedometr1.Value = Speed * 20
End Select
End Sub
Public Sub DrawRoad()
    'Рисуем:
    'левый зеленый бордюр
    picRoad.Line (0, 0)-(picRoad.Width * 0.05, _
picRoad.Height), vbGreen, BF
    'правый зеленый бордюр
    picRoad.Line (picRoad.Width * 0.95, 0)-(picRoad.Width, _
picRoad.Height), _
```

```
        vbGreen, BF
    'дорога
    picRoad.Line (picRoad.Width * 0.05, 0)-(picRoad.Width * _
0.95, _
        picRoad.Height), vbBlack, BF
    'осевая линия
    picRoad.Line (picRoad.Width * 0.5, 0)-(picRoad.Width * _
0.5, _picRoad.Height), vbYellow, BF
End Sub
Public Sub DrawAuto()
    'копируем наше авто из общей картинки на дорогу
    StretchBlt picRoad.hdc, Auto.x, Auto.y, _
        23, 31, picSrc.hdc, 0, 0, 23, 31, SRCCOPY
End Sub
'Перерисовка всей графики
Public Sub RedrawPic()
Dim i%
    DrawRoad
    DrawAuto
    'встречные авто
    For i = 0 To Options(GetOption, AutoDown) - 1
        DrawOtherAuto i
    Next
    'попутные авто
    For i = 6 To Options(GetOption, AutoUp) + 5
        DrawOtherAuto i
    Next
End Sub
'копируем прочие авто из общей картинки на дорогу
Public Sub DrawOtherAuto(Num As Integer)
    Select Case Num
    Case 0 To 5 'встречные
    StretchBlt picRoad.hdc, OtherAuto(Num).x, _
OtherAuto(Num).y, _
        23, 31, picSrc.hdc, Num * 23, 31, 23, 31, SRCCOPY
```

```
    Case 6 To 11 'попутные
    StretchBlt picRoad.hdc, OtherAuto(Num).x, _
OtherAuto(Num).y, _
        23, 31, picSrc.hdc, (Num - 6) * 23, 62, 23, 31, _
        SRCCOPY
    End Select
End Sub
Private Sub tmrMove_Timer()
Dim i%
'для встречных авто
For i = 0 To Options(GetOption, AutoDown) - 1
    'если выходит за край картинки дороги
    If OtherAuto(i).y >= picRoad.Height Then
        OtherAuto(i).y = -31
    ElseIf OtherAuto(i).y <= -31 Then
        OtherAuto(i).y = picRoad.Height
    End If
    '+2 - смещение на 2 пиксела
    OtherAuto(i).y = OtherAuto(i).y + 2 + i + Speed
Next
'для попутных авто
For i = 6 To Options(GetOption, AutoUp) + 5
    'если выходит за край картинки дороги
    If OtherAuto(i).y >= picRoad.Height Then
        OtherAuto(i).y = -31
    ElseIf OtherAuto(i).y <= -31 Then
        OtherAuto(i).y = picRoad.Height
    End If
    '-2 - смещение на 2 пиксела
    OtherAuto(i).y = OtherAuto(i).y - 2 - (13 - i) + Speed
Next
    'проверка на аварию
    Crash
    'пройденное расстояние
    lblPath.Caption = lblPath.Caption + (lblSpeed.Caption * _
0.01)
```

```
    RedrawPic
End Sub
Private Sub Crash()
Dim i%
For i = 0 To 11
'при совпадении координат
    'по вертикали
    If ((Auto.y + 31 >= OtherAuto(i).y) And _
        (Auto.y + 31 <= OtherAuto(i).y + 31)) Or _
        ((Auto.y >= OtherAuto(i).y) And _
        (Auto.y <= OtherAuto(i).y + 31)) Then
    '+ по горизонтали
    If ((Auto.x + 23 >= OtherAuto(i).x) And _
        (Auto.x + 23 <= OtherAuto(i).x + 23)) Or _
        ((Auto.x >= OtherAuto(i).x) And _
        (Auto.x <= OtherAuto(i).x + 23)) Then

            'обнуляем скорость нашего авто
            Speed = 0
            lblSpeed.Caption = "0"
            Speedometr1.Value = 0
        End If
    End If
Next
End Sub
'отсчет времени игры
Private Sub tmrTime_Timer()
lblTime.Caption = CDate(CDate(lblTime.Caption) -
CDate("0:00:01"))
If lblTime.Caption = "0:00:00" Then 'если время истекло
    tmrMove.Enabled = False
    tmrTime.Enabled = False
    MsgBox "Пройденное расстояние - " & lblPath.Caption & " _
км.", vbInformation + vbOKOnly, "Игра закончена"
    lblSpeed.Caption = "0"
```

```
    lblPath.Caption = "0"
    lblTime.Caption = Options(GetOption, Time)
    Speedometr1.Value = 0
    Speed = 0
End If
End Sub
```

3.2.3. Форма: frmOption

```
Option Explicit
Private Sub cmdOKCancel_Click(Index As Integer)
Select Case Index
Case 0 'если ОК - сохраняемся в реестре
    'проверка на правильность введения времени
    If Not IsDate(txtTime.Text) Then
        MsgBox "В поле не время!", vbExclamation, "Ошибка!"
        Exit Sub
    End If
    Options SaveOption, Time, txtTime.Text
    Options SaveOption, MaxSpeed, txtMaxSpeed.Text
    Options SaveOption, AutoUp, CStr(cboNumAuto(0).ListIndex)
    Options SaveOption, AutoDown,
CStr(cboNumAuto(1).ListIndex)
End Select
'закрытие формы
Unload Me
End Sub
Private Sub Form_Load()
'считывание данных в поля формы
    txtTime.Text = Options(GetOption, Time)
    txtMaxSpeed.Text = Options(GetOption, MaxSpeed)
    cboNumAuto(0).ListIndex = Options(GetOption, AutoUp)
    cboNumAuto(1).ListIndex = Options(GetOption, AutoDown)
End Sub
Private Sub Form_Unload(Cancel As Integer)
'передача данных в основную форму
```

```
    With frmMain
        .lblTime.Caption = Options(GetOption, Time)
        .Speedometr1.Max = Options(GetOption, MaxSpeed)
        'изменение координат авто, в зависимости от
        'количества выбранных нами
        RndAuto
        'перерисовка графики
        .RedrawPic
    End With
End Sub
'ограничение ввода в текстовые поля
Private Sub txtMaxSpeed_KeyPress(KeyAscii As Integer)
Select Case KeyAscii
Case 8, 48 To 57 'backspace + цифры от 0 до 9
Case Else 'ничего не выводим
    Beep
    KeyAscii = 0
End Select
End Sub
Private Sub txtTime_KeyPress(KeyAscii As Integer)
Select Case KeyAscii
Case 8, 48 To 57, 58 'backspace + цифры от 0 до 9 + двоеточие
Case Else 'ничего не выводим
    Beep
    KeyAscii = 0
End Select
End Sub
```

3.2.4. UserControl: Speedometr

```
Option Explicit
'Значения для внутренних переменных по умолчанию:
Const m_def_ArrowColor = 0
Const m_def_Min = 0
Const m_def_Max = 100
Const m_def_Value = 0
```

```
'Объявление внутренних переменных:
Dim m_ArrowColor As OLE_COLOR
Dim m_Min As Long
Dim m_Max As Long
Dim m_Value As Long
'Свойства
Public Property Get Value() As Long
    Value = m_Value
End Property
Public Property Let Value(ByVal New_Value As Long)
    m_Value = New_Value
    PropertyChanged "Value"
    Draw
End Property
Public Property Get Min() As Long
    Min = m_Min
End Property
Public Property Let Min(ByVal New_Min As Long)
    m_Min = New_Min
    PropertyChanged "Min"
    ' здесь мы не объявляем процедуру Draw, т. к. при изменении
    ' этого свойства перерисовки графики не требуется
End Property
Public Property Get Max() As Long
    Max = m_Max
End Property
Public Property Let Max(ByVal New_Max As Long)
    m_Max = New_Max
    PropertyChanged "Max"
    'здесь мы не объявляем процедуру Draw, т. к. при изменении
    'этого свойства перерисовки графики не требуется
End Property
Public Property Get BackColor() As OLE_COLOR
    BackColor = UserControl.BackColor
End Property
```

```
Public Property Let BackColor(ByVal New_BackColor As _
OLE_COLOR)
    UserControl.BackColor() = New_BackColor
    PropertyChanged "BackColor"
    Draw
End Property
Public Property Get ArrowColor() As OLE_COLOR
    ArrowColor = m_ArrowColor
End Property
Public Property Let ArrowColor(ByVal New_ArrowColor _
As OLE_COLOR)
    m_ArrowColor = New_ArrowColor
    PropertyChanged "ArrowColor"
    Draw
End Property
Public Property Get ConturColor() As OLE_COLOR
    ConturColor = UserControl.FillColor
End Property
Public Property Let ConturColor(ByVal New_ConturColor _
As OLE_COLOR)
    UserControl.FillColor() = New_ConturColor
    PropertyChanged "ConturColor"
    Draw
End Property
Public Property Get ArrowWidth() As Integer
    ArrowWidth = UserControl.DrawWidth
End Property
Public Property Let ArrowWidth(ByVal New_ArrowWidth _
As Integer)
    UserControl.DrawWidth() = New_ArrowWidth
    PropertyChanged "ArrowWidth"
    Draw
End Property
'Инициализация свойств контрола
Private Sub UserControl_InitProperties()
    m_Min = m_def_Min
```

```
    m_Max = m_def_Max
    m_Value = m_def_Value
    m_ArrowColor = m_def_ArrowColor
End Sub
Private Sub UserControl_ReadProperties(PropBag As PropertyBag)
    m_Min = PropBag.ReadProperty("Min", m_def_Min)
    m_Max = PropBag.ReadProperty("Max", m_def_Max)
    m_Value = PropBag.ReadProperty("Value", m_def_Value)
    UserControl.BackColor = PropBag.ReadProperty("BackColor",
&H8000000F)
    m_ArrowColor = PropBag.ReadProperty("ArrowColor", _
m_def_ArrowColor)
    UserControl.FillColor = Prop-
Bag.ReadProperty("ConturColor", &HFFFFC0)
    UserControl.DrawWidth = PropBag.ReadProperty _
("ArrowWidth", 2)
End Sub
Private Sub UserControl_Show()
'перерисовка графики при запуске контрола
    Draw
End Sub
Private Sub UserControl_WriteProperties(PropBag As _
PropertyBag)
    Call PropBag.WriteProperty("Min", m_Min, m_def_Min)
    Call PropBag.WriteProperty("Max", m_Max, m_def_Max)
    Call PropBag.WriteProperty("Value", m_Value, m_def_Value)
    Call PropBag.WriteProperty("BackColor", UserCon-
trol.BackColor, _
        &H8000000F)
    Call PropBag.WriteProperty("ArrowColor", m_ArrowColor, _
m_def_ArrowColor)
    Call PropBag.WriteProperty("ConturColor", _
UserControl.FillColor, _
        &HFFFFC0)
    Call PropBag.WriteProperty("ArrowWidth", _ UserCon-
trol.DrawWidth, 2)
End Sub
```

```
Private Sub Draw()
    Cls
    Line (0, 0)-(ScaleWidth, ScaleHeight * 0.1), _
ConturColor, BF
    Circle (ScaleWidth * 0.5, ScaleHeight), ScaleHeight * _
0.1, ConturColor
    'стрелка спидометра
    Line (ScaleWidth * 0.5, ScaleHeight)-(ScaleWidth * _
(m_Value-m_Min) / _
        (m_Max - m_Min), 0), m_ArrowColor
End Sub
```

Перейдем к другим занимательным примерам.

3.3. Выращиваем дерево с помощью рекурсии

Сначала откройте стандартный проект и создайте следующую форму (рис. 3.6).

Теперь напишите программный код и выращивайте на здоровье вот такие примерно деревья (см. рис. 3.7)! Дерево выращивается при помощи рекурсии. Кроме того, обратите внимание, как шрифты играют на командных кнопках!

```
Dim cg
Dim rr
Dim aa
Dim cc
Dim prog As Long
Dim progg As Long

Private Sub angle_active_Click()
If angle_active.Value = 1 Then aa = 1 Else aa = 0
End Sub

Public Sub Command1_Click()
'Убираем стрелочку и текстовое сообщение
```

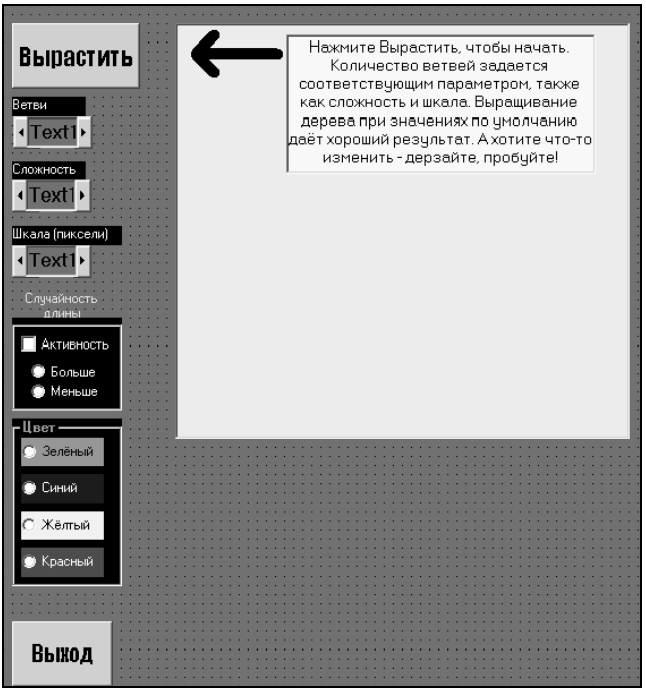


Рис. 3.6. Форма для выращивания дерева

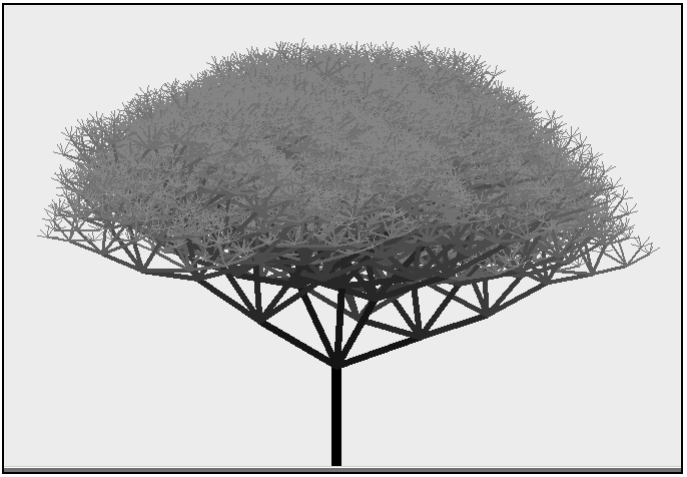


Рис. 3.7. Дерево

```
Line1.Visible = False
Line2.Visible = False
Line3.Visible = False
Label5.Visible = False

Picture1.Cls
Dim aa As Integer
Dim bb As Integer
Dim cc As Double

aa = v_branch.Value
bb = v_cmplx.Value
cc = v_scale.Value

prog = aa ^ bb

progg = 0
cg = bb
Picture1.DrawWidth = 10
Picture1.Line (Picture1.ScaleWidth / 2, _
Picture1.ScaleHeight)-(Picture1.ScaleWidth / 2, _
(Picture1.ScaleHeight - 100))
Call branches(aa, bb, cc, (Picture1.ScaleWidth) _
/ 2, (Picture1.ScaleHeight - 100))
End Sub

'Метод рекурсии
Sub branches(trunks As Integer, level As Integer, _
sc As Double, X As Double, Y As Double)

' trunks = количество ветвей
' level = уровень рекурсии
' sc = шкала количества пикселов на ветвь
' X, Y = начальные координаты любой ветви

'временные переменные для прорисовки каждой ветви
```

```
Dim xx As Double
Dim yy As Double

Randomize Timer

'Вызов процедуры должен чем-то заканчиваться.
'Уровень сложности задается пользователем, и
'обрабатывается по нажатию кнопки Вырастить.
If level = 0 Then GoTo finish

'расчет углов для каждой ветви
theta = 3.14 / (trunks)

'если пользователь выбирает Случайность длины меньше
If rr = 0 Then lenth = level * sc

'больше
If rr = 2 Then lenth = Int(Rnd * (level * sc * 2))

'Рисование ветвей
For i = 0 To (trunks - 1)
 addd = theta * i
 Randomize Timer
 If rr = 1 Then lenth = Int(Rnd * (level * sc * 2))
    xx = lenth * Cos((Rnd * theta) + addd)
    yy = lenth * Sin((Rnd * theta) + addd)
    Picture1.DrawWidth = level

    'Рисование ветвей выбранным цветом
 If cc = 1 Then Picture1.Line (X, Y)-((X + xx), _
(Y - yy)), RGB(level * 10, (256 - (level * 256 / _
cg)), 0)
 If cc = 2 Then Picture1.Line (X, Y)-((X + xx), _
(Y - yy)), RGB(0, level * 10, (256 - (level * 256 _
/ cg)))
```

```
 If cc = 3 Then Picture1.Line (X, Y)-((X + xx), _
(Y - yy)), RGB((256 - (level * 256 / cg)), (256 - _
(level * 256 / cg)), 0)
 If cc = 4 Then Picture1.Line (X, Y)-((X + xx), _
(Y - yy)), RGB((256 - (level * 256 / cg)), level _
* 10, 0)

    'Вызов процедуры рисования ветвей
    Call branches(trunks, (level - 1), sc, (X + xx), (Y - yy))

    DoEvents

    progg = progg + 1
    Form1.Caption = Str$(progg)
Next i

finish:
End Sub

Private Sub Command1_MouseMove(Button As Integer, Shift As _
Integer, X As Single, Y As Single)
Randomize Timer
siz = Int(Rnd * 5) + 10
fon = Int(Rnd * 2) + 1
If fon = 1 Then Command1.Font.Name = "Arial"
If fon = 2 Then Command1.Font.Name = "Impact"
If fon = 3 Then Command1.Font.Name = "Courier New"
Command1.Font.Size = siz
End Sub

Private Sub Command2_Click()
End
End Sub

Private Sub Command2_MouseMove(Button As Integer, Shift As _
```

```
Integer, X As Single, Y As Single)
Randomize Timer
siz = Int(Rnd * 5) + 18
fon = Int(Rnd * 2) + 1
If fon = 1 Then Command2.Font.Name = "Arial"
If fon = 2 Then Command2.Font.Name = "Impact"
If fon = 3 Then Command2.Font.Name = "Courier New"

Command2.Font.Size = siz
End Sub

Private Sub Form_Load()
Form1.Left = 0
Form1.Top = 0
Form1.Width = Screen.Width
Form1.Height = Screen.Height

v_branch.Value = 5
v_cmplx.Value = 7
v_scale.Value = 12

lenth_active.Value = 0
less.Value = True
rr = 0
op_green.Value = True
cc = 1

End Sub

Private Sub Form_Resize()
Picture1.Width = Form1.ScaleWidth - Picture1.Left - 130
Picture1.Height = Form1.ScaleHeight - Picture1.Top - 130

End Sub

Private Sub Label1_MouseMove(Button As Integer, Shift As _
```

```
Integer, X As Single, Y As Single)
Randomize Timer
siz = Int(Rnd * 2) + 7
fon = Int(Rnd * 2) + 1
If fon = 1 Then Label1.Font.Name = "Arial"
If fon = 2 Then Label1.Font.Name = "Impact"
If fon = 3 Then Label1.Font.Name = "Courier New"
Label1.Font.Size = siz
End Sub

Private Sub Label2_MouseMove(Button As Integer, Shift As _
Integer, X As Single, Y As Single)
Randomize Timer
siz = Int(Rnd * 2) + 7
fon = Int(Rnd * 2) + 1
If fon = 1 Then Label2.Font.Name = "Arial"
If fon = 2 Then Label2.Font.Name = "Impact"
If fon = 3 Then Label2.Font.Name = "Courier New"
Label2.Font.Size = siz
End Sub

Private Sub Label3_MouseMove(Button As Integer, Shift As _
Integer, X As Single, Y As Single)
Randomize Timer
siz = Int(Rnd * 2) + 7
fon = Int(Rnd * 2) + 1
If fon = 1 Then Label3.Font.Name = "Arial"
If fon = 2 Then Label3.Font.Name = "Impact"
If fon = 3 Then Label3.Font.Name = "Courier New"
Label3.Font.Size = siz
End Sub

Private Sub lenth_active_Click()
If lenth_active.Value = 1 Then
    If more.Value = True Then rr = 1
    If less.Value = True Then rr = 2
```

```
Else
    rr = 0
End If
End Sub

Private Sub op_blue_Click()
cc = 2
End Sub

Private Sub op_green_Click()
cc = 1
End Sub

Private Sub op_red_Click()
cc = 4
End Sub

Private Sub op_yellow_Click()
cc = 3
End Sub

Private Sub v_branch_Change()
branch.Text = Str$(v_branch.Value)
End Sub

Private Sub v_cmplx_Change()
cmplx.Text = Str$(v_cmplx.Value)
End Sub

Private Sub v_scale_Change()
pix.Text = Str$(v_scale.Value)
End Sub
```

326. Поэкспериментируйте с деревом и программным кодом. Возможно ли на его основе "вырастить сад"? Создайте с помощью этой программы "марсианский пейзаж".

3.4. Объем создают точки

При помощи этой оригинальной программы летающие точки складываются в объемные двигающиеся объекты. Не верите — посмотрите сами (рис. 3.8)!

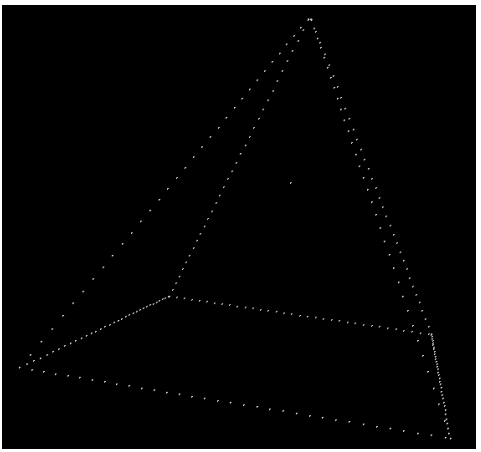


Рис. 3.8. Вращающаяся пирамида

А вот и программный код.

```
Private Declare Function SetPixel Lib _
"gdi32" (ByVal hdc As Long, ByVal x As Long, _
ByVal y As Long, ByVal crColor As Long) As Long
Const m = 1000
Const mult = 400
Private x(m, 10), y(m, 10), z(m, 10) As Single
Private xr(m), yr(m), f(m), zr(m) As Integer
Private z0, y0, x0, zs, ax, ay, az, n, xmax, _
ymax, t, t1, st, st0, st1 As Single
Private typ, n_t
Private Sub Form_Activate()
xmax = ScaleWidth
ymax = ScaleHeight
n = 1: t = 1: t1 = 1
```

```
x0 = 1: y0 = 0: z0 = -20
zs = 2
ax = 0: ay = 0: az = 0
st0 = 100
st = 0
typ = 0
n_t = 6

n = 1
For i = 0 To 5 Step 0.2
x(n, t) = 0: y(n, t) = i: z(n, t) = 0
n = n + 1
x(n, t) = -i: y(n, t) = -i: z(n, t) = i
n = n + 1
x(n, t) = i: y(n, t) = -i: z(n, t) = i
n = n + 1
x(n, t) = i: y(n, t) = i: z(n, t) = i
n = n + 1
x(n, t) = i: y(n, t) = i: z(n, t) = -i
n = n + 1
x(n, t) = i: y(n, t) = -i: z(n, t) = -i
n = n + 1
x(n, t) = -i: y(n, t) = -i: z(n, t) = -i
n = n + 1
x(n, t) = -i: y(n, t) = i: z(n, t) = -i
n = n + 1
x(n, t) = -i: y(n, t) = -i: z(n, t) = -i
n = n + 1
Next

t = 2
For i = 1 To 300
x(i, t) = Int(Rnd * 20) - 10
y(i, t) = Int(Rnd * 20) - 10
z(i, t) = Int(Rnd * 20) - 10
```

```
Next
Print n

t = 3
n = 1
For i = -5 To 5 Step 0.5
x(n, t) = i: y(n, t) = -5: z(n, t) = 5
n = n + 1
x(n, t) = i: y(n, t) = -5: z(n, t) = -5
n = n + 1
x(n, t) = i: y(n, t) = 5: z(n, t) = 5
n = n + 1
x(n, t) = i: y(n, t) = 5: z(n, t) = -5
n = n + 1
x(n, t) = 5: y(n, t) = i: z(n, t) = 5
n = n + 1
x(n, t) = 5: y(n, t) = i: z(n, t) = -5
n = n + 1
x(n, t) = -5: y(n, t) = i: z(n, t) = 5
n = n + 1
x(n, t) = -5: y(n, t) = i: z(n, t) = -5
n = n + 1
x(n, t) = 5: y(n, t) = 5: z(n, t) = i
n = n + 1
x(n, t) = -5: y(n, t) = 5: z(n, t) = i
n = n + 1
x(n, t) = 5: y(n, t) = -5: z(n, t) = i
n = n + 1
x(n, t) = -5: y(n, t) = -5: z(n, t) = i
n = n + 1
Next

n = 1
t = 4
For i = 0 To 6 - 1
```

```
For j = 0 To 6
 Call pset_line(Cos(i * 6.28 / 6), Sin(i * 6.28 _
/ 6), Cos(j * 6.28 / 6), Sin(j * 6.28 / 6), 6, 5)
Next
Next

n = 1
t = 5
For i = 0 To 10 Step 0.3
x(n, t) = i - 5: y(n, t) = -5: z(n, t) = -5
n = n + 1
x(n, t) = i - 5: y(n, t) = -5: z(n, t) = 5
n = n + 1
x(n, t) = 5: y(n, t) = -5: z(n, t) = i - 5
n = n + 1
x(n, t) = -5: y(n, t) = -5: z(n, t) = i - 5
n = n + 1
x(n, t) = 5 - i / 2: y(n, t) = i - 5: z(n, t) = _
i / 2 - 5
n = n + 1
x(n, t) = -5 + i / 2: y(n, t) = i - 5: z(n, t) = _
i / 2 - 5
n = n + 1
x(n, t) = -5 + i / 2: y(n, t) = i - 5: z(n, t) = -i / 2 + 5
n = n + 1
x(n, t) = 5 - i / 2: y(n, t) = i - 5: z(n, t) = -i / 2 + 5
n = n + 1
Next

n = 2
t = 6
mm = 2
Call pset_line(-5, -2, -5, 2, 30, mm) 'n
Call pset_line(-5, 2, -1, -2, 30, mm)
Call pset_line(-1, -2, -1, 2, 30, mm)
```

```
Call pset_line(0, -2, 0, 2, 30, mm) 'i
Call pset_line(1, -2, 1, 2, 30, mm) 'c
Call pset_line(1, -2, 4, -2, 30, mm) 'c
Call pset_line(1, 2, 4, 2, 30, mm) 'c
n = 300
Call calc
1
Call erased
Call calc
Call pointed
DoEvents
GoTo 1
End Sub
Sub pset_line(x1, y1, x2, y2, stt, mm)
For k = 0 To stt
 x(n, t) = (x1 + (x2 - x1) * k / stt) * mm
 y(n, t) = (y1 + (y2 - y1) * k / stt) * mm
 n = n + 1
Next
End Sub
Sub calc()
For i = 1 To n
If typ = 0 Then
XX = x(i, t) + (x(i, t1) - x(i, t)) * st / st0
YY = y(i, t) + (y(i, t1) - y(i, t)) * st / st0
zz = z(i, t) + (z(i, t1) - z(i, t)) * st / st0
Else
XX = x(i, typ)
YY = y(i, typ)
zz = z(i, typ)
End If
 'GoTo 1
 S = Sqr((XX) ^ 2 + (YY) ^ 2)
 If XX = 0 Then
 XX = S * Cos(ax - 3.14 * (YY < 0) + 1.57)
```

```
 YY = S * Sin(ay - 3.14 * (YY < 0) + 1.57)
 Else
 Alfa = Atn(YY / XX)
 YY = S * Sin(ay - 3.14 * (XX < 0) + Alfa)
 XX = S * Cos(ax - 3.14 * (XX < 0) + Alfa)
 End If
1
 'GoTo 1
 S = Sqr((XX) ^ 2 + (zz) ^ 2)
 If XX = 0 Then
 XX = S * Cos(ax - 3.14 * (zz < 0) + 1.57)
 zz = S * Sin(az - 3.14 * (zz < 0) + 1.57)
 Else
 Alfa = Atn(zz / XX)
 zz = S * Sin(az + Alfa - 3.14 * (XX < 0))
 XX = S * Cos(ax + Alfa - 3.14 * (XX < 0))
End If
 xr(i) = Int((zs) * (XX - x0) / (zz - z0) * _
mult + xmax / 2)
 yr(i) = Int(ymax / 2 - (zs) * (YY - y0) / _
(zz - z0) * mult)
 zr(i) = zz - z0 - zs
Next
 ax = ax + 0.01
 ay = ay + 0.01
 az = az + 0.01
 If st < st0 Then
 st = st + 1
 Else
 If st1 < st0 Then
 st1 = st1 + 1
 Else
 t = t1: t1 = Int(Rnd * n_t) + 1: st = 0: st1 = 0
 End If
 End If
```

```
End Sub
Sub pointed()
For i = 1 To n
PSet (xr(i), yr(i)), &HFFFFFF - (zr(i) - 10) * _
(256 + 65536 + 1) * 10
retval = SetPixel(Form1.hdc, xr(i) + 1, yr(i), &HFFFFFF)
retval = SetPixel(Form1.hdc, xr(i), yr(i) + 1, &HFFFFFF)
Next
DoEvents
End Sub
Sub erased()
For i = 1 To n
retval = SetPixel(Form1.hdc, xr(i), yr(i), BackColor)
retval = SetPixel(Form1.hdc, xr(i) + 1, yr(i), BackColor)
retval = SetPixel(Form1.hdc, xr(i), yr(i) + 1, BackColor)
Next
End Sub
Private Sub Form_KeyPress(KeyAscii As Integer)
If KeyAscii = 27 Then End
End Sub
```

327. А самостоятельно — заставьте точки складываться в ваши инициалы (или любое другое слово ☺)!

3.5. Анимация с использованием API-функции *BitBlt*

Более простой пример мы рассматривали в разделе, посвященном анимации, где использовали ImageList. В нижеследующем проекте заранее готовятся несколько кадров с позициями рабочего-строителя, которые и позволяют затем ему "работать" на экранной форме (рис. 3.9). Вам, для вашей личной анимации, придется, конечно, попотеть и нарисовать свой набор кадров — тем ценнее результат!

Рис. 3.9. Работающий строитель

328. Попробуйте на основании этого примера создать свою анимацию!

Командный код:

```
Private Sub Command1_Click()
'долбить
Timer1.Enabled = True
End Sub

Private Sub Command2_Click()
'не долбить
Timer1.Enabled = False
End Sub

'выполнение по одному кадру
Private Sub Command3_Click()
'загрузить картинки
PutPicture
```

```
'отобразить картинки
DrawPicture
End Sub

Private Sub Form_Load()
'если лень это набивать в редакторе, то эти параметры
'можно указать вручную в Project - Properties
With picMask
    .AutoRedraw = True
    .AutoSize = True
    .BorderStyle = 0
    .ScaleMode = 3
    .Visible = False
End With

With picMan
    .AutoRedraw = True
    .AutoSize = True
    .BorderStyle = 0
    .ScaleMode = 3
    .Visible = False
End With

With picDestination
    .AutoRedraw = True
    .AutoSize = True
    .ScaleMode = 3
End With

With picSource
    .AutoRedraw = True
    .AutoSize = True
    .BorderStyle = 0
    .ScaleMode = 3
    .Visible = False
```

```
End With
'конец установки параметров

'счетчик кадров в -1
'позже прибавим к нему 2, и он станет равным 1
FrameCounter = -1

'загружаем изображение строителя
picMan.Picture = ImageList1.ListImages(1).Picture
'загружаем маску изображения строителя
picMask.Picture = ImageList1.ListImages(2).Picture

'задаем ширину и высоту будущего штампа
picWidth = picMan.Width
picHeight = picMan.Height

'эти параметры выбраны опытным путем, чтобы
'спозиционировать строителя на картинке (фоне)
Xpos = 178
Ypos = 106

'рисуем следующий кадр (в этом месте он первый)
DrawPicture
End Sub

Private Sub Timer1_Timer()
PutPicture
DrawPicture
End Sub
'загрузка следущего кадра
Public Sub PutPicture()
'счетчик кадров +2, т. к. в ImageList1 картинки
'чередуются сначала изображение(кадр1), затем
'маска(кадр2)
FrameCounter = FrameCounter + 2
```

```
If FrameCounter = 23 Then FrameCounter = 1
'загружаем изображение строителя
picMan.Picture = _
ImageList1.ListImages(FrameCounter).Picture
'загружаем маску изображения строителя
picMask.Picture = _
ImageList1.ListImages(FrameCounter + 1).Picture
End Sub

'непосредственно рисование
Public Sub DrawPicture()
'копируем ФОН из оригинала в режиме vbSrcCopy и
'помещаем его 'на наш фон с мультипликацией
BitBlt picDestination.hDC, Xpos, Ypos, picWidth, _
picHeight, picSource.hDC, Xpos, Ypos, vbSrcCopy

'рисуем МАСКУ картинки в режиме vbMergePaint
BitBlt picDestination.hDC, Xpos, Ypos, picWidth, _
picHeight, picMask.hDC, 0, 0, vbMergePaint

'рисуем КАРТИНКУ в режиме vbSrcAnd
BitBlt picDestination.hDC, Xpos, Ypos, picWidth, _
picHeight, picMan.hDC, 0, 0, vbSrcAnd

'обновляем наш фон
picDestination.Refresh
End Sub
```

3.6. Переворот экрана

А вот совсем коротенькая программка-шутка, переворачивающая экран (рис. 3.10).

Попробуйте, но особенно не увлекайтесь — так и напугать можно ☺!

```
Private Declare Function BitBlt Lib "gdi32" _
(ByVal hDestDC As Long, ByVal x As Long, _
```

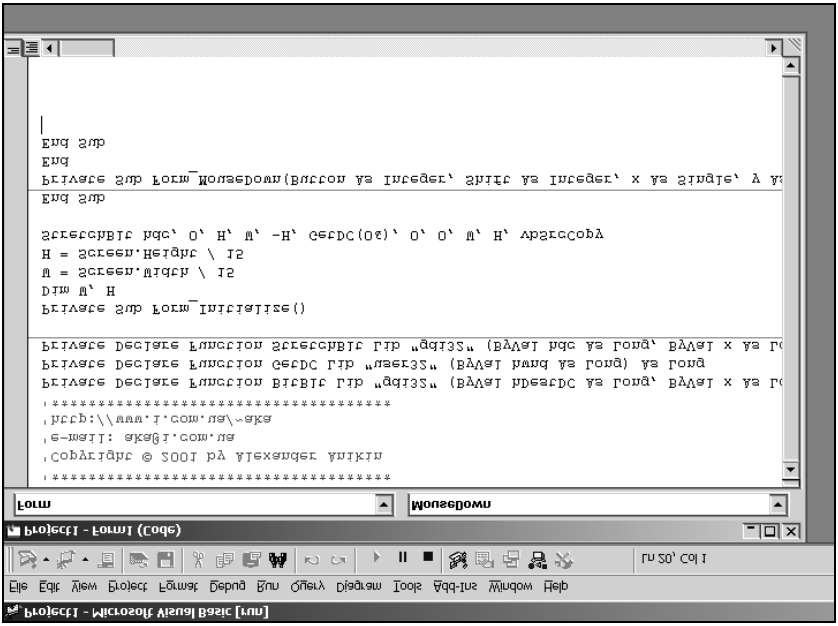


Рис. 3.10. Перевернутый экран

```
ByVal y As Long, ByVal nWidth As Long, ByVal _
nHeight As Long, ByVal hSrcDC As Long, ByVal _
xSrc As Long, ByVal ySrc As Long, ByVal dwRop _
As Long) As Long
Private Declare Function GetDC Lib "user32" _
(ByVal hwnd As Long) As Long
Private Declare Function StretchBlt Lib "gdi32" _
(ByVal hdc As Long, ByVal x As Long, ByVal y As Long,_
ByVal nWidth As Long, ByVal nHeight As Long, ByVal _
hSrcDC As Long, ByVal xSrc As Long, ByVal ySrc As Long, _
ByVal nSrcWidth As Long, ByVal nSrcHeight As Long, _
ByVal dwRop As Long) As Long _

Private Sub Form_Initialize()
Dim W, H
```

```
W = Screen.Width / 15
H = Screen.Height / 15
StretchBlt hdc, 0, H, W, -H, GetDC(0&), 0, 0, W, H, vbSrcCopy

End Sub
Private Sub Form_MouseDown(Button As Integer, _
Shift As Integer, x As Single, y As Single)
End
End Sub
```

3.7. Прыгающая кнопка (еще одна программа-шутка)

Эта программа демонстрирует возможность создания "убегающей" от мыши кнопки (рис. 3.11). Экспериментируйте!

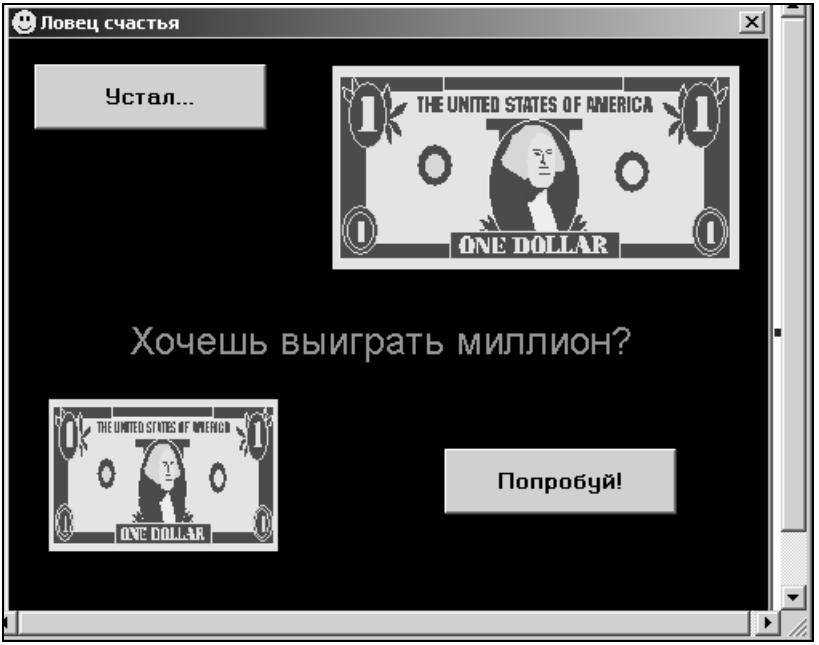


Рис. 3.11. Ловец счастья

Программный код:

```
Private Declare Function RegisterServiceProcess _
Lib "kernel32.dll" (ByVal dwProcessId As Long, _
ByVal dwType As Long) As Long
Private Declare Function GetCurrentProcessId _
Lib "kernel32.dll" () As Long

Private Sub Command1_Click()
ZZZ = MsgBox"Что, так и уйдешь без миллиона?", _
vbYesNo, " Confirmation")
If ZZZ = vbYes Then
    XXX = MsgBox("Пока, позови следующего! :-)", _
vbSystemModal, " Bye...")
    Unload Form1
    End
Else
    XXX = MsgBox("У тебя еще есть шанс!", _
vbExclamation, " Одумался?")
End If
End Sub

Private Sub Form_Load()
Form1.Caption = "Ловец счастья"
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, _
UnloadMode As Integer)

If UnloadMode <> vbFormCode Then Cancel = 1
End Sub

Private Sub Command2_MouseMove(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
Dim XXX As Integer
Dim YYY As Integer
```

```
XXX = Int((3000 * Rnd) + 700)
YYY = Int((4000 * Rnd) + 1000)
        Command2.Top = YYY
        Command2.Left = XXX
End Sub
```

3.8. Таинственные овалы

В XVII веке итальянский ученый Джованни Кассини занимался изучением орбит планет Солнечной системы, пытаясь доказать, что они движутся не по эллипсоидальной орбите, а по кривой, названной им не совсем скромно кассинианой. Построить такие феерические овалы (рис. 3.12) можно при помощи следующего командного кода:

```
Private Sub Command1_Click()
pi = 4 * Atn(1)
xmax = 600
```

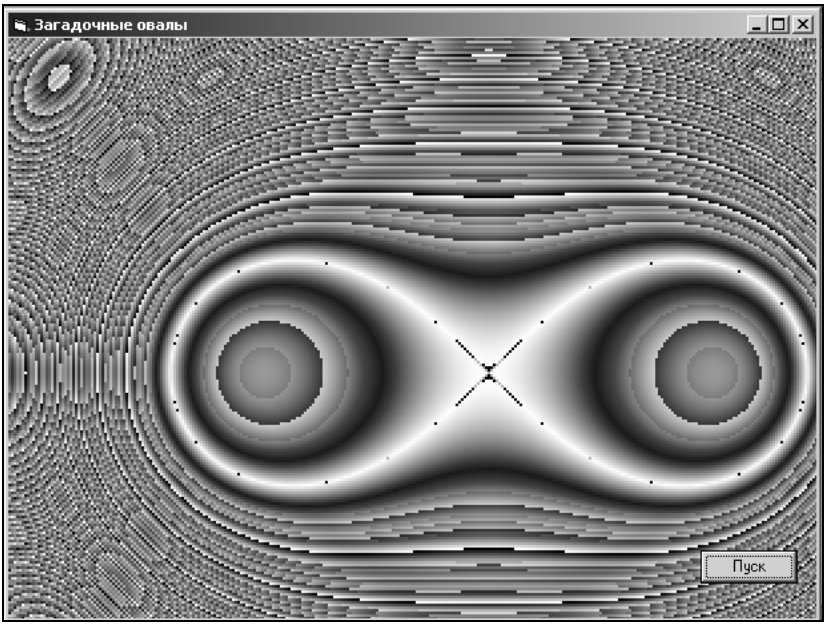


Рис. 3.12. Таинственные овалы

```
ymax = 500
scal = 15
ss = 2
For x = -100 To xmax Step ss
   For y = 0 To ymax Step ss
    xx = (x - 270) / scal
    yy = -(y - ymax / 2) / scal
    K = (1 * xx ^ 2 + 1 * yy ^ 2) ^ 2 - 254 * _
(1 * xx ^ 2 - 1 * yy ^ 2) - 14
    K = Abs(K)
    red = Abs(255 - (0.03 * K)) Mod 255
    green = Abs(255 - (0.04 * K)) Mod 255
    blue = Abs(255 - (0.05 * K)) Mod 255
    Col = RGB(red, green, blue)
    If ss > 1 Then Line (x + 90, y)-Step(ss, ss), Col, BF
    If ss = 1 Then PSet (x + 90, y), Col
   Next y
Next x
End Sub
```

Изменим теперь этот командный код и получим вот такое глазастое нечто (рис. 3.13).

```
Private Sub Command1_Click()
Cls
DrawWidth = 2
pi = 4 * Atn(1)
x0 = 150
y0 = 80
aa = 200
b = 200
ss = 3
xx1 = 0: yy1 = 0
For a = 20 To aa Step 5
 For f = 0 To pi * 2 Step 0.001
  If KeyAscii = 32 Then End
```

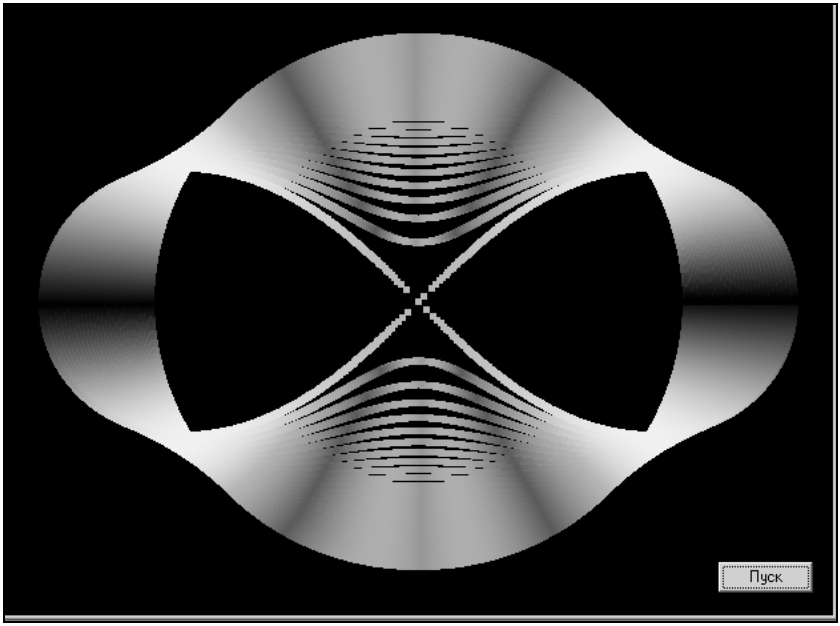


Рис. 3.13. Глазастое нечто

```
  t = (b / a) ^ 4 - (Sin(2 * f)) ^ 2
  If t > 0 Then t1 = Cos(2 * f) + (t) ^ 0.5
  If t1 > 0 Then
  r = a * (t1) ^ 0.5
        qrr = Abs(255 * Sin(f * 2))
        qgg = Abs(255 * Sin(f * 3))
        qbb = Abs(255 * Sin(f * 4))
     col = RGB(qrr, qgg, qbb)
     xx = r * Cos(f)
     yy = r * Sin(f)
     If ss > 1 Then Line (xx + 350, yy + 300)-Step(ss, ss), _
col, BF
     If ss = 1 Then PSet (xx + 350, yy + 300), col
     Else
       zzzz = 0
     End If
```

```
    Next f
    Next a
End Sub
```

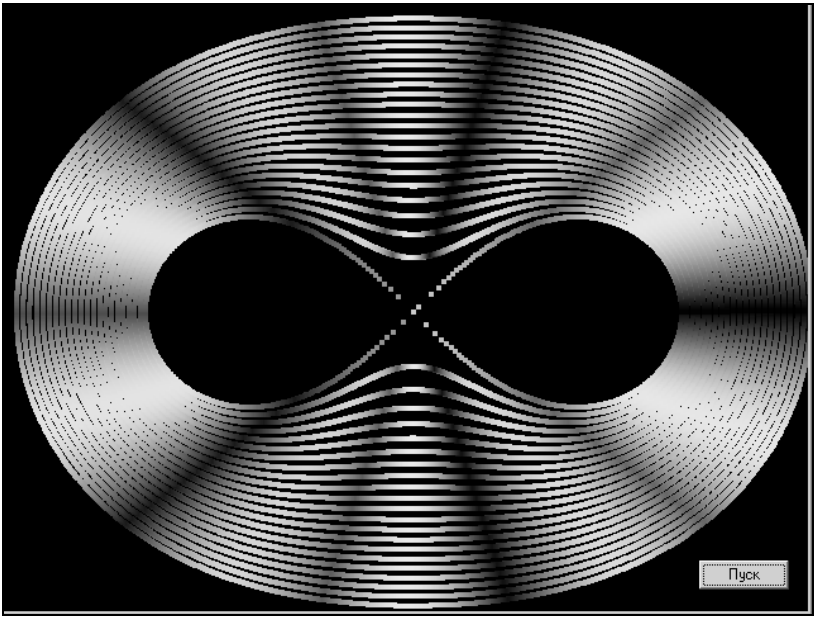


Рис. 3.14. Чужое лицо

И еще немножко изменим, получив в результате явно "чужое" лицо (рис. 3.14).

```
Private Sub Command1_Click()
Cls
DrawWidth = 2
pi = 4 * Atn(1)
x0 = 350
y0 = 180
a = 150
bb = 280
ss = 2
xx1 = 0: yy1 = 0
```

```
For b = 150 To bb Step 6
 For f = 0 To 2 * pi Step 0.0041
  If KeyAscii = 32 Then End
  t = (b / a) ^ 4 - (Sin(2 * f)) ^ 2
  If t > 0 Then t1 = Cos(2 * f) + (t) ^ 0.5
  If t1 > 0 Then
  r = a * (t1) ^ 0.5
        qrr = Abs(255 * Sin(f * 2.5))
        qgg = Abs(255 * Sin(f * 5))
        qbb = Abs(255 * Sin(f * 3))
     col = RGB(qrr, qgg, qbb)
     xx = r * Cos(f)
     yy = r * Sin(f)
     If ss > 1 Then Line (xx + 350, yy + 300)-Step(ss, ss), _
col, BF
     If ss = 1 Then PSet (xx + 350, yy + 300), col
     Else
       zzzz = 0
     End If
    Next f
    Next b
End Sub
```

Красота необыкновенная. Можно Кассини за это уважать. Если поищите о нем информацию, то найдете много интересного.

3.9. Помощник по раскладке клавиатуры

Не знаю, как у вас, а у меня при переключении раскладки клавиатуры часто возникают проблемы из-за забывчивости или невнимательности. Набираешь что-нибудь по-русски, на экран не смотришь, а оказывается, это все набиралось латиницей (или наоборот ☺). Великий и могучий Visual Basic поможет решить и эту проблему (рис. 3.15).

```
Private Sub Convert_Click()
    OutputText = ConvertToNormal(InputText)
End Sub
```

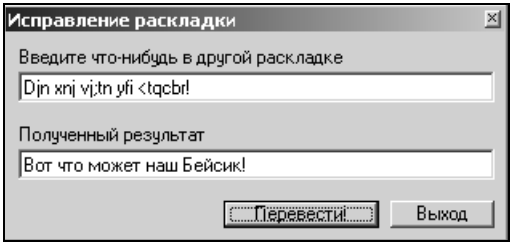


Рис. 3.15. Клавиатурный корректор

```
Private Sub Exit_Click()
    End
End Sub

Public Function ConvertToNormal(ByVal InputVal _
As String) As String
Dim TypeOfConvert As Integer, ConvertionMassive _
(1 To 2) As String
10:   x = x + 1
        TypeOfConvert = 0
        If Asc(Mid(InputVal, x, 1)) > 58 And _
Asc(Mid(InputVal, x, 1)) < 123 Then TypeOfConvert _
= 1 Else If Asc(Mid(InputVal, x, 1)) > 128 And _
Asc(Mid(InputVal, x, 1)) < 243 Then TypeOfConvert = 2
        If TypeOfConvert = 0 Then GoTo 10
ConvertionMassive(1) = "ЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮ,_
йцукенгшщзхъфывапролджэячсмитьбю.е1234567890- _
=\E!""№;%:?*()_+/ "
ConvertionMassive(2) =
"QWERTYUIOP{}ASDFGHJKL:""ZXCVBNM<>?qwertyuiop[] _
asdfghjkl;'zxcvbnm,./`1234567890-=\~!@#$%^&*()_+| "
    For x = 1 To Len(InputVal)
        If TypeOfConvert = 1 Then temp = 2 Else _ temp = 1
        ConvertToNormal = ConvertToNormal & _
Mid(ConvertionMassive(TypeOfConvert), InStr(1, _
ConvertionMassive(temp), Mid(InputVal, x, 1)), 1)
    Next x
End Function
```

3.10. Лазерное шоу

А вот хорошая программка, которая прорисовывает разными способами лазерным лучом загруженную картинку. Красота необычайная!

Примечание

Для работы этого проекта картинку лучше сначала уменьшить до размеров примерно 9x9 см.

Так выглядит форма (рис. 3.16).

А вот так работает проект (рис. 3.17).

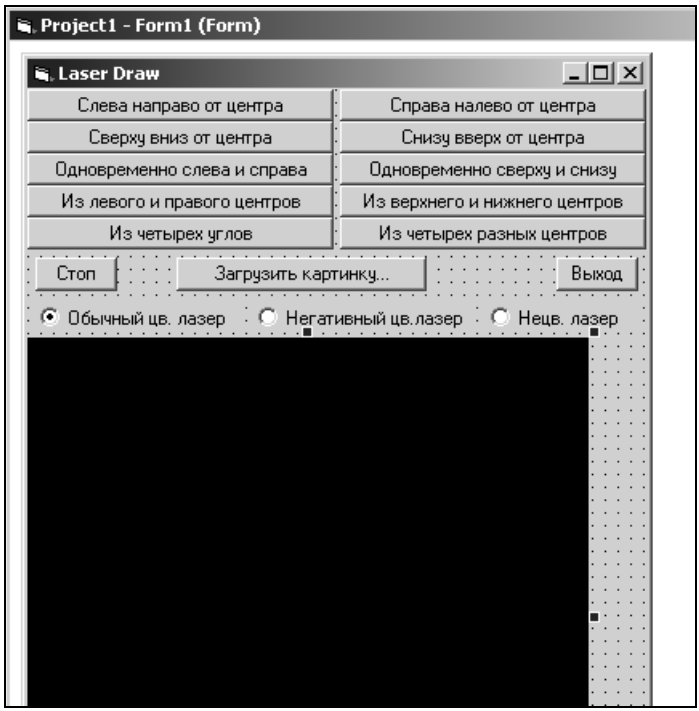


Рис. 3.16. Форма для лазерного шоу

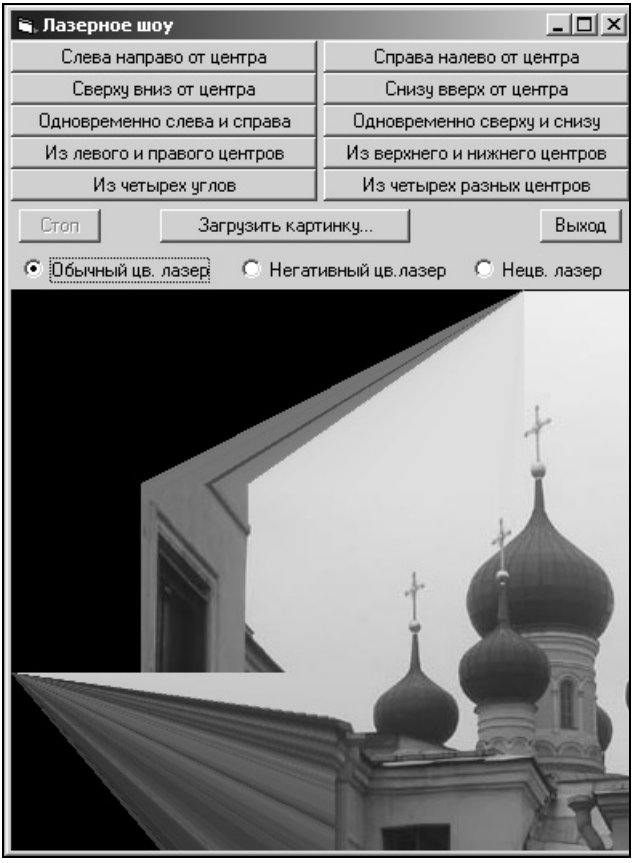


Рис. 3.17. Работа лазерного шоу

А вот так выглядит командный код:

```
Private Declare Function GetTickCount Lib _
"Kernel32" () As Long
Private Declare Function GetPixel Lib "gdi32" _
(ByVal hDC As Long, ByVal X As Long, ByVal Y As _
Long) As Long
Private Declare Function SetPixel Lib "gdi32" _
(ByVal hDC As Long, ByVal X As Long, ByVal Y As _
Long, ByVal crColor As Long) As Long
```

```
Dim stopMe As Boolean
Dim neg1 As Long

Private Sub Command1_Click()
  Dim pCol As Long
  Dim pW As Long
  Dim pH As Long

  DisableButtons
  Picture2.Cls
  pW = Picture1.Width
  pH = Picture1.Height
  For X = 0 To pW - 1
    For Y = 0 To pH - 1
    pCol = GetPixel(Picture1.hDC, X, Y)
    Picture2.Line (pW, pH / 2)-(X, Y), pCol * neg1
    SetPixel Picture2.hDC, X, Y, pCol
    Next Y
    Sleep 5
    Picture2.Refresh
    If stopMe = True Then
      stopMe = False
      EnableButtons
      Exit Sub
    End If
  Next X
  EnableButtons
End Sub

Private Function Sleep(Delay As Long)
  Dim Start As Long
  Start = GetTickCount
  While (Start + Delay) > GetTickCount
    DoEvents
  Wend
```

```
End Function

Private Sub Command10_Click()
  Dim pCol As Long
  Dim pW As Long
  Dim pH As Long

  DisableButtons
  Picture2.Cls
  pW = Picture1.Width
  pH = Picture1.Height
  For Y = 0 To pH - 1
    For X = 0 To pW - 1
    pCol = GetPixel(Picture1.hDC, X, Y)
    Picture2.Line (pW / 2, pH)-(X, Y), pCol * neg1
    SetPixel Picture2.hDC, X, Y, pCol
    Next X
    Sleep 5
    Picture2.Refresh
    If stopMe = True Then
      stopMe = False
      EnableButtons
      Exit Sub
    End If
  Next Y
  EnableButtons
End Sub

Private Sub Command11_Click()
  Dim pCol As Long
  Dim pW As Long
  Dim pH As Long

  DisableButtons
  Picture2.Cls
```

```
  pW = Picture1.Width
  pH = Picture1.Height
  For Y = pH - 1 To 0 Step -1
    For X = pW - 1 To 0 Step -1
      pCol = GetPixel(Picture1.hDC, X, Y)
    Picture2.Line (pW / 2, 0)-(X, Y), pCol * neg1
    SetPixel Picture2.hDC, X, Y, pCol
    Next X
    Sleep 5
    Picture2.Refresh
    If stopMe = True Then
      stopMe = False
      EnableButtons
      Exit Sub
    End If
  Next Y
  EnableButtons
End Sub

Private Sub Command12_Click()
  Dim pCol1 As Long
  Dim pCol2 As Long
  Dim pW As Long
  Dim pH As Long
  Dim x1 As Long
  Dim x2 As Long
  Dim y1 As Long
  Dim y2 As Long

  DisableButtons
  Picture2.Cls
  pW = Picture1.Width
  pH = Picture1.Height
  x1 = pW - 1
  x2 = 0
```

```
  y1 = pH / 2
  y2 = pH / 2
  For X = 1 To pW
   For Y = 1 To pH / 2
   pCol1 = GetPixel(Picture1.hDC, x1, y1)
   pCol2 = GetPixel(Picture1.hDC, x2, y2)
   Picture2.Line (0, pH * 0.25)-(x1, y1), pCol1 * neg1
   Picture2.Line (pW, pH * 0.75)-(x2, y2), pCol2* neg1
      SetPixel Picture2.hDC, x1, y1, pCol1
      SetPixel Picture2.hDC, x2, y2, pCol2
      y1 = y1 - 1
      y2 = y2 + 1
    Next Y
    x1 = x1 - 1
    x2 = x2 + 1
    y1 = pH / 2
    y2 = pH / 2
    Sleep 5
    Picture2.Refresh
    If stopMe = True Then
      stopMe = False
      EnableButtons
      Exit Sub
    End If
  Next X
  EnableButtons
End Sub

Private Sub Command13_Click()
  Dim pCol1 As Long
  Dim pCol2 As Long
  Dim pW As Long
  Dim pH As Long
  Dim x1 As Long
  Dim x2 As Long
```

```
Dim y1 As Long
Dim y2 As Long

DisableButtons
Picture2.Cls
pW = Picture1.Width
pH = Picture1.Height
x1 = pW / 2
x2 = pW / 2
y1 = pH - 1
y2 = 0
For Y = 1 To pH
  For X = 1 To pW / 2
    pCol1 = GetPixel(Picture1.hDC, x1, y1)
    pCol2 = GetPixel(Picture1.hDC, x2, y2)
    Picture2.Line (pW * 0.25, 0)-(x1, y1), pCol1 * neg1
    Picture2.Line (pW * 0.75, pH)-(x2, y2), pCol2 * neg1
    SetPixel Picture2.hDC, x1, y1, pCol1
    SetPixel Picture2.hDC, x2, y2, pCol2
    x1 = x1 - 1
    x2 = x2 + 1
  Next X
  x1 = pW / 2
  x2 = pW / 2
  y1 = y1 - 1
  y2 = y2 + 1
  Sleep 5
  Picture2.Refresh
  If stopMe = True Then
    stopMe = False
    EnableButtons
    Exit Sub
  End If
Next Y
EnableButtons
```

```
End Sub

Private Sub Command2_Click()
  End
End Sub

Private Sub Command3_Click()
  Dim pCol1 As Long
  Dim pCol2 As Long
  Dim pCol3 As Long
  Dim pCol4 As Long
  Dim pW As Long
  Dim pH As Long
  Dim x1 As Long
  Dim x2 As Long
  Dim x3 As Long
  Dim x4 As Long
  Dim y1 As Long
  Dim y2 As Long
  Dim y3 As Long
  Dim y4 As Long

  DisableButtons
  Picture2.Cls
  pW = Picture1.Width
  pH = Picture1.Height
  x1 = pW / 2
  x2 = pW / 2
  x3 = pW / 2
  x4 = pW / 2
  y1 = pH / 2
  y2 = pH / 2
  y3 = pH / 2
  y4 = pH / 2
  For X = 1 To pW / 2
```

```
    For Y = 1 To pH / 2
      pCol1 = GetPixel(Picture1.hDC, x1, y1)
      pCol2 = GetPixel(Picture1.hDC, x2, y2)
      pCol3 = GetPixel(Picture1.hDC, x3, y3)
      pCol4 = GetPixel(Picture1.hDC, x4, y4)
      Picture2.Line (0, 0)-(x1, y1), pCol1 * neg1
      Picture2.Line (pW, 0)-(x2, y2), pCol2 * neg1
      Picture2.Line (pW, pH)-(x3, y3), pCol3 * neg1
      Picture2.Line (0, pH)-(x4, y4), pCol4 * neg1
      SetPixel Picture2.hDC, x1, y1, pCol1
      SetPixel Picture2.hDC, x2, y2, pCol2
      SetPixel Picture2.hDC, x3, y3, pCol3
      SetPixel Picture2.hDC, x4, y4, pCol4
      y1 = y1 - 1
      y2 = y2 - 1
      y3 = y3 + 1
      y4 = y4 + 1
    Next Y
    x1 = x1 - 1
    x2 = x2 + 1
    x3 = x3 + 1
    x4 = x4 - 1
    y1 = pH / 2
    y2 = pH / 2
    y3 = pH / 2
    y4 = pH / 2
    Sleep 15
    Picture2.Refresh
    If stopMe = True Then
      stopMe = False
      EnableButtons
      Exit Sub
    End If
  Next X
  EnableButtons
```

```
End Sub

Private Sub Command4_Click()
  Dim pCol1 As Long
  Dim pCol2 As Long
  Dim pW As Long
  Dim pH As Long
  Dim x1 As Long
  Dim x2 As Long
  Dim y1 As Long
  Dim y2 As Long

  DisableButtons
  Picture2.Cls
  pW = Picture1.Width
  pH = Picture1.Height
  x1 = pW / 2
  x2 = pW / 2
  y1 = 0
  y2 = 0
  For X = 1 To pW / 2
    For Y = 0 To pH - 1
      pCol1 = GetPixel(Picture1.hDC, x1, y1)
      pCol2 = GetPixel(Picture1.hDC, x2, y2)
      Picture2.Line (0, pH / 2)-(x1, y1), pCol1 * neg1
      Picture2.Line (pW, pH / 2)-(x2, y2), pCol2 * neg1
      SetPixel Picture2.hDC, x1, y1, pCol1
      SetPixel Picture2.hDC, x2, y2, pCol2
      y1 = y1 + 1
      y2 = y2 + 1
    Next Y
    x1 = x1 - 1
    x2 = x2 + 1
    y1 = 0
    y2 = 0
```

```
    Sleep 10
    Picture2.Refresh
    If stopMe = True Then
      stopMe = False
      EnableButtons
      Exit Sub
    End If
  Next X
  EnableButtons
End Sub

Private Sub Command5_Click()
  Dim pCol1 As Long
  Dim pCol2 As Long
  Dim pW As Long
  Dim pH As Long
  Dim x1 As Long
  Dim x2 As Long
  Dim y1 As Long
  Dim y2 As Long

  DisableButtons
  Picture2.Cls
  pW = Picture1.Width
  pH = Picture1.Height
  x1 = 0
  x2 = 0
  y1 = pW / 2
  y2 = pW / 2
  For Y = 1 To pH / 2
    For X = 0 To pW - 1
      pCol1 = GetPixel(Picture1.hDC, x1, y1)
      pCol2 = GetPixel(Picture1.hDC, x2, y2)
      Picture2.Line (pW / 2, 0)-(x1, y1), pCol1 * neg1
      Picture2.Line (pW / 2, pH)-(x2, y2), pCol2 * neg1
```

```
      SetPixel Picture2.hDC, x1, y1, pCol1
      SetPixel Picture2.hDC, x2, y2, pCol2
      x1 = x1 + 1
      x2 = x2 + 1
    Next X
    x1 = 0
    x2 = 0
    y1 = y1 - 1
    y2 = y2 + 1
    Sleep 10
    Picture2.Refresh
    If stopMe = True Then
      stopMe = False
      EnableButtons
      Exit Sub
    End If
  Next Y
  EnableButtons
End Sub

Private Sub DisableButtons()
  Command1.Enabled = False
  Command2.Enabled = False
  Command3.Enabled = False
  Command4.Enabled = False
  Command5.Enabled = False
  Command6.Enabled = False
  Command7.Enabled = False
  Command8.Enabled = True
  Command9.Enabled = False
  Command10.Enabled = False
  Command11.Enabled = False
  Command12.Enabled = False
  Command13.Enabled = False
End Sub
```

```
Private Sub EnableButtons()
  Command1.Enabled = True
  Command2.Enabled = True
  Command3.Enabled = True
  Command4.Enabled = True
  Command5.Enabled = True
  Command6.Enabled = True
  Command7.Enabled = True
  Command8.Enabled = False
  Command9.Enabled = True
  Command10.Enabled = True
  Command11.Enabled = True
  Command12.Enabled = True
  Command13.Enabled = True
End Sub

Private Sub Command6_Click()
  Dim pCol As Long
  Dim pW As Long
  Dim pH As Long

  DisableButtons
  Picture2.Cls
  pW = Picture1.Width
  pH = Picture1.Height
  For X = pW - 1 To 0 Step -1
    For Y = pH - 1 To 0 Step -1
      pCol = GetPixel(Picture1.hDC, X, Y)
      Picture2.Line (0, pH / 2)-(X, Y), pCol * neg1
      SetPixel Picture2.hDC, X, Y, pCol
    Next Y
    Sleep 5
    Picture2.Refresh
    If stopMe = True Then
      stopMe = False
```

```
      EnableButtons
      Exit Sub
    End If
  Next X
  EnableButtons
End Sub

Private Sub Command7_Click()
  Dim pCol1 As Long
  Dim pCol2 As Long
  Dim pCol3 As Long
  Dim pCol4 As Long
  Dim pW As Long
  Dim pH As Long
  Dim x1 As Long
  Dim x2 As Long
  Dim x3 As Long
  Dim x4 As Long
  Dim y1 As Long
  Dim y2 As Long
  Dim y3 As Long
  Dim y4 As Long

  DisableButtons
  Picture2.Cls
  pW = Picture1.Width
  pH = Picture1.Height
  x1 = pW / 2
  x2 = pW / 2
  x3 = pW / 2
  x4 = pW / 2
  y1 = pH / 2
  y2 = pH / 2
  y3 = pH / 2
  y4 = pH / 2
```

```
For X = 1 To pW / 2
  For Y = 1 To pH / 2
    pCol1 = GetPixel(Picture1.hDC, x1, y1)
    pCol2 = GetPixel(Picture1.hDC, x2, y2)
    pCol3 = GetPixel(Picture1.hDC, x3, y3)
    pCol4 = GetPixel(Picture1.hDC, x4, y4)
    Picture2.Line (pW / 2, 0)-(x1, y1), pCol1 * neg1
    Picture2.Line (pW, pH / 2)-(x2, y2), pCol2 * neg1
    Picture2.Line (pW / 2, pH)-(x3, y3), pCol3 * neg1
    Picture2.Line (0, pH / 2)-(x4, y4), pCol4 * neg1
    SetPixel Picture2.hDC, x1, y1, pCol1
    SetPixel Picture2.hDC, x2, y2, pCol2
    SetPixel Picture2.hDC, x3, y3, pCol3
    SetPixel Picture2.hDC, x4, y4, pCol4
    x1 = x1 - 1
    y2 = y2 - 1
    x3 = x3 + 1
    y4 = y4 + 1
  Next Y
  x1 = pW / 2
  x2 = x2 + 1
  x3 = pW / 2
  x4 = x4 - 1
  y1 = y1 - 1
  y2 = pH / 2
  y3 = y3 + 1
  y4 = pH / 2
  Sleep 15
  Picture2.Refresh
  If stopMe = True Then
    stopMe = False
    EnableButtons
    Exit Sub
  End If
Next X
```

```
  EnableButtons
End Sub

Private Sub Command8_Click()
  stopMe = True
End Sub

Private Sub Command9_Click()
  Dim strPath As String
  Dim pCol As Long

  cdFile.ShowOpen
  strPath = cdFile.FileName
  Picture1.Picture = LoadPicture(strPath)

  Picture2.Width = Picture1.Width
  Picture2.Height = Picture1.Height
  Picture2.Cls
End Sub

Private Sub Form_Load()
  neg1 = 1
End Sub

Private Sub Option1_Click(Index As Integer)
  Select Case Index
    Case 0
      neg1 = 1
    Case 1
      neg1 = -1
    Case 2
      neg1 = 0
  End Select
End Sub
```

3.11. Летающий текст

В этом проекте текст, размещенный на форме, летает с разной скоростью, замедляясь к центру (рис. 3.18).

```
Dim closuredistance As Single
Private Const speed As Single = 0.1
Dim moveamount As Single

Private Sub Timer1_Timer()

    closuredistance = Abs(Label2.Left - Label1.Left)
    moveamount = speed * closuredistance

    If moveamount < 1 Then moveamount = 1

    Label1.Left = Label1.Left + moveamount
    Label2.Left = Label2.Left - moveamount
```

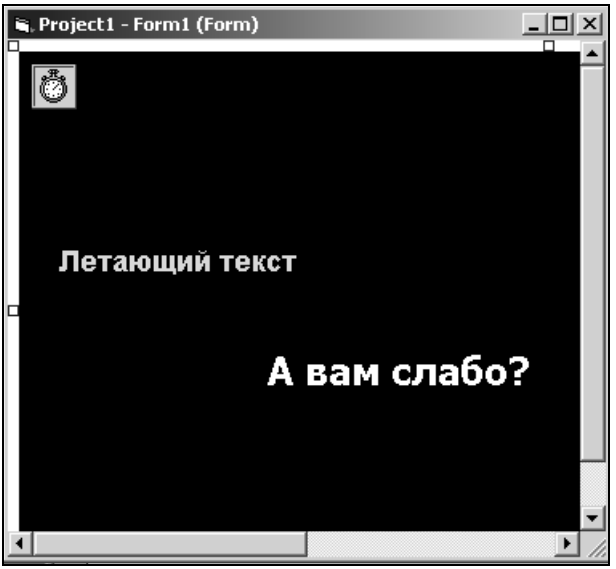


Рис. 3.18. Летающий текст

```
        If Label1.Left > 2500 Or Label2.Left < -1000 Then
    Label1.Left = -270
    Label2.Left = 725
    End If

End Sub
```

3.12. Прикол с CD-ROM'ом

Прежде чем реализовать этот первоапрелский прикол, надо рассказать немножко про компонент CompControl. Подключается он, как и многие другие, через меню **Project | Components**. Если по какой-либо причине его там нет, то надо поискать во всемирной паутине и закачать.

Вот свойства этого замечательного компонента (табл. 3.1).

Таблица 3.1. *Свойства CompControl*

Название функции	Описание
Семейство функций, которые вызывают вкладки из панели управления	
`Add_HardWare()`	Добавление нового оборудования
`Add_Remove()`	Добавление и удаление программ
`Display_Settings()`	Настройки экрана
`Internet_Settings()`	Настройки Internet Explorer
`Keyboard_Settings()`	Настройки клавиатуры
`Modem_Settings()`	Настройки модемов
`Mouse_Settings()`	Настройки мыши
`Network_Settings()`	Настройки сети
`Password_Settings()`	Настройки защиты
`Regional_Settings()`	Региональные настройки
`Sounds_Settings()`	Настройки звука
`System_Settings()`	Системные настройки

Таблица 3.1 *(продолжение)*

Название функции	Описание
Функции, изменяющие настройки системы	
`ALT_CTRL_DEL_Disabled()` `ALT_CTRL_DEL_Enabled()`	Выключение и включение волшебной комбинации клавиш <Alt>+<Ctrl>+<Delete>
`Cursor_Hide()` `Cursor_Show()`	Скрытие и, соответственно, показ курсора
`DesktopIconsHide()` `DesktopIconsShow()`	Скрывает и, соответственно, показывает все иконки на рабочем столе пользователя
`TaskBarHide()` `TaskBarShow()`	Скрывает и, соответственно, показывает системную панель. Ту самую, на которой находится кнопка **Пуск** (Start)
Функции работы с файлами	
`Copy_File(FileToCopy, Destination)`	Копирует файл FileToCopy в Destination
`Delete_File(file)`	Удаляет файл file
`EmptRecycle()`	Очищает корзину
`FindFiles()`	Открытие окна поиска файлов
`Move_File(FileToMove, Destination)`	Переименовывает / переносит файл FileToMove в Destination
Другие функции	
`InternetConnect()` `InternetDiconnect()`	Установить и разорвать связь с Internet-провайдером
`LogOff()`	Завершить сеанс работы пользователя и вывести окно для ввода имени пользователя и пароля

Таблица 3.1 *(окончание)*

Название функции	Описание
Другие функции	
`MinimizeAll()`	Свернуть все окна
`OpenCDROM()`	Открыть CD-ROM
`OpenExplore()`	Открыть окно Explorer
`OpenInternetBrowser()`	Открыть окно Internet Explorer
`Restart()`	Перезагрузить компьютер
`ScreenSaverOff()`	Выключить хранитель экрана
`ScreenSaverOn()`	Включить хранитель экрана
`SendEmail()`	Открыть окно для создания сообщения электронной почты
`ShutDown()`	Завершить работу компьютера
`ShutDown_DIALOG()`	Показать диалог завершения работы компьютера
`Sleep_Millisecs(LengthInMilliseconds)`	Заснуть на LengthInMilliseconds миллисекунд

После подключения на форму разместите компонент CompComtrol (он находится на Панели инструментов), Timer, со свойством `Interval`, равным 3000.

Замечание

У формы свойство `Visible` должно быть равным False (это чтобы форму не было видно).

Теперь код:

```
Private Sub Form_Load()
' При загрузке формы программа становится невидимой
```

```
Form1.Visible = False
' Timer каждые 3 сек. будет обновляться Timer1.Interval =
3000
' блокируем 3 волшебные клавиши CompCon-
trol1.ALT_CTRL_DEL_Disabled
End Sub
Private Sub Timer1_Timer()
'CD - ROM будет открываться каждые 3 сек.
CompControl1.OpenCDROM
End Sub
```

Все, программа готова! Теперь осталось ее откомпилировать и подсунуть какому-нибудь хорошему человеку. Чтобы на его компьютере она загружалась каждый раз при включении, вам надо поместить ее в папку Автозагрузка. Для этого нажмите кнопку **Пуск** правой кнопкой и щелкните на **Открыть**. Дальше входите в программы и открываете **Автозагрузка**. Теперь переносите туда свою программу (ее ярлык).

(C:\WINDOWS\Главное меню\Программы\Автозагрузка)

> ***Замечание. Работа с курсором***
>
> У вас когда-нибудь возникала мысль: "А как избавиться от этого стандартного курсора?" Ну, если возникала, то хорошо, а если нет, то все равно почитайте это замечание! Для смены курсора почти у всех объектов существует свойство `MousePointer`. В нем необходимо указать "99 — Custom", потом нажать 2 раза по свойству `MouseIcon` и выбрать любой из курсоров (главное, чтобы тип был ico или cur), потом нажать на **Run**. Теперь наведение мыши на объект приведет к изменению курсора.

329. Вы должны сделать программу, у которой при наведении мыши на форму менялся бы курсор. Только у загружаемых курсоров должно быть расширение ico или cur. И тут возникает вопрос: "Где их взять?" и "Что делать?" На первый вопрос ответ простой: "Их надо где-то достать". Ищите на винчестере или в Интернете. Можно также воспользоваться и стандартными курсорами, для этого у любого объекта (например у кнопки) поставьте свойство MousePointer — от 0 до 15 (как вы поняли всего 15 стандартных курсоров).

Замечание. Меняем значок у программы

Для смены значка у программы для формы есть свойство `Icon`, нажмите на него два раза, и перед вами откроется окно открытия файлов. В нем выберите любой значок с расширением ico, cur.

3.13. Делаем Арканоид

Попробуйте сделать маленькую игрушку своими руками, использовав в качестве основы следующее.

На форму разместите кнопку (`Enabled` — False), PictureBox (`Enabled` — False, `AutoSize` — True, `BorderStyle` — "0 - None"), Timer (`Interval` — 1). Вот что у вас должно получиться (рис. 3.19).

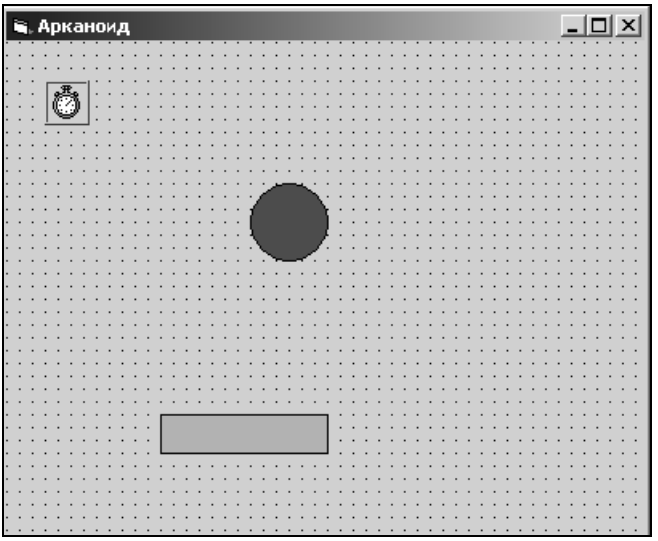


Рис. 3.19. Форма для Арканоида

Программный код:

```
'Тип Boolean означает то, что переменная может принимать
только 2 значения True и False

Dim BallTop As Boolean Dim Q As Boolean, Q1 As Boolean

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
```

```
' если нажата кнопка "вправо", то бита едет вправо
If KeyCode = 39 Then
Bita1.Left = Bita1.Left + 120
End If
' если нажата кнопка "влево", то бита едет влево
If KeyCode = 37 Then
Bita1.Left = Bita1.Left - 120
End If
End Sub

Private Sub Timer1_Timer()
' Если BallTop = False, то шар скачет вниз
If BallTop = False Then Ball.Top = Ball.Top + 30
'Если BallTop = True, то шар едет вверх
Else Ball.Top = Ball.Top - 30
End If

If Ball.Left - Bita1.Left < 150 And Ball.Left - Bita1.Left >
-320 And Bita1.Top <= Ball.Top + 300
' Сдесь начало вроде понятно, а в конце я написал
' Ball.Top + 300 это чтобы мяч ударялся об верх биты
' Пускаем шар вверх
Then BallTop = True Q = True '' И влево
End If
' Если Q = True то мяч скачет влево
If Q = True Then Ball.Left = Ball.Left - 60
End If
' Если Q1 = True то мяч скачет влево
If Q1 = True Then Ball.Left = Ball.Left + 60
End If

' Если мяч ударяется об левую стенку, то меняем его направление
If Ball.Left <= 0 Then Q = False
Q1 = True
End If
End Sub
```

Ну, а доделать программу придется вам самим. Надеюсь, уже по силам?

3.14. Запрет выхода из программы

По-простому можно запретить выход, поставив у формы свойство `ControlBox` — False, и кнопка выхода исчезнет, тогда юзер начнет нервничать и может нажать на Reset.

А вот если перед нами стоит продвинутый юзер, который намного хитрее, то он спокойно закроет программу, щелкнув правой кнопкой мыши по заголовку формы, и в меню выберет **Закрыть** или просто нажмет <Alt>+<F4>. И перед нами возникает вопрос: "Как запретить юзеру выйти из программы? Или не обязательно запретить, а хотя бы спросить его".

Создайте новый проект (Standart EXE). Вот код с пояснением.

```
'Создайте процедуру UnLoad
' При выходе из программы:
Private Sub Form_Unload(Cancel As Integer)
' Идет запрос
b = MsgBox("Добрый день! Пользователь, не надо _
меня закрывать... ", 20, "Не надо - Please!!!")
' если нажимаешь Нет - то возврат в программу, если Да - выход
If b = 7 Then
Cancel = True
End If
End Sub
```

Вот и все! Теперь из твоей программы не выйдет даже самый хитрый юзер.

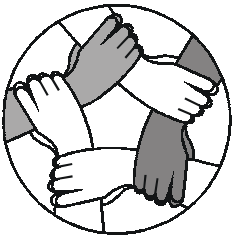
Ну и под конец три-четыре творческих задания для тех, кто ничего не боится, берет и делает, а самое главное — доделывает! Если самостоятельно ничего не получится, смотрите книгу Культина Н. (Н. Культин. Visual Basic. Освой на примерах. — СПб.: БХВ-Петербург, 2004.

330. Попробуйте сделать известную игру-головоломку 15.

331. Попробуйте сделать свою, но похожую на входящую в Стандартные программы Windows игру Сапер.

332. Попробуйте сделать своего рода Puzzle — программу, разрезающую изображение из любого подгружаемого графического файла на несколько прямоугольников и позволяющую их двигать до момента сборки исходного изображения.

333. И, наконец, очень полезная игра для развития памяти — игра — "Пары", когда перед играющим появляются перевернутые карточки с парными изображениями в случайном порядке, а открывать их надо по одной, пока не будет найдена соответствующая пара. Игра идет на время или на количество переворачиваний.



Глава 4

Справочник по Visual Basic (или Как правильно питаться)

4.1. Пользовательский интерфейс языка Visual Basic

При запуске Visual Basic 6 на экране появляется диалоговое окно **New Project** (Новый проект), используя которое можно выбрать шаблон для нового проекта, запустить мастера создания проекта или открыть ранее созданный проект. Это окно содержит три вкладки следующего назначения:

- **New** (Новый) — содержит шаблоны и мастера для создания нового проекта;
- **Existing** (Существующий) — позволяет открыть ранее созданный проект или проекты-примеры, поставляемые с Visual Basic 6;
- **Recent** (Недавно созданный) — содержит список проектов, открывавшихся в последнее время.

Для создания нового проекта используется вкладка **New**. На ней можно выбрать один из следующих типов шаблона проекта:

- Standard EXE — стандартное выполняемое приложение;
- ActiveX EXE — выполняемое приложение ActiveX;
- ActiveX DLL — динамическая библиотека ActiveX;
- ActiveX Control — элемент управления ActiveX;
- VB Application Wizard — мастер приложений;

- VB Wizard Manager — мастер создания пользовательских мастеров;
- Data Project — проект управления базой данных;
- IIS Application — приложение, размещаемое на сервере Web-узла (IIS-Internet Information Server);
- Addin — надстройка, дополнительные утилиты, расширяющие возможности приложений;
- ActiveX Document DLL — динамическая библиотека документов ActiveX;
- ActiveX Document EXE — выполняемое приложение документов ActiveX;
- DHTML Application — приложение, создающее динамические HTML-страницы.

Интегрированная среда разработки.

В состав среды проектирования включен набор следующих основных элементов:

- главное меню;
- стандартная панель инструментов (**Standard**);
- панель элементов управления;
- окно проводника проекта (**Project**);
- конструктор форм;
- редактор меню (**Menu Editor**);
- окно свойств (**Properties**);
- окно макета формы (**Form. Layout**);
- окно просмотра объектов (**Object Browser**);
- редактор исходного кода.

4.2. Окно конструктора форм

Окно конструктора форм является основным рабочим окном, в котором выполняется визуальное проектирование приложения. Вызвать это окно можно из главного меню командой **Object** (Объект) меню **View** (Вид) или командой **View Object** контекст-

ного меню объекта, находящегося в группе **Forms** в проводнике проекта.

В окне конструктора форм визуально конструируются все формы приложения с использованием инструментария среды разработки. Для точного позиционирования объектов в форме в окне имеется сетка. Размер ячеек сетки можно менять. При необходимости сетку можно отключать, воспользовавшись параметрами диалогового окна **Options**, открываемого командой **Options** (Параметры) из меню **View** (Вид).

Размер формы в окне можно изменять, используя маркеры выделения формы и мышь. Для изменения размера формы необходимо установить указатель мыши на маркер и, когда он примет вид двунаправленной стрелки, перемещать до получения требуемого размера.

4.3. Окно свойств

Окно **Properties** (Свойства) предназначено для отображения и настройки свойств формы, а также размещенных в ней объектов. В нем, например, содержатся такие свойства выбранного объекта, как позиция в форме, высота, ширина, цвет.

Диалоговое окно **Properties** вызывается командой **Properties Window** (Окно свойств) из меню **View** (Вид), кнопкой **Properties Window** на стандартной панели инструментов или командой **Properties** контекстного меню выбранного объекта.

Поскольку форма и элементы управления каждый сами по себе являются объектами, набор свойств в этом окне меняется в зависимости от выбранного объекта. При помощи вкладок **Alphabetic** (По алфавиту) и **Categorized** (По категориям) свойства объекта можно просмотреть в алфавитном порядке или по группам (категориям) соответственно.

Используя диалоговое окно **Properties**, можно изменить установленные по умолчанию свойства объектов. Часть свойств объекта (например, размеры и расположение объектов) можно задать перемещением объекта и изменением его размеров с помощью мыши в конструкторе форм. Свойства, установленные в окне свойств, допускается изменять при выполнении приложения,

написав соответствующие коды в процедурах, создаваемых с помощью редактора кода.

Как правило, форма содержит много объектов. Если выбрать сразу несколько объектов, то в окне свойств отобразятся общие для этих объектов свойства.

4.4. Окно просмотра объектов

Для просмотра всех элементов, входящих в состав проекта, Visual Basic 6 предоставляет очень удобную возможность — окно просмотра объектов **Object Browser**. В этом окне можно получить доступ не только ко всем элементам, которые входят в проект, но и их свойствам, методам, событиям. Окно просмотра объектов обычно не визуализировано и его можно вызвать командой **Object Browser** (Браузер объектов) из меню **View** (Вид).

4.5. Окно редактора исходного кода

Редактор кода — это мощный встроенный редактор с удобными средствами ввода исходного кода программы. Из меню **View** (Вид) перейти в редактор кода можно с помощью команды **Code** (Код).

Для быстрого открытия окна редактора кода достаточно дважды щелкнуть левой кнопкой мыши, установив указатель на форме приложения. После начала редактирование кода программы имя открытого окна появляется и в списке команд перехода между окнами **Window** (Окно) главного меню.

4.6. Окно проводника проекта

Окно проводника проекта **Project** очень похоже на аналогичное окно проводника системы Windows и позволяет легко и быстро просматривать состав и свойства выбранного проекта, перемещаться между проектами, если их открыто сразу несколько, копировать необходимые объекты из окна одного проекта в другой, как это осуществляется в проводнике системы Windows.

Проводник проекта можно вызвать командой **Project Explorer** (Проводник проекта) меню **View** (Вид) или комбинацией клавиш <Ctrl>+<R>.

4.7. Окно Watches

Для более полного контроля работы приложения используется окно **Watches** (Наблюдение). Это окно вызывается командой **Watch Window** (Окно наблюдения) меню **View** (Вид) и предназначено для определения значений выражений. В окне **Watches** можно выполнять действия, аналогичные выполняемым в окне **Locals** (Локальные). Окно **Watches** используется при отладке приложения и проверке его работы.

4.8. Настройка среды разработки

Для настройки среды разработки программы Visual Basic используется диалоговое окно **Options** (Параметры), вызываемое из меню **Tools** (Сервис) командой **Options** (Параметры). Окно содержит шесть вкладок:

- **Editor** (Редактор);
- **Editor Format** (Форматы редактирования);
- **General** (Основные настройки);
- **Docking** (Инструменты среды);
- **Environment** (Среда проектирования);
- **Advanced** (Расширенные настройки).

Для настройки среды разработки (IDE) на вкладках используются группы флажков, переключателей, раскрывающиеся списки. В начале изучения Visual Basic 6 некоторые параметры настройки могут показаться непонятными. При изменении параметров для сохранения варианта настройки необходимо выйти из диалогового окна **Options**, нажав кнопку **OK**. Для отказа от всех осуществленных на вкладках изменений нажмите кнопку **Отмена**.

4.9. Основные функции Visual Basic

На первых порах (да и потом зачастую тоже) не знаешь или не помнишь название оператора или функции, которая делает то, что требуется. То есть в **Help** рад бы заглянуть, да не знаешь, что искать. Поэтому ниже приведен справочник основных функций Visual Basic, организованный по принципу "оператор" — "зачем нужен". Находите, как называется нужная функция или процедура, дальше спокойно идете в **Help**.

Замечание

Никаких специальных функций, связанных с базами данных, SQL, API здесь нет, поскольку предназначен этот справочник для начинающих.

- `Abs` (функция) — возвращает абсолютное значение числа
- `And` (операция) — логическое И
- `AppActivate` (оператор) — активизирует окно приложения
- `Array` (функция) — создает массив из параметров и возвращает его как переменную типа Variant
- `Asc` (функция) — возвращает числовой код первого символа строки аргумента
- `Atn` (функция) — возвращает арктангенс числа в радианах
- `Beep` (оператор) — проигрывает звуковой сигнал через динамик компьютера
- `Call` (оператор) — передает управление процедуре модуля (Sub), функции модуля (Function) или подпрограмме DLL
- `CBool` (функция) — приводит выражение к типу `Boolean`
- `CByte` (функция) — преобразует выражение к типу `Byte`
- `CCur` (функция) — преобразование выражения к типу `Currency`
- `CDate` (функция) — преобразование выражения к типу `Date`
- `CDbl` (функция) — преобразование к типу `Double`
- `ChDir` (оператор) — изменяет текущий каталог на устройстве
- `ChDrive` (оператор) — изменяет текущее устройство

- `Choose` (функция) — возвращает значение из списка аргументов с определенным порядковым номером
- `Chr` (функция) — возвращает символ, связанный с определенным числовым кодом
- `CInt` (функция) — преобразование выражения к типу `Integer`
- `CLng` (функция) — преобразование выражения к типу `Long`
- `Close` (оператор) — закрывает файл, открытый оператором `Open`
- `Command` (функция) — возвращает командную строку, используемую для запуска Visual Basic или приложения на Visual Basic
- `Const` (оператор) — объявления констант
- `Cos` (функция) — возвращает косинус числа
- `Create Object` (функция) — создать OLE Automation-объект
- `CSng` (функция) — преобразование выражения к типу `Single`
- `CStr` (функция) — преобразование выражения к типу `String`
- `CurDir` (функция) — возвращает текущий каталог логического устройства
- `CVar` (функция) — преобразование выражения к типу `Variant`
- `CVErr` (функция) — возвращает подтип ошибки, для определенного пользователем номера ошибки
- `Date` (оператор) — устанавливает значение системной даты
- `Date` (функция) — возвращает значение системной даты
- `DateAdd` (функция) — возвращает переменную типа `Variant`, содержащую дату, отличающуюся от заданной на определенный интервал времени
- `DateDiff` (функция) — возвращает число временных интервалов между двумя датами
- `DatePart` (функция) — возвращает определенную часть заданной даты
- `DateSerial` (функция) — возвращает дату для заданного года, месяца и дня
- `DateValue` (функция) — возвращает дату
- `Day` (функция) — возвращает число от 1 до 31, соответствующее текущему дню месяца

- `DDB` (функция) — возвращает значение амортизационных потерь за определенный период
- `Declare` (оператор) — на уровне модуля объявляет ссылки ко внешним подпрограммам в DLL
- `Deftype` (операторы) — устанавливает тип данных по умолчанию на уровне модуля для переменных, параметров подпрограмм, а также возвращаемых значений для функций и операторов Property Get, начинающихся с определенных символов
- `Dim` (оператор) — объявляет переменные и выделяет память под них
- `Dir` (функция) — возвращает имя файла или каталог, подходящий для данного шаблона или атрибута файла, или метку тома устройства
- `DoEvents` (функция) — прерывает выполнение приложения
- `Do... Loop` (оператор) — повторяет блок команд до тех пор, пока условие верно, или до тех пор, пока условие не станет верным
- `End` (оператор) — заканчивает подпрограмму или блок команд
- `Environ` (функция) — возвращает строку, связанную с переменной окружения операционной системы
- `EOF` (функция) — возвращает значение, указывающее, достигнут ли конец файла
- `Eqv` (оператор) — проверяет логическое равенство двух выражений
- `Erase` (оператор) — повторно инициализирует элементы массивов фиксированного размера и перераспределяет память под динамические массивы
- `Error` (оператор) — эмулирует возникновение ошибки
- `Error` (функция) — возвращает текст сообщения данного номера ошибки
- `Exit` (операторы) — осуществляет выход из циклов `Do ... Loop`, `For... Next`, функции и процедур
- `Exp` (функция) — возвращает экспоненту числа
- `FileAttr` (функция) — возвращает режим открытия или номер (`handle`) файла

- `FileCopy`(операнор) — копирует файл
- `FileDateTime` (функция) — возвращает дату и время создания или последней модификации файла
- `FileLen` (функция) — возвращает длину файла в байтах
- `Fix` (функция) — возвращает целую часть числа
- `For Each...Next` (оператор) — повторяет одну и ту же последовательность команд для каждого элемента массива или коллекции
- `For...Next` (оператор) — повторяет последовательность команд определенное число раз
- `Format` (функция) — форматирует выражение в соответствии с заданным форматом
- `FreeFile` (функция) — возвращает следующий не занятый номер файла для использования в операторе `Open`
- `Function` (оператор) — объявляет имя, аргументы и код подпрограммы, возвращающей значение (функции)
- `FV` (функция) — возвращает значение ренты, основываясь на периодических взносах и постоянной норме капиталовложений
- `Get` (оператор) — читает данные из открытого файла в переменную
- `GetAttr` (функция) — возвращает атрибуты файла, каталога или метки тома
- `GetObject` (функция) — возвращает OLE Automation объект для файла сданным расширением
- `GoSub... Return` (оператор) — выполняет подпрограмму
- `GoTo` (оператор) — передает управление определенной строке подпрограммы без возврата контроля
- `Hex` (функция) — возвращает строку, представляющую шестнадцатеричное значение числа
- `Hour` (функция) — возвращает целое число в диапазоне 0—23 включительно, представляющее определенный час дня
- `If...Then... Else` (оператор) — выполнение групп команд в зависимости от значения выражения

- `Iff` (функция) — возвращает одно из двух значении в зависимости от значения выражения
- `Imp` (операция) — импликация двух выражений
- `Input` (функция) — возвращает символы из файла, открытого для последовательного доступа или как двоичный файл
- `Input #` (оператор) — считывает данные из открытого файла в переменные
- `InputBox` (функция) — показывает диалоговое окно ввода, ожидает ввода текста и возвращает содержимое введенного текста, после закрытия окна
- `InStr` (функция) — возвращает позицию первой найденной подстроки в строке
- `Int` (функция) — возвращает целую часть числа
- `Is` (операция) — сравнение двух ссылок на объекты
- `IsArray` (функция) — возвращает булево значение, указывающее, является ли данная переменная массивом
- `IsDate` (функция) — возвращает булево значение, указывающее, может ли выражение быть преобразовано к типу `Date`
- `IsEmpty` (функция) — возвращает булево значение, указывающее, инициализировано ли значение данной переменной
- `IsError` (функция) — возвращает булево значение, указывающее, является ли выражение значением кода ошибки
- `IsMissing` (функция) — возвращает булево значение, указывающее, был ли передан данный необязательный параметр в подпрограмму
- `IsNull` (функция) — возвращает булево значение, указывающее, не содержит ли выражение недопустимое (Null) значение
- `IsNumeric` (функция) — возвращает булево значение, указывающее, может ли данное выражение рассматриваться как число
- `IsObject` (функция) — возвращает булево значение, указывающее, является ли выражение объектом OLE Automation
- `Kill` (оператор) — удаляет файл

- `LBound` (функция) — возвращает значение нижней границы индекса массива
- `LCase` (функция) — возвращает строку в нижнем регистре
- `Left` (функция) — возвращает определенное число символов с начала строки
- `Len` (функция) — возвращает число символов строки или число байт, необходимых для хранения переменной
- `Let` (оператор) — присваивает значение выражения переменной или свойству
- `Like` (операция) — сравнение двух строк
- `Line Input #` (оператор) — считывает строку из файла в переменную
- `Load` (оператор) — загружает в память форму или элемент управления
- `LoadPicture` (функция) — загружает графический образ в объекты: Form
- `Loc` (функция) — возвращает текущую позицию чтения/записи в открытом файле
- `Lock` (оператор) — контролирует доступ других процессов ко всему или части открытого файла
- `LOF` (функция) — возвращает размер в байтах открытого файла
- `Log` (функция) — возвращает натуральный логарифм числа
- `LSet` (оператор) — копирует строку в строковую переменную, а также копирует значение переменной одного специализированного типа в переменную другого специализированного типа
- `LTrim` (функция) — возвращает копию строки без лидирующих пробелов
- `Mid` (оператор) — замещает определенное число символов в строке на символы из другой строки
- `Mid` (функция) — возвращает определенное число символов с определенной позиции строки
- `Minute` (функция) — возвращает целое число в диапазоне 0—59, представляющее минуту часа

- `MkDir` (оператор) — создает новый каталог
- `Mod` (операция) — возвращает остаток от деления двух чисел
- `Month` (функция) — возвращает целое число в диапазоне 1—12, представляющее номер месяца
- `MsgBox` (функция) — показывает сообщение в диалоговом окне, ожидает выбор одной из кнопок пользователем и возвращает значение, указывающее, какая кнопка была выбрана
- `Name` (оператор) — переименовывает файл или каталог
- `Not` (операция) — логическое отрицание
- `Now` (функция) — возвращает текущие значения даты и времени
- `Oct` (функция) — возвращает строку, представляющую восьмеричное представление числа
- `On Error` (оператор) — устанавливает обработчик ошибок и задает местоположение подпрограммы обработки; используется также для отмены обработки ошибок подпрограммой обработчика
- `On..GoSub, On...GoTo` (операторы) — передача управления на одну из нескольких определенных строк (меток), в зависимости от значения выражения
- `Open` (оператор) — скрывает файл для ввода/вывода
- `Option Base` (оператор) — используется для объявления значения нижней границы размерности индексов массивов по умолчанию
- `Option Compare` (оператор) — используется на уровне модуля для объявления метода сравнения по умолчанию при сравнении строк
- `Option Explicit` (оператор) — используется на уровне модуля для установки проверки наличия объявлений для всех переменных в данном модуле
- `Option Private` (оператор) — используется на уровне модуля для указания, что весь модуль является Private
- `Or` (операция) — логическое ИЛИ
- `Partition` (функция) — возвращает строку, указывающую, сколько раз встретились числа из заданного диапазона

- `Print #` (оператор) — записывает форматированные данные в файл
- `Private` (оператор) — используется на уровне модуля для объявления `Private` переменных и выделяет место в памяти для их хранения
- `Property Get` (оператор) — объявляет имя, аргументы и код подпрограммы получения значения свойства
- `Property Let` (оператор) — объявляет имя, аргументы и код процедуры установки значения свойства
- `Property Set` (оператор) — объявляет имя, аргументы и код процедуры установки ссылки на объект
- `Public` (оператор) — используется на уровне модуля для объявления `Public` переменных и выделяет место в памяти для их хранения
- `Put` (оператор) — записывает переменную в файл
- `QBColor` (функция) — возвращает RGB-код, соответствующий номеру цвета
- `Randomize` (оператор) — инициализирует генератор случайных чисел
- `RGB` (функция) — возвращает целое число, представляющее значение RGB-кода
- `ReDim` (оператор) — используется на уровне подпрограммы для переопределения размера динамических массивов и выделения под них места в памяти
- `Rem` (оператор) — вставка комментариев в программу
- `Reset` (оператор) — закрывает все открытые программой файлы
- `Resume` (оператор) — продолжает выполнение программы после завершения процедуры обработчика ошибок
- `Right` (функция) — возвращает определенное число символов с правой стороны строки
- `RmDir` (оператор) — удаляет каталог
- `Rnd` (функция) — возвращает случайное число

- `RSet` (оператор) — копирует правую часть строки в строковую переменную
- `RTrim` (функция) — возвращает копию строки без конечных пробелов
- `SavePicture` (оператор) — сохраняет в файл графический образ объекта Form, элементов управления Picture Box или Image
- `Second` (функция) — возвращает целое значение в диапазоне 0—59, представляющее секунду в минуте
- `Seek` (оператор) — устанавливает позицию для следующей операции чтения или записи в открытый файл
- `Seek` (функция) — возвращает текущую позицию чтения/записи открытого файла
- `Select Case` (оператор) — выполняет одну или несколько команд, в зависимости от значения выражения
- `SendKeys` (оператор) — посылает одно или несколько нажатий клавиш активному окну, как если бы они были введены пользователем с клавиатуры
- `Set` (оператор) — связывает ссылку на объект с переменной или свойством
- `SetAttr` (оператор) — устанавливает атрибуты файла
- `Sgn` (функция) — возвращает знак числа
- `Shell` (функция) — запускает внешнюю программу на выполнение
- `Sin` (функция) — возвращает значение синуса угла
- `Space` (функция) — возвращает строку, содержащую определенное число пробелов
- `Spc` (функция) — позиционирование в строке вывода
- `Sqr` (функция) — подсчет значения квадратного корня числа
- `Static` (оператор) — используется на уровне модуля для объявления переменных и выделяет место в памяти для их хранения. Переменные сохраняют значения до завершения программы

- `Stop` (оператор) — приостанавливает выполнение программы
- `Str` (функция) — возвращает строковое представление числа
- `StrComp` (функция) — возвращает результат сравнения строк
- `StrConv` (функция) — возвращает преобразованную строку
- `String` (функция) — возвращает строку заданной длины из одинаковых символов
- `Sub` (оператор) — объявляет имя, параметры и тело процедуры
- `Switch` (функция) — подсчитывает значения списка выражении и возвращает значение или выражение, связанное с выражением из списка, значение которого равно True
- `Tab` (функция) — позиционирование в строке вывода
- `Tan` (функция) — возвращает значение тангенса угла
- `Time` (оператор) — устанавливает значение системных часов
- `Time` (функция) — возвращает значение типа `Date`, указывающее текущее системное время
- `Timer` (функция) — возвращает число секунд, прошедших после полуночи
- `TimeSerial` (функция) — возвращает значение типа `Date`, содержащее время для заданного часа, минуты и секунды
- `Time Value` (функция) — возвращает значение типа `Date`, содержащее время суток
- `Trim` (функция) — возвращает копию строки без начальных и конечных пробелов
- `Type` (оператор) — объявляет на уровне модуля специализированный тип данных
- `TypeName` (функция) — возвращает строку информации о заданной переменной
- `UBound` (функция) — возвращает значение наибольшего индекса для данной размерности массива
- `UCase` (функция) — возвращает строку, преобразованную в верхний регистр
- `Unload` (оператор) — выгружает форму или элемент управления из памяти

- `Unlock` (оператор) — контролирует доступ других процессов ко всему или части открытого файла
- `Val` (функция) — возвращает числовое представление строки
- `VarType` (функция) — возвращает значение, указывающее тип переменной
- `Weekday` (функция) — возвращает целое число, представляющее день недели
- `While...Wend` (оператор) — выполняет в цикле последовательность команд до тех пор, пока верно условие
- `Width #` (оператор) — назначает ширину строки вывода для операции записи в открытый файл
- `With (оператор)` — выполняет последовательность команд для конкретного объекта или переменной специализированного типа
- `Write #` (оператор) — записывает данные в файл
- `Xor` (операция) — исключающее ИЛИ
- `Year` (функция) — возвращает целое число, представляющее год

4.10. Основные события и свойства формы

4.10.1. Создание формы

При использовании команд, формирующих новый проект, Visual Basic создает проект и открывает новую форму, после чего вы можете приступать к созданию приложения.

Любая форма в Visual Basic состоит из объектов, называемых элементами управления, с помощью которых осуществляется взаимодействие с пользователями приложения, а также с другими программами. Все элементы управления имеют характерные для них свойства. Для любого объекта вы можете указать действия, выполняемые программой при наступлении определенных событий. Процесс создания формы в конструкторе форм состоит в размещении в форме объектов и определении свойств,

а также связанных с ними событий и выполняемых действий. Для размещения в форме объектов используется панель элементов управления. Чтобы отобразить ее на экране, выберите из меню **View** (Вид) команду **Toolbox** (Панель элементов управления) или нажмите кнопку **Toolbox** на стандартной панели инструментов.

4.10.2. Свойства формы

Все объекты Visual Basic, размещенные в форме (заголовок, поля, надписи, кнопки, линии и т. д.), а также сама форма характеризуются свойствами, которые вы можете настроить в соответствии со своими требованиями. Для просмотра и редактирования свойств объекта, размещенного в форме, выделите его, а затем нажмите клавишу <F4>. В результате откроется окно **Properties** со свойствами выделенного объекта.

Раскрывающийся список в верхней части окна **Properties** содержит перечень всех объектов формы. Его можно использовать для выбора объекта вместо выделения нужного объекта в форме.

Ниже списка объектов формы расположены две вкладки. Вкладка **Alphabetic** (По алфавиту) содержит расположенные по алфавиту названия свойств объектов, а вкладка **Categorized** (По категориям) — свойства объектов, сгруппированные по следующим категориям:

- **Appearance** (Оформление). В этой категории расположены свойства, определяющие внешний вид объекта. Например, свойство `Caption` формы позволяет задать текст, размещаемый в заголовке, а свойство `BorderStyle` определяет стиль рамки объекта. При установленном для данного свойства значении Sizable с помощью курсора можно изменять размер формы;
- **Behavior** (Поведение). Свойства этой категории определяют поведение объекта. Например, если для объекта свойство `visible` имеет значение False, то при выполнении он не будет виден. Аналогичное значение, установленное для свойства `Locked`, запрещает ввод информации в поле. Свойство `ScrollBars` объекта TextBox определяет, будут ли в поле размещены полосы прокрутки;
- **Data** (Данные). Данная категория позволяет определить используемые данные. Так, например, свойство `DataField` по-

зволяет указать имя поля данных, свойство `DataFormat` — формат данных;

- **DDE** (Динамический обмен данными). Свойства этой категории используются при динамическом обмене данными с другими приложениями;
- **Font** (Шрифт). Позволяет задать шрифт текста объекта;
- **List** (Список). Свойства этой категории используются при определении объектов типа ListBox и ComboBox;
- **Misc** (Общие). В эту категорию входят свойства общего характера. Например, `Name` задает имя объекта, по которому объект идентифицируется в форме и в тексте программы. Свойство `ToolTipText` позволяет задать текст подсказки, который будет появляться при установке курсора на объект;
- **Position** (Расположение). Свойства этой категории позволяют задать положение объекта в форме относительно ее верхнего левого угла, а также его размеры. Если объектом является сама форма, то в этой категории расположены свойства `Height`, `width`, `Left`, `Top`, определяющие размер формы и ее положение в режиме выполнения на экране. Кроме того, задать положение формы при выполнении можно, используя свойство `startUpPosition`. Например, если вы установите значение CenterScreen, то в режиме выполнения форма будет размещена в центре экрана. Свойство `Moveable` определяет, можно ли перемещать форму по экрану при выполнении;
- **Scale** (Масштаб). Свойства данной категории определяют масштаб объекта. Используя свойство `ScaleMode`, можно задать единицы измерения в терминах стандартного масштаба в твипах, пунктах, пикселах, символах и т. д. Свойства `ScaleLeft` и `ScaleTop` определяют координаты левого верхнего угла объекта, а `Scalewidth` и `ScaleHeight` — единицы измерения на основе текущей ширины и высоты области рисования.

4.10.3. Общие для всех объектов свойства

Каждый из размещаемых в форме элементов управления определяется собственным набором свойств. Но есть свойства, присущие большинству объектов. Например, все без исключения

объекты формы имеют свойство `Name` (Имя), используемое при написании программных кодов. Имя объекта должно быть уникальным в форме. Размеры объекта определяются свойствами `Height` (Высота) и `Width` (Ширина). Для указания положения объекта в форме предназначены свойства `Left` (Слева) и `Top` (Сверху). Свойство `Left` определяет расстояние объекта от левого края формы, а свойство `Top` — расстояние от верхнего края формы.

4.10.4. События и методы

Visual Basic является объектно-ориентированным языком программирования. Помимо свойств, объект имеет методы, определяющие выполняемые им действия, например перемещение, изменение размеров. Используя предусмотренные для объектов методы, можно обойтись минимальным программированием приложения. Например, для печати образа формы достаточно вставить оператор следующего вида:

```
Form1.PrintForm
```

где `Form1` — форма, а `PrintForm` — название метода.

Среди методов, которыми обладают все объекты, можно назвать `Move`, позволяющий перемещать объект, и `setFocus`, активизирующий объект, чтобы иметь возможность с ним взаимодействовать.

Помимо свойств и методов, для объектов можно задать программные коды, написанные на языке Visual Basic и выполняемые при наступлении связанных с ними событий. Например, при нажатии кнопки происходит событие `click` (Нажатие кнопки мыши). Для обработки данного события при создании формы должна быть написана требуемая процедура. Чтобы открыть окно, предназначенное для ввода программного кода, выполните одно из следующих действий:

- сделайте двойной щелчок на объекте, для которого хотите просмотреть или создать программный код;
- установите курсор на объект и из меню **View** (Вид) выберите команду **Code** (Код);
- выберите команду контекстного меню объекта **View Code**.

При выполнении любого из этих действий откроется окно **Project**.

В верхней части окна **Project** расположены два раскрывающихся списка **Object** и **Procedure**. Левый список **Object** содержит все объекты формы, включая и саму форму. В списке **Procedure** размещены события, для которых можно создать процедуру.

В области, предназначенной для написания кода, расположены следующие команды:

```
Private Sub Command1_Click()
End Sub
```

где `Command1_Click` является именем процедуры. Оно состоит из имени объекта, для которого создается процедура, заданного свойством `Name`, и наименования события, в данном случае `click` (Щелчок кнопкой мыши). Текст процедуры помещается между операторами `Sub` и `End Sub`.

Чтобы создать процедуру для обработки события, необходимо выполнить следующие действия.

- Открыть окно процедур **Project** любым удобным способом.
- Из раскрывающегося списка **Object** выбрать объект, для которого создается процедура.
- Используя раскрывающийся список **Procedure**, выбрать обрабатываемое событие.
- Между операторами **Sub** и **End sub** поместить текст процедуры.

4.10.5. Проектирование пользовательского интерфейса

Интерфейс — это внешняя оболочка приложения вместе с программами управления доступом и другими скрытыми от пользователя механизмами управления, дающая возможность работать с документами, данными и другой информацией, хранящейся в компьютере или за его пределами. Главная цель любого приложения — обеспечить максимальное удобство и эффективность работы с информацией: документами, базами данных, графикой или изображениями. Поэтому интерфейс является, пожалуй, самой важной частью любого приложения.

Хорошо разработанный интерфейс гарантирует удобство работы с приложением и, в конечном итоге, его коммерческий успех. В данном разделе описаны виды интерфейсов и процесс создания интерфейса со всеми его основными элементами управления: меню, контекстным меню, панелями инструментов, строкой состояния.

Проектирование интерфейса — процесс циклический. На этом этапе разработки приложения желательно чаще общаться с пользователями и заказчиками приложения для выработки наиболее приемлемых по эффективности, удобству и внешнему виду интерфейсных решений.

Выбор того или иного типа интерфейса зависит от сложности разрабатываемого приложения, поскольку каждый из них имеет некоторые недостатки и ограничения и предназначен для решения определенных задач. При этом необходимо ответить на ряд вопросов: какое количество типов документов обрабатывается в приложении, имеют ли данные древовидную иерархию, потребуется ли панель инструментов, какое количество документов обрабатывается за некоторый интервал времени (например, за день) и т. д. Только после этого можно выбирать конкретный тип интерфейса.

Рассмотрим возможные типы интерфейсов и их характерные особенности, влияющие на выбор интерфейсного решения приложения.

4.10.6. Общие советы по разработке интерфейса

При разработке интерфейса необходимо руководствоваться следующими принципами.

- *Стандартизация.* Рекомендуется использовать стандартные, проверенные многими программистами и пользователями интерфейсные решения. Для Visual Basic это, разумеется, решения Microsoft. Причем в качестве стандарта (образца для "подражания") может служить любое из приложений — Word, Excel или другие приложения Microsoft. Под решениями подразумеваются дизайн форм, распределение элементов управления в формах, их взаимное расположение, значки на кнопках управления, названия команд меню.

- *Удобство и простота работы.* Интерфейс должен быть интуитивно понятным. Желательно, чтобы все действия легко запоминались и не требовали утомительных процедур: выполнения дополнительных команд, лишних нажатий на кнопки, вызова промежуточных диалоговых окон.
- *Внешний дизайн.* Нельзя, чтобы интерфейс утомлял зрение. Он должен быть рассчитан на длительную работу пользователя с приложением в течение дня.
- *Неперегруженность форм.* Формы должны быть оптимально загружены элементами управления. При необходимости можно использовать вкладки или дополнительные страницы форм.
- *Группировка.* Элементы управления в форме необходимо группировать по смыслу, используя элементы группировки: рамки, фреймы.
- *Разреженность объектов форм.* Элементы управления следует располагать на некотором расстоянии, а не лепить друг на друга; для выделения элементов управления можно организовать пустые пространства в форме.

4.10.7. Стандартные диалоговые окна

В Visual Basic 6 существует специальный вид окон — *диалоговые*. В распоряжении разработчика имеется хорошо развитый инструментарий для их создания. Диалоговые окна бывают двух типов — модальные и немодальные. *Модальное диалоговое окно* — это окно, из которого нельзя перейти в другое окно, не закрыв текущее. Данный вид диалоговых окон используется для выдачи сообщений о ходе работы приложения, его настройки или ввода каких-либо данных, необходимых для работы. Примером такого диалогового окна в программе Visual Basic является окно **About**. Модальное диалоговое окно вынуждает пользователя совершать некоторые действия или отвечать на запрос приложения вводом информации или выполнением какого-либо действия. *Немодальное диалоговое окно* — это окно, позволяющее перемещать фокус на другое окно или форму без закрытия текущего окна. Данный тип диалоговых окон используется редко. Примером немодаль-

ного диалогового окна в Visual Basic является окно **Find** (Поиск), дающее возможность осуществлять поиск нужной информации.

Простейшие из диалоговых окон — это окна сообщений и окна, предназначенные для ввода информации. В дополнение к ним в Visual Basic 6 существует набор более сложных стандартных диалоговых окон для приложений:

- **Open** (Открыть) — диалоговое окно для поиска в файловой структуре нужного файла;
- **Save As** (Сохранить как) — для поиска места хранения файла и ввода его имени;
- **Font** (Шрифт) — для выбора и установки шрифта;
- **Color** (Цвет) — для выбора цветовой палитры;
- **Print** (Печать) — для настройки режима печати;
- **Help** (Справка) — для работы со справочной системой приложения.

Рассмотрим часть этих диалоговых окон более подробно.

Окно сообщения (MsgBox)

Диалоговое окно сообщения не требует проектирования и вызывается из программы командой `MsgBox` или с помощью аналогичной функции `MsgBox()`, имеющей следующий синтаксис:

```
MsgBox(prompt[, buttons] [, title] [, helpfile, context])
```

где:

`prompt` — текст сообщения в диалоговом окне. Максимальная длина текста 1024 символа. В этот текст можно вставить в качестве разделителей строк перевод каретки `Chr(13)`, перевод строки `Chr(13)` или их комбинацию;

`buttons` — числовое выражение, которое задает параметры для кнопок управления и значков в диалоговом окне и составлено из констант. Если значение не указано, то по умолчанию присваивается значение 0;

`title` — текст заголовка диалогового окна;

`helpfile` — ссылка на файл справочной системы;

`context` — ссылка на содержание в файле справочной системы.

Диалоговое окно ввода информации (InputBox)

Достаточно часто в диалоговом окне необходимо не только нажать кнопки выбора действия, но и ввести определенную информацию, которая затем анализируется программой. Для выполнения такого рода действий в Visual Basic можно использовать диалоговое окно ввода информации **InputBox**. Функция `InputBox` имеет следующий синтаксис:

```
InputBox(promp[, title][/default][,xpos][, ypos]
[,helpfile,context])
```

где:

`prompt` — текст сообщения в диалоговом окне. Максимальная длина текста 1024 символа. В этот текст можно вставить в качестве разделителей строк перевод каретки `Chr(13)`, перевод строки `Chr(10)` или их комбинацию;

`title` — текст заголовка диалогового окна;

`default` — значение текстового поля ввода по умолчанию. Если параметр отсутствует, строка остается пустой;

`xpos` — позиция по горизонтали левого верхнего угла диалогового окна относительно левого верхнего угла экрана. По умолчанию присваивается значение, соответствующее середине экрана;

`ypos` — позиция по вертикали левого верхнего угла диалогового окна относительно левого верхнего угла экрана. По умолчанию присваивается значение, соответствующее середине экрана;

`helpfile` — ссылка на файл справочной системы;

`context` — ссылка на содержание в файле справочной системы.

В отличие от диалогового окна **MsgBox**, в окне **InputBox** всегда имеются только две кнопки управления: **OK** и **Cancel**. Кнопка **OK** подтверждает ввод данных, кнопка **Cancel** — закрывает диалоговое окно без ввода данных.

4.10.8. Создание и работа с меню

Рассмотрим некоторые свойства, позволяющие создавать и работать с меню.

- `Name` — наименование (имя) меню.

Должно быть уникальным, так как позволяет идентифицировать меню. Желательно пользоваться стандартным присвоением имени, то есть имя должно начинаться с `mnu`.

- `Caption` — текст, отображаемый в пункте меню.

  Если в этом тексте перед одной из букв поместить символ "&", то буква в пункте меню будет подчеркнута и клавиша этой буквы будет назначена "горячей" клавишей для быстрого доступа к данному пункту меню.

- `Checked`

  Если это свойство имеет значение True, при работе приложения слева от наименования выбранного пункта меню появляется галочка.

- `Enabled` — свойство, определяющее возможность выполнения команды (пункта) меню.

  В зависимости от контекста объекта команды запрещаются или разрешаются.

- `HelpContextiD`

  Идентификатор справочной системы, соответствующий справке об этом меню.

- `Index`

  Идентификатор пункта меню в массиве элементов управления приложения.

- `NegotiatePosition`

  Определяет положение меню на экране.

- `Shortcut`

  Комбинация клавиш для быстрого выполнения пункта меню.

- `Visible`

  Определяет видимость на экране пункта меню. При работе приложения с помощью этого свойства пункты меню можно динамически прятать или показывать.

- `windowList`

  Назначает свойство формирования динамического списка окон. При установке этого свойства в меню будет добавляться список окон по мере их запуска при работе приложения. Это

свойство обычно используется для пункта меню самого верхнего уровня и для родительского окна приложений с интерфейсом типа MDI.

Любое приложение создается для реализации комплекса функций, обеспечивающих выполнение общей задачи приложения. Для быстрого доступа ко всем функциям приложения используются: главное меню приложения и контекстное меню отдельных объектов приложения (форм, панелей). Как и любой другой объект приложения, меню имеет набор свойств. Свойства меню доступны для редактирования в окне **Properties** (Свойства) формы, которой принадлежит меню.

4.10.9. Редактор меню *Menu Editor*

Для проектирования меню всех видов используется редактор меню **Menu Editor** (Редактор меню) среды проектирования IDE. Редактор меню вызывается одним из следующих способов:

- командой **Menu Editor** (Редактор меню) меню **Tools** (Инструменты);
- нажатием кнопки **Menu Editor** на стандартной панели инструментов;
- нажатием комбинации клавиш <Ctrl>+<E>.

Редактор создает меню для активного в данный момент окна, то есть, если активно MDI-окно, меню проектируется именно для него, если активна дочерняя форма, проектируется меню для дочерней формы.

Редактор меню состоит из двух групп: *элементов управления свойствами* и *элементов конструирования структуры меню*. Управлять основными свойствами меню, о которых было сказано ранее, можно с помощью следующих элементов редактора меню:

- поле **Caption** (Заголовок) — наименование пункта меню, то есть текст, появляющийся в меню;
- поле **Name** (Имя) — имя меню. Используется для идентификации объекта при написании программных кодов;
- раскрывающийся список **Shortcut** (Оперативная клавиша) — назначает комбинацию клавиш для быстрого вызова команды меню;

- поле **HelpContextID** (Идентификатор справки) — ссылка на тему в справочной системе;
- флажок **Enabled** (Доступно) — доступ к пункту меню;
- флажок **Visible** (Видимость) — определяет, будет ли виден на экране элемент меню;
- флажок **WindowList** (Список окон) — определяет наличие списка открытых окон.

Элементы группы конструирования структуры меню позволяют добавлять и удалять новые пункты, перемещать их по вертикали, меняя порядок следования, и по горизонтали, меняя расположение пунктов в иерархии системы меню:

- кнопки с направленными вправо и влево стрелками перемещают пункты или команды меню в иерархии меню;
- кнопки с направленными вверх и вниз стрелками перемещают пункты или команды меню по структуре меню;
- **Next** (Следующий) — перемещает указатель к следующему пункту меню. Если указатель находится на последнем пункте меню, то создается новый пункт меню или новая команда меню такого же уровня иерархии;
- **Insert** (Вставить) — добавляет пункт меню или команду в пункт меню;
- **Delete** (Удалить) — удаляет пункт меню или команду из пункта меню.

4.10.10. Элементы управления

Создание Windows-приложений в Visual Basic практически невозможно без использования элементов управления, так как они позволяют пользователю взаимодействовать с этими приложениями. Набор таких элементов управления не ограничен и может расширяться за счет так называемых пользовательских элементов управления (custom controls). Главное, что следует знать при работе с элементами управления, — это то, что к ним можно обращаться как к переменной, присваивая значения определенным свойствам или считывая их. Свойства определяют внешний вид и функционирование элемента управления.

Большинство свойств элементов управления доступно как для считывания, так и для изменения. Но есть свойства, которые во время выполнения доступны только для чтения (Read Only); другие же могут быть недоступны при проектировании. Сведения о доступности свойств (для чтения или для изменения, при проектировании или во время выполнения) содержатся в справочном файле Visual Basic.

Запомнить все свойства всех элементов управления практически невозможно. Для получения информации о каком-либо элементе управления, его свойствах, методах и событиях следует обратиться к справке. Для этого выделите соответствующий элемент управления на панели элементов и нажмите клавишу <F1>. После этого Visual Basic предоставит всю необходимую информацию.

Основные свойства элементов управления

Рассмотрим основные свойства, которыми обладает большинство элементов управления.

Позиция

Позицию элемента управления определяют четыре свойства: `Left`, `Top`, `Height` и `Width`. Эти значения по умолчанию используют в качестве единицы измерения твип (`twip`). *Твип* — это экранно-независимая единица измерения, равная 1/20 точки принтера и гарантирующая независимость отображения элементов приложения от разрешения дисплея.

Цвет

Управление цветовым оформлением элементов осуществляется с помощью свойств `BackColor`, `FillColor` и `ForeColor`, которым по умолчанию назначаются стандартные цвета Windows.

Цвет фона устанавливается с помощью свойства BackColor. При проектировании цвет выбирают в диалоговом окне настройки цвета, а во время работы приложения цвета задаются либо с использованием цветовой схемы RGB, либо константами библиотеки VBRUN.

С помощью свойства `ForeColor` можно определить или установить цвет, используемый для отображения текста и графики в элементе управления, а `FillColor` — установить цвет заполнения так называемых Shapes (рисованных объектов).

Параметры шрифта

Вид шрифта в элементах управления выбирается путем установки значений свойства `Font`.

Свойство — Значение

`Font.Name` — Имя шрифта

`Font.Size` — Размер шрифта

`Font.Bold` — Полужирный

`Font.Italic` — Курсив

`FontUnderline` — Подчеркивание

`Font.StrikeThrough` — Перечеркивание

`Font.Weight` — Толщина символа

Доступность и видимость элемента управления

Часто при работе приложения требуется сделать недоступными для пользователя некоторые элементы управления. Для этого используют два свойства — `Enabled` и `Visible`.

Свойство `Enabled` определяет, будет ли элемент управления реагировать на событие или нет. Если значение свойства равно False, элемент управления будет недоступен и пользователь не сможет его использовать. Обычно при этом элемент подсвечивается серым цветом, так же, как элементы меню, которые нельзя выбрать.

Свойство `Visible` позволяет сделать элемент управления невидимым. Если его значение равно False, то он не виден и обратиться к нему нельзя.

Выбор свойства (`Visible` либо `Enabled`) остается за вами. Если вы выбираете свойство `Enabled`, это означает, что элемент управления есть, но обратиться к нему пока невозможно. А свойство `Visible` позволяет "скрыть" элемент от пользователя.

```
Private Sub Cominand1_Click ()
Command1.Enabled = False
End Sub Private Sub
Command2_Click()
Command2 .Visible = False
End Sub
```

Свойство *Name*

Свойство `Name` играет особую роль. Ошибки при его задании часто приводят к серьезным последствиям. Имя является идентификатором элемента управления. Если в приведенном примере изменить имя второй кнопки, то код больше не будет выполняться, так как элемента с именем Command2 нет.

Окажется невозможным и изменить свойства кнопки Command2. Поэтому сначала всегда следует задавать имя элемента управления и лишь затем писать для него код обработки его события.

Внешний вид

Большинство элементов управления имеет свойство `Appearance`, отвечающее за отображение элемента управления (без визуальных эффектов или в трехмерном виде).

Кроме того, для большинства элементов управления можно установить значение свойства `ToolTipText`. Введенный текст отображается в подсказке, которая появляется, если пользователь установит указатель мыши на элементе управления в форме.

Свойства *Parent* и *Container*

Большая часть элементов управления Visual Basic имеет также свойства `Parent` и `Container`.

Свойство `Parent` указывает на родительский объект. Благодаря этому свойству возможен доступ к его методам или свойствам. В следующем примере строка заголовка формы, которой принадлежит соответствующая кнопка, сохраняется в переменной `strCaption`:

```
strCaption$ = Commandl.ParentCaption
```

Свойство `Container`, на первый взгляд, действует аналогично. Однако в отличие от свойства `Parent`, доступного только для чтения, свойство `Container` позволяет не только считывать, но и изменять контейнер элемента управления.

В примере путем изменения свойства `Container` кнопка перемещается в элемент pictureBox:

```
Set Commandl.Contanier = Picturel
```

Следующие элементы управления могут служить контейнером для других элементов управления:

- Form;
- Frame;
- Picture;
- Toolbar (издание для профессионалов и промышленное).

Свойство *Tag*

В отличие от других свойств. Visual Basic не использует свойство `Tag` для управления элементом управления — это свойство предназначено для хранения любых дополнительных данных, необходимых разработчику:

```
Text.Tag = "Ввод имени по-русски"
```

Основные события элементов управления

В этом разделе рассматриваются события, которые могут обрабатываться большинством элементов управления.

События щелчка мыши

Есть два события, вызываемые щелчком мыши: `Click` и `DblClick`.

Событие *Click*

Событие `Click` вызывается, как только пользователь выполнит щелчок на элементе управления.

Событие *DblClick*

Событие `DblClick` вызывается двойным щелчком кнопкой мыши на элементе управления. Временной интервал между двумя щелчками двойного щелчка устанавливается в панели управления Windows. Параметры для процедур обработки этих событий не передаются.

Кроме основных событий `Click` и `DblClick`, есть еще три.

Событие *MouseDown*

Событие `MouseDown` вызывается при нажатии кнопки мыши. При этом процедуре обработки события передается несколько параметров:

```
Control MouseDown(Button As Integer, Shift As Integer, _
```

```
X As Single, Y As Single)
Private Sub Commandl_MouseDown(Button As Integer, Shift As_
Integer, X As Single, Y As Single) End Sub
```

Передаваемые параметры определяют состояние кнопок мыши (Button), управляющих клавиш (<Shift>) и позицию курсора (*X* и *Y*). Параметры *X* и *Y* определяют позицию курсора мыши на экране относительно верхней левой точки элемента управления.

Параметр — Значение

`Button` — Нажата кнопка мыши: 1 = левая, 2 = правая, 4 = средняя

`Shift` — Нажата клавиша: 0 = ничего, 1 = [Shift], 2 = [Ctrl], 4 = [Alt], а также их комбинации

`X` — Координата X

`Y` — Координата Y

Событие *MouseUp*

Событие `Mouseup` вызывается при отпускании кнопки мыши.

Событие *MouseMove*

Это событие вызывается, когда пользователь передвигает курсор мыши. Синтаксис процедуры обработки этого события следующий:

```
Control_MouseMove (Button As Integer, Shift As Integer,
X Single,Y As Single)
```

Событие *KeyPreview*

Подобно событиям, связанным с мышью, есть также события, связанные с клавиатурой: `Keypress`, `KeyUp` и `KeyDown`. Обычно событие вызывается для активного элемента управления. Если свойству формы `KeyPreview` присвоить значение True, то событие, связанное с клавиатурой, передается сначала форме, а затем текущему элементу управления.

Событие *KeyPress*

Событие `KeyPress` возвращает код ASCII нажатой клавиши. При этом не перехватываются специальные клавиши, такие как <PrintScreen> или <Alt>, а только <Enter>, <Esc> и <Backspace>.

Процедура передает параметр `KeyASCII`, содержащий код ASCII нажатой клавиши. Этот параметр передается как значение, т. е. его можно изменять. Это можно использовать, например, для фильтрации вводимых пользователем символов — если символ недопустимый, то, установив значение `KeyASCII` равным нулю, вы предотвратите его передачу для дальнейшей обработки (отображение и т. п.).

```
ControI_KeyPress (KeyAscii As Integer)
```

События *KeyDown*, *KeyUp*

Эти события вызываются при нажатии (`KeyDown`) или отпускании (`KeyUp`) клавиши. Они происходят даже при нажатии специальных клавиш управления, например функциональных клавиш. При этом передаются два параметра: `KeyCode` и `Shift`. Параметр `KeyCode` содержит клавиатурный код (а не ASCII) нажатой клавиши, например `vbKeyF1`, а параметр `Shift` информирует о состоянии клавиш <Shift>, <Ctrl> и <Alt>.

```
Control_KeyUp( KeyCode As Integer, Shift As Integer)
Control_KeyDown (KeyCode As Integer, Shift As Integer)
```

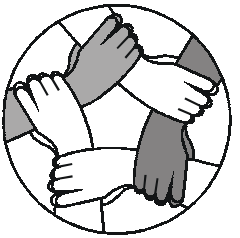
После нажатия клавиши события наступают в такой последовательности: `KeyDown`, `KeyPress` и `KeyUp`.

Фокус

Фокус — это одно из важных понятий при обращении к элементам управления в Windows. Как уже упоминалось, система Windows решает, какому приложению передавать нажатие клавиши — управление получает активный элемент, т. е. элемент, имеющий фокус. Если элемент получает фокус, то это соответствующим образом отображается на экране — текстовое поле отображается с мерцающим маркером ввода, командная кнопка выделяется пунктирной рамкой вокруг надписи.

```
События LostFocus, GotFocus
```

Visual Basic позволяет обрабатывать два события, связанных с передачей фокуса: `LostFocus` и `GotFocus`. Если перейти от одного элемента управления к другому, то для предыдущего элемента вызывается событие `LostFocus`, а для нового — `GotFocus`.



Глава 5

Для тех, кому тяжело… Решения некоторых алгоритмических задач

Для первых решений я рассказываю, что означают те или иные предполагаемые объекты на экранной форме, а затем — развивайте пространственное воображение и логическое мышление. Вообще, я буду давать чаще алгоритмическое решение, а воплощение алгоритма в визуальной форме VB — ваше дело.

Задание 7. Тригонометрический калькулятор

Пусть в TextBox1 будут размещены градусы, в TextBox2 — минуты, а в TextBox3 — секунды. В TextBox4 выводится результат вычислений.

Тогда программный код будет выглядеть так:

```
Const Pi = 3.14
Dim grad As Single
Dim min As Single
Dim sec As Single
Dim angle As Single
Private Sub cmdSin_Click()
  grad = Text1.Text
  min = Text2.Text
  sec = Text3.Text
  angle = (grad + min / 60 * 100 + sec / 60 * 100) * Pi / 180
```

```
  Text4.Text = Sin(angle)
End Sub
Private Sub cmdCos_Click()
  grad = Text1.Text
  min = Text2.Text
  sec = Text3.Text
  angle = (grad + min / 60 * 100 + sec / 60 * 100) * Pi / 180
  Text4.Text = Cos(angle)
End Sub
Private Sub cmdTg_Click()
  grad = Text1.Text
  min = Text2.Text
  sec = Text3.Text
  angle = (grad + min / 60 * 100 + sec / 60 * 100) * Pi / 180
  Text4.Text = Tan(angle)
End Sub
Private Sub cmdCtg_Click()
  grad = Text1.Text
  min = Text2.Text
  sec = Text3.Text
  angle = (grad + min / 60 * 100 + sec / 60 * 100) * Pi / 180
  Text4.Text = 1 / Tan(angle)
End Sub
```

Задание 8. 5000 прожитых дней

```
Private Sub Command1_Click()
Dim n As Date
Cls
n = InputBox("Введите дату вашего рождения")
n = n + 5000
Print "Вы прожили 5000 дней в этот день →";n
End Sub
```

Задание 12. Площадь треугольника по формуле Герона

На форме размещаются четыре TextBox (три — для ввода длин сторон и одно — для вывода результата) и одна CommandButton.

```
Private Sub Command1_Click()
' Считывание сторон треугольника
a = Val(Text1.Text)
b = Val(Text2.Text)
c = Val(Text3.Text)
' расчет полупериметра
p = (a+b+c) / 2
' расчет площади
Text4.Text = (p * (p - a) * (p - b) * (p - c)) ^ (1 / 2)
End Sub
```

Задание 13. Обмен

```
Private Sub Command1_Click()
a = Val(Text1.Text)
b = Val(Text2.Text)
r = a
a = b
b = r
Text1.Text = a
Text2.Text = b
End Sub
```

Задание 16. Теория биоритмов

```
Private Sub Command1_Click()
Dim d1 As Date, d2 As Date
d1 = InputBox("Введите дату вашего рождения")
d2 = InputBox("Введите сегодняшнее число")
fiz = (d2 - d1) Mod 23
```

```
emot = (d2 - d1) Mod 28
intel = (d2 - d1) Mod 33
Print "Номер дня вашего физического цикла ";fiz
Print "Номер дня вашего эмоционального цикла ";, emot
Print "Номер дня вашего интеллектуального _
цикла "; intel
End Sub
```

Задание 28. Дальность полета

Для универсальности программы запросим все исходные данные с клавиатуры. Кроме того, напомним ту самую злополучную формулу дальности полета:

$$l = \frac{V_0 \sin 2\alpha}{g^2}$$

```
Private Sub Command1_Click()
Dim Speed As Single
Dim Angle As Single
Dim Dalnost As Single
Const G = 9.81
Const PI = 3.14
Speed = Val(Text1.Text)
Angle = Val(Text2.Text)
Dalnost = Speed*SIN (Angle*PI/180)/G^2
Text3.Text = Dalnost
End Sub
```

Для заданий 38–42 привожу только расчетные формулы.

Задание 38. Площадь круга

$S = \pi R^2$.

Задание 39. Длина окружности

$\tilde{N} = 2\pi R$.

Задание 40. Площадь ромба

$S = ab/2$, где a и b — диагонали ромба.

Задание 41. Площадь равнобедренной трапеции

$S = h(a + b/2)$, где a и b — основания трапеции, а h — высота.

Задание 42. Объем цилиндра

$V = \pi R^2 h$.

Задание 43. Нахождение координат середины заданного отрезка

Если известны координаты двух точек ($x1$, $y1$) и ($x2$, $y2$), то расстояние между ними можно вычислить по формуле:

$$S = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}\ .$$

А дальше — дело математики — Система уравнений (и золотой ключик у нас в кармане).

С одной стороны, x и y — искомые координаты середины отрезка

$$(S/2)^2 = (x - x1)^2 + (y - y1)^2,$$

а с другой стороны

$$(S/2)^2 = (x - x2)^2 + (y - y2)^2.$$

На форме задаются координаты концов отрезка, нажимается кнопка **Расчет** и появляются координаты середины.

Задание 65. Биоритмы

```
Option Explicit
Dim x As Date, y As Date
```

```
Dim z As Long, f As Long, e As Long, i As Long
Private Sub Command1_Click()
x = CDate(Text1.Text)
y = CDate(Text2.Text)
z = y - x
Text3.Text = "Вы прожили на свете " & z & " дней"
f = z Mod 23
e = z Mod 28
i = z Mod 33
Select Case f
Case Is > 11
Text4.Text = Str(f) & "-й день физического цикла. Он в отри-
цательной фазе"
Case Is < 12
Text4.Text = Str(f) & "-й день физического цикла. Он в поло-
жительной фазе"
End Select

Select Case e
Case Is > 14
Text5.Text = Str(e) & "-й день эмоционального цикла. Он в от-
рицательной фазе"
Case Is < 14
Text5.Text = Str(e) & "-й день эмоционального цикла. Он в по-
ложительной фазе"
End Select

Select Case f
Case Is > 15
Text6.Text = Str(i) & "-й день интеллектуального цикла. Он в
отрицательной фазе"
Case Is < 16
Text6.Text = Str(i) & "-й день интеллектуального цикла. Он в
положительной фазе"
End Select
End Sub
```

Задание 102. Гиперболоид

```
Option Explicit
Dim a As Integer, b As integer, c As Integer
Dim x As Integer, y As Single
Dim i As Integer
Private Sub Command1_Click()
Picture1.Scale (-5, 8)-(5, -8)
a = Text1.Text
b = Text2.Text
c = Text3.Text

Picture1.Line (-5, 0)-(5, 0)
For i = -5 To 5
Picture1.PSet (i, 0)
Picture1.Print i
Next

Picture1.Line (0, -8)-(0, 8)
For i = -8 To 8
Picture1.PSet (0, i)
Picture1.Print i
Next

For x = -5 To 5 Step 0.05
y = a * x * x + b * x + c
Picture1.Circle (0, y), Abs(x), , , , 0.1
Next
End Sub
```

Задание 110. Циклы с зависимыми переменными

```
Option Explicit
Dim x As Integer, y As Integer, r As integer
```

```
Dim i As Integer, a As integer, b As Integer
Dim c As Integer
Dim k As Single
'Рупор
Private Sub Command1_Click()
Picture1.Line (0, 0)-(6000, 3500), vbWhite, BF
For x = 0 To 4000 Step 100
Picture1.Circle (x, 1750), x / 3
Next
End Sub
'Четыре рупора
Private Sub Command2_Click()
Picture1.Scale (0, 3500)-(6000, 0)
'Picture1.Line (0, 0)-(6000, 3500), vbWhite, BF
Cls
For x = 0 To 3000 Step 75
Picture1.Circle (x, 35 * x / 60), x / 5
Next
For x = 0 To 3000 Step 75
Picture1.Circle (6000 - x, 35 * x / 60), x / 5
Next
For x = 0 To 3000 Step 75
Picture1.Circle (x, 3500 - 35 * x / 60), x / 5
Next
For x = 0 To 3000 Step 75
Picture1.Circle (6000 - x, 3500 - 35 * x / 60), x / 5
Next
End Sub
'Лестница
Private Sub Command3_Click()
Picture1.Cls
For x = 1000 To 4500 Step 400
Picture1.Line (x, 4400 - x)-(x + 1000, 4200 - x), , B
Picture1.Line (x, 4200 - x)-(x + 400, 4000 - x)
Picture1.Line (x + 1000, 4200 - x)-(x + 1400, 4000 - x)
```

```
Next
End Sub
'Пирамида сверху
Private Sub Command4_Click()
Picture1.Line (0, 0)-(6000, 3500), vbWhite, BF
For x = 1500 To 3000 Step 150
Picture1.Line (x, x - 1250)-(6000 - x, 4750 - x), , B
Next x
End Sub
'Пирамида сбоку
Private Sub Command5_Click()
Picture1.Line (0, 0)-(6000, 3500), vbWhite, BF
For x = 500 To 3000 Step 150
Picture1.Line (x, 4000 - x)-(6000 - x, 3850 - x), , B
Next x
End Sub
```

Задание 116. Вычисление синуса

```
Option Explicit
Dim x As Integer, y As Single
Private Sub Command1_Click()
FontSize = 12
CurrentY = 200
CurrentX = 200
Print "Вычисление значений функции y=sin x"
CurrentX = 200
Print "на интервале от -10 до 10 с шагом 2"
For x = -10 To 10 Step 2
CurrentX = 300
y = Sin(x)
Print "x="; x, "y="; y
Next
End Sub
```

Задание 126. Нахождение первого числа Фибоначчи больше заданного

```
m=inputbox("введите любое  число, большее 1 ", "окно ввода")
m=cInt(m)
a=1
b=1
F=0
Do
F=a+b
a=b
b=F
LOOP while m>=F
msgbox "F="&F,,"окно вывода"
```

Задание 130. Сумма цифр заданного натурального числа

```
'm-исходное число, s-сумма цифр числа
‘x-цифры числа, начиная с цифры справа
‘n-исходное число для печати
 m=inputbox("введите любое  натуральное число ", "окно
ввода")
s=0
n=m
do
x=m mod 10
s=s+x
m=(m-x)/10
Loop while m>0
msgbox "сумма цифр числа "&n&"="&s,,"окно вывода"
```

Задание 155. Шахматная доска и лоскутный ковер

```
Option Explicit
Dim x As Integer, y As Integer
Dim As Integer n As Integer
Dim a As Integer, b As Integer, c As Integer
Private Sub Command1_Click()
Picture1.Line (0, 0)-(4800, 4800), vbWhite, BF
n = 2
For x = 400 To 4000 Step 500
For y = 400 To 4000 Step 500
If n Mod 2 = 0 Then
Picture1.Line (x, y)-(x + 500, y + 500), vbRed, BF
Picture1.Line (x, y)-(x + 500, y + 500), , B
Else
Picture1.Line (x, y)-(x + 500, y + 500), vbYellow, BF
Picture1.Line (x, y)-(x + 500, y + 500), , B
End If
n = n + 1
Next
n = n + 1
Next
End Sub

Private Sub Command2_Click()
Randomize
For x = 0 To 4600 Step 200
For y = 0 To 4600 Step 200
a = Fix(Rnd * 256)
b = Fix(Rnd * 256)
c = Fix(Rnd * 256)
Picture1.Line (x, y)-(x + 200, y + 200), RGB(a, b, c), BF
Picture1.Line (x, y)-(x + 200, y + 200), , B
Next
```

```
Next
End Sub
Задание 157. Метод Монте-Карло
Private Sub Command1_Click()
Randomize
Picture1.Scale (-0.5, 2.5)-(2.5, -0.5)
Picture1.Line (0, -0.5)-(0, 2.5)
Picture1.Line (-0.5, 0)-(2.5, 0)
Picture1.Line (0, 2)-(2, 0), , B
Picture1.Circle (1, 1), 1
For k = 0 To 2 Step 1
Picture1.CurrentX = k
Picture1.CurrentY = 0
Picture1.Print k
Next
n = Text1.Text: n1 = 0
For i = 1 To n
x = Rnd(1) * 2
y = Rnd(1) * 2
Picture1.PSet (x, y)
If (x - 1) ^ 2 + (y - 1) ^ 2 <= 1 Then n1 = n1 + 1
Next
Pi = 4 * n1 / n
Picture1.Line (0.05, 1.25)-(1.95, 0.75), vbWhite, BF
Picture1.Line (0.05, 1.25)-(1.95, 0.75), vbBlack, B
Picture1.CurrentX = 0.15
Picture1.CurrentY = 1.1
Picture1.FontSize = 12
Picture1.Print "Пи="; Pi
End Sub
```

Задание 161. "Муха в графине"

```
Option Explicit
Dim x As Integer
Dim y As Integer, dx As Integer, dy As Integer
```

```
Dim n As Integer, i As Integer
Private Sub Command1_Click()
x = 2183: y = 1777
Picture1.Line (1300, 1600)-(4200, 4000), , BF
Picture1.Line (2200, 600)-(2600, 1600), , BF

Picture1.DrawWidth = 6
n = 1
dx = 1: dy = 1
While n < 32000
If x = 50 Or x = 5900 Then dx = -dx
If y = 50 Or y = 4500 Then dy = -dy
If x = 2250 And y < 1650 And y > 550 Then dx = -dx
If y = 1650 And x > 1350 And x < 2250 Then dy = -dy
If x = 1350 And y < 3950 And y > 1650 Then dx = -dx
If y = 3950 And x > 1350 And x < 4150 Then dy = -dy
If x = 4150 And y < 3950 And y > 1650 Then dx = -dx
If y = 1650 And x > 2550 And x < 4150 Then dy = -dy
If x = 2550 And y < 1650 And y > 550 Then dx = -dx
Picture1.PSet (x, y), vbRed
For i = 1 To 5000: Next
Picture1.PSet (x, y), vbBlack
x = x + dx: y = y + dy
n = n + 1
Wend
End Sub
```

Задание 166. Косой дождь

```
Dim B As Boolean
Private Sub Form_Load()
B = False
End Sub

Private Sub Form_Click()
```

```
Dim c As Long
Randomize
B = Not B
c = 0
Do While B And c < 5000
x = Rnd * 5640 'ScaleWidth
y = Rnd * 4690 'ScaleHeigth
Line (x, y)-(x - 150, y + 200)
For i = 1 To 30000: Next
DoEvents
c = c + 1
Loop
End Sub
```

Задание 173. Графическая интерпретация числового одномерного массива

```
Private Sub Command1_Click()
Cls
Randomize
Dim M(1 To 10)
For i = 1 To 10
  M(i) = Fix(Rnd(1) * 150) + 50
Next i
P.Scale (0, 250)-(400, 0)
x = 50
For i = 1 To 10
P.Line (x, 0)-(x + 30, M(i)), vbGreen, BF
P.Line (x, 0)-(x + 30, M(i)), vbBlack, B
P.PSet (x + 2, 40), vbGreen
P.Print (M(i))
x = x + 30
Next i
End Sub
```

Задание 174. Графическая интерпретация числового одномерного массива

```
Private Sub Command1_Click()
Cls
Randomize
Dim M(1 To 11)
For i = 1 To 10
  M(i) = Fix(Rnd(1) * 30) + 5
Next i
P.Scale (0, 250)-(400, 0)
x = M(1) + 5
P.FillColor = vbRed
P.FillStyle = 0
For i = 1 To 10
P.Circle (x, 100), M(i), vbBlack
P.PSet (x - 7, 20), vbGreen
P.Print M(i);
x = x + M(i) + M(i + 1)
Next i
End Sub
```

Задание 202. Пожиратели звезд

```
Private Sub Command1_Click()
Randomize
Cls
Dim x(1 To 10000) As Integer
Dim y(1 To 10000) As Integer
p.Scale (0, 300)-(500, 0)
For i = 1 To 10000
x(i) = Fix(Rnd(1) * 500)
y(i) = Fix(Rnd(1) * 300)
```

```
p.PSet (x(i), y(i)), vbWhite
Next i
X2 = -30: x3 = -30: x4 = -20
p.FillStyle = 0
For X1 = 1 To 500
Y1 = 150 - 30 * (2 * Sin(X1 / 60) + 0.5 * Cos(2 * X1 / 30))
Y2 = 160 - 30 * (2 * Sin(X1 / 60) + 0.5 * Cos(2 * X1 / 30))
Y3 = 140 - 30 * (2 * Sin(X1 / 60) + 0.5 * Cos(2 * X1 / 30))
Y4 = 150 - 30 * (2 * Sin(X1 / 60) + 0.5 * Cos(2 * X1 / 30))
p.FillColor = vbGreen
p.Circle (X1, Y1), 2, vbGreen
p.Circle (X2, Y2), 3, vbGreen
p.Circle (x3, Y3), 3, vbGreen
p.Circle (x4, Y4), 3, vbGreen
For i = 1 To 10000
If Abs(X1 - x(i)) <= 20 And Abs(Y1 - y(i)) <= 20 Then
p.PSet (x(i), y(i))
End If
Next
p.FillColor = vbBlack
p.Circle (X1, Y1), 2
p.Circle (X2, Y2), 3
p.Circle (x3, Y3), 3
p.Circle (x4, Y4), 3
X2 = X2 + 1: x3 = x3 + 1: x4 = x4 + 1
Next X1
p.FillColor = vbGreen
p.Circle (X1, Y1), 2, vbGreen
p.Circle (X2, Y2), 3, vbGreen
p.Circle (x3, Y3), 3, vbGreen
p.Circle (x4, Y4), 3, vbGreen
End Sub
```

Задание 212. Сортировка методом "пузырька"

```
Private Sub Command1_Click()
Cls
Randomize
P.Scale (0, 250)-(400, 0)
Dim M(1 To 11)
For I = 1 To 10
  M(I) = Fix(Rnd(1) * 200) + 50
Next I
X = 50
For I = 1 To 10
P.Line (X, 0)-(X + 30, M(I)), vbGreen, BF
P.Line (X, 0)-(X + 30, M(I)), vbBlack, B
P.PSet (X + 2, 20), vbGreen
P.Print M(I);
X = X + 30
Next I

P1.Scale (0, 250)-(400, 0)
For I = 1 To 9
  For j = 1 To 9
    If M(j) > M(j + 1) Then
    r = M(j)
    M(j) = M(j + 1)
    M(j + 1) = r
    End If
  Next j
Next I
X = 50
For I = 1 To 10
P1.Line (X, 0)-(X + 30, M(I)), vbGreen, BF
P1.Line (X, 0)-(X + 30, M(I)), vbBlack, B
P1.PSet (X + 2, 20), vbGreen
```

```
P1.Print M(I);
X = X + 30
Next I

End Sub
```

Задание 274. Программа вычисления числового значения

```
function fact(k)
fact=1
for i=1 to k
fact=fact*i
next
end function

n=inputbox("Введите значение n")
if n>=5 then
msgbox "y= "&(fact(n)^3)+4
else
msgbox "y= "&sin(fact(n))
end if
```

Задание 278. Нахождение наибольшего общего делителя (НОД)

```
function nod(a,b)
while (a<>0) and (b<>0)
if (a>=b) then
a=a mod b
else
b=b mod a
end if
nod=a+b
wend
```

```
end function

x=inputbox("Введите целое число X")
y=inputbox("Введите целое число Y")
z=inputbox("Введите целое число Z")
n=nod(x,y)
n=nod(n,z)
msgbox "Значение НОД ="&n
```

Задание 280. Сравнение площадей треугольников

```
function s(a1,b1,a2,b2,a3,b3)
a=sqr((a1-a2)^2+(b1-b2)^2)
b=sqr((a1-a3)^2+(b1-b3)^2)
c=sqr((a2-a3)^2+(b2-b3)^2)
p=(a+b+c)/2
s=sqr(p*(p-a)*(p-b)*(p-c))
end function

sub sr(t) 'сравнение
if m>n then t=true else t=false
end sub

x1=inputbox("Введите координату X1")
y1=inputbox("Введите координату Y1")
x2=inputbox("Введите координату X2")
y2=inputbox("Введите координату Y2")
x3=inputbox("Введите координату X3")
y3=inputbox("Введите координату Y3")

k1=inputbox("Введите координату k1")
g1=inputbox("Введите координату g1")
k2=inputbox("Введите координату k2")
g2=inputbox("Введите координату g2")
```

```
k3=inputbox("Введите координату k3")
g3=inputbox("Введите координату g3")

m=s(x1,y1,x2,y2,x3,y3)
n=s(k1,g1,k2,g2,k3,g3)
sr q

if q then
msgbox "1-й больше, его площадь " _
&m&" у второго площадь "&n
else
msgbox "2-й больше, его площадь " _
&m&" у первого площадь "&n
end if
```

Заключение

Вот и закончилась моя немного, может быть, сумбурная книга. IMHO (как говорят в Интернете, по моему скромному мнению) для тех, кто хочет стать программистом, очень важна алгоритмика. Задач на нее в этой книге предостаточно. Алгоритмика вообще полезна для всех. Освоив основные алгоритмические конструкции, можно переходить на другие языки программирования более просто, так же, как однажды научившись водить машину, легко пересеть на другую.

В чем, собственно, и желаю вам удачи!

И до новых встреч!

Интернет-ресурсы и рекомендуемая литература

1. **www.vbstreets.narod.ru**
2. **www.vbrussian.com**
3. **www.omegasoft.narod.ru**
4. **www.basicsoft.narod.ru**
5. **www.ishodniki.ru**
6. **www.vbnet.ru**
7. **www.easy-vb.narod.ru**
8. **www.intsoftware.km.ru**
9. Культин Н. Visual Basic Освой на примерах. — СПб.: БХВ-Петербург, 2004.
10. Ананьев А., Федоров А. Самоучитель Visual Basic 6.0. — СПб.: БХВ-Петербург, 2003.
11. Microsoft Visual Basic. Шаг за шагом. — М.: ЭКОМ, 2002.
12. Симонович С., Евсеев Г. Занимательное программирование Visual Basic. — М.: АСТ-ПРЕСС КНИГА, 2004.
13. Волчёнков Н. Г. Программирование на Visual Basic 6. — М.: ИНФРА-М, 2002.
14. Литвиненко Т. В. Visual Basic 6.0. — М.: Горячая линия — Телеком, 2001.
15. Угринович Н., Босова Л., Михайлова Н. Практикум по информатике и информационным технологиям. — М.: Лаборатория базовых знаний, 2001.