Андрей Гарнаев
Лада Рудикова

Microsoft Office Excel 2010 разработка приложений

Санкт-Петербург
«БХВ-Петербург»
2011

УДК 681.3.06
ББК 32.973.26-018.2
Г20

Гарнаев, А. Ю.

Г20 Microsoft Office Excel 2010: разработка приложений / А. Ю. Гарнаев, Л. В. Рудикова. — СПб.: БХВ-Петербург, 2011. — 528 с.: ил. + (CD-ROM) — (Профессиональное программирование)

ISBN 978-5-9775-0042-5

Продемонстрированы широкие возможности Microsoft Office Excel 2010 по созданию приложений средствами VBA, работе с макросами, технологии ООП, конструированию пользовательского интерфейса и форм. Рассмотрены вопросы автоматизации операций с рабочим листом и диаграммами, в том числе при обработке и анализе данных и принятии решений. Изложены методы интеграции офисных приложений, работы с Интернетом и базами данных, применения XML. Книга содержит более 300 примеров тщательно разработанных приложений: от создания пользовательских функций до построения информационных систем по сбору и обработке данных, программный код которых может быть непосредственно использован читателем при разработке собственных проектов.

Прилагаемый компакт-диск содержит файлы рассмотренных в книге примеров.

Для программистов, преподавателей и студентов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 27.06.11.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 42,57.
Тираж 1500 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 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 978-5-9775-0042-5

Оглавление

Введение

Microsoft Excel — ведущая программа обработки электронных таблиц. Первая версия MS Excel появилась в 1985 году и обеспечивала только простые арифметические операции в строку или в столбец.

В настоящее время Microsoft Office Excel 2010 представляет собой достаточно мощное средство разработки информационных систем, которое включает как электронные таблицы (со средствами финансового и статистического анализа, набором стандартных математических функций, доступных в компьютерных языках высокого уровня, рядом дополнительных функций, встречающихся только в библиотеках дорогостоящих инженерных подпрограмм), так и средства визуального программирования (Visual Basic for Applications). Электронные таблицы позволяют производить обработку чисел и текста, задавать формулы и функции для автоматического выполнения, прогнозировать бюджет на основе сценария, представлять данные в виде диаграмм, публиковать рабочие листы и диаграммы в Интернете. С помощью VBA можно автоматизировать всю работу, начиная от сбора информации, ее обработки до создания итоговой документации, как для офисного пользования, так и для размещения на веб-узле.

О чем эта книга?

Популярность табличного процессора Microsoft Office Excel показывает, что интерес к этому приложению будет расти и дальше. Поэтому рассмотрение тех или иных задач, которые можно решить с использованием его возможностей, а тем более, с использованием языка Visual Basic for Applications, несомненно, расширяет области применения Microsoft Office Excel.

Предлагаемая книга на различных примерах демонстрирует широкие возможности Microsoft Office Excel 2010 как для решения расчетных задач, задач визуализации, обработки информации и т. д., так и для создания пользовательских приложений средствами VBA. Так, в книге описан объектно-ориентированный подход к программированию офисных приложений, рассмотрены вопросы создания пользовательского интерфейса, автоматизации операций с рабочим листом и диаграммами, в том числе и при обработке и анализе данных и принятии решений. Кроме того, изложены методы интеграции офисных приложений, работа с Интернетом. Достаточно подробно приведен материал по работе с базами данных. Несомненно, все

это позволит овладеть приемами создания завершенных приложений, позволяющих автоматизировать работу пользователя и обрабатывать данные различного плана.

Для большей наглядности и легкости усвоения материала в книге приводится более 300 примеров тщательно разработанных приложений: от создания пользовательских функций до построения информационных систем по сбору и обработке данных. Все примеры, встречающиеся по ходу изложения материала, сгруппированы по главам и содержатся на прилагаемом к книге компакт-диске. Код в любом разделе самодостаточен и может быть непосредственно использован читателем при разработке собственных проектов.

Книга состоит из 12 глав, каждая из которых посвящена определенной тематике и использованию соответствующих возможностей Microsoft Office Excel 2010. В конце книге имеются четыре приложения.

Для кого предназначена эта книга?

Материал книги может быть полезен:

- в качестве учебного пособия — при преподавании VBA и MS Excel для студентов математических, экономических и технических специальностей, изучающих MS Excel в различных курсах информатики, информационных технологий и систем обработки данных;
- преподавателям — при подготовке лекций и проведении практических и лабораторных работ;
- начинающим пользователям — для самостоятельного изучения MS Excel и VBA;
- профессионалам — в качестве справочника по самому языку, его объектной модели и современным приемам программирования, используемым при создании приложений.

Типографские соглашения

Способ отображения	Применение
Полужирное начертание	Команды меню, кнопки панелей инструментов, заголовки и вкладки диалоговых окон Visual Basic
Курсивное начертание	Имена файлов, которые вы найдете на компакт-диске, прилагаемом к книге, а также понятия и определения, на которые авторы книги хотят обратить внимание читетелей
`Шрифт Courier`	Методы, события, объекты и их свойства, листинги и фрагменты кода, составленные на языке VBA
`[Параметр]`	При описании синтаксиса параметр, заключенный в квадратные скобки, является необязательным
`{Параметр1 | Параметр2}`	Если при описании синтаксиса несколько параметров заключены в фигурные скобки и разделителем между ними является вертикальная черта, то эти параметры являются альтернативными, и предполагается использование только одного из них

(окончание)

Способ отображения	Применение
↳	Если при описании синтаксиса в начале строки расположен указанный символ, то это говорит о том, что данная строка является продолжением предыдущей
<Ctrl>+<R>	Знак "плюс" между названиями клавиш обозначает комбинацию клавиш (не отпуская первой клавиши, нужно нажать вторую)

От издательства

Чтобы получить дополнительные сведения или выразить свое отношение к нашей книге, вы можете обратиться в издательство "БХВ-Петербург" по адресу: **http://www.bhv.ru**.

Благодарности

Авторы выражают искреннюю благодарность Герману Ломакину, студенту факультета прикладной математики и информатики Белорусского государственного университета, за помощь, оказанную при подготовке рукописи к изданию. Кроме того, хотелось бы отметить весь коллектив издательства "БХВ-Петербург", без поддержки и понимания которого не появилась бы данная книга.

Заранее благодарим и всех наших читателей, для которых мы подготовили предлагаемый материал. Авторы надеются, что чтение книги для вас будет не только полезным, но и увлекательным. Все свои пожелания, замечания и предложения вы можете отправить на нашу электронную почту: **garnaev@yahoo.com** (Андрей Юрьевич Гарнаев, доктор физико-математических наук, профессор, Санкт-Петербург, Россия) и **rudikowa@gmail.com** (Лада Владимировна Рудикова, кандидат физико-математических наук, доцент, Гродно, Беларусь).

Глава 1

Быстрое начало — первые программы на VBA

Что такое VBA?

Visual Basic for Applications (VBA, Visual Basic для приложений) — это объектно-ориентированный язык программирования, который специально разработан для приложений Microsoft Office. Отличительной особенностью VBA является использование наряду с обычными переменными и константами также и имеющихся объектов приложений Microsoft Office, например, в Microsoft Office Excel 2010 это могут быть рабочие книги, рабочие листы, диапазоны ячеек, диаграммы и т. д. С помощью VBA можно разрабатывать приложения, которые включают различные компоненты нескольких приложений Microsoft Office и способствуют тем самым интеграции и совместному использованию данных.

VBA — достаточно легкий язык программирования. Он прост в освоении и позволяет быстро получать ощутимые результаты — конструировать профессиональные приложения, решающие практически все задачи, встречающиеся в среде Windows, а также удобен при создании клиент-серверных приложений. При этом проектирование многих приложений с использованием VBA проще и быстрее, чем с применением других языков программирования.

VBA использует технологию визуального программирования, т. е. конструирование рабочей поверхности приложения и элементов его управления непосредственно на экране, а также запись всей программы или ее частей при помощи макрорекордера.

VBA позволяет создавать универсальные, автоматизированные приложения на платформе MS Excel. Кроме того, используя технологию ActiveX, VBA позволяет как внедрять в разрабатываемое приложение объекты других приложений, так и, не выходя из созданного приложения, управлять другими Windows-приложениями. VBA позволяет создавать клиент-серверные приложения, связывая ваш компьютер со всем остальным миром.

Для того чтобы писать программы на языке VBA, необходимо четко представлять себе технологию объектно-ориентированного программирования (ООП). ООП можно описать как совокупность принципов, технологии и инструментальных средств для создания программных систем, основу которых составляет архитектура взаимодействия объектов. Объектная модель Microsoft Office и, в частности, объектная модель Microsoft Office Excel 2010 содержит множество различных объектов, которые образуют достаточно сложную иерархию. В свою очередь, каждый

объект обладает набором функциональных возможностей, а также способами воздействия на его состояние.

Итак, прежде чем приступить к созданию первых программ на языке VBA, познакомимся вкратце с такими важными понятиями ООП, как объект, свойство, метод, событие, класс и семейство объектов.

Примечание

Примеры, относящиеся к данной главе, вы можете найти в папке Glava_1 на прилагаемом к книге компакт-диске.

Объекты и не только

Объект (object) является основным понятием (основной парадигмой) ООП и представляет собой изменяемый элемент, содержащий данные вместе с кодом, предназначенным для их обработки. Любой объект в ООП обладает определенными *свойствами* (properties), которые описывают данный объект или его состояние, и *методами* (methods), отвечающими за действия, которые можно выполнить над объектом. На любой объект можно воздействовать одним из двух способов, изменяя его состояние: во-первых, можно изменить одно из свойств данного объекта и, во-вторых, можно выполнить некоторые действия, применив один из методов рассматриваемого объекта.

Объекты, которые имеют одинаковые свойства и методы, объединяются в ООП в абстрактное понятие — *класс* (class). Важной особенностью классов является возможность их организации в виде некоторой древовидной *иерархической* структуры. Верхний уровень данной иерархии занимает класс (родительский класс или предок), который охватывает наиболее общие свойства и методы, характерные для всех его подчиненных объектов. На низшем же уровне располагаются лишь те объекты (потомки), дальнейшая конкретизация которых невозможна.

Введем еще одно понятие, связанное с объектами, которое встретится вам при написании программ на VBA. Довольно часто можно столкнуться с набором однотипных объектов, которые, в свою очередь, образует объект *семейство*. Следует отметить, что если любой объект является реализацией конкретного класса, то семейство представляет собой коллекцию уже имеющихся похожих объектов.

Естественно, что, знакомясь с общей методологией ООП, нельзя обойти вниманием и такие его основные принципы, как наследование, инкапсуляция и полиморфизм.

Наследование (inheritance) связано непосредственно с иерархией классов и определяет возможность обладания определенными свойствами и методами. Так, если некоторый общий (родительский) класс обладает определенным набором свойств и методов, то класс, который связан с ним и находится на уровне ниже (потомок), обязан включать эти же свойства и методы, а также дополнительные, которые будут характеризовать уникальность данного потомка.

Сокрытие некоторых деталей реализации объектов по отношению к внешним объектам или пользователям называется *инкапсуляцией* (encapsulation). Как правило, в действительности не столь важно, каким образом реализован, например, тот или иной метод класса, к которому обращается пользователь. Идея инкапсуляции взята от деления модулей в некоторых языках программирования на две части: интерфейс и реализацию, и отражает наше представление об окружающем мире. Так, в ин-

терфейсной части дается вся информация, которая необходима для взаимодействия с другими объектами. Вся остальная информация, которая не относится к процессу взаимодействия объектов, скрывается в части, относящейся к реализации объекта.

Полиморфизм (polymorphism) предполагает, что действия, которые выполняются одноименными методами, но для различных классов объектов, могут существенно отличаться.

Если же мы обратимся к VBA, то убедимся, что объектная модель Microsoft Office содержит много различных объектов и образует сложную иерархию. Например, все визуальные элементы, такие как рабочий лист (`Worksheet`), диапазон (`Range`), диаграмма (`Chart`), форма (`UserForm`), являются в Microsoft Office Excel 2010 объектами. Более того, практически, все элементы, которые можно увидеть в этом приложении, включая само окно рабочей книги, являются объектами. Исключение составляют, например, кнопки **Свернуть** (Minimize) и две взаимозаменяемые кнопки **Свернуть в окно** (Restore) и **Развернуть** (Maximize), находящиеся в заголовке окна приложения Microsoft Excel.

Многие однотипные объекты объединяются в VBA в семейства (объект `Collection`). Например, объект `Workbooks` содержит все открытые объекты `Workbook` (рабочая книга). Каждый элемент семейства нумеруется и может быть идентифицирован либо по номеру, либо по имени. Например, `Worksheets(1)` обозначает первый рабочий лист активной книги, а `Worksheets("Лист1")` — рабочий лист с именем **Лист1.**

Если в программе на VBA нам необходимо указать ссылку на объект, то для конкретного объекта следует уточнить весь его иерархический путь. Например, чтобы присвоить значение 10 первой ячейке второго листа третьей рабочей книги Excel, следует записать такой оператор VBA:

```
Application.Workbooks(3).Worksheets(2).Range("A1")=1
```

Здесь используется не только простой объект "Диапазон" (`Range`), объекты, которые являются коллекциями (`Collection`) — `Workbooks` (семейство "Рабочие книги") и `Worksheets` (семейство "Рабочие листы").

В дальнейшем, при детальном изучении VBA, мы убедимся, что он не является объектно-ориентированным языком программирования в строгом понимании этого слова. Однако объектный подход играет в VBA основную роль. В последующих главах книги даются более корректные понятия для основных объектов VBA и особенности их использования.

Создание функции пользователя в VBA

Если вы уже использовали Microsoft Excel для каких-либо расчетов, то по достоинству могли оценить тот набор встроенных функций, которые предоставляет библиотека данного приложения. С другой стороны, возможно, какие-то расчеты вы хотели бы ускорить, однако нужной функции найти не удалось. Сейчас мы непосредственно займемся созданием собственных функций и убедимся, что ничего сложного в этом нет.

Итак, на нескольких примерах, которые приводятся далее, мы рассмотрим, как строятся функции пользователя. Начнем с простейшего примера — функции, вычисляющей значение по одной формуле. Затем обсудим более сложный пример, а именно расчет стоимости партии книг, когда значение функции зависит от условия.

Где пишется код функции пользователя?

Для того чтобы написать пользовательскую функцию, необходимо перейти в редактор VBA. Для начала убедитесь в том, чтобы в Microsoft Office Excel 2010 на ленте отображалась вкладка **Разработчик (**рис. 1.1).

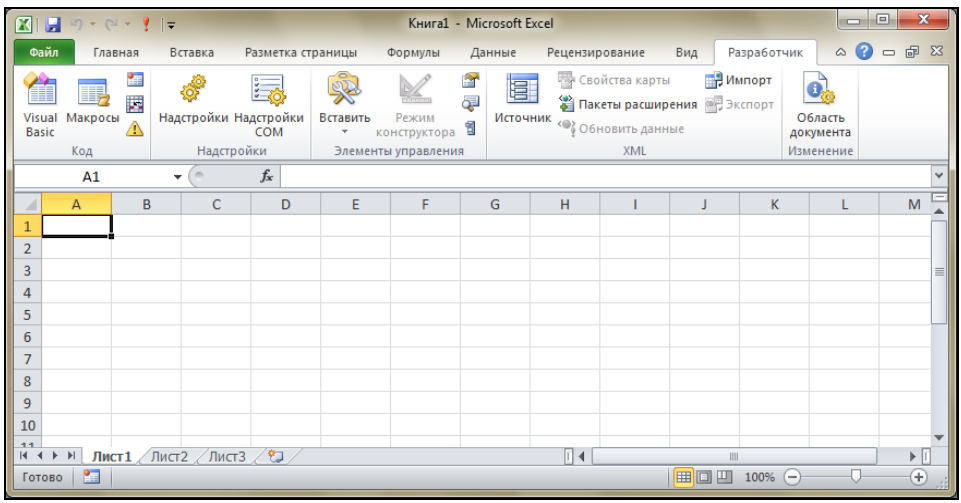


Рис. 1.1. Вкладка **Разработчик** на ленте

Если же она отсутствует, выполните следующие действия:

1. Перейдите на вкладку **Файл** ленты и нажмите кнопку **Параметры**.
2. В открывшемся окне **Параметры Excel** выберите слева категорию **Настройка ленты**, а справа в группе **Настройка ленты** выберите из раскрывающегося списка **Основные вкладки**.
3. Установите флажок **Разработчик** (рис. 1.2) и нажмите кнопку **OK**.

Итак, перейдите на вкладке **Разработчик** к группе **Код** и нажмите кнопку **Visual Basic**: у вас открылась интегрированная среда разработки приложений IDE редактора Visual Basic (рис. 1.3).

> ***Примечание***
>
> Для быстрого запуска редактора VBA достаточно всего лишь нажать комбинацию клавиш <Alt>+<F11>.

Среда разработки имеет стандартный интерфейс, характерный для Windows-приложений: строка заголовка, линейка меню, панель инструментов (в данном случае **Standard**), а также два окна: **Project - VBAProject** и **Properties**.

Отметим, что в окне **Project - VBAProject** перечисляются все модули и формы, входящие в создаваемый проект. Модуль представляет собой окно **Module**, в котором основную часть занимает рабочая область — лист (не путать с рабочим листом), в котором набирается код. Для открытия модуля в окне **Project - VBAProject** достаточно выполнить двойной щелчок на соответствующем значке. Для активного модуля его значок выделяется в окне **Project - VBAProject** серым цветом.

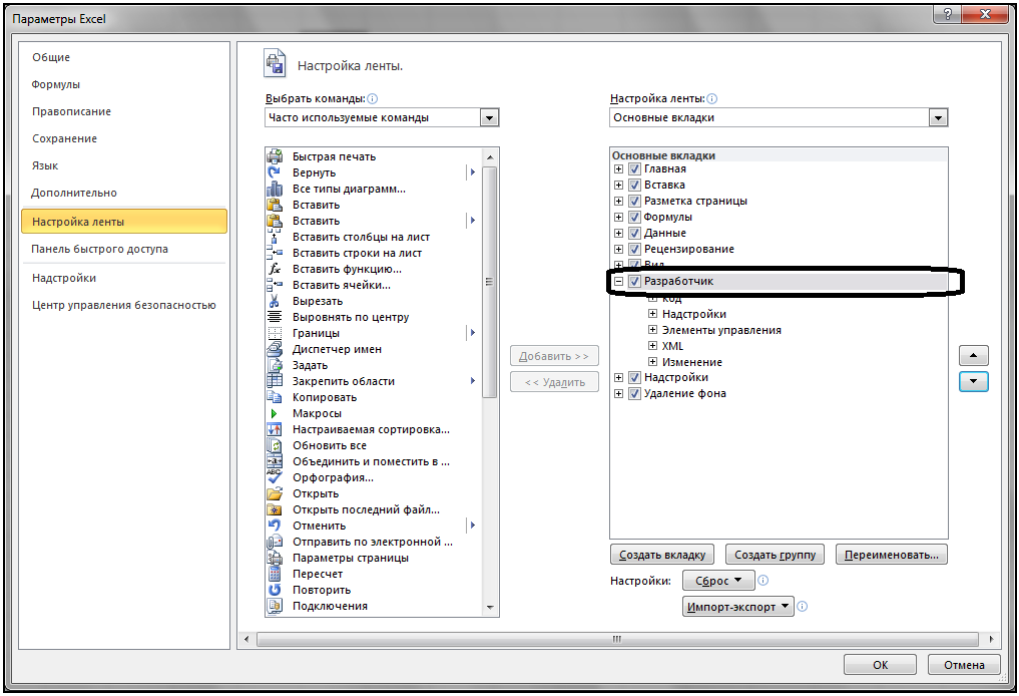


Рис. 1.2. Окно **Параметры Excel**

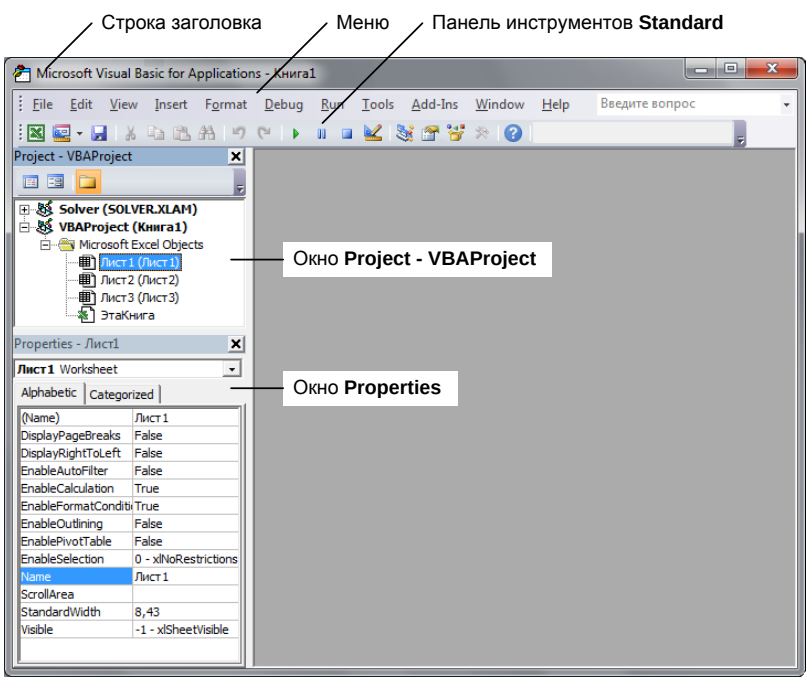


Рис. 1.3. Редактор Visual Basic

В VBA у каждого рабочего листа есть собственный модуль, и у рабочей книги также имеется свой. Более того, если в проекте создаются пользовательские формы, то каждая из них имеет по модулю. Вы можете добавлять в разрабатываемый проект модули классов для описания создаваемых пользовательских классов. Однако для создания пользовательской функции нам потребуется стандартный модуль, добавление которого в проект осуществляется командой **Insert | Module**.

Структура кода функции пользователя

В общем случае, функция пользователя имеет следующий вид:

```
Function ИмяФункции(СписокПараметров)
   Инструкции
End Function
```

- *СписокПараметров* — это список параметров, от которых зависит функция. Разделителем в списке параметров является запятая.
- *Инструкции* — это последовательность инструкций, выполняемых при нахождении значения функции. В совокупности они образуют так называемое *тело функции*. Важной особенностью функции пользователя является то, что носителем возвращаемого ею значения является *ИмяФункции*. Поэтому в теле функции должен присутствовать, по крайней мере, один оператор, который присваивает имени функции значение какого-либо выражения.

> ***Примечание***
>
> Допустим досрочный выход из функции по инструкции `Exit Function`. В теле функции может располагаться несколько инструкций `Exit Function`.

Ваша первая функция пользователя

Теперь можно непосредственно перейти к написанию функции пользователя. Давайте для начала напишем код для вычисления простой функции, например,

$$F\ x = x^3 + x^2.$$

Для реализации данной задачи вам необходимо выполнить следующие действия.

1. В окне редактора VBA добавьте лист стандартного модуля (если он вами еще не создан), выполнив команду **Insert | Module**.
2. В окне созданного модуля (рис. 1.4) наберите код из листинга 1.1 (см. также файл *1-Функции пользователя.xlsm* на компакт-диске).

Листинг 1.1. Пользовательская функция

```
Function F(x As Double) As Double
   F = x ^ 3 + x ^ 2
End Function
```

Следует отметить, что в VBA имеется универсальный тип данных `Variant`, который подразумевается по умолчанию, если явно не был объявлен тип переменной или функции. Поэтому ту же самую функцию можно было бы закодировать следующим образом (листинг 1.2).

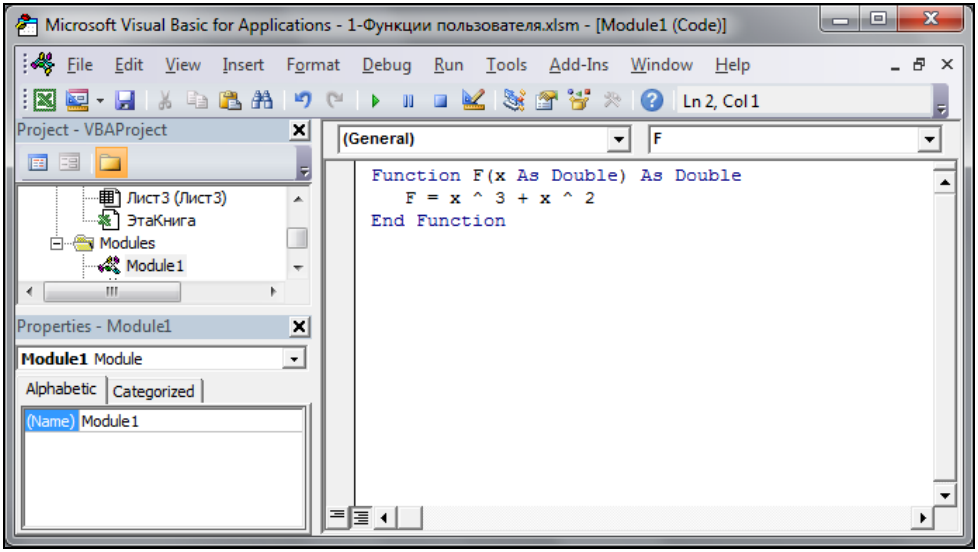


Рис. 1.4. Код пользовательской функции в модуле

Листинг 1.2. Пользовательская функция с использованием типа `Variant`

```
Function F(x)
   F = x ^ 3 + x ^ 2
End Function
```

> ***Примечание***
>
> Еще раз обратите внимание на то, что код пользовательской функции вводится в стандартном модуле, который добавляется в проект командой **Insert | Module**. В проекте много модулей, случайно не перепутайте. Активный модуль в окне **Project - VBAProject** выделяется серым цветом.

Итак, пользовательская функция нами создана. По умолчанию она попадает в раздел **Определенные пользователем** списка **Категория** окна **Мастер функций**. Найдем, например, значение этой функции при $x = 4{,}7$. Для этого:

1. Перейдите к окну рабочей книги Microsoft Office Excel 2010.
2. Введите число `4,7` в ячейку **A1** рабочего листа (выбрав, например, **Лист1**).
3. Перейдите к ячейке **B1**, в которой найдем значение функции.
4. Перейдите на вкладку **Формулы** ленты и в группе **Библиотека функций** щелкните по кнопке **Вставить функцию**.

5. В первом окне мастера функций укажите из списка категорию **Определенные пользователем** и выберите функцию `F`. Нажмите кнопку **OK**.
6. Во втором окне мастера функций в поле **X** введите ссылку на ячейку **A1** (или щелкните по соответствующей ячейке рабочего листа мышью) и нажмите кнопку **OK** (рис. 1.5). Значение функции найдено.

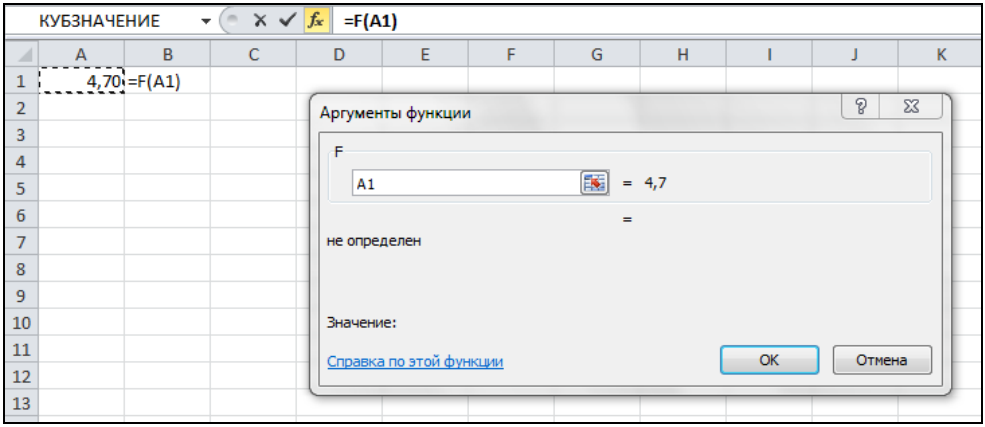


Рис. 1.5. Вычисление значения пользовательской функции при помощи мастера функций

Вычисление стоимости партии продаваемых книг при помощи пользовательской функции

Рассмотрим более сложный пример построения функции пользователя. Представим себе, что вы являетесь менеджером по оптовой продаже книг в издательстве. Для привлечения покупателей в вашем издательстве введена прогрессивная шкала цен. Так, если продается от 100 до 200 экземпляров книги, то скидка от ее отпускной цены составляет 7%, если продается от 201 до 300 экземпляров, то скидка составляет 10%, а если свыше 300 экземпляров, то — 15%. Кроме того, для постоянных клиентов предусмотрена дополнительная скидка в размере 5%. Создадим функцию пользователя с именем `Стоимость` для расчета стоимости партии книг. Параметры этой функции назовем `ЦенаОднойКниги`, `Количество` и `Скидка`. Для параметра `Скидка` предусмотрим только два допустимых значения: 1 — для постоянных клиентов и 0 — для всех остальных. Определим пользовательскую функцию `Стоимость` следующим кодом (листинг 1.3, а также файл *1-Функции пользователя.xlsm* на компакт-диске).

Листинг 1.3. Расчет стоимости партии книг

```
Function Стоимость(ЦенаОднойКниги, Количество, Скидка)
   If Количество < 100 Then
      СтоимостьБезСкидки = ЦенаОднойКниги * Количество
   ElseIf Количество <= 200 Then
      СтоимостьБезСкидки = ЦенаОднойКниги * Количество * 0.93
   ElseIf Количество <= 300 Then
      СтоимостьБезСкидки = ЦенаОднойКниги * Количество * 0.9
   Else
      СтоимостьБезСкидки = ЦенаОднойКниги * Количество * 0.85
   End If

   If Скидка = 0 Then
```

```
      Стоимость = СтоимостьБезСкидки
   Else
      Стоимость = СтоимостьБезСкидки * 0.95
   End If
End Function
```

Итак, функция пользователя `Стоимость` создана. Благодаря тому, что в VBA допустимо применение русскоязычных имен, текст написанной программы очевиден для понимания. Кроме того, это обеспечивает и простоту использования диалогового окна мастера функций для данной функции (рис. 1.6).

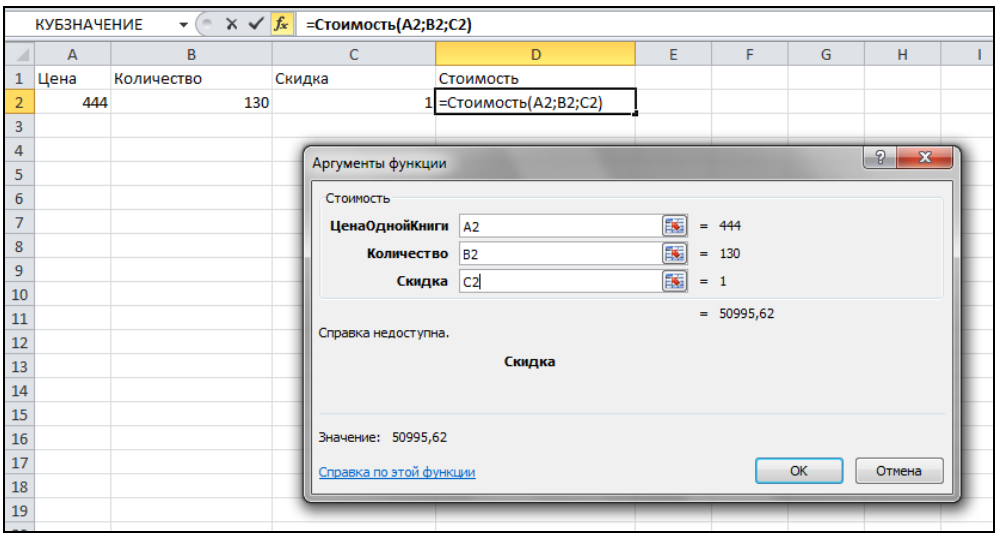


Рис. 1.6. Расчет стоимости партии книг

Названия всех параметров функции `Стоимость` в окне мастера функций выводятся также на русском языке, что позволяет ее применять любому пользователю, даже не владеющему VBA. Следует заметить, что для удобства использования созданной функции `Стоимость` рекомендуется предварительно на рабочем листе задать значения для входных параметров функции, т. е. для `ЦенаОднойКниги`, `Количество` и `Скидка`. Однако это не является обязательным условием: задать необходимые значения вы можете также и сразу в окне мастера функций.

Использование ссылок на диапазон в качестве параметров пользовательских функций

Предыдущий пример о вычислении стоимости партии книг выглядит уже достаточно убедительно. VBA действительно может облегчить жизнь пользователя. Возникает только один вопрос по двум приведенным примерам. В созданных в них пользовательских функциях в качестве значений параметров были только ссылки на ячейки. Можно ли построить пользовательскую функцию, у которой в качестве

значений параметров могут быть ссылки на диапазоны ячеек? Код из листинга 1.4 демонстрирует возможность использования ссылок на диапазон значений и решает задачу суммирования значений для указанного диапазона ячеек (см. также файл *1-Функции пользователя.xlsm* на компакт-диске).

Листинг 1.4. Нахождение суммы

```
Function MySum(ByVal rng As Range) As Double
   Dim c As Range
   Dim s As Double
   s = 0
   For Each c In rng.Cells
      s = s + c.Value
   Next
   MySum = s
End Function
```

Об элементах автоматизации Microsoft Office Excel

Зачем нужны макросы?

А сейчас мы познакомимся с вами с основами автоматизации деятельности, которая не возможна без использования макросов. *Макрос* — это программа, состоящая из списка команд, которые должны быть выполнены приложением. Макрос служит для объединения нескольких различных действий в одну процедуру. Такой список команд состоит, в основном, из *макрооператоров*, тесно связанных с командами приложений из Microsoft Office. Большая часть макрооператоров соответствует командам меню или параметрам, которые задаются в диалоговых окнах.

Можно выделить три основные разновидности макросов:

- *командные* — наиболее распространенные макросы, которые обычно состоят из операторов, эквивалентных тем или иным командам меню или параметрам диалоговых окон. Основным предназначением таких макросов является выполнение действий, аналогичных командам меню, т. е. изменение окружения и основных объектов приложения. Например, изменение рабочего листа или рабочего пространства Microsoft Excel, сохранение или вывод на печать и т. п. Таким образом, в результате выполнения макроса вносятся изменения либо в обрабатываемый документ, либо в общую среду приложения;
- *пользовательские функции* — работают аналогично имеющимся встроенным функциям Microsoft Excel. Отличие этих функций от командных макросов состоит в том, что они используют значения передаваемых им аргументов, производят некоторые вычисления и возвращают результат в точку вызова, но не изменяют среду приложения;
- *макрофункции* — представляют собой сочетание командных макросов и пользовательских функций. Наряду с тем, что они, подобно пользовательским функциям,

могут использовать аргументы и возвращать результат, макрофункции, как и командные макросы, способны еще и изменять среду приложения. Чаще всего макрофункции вызываются из других макросов и активно используются для модульного программирования. Если необходимо в различных макросах выполнить ряд одинаковых действий, то эти действия обычно выделяются в отдельную макрофункцию (подпрограмму).

Как правило, макросы используются для быстрого получения чернового варианта кода. Следует помнить о последовательности действий, связанной с разработкой макросов:

1. *Логическая разработка процедуры*. Прежде всего, вам необходимо точно определить, что следует получить в результате выполнения макроса и какова логическая последовательность действий для получения данного результата.
2. *Подготовка документа*. Произведите предварительные действия, которые не нужно включать в процедуру (например, создание нового рабочего листа или перемещение в конкретную часть рабочего листа и т. д.).
3. *Запись макроса с помощью макрорекордера*. *Макрорекордер* представляет собой транслятор, создающий программу (макрос) на языке VBA, которая является результатом перевода на языке VBA действий пользователя с момента запуска макрорекордера до окончания записи макроса. Для записи макроса с помощью макрорекордера:
   - перейдите на вкладку **Разработчик** ленты и в группе **Код** щелкните по кнопке **Запись макроса**;
   - в открывшемся диалоговом окне **Запись макроса** установите параметры записываемой процедуры (ее имя, описание, сочетание клавиш для выполнения записанной процедуры, указание, для каких документов доступен макрос) и войдите в режим записи макроса — на экране на вкладке **Разработчик** ленты в группе **Код** кнопка **Запись макроса** изменится на кнопку **Остановить запись**; кроме того, кнопка **Пауза** также станет активной (если вы на время захотите приостановить запись макроса и выполнить другие действия с документом);
   - последовательно выполните все необходимые действия с документом и его содержимым, предусмотренные на первом этапе;
   - остановите запись (кнопка **Остановить запись** в группе **Код** на вкладке **Разработчик**).
4. *Просмотр и редактирование созданной процедуры:*
   - нажмите кнопку **Макросы** в группе **Код** на вкладке **Разработчик**, а затем в открывшемся диалоговом окне **Макрос** выберите в списке имя нужного макроса и нажмите кнопку **Изменить**, далее откроется главное окно редактора Microsoft Visual Basic и окно **Модуль** (Module) с текстом выбранного макроса;
   - внесите в текст макроса необходимые изменения и закройте окно редактора.
5. *Выполнение макроса.* Нажмите кнопку **Макросы** в группе **Код** на вкладке **Разработчик**, затем в открывшемся диалоговом окне **Макрос** в списке выберите имя макроса и нажмите кнопку **Выполнить**.

> ***Примечание***
>
> Вы можете назначить записанной процедуре кнопку и расположить ее на **Панели быстрого доступа** для упрощения вызова макроса.

Запись макроса и размещение его на панели быстрого доступа

Пусть нам необходимо создать макрос, который активизирует рабочий лист: так, если у нас активным является **Лист1**, и мы запишем макрос, который активизирует рабочий лист **Лист2**.

1. Запустите Microsoft Office Excel 2010 и убедитесь, что указатель ячейки находится на рабочем листе **Лист1**.
2. Для активизации макрорекордера перейдите на вкладку **Разработчик** ленты и в группе **Код** щелкните по кнопке **Запись макроса**.
3. В открывшемся окне **Запись макроса** установите необходимые параметры записываемой процедуры (рис. 1.7): присвойте, например, макросу имя `АктивизироватьЛист`, в поле **Описание** введите соответствующие пояснения — для чего создается вами данный макрос, а поле **Сохранить в** оставьте без изменения. Нажмите кнопку **ОК**.

   Примечание

   Поля **Имя макроса** и **Описание** используются для задания имени макроса и его описания. Помните, что в имени макроса не должно быть пробелов, а описание важно для многократно используемых макросов, т. к. через некоторое время вам будет трудно вспомнить, для чего создавался тот или иной макрос. По умолчанию макросам присваиваются имена `Макрос1`, `Макрос2` и т. д. С целью облегчения узнаваемости макроса лучше не использовать стандартное имя, а присвоить ему какое-нибудь уникальное имя, поясняющее, для чего он предназначен.

   Область **Сочетание клавиш** позволяет назначить макросу комбинацию клавиш, т. е. указать символ, который в комбинации с клавишей <Ctrl> позволит его выполнить. Назначать комбинацию клавиш макросу совсем не обязательно. Пожалуй, это стоит делать только для постоянно используемых макросов — для быстрого доступа к ним. Без помощи комбинации клавиш макрос в Microsoft Office Excel 2010 можно всегда вызвать следующим образом: следует перейти на вкладку **Разработчик** ленты и в группе **Код** щелкнуть по кнопке **Макросы**.

   Раскрывающийся список **Сохранить в** предназначен для выбора книги, в которой будет сохранен макрос. Если выбрать **Личная книга макросов**, то макрос будет сохранен в специальной скрытой книге, в которой хранятся макросы. Эта книга всегда открыта, хотя и скрыта, и записанные в ней макросы доступны для других рабочих книг. Для отображения личной книги макросов перейдите на вкладку **Вид** и в группе **Окно** щелкните по кнопке **Отобразить**. Если в раскрывающемся списке выбрать **Эта книга** (т. е. выбор, который по умолчанию предлагает компьютер), то макрос сохранится на новом листе модуля в активной рабочей книге, а если выбрать **Новая книга**, то он сохранится в новой рабочей книге.

4. В режиме записи макроса (рис. 1.8) перейдите на **Лист2** (указатель устанавливается в ячейку **А1**).
5. Нажмите кнопку **Остановить запись**, расположенную в группе **Код** на вкладке **Разработчик** для остановки записи макроса.
6. Сохраните вашу рабочую книгу в формате, поддерживающем макросы: перейдите на вкладку **Файл** ленты и выберите команду **Сохранить как**. В открывшемся окне **Сохранение документа** (рис. 1.9) выберите место для рабочей книги, указав его в верхней части окна, введите имя рабочей книги — в поле **Имя файла**, а также выберите в поле со списком **Тип файла** формат — **Книга Excel с поддержкой макросов (*.xlsm)**. Нажмите кнопку **Сохранить**.

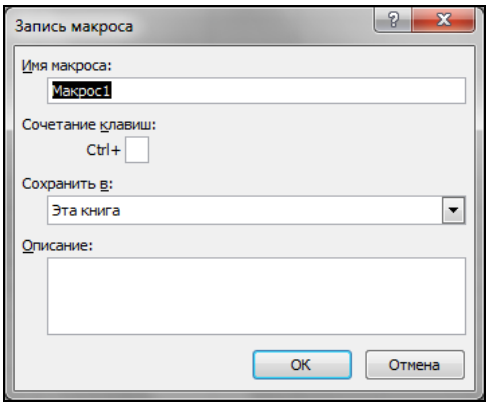


Рис. 1.7. Окно **Запись макроса**

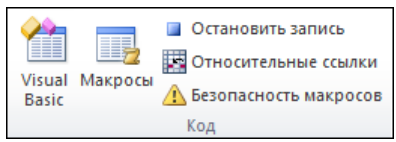


Рис. 1.8. Группа **Код** на вкладке **Разработчик** в режиме записи макроса

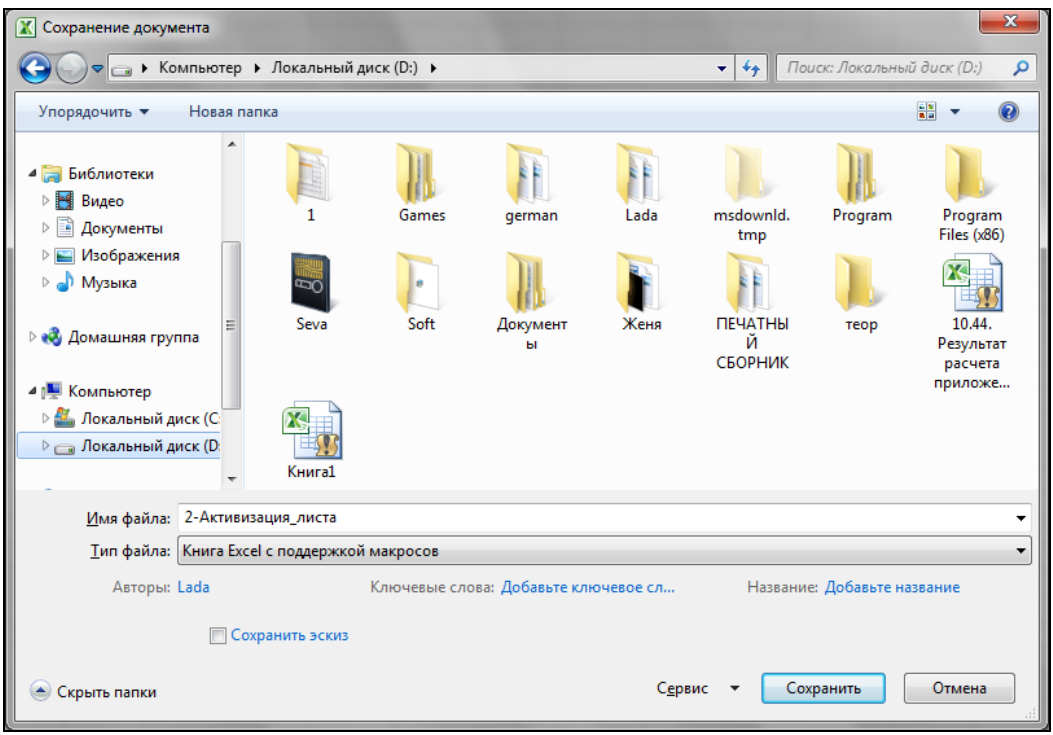


Рис. 1.9. Сохранение книги Excel с поддержкой макросов

7. Перейдите на вкладку **Разработчик** и в группе **Код** щелкните по кнопке **Макросы**: в открывшемся окне **Макрос** выделите в списке имя созданного макроса (рис. 1.10) и нажмите кнопку **Изменить**. На экране отобразится окно редактора VBA с активизированным стандартным модулем, в котором будет код (листинг 1.5) только что записанного макроса (см. также файл *2-Активизация_листа.xlsm* на компакт-диске).

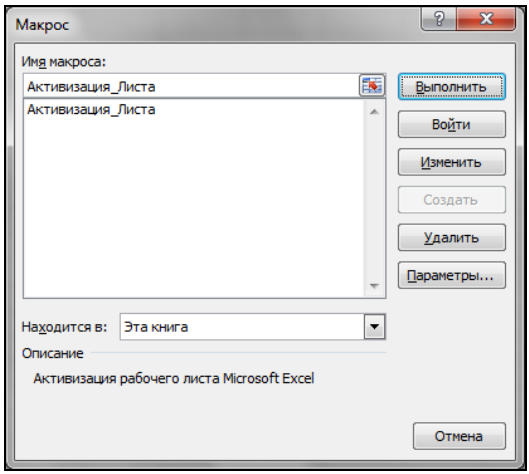


Рис. 1.10. Окно **Макрос** для открытой рабочей книги

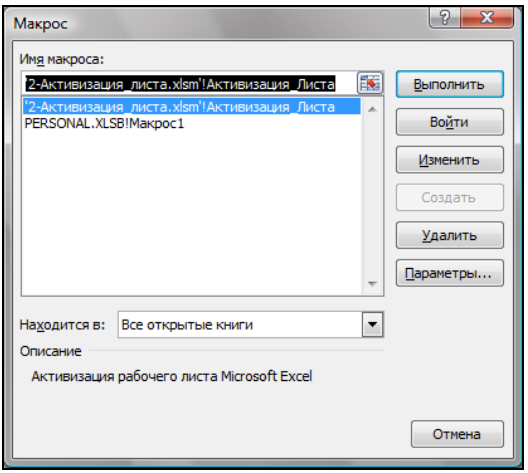


Рис. 1.11. Окно **Макрос**

Листинг 1.5. Макрос, активизирующий рабочий лист *Лист2*. Стандартный модуль

```
Sub Активизация_Листа ()
'
' Активизация_Листа Макрос
' Активизация рабочего листа Microsoft Excel
'
    Sheets("Лист2").Select
End Sub
```

8. Теперь, не закрывая данной рабочей книги, создайте еще одну рабочую книгу, для чего перейдите на вкладку ленты **Файл**, выберите команду **Создать** и в группе **Доступные шаблоны** укажите **Новая книга**.
9. Перейдите на вкладку **Разработчик** ленты и в группе **Код** щелкните по кнопке **Макросы**.
10. В открывшемся окне **Макрос** (рис. 1.11) укажите имя созданного макроса и нажмите кнопку **Выполнить**. Убедитесь, что в вашей рабочей книге данный макрос активизировал **Лист2**.

Примечание

На вкладке **Разработчик** в группе **Код** находится кнопка **Безопасность макросов**, которая открывает окно **Центр управления безопасностью** в категории **Параметры макросов** (рис. 1.12). Вы всегда можете выбрать требуемый параметр, чтобы предотвратить нежелательное выполнение вредоносного кода, который может содержаться в макросах, полученных из неизвестных источников.

Выбрав слева в окне **Центр управления безопасностью** категорию **Надежные расположения** и нажав справа кнопку **Добавить новое расположение**, можно указать место, откуда ваши файлы, содержащие код на VBA, будут открываться без блокирования соответствующих действий (рис. 1.13).

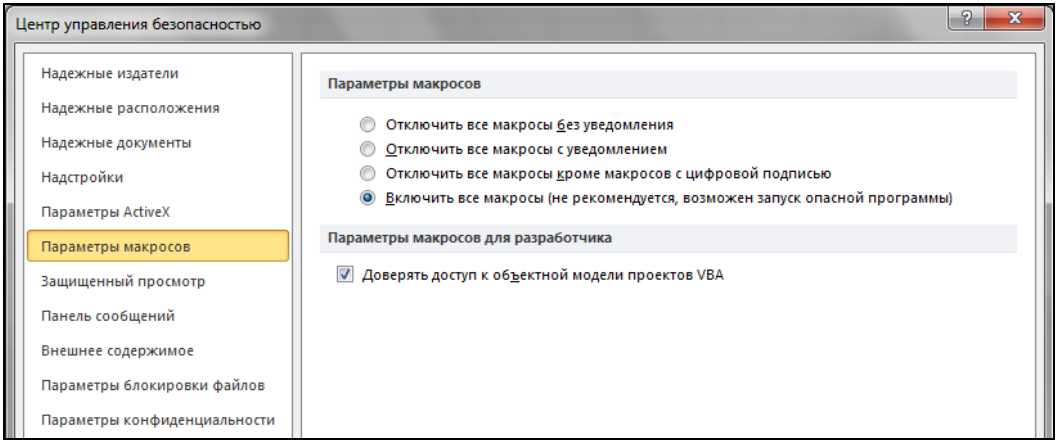


Рис. 1.12. Окно **Центр управления безопасностью**, категория **Параметры макросов**

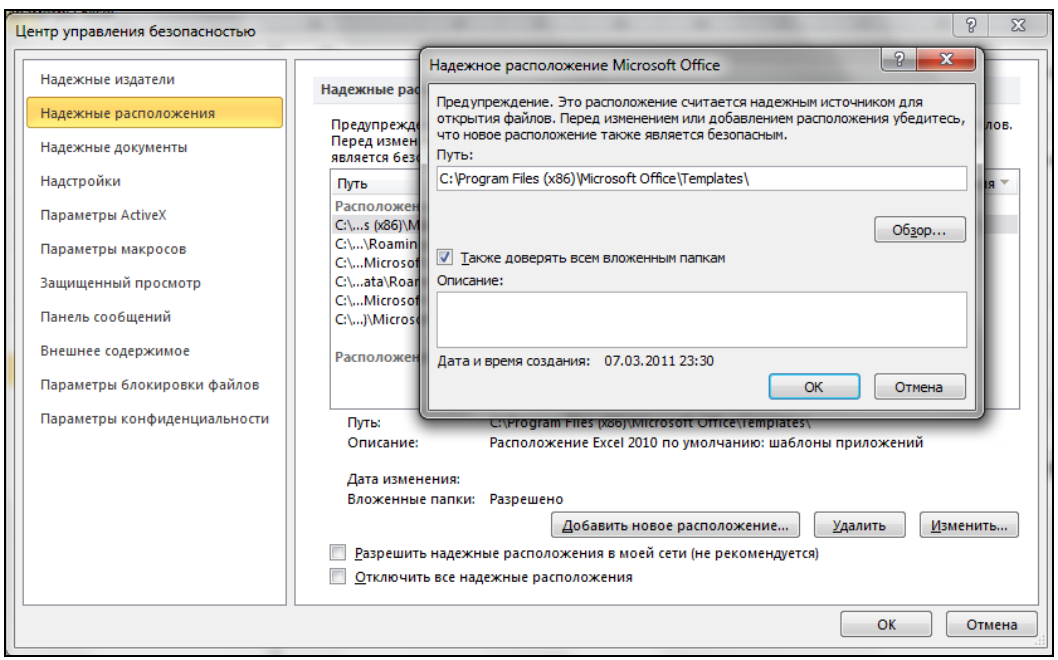


Рис. 1.13. Окно **Центр управления безопасностью**, категория **Надежные расположения**

Ну, а теперь назначим наш созданный макрос кнопке, которую разместим на панели быстрого доступа.

1. Щелкните правой кнопкой мыши по панели быстрого доступа и выберите команду **Другие команды**.
2. В открывшемся окне **Параметры Excel** в категории **Панель быстрого доступа** выберите в поле со списком **Выбрать команды из** объект **Макросы**.

3. Выберите в левом столбце созданный вами макрос `Активизация_Листа` (рис. 1.14) и с помощью кнопки **Добавить >>** перенесите его в правый столбец. Обратите внимание на то, что внизу правого столбца стала доступной кнопка **Изменить**: она служит для назначения кнопки соответствующему макросу. Нажмите кнопку **Изменить**.
4. В открывшемся окне **Изменение кнопки** (рис. 1.15) укажите мышью символ для кнопки и при необходимости в поле **Отображаемое имя** измените имя для макроса, которое будет всплывающей подсказкой на панели быстрого доступа. Нажмите кнопку **OK**.

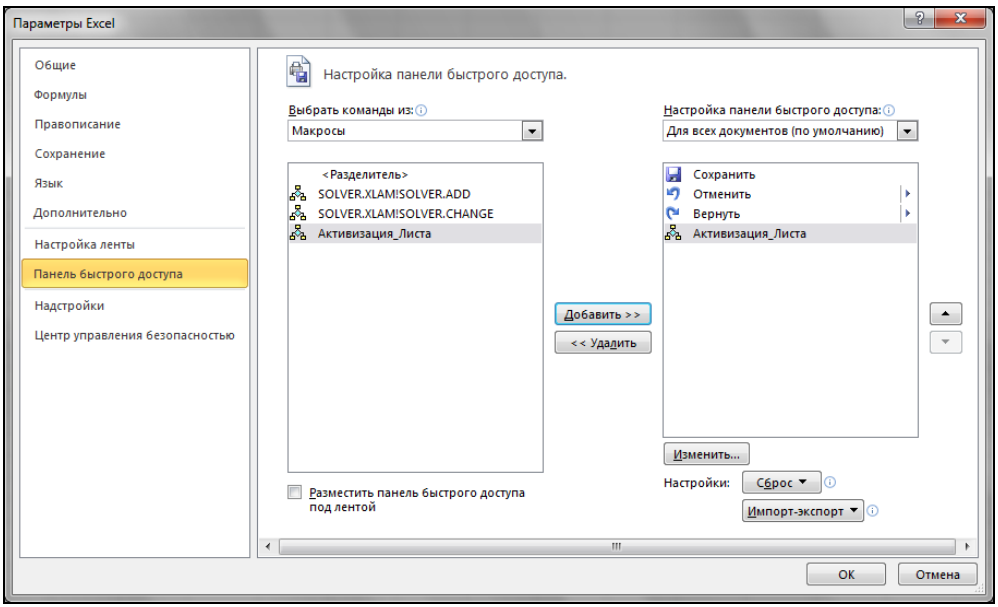


Рис. 1.14. Выбор объекта **Макросы** в окне **Параметры Excel** в категории **Панель быстрого доступа**

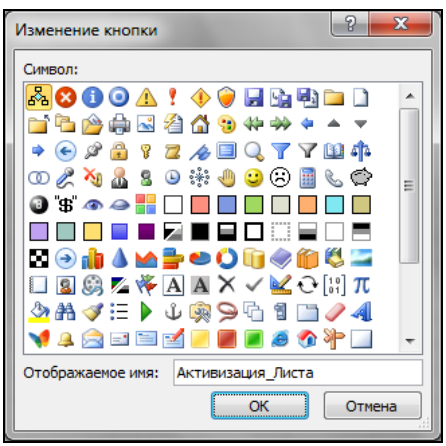


Рис. 1.15. Окно **Изменение кнопки**

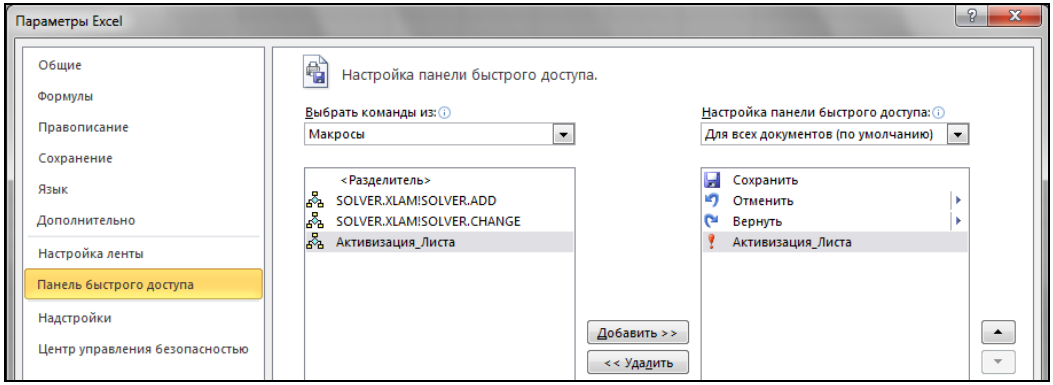


Рис. 1.16. Кнопка для макроса в окне **Параметры Excel** в категории **Панель быстрого доступа**

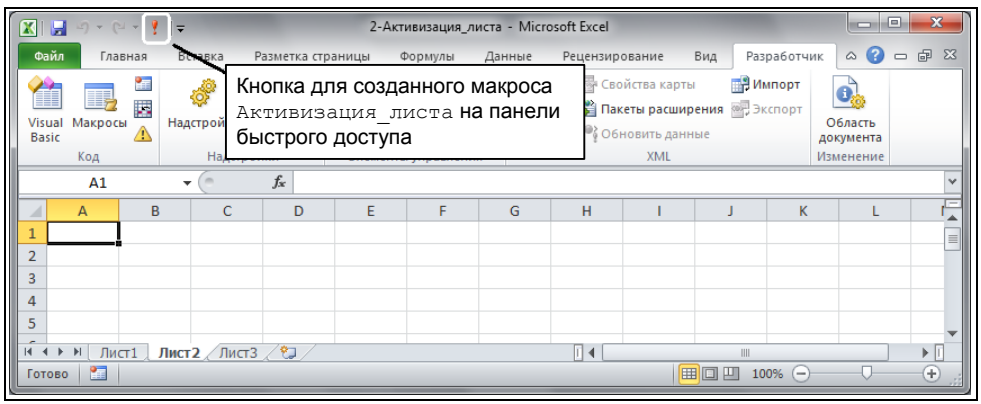


Рис. 1.17. Кнопка созданного макроса на панели быстрого доступа

5. В окне **Параметры Excel** в категории **Панель быстрого доступа** кнопка для макроса отображается в правом столбце (рис. 1.16). Нажмите кнопку **OK**.
6. Убедитесь, что кнопка с записанным макросом появилась на панели быстрого доступа (рис. 1.17), и ее нажатие приводит к выполнению записанных вами действий.

Структура кода процедуры

Итак, макрос нами записан, причем макрорекордер, который мы использовали, создал в модуле процедуру. В общем случае процедура имеет следующий вид:

```
Sub ИмяПроцедуры(СписокПараметров)
    Инструкции
End Sub
```

- *СписокПараметров* — это список параметров, от которых зависит функция. Разделителем в списке параметров является запятая. Список параметров может и отсутствовать — быть пустым (см., например, листинг 1.5).
- *Инструкции* — это последовательность инструкций, выполняемых при работе процедуры. В совокупности они образуют так называемое *тело процедуры*.

Примечание

Допустим досрочный выход из процедуры по инструкции `Exit Sub`. В теле процедуры может располагаться несколько инструкций `Exit Sub`.

Процедура обработки события

Кроме обычных процедур, в VBA имеются процедуры, которые обрабатывают события, связанные с тем или иным объектом. В общем случае такие процедуры записываются в виде:

```
[Private] Sub ИмяОбъекта_ИмяСобытия(СписокПараметров)
    Инструкции
End Sub
```

Ключевое слово `Private` является модификатором доступа и обозначает, что процедура видна другим процедурам только в модуле, в котором она располагается. Например, процедура обработки события активизации рабочего листа имеет вид:

```
Private Sub Worksheet_Activate()
    Инструкции
End Sub
```

Автоматизация работы рабочего листа при помощи элементов управления

Microsoft Office Excel 2010 обладает также и полноценным набором различных элементов управления: кнопка, поля со списком, переключатель, флажок и т. д., которые при необходимости размещаются на рабочем листе (рис. 1.18). Для того чтобы увидеть список доступных элементов управления, перейдите на вкладку **Разработчик** и в группе **Элементы управления** щелкните по кнопке со списком **Вставить**.

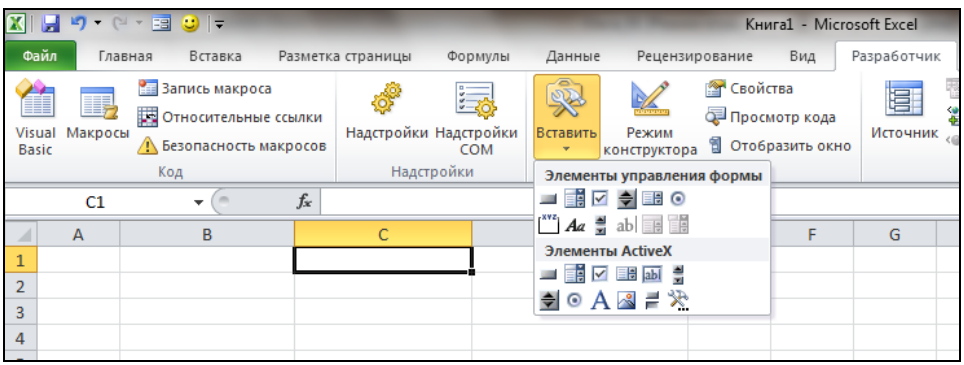


Рис. 1.18. Коллекция элементов управления в Microsoft Office Excel 2010

Следует отметить, что элементы управления, расположенные в группе **Элементы управления формы**, предназначены, прежде всего, для обеспечения совместимости с файлами старых версий Excel (до Excel 97), в которых используются эти элементы управления. Они обладают значительно меньшими возможностями по сравнению с элементами управления, расположенными в группе **Элементы ActiveX**.

Некоторые из этих элементов вообще не могут быть использованы в документах Excel последних версий — это **Edit Box** (поле ввода), **Combination List-Edit** (поле ввода со списком) и **Combination Drop-Down Edit** (поле ввода с раскрывающимся списком). Однако данные элементы управления имеют и ряд преимуществ, которые отсутствуют у элементов управления, расположенных на панели **ActiveX** (ActiveX Controls) — в частности, данные элементы управления могут быть помещены на листы диаграмм.

Элементы управления группы **Элементы ActiveX** являются независимыми компонентами различных приложений и, в том числе, могут использоваться и в Microsoft Excel. Данная группа включает также и элементы управления, которые аналогичны многим элементам управления из группы **Элементы управления формы**, но которые являются недоступными в Microsoft Office Excel 2010.

Кроме стандартных элементов управления, можно использовать дополнительные элементы управления. Вместе с Excel поставляется некоторое количество таких элементов, например, элементы управления мультимедиа, с помощью которых можно воспроизводить звук или видео непосредственно с рабочего листа. Кроме того, существует возможность подключать элементы управления, которые используются в других программах, или созданные отдельно элементы управления.

В модуле рабочего листа можно создать процедуры, которые будут обрабатывать те или иные события, происходящие с элементами управления. Например, нажатие кнопки, выбор элемента из списка, выбор переключателя, установка флажка и т. д. будут автоматически приводить к тем или иным вычислением, построению диаграмм или смене их типа и т. п. В дальнейшем мы подробно познакомимся с элементами управления и возможностью автоматизации рабочих листов *(см. главу 6)*. Здесь же мы приведем лишь небольшой пример использования элемента управления **Кнопка** (CommandButton)  , а также события, связанного с ней — `Click`, которое генерируется при нажатии кнопки.

Использование элемента управления *Кнопка* на рабочем листе

Продемонстрируем использование на рабочем листе элемента управления **Кнопка** (CommandButton) из группы **Элементы ActiveX**. Пусть при нажатии на элемент управления **Кнопка** (CommandButton), который мы расположим на рабочем листе **Лист1**, будет выполняться активизация рабочего листа **Лист2**.

1. Запустите Microsoft Office Excel 2010 и убедитесь, что указатель ячейки находится на рабочем листе **Лист1**.
2. Перейдите на вкладку **Разработчик** и в группе **Элементы управления** щелкните по кнопке со списком **Вставить**.
3. Нажмите элемент управления **Кнопка** (CommandButton)  из группы **Элементы ActiveX** и перейдите непосредственно на рабочий лист, указатель при этом приобретает вид тонкого крестика.
4. Выберите место на рабочем листе, нажмите левую кнопку мыши и, не отпуская ее, нарисуйте кнопку необходимого размера, после чего отпустите кнопку мыши. Обратите внимание на то, что после появления

элемента управления **Кнопка** (CommandButton) на рабочем листе кнопка **Режим конструктора** стала активной (включенной). На поверхности первого созданного вами элемента управления **Кнопка** (CommandButton) автоматически будет отображена надпись **CommandButton1** (рис. 1.19). Если же вы теперь создадите второй элемент управления **Кнопка** (CommandButton), то на его поверхности отобразится надпись **CommandButton2** и т. д.

Примечание

Как и любой графический объект, кнопку можно рисовать при нажатой клавише <Shift>, что обеспечит ей квадратную форму, либо при нажатой клавише <Alt> — для привязки кнопки к сетке рабочего листа. У созданной кнопки при помощи маркеров изменения размеров можно задать размеры, а при помощи маркера перемещения — установить ее местоположение.

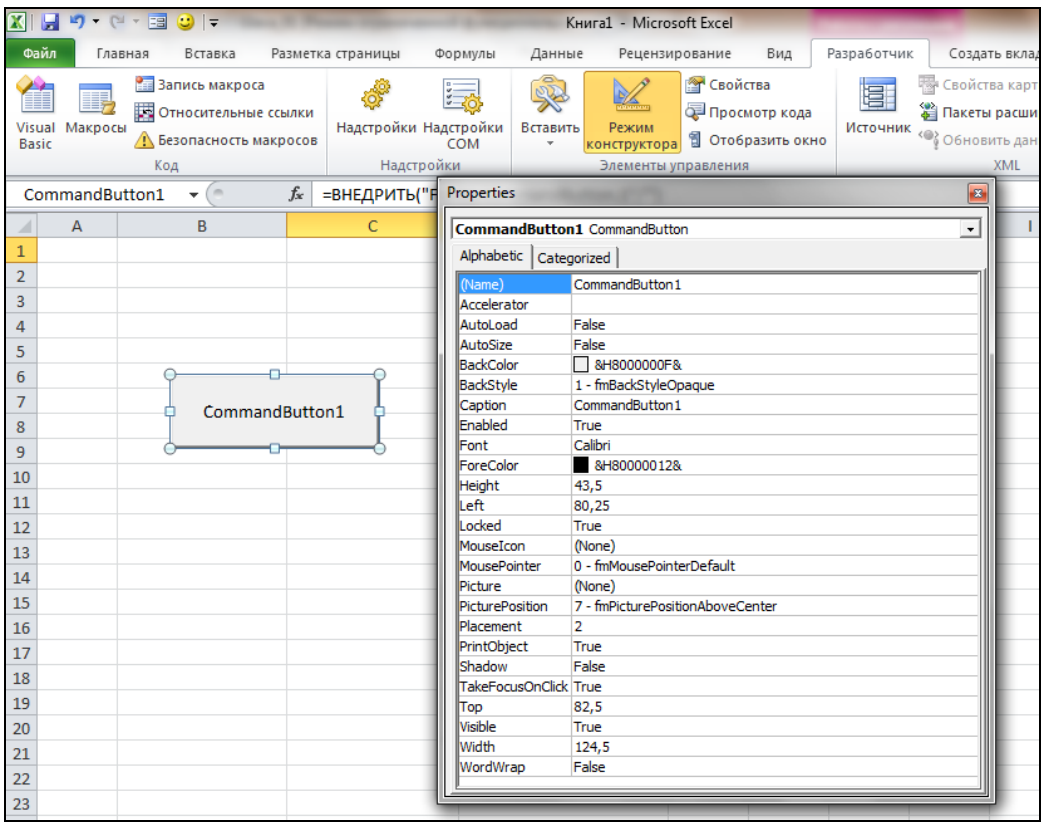


Рис. 1.19. Элемент управления **Кнопка** и окно **Properties**

5. Щелкните по созданной кнопке правой кнопкой мыши и из появившегося контекстного меню выберите команду **Свойства** (Properties) для открытия окна **Properties** (см. рис. 1.19). Элемент управления **Кнопка** является объектом, т. е. он, как и любой объект, обладает свойствами, методами и событиями. Надпись, отображаемая на поверхности элемента управления **Кнопка**, задается значением

свойства `Caption`. Кроме того, элемент управления **Кнопка** имеет свойство `Name`, которое в коде идентифицирует его как объект. В данном случае, оно тоже равно `CommandButton1`. Измените в открытом окне **Properties** значение свойства `Caption`, т. е. надпись, отображаемую на поверхности кнопки, с **CommandButton1** на **Лист2** (с тем, чтобы подчеркнуть, что эта кнопка будет активизировать рабочий лист **Лист2**). При желании поэкспериментируйте со следующими свойствами элемента управления **Кнопка**: `BackColor`, `Font`, `ForeColor`, `Shadow`. Закройте окно **Properties**.

> ***Примечание***
>
> Для открытия окна **Properties** можно также поступить и следующим образом: выделите мышью требуемый элемент управления, а затем на вкладке **Разработчик** ленты в группе **Элементы управления** щелкните по кнопке **Свойства**.

6. Создадим теперь код процедуры, обрабатывающей событие "нажатие кнопки". В результате обработки этого события должен активизироваться рабочий лист **Лист2**. Итак:
   - дважды щелкните на созданной кнопке (напоминаем, что кнопка **Режим конструктора** в группе **Элементы управления** на вкладке **Разработчик** нажата): откроется редактор VBA с активизированным модулем рабочего листа (в данном случае, **Лист1**), в который автоматически добавились первая и последняя инструкции процедуры обработки события "нажатие кнопки" (`Click`) (листинг 1.6, *а*).

Листинг 1.6, *а*. Первая и последняя инструкции процедуры обработки события "нажатие кнопки". Модуль рабочего листа *Лист1*

```
Private Sub CommandButton1_Click()

End Sub
```

   - откройте файл из ОС Windows, в котором вы создавали макрос `Активизация_Листа`; проверьте, чтобы в окне **Project - VBAProject** отобразился необходимый модуль;
   - скопируйте через буфер обмена (рис. 1.20) инструкцию из макроса:
     `Sheets("Лист2").Select`

     в процедуру обработки события "нажатие кнопки" (листинг 1.6, *б*, файл *3-Кнопка.xlsm*). Конечно, вы могли бы набрать данную инструкцию и вручную. Однако это довольно медленно и чревато появлением случайных опечаток, которые неизбежно возникают при наборе кода.

Листинг 1.6, *б*. Процедура обработки события "нажатие кнопки", при котором будет активизирован рабочий лист *Лист2*. Модуль рабочего листа *Лист1*

```
Private Sub CommandButton1_Click()
   Sheets("Лист2").Select
End Sub
```

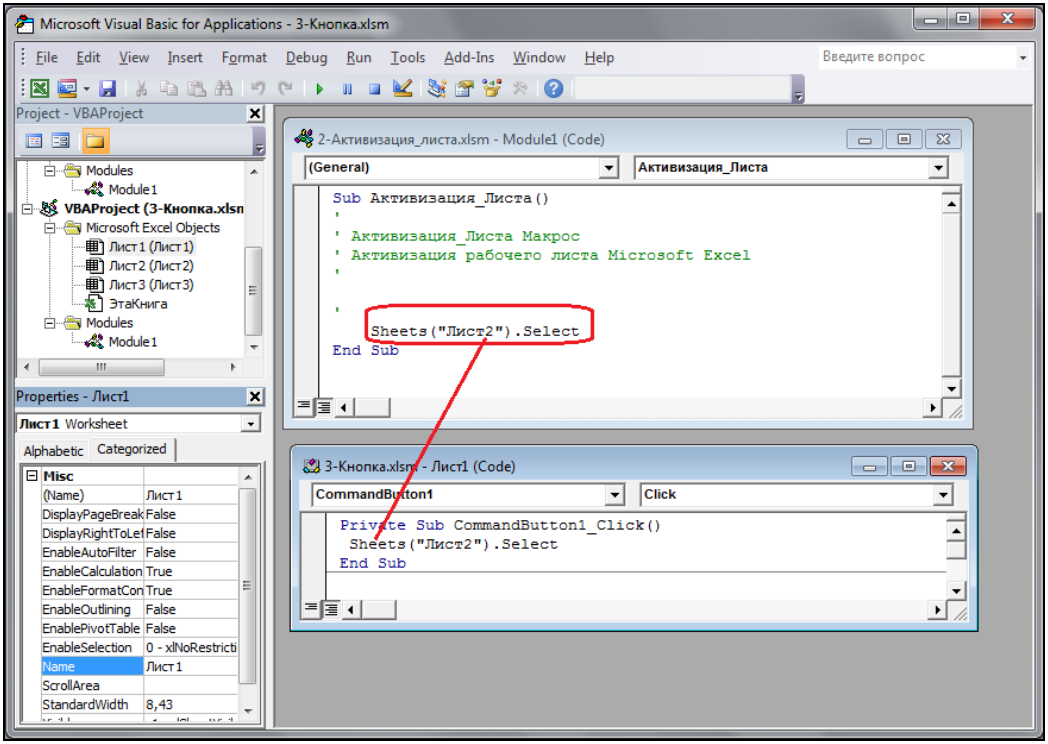


Рис. 1.20. Окно редактора Visual Basic и открытые окна модулей

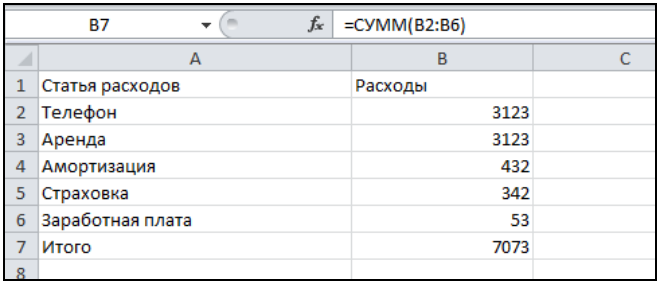
7. Вернитесь на рабочий лист **Лист1**. Созданная кнопка будет обрабатывать событие (нажатие на нее) только после выхода из режима конструктора. Поэтому отключите режим конструктора нажатием кнопки **Режим конструктора**, находящейся в группе **Элементы управления** на вкладке **Разработчик** ленты.

8. Протестируйте созданную кнопку: нажмите ее, и если вы все правильно сделали, то это приведет к активизации рабочего листа **Лист2**.

В качестве упражнения и закрепления изложенного материала рекомендуем вам создать макрос, который добавляет новый рабочий лист в книгу Excel и устанавливает указатель ячейки на рабочем листе **Лист1**, а затем, соответственно, связать созданную процедуру с другой кнопкой, также расположенной на рабочем листе **Лист1**.

Построение шаблона таблицы

Рассмотрим еще один пример, автоматизирующий деятельность некоторой коммерческой фирмы. Пусть, например, вы являетесь менеджером некоторого ООО "Бескрайние просторы", и вам необходимо каждый месяц составлять таблицу учета расходов (рис. 1.21, файл *4-Шаблоны таблиц.xlsm* на компакт-диске). Создадим, а потом отредактируем макрос, который позволит вам быстро получать требуемый шаблон для простой табличной ведомости.



B7 =СУММ(B2:B6)

	A	B	C
1	Статья расходов	Расходы	
2	Телефон	3123	
3	Аренда	3123	
4	Амортизация	432	
5	Страховка	342	
6	Заработная плата	53	
7	Итого	7073	
8			

Рис. 1.21. Шаблон таблицы расходов

Итак, для создания шаблона неотформатированной таблицы учета расходов выполните следующие действия.

1. Запустите макрорекордер, для чего перейдите на вкладку **Разработчик** ленты и в группе **Код** нажмите кнопку **Запись макроса**.
2. В открывшемся окне **Запись макроса** ведите в поле **Имя макроса** — `Построение_шаблона_таблицы_Простой`, а в поле **Сохранить в** выберите из списка **Эта книга**.
3. Перейдите к ячейке **B1** и введите в нее слово `Расходы`.
4. Заполните ячейки **A1**, ..., **A7** так, как показано на рис. 1.21.
5. Далее, укажите ячейку **B7** и введите в нее формулу `=СУММ(B2:B6)`.
6. Выделите столбец **A** и перейдите на вкладку **Главная**, в группе **Ячейки** щелкните по кнопке с выпадающим списком **Формат** и выберите команду **Автоподбор ширины столбца**.
7. Проделайте предыдущее действие и для столбца **B**: выделите столбец **B**, перейдите на вкладку **Главная**, далее в группе **Ячейки** щелкните по кнопке с выпадающим списком **Формат** и выберите команду **Автоподбор ширины столбца**.
8. Установите указатель в ячейку **B2**.
9. Завершите работу макрорекордера, для чего перейдите на вкладку **Разработчик** ленты и в группе **Код** нажмите на кнопку **Остановить запись**.

Итак, в результате на листе стандартного модуля будет записан следующий макрос (листинг 1.7, файл *4-Шаблоны таблиц.xlsm* на компакт-диске).

Листинг 1.7. Макрос по составлению шаблона отчета

```
Sub Построение_шаблона_таблицы_Простой()
' Построение_шаблона_таблицы_Простой Макрос
    ActiveCell.FormulaR1C1 = "Статья расходов"
    Range("B1").Select
    ActiveCell.FormulaR1C1 = "Расходы"
    Range("A2").Select
    ActiveCell.FormulaR1C1 = "Телефон"
    Range("A3").Select
    ActiveCell.FormulaR1C1 = "Аренда"
    Range("A4").Select
    ActiveCell.FormulaR1C1 = "Амортизация"
    Range("A5").Select
```

```
    ActiveCell.FormulaR1C1 = "Страховка"
    Range("A6").Select
    ActiveCell.FormulaR1C1 = "Заработная плата"
    Range("A7").Select
    ActiveCell.FormulaR1C1 = "Итого"
    Range("B7").Select
    ActiveCell.FormulaR1C1 = "=SUM(R[-5]C:R[-1]C)"
    Columns("A:A").Select
    Selection.Columns.AutoFit
    Columns("B:B").Select
    Selection.Columns.AutoFit
    Range("B2").Select
End Sub
```

Итак, макрос нами записан, а теперь обсудим, как можно упростить его код для создания шаблона отчета.

Первые пятнадцать строк кода макроса осуществляют ввод текстовой информации в выбранные ячейки рабочего листа. Причем, начиная со второй строки кода, для ввода используется парная инструкция. Поэтому представляется разумным вместо

```
Range("B1").Select
ActiveCell.FormulaR1C1 = "Расходы"
```

записать инструкцию

```
Range("B1").Value= "Расходы"
```

Аналогично можно записать и самую первую инструкцию макроса, т. е.:

```
Range("A1").Value= "Статья расходов"
```

Следующие две инструкции реализуют ввод формулы `=СУММ(B2:B6)` в ячейку **B7**. В макросе эта формула записана в относительном формате R1C1 (этот стиль ссылок используется в программах на VBA):

```
Range("B7").Select
ActiveCell.FormulaR1C1 = "=SUM(R[-5]C:R[-1]C)"
```

Разумно эти две инструкции заменить только одной, используя привычный стиль ссылок A1 и формулу локальной версии:

```
Range("B7").FormulaLocal = "=СУММ(B2:B6)"
```

Предпоследние четыре инструкции макроса реализуют автоматический подбор ширины для столбцов **A** и **B** так, чтобы в них помещались все введенные строки текста:

```
Columns("A:A").Select
Selection.Columns.AutoFit
Columns("B:B").Select
Selection.Columns.AutoFit
```

Можно сократить данный код, заменив эти четыре инструкции двумя:

```
Columns("A:A").AutoFit
Columns("B:B").AutoFit
```

Последняя инструкция макроса устанавливает указатель в ячейку **B2**. Оставим ее без изменения.

Отметим также, что для пользователя было бы удобно, если имя рабочего листа будет совпадать с названием месяца, за который составляется отчет. Поэтому естественно в процедуру добавить инструкции, которые привели бы к отображению на экране диалогового окна, предлагающего изменить название листа. Если пользователь соглашается с этим предложением, то он вводит в поле ввода появившегося диалогового окна новое имя рабочего листа и нажимает на кнопку **OK**, а процедура сама уж переименует лист.

Итак, код окончательного варианта процедуры составления шаблона состоит всего из нескольких инструкций и, что важно, значительно более функционален, чем макрос, на основе которого он был составлен (см. файл 4-*Шаблоны таблиц.xlsm* на компакт-диске).

Рекомендуем вам также для закрепления изложенного здесь материала создать макрос форматирования для получения шаблона таблицы, оформленной по вашему желанию, и макрос, который будет очищать рабочий лист от имеющихся данных и форматирования, полученного в результате выполнения макроса форматирования.

Управление диаграммой

А теперь представим, что вам, как менеджеру фирмы "ABC", необходимо создать годовой отчет о доходах фирмы, причем построить диаграммы дохода только за январь, за январь и февраль, с января по март и т. д., всего 12 диаграмм (см. файл *5-Управление диаграммой.xlsm* на компакт-диске). Конечно, можно построить все эти диаграммы, а можно обойтись и одной. В этом случае нужно применить инструмент, позволяющий вводить заданный временной интервал, а диаграмма автоматически должна перестраиваться в соответствии с ним. Тут, конечно, на помощь приходят элементы управления — либо **Список**, либо **Поле со списком**. Итак, выполните следующие действия:

1. В ячейки **A1** и **B1** введите соответственно текст для заголовков столбцов: `Месяц` и `Доход`.
2. В диапазон ячеек **A2:A13** добавьте названия месяцев (рис. 1.22), используя функцию автозаполнения в MS Excel.
3. В диапазон **B2:B13** введите доходы фирмы.
4. По этим двум диапазонам (**A2:A13** и **B2:B13**) постройте диаграмму. Для этого: выделите диапазон **A2:B13**, перейдите на вкладку **Вставка** ленты, далее в группе **Диаграммы** щелкните по кнопке со списком **Гистограммы** и выберите из открывшейся коллекции необходимый вид для вашей диаграммы
5. На рабочем листе расположите элемент управления **Поле со списком**, который и позволит произвести выбор временного интервала: перейдите на вкладку **Разработчик** ленты, в группе **Элементы управления** щелкните по кнопке со списком **Вставить**, из открывшейся коллекции выберите в разделе **Элементы ActiveX** элемент управления **Поле со списком**  . Далее перейдите на рабочий лист и нарисуйте элемент управления необходимого размера.
6. Установите в окне **Properties** для элемента управления **Поле со списком** значение свойства `ListFillRange` равным `A2:A13`. Это свойство заполняет список на основе данных из указанного диапазона.

7. Щелкните дважды по созданному элементу управления **Поле со списком** и перейдите в окно модуля рабочего листа.
8. Наберите в модуле рабочего листа код на языке VBA из листинга 1.8.
9. Перейдите на лист рабочего листа и проверьте с использованием элемента управления **Поле со списком** возможность быстрого построения диаграмм о доходах фирмы "ABC".

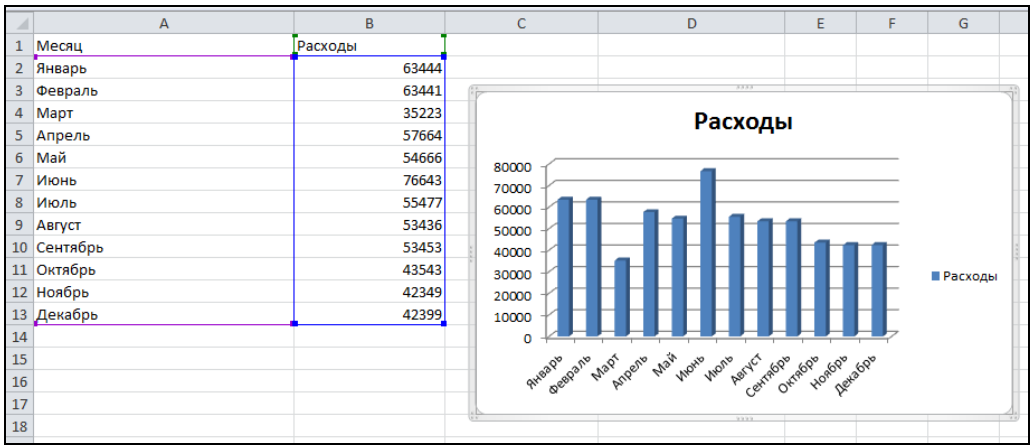


Рис. 1.22. Управление диаграммой

Листинг 1.8. Управление диаграммой

```
Private Sub ComboBox1_Change()
   Dim r As Integer
   ActiveSheet.ChartObjects(1).Activate
   r = ComboBox1.ListIndex + 2
   With ActiveChart
     .SetSourceData Source:= _
           Sheets(1).Range(Cells(2, 2), Cells(r, 2)), PlotBy:=xlColumns
     .SeriesCollection(1).XValues = Sheets(1).Range(Cells(2, 1), Cells(r, 1))
   End With
End Sub
```

Наши итоги

Итак, кратко познакомившись с языком программирования VBA, вы научились писать простейшие пользовательские функции и макросы, редактировать их код в соответствующем окне модуля, использовать некоторые элементы управления. Кроме того, вы познакомились с теоретическими основами объектно-ориентированного программирования, что, несомненно, поможет вам в дальнейшем при создании собственных приложений с использованием возможностей языка VBA.

Глава 2

Как организуются программы на языке VBA

При создании собственных приложений вам, конечно же, не обойтись без программирования. Поэтому в этой главе мы рассмотрим некоторые структуры и функции языка Visual Basic for Applications. Отметим, что язык VBA — это полнофункциональный язык для разработки приложений. Поэтому ему присущи все те конструкции, которые встречаются в других алгоритмических языках. Кроме того, VBA является объектно-ориентированным языком программирования и общим инструментом для всех приложений Microsoft Office, позволяющим решать любые задачи программирования, начиная от автоматизации действий конкретного пользователя и заканчивая разработкой полномасштабных приложений, использующих Microsoft Office в качестве среды разработки.

При изучении VBA важны следующие аспекты:

- синтаксис языка Visual Basic for Applications;
- объектные модели, применяемые в приложениях Excel *(см. приложение 1)*;
- интегрированная среда разработки VBA, которая включает в себя как редактор кода программных модулей, так и большое количество средств отладки этого кода *(см. приложение 2)*.

В данной главе мы познакомимся с синтаксисом языка VBA, рассмотрим управляющие конструкции и разберем некоторые примеры, связанные с ними.

> ***Примечание***
>
> Файлы рассмотренных в данной главе примеров можно найти в папке Glava_2 на компакт-диске.

Язык Visual Basic for Applications: как он устроен?

Быстрый взгляд на процедуры и функции

Основными компонентами программы на VBA являются процедуры и функции, которые представляют собой фрагменты программного кода, заключенные между операторами `Sub` и `End Sub` или между операторами `Function` и `End Function`. Например, процедуру на языке VBA можно записать в виде:

```
Sub <имяПроцедуры>(<аргумент1>, <аргумент2>, ... )
  <операторVBA1>
```

```
  <операторVBA2>
  ...
End Sub
```

В отличие от процедуры, имя функции выступает также в качестве переменной и используется для возвращения значения в точку вызова функции. Функция в общем виде записывается следующим образом:

```
Function <имяФункции>(<аргумент1>, <аргумент2>, ... )
  <операторVBA1>
  <операторVBA2>
  ...
  <имяФункции> = <возвращаемоеЗначение>
End Function
```

Как правило, программа состоит из многих процедур и функций, которые могут размещаться в одном или в нескольких *модулях*. Модули группируются в *проекты*, при этом в одном проекте могут размещаться несколько различных программ, использующих общие модули или процедуры.

Каждая из процедур, находящихся в одном модуле, должна иметь уникальное имя, однако в проекте может быть несколько различных модулей. Обычно рекомендуется в пределах одного проекта использовать только уникальные имена процедур, но допустимы и исключения. В том случае, если в проекте содержится несколько различных процедур с одним и тем же именем, для уточнения имени при вызове процедуры следует использовать следующий синтаксис:

```
<имяМодуля>.<имяПроцедуры>
```

При этом если имя модуля состоит из нескольких слов, следует заключать это имя в квадратные скобки. Например, если модуль называется `Математические процедуры`, а процедура — `Секанс()`, вызов может выглядеть так:

```
[Математические процедуры].Секанс
```

Допускается также использование процедур, расположенных и в других проектах. При этом может потребоваться еще один уровень уточнения имени:

```
<имяПроекта>.<имяМодуля>.<имяПроцедуры>
```

Заметим, что для использования написанных процедур или функций их необходимо вызвать. Процедуру с непустым списком аргументов можно вызвать только из другой процедуры или функции, использовав ее имя со списком фактических значений аргументов в качестве одного из операторов VBA. Функцию можно вызвать не только с помощью отдельного оператора VBA, но и поместив ее имя со списком фактических значений аргументов непосредственно в формулу или выражение в программе на VBA или, например, прямо в формулу активной ячейки рабочего листа Excel.

Если вызываемая процедура имеет уникальное имя и находится в том же модуле, что и вызывающая процедура, то для ее вызова достаточно указать это имя и задать список фактических значений аргументов, не заключая его в скобки. Второй способ вызова процедуры состоит в использовании оператора `Call`. Сначала идет оператор `Call`, затем имя процедуры и список параметров, в этом случае обя-

зательно заключенный в скобки. Вот примеры вызова процедуры под именем `CRC()` с передачей ей двух аргументов (константы и выражения):

```
CRC 7, i + 2
```

или

```
Call CRC(7, i + 2)
```

Функцию можно вызывать аналогично процедуре, однако чаще применяется другой способ вызова функций: использование ее имени с заключенным в скобки списком параметров в правой части оператора присваивания. Вот пример вызова двух функций — `Left()` и `Mid()` и использования возвращаемых ими значений в выражении:

```
yStr = Left(y, 1) & Mid(y, 2, 1)
```

Возможны два разных способа передачи переменных процедуре или функции: по ссылке и по значению. Если переменная передается *по ссылке*, то это означает, что процедуре или функции будет передан адрес этой переменной в памяти. При этом происходит отождествление формального аргумента процедуры и переданного ей фактического параметра. Поэтому вызываемая процедура может изменить значение фактического параметра: если будет изменен формальный аргумент процедуры, то это скажется на значении переданного ей при вызове фактического параметра. Если же фактический параметр передается *по значению*, то формальный аргумент вызываемой процедуры или функции получает только значение фактического параметра, но не саму переменную, используемую в качестве этого параметра. Тем самым все изменения значения формального аргумента не скажутся на значении переменной, являющейся фактическим параметром.

Способ передачи параметров процедуре или функции указывается при описании ее аргументов: имени аргумента может предшествовать явный описатель способа передачи. Описатель `ByRef` определяет передачу по ссылке, а `ByVal` — по значению. Если явное указание способа передачи параметра отсутствует, то по умолчанию подразумевается передача по ссылке.

Поясним сказанное на примере. Пусть имеются следующие описания двух процедур:

```
Sub Main()
    a = 10
    b = 20
    c = 30
    Call Example1(a, b, c)
    Call MsgBox(a)
    Call MsgBox(b)
    Call MsgBox(c)
End Sub

Sub Example1(x, ByVal y, ByRef z)
    x = x + 1
    y = y + 1
    z = z + 1
    Call MsgBox(x)
    Call MsgBox(y)
    Call MsgBox(z)
End Sub
```

Вспомогательная процедура `Example1()` использует в качестве формальных аргументов три переменные, описанные по-разному. В теле этой процедуры каждый из них увеличивается на единицу, а затем их значения выводятся на экран с помощью функции `MsgBox()`. Основная процедура `Main()` устанавливает значения переменных `a`, `b` и `c`, а затем передает их в качестве фактических аргументов процедуре `Example1()`. При этом первый аргумент передается по ссылке (по умолчанию), второй — по значению, а третий — тоже по ссылке. После возврата из процедуры `Example1()` основная процедура также выводит на экран значения трех переменных, передававшихся в качестве аргументов. Всего на экран выводится шесть значений:

- числа 11, 21 и 31 (все полученные значения увеличены на 1 и выводятся процедурой `Example1()`);
- числа 11, 20 и 31 (эти значения выводятся процедурой `Main()`, причем переменные, переданные по ссылке, увеличились, а переменная, переданная по значению, — нет).

Переменные, константы и типы данных

В VBA для хранения временных значений, передачи параметров и проведения вычислений используются переменные. Обычно перед тем, как использовать переменную, выполняется ее объявление, т. е. вы заранее сообщаете, какие именно имена переменных вы будете использовать в своей программе, при этом объявляется также тип данных, для хранения которых предназначена эта переменная. В VBA для объявления переменной используется оператор `Dim`:

```
Dim <имяПеременной> [As <типДанных>]
```

Например,

```
Dim i As Integer, j As Integer
Dim x As Double
```

В VBA установлены следующие правила именования переменных. Имя не может быть длиннее 255 символов и должно начинаться с буквы, за которой могут следовать буквы, цифры или символ подчеркивания. Буквы в верхнем и нижнем регистре не различаются. Имя не должно содержать пробелов, знаков препинания или специальных символов, за исключением самого последнего знака. В конце к имени переменной может быть добавлен еще один из шести специальных символов — *описателей типа данных*:

```
! # $ % & @
```

Эти символы не являются частью имени переменной: если в программе используются одновременно имена `string1$` и `string1`, то они ссылаются на одну и ту же строковую переменную.

Кроме того, не допускается использование в качестве имен переменных ключевых слов VBA и имен стандартных объектов. Именно поэтому рекомендуется начинать имена переменных со строчной, а не с прописной буквы.

Допускается использование в именах переменных букв не только латинского алфавита, но и кириллицы, что может оказаться удобным для русскоязычных пользователей.

Заметим также, что в VBA объявление переменных не является обязательным, и допускается использование неописанных переменных. Выделение памяти переменным может выполняться динамически, а тип данных, хранящихся в переменной, может определяться по последнему символу имени переменной.

В принципе, программист решает сам, каким образом он будет использовать переменные в своей программе. Для этой цели в VBA существует оператор:

```
Option Explicit
```

Если вы начнете свой модуль с данного оператора (он должен быть расположен в самом начале модуля, до того, как начнется первая процедура этого модуля), то VBA будет требовать обязательного объявления переменных в этом модуле и генерировать сообщения об ошибке каждый раз, когда встретит необъявленную переменную. Кроме того, если вы установите параметр **Require Variable Declaration** (Явное описание переменных) на вкладке **Editor** (Редактор) диалогового окна **Options** (Параметры) редактора VBA (для перехода к окну **Options** воспользуйтесь командой **Tools | Options** в интегрированной среде разработки приложений редактора Visual Basic), то VBA каждый раз будет генерировать сообщение об ошибке, если встретит необъявленную переменную.

Установка этого параметра приводит к тому, что редактор Visual Basic будет автоматически добавлять оператор `Option Explicit` в начало каждого вновь создаваемого модуля. Однако данный установленный параметр *не влияет* на все ранее созданные модули — если вы хотите добавить подобный оператор к уже существующим модулям, вам необходимо прописать это вручную.

Краткий перечень используемых типов данных VBA приведен в табл. 2.1.

Таблица 2.1. *Типы данных VBA*

Тип данных	Описание
`Array`	Массив переменных, для ссылки на конкретный элемент массива используется индекс. Требуемая память зависит от размеров массива
`Boolean`	Принимает одно из двух логических значений: `True` или `False`. Требуемая память — 2 байта
`Byte`	Число без знака от 0 до 255. Требуемая память — 1 байт
`Currency`	Используется для выполнения денежных вычислений с фиксированным количеством знаков после десятичной запятой, в тех случаях, когда важно избежать возможных ошибок округления. Диапазон возможных значений: от –922 337 203 685 477,5808 до 922 337 203 685 477,5807. Требуемая память — 8 байтов. Символ определения типа по умолчанию — `@`
`Date`	Используется для хранения дат и времени. Диапазон возможных значений: от 1 января 0100 г. до 31 декабря 9999 г. Требуемая память — 8 байтов
`Double`	Числовые значения с плавающей запятой двойной точности. Диапазон возможных значений для отрицательных чисел: от –1,7976939486232E308 до –4,94065645841247E–324. Диапазон возможных значений для положительных чисел: от 4,94065645841247E–324 до 1,7976939486232E308. Требуемая память — 8 байтов. Символ определения типа по умолчанию — `#`

Таблица 2.1 *(окончание)*

Тип данных	Описание
`Integer`	Короткие целые числовые значения. Диапазон возможных значений: от –32 768 до 32 767. Требуемая память — 2 байта. Символ определения типа по умолчанию — `%`
`Long`	Длинные целые числовые значения. Диапазон возможных значений: от –2 147 483 648 до 2 147 483 647. Требуемая память — 4 байта. Символ определения типа по умолчанию — `&`
`Object`	Используется для хранения ссылок на объекты. Требуемая память — 4 байта
`Single`	Числовые значения с плавающей точкой обычной точности. Диапазон возможных значений для отрицательных чисел: от –3,402823E38 до –1,401298E–45. Диапазон возможных значений для положительных чисел: от 1,401298E–45 до 3,402823E38. Требуемая память — 4 байта. Символ определения типа по умолчанию — `!`
`String`	Используется для хранения строковых значений. Длина строки — от 0 до 64 Кбайт. Требуемая память — 1 байт на символ. Символ определения типа по умолчанию — `$`
`Variant`	Может использоваться для хранения различных типов данных: даты/времени, чисел с плавающей точкой, целых чисел, строк, объектов. Требуемая память — 16 байт плюс 1 байт на каждый символ строковых значений. Данные типа `Variant` могут иметь особое значение `Null`, которое означает, что данные отсутствуют, неизвестны или неприменимы. Например, по умолчанию данные в полях таблицы базы данных имеют тип `Variant`. Поэтому, если оставить поле пустым, ему будет присвоено значение `Null`. Функция `IsNull` проверяет, есть ли указанное значение `Null`
Определяемый пользователем тип (с помощью ключевого слова `Type`)	Назначение и размер выделяемой памяти зависят от определения. Используется для описания структур данных. Позволяет хранить в одной переменной такого типа множество различных значений разного типа

При описании переменной указание типа данных может быть опущено. Тип переменной в таком случае будет определяться последним символом имени переменной: `@`, `#`, `%`, `&`, `!` или `$` (`Currency`, `Double`, `Integer`, `Long`, `Single` или `String`, соответственно).

Если последний символ не является ни одним из перечисленных ранее и явное указание типа тоже не используется, то переменной будет назначен по умолчанию тип данных `Variant`, который позволяет хранить в ней данные любого типа.

Следует запомнить, что нельзя использовать в одной и той же процедуре имена переменных, отличающиеся друг от друга только специальным символом определения типа в конце имени переменной. Например, не допускается одновременное использование переменных `var$` и `var%`. Не допускается и явное объявление переменной, содержащей символ определения типа в конце имени, с помощью описа-

теля `Dim <имяПеременной> As <типПеременной>`. Так, например, вы получите сообщение об ошибке, попытавшись ввести любое из следующих определений:

```
Dim var1% As String
Dim var2% As Integer
```

Для определения типа данных аргументов процедуры или функции используется описание типа данных непосредственно в заглавной строке процедуры или функции. Например, следующая заглавная строка процедуры описывает ее параметры как переменные строкового типа:

```
Sub SpSt(str1 As String, str2 As String, str3 As String)
```

Определение типа данных возвращаемого функцией значения завершает заглавную строку функции. Например, такое объявление описывает возвращаемое функцией значение как переменную короткого целого типа:

```
Function FSpS(str1 As String) As Integer
```

Чтобы программа работала быстрее и занимала меньше памяти, рекомендуется использовать, когда это возможно, конкретные типы переменных, а не универсальный тип `Variant`. На обработку переменных типа `Variant` требуется не только дополнительная память (сравните размеры, приведенные в табл. 2.1), но и дополнительное время: в момент обработки определяется, к какому конкретному типу данных принадлежит такая переменная, а также при необходимости выполняется преобразование данных к нужному типу. Однако бывают случаи, когда переменные типа `Variant` просто необходимы: например, когда вы точно не знаете, какие именно данные будут присвоены переменной.

Рассмотрим также использование именованных констант. Для их описания служит оператор `Const`, схожий с оператором описания переменных `Dim`. Синтаксис этого оператора следующий:

```
Const <имяКонстанты> [As <типДанных>] = <выражение>
```

где `<выражение>` — это любое значение или формула, возвращающая значение, которое должно использоваться в качестве константы. Например, следующий оператор определяет целую константу `maxLen`:

```
Const maxLen% = 30
```

Как и переменные, константы могут содержать значения различных типов данных, но при этом они не меняют своих значений во время выполнения программы.

> ***Совет***
>
> Если вы собираетесь использовать в вашей программе какие-либо константы, то рекомендуется давать этим константам осмысленные имена и описать их в самом начале модуля, а затем использовать всюду только именованные константы. Это делает программу не только понятнее, но и проще в сопровождении и отладке.

Кроме описываемых пользователем констант, существуют еще предопределенные встроенные константы, которые включаются в тексты программ без предварительного описания. Сведения о предопределенных встроенных константах, используемых для различных объектов приложений Microsoft Office и Visual Basic, можно найти в справке — в разделах описания свойств объектов (реже в разделах описания методов). При именовании встроенных констант используется стандартное со-

глашение, позволяющее определить, к объектам какого приложения относится эта константа. Например, встроенные константы, относящиеся к объектам Excel, начинаются с префикса `xl`, к объектам VBA — `vb`.

Ссылки на объекты

Кроме обычных переменных, в Visual Basic часто используются переменные, представляющие собой ссылку на объект. Оказывается, использование переменных для ссылок на объекты позволяет не только сократить и упростить текст программы, но и существенно ускорить ее работу.

Использование переменной-объекта немного отличается от применения обычных переменных: нужно не только объявить такую переменную, но и перед ее использованием назначить ей соответствующий объект с помощью специального оператора `Set`. Синтаксис объявления и назначения следующий:

```
Dim <имяПеременной> As Object
Set <имяПеременной> = <ссылкаНаОбъект>
```

Иногда при объявлении такой переменной удобно заранее указать конкретный тип объекта — можно использовать любой конкретный объект из объектной модели Microsoft Office, например:

```
Dim MyBase As Database
Set MyBase = CurrentDb()
```

> ***Совет***
>
> Если вам необходимо повысить быстродействие вашей программы, то рекомендуется при описании объектных переменных использовать конкретные объекты модели Microsoft Office, а не универсальное описание `Object`.

Объектная переменная будет указывать на объект до тех пор, пока мы другим оператором `Set` не присвоим ей ссылку на другой объект этого же типа или значение `Nothing`, означающее, что переменная не содержит никакой ссылки, например:

```
Set txt = Nothing
```

После такого действия переменная продолжает существовать, хотя и не ссылается ни на какой объект. Другим оператором `Set` ей можно снова присвоить ссылку на объект.

> ***Примечание***
>
> Обратите внимание, что объектные переменные, в отличие от обычных переменных, содержащих значения, содержат только ссылки на объекты, а не сами объекты или их копии.

Область действия переменных и процедур

Все процедуры, функции, переменные и константы в VBA имеют свою *область действия*. Это означает, что они могут использоваться только в определенном месте программного кода — именно там, где они описаны. Например, если переменная `A` описана с помощью оператора `Dim` в теле процедуры с именем `Pro1()`, именно эта

процедура и является ее областью действия. Таким образом, если существует другая процедура `Pro2()`, вы не можете использовать в ней эту же переменную. Если вы попытаетесь сделать это, то либо получите сообщение об ошибке из-за использования неописанной переменной (в том случае, если используется упоминавшийся ранее оператор `Option Explicit`), либо просто получите другую переменную — с тем же самым именем, но никак не связанную с одноименной переменной из первой процедуры.

Существуют три типа области видимости переменной:

- переменные уровня процедуры распознаются только в процедуре, в которой они описаны при помощи инструкции `Dim` или `Static`. Такие переменные называются *локальными*;
- переменные уровня модуля используются только в модуле, в котором они описаны, но не в других модулях данного проекта. Они описываются при помощи оператора `Dim` или `Private` в области описания модуля, т. е. перед описанием процедур;
- переменные уровня модуля, описанные при помощи инструкции `Public`, являются доступными для всех процедур проекта. Такие переменные называются *открытыми.*

Закрытая (private) *переменная* сохраняет свое значение, только пока выполняется процедура, в которой эта переменная описана. По завершении процедуры значение переменной теряется, и при повторном запуске процедуры его надо заново инициализировать. Переменные, описанные оператором `Static`, сохраняют свое значение после выхода из процедуры, пока работает программа.

Рассмотрим теперь область действия процедур и функций. Процедуры и функции имеют только два уровня области действия: уровень модуля и уровень проекта. По умолчанию используется уровень проекта. Таким образом, процедура или функция может быть вызвана любой другой процедурой или функцией в этом проекте. При описании процедур и функций на уровне проекта может также использоваться необязательное ключевое слово `Public`. Никакого влияния на процедуру наличие или отсутствие этого слова не оказывает.

Если требуется описать процедуру, используемую только на уровне модуля, то применяется ключевое слово `Private`. Учтите, что такое описание не только сужает область действия процедуры, но и запрещает ее использование как самостоятельной процедуры — ее можно вызвать только из другой процедуры.

Наконец, при описании процедур или функций может использоваться ключевое слово `Static`. Оно никак не влияет на область действия процедуры, но воздействует на все переменные, описанные внутри этой процедуры или функции. В этом случае все локальные переменные получают статус `Static`, а следовательно, остаются в памяти после завершения такой процедуры и при повторном ее вызове сохраняют свои прежние значения.

Рассмотрим пример модуля (листинг 2.1, см. также файл *1-Примеры использования некоторых структур данных.xlsm* на компакт-диске).

Листинг 2.1. Пример модуля

```
Public A1 As String
Private A2 As Integer
Dim A3 As Single
Sub Pro1()
    Dim A4 As Integer
    Static A5 As Integer
    A1 = "Текстовая строка 1"
    A2 = 2
    A3 = 3.14
    A4 = A4 + 4
    A5 = A5 + 5

    MsgBox A4
    MsgBox A5
End Sub

Sub Pro2()
    Pro1

    MsgBox A1
    MsgBox A2
    MsgBox A3
    MsgBox A4
    MsgBox A5

    Proc1
End Sub
```

В этом примере переменная `A1` определена на уровне всего проекта (использовано ключевое слово `Public`), переменные `A2` и `A3` определены на уровне модуля, переменная `A4` — на уровне процедуры `Pro1()`, а переменная `A5` хотя и определена в теле процедуры `Pro1()`, но описана как статическая.

При вызове процедуры `Pro2()` произойдет следующее: из этой процедуры будет вызвана процедура `Pro1()`, которая присвоит значения всем пяти переменным `A1`, `A2`, `A3`, `A4` и `A5`, а затем покажет текущие значения переменных `A4` и `A5` в диалоговом окне.

После завершения процедуры `Pro1()` будут выведены текущие значения переменных `A1`—`A5`. При этом окажется, что переменные `A1`—`A3` сохранили свои значения, поскольку они описаны на уровне модуля, а переменные `A4` и `A5` имеют пустые значения, поскольку областью действия этих переменных являются процедуры, в которых они используются. Никакие изменения этих переменных внутри одной из процедур не имеют отношения к аналогичным переменным в другой процедуре — на самом деле это разные переменные, просто для них используются совпадающие имена.

После этого происходит еще один вызов процедуры `Pro1()`, и она вновь изменяет и выводит на экран значения переменных `A4` и `A5`. При этом переменная `A4` вновь получит значение 4, поскольку при новом вызове процедуры для этой переменной будет заново выделена память, и она будет инициализирована пустым значением. В отличие от `A4`, переменная `A5`, описанная как статическая, сохранит свое прежнее значение от предыдущего вызова этой процедуры, в результате ее значение при повторном вызове окажется равным 10.

Что нужно знать о массивах?

Как используются массивы?

Массив — это переменная, в которой хранится одновременно несколько значений одинакового типа. Таким образом, массив представляет собой совокупность однотипных индексированных переменных.

Количество используемых индексов массива также может быть различным. Чаще всего используются массивы с одним или двумя индексами, реже — с тремя, еще большее количество индексов встречается крайне редко. В VBA допускается использование до 60 индексов. О количестве индексов массива обычно говорят как о размерности массива. Массивы с одним индексом называют одномерными, с двумя — двумерными и т. д. Массивы с бóльшим количеством измерений могут занимать очень большие объемы памяти, так что следует быть осторожным в их применении.

Прежде чем использовать массив, нужно обязательно объявить его с помощью оператора `Dim` и указать тип хранящихся в массиве значений. Все значения в массиве принадлежат к одному типу данных. Это ограничение на практике можно обойти, использовав при объявлении массива тип `Variant` — в этом случае элементы массива смогут принимать значения разных типов. Синтаксис оператора объявления массива следующий:

```
Dim <имяМассива>(<размер1>, <размер2>, ... ) As <типДанных>
```

где указанные в скобках величины *<размер1>*, *<размер2>* задают размеры массива — количество индексов и максимально допустимое значение для каждого конкретного индекса. При этом индексирование элементов массива по умолчанию начинается с нуля. Так, объявление:

```
Dim Array1(9) As Integer
```

определяет одномерный массив из 10 элементов, являющихся переменными целого типа, а объявление:

```
Dim Array2(4, 9) As Variant
```

определяет двумерный массив 5×10 из 50 элементов, являющихся переменными универсального типа `Variant`.

> ***Примечание***
>
> В качестве стандартного значения нижней границы массива (индекса) может использоваться не только ноль. Чтобы изменить это стандартное значение, нужно воспользоваться оператором `Option Base`. Например, если поместить в начало вашего

модуля оператор `Option Base 1`, то индексирование элементов массивов по умолчанию будет начинаться не с нуля, а с единицы.

Например, в следующем операторе объявляется вектор, состоящий из 11 элементов.

```
Option Base 1
Dim A(11) As Integer
```

Другим способом изменения базового индекса является использование ключевого слова `To` при объявлении массива.

```
Dim B(1 To 3, 1 To 3) As Single
Dim A(1 To 12) As Integer
```

При объявлении массива можно указать не только верхнюю границу индекса, но и его нижнюю границу, т. е. явно задать диапазон изменения конкретного индекса массива, причем нижняя граница может быть любым целым числом, не обязательно неотрицательным. Синтаксис такого определения будет выглядеть так:

```
Dim <имяМассива>(<мин1> To <макс1>, ...) As <типДанных>
```

Например, если вы собираетесь работать с массивом метеорологических данных, представляющих собой средние дневные температуры за последние две недели, то может оказаться весьма удобным дать следующее определение массива:

```
Dim Temperature(-14 To 0) As Single
```

При этом, например, `Temperature(-2)` будет соответствовать позавчерашней температуре, а для определения нужного индекса для интересующего вас дня будет достаточно использовать разность дат.

В приведенных ранее примерах речь шла о массивах фиксированного размера, количество элементов в которых было явно указано во время описания массива в операторе `Dim`. Такие массивы называются *статическими*. В VBA допускается использование и *динамических* массивов, размеры которых при описании не фиксируются. Определение размера динамического массива может быть сделано непосредственно во время выполнения программы.

При определении динамического массива в операторе `Dim` после имени массива стоят лишь пустые скобки и описание типа переменных. Количество индексов и диапазон их изменения не задаются. Однако перед тем как использовать массив, нужно выполнить оператор `ReDim`, который задаст размерность и диапазоны изменения индексов динамического массива.

Синтаксис объявления и определения размеров динамического массива такой:

```
Dim <имяМассива>() As <типДанных>
ReDim <имяМассива>(<размер1>, <размер2>, ...)
```

Вот как может выглядеть объявление, определение размеров и использование динамического массива, а затем последующее изменение размерности и размеров этого же массива:

```
Dim dArray() As Variant
ReDim dArray(1, 2)
dArray(0, 0) = 2
dArray(0, 1) = 3
k = dArray(0, 0) + dArray(0, 1)
ReDim dArray(k)
dArray(0) = "Строка1"
```

В этом примере массив `dArray` сначала определяется как двумерный массив из шести элементов, а затем переопределяется как одномерный массив, причем верхняя граница индекса задается значением переменной `k`.

ПРИМЕЧАНИЕ

Чтобы определить текущую нижнюю или верхнюю границу массива, можно использовать функции `Lbound()` и `Ubound()`, соответственно.
Например, в следующем коде отобразится 100 и 5.

```
Dim A(1 To 100, 0 To 5)
MsgBox UBound(A, 1) & vbCr & UBound(A, 2)
```

Следующие инструкции позволяют перебирать элементы массива без указания его размерности.

```
Dim d As Variant
Dim i As Integer
d = Array("Пн", "Вт", "Ср", "Чт", "Пт")
For i = LBound(d) To UBound(d)
   MsgBox d(i)
Next
```

Учтите, что по умолчанию при изменении размеров массива ему заново выделяется память и текущие значения его элементов теряются. Чтобы не потерять текущие значения массива при изменении его размеров, используется ключевое слово `Preserve`. Например, чтобы увеличить размер массива `dArray` на один элемент, не потеряв значений существующих элементов, можно поступить следующим образом:

```
ReDim Preserve dArray(UBound(dArray) + 1)
```

Рассмотрим более подробно отдельные примеры работы с массивами.

Поэлементная инициализация массива

Поэлементную инициализацию массива можно производить:

- последовательностью операторов:

```
Dim B(1, 1) As Single
B(0, 0) = 2
B(0, 1) = 4
B(1, 0) = 1
B(1, 1) = 6
Dim M(1 To 9, 1 To 9) As Integer
```

- оператором цикла:

```
Dim i As Integer
Dim j  As Integer
For i = 1 To 9
   For j = 1 To 9
      M(i, j) = i * j
   Next
Next
```

Инициализация массива при помощи функции *Array()*

Как указывалось ранее, удобным способом определения одномерных массивов является функция `Array()`, преобразующая список элементов, разделенных запятыми, в вектор из этих значений и присваивающая их переменной типа `Variant`. Допустима инициализация как одномерного массива, так и многомерного, за счет применения вложенных конструкций с функциями `Array()`.

- Инициализация одномерного массива:

```
Dim num As Variant
Dim s As Double
num = Array(10, 20)
s = num(0) + num(1)
MsgBox s
```

- Инициализация многомерного массива:

```
Dim CityCountry As Variant
CityCountry = Array(Array("Санкт-Петербург", "Россия"), _
                    Array("Кейптаун", "ЮАР"))
MsgBox CityCountry(0)(0)
' Отобразится Санкт-Петербург
MsgBox CityCountry(0)(1)
' Отобразится Россия
```

Массив и диапазон

В VBA имеется тесная связь между диапазонами и массивами. Допустимо как заполнение массива одним оператором присваивания значениями из ячеек диапазона, так и наоборот, заполнение диапазона ячеек одним оператором присваивания элементами массива. При этом массив должен быть объявлен как переменная типа `Variant`.

- Инициализация массива из диапазона одним оператором присваивания:

```
Dim r As Range
Set r = Range("C1:D3")
Dim M As Variant
M = r.Value
Dim i As Integer
Dim j As Integer
For i = 1 To r.Rows.Count
   For j = 1 To r.Columns.Count
      Cells(i, j).Value = M(i, j)
   Next
Next
```

- Заполнение диапазона из массива посредством одного оператора присваивания:

```
Dim M(1 To 9, 1 To 9)
Dim i As Integer
```

```
   Dim j As Integer
   For i = 1 To 9
      For j = 1 To 9
         M(i, j) = i * j
      Next
   Next
   Dim r As Range
   Set r = Range(Cells(1, 1), Cells(9, 9))
   r.Value = M
```

Использование динамических массивов

Приведем пример использования инструкции `ReDim` для изменения числа элементов и размерностей массива (листинг 2.2). В приводимом примере создается массив с результатами бросания монеты. Монета бросается до тех пор, пока три раза не выпадет орел. Размерность динамического массива после каждого броска корректируется с сохранением в нем ранее записанных результатов бросания монеты, за счет использования ключевого слова `Preserve`.

Листинг 2.2. Изменение размерности динамического массива с сохранением содержания

```
Dim Attempt()
Dim i As Integer
Dim score As Integer
Dim coin As Integer
i = 0
score = 0
Do
   i = i + 1
   coin = Int(2 * Rnd())
   ' 0 — решка
   ' 1 — орел
   If coin = 1 Then score = score + 1
   ReDim Preserve Attempt(i)
   Attempt(i) = coin
Loop Until score = 3
```

Как проверить, содержит ли переменная типа *Variant* массив значений?

Функция `IsArray()` возвращает значение `True`, если указанная переменная типа `Variant` содержит массив, и значение `False` — в противном случае. Например, следующая процедура (листинг 2.3), обрабатывающая событие `SelectionChange` объ-

екта Worksheet, отображает сообщение, если в выделенном диапазоне имеется массив ячеек.

Листинг 2.3. Как проверить, содержит ли переменная типа Variant массив значений. Модуль рабочего листа

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
   Dim A As Variant
   A = Target.Value
   If IsArray(A) Then
      MsgBox "Содержит массив выделенных ячеек"
   Else
     MsgBox "Не содержит массив выделенных ячеек"
   End If
End Sub
```

Повторная инициализация массива и высвобождение памяти, выделенной под массив

Оператор Erase повторно инициализирует элементы массивов фиксированной длины и освобождает память, отведенную для динамического массива. Оператор Erase устанавливает элементы массивов фиксированной длины следующим образом:

- массив чисел или строк фиксированной длины (присваивает каждому элементу значение 0);
- массив строк переменной длины (присваивает каждому элементу значение пустой строки);
- массив типа Variant (присваивает каждому элементу значение Empty).

Например, в следующем коде отобразится сначала 1, а затем 0.

```
Dim A(2) As Integer
A(2) = 1
MsgBox A(2)
Erase A
MsgBox A(2)
```

Оператор Erase также освобождает память, используемую динамическими массивами. Перед тем как из программы вновь станет возможна ссылка на динамический массив, необходимо переопределить размерности переменной массива с помощью инструкции ReDim. Например:

```
Dim B() As Integer
ReDim B(6)
Erase B
ReDim B(3)
```

Структурированные типы данных: что это такое?

Строки

Строка (string) представляет собой последовательность символов, каждый из которых имеет тип `Char`.

Последовательность символов, присваиваемая строковой переменной, должна быть окружена кавычками.

```
Dim str As String
str = "Это строка"
```

Строковая переменная может быть объявлена как строка переменной длины и как строка постоянной длины. В следующем примере переменная `LastName` объявлена как строка переменной длины, а переменная `State` — как строка постоянной длины, состоящая из трех литер. Если переменной `State` будет присвоено значение какого-то строкового выражения, состоящего более чем из двух литер, лишние справа литеры при присваивании будут отброшены.

```
Dim LastName As String
Dim State As String * 2
```

Заметим, что в VBA имеется единственная строковая операция — *конкатенация*. Она применяется для объединения нескольких строк в одну. Операция конкатенации обозначается символом амперсанда (`&`) или сложения (`+`). Во избежание путаницы, как правило, применяется операция конкатенации со знаком `&`. При объединении двух строк вторая строка добавляется непосредственно в конец первой. Результатом является строка большего размера, содержащая целиком обе исходные строки.

В следующем примере переменной `s` присваивается значение `"Visual Basic for Applications"`.

```
Dim s As String
s = "Visual Basic " & "for Applications"
```

Ключевое слово `Empty` возвращает ссылку на пустую строку. Того же эффекта можно добиться, поставив рядом пару кавычек (`""`). Например, следующий оператор отобразит окно с сообщением `"Равносильны"`.

```
If Empty = "" Then MsgBox "Равносильны"
```

Перечисляемый тип

Перечисляемый тип предоставляет удобный способ работы с константами и позволяет ассоциировать значения констант с их именами.

```
[Public | Private] Enum ИмяТипа
   имяЭлемента [= Выражение]
   имяЭлемента [= Выражение]
...
End Enum
```

- *ИмяТипа* — имя перечисляемого типа.
- *имяЭлемента* — имя константы. По умолчанию значение первой константы равно 0, второй 1 и т. д. Через параметр *Выражение* константам можно назначать произвольные значения.
- *Выражение* — значение константы.

В следующем примере (листинг 2.4, файл *1-Примеры использования некоторых структур данных.xlsm* на компакт-диске) с использованием перечисляемого типа задаются значения для идентификации стороны монеты.

Листинг 2.4. Использование перечисляемого типа в примере бросания монеты

```
Enum Coin
    Head = 1
    Tail = -1
End Enum
Sub Attemp()
   Dim r As Integer
   Randomize
   r = 2 * Int(2 * Rnd()) – 1
   Select Case r
    Case Head
        MsgBox "Орел"
    Case Tail
        MsgBox "Решка"
    End Select
End Sub
```

Тип данных, определенный пользователем

Достаточно часто при написании программ применяется тип данных, который позволяет использовать различные типы. *Запись* — это совокупность нескольких элементов, каждый из которых может иметь свой тип. Элемент записи называется *полем*. Запись является частным случаем класса, в котором не определены свойства и методы:

```
[Private | Public] Type ИмяТипа
   имяЭлемента [([размер])] As Тип
   имяЭлемента [([размер])] As Тип
...
End Type
```

- `Private` — используется для описания определяемых пользователем типов, которые доступны только в модуле, содержащем описание.
- `Public` — используется для описания определяемых пользователем типов, которые доступны для всех процедур во всех модулях всех проектов.
- *ИмяТипа* — имя типа, определяемого пользователем.
- *имяЭлемента* — имя элемента, определяемого пользователем типа.
- *размер* — размер элемента, являющегося массивом.

- *Тип* — тип данных элемента; поддерживаются типы `Byte`, `Boolean`, `Integer`, `Long`, `Currency`, `Single`, `Double`, `Date`, `String` (для строк переменной длины), `String * длина` (для строк фиксированной длины), `Object`, `Variant`, другой определяемый пользователем тип или объектный тип.

Инструкция `Type` используется только на уровне модуля. При появлении в модуле класса инструкции `Type` должно предшествовать ключевое слово `Private`.

В листинге 2.5 (см. также файл *1-Примеры использования некоторых структур данных.xlsm* на компакт-диске) инструкция `Type` служит для определения типа данных, инкапсулирующего в себе информацию о сотруднике некоторой фирмы.

Листинг 2.5. Пример типа, определенного пользователем

```
Type Employee
   FirstName As String
   LastName  As String
   Position  As String
   BirthDate As Date
End Type
Sub InitialData()
   Dim emp As Employee
   With emp
      .FirstName = "James"
      .LastName = "Bond"
      .Position = "Secret agent 007"
      .BirthDate = #17/05/80#
   End With
   With emp
      MsgBox .FirstName & vbCr & .LastName & vbCr & .Position & vbCr & _
             .BirthDate
   End With
End Sub
```

ПРИМЕЧАНИЕ

В тексте листинга 2.5 в операторе `MsgBox` использовался такой прием: перенос оператора на следующую строку — символы "пробел" и "_" в конце строки. Кроме того, в операторе `MsgBox` указана встроенная константа `vbCr`, позволяющая вывести необходимую информацию с новой строки в окне сообщений.

Можно создавать массив, содержащий элементы собственного типа. Например, следующий массив (листинг 2.6) состоит из сведений о 20 сотрудниках отдела продаж некоторой фирмы.

Листинг 2.6. Пример массива, элементы которого имеют тип, определенный пользователем

```
Sub InitialData()
   Dim emp(3) As Employee
   With emp(0)
```

```
      .FirstName = "James"
      .LastName = "Bond"
      .Position = "Secret agent 007"
      .BirthDate = #17/05/80#
   End With
   With emp(1)
      .FirstName = "Alice"
      .LastName = "Smith"
      .Position = "Just a secret agent"
      .BirthDate = #06/09/89#
   End With
End Sub
```

Дополнительные элементы языка VBA: как они помогают при написании программ?

Комментарии

Текст, следующий в программе за символом ' вплоть до конца строки, игнорируется компилятором и представляет собой *комментарий*.

Комментарии позволяют добавлять описания и пояснения для тех программистов, которые в будущем станут разбираться в вашей программе. Комментируя всю программу, вы экономите время. Кроме того, комментарии облегчают понимание программы как самим автором, так и теми, кому, при необходимости, вы объясняете код программы.

Комментарии также полезны при отладке программ. Они позволяют временно отключать строки кода программы.

Далее записаны возможные варианты использования комментариев в коде программы.

```
Dim a As Integer
' *************************
' *  a — целая переменная  *
' *************************
Dim b As String ' b — строковая переменная
' b = sin(2) — этот оператор отключен
```

Перенос строки кода

Расположение символов "пробел" и "_" в конце строки позволяет разбить одну строку с оператором на несколько строк, но при этом компилятор будет воспринимать их как единый оператор. При этом надо помнить, что:

- нельзя разбивать переносом строковые константы;
- допустимо не более семи продолжений одной и той же строки;
- сама строка не может состоять более чем из 1024 символов.

В следующем примере первая из конструкций является разбиением второй на две строки:

```
ActiveCell.Offset(rowoffset:=0, _
                  columnoffset:=1).Value = Сумма
```

и

```
ActiveCell.Offset(rowoffset:=0, columnoffset:=1).Value = Сумма
```

Для переноса строковой константы ее надо представить как результат конкатенации нескольких строковых констант, и перенос строки производить по операции конкатенации (`&`).

Приведем пример корректного и некорректного переноса строки `"Visual Basic for Applications"`:

Некорректный перенос:

```
s = "Visual Basic for _
        Applications"
```

Корректный перенос:

```
s = "Visual Basic " _
  & "for Applications"
```

Расположение нескольких операторов в одной строке

Использование знака двоеточия (`:`) позволяет разместить несколько операторов на одной строке. Таким образом, следующие две конструкции эквивалентны:

```
x = 1     ' в переменной x содержится 1
x = x + 2 ' в переменной x содержится 3
```

и

```
x = 1 : x = x + 2
```

Операции VBA

В программах на VBA можно использовать стандартный набор операций над данными. Имеются три типа операций:

- *математические* — выполняются над числами, и их результатом являются числа;
- *отношения* — применяются не только к числам, и их результатом являются логические значения, например `x>y`;
- *логические* — применяются логическими выражениями, и их результатом являются логические значения, например `Not x And y`.

Математические операции

В VBA поддерживается стандартный набор математических операций от сложения до возведения в степень (табл. 2.2).

Таблица 2.2. *Математические операции*

Операция	Описание
exp1 + exp2	Сложение
exp1 – exp2	Вычитание
–exp	Перемена знака

Таблица 2.2 *(окончание)*

Операция	Описание
exp1 * *exp2*	Умножение
exp1 / *exp2*	Деление
exp1 \ *exp2*	Целочисленное деление
exp1 Mod *exp2*	Остаток от деления по модулю
exp1 ^ *exp2*	Возведение в степень

Операции отношения

В табл. 2.3 представлены операции отношения, используемые в VBA.

Таблица 2.3. *Операции отношения*

Операция	Описание
exp1 < *exp2*	Меньше
exp1 > *exp2*	Больше
exp1 <= *exp2*	Меньше или равно
exp1 >= *exp2*	Больше или равно
exp1 <> *exp2*	Не равно
exp1 = *exp2*	Равно
exp1 Is *exp2*	Сравнение двух операндов, содержащих ссылки на объекты
exp1 Like *exp2*	Сравнение двух строковых выражений

Логические операции

В табл. 2.4 приведены основные логические операции, используемые в VBA.

Таблица 2.4. *Основные логические операции*

Операция	Описание
exp1 And *exp2*	Логическое умножение
exp1 Or *exp2*	Логическое сложение
exp1 Xor *exp2*	Исключающее ИЛИ, т. е. возвращает `True` тогда и только тогда, когда только один операнд возвращает `True`
exp1 Not *exp2*	Логическое отрицание

Директива *Option Compare*

Действие операции сравнения `Like` зависит от директивы `Option Compare`, которая располагается в области описания модуля. По умолчанию для каждого модуля считается установленной инструкция `Option Compare Binary`, при которой

различаются строчные и прописные буквы, т. е. следующий оператор вернет значение `False`:

```
Debug.Print "AA" Like "aa"
```

Если же в области описания модуля указана инструкция `Option Compare Text`, то при сравнении строчные и прописные буквы не различаются, и тот же самый оператор на этот раз вернет значение `True`.

В следующем примере (листинг 2.7, см. также файл *1-Примеры использования некоторых структур данных.xlsm* на компакт-диске) в поле ввода диалогового окна пользователь должен ввести свое имя латинскими буквами. При нажатии кнопки **OK** программа анализирует информацию и отображает сообщение. А именно:

- если пользователь забыл ввести имя, то его об этом информируют;
- если во введенном имени присутствуют знаки, отличные от букв латинского алфавита, его об этом информируют;
- если имя состоит только из букв латинского алфавита, то с ним здороваются.

Листинг 2.7. Директива `Option Compare`

```
Option Compare Text
Sub DemoCompare()
   Dim name As String, lng As Integer, i As Integer
   name = InputBox("Введите имя")
   lng = Len(Trim(name))
   If lng = 0 Then
      MsgBox "Забыли ввести имя"
      Exit Sub
   Else
      For i = 1 To lng
         If Not Mid(name, i, 1) Like "[A-Z]" Then
            MsgBox "Имя состоит только" & vbCr "из букв латинского алфавита"
            MsgBox " Привет, " & name
            Exit Sub
         End If
      Next i
   End If
End Sub
```

Приоритеты операций

VBA выполняет операции в соответствии с их приоритетами, что обеспечивает однозначность в трактовке значений выражений. В табл. 2.5 перечислены приоритеты выполнения операций.

Таблица 2.5. *Приоритеты выполнения операций*

Приоритет	Операция
1	Вызов функции и скобки
2	^
3	– (смена знака)
4	* и /
5	\ (деление нацело)
6	Mod (остаток от деления нацело)
7	+ и –
8	>, <, >=, <=, <> и =
9	Not
10	And
11	Or
12	Xor
13	Equ
14	Imp

Встроенные функции VBA

В VBA имеется большой набор встроенных функций и процедур, использование которых существенно упрощает программирование. Отметим, что всю необходимую информацию, связанную с использованием имеющихся функций, можно найти в справочной системе по VBA.

- Перейдите к окну редактора **Visual Basic for Applications** и выберите команду **Help | Visual Basic for Applications Help** (или нажмите клавишу <F1>).
- В окне **Справка Excel** выберите в оглавлении слева раздел **Visual Basic for Applications Language Reference | Visual Basic Language Reference | Functions**. Далее, можно увидеть весь список функций, который имеется в VBA, и примеры их использования.

Встроенные диалоговые окна

В проектах VBA часто встречаются две разновидности диалоговых окон: окна сообщений и окна ввода. Они встроены в VBA, и если их возможностей достаточно, то можно обойтись без проектирования диалоговых окон. Окно сообщений (`MsgBox`) выводит простейшие сообщения для пользователя, а окно ввода (`InputBox`) обеспечивает ввод информации.

Окно ввода

Функция `InputBox()` выводит на экран диалоговое окно, содержащее сообщение, поле ввода и две кнопки **OK** и **Cancel**. Устанавливает режим ожидания ввода

текста пользователем и нажатия кнопки, а затем, при нажатии кнопки **OK**, возвращает значение типа `String`, содержащее текст, введенный в поле ввода. При нажатии кнопки **Cancel** возвращает пустую строку (`Empty`).

```
InputBox(Prompt[, Title][, Default][, Xpos][, Ypos] [,Helpfile, Context])
```

- *Prompt* — строковое выражение, отображаемое как сообщение в диалоговом окне. Строковое выражение *Prompt* может содержать несколько строк. Для разделения строк допускается использование символа возврата каретки (`Chr(13)`), символа перевода строки (`Chr(10)`) или комбинации этих символов (`Chr(13) & Chr(10)`).
- *Title* — строковое выражение, отображаемое в строке заголовка диалогового окна. Если этот параметр опущен, то в строку заголовка помещается имя приложения.
- *Default* — строковое выражение, отображаемое в поле ввода как используемое по умолчанию, если пользователь не введет другую строку. Если этот параметр опущен, то поле ввода изображается пустым.
- *Xpos* — числовое выражение, задающее расстояние по горизонтали между левой границей диалогового окна и левым краем экрана. Если этот параметр опущен, то диалоговое окно выравнивается по центру экрана по горизонтали.
- *Ypos* — числовое выражение, задающее расстояние по вертикали между верхней границей диалогового окна и верхним краем экрана. Если этот параметр опущен, то диалоговое окно помещается по вертикали примерно на одну треть высоты экрана.
- *Helpfile* — строковое выражение, определяющее имя файла справки, содержащего справочные сведения о данном диалоговом окне. Если этот параметр указан, то необходимо указать также параметр *Context*.
- *Context* — числовое выражение, определяющее номер соответствующего раздела справочной системы. Если этот параметр указан, то необходимо указать также параметр *Helpfile*.

Например, следующий код (также файл *2-Примеры использования встроенных диалоговых окон.xlsm* на компакт-диске) отображает на экране окно ввода, показанное на рис. 2.1.

```
Sub DemoInputBox1()
   Dim n As String
   n = InputBox("Введите ваше имя", "Пример окна ввода")
   Debug.Print n
End Sub
```

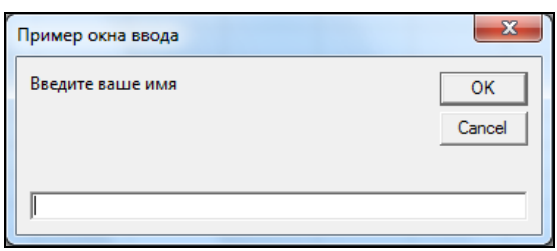


Рис. 2.1. Окно ввода

Как обработать нажатие кнопки *Cancel*?

При вводе данных при помощи функции `InputBox()` разумно в программе предусмотреть обработку события нажатия кнопки **Cancel** окна ввода. Например, в следующей простой ситуации

```
x = InputBox("Введите x", "Пример")
y = x ^ 2
```

в том случае, если пользователь нажмет кнопку **Cancel**, произойдет прерывание выполнения кода с сообщением об ошибке несовпадения типов. Этой ситуации легко избежать, добавив только одну строку кода (см. файл *2-Примеры использования встроенных диалоговых окон.xlsm* на компакт-диске), проверяющую, не введена ли в поле ввода окна ввода пустая строка (что, собственно говоря, и происходит при нажатии кнопки **Cancel**).

```
Sub DemoInputBox2()
   Dim x As String
   Dim y As Double
   x = InputBox("Введите x", "Пример")
   If x = Empty Then Exit Sub
   y = x ^ 2
   Debug.Print y
End Sub
```

Окно сообщения

Процедура `MsgBox()` выводит на экран диалоговое окно, содержащее сообщение, устанавливает режим ожидания нажатия кнопки пользователем, а затем возвращает значение типа `Integer`, указывающее, какая кнопка была нажата.

`MsgBox(`*`Prompt`*`[, `*`Buttons`*`] [, `*`Title`*`] [, `*`Helpfile`*`, `*`Context`*`])`

- *`Prompt`* — строковое выражение, отображаемое как сообщение в диалоговом окне.
- *`Buttons`* — числовое выражение, представляющее сумму значений, которые указывают число и тип отображаемых кнопок, тип используемого значка, основную кнопку и модальность окна сообщения. Значение по умолчанию этого параметра равняется 0. Значения констант, определяющих число, тип кнопок и тип используемого значка, приведены в табл. 2.6—2.8.
- *`Title`* — строковое выражение, отображаемое в строке заголовка диалогового окна. Если этот параметр опущен, то в строку заголовка помещается имя приложения.
- *`Helpfile`* — строковое выражение, определяющее имя файла справки, содержащего справочные сведения о данном диалоговом окне. Если этот параметр присутствует, необходимо указать также параметр *`Context`*.
- *`Context`* — числовое выражение, определяющее номер соответствующего раздела справочной системы. Если этот параметр присутствует, то необходимо указать также параметр *`Helpfile`*.

Таблица 2.6. *Значения параметра* `Buttons` *процедуры* `MsgBox()`*, определяющие кнопки, отображаемые в диалоговом окне*

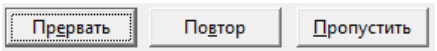
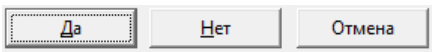
Константа	Значение	Отображаются кнопки
`vbOKOnly`	0	
`vbOKCancel`	1	
`vbAbortRetryIgnore`	2	
`vbYesNoCancel`	3	
`vbYesNo`	4	
`vbRetryCancel`	5	

Таблица 2.7. *Значения параметра* `Buttons` *процедуры* `MsgBox()`*, определяющие отображаемые в диалоговом окне информационные значки*

Константа	Значение	Значок сообщения
`vbCritical`	16	
`vbQuestion`	32	
`vbExclamation`	48	
`vbInformation`	64	

Таблица 2.8. *Значения параметра* `Buttons` *процедуры* `MsgBox()`*, определяющие основную кнопку в диалоговом окне*

Константа	Значение	Номер основной кнопки
`vbDefaultButton1`	0	1
`vbDefaultButton2`	256	2
`vbDefaultButton3`	512	3
`vbDefaultButton4`	768	4

При написании программ с откликом в зависимости от того, какая кнопка диалогового окна нажата, вместо возвращаемых значений удобнее использовать константы VBA, перечисленные в табл. 2.9, которые делают код программы более прозрачным для чтения, и к тому же их легко запомнить.

Таблица 2.9. *Константы, идентифицирующие нажатую кнопку*

Константа	Значение	Нажатая кнопка
vbOK	1	**OK**
vbCancel	2	**Отмена** (Cancel)
vbAbort	3	**Прервать** (Abort)
vbRetry	4	**Повторить** (Retry)
vbIgnore	5	**Пропустить** (Ignore)
vbYes	6	**Да** (Yes)
vbNo	7	**Нет** (No)

В файле *2-Примеры использования встроенных диалоговых окон.xlsm* на компакт-диске приведен пример использования окон сообщений: вычисляется квадрат введенного значения, и найденное значение затем последовательно отображается в пяти окнах сообщения. Первое окно является простым окном с сообщением (рис. 2.2), второе — с сообщением, выводимым в две строчки (рис. 2.3), третье — с сообщением и информационным значком (рис. 2.4), четвертое — с сообщением, информационным значком и двумя кнопками, причем кнопка **Да** установлена как кнопка, выполняемая по умолчанию (рис. 2.5), пятое — с сообщением, информационным значком и пользовательским заголовком (рис. 2.6).

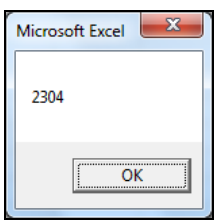


Рис. 2.2. Простое окно сообщения

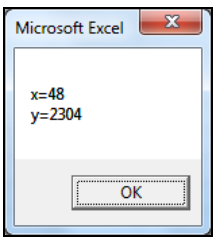


Рис. 2.3. Окно с сообщением, выводимым в две строчки

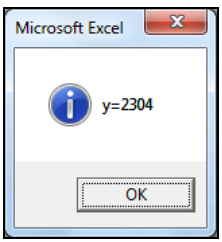


Рис. 2.4. Окно с сообщением и информационным значком

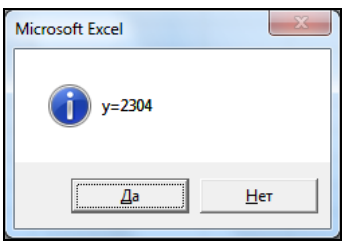


Рис. 2.5. Окно с сообщением, информационным значком и двумя кнопками

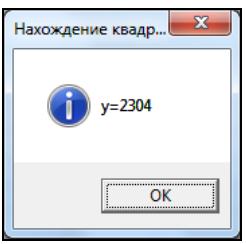


Рис. 2.6. Окно с сообщением, информационным значком и пользовательским заголовком

Определение нажатой кнопки в окне ввода

Процедура `MsgBox()` удобна для вывода той или иной информации. Однако если необходимо узнать, какой выбор сделал пользователь при нажатии отображаемых в диалоговом окне кнопок, то процедуру `MsgBox()` нужно использовать как функцию. В этом случае значение, возвращаемое `MsgBox()`, надо присваивать какой-то переменной, а ее параметры заключать в скобки. Рассмотрим пример использования функции `MsgBox()` с анализом нажатой кнопки (листинг 2.8, см. также файл *2-Примеры использования встроенных диалоговых окон.xlsm* на компакт-диске). В этом примере на экране отображается диалоговое окно с тремя кнопками (**Да**, **Нет** и **Отмена**) и информационным значком. Клавише <Enter> назначена функция кнопки **Да**. По нажатию одной из этих кнопок на экране отображается сообщение, подтверждающее нажатие.

Листинг 2.8. Пример использования окна сообщений с тремя кнопками

```
Sub DemoThreeButtonsMsgBox()
   Dim structure As Integer
   Dim btn As Integer
   structure = vbYesNoCancel + vbQuestion + vbDefaultButton1
   btn = MsgBox("Выберите <Yes>, <No> или <Cancel>", structure, "Еще пример")
   Select Case btn
     Case vbYes
        MsgBox "Выбрали <Yes>", vbInformation, "Еще пример"
     Case vbNo
        MsgBox "Выбрали <No>", vbInformation, "Еще пример"
      Case vbCancel
        MsgBox "Выбрали <Cancel>", vbInformation, "Еще пример"
   End Select
End Sub
```

Управляющие конструкции: формируем логику программы

Как и во всех других языках программирования, в VBA имеются различные управляющие конструкции, позволяющие изменять порядок выполнения программы. Если в программе нет управляющих конструкций, то происходит последовательное выполнение операторов, начиная с самого первого и кончая последним. В самых простых случаях этого бывает достаточно, однако обычно все-таки необходимо изменять порядок исполнения операторов при выполнении определенных условий, либо пропускать выполнение некоторых операторов, либо, наоборот, многократно повторять их. Отметим, что для реализации любых алгоритмов достаточно иметь два вида конструкций управления: циклы и ветвления. Рассмотрим подробнее использование операторов в языке VBA.

Оператор присваивания

Оператор присваивания присваивает значение выражения переменной, константе или свойству объекта. Оператор присваивания всегда включает знак равенства (=).

<переменная> = *<выражение>*

Оператор присваивания предписывает выполнить *<выражение>*, заданное в его правой части, и присвоить результат переменной *<переменная>*, имя которой указано в левой части. В результате, например, действия следующей пары операторов

```
x = 2
x = x + 2
```

переменной `x` будет присвоено 4.

Присваивание переменной ссылки на объект. Для присваивания переменной ссылки на объект в операторе присваивания применяется ключевое слово `Set`. В следующем примере переменной `r` присваивается ссылка на ячейку **A1**, и уже через эту переменную в ячейку **A1** вводится значение `3`.

```
Dim r As Range
Set r = Range("A1")
r.Value = 3
```

В общем случае инструкция `Set` имеет следующий синтаксис.

`Set` *<переменная>*= {[New] *<выражение>*| Nothing}

где ключевое слово `New` используется при создании нового экземпляра класса, а ключевое слово `Nothing` позволяет освободить все системные ресурсы и ресурсы памяти, выделенные для объекта, на который имелась ссылка (проще говоря, она удаляет объект из памяти).

Оператор `With`. Оператор `With` избавляет программиста от утомительной обязанности использовать большое количество повторений имени одного и того же объекта при работе с его свойствами и методами. Кроме того, он структурирует код, делая его более прозрачным:

```
With Range("A1")
    .Value = 3
    .Font.Italic = True
End With
```

Допустимо также использование вложенных операторов `With`:

```
With Range("A1")
    .Value = 3
    With .Font
        .Italic = True
        .Size = 12
        .Bold = True
        .Color = RGB(255, 30, 255)
    End With
End With
```

Ветвления

Управляющие конструкции ветвления (перехода) позволяют проверить некоторое условие и, в зависимости от результатов этой проверки, выполнить ту или иную группу операторов. Для организации ветвлений в VBA используются различные формы оператора ветвления `If` и оператор выбора `Select Case`.

Простейшая краткая форма оператора `If` применяется для проверки одного условия, а затем либо выполнения, либо пропуска одного оператора или блока из нескольких операторов. Краткая форма оператора ветвления `If` может иметь как однострочную, так и блочную форму. В одну строку краткая форма `If` может быть записана так:

```
If <условие> Then <оператор>
```

В блочной форме краткое ветвление выглядит следующим образом:

```
If <условие> Then
    <оператор1>
    <оператор2>
    ...
End If
```

В качестве условия можно использовать логическое выражение, возвращающее значение `True` или `False`, или любое арифметическое выражение. Если используется арифметическое выражение, то нулевое значение этого выражения эквивалентно логическому значению `False`, а любое ненулевое значение — `True`. В том случае, когда условие возвращает значение `False`, оператор или блок операторов, заключенных между ключевыми словами `Then` и `End If` и составляющих тело краткого оператора ветвления, не будет выполняться.

> ***Примечание***
>
> Обратите внимание, что при записи краткого оператора ветвления в одну строку ключевые слова `End If` не используются.

Полная форма оператора `If` применяется в тех случаях, когда есть два различных блока операторов, и по результатам проверки условия нужно выполнить один из них. Такая форма `If` не может записываться в одну строку и всегда имеет блочную форму записи:

```
If <условие> Then
    <блокОператоров1>
Else
    <блокОператоров2>
End If
```

Если условие истинно, выполняется первый блок операторов, заключенный между ключевыми словами `Then` и `Else`, в противном случае — второй блок, заключенный между ключевыми словами `Else` и `End If`.

> ***Совет***
>
> Для того чтобы текст процедуры был понятным и удобным для восприятия, рекомендуется делать отступы для групп операторов так, как это указано при описании их син-

таксиса. В VBA предусмотрено удобное средство изменения отступов — нажатие клавиши <Tab> увеличивает отступ вправо, нажатие комбинации клавиш <Shift>+<Tab> уменьшает этот отступ.

Иногда приходится делать выбор одного действия из целой группы действий на основе проверки нескольких различных условий. Для этого можно использовать цепочку операторов ветвления `If...Then...ElseIf`:

```
If <условие1> Then
    <блокОператоров1>
ElseIf <условие2> Then
    <блокОператоров2>
ElseIf <условие3> Then
    <блокОператоров3>
...
ElseIf <условиеN> Then
    <блокОператоровN>
Else
    <блокОператоровElse>
End If
```

Такие цепочки операторов `If...Then...ElseIf` обладают большой гибкостью и позволяют решить все проблемы, однако если выбор одной из нескольких возможностей все время основан на различных значениях одного и того же выражения, гораздо удобнее использовать специально предназначенный для этого оператор выбора `Select Case`, имеющий следующий синтаксис:

```
Select Case <проверяемоеВыражение>
    Case <списокЗначений1>
        <блокОператоров1>
    Case <списокЗначений2>
        <блокОператоров2>
    Case <списокЗначений3>
        <блокОператоров3>
    ...
    Case Else
        <блокОператоровElse>
End Select
```

Проверяемое выражение вычисляется в начале работы оператора `Select Case`. Это выражение может возвращать значение любого типа, например логическое, числовое или строковое.

Список значений представляет собой одно или несколько выражений, разделенных запятой. При выполнении оператора проверяется, соответствует ли хотя бы один из элементов этого списка значению проверяемого выражения. Элементы списка значений могут иметь одну из следующих форм:

- *<выражение>* — проверяется, совпадает ли значение проверяемого выражения со значением этого выражения;
- *<выражение1>* `To` *<выражение2>* — проверяется, находится ли значение проверяемого выражения в указанном диапазоне значений;

- `Is <логическийОператор> <выражение>` — проверяемое выражение сравнивается с указанным значением с помощью заданного логического оператора (например, условие `Is >= 10` считается выполненным, если проверяемое значение не меньше 10).

Если хотя бы один из элементов списка соответствует проверяемому выражению, то выполняется соответствующая группа операторов, и на этом выполнение оператора `Select Case` заканчивается, а остальные списки выражений не проверяются, т. е. отыскивается только первый подходящий элемент списков выражений. Если ни один из элементов всех этих списков не соответствует значению проверяемого выражения, выполняются операторы группы `Else`, если такая присутствует.

Рассмотрим простейшие примеры использования операторов ветвления.

В листинге 2.9 (см. также файл *3-Примеры использования операторов управления.xlsm* на компакт-диске) в зависимости от величины вводимого числа отображается сообщение о принадлежности числа:

- либо интервалу [0, 1];
- либо интервалу (1, 2];
- либо о не принадлежности числа этим двум интервалам.

Листинг 2.9. Определение, какому интервалу принадлежит вводимое число

```
Sub DemoElseIf()
   x = InputBox("Введите число")
   If 0 <= x And x <= 1 Then
      MsgBox "Число из интервала [0, 1]"
   ElseIf 1 < x And x <= 2 Then
      MsgBox "Число из интервала (1, 2]"
   Else
      MsgBox "Число либо отрицательное, либо больше 2"
   End If
End Sub
```

В примере, описанном в листинге 2.10 (см. также файл *3-Примеры использования операторов управления.xlsm* на компакт-диске), демонстрируется использование оператора `Case` для вывода сообщения о том, какому диапазону принадлежит вводимое целое число.

Листинг 2.10. Пример использования оператора выбора `Select Case`

```
Sub DemoSelect
   Dim x As Integer
   x = InputBox("Введите целое число")
   Select Case x
      Case 1
         MsgBox "Число равно 1"
      Case 2, 3
         MsgBox "Число равно 2 или 3"
```

```
      Case 4 To 6
         MsgBox "Число от 4 до 6"
      Case Is >= 7
         MsgBox "Число не менее 7"
   End Select
End Sub
```

Циклы

В VBA есть богатый выбор средств организации циклов, которые можно разделить на две основные группы: *циклы с условием* `Do...Loop` и *циклы с перечислением* `For...Next`.

Циклы типа `Do...Loop` используются в тех случаях, когда заранее неизвестно, сколько раз должно быть повторено выполнение блока операторов, составляющего тело цикла. Такой цикл продолжает свою работу до тех пор, пока не будет выполнено определенное условие. Существуют четыре вида циклов `Do...Loop`, которые различаются типом проверяемого условия и временем выполнения этой проверки. В табл. 2.10 приведен синтаксис этих четырех конструкций.

Таблица 2.10. *Синтаксис операторов цикла* `Do...Loop`

Конструкция	Описание

`Do While <условие>`
`  <блокОператоров>`
`Loop`
	Условие проверяется до того, как выполняется группа операторов, образующих тело цикла. Цикл продолжает свою работу, пока это условие выполняется (т. е. имеет значение `True`). Иными словами, в этой конструкции указывается условие продолжения работы цикла

`Do`
`  <блокОператоров>`
`Loop While <условие>`
	Условие проверяется после того, как операторы, составляющие тело цикла, будут выполнены хотя бы один раз. Цикл продолжает свою работу, пока это условие остается истинным. Иными словами, в этой конструкции указывается условие продолжения работы цикла

`Do Until <условие>`
`  <блокОператоров>`
`Loop`
	Условие проверяется до того, как выполняется группа операторов, образующих тело цикла. Цикл продолжает свою работу, если это условие еще не выполнено, и прекращает работу, когда оно становится истинным. Иными словами, в этой конструкции указывается условие прекращения работы цикла

`Do`
`  <блокОператоров>`
`Loop Until <условие>`
	Условие проверяется после того, как операторы, составляющие тело цикла, будут выполнены хотя бы один раз. Цикл продолжает свою работу, если это условие еще не выполнено, а когда оно станет истинным, цикл прекращает работу. Иными словами, в этой конструкции указывается условие прекращения работы цикла

Существуют также две разновидности оператора цикла с перечислением `For...Next`. Очень часто при обработке массивов, а также в тех случаях, когда требуется повторить выполнение некоторой группы операторов заданное число раз, применяется цикл `For...Next` со счетчиком. В отличие от циклов `Do...Loop`, дан-

ный тип цикла использует специальную переменную, называемую *счетчиком*, значение которой увеличивается или уменьшается на определенную величину при каждом выполнении тела цикла. Когда значение этой переменной достигает заданной величины, выполнение цикла заканчивается.

Синтаксис этого цикла выглядит следующим образом (в квадратные скобки заключены необязательные элементы синтаксической конструкции):

```
For <счетчик> = <начальноеЗначение> To <конечноеЗначение>
    [Step <приращение>]
    <блокОператоров>
Next [<счетчик>]
```

В этой конструкции цикла:

- *<приращение>* может быть как положительным, так и отрицательным числом. Если использовать отрицательное приращение, то конечное значение должно быть меньше начального значения либо равно ему для того, чтобы тело цикла выполнилось хотя бы один раз;
- после завершения работы цикла `For...Next` переменная, которая использовалась в качестве счетчика, получает значение, обязательно превосходящее конечное значение в том случае, если приращение положительно, и строго меньшее конечного значения, если приращение отрицательно;
- если начальное и конечное значения совпадают, тело цикла выполняется лишь один раз.

Рассмотрим еще одну разновидность цикла `For...Next`, часто использующуюся в VBA при обработке объектов, составляющих *массив* или *семейство* однородных объектов. В этой разновидности цикла счетчик отсутствует, а тело цикла выполняется для каждого элемента массива или семейства объектов. Синтаксис такого цикла следующий:

```
For Each <элемент> In <совокупность>
    <блокОператоров>
Next [<элемент>]
```

где *<элемент>* — это переменная, используемая для ссылки на элементы семейства объектов; *<совокупность>* — имя массива или семейства.

Выход из циклов и процедур

Обычно выполнение процедуры заканчивается после выполнения ее последнего оператора, а выполнение цикла — после нескольких выполнений тела цикла, когда достигнуто условие завершения его работы. Однако в некоторых случаях бывает нужно прекратить выполнение процедуры или цикла досрочно, избежав выполнения лишних операторов процедуры или лишних повторений цикла.

Например, если при выполнении процедуры произошла ошибка, которая делает продолжение ее работы бессмысленным, можно выполнить команду немедленного выхода из процедуры. Другой пример: если цикл `For...Next` используется для поиска нужного значения в массиве, то после того, как нужный элемент массива найден, нет смысла продолжать дальнейший перебор элементов массива. Досрочный выход из управляющей конструкции можно осуществить с помощью одного

из операторов `Exit`. Для досрочного выхода из циклов `Do...Loop` используется оператор `Exit Do`, а для выхода из циклов `For` — оператор `Exit For`. Для досрочного выхода из процедур и функций применяются операторы `Exit Sub` и `Exit Function`, соответственно.

Следует отметить, что хотя использование оператора `Exit` может быть вполне оправданным, необходимо избегать излишнего употребления этого оператора, прибегая к нему только в крайних случаях. Частое употребление данного оператора затрудняет понимание написанного текста программы и его отладку.

Примеры использования операторов цикла

Оператор *For...Next*

Оператор `For...Next` в листинге 2.11 (см. также файл *3-Примеры использования операторов управления.xlsm* на компакт-диске) используется для нахождения в цикле суммы элементов массива.

Листинг 2.11. Суммирование элементов массива

```
Sub DemoFor
   Dim A As Variant
   A = Array(1, 4, 12, 23, 34, 3, 23)
   s = 0
   For i = LBound(A) To UBound(A)
      s = s + A(i)
   Next
   Msgbox s
End Sub
```

В следующем примере (листинг 2.12, файл *3-Примеры использования операторов управления.xlsm* на компакт-диске) находится произведение первых *n* натуральных чисел (*n*-факториал).

Листинг 2.12. Нахождение произведения первых *n* натуральных чисел

```
Sub Factor()
   n = 20
   Fact = 1
   For i = 1 To n
      Fact = Fact * i
   Next i
   MsgBox Format(Fact, "############")
' Будет выведено 2432902008176640000
End Sub
```

В файле *3-Примеры использования операторов управления.xlsm* на компакт-диске существует пример, в котором находится сумма значений из выделенного

диапазона. При этом используются вложенные операторы циклов. Результат выводится в ячейки строки, находящейся непосредственно под выделенным диапазоном. А именно в ячейку под первым столбцом выделенного диапазона вводится строка `"Сумма"`, а в соседнюю с ней справа ячейку выводится найденное значение.

Оператор *For Each*

Оператор `For Each` можно использовать для суммирования элементов массива. Как это делается, показано в листинге 2.13 (см. также файл *3-Примеры использования операторов управления.xlsm* на компакт-диске).

Листинг 2.13. Суммирование элементов массива с помощью оператора `For...Each`

```
Sub DemoForEach
   Dim A As Variant, s As Double
   A = Array(1, 4, 12, 23, 34, 3, 23)
   s = 0
   For Each b In A
      s = s + b
   Next
   Msgbox s
End Sub
```

Оператор `For Each` можно также использовать при переборе ячеек диапазона и, в частности, при нахождении суммы значений, введенных в ячейки диапазона. Как это делается, продемонстрировано в следующем коде (листинг 2.14, файл *3-Примеры использования операторов управления.xlsm* на компакт-диске), в котором производится суммирование значений из выделенного диапазона.

Листинг 2.14. Суммирование значений из диапазона ячеек с помощью оператора `For Each`

```
Sub DemoSumSelection()
   Dim c As Range
   Dim s As Double
   s = 0
   For Each c In Selection.Cells
      s = s + c.Value
   Next
   MsgBox s
End Sub
```

В следующем примере (листинг 2.15, файл *3-Примеры использования операторов управления.xlsm* на компакт-диске) цикл `For Each` используется для изменения цвета ячеек: все ячейки диапазона **A1:C4** с положительными значениями окрашиваются в синий цвет, а с неположительными — в красный.

Листинг 2.15. Изменение цвета ячеек

```
Sub DemoChangeColor()
   Dim c As Range
   For Each c In Range("A1:C4").Cells
      With c
         If .Value <= 0 Then
            .Interior.ColorIndex = 3
         Else
            .Interior.ColorIndex = 5
         End If
      End With
   Next
End Sub
```

Оператор `For Each` используется и при переборе элементов семейства. Например, следующий код (листинг 2.16, файл *3-Примеры использования операторов управления.xlsm* на компакт-диске) демонстрирует применение оператора цикла `For Each` для работы с семейством рабочих листов. Оператор удаляет из рабочей книги рабочий лист **Тест**, если он, конечно, имеется в рабочей книге.

Листинг 2.16. Работа с семейством рабочих листов

```
Sub DemoDeleteSheet()
   Dim ws As Worksheet
   For Each ws In Worksheets
      If ws.Name = "Тест" Then
        ws.Delete
      End If
   Next
End Sub
```

Оператор *While*

Оператор `While`, как указывалось ранее, в отличие от `For`, повторяется не заданное число раз, а пока выполняется условие. В следующем примере (листинг 2.17, см. также файл *3-Примеры использования операторов управления.xlsm* на компакт-диске) имитируется бросание игральной кости до тех пор, пока не выпадет шесть очков. При выпадении шести очков игра заканчивается, и отображается сообщение с указанием, на каком броске она закончилась.

Листинг 2.17. Бросание игральной кости

```
Sub DemoWhile()
   Dim attempt As Integer
   Dim score As Integer
   Randomize
```

```
   score = Int(6 * Rnd()) + 1
   attempt = 1
   While score < 6
      attempt = attempt + 1
      score = Int(6 * Rnd()) + 1
   Wend
   MsgBox "Победили на попытке: " & attempt
End Sub
```

Использование в качестве условия оператора `While` значения `True` приводит к созданию бесконечного цикла. Для прерывания его выполнения в теле цикла надо расположить инструкцию, обеспечивающую выход из него при соблюдении некоторого условия.

```
While True
   <блокОператоров>
Wend
```

Оператор *Do*

Примером использования оператора цикла `Do...Until` может быть код из листинга 2.18 (файл *3-Примеры использования операторов управления.xlsm* на компакт-диске), который обеспечивает повторение цикла до тех пор, пока в поле ввода диалогового окна не будет введен пароль — слово `time`.

Листинг 2.18. Пример использования оператора `Do`

```
Sub DemoPassword()
   Dim ps as String
   Do
      ps = InputBox("Введите пароль")
   Loop Until ps = "time"
End Sub
```

Для оператора цикла `Do While` приведем пример (листинг 2.19, файл *3-Примеры использования операторов управления.xlsm* на компакт-диске)), который обеспечивает последовательное отображение имен всех jpg-файлов корневого каталога диска D:. Здесь используется функция работы с файлами `Dir()`, которая возвращает имя первого подходящего файла. Если подходящего файла нет, то функция `Dir()` возвращает пустую строку. При повторном вызове функции `Dir()` ее параметр с маской, по которой ищется подходящий файл, опускается.

Листинг 2.19. Последовательное отображение имен файлов с расширением jpg

```
Sub DemoShowFiles()
   Dim f As String
   Dim res As String
   f = Dir("D:\*.jpg")
```

```
   res = f & vbCr
   Do While Len(f)
      f = Dir
      res = res & f & vbCr
   Loop
   MsgBox res
End Sub
```

Альтернативный выход из цикла

Примером использования альтернативного выхода `Exit Do` из цикла может быть следующий код (листинг 2.20, файл *3-Примеры использования операторов управления.xlsm* на компакт-диске), в котором суммируются вводимые числа. Цикл завершается в случае ввода любого строкового выражения.

Листинг 2.20. Суммирование всех вводимых чисел

```
Sub DemoExitDo()
  Dim s As Double, x As Variant
  s = 0
  Do
      x = InputBox("Введите число")
      If Not IsNumeric(x) Then Exit Do
      s = s + x
  Loop
  MsgBox s
End Sub
```

Создание бесконечного цикла оператором *Do*

Оператор `Do` без условий создает бесконечный цикл. Такие циклы применяются при программировании различных фоновых процессов типа печати документа.

```
Do
Loop
```

Оператор безусловного перехода *GoTo*

Оператор безусловного перехода задает переход на указанную строку внутри процедуры. Обязательный параметр *`line`* может быть любой меткой строки или номером строки.

```
GoTo line
```

Для использования оператора безусловного перехода надо какой-то строке присвоить метку. Метка должна начинаться с буквы и заканчиваться двоеточием. В качестве примера использования оператора безусловного перехода рассмотрим игру, в которой игроку даны десять попыток броска игральной кости. В случае, если при какой-то из этих попыток выпадает шесть очков, игрок выигрывает, и игра заканчивается (листинг 2.21, файл *3-Примеры использования операторов управления.xlsm* на компакт-диске).

Листинг 2.21. Бросание игральной кости

```
Sub DemoGoTo()
   Dim i As Integer
   Dim score As Integer
   Randomize
   For i = 1 To 10
      score = Int(6 * Rnd()) + 1
      If score = 6 Then GoTo lblMessage
   Next
   GoTo lblEnd
lblMessage:
   MsgBox "Выиграли при броске " & i
lblEnd:
End Sub
```

Отметим, что в данном примере использование оператора `GoTo` представляется слишком сложным. Применение цикла `Do` может существенно упростить и сократить код.

Процедуры: знакомимся с деталями

Процедура является самостоятельной частью кода, которая имеет имя и может содержать параметры, выполнять последовательность инструкций и изменять значения своих параметров.

```
[Private | Public | Friend] [Static] Sub <имяПроцедуры> [(<аргументы>)]
   <инструкции>
   [Exit Sub]
   <инструкции>
End Sub
```

- `Private` — ключевое слово, указывающее на то, что процедура является *закрытой*, и областью ее видимости является модуль.
- `Public` — ключевое слово, указывающее на то, что процедура является *открытой* и доступна для всех других процедур во всех модулях.
- `Friend` — ключевое слово, используемое только в модуле класса и указывающее на то, что процедура является *дружественной*, и областью ее видимости является проект.
- `Static` — ключевое слово, указывающее на то, что локальные переменные процедуры сохраняются в промежутках времени между вызовами этой процедуры.
- *<имяПроцедуры>* — имя процедуры, удовлетворяющее стандартным правилам именования переменных.
- *<аргументы>* — список параметров, значения которых передаются в процедуру или возвращаются из процедуры при ее вызове. Разделителем в списке параметров является запятая.

- *<инструкции>* — любая группа инструкций (как правило, совокупность операторов), выполняемых в процедуре `Sub`.
- `Exit Sub` — оператор, приводящий к немедленному выходу из процедуры.

В следующих примерах (листинги 2.22—2.24) имеются две процедуры `DemoSub1` и `DemoSub2`, расположенные в разных модулях, причем процедура `DemoSub2` объявлена как закрытая, и поэтому вызов процедуры `DemoSub1` приводит к генерации ошибки.

Листинг 2.22. Процедура `DemoSub1` не видит процедуру `DemoSub2`. Один стандартный модуль

```
Sub DemoSub1()
   DemoSub2
End Sub
```

Листинг 2.23. Процедура `DemoSub1` не видит процедуру `DemoSub2`. Другой стандартный модуль

```
Private Sub DemoSub2()
   Beep
   MsgBox "Вызов принят"
End Sub
```

Для того чтобы избежать этой ошибки, процедура `DemoSub2` должна быть объявлена либо как открытая, либо без спецификации области ее видимости.

Листинг 2.24. Процедура `DemoSub1` теперь будет видеть процедуру `DemoSub2`. Другой стандартный модуль

```
Public Sub DemoSub2()
   Beep
   MsgBox "Вызов принят"
End Sub
```

Создание пользовательских функций

Кроме процедуры `Sub`, в VBA имеется процедура `Function` или просто функция.

```
[Public | Private | Friend] [Static] Function <имяФункции> _
     [(<аргументы>)] [As Тип]
   <инструкции>
   <имяФункции> = выражение
   [Exit Function]
   <инструкции>
   <имяФункции> = выражение
End Function
```

Синтаксис процедуры `Function` содержит те же элементы, что и процедуры `Sub`. Инструкция `Exit Function` приводит к немедленному выходу из процедуры `Function`. Подобно процедуре `Sub`, функция является самостоятельной процедурой, которая может получать значения параметров, выполнять последовательность инструкций и изменять значения своих параметров. Однако, в отличие от процедуры `Sub`, когда требуется использовать возвращаемое функцией значение, процедура `Function` может применяться в правой части выражения, как и любая другая встроенная функция, например `Cos`. Процедура `Function` вызывается в выражении по своему имени, за которым следует список параметров, заключенный в скобки. Для возврата значения из функции следует присвоить значение имени функции. Любое число таких инструкций присваивания может находиться в любом месте процедуры.

В коде из листинга 2.25 (файл *4-Примеры пользовательских функций.xlsm* на компакт-диске) представлен пример функции `F`, которая находит сумму двух значений.

Листинг 2.25. Пример функции

```
Sub DemoFun()
   MsgBox F(1, 3)
End Sub
Function F(x As Double, y As Double) As Double
   F = x + y
End Function
```

В листинге 2.26 (файл *4-Примеры пользовательских функций.xlsm* на компакт-диске) вычисляется значение функции function1 = $x^2 + \sin(x + z)$ для данных x и z, которые можно ввести в ячейки на рабочем листе.

Листинг 2.26. Пример функции

```
Function function1(x As Double, z As Double) As Double
   function1 = x ^ 2 + Sin(x + z)
End Function
```

Заметим, что созданные вами пользовательские функции становятся доступными в списке имеющихся функций при вводе формул в ячейку рабочего листа Excel. Для того чтобы воспользоваться функцией, которую вы создали, перейдите на вкладку **Формулы** ленты и в группе команд **Библиотека функций** нажмите кнопку **Вставить функцию**. В открывшемся окне **Мастер функций** (рис. 2.7) выберите **Определенные пользователем** из списка **Категория** и далее в поле **Выберите функцию** укажите необходимую вам созданную функцию.

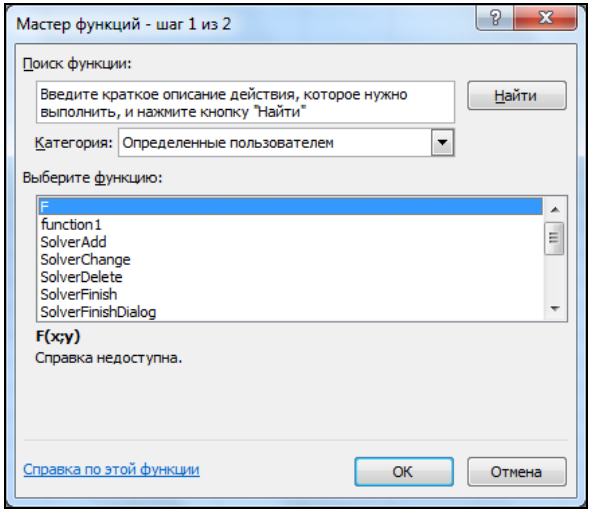


Рис. 2.7. Окно **Мастер функций**

Список параметров процедуры

Список параметров процедуры *<аргументы>* имеет следующий синтаксис:

```
[Optional] [ByVal | ByRef] [ParamArray] имяПеременной[()] _
    [As Тип][= значениеПоУмолчанию]
```

- `Optional` — ключевое слово, указывающее на то, что параметр не является обязательным. При использовании этого элемента все последующие параметры, которые содержит список *<аргументы>*, также должны быть необязательными, и их надо описать с помощью ключевого слова `Optional`. Все параметры, описанные как `Optional`, должны иметь тип `Variant`. Не допускается использование ключевого слова `Optional` для любого из параметров, если указано ключевое слово `ParamArray`.
- `ByVal` — ключевое слово, указывающее на то, что этот параметр передается по значению.
- `ByRef` — ключевое слово, указывающее на то, что этот параметр передается по ссылке. Описание `ByRef` используется в VBA по умолчанию.
- `ParamArray` — ключевое слово, которое используется только в качестве последнего элемента в списке *<аргументы>* для указания, что конечным параметром является описанный как `Optional` массив значений типа `Variant`. Ключевое слово `ParamArray` позволяет задавать произвольное количество параметров. Оно не может быть использовано с ключевыми словами `ByVal`, `ByRef` или `Optional`.
- *имяПеременной* — имя переменной, удовлетворяющее стандартным правилам именования переменных.
- *Тип* — тип значений параметра, переданного в процедуру. Допустимые значения: `Byte`, `Boolean`, `Integer`, `Long`, `Currency`, `Single`, `Double`, `Date`, `String` (только строки переменной длины), `Object`, `Variant`. Если отсутствует ключевое

слово `Optional`, может быть также указан определяемый пользователем тип или объектный тип.

- *значениеПоУмолчанию* — задает значение, которое параметр принимает по умолчанию. Используется только вместе с параметром `Optional`. Если указан тип `Object`, единственным значением по умолчанию может быть значение `Nothing`.

Организация программы на языке VBA

Программа на языке VBA состоит из одного или нескольких модулей. Обычно текст программы в модуле начинается с директив, которые управляют описанием переменных, способом сравнения строк и т. д. Затем идет объявление переменных и констант уровня модуля или проекта, т. е. переменных и констант, которые можно использовать во всех процедурах либо модуля, либо проекта. Далее располагается код процедур `Sub` и `Function`, составляющих саму программу. Приведем один пример организации модуля (листинг 2.27, файл *5-Пример организации модуля.xlsm* на компакт-диске).

Листинг 2.27. Пример организации модуля

```
Option Base 1
Option Explicit
Private Const Pi = 3.14159265358979
Private Out(2) As Double
Private Function CircleLength(r As Double) As Double
   CircleLength = 2 * Pi * r
End Function
Private Function AreaDisc(r As Double) As Double
   AreaDisc = 4 * Pi * r ^ 2
End Function
Private Sub JointResult(r As Double)
   Out(1) = CircleLength(r)
   Out(2) = AreaDisc(r)
   MsgBox Out(1) & vbCr & Out(2)
End Sub
Private Sub ShowResults()
   JointResult 1
End Sub
```

Примечание

Для тестирования программы, приведенной в листинге 2.29, не забудьте предварительно установить параметр **Require Variable Declaration** (Явное описание переменных) на вкладке **Editor** диалогового окна **Options** редактора VBA (для перехода к окну **Options** воспользуйтесь командой **Tools | Options** в интегрированной среде разработки приложений редактора Visual Basic).

Вызов процедуры и передача значений параметров

Вызов процедуры `Sub` из другой процедуры можно произвести несколькими способами.

- Первый способ вызова процедуры `Sub`:

  Имя <аргументы>

  Здесь:

  - *Имя* — имя вызываемой процедуры;
  - *<аргументы>* — список фактических параметров, т. е. список параметров, передаваемых процедуре. Он должен соответствовать по количеству и типу списку параметров, заданному в процедуре при ее определении.
- Второй способ вызова процедуры `Sub` — с помощью инструкции `Call`:

  `Call` *Имя* (*аргументы*)

  Обратите внимание, что в этом случае список фактических параметров заключается в скобки. В первом способе скобки не использовались.

Применяя именованные параметры, VBA позволяет вводить фактические параметры в любом порядке и опускать необязательные (`Optional`). При этом после имени параметра ставятся двоеточие и знак равенства, после которого помещается значение параметра (фактический параметр).

Приводимый в листинге 2.28 код показывает основные способы передачи параметров на примере процедуры `ShowResults` из предыдущего раздела (см. листинг 2.27).

Листинг 2.28. Основные способы передачи параметров на примере процедуры `ShowResults`

```
Private Sub ShowResults()
   JointResult r:=1
End Sub
Private Sub ShowResults1()
   Dim Rs As Double
   Rs = 2
   JointResult Rs
  End Sub
Private Sub ShowResults2()
   Call JointResult(4)
End Sub
```

Процедура с необязательными параметрами

Приведем пример функции с необязательными параметрами (листинг 2.29, файл *6-Примеры процедур.xlsm* на компакт-диске). Функция `UserName()` возвращает строку, состоящую из имени и фамилии, указанных в качестве значений ее параметров. Если какой-то параметр опущен, то она возвращает неполное имя. При работе с необязательными параметрами необходимо использовать функцию `IsMissing()`, возвращающую значение `True`, если соответствующий параметр не был передан в процедуру, и `False` — в противном случае.

Листинг 2.29. Процедура `Function` с необязательными параметрами

```
Function UserName(Optional LastName As String, _
                  Optional FirstName As String) As String
   If Not (IsMissing(LastName)) And Not (IsMissing(FirstName)) Then
      UserName = LastName & Space(1) & FirstName
   ElseIf IsMissing(LastName) And Not (IsMissing(FirstName)) Then
      UserName = FirstName
   ElseIf Not IsMissing(LastName) And IsMissing(FirstName) Then
      UserName = LastName
   End If
End Function

Sub DemoUserName()
    MsgBox UserName(LastName:="Bond", FirstName:="James")
    '  Bond James
    MsgBox UserName("James", "Bond")
    '  James Bond
    MsgBox UserName("Bond")
    '  Bond
    MsgBox UserName(, "James")
    '  James
    MsgBox UserName()
    ' Nothing
End Sub
```

Специфицирование значений по умолчанию необязательным параметром

Для необязательного параметра можно специфицировать значение по умолчанию. В следующем примере (листинг 2.30, файл *6-Примеры процедур.xlsm* на компакт-диске) устанавливаются применяемые по умолчанию интенсивности красной, зеленой и синей составляющих цвета ячеек выделенного диапазона.

Листинг 2.30. Специфицирование значения по умолчанию необязательным параметром

```
Sub Color(Optional Red As Integer = 0, Optional Green As Integer = 255, _
         Optional Blue As Integer = 0)
   Selection.Interior.Color = RGB(Red, Green, Blue)
End Sub
Sub DemoDefaultValues()
   Color Blue:=175
End Sub
```

Использование неопределенного количества параметров

Как правило, количество передаваемых в процедуру параметров совпадает с количеством определенных у этой процедуры параметров. Однако ключевое слово `ParramArray` предоставляет возможность ввода в процедуру произвольного, заранее не указанного числа параметров (например, как это происходит в функции `СУММ()` рабочего листа в MS Excel).

В качестве примера приведем процедуру `Function`, которая из строк, используемых в качестве значений параметров, образует одну строку (листинг 2.31. файл *6-Примеры процедур.xlsm* на компакт-диске).

Листинг 2.31. Пример процедуры с неопределенным числом параметров

```
Function PutTogether(FirstWord As String, _
                     ParamArray Words() As Variant) As String
   Dim s As String
   Dim a As Variant
   s = FirstWord
   For Each a In Words
      s = s & a
   Next a
   PutTogether = s
End Function
Sub DemoPutTogether()
Dim a As String, b As String, c As String
    a = "Капля "  :  b = "камень "  :  c = "точит"
    MsgBox PutTogether("Пословица: ", a, b, c)
' Отобразится: Пословица: Капля камень точит
    MsgBox PutTogether("Поговорка: ", "Упорство ", "и труд", _
                       " все ", "перетрут")
' Отобразится: Поговорка: Упорство и труд все перетрут
End Sub
```

Использование массива в качестве параметра процедуры

В VBA допустимо использование массива в качестве параметра процедуры. В этом случае массив, используемый в качестве параметра, при описании процедуры надо объявить как динамический. В следующем примере (листинг 2.32, файл *6-Примеры процедур.xlsm* на компакт-диске) процедура `SumM` возвращает сумму элементов массива, который является ее первым параметром.

Листинг 2.32. Использование массива в качестве параметра процедуры

```
Option Base 1
Sub SumM(ByRef A() As Double, ByRef s As Double)
   Dim x As Variant
   s = 0
   For Each x In A
        s = s + x
   Next
End Sub
Sub DemoSumM()
   Dim B(2, 2) As Double
   Dim s As Double
   B(1, 1) = 1  :  B(1, 2) = 5
   B(2, 1) = 4  :  B(2, 2) = 5
   SumM B, s
   MsgBox s
' Результат: 15
End Sub
```

Передача параметров по ссылке и значению

При вызове процедуры вы передаете в нее некоторые параметры. Внутри процедуры этим параметрам могут быть присвоены какие-то значения, которые сохранятся в них после выхода из процедуры. Таким образом, по умолчанию при передаче переменных в качестве параметров, в процедуру передаются физические адреса переменных. Поэтому внутри процедуры может быть модифицировано их содержание. Для явного указания передачи параметров в процедуру по ссылке используется ключевое слово `ByRef`. Другим способом передачи параметров в процедуру является передача их по значению. При этом способе передачи параметра в процедуру попадает не сама переменная, а ее значение. Передача параметра по значению задается ключевым словом `ByVal`. В следующем примере (листинг 2.33, файл *6-Примеры процедур.xlsm* на компакт-диске) показано отличие передачи параметра по ссылке от передачи параметра по значению.

Листинг 2.33. Передача параметров по ссылке и значению

```
Sub DemoByValByRef(ByVal a, b, ByRef c)
' a передается по значению, по умолчанию b передается по ссылке
' c передается по ссылке
   a = a + 1
   b = b + a
   c = c + a
End Sub

Sub Test()
```

```
   a = 1  :  b = 10  :  c = 100
   DemoByValByRef a, b, c
   MsgBox a
' Отобразится 1
   MsgBox b
' Отобразится 12
   MsgBox c
' Отобразится 102
End Sub
```

Рекурсивные процедуры

В VBA возможно создание рекурсивных процедур, т. е. процедур, вызывающих самих себя. Классическим примером рекурсивной процедуры является процедура, возвращающая очередной член последовательности чисел Фибоначчи (листинг 2.34, *а* и *б*, файл *6-Примеры процедур.xlsm* на компакт-диске). Два первых члена ряда Фибоначчи равны 1, а каждый его последующий член представляет собой сумму двух предыдущих. Таким образом, *n*-е число Фибоначчи $Fi(n)$ определяется следующим соотношением:

$$Fi(n) = Fi(n-1) + Fi(n-2),$$

где $Fi(1) = Fi(2) = 1.$

Листинг 2.34, *а*. Нахождение *n*-го числа Фибоначчи

```
Function Fi(n As Long) As Long
   If n = 1 Or n = 2 Then
      Fi = 1
   Else
      Fi = Fi(n — 1) + Fi(n — 2)
   End If
End Function
```

Листинг 2.34, *б*. Нахождение числа Фибоначчи без рекурсии

```
Sub Fibonacci()
   Dim n As Long, i As Long, s As Long
   Dim f1 As Long, f2 As Long
   n = 30
   f1 = 1  :  f2 = 1  :  s = 1
   For i = 3 To n
      S = f1 + f2
      f1 = f2
      f2 = s
   Next
   MsgBox s
End Sub
```

Примечание

Несмотря на элегантность рекурсивных процедур, применять их надо с осторожностью, т. к. неаккуратное использование может привести к проблемам с памятью — многократный вызов такой процедуры быстро исчерпывает стековую память. Кроме того, программы с рекурсией выполняются существенно дольше, чем программы, использующие циклы, при решении тех же задач. С другой стороны, следует помнить также о том, что и многие вычислительные задачи также приводят к переполнению стековой памяти.

Другим классическим примером использования рекурсивных функций является нахождение наибольшего общего делителя двух целых чисел по алгоритму Евклида. Наибольший общий делитель (НОД) двух целых чисел — это наибольшее целое, на которое делятся оба числа. Например:

- `НОД(10, 14) = 2`;
- `НОД(15, 31) = 1`.

Алгоритм Евклида состоит из следующих шагов:

1. Если `a` делится на `b`, то `НОД(a, b) = b`.
2. В противном случае `НОД(a, b) = НОД(b, a Mod b)`.

Приводимая в файле *6-Примеры процедур.xlsm* на компакт-диске рекурсивная функция программирует алгоритм Евклида.

Фракталы

Еще одним интересным примером рекурсивных процедур является построение фракталов. Фракталы состоят из последовательности геометрических объектов, причем первоначальный объект рисуется в большом масштабе. В последующем к этому объекту добавляются его уменьшенные копии. При этом копии могут иметь как ту же самую ориентацию, что и первоначальный объект, так и измененную ориентацию. Конечно, в VBA, в отличие от Visual Basic, отсутствуют средства графических построений непосредственно в форме. Зато, благодаря семейству объектов `Shapes`, они в избытке имеются для создания графики на рабочем листе. В приводимом далее примере (файл *7-Пример построения фракталов на основе прямоугольников.xlsm* на компакт-диске) мы разместим на рабочем листе геометрический узор при помощи фракталов. Данный фрактал имеет простую структуру. Первоначально рисуется большой квадрат, а потом к его углам пристраиваются его уменьшенные копии и так до тех пор, пока пристраиваемые квадраты не станут достаточно маленькими (рис. 2.8). Для того чтобы увидеть всю картинку на рабочем листе, уменьшите масштаб при помощи соответствующей линейки **Масштаб**, находящейся в правом нижнем углу окна Microsoft Excel.

Приведем еще один пример построения фракталов. В нем фракталы строятся на основе правильных многоугольников, в частности, на рис. 2.9 показан результат действия программы при построении фракталов на основе пятиугольных звезд (файл *8-Пример построения фракталов на основе пятиконечных звезд.xlsm* на компакт-диске). В этой программе имеется одна особенность, на которую стоит обратить внимание. А именно на применение пользовательского типа для работы с точками плоскости. Пользовательский тип в данном случае обеспечивает дополнительную прозрачность кода.

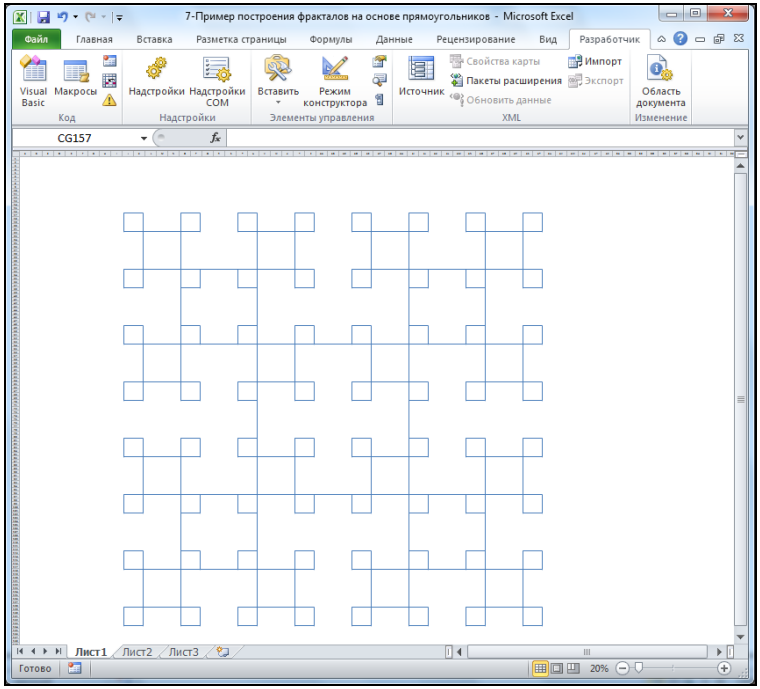


Рис. 2.8. Пример построения фракталов на рабочем листе

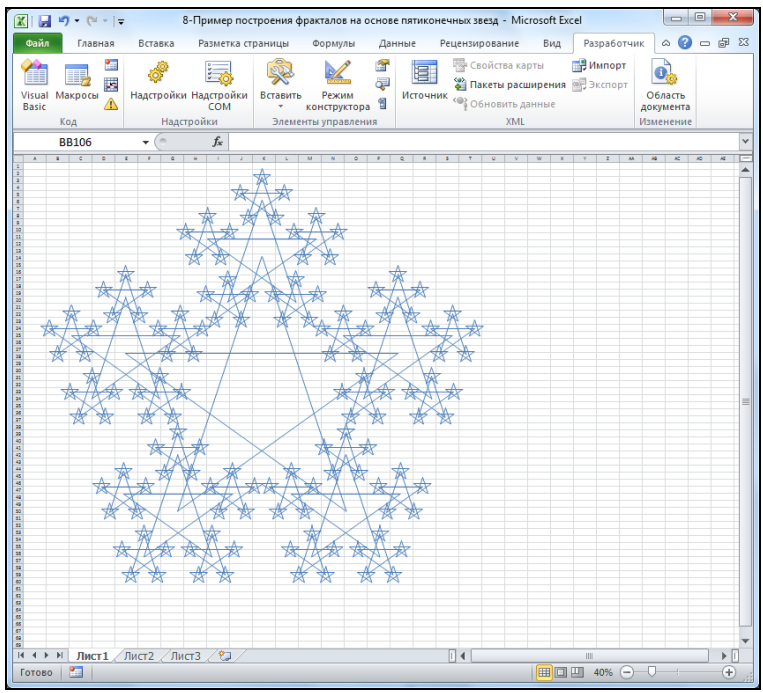


Рис. 2.9. Пример построения фракталов на основе звезд

Создаем классы, объекты и семейства

В VBA наряду с огромным числом встроенных объектов (`Application`, `Workbook`, `Worksheet`, `Range` и т. д.) предусмотрена возможность создания пользовательских объектов. Применение пользовательских объектов позволяет сократить текст программ, делать их более прозрачными и понятными. Пользовательские объекты в VBA обладают свойствами, полями, методами, могут прослушивать события, но, к сожалению, не обладают привычным для объектно-ориентированных языков механизмом наследования.

Объекты являются представителями или экземплярами классов. Метод, активизируемый объектом в ответ на сообщение, определяется классом, к которому принадлежит получатель сообщения. Все объекты одного и того же класса используют одинаковые методы в ответ на одинаковые сообщения. Членами класса являются поля, свойства, методы и события. Рассмотрим их подробнее.

- *Полем* называется переменная, содержащая некоторое значение. Таким образом, поле предоставляет информацию об объекте. Например, если имеется объект `Car` (автомобиль), то его полем может быть поле `Cylinders` (цилиндры), хранящее информацию о числе цилиндров в двигателе автомобиля.
- *Метод* — это код, программирующий некоторые действия, которые должен совершить объект. Например, объект `Car` может быть перекрашен методом `Repaint`.
- *Свойство* играет ту же роль, что и поле, а именно предоставляет информацию об объекте, но при его создании используются специальные процедуры `Property`, которые предоставляют больше возможностей как по установке значения, так и его возвращения. Свойство позволяет отделить данные от объекта, сделав их более устойчивыми.
- *Событие* позволяет объектам оповещать друг друга о произошедшем изменении в ситуации. События часто используются в графических интерфейсах для оповещения, например, о нажатии пользователем кнопки, но они также хорошо подходят и для любого другого вида оповещений, например, о получении электронной почты.

Объявление класса

Классы конструируются в модулях классов, которые создаются выбором команды **Insert | Class Module** в редакторе VBA. При создании классов надо предусмотреть его инициализацию, описание свойств и методов, которыми будет наделен объект.

Опишем процесс создания класса в виде такой последовательности шагов:

1. Выберите команду **Insert | Class Module**. Откроется окно нового модуля класса.
2. Нажмите клавишу <F4> и в появившемся окне **Properties** установите значение свойства `Name` равным имени класса. Имя модуля класса совпадает с именем класса.
3. Класс инициализируется при помощи необязательной для объявления процедуры `Class_Initialize`. В ней можно задать значения, назначаемые полям при конструировании экземпляра класса.
4. Допустимо также объявление процедуры `Class_Terminate` для описания процесса удаления объекта из памяти по завершении работы с ним.

В качестве примера создадим класс `Point`, моделирующий точку плоскости (листинг 2.35, файл *9-Пример объявления и создания экземпляра класса.xlsm* на компакт-диске). В этом классе имеются только два поля: `x` и `y`, которые задают координаты точки. Модификатор доступа `Public` устанавливает, что поля доступны для всех внешних кодов и могут быть опрошены и изменены любым из них.

Листинг 2.35. Модуль класса `Point`

```
Public x As Integer ' Поле x
Public y As Integer ' Поле y
```

Создание экземпляра класса

Для того чтобы использовать созданный класс, нужно иметь возможность получения *экземпляра* этого класса. Экземпляр — это объект, имеющий тип данного класса. Для любого созданного класса можно получить экземпляры так же, как для любого другого типа данных, но при этом при объявлении объекта надо указать ключевое слово `New`. После создания экземпляра класса у него можно считывать или устанавливать значения полей, свойств, применять к нему методы. Например, в следующем коде (листинг 2.36, файл *9-Пример объявления и создания экземпляра класса.xlsm* на компакт-диске) создается экземпляр класса `Point`, у которого устанавливаются значения полей `x` и `y`. Кроме того, некоторая переменная может быть сначала объявлена как объектная (в данном случае типа `Point`), а затем при помощи оператора присваивания `Set` ей уже может быть назначено значение. В окне **Immediate** (рис. 2.10) можно увидеть созданный экземпляр класса `Point`, а также значение объектной переменной.

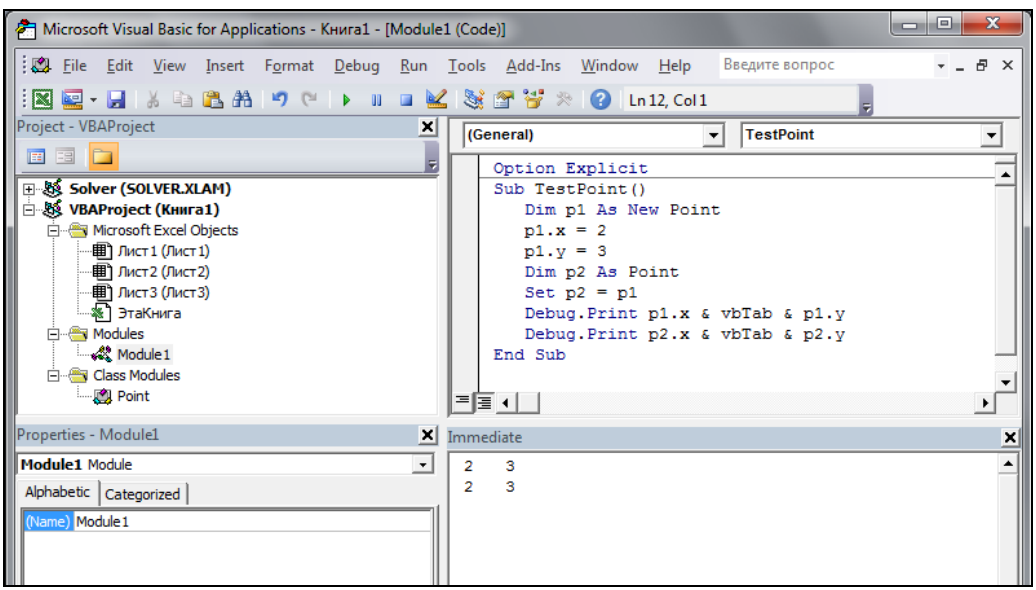


Рис. 2.10. Редактор VBA с отображаемым окном **Immediate**

Листинг 2.36. Создание экземпляра класса. Стандартный модуль

```
Sub TestPoint()
   Dim p1 As New Point
   p1.x = 2   :   p1.y = 3
   Dim p2 As Point
   Set p2 = p1
   Debug.Print p1.x & vbTab & p1.y
   Debug.Print p2.x & vbTab & p2.y
End Sub
```

Инициализация значений полей

Начальные значения полей экземпляра класса можно устанавливать в зависимости от бизнес-логики проекта. Для этого достаточно в процедуре обработки события `Initialize` модуля класса полям присвоить требуемые значения. Например, следующая модификация класса `Point` (листинг 2.37, *а* и *б*) обеспечит то, что все создаваемые экземпляры этого класса являются точками (1, 1), а не (0, 0), как это было раньше.

Листинг 2.37, *а*. Модуль класса `Point`

```
Public x As Integer
Public y As Integer
Private Sub Class_Initialize()
   x = 1  :  y = 1
End Sub
```

Листинг 2.37, *б*. Создание экземпляра класса. Стандартный модуль

```
Sub TestInitPoint()
   Dim p As New Point
   Debug.Print p.x & vbTab & p.y
End Sub
```

Ключевое слово *Me*

Ключевое слово `Me` возвращает экземпляр данного класса, через который можно обращаться к полям. Поэтому код инициализации значений полей класса `Point` можно оформить следующим равносильным способом (листинг 2.38).

Листинг 2.38. Класс `Point` с ключевым словом `Me`

```
Public x As Integer
Public y As Integer
Private Sub Class_Initialize()
   Me.x = 1
   Me.y = 1
End Sub
```

Ключевое слово *Nothing* и удаление объекта из памяти

Для удаления ссылки из объектной переменной используется ключевое слово `Nothing`. Например:

```
Dim p As New Point
p.x = 1
Set p = Nothing
```

Если на используемый объект не ссылаются другие переменные, то Windows удаляет его из памяти. Обратите внимание на то, что значение `Nothing` должно присваиваться посредством оператора `Set`, как при любом присваивании объектов.

Методы

Методы класса обычно содержат код, который анализирует состояние объекта и изменяет его. Например, классы часто обладают определенными функциями, которые не сводятся к простому чтению или установке значений некоторых параметров, а требуют проведения определенных вычислений. В этом случае на помощь и приходят методы. По своей сути методы представляют собой процедуры. В том случае, если они возвращают или устанавливают значения, их удобнее программировать в виде функций.

Вызов метода представляет собой операцию, выполняемую с объектом посредством ссылки на него, затем указания точки, после которой идет имя метода, за которым в скобках перечисляется список фактических параметров.

Объект`.`*Метод*`(`*списокПараметров*`)`

В коде из листинга 2.39, *а* (файл *10-Пример создания методов для класса.xlsm* на компакт-диске) расширяются функциональные возможности класса `Point` путем добавления в него трех методов. Метод `Move()` перемещает точку в направлении, заданном другой точкой. Метод `Length()` возвращает длину вектора, т. е. расстояние от начала координат до точки. Метод `ToString()` аккумулирует информацию об экземпляре класса в виде строки. Первый из этих методов реализован в виде процедуры, а второй и третий — в виде функций. В листинге 2.39, *б* приведен код из стандартного модуля.

Листинг 2.39, *а*. Методы. Модуль класса `Point`

```
Public x As Integer
Public y As Integer
Public Sub Move(ByVal pt As Point)
   x = x + pt.x
   y = y + pt.y
End Sub
Public Function Length() As Double
   Length = Sqr(x ^ 2 + y ^ 2)
End Function
Public Function ToString() As String
   ToString = "(" & x & "," & y & ")"
End Function
```

Листинг 2.39, *б*. Методы. Стандартный модуль

```
Sub TestPointWithMethods()
   Dim p1 As New Point
   p1.x = 1   :   p1.y = 1
   Debug.Print p1.ToString
   Debug.Print p1.Length
   Dim p2 As New Point
   p2.x = 2   :   p2.y = 2
   Debug.Print p2.ToString
   p1.Move p2
   Debug.Print p1.ToString
   Debug.Print p1.Length
End Sub
```

Свойства как средство ограничения доступа к полям класса

Многие классы имеют открытые поля (public), к которым программисты могут обращаться напрямую, но в большинстве случаев такой подход оказывается не слишком удобным. Более подходящим является использование закрытых полей (private), т. е. таких полей, которые недоступны вне данного класса. Использование закрытых полей позволяет защищать данные от внешнего воздействия. Поэтому процесс получения и установления значений параметров текущего состояния объекта желательно реализовать через свойства или специальные методы. Для этого при описании свойства надо воспользоваться специальными процедурами `Property`:

- процедура `Property Let` служит для объявления имен свойств, значениями которых являются данные, отличные от объектов;
- процедура `Property Set` служит для объявления имен свойств, значениями которых являются объекты;
- процедура `Property Get` обеспечивает возможность считывания значения свойств.

В приводимом далее коде (листинг 2.40, *а* и *б,* файл *11-Пример использования свойств для полей класса.xlsm* на компакт-диске) создается класс `Employee`, инкапсулирующий информацию о сотруднике фирмы. В этом классе имеются три закрытых поля: `fname`, `lname` и `pos`, которые описывают имя, фамилию и должность сотрудника. Для установки и считывания значений этих полей в классе определены три свойства: `FirstName`, `LastName` и `Position`. Также в классе описан метод `ToDebug()`, который выводит комбинированную информацию об экземпляре класса в окно **Immediate** (рис. 2.11).

Листинг 2.40, *а*. Свойства. Модуль класса `Employee`

```
Private fname As String
Private lname As String
Private pos As String
```

```
Property Get FirstName() As String
   FirstName = fname
End Property
Property Let FirstName(ByRef newfname As String)
   fname = newfname
End Property
Property Get LastName() As String
   LastName = lname
End Property
Property Let LastName(ByRef newlname As String)
   lname = newlname
End Property
Property Get Position() As String
   Position = pos
End Property
Property Let Position(ByRef newpos As String)
   pos = newpos
End Property
Public Sub ToDebug()
   Debug.Print "First Name: " & fname & vbCr & _
               "Last Name: " & lname & vbCr & "Position: " & pos
End Sub
```

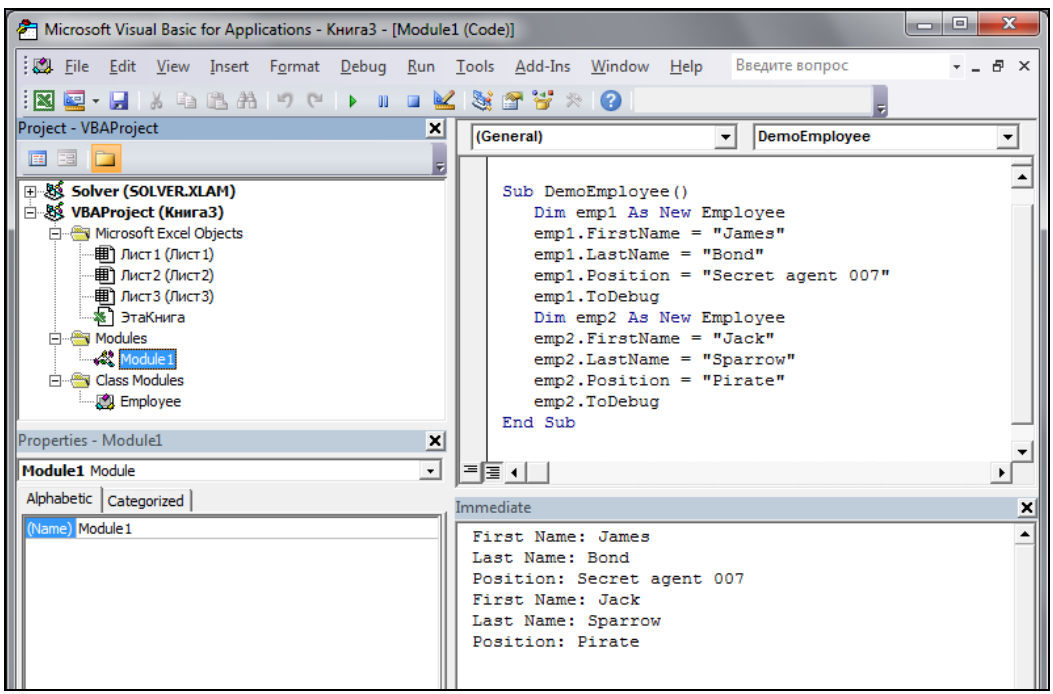


Рис. 2.11. Информация об экземпляре класса в окне **Immediate**

Листинг 2.40, *б*. Свойства. Стандартный модуль

```
Sub DemoEmployee()
   Dim emp1 As New Employee
   emp1.FirstName = "James"
   emp1.LastName = "Bond"
   emp1.Position = "Secret agent 007"
   emp1.ToDebug
   Dim emp2 As New Employee
   emp2.FirstName = "Jack"
   emp2.LastName = "Sparrow"
   emp2.Position = "Pirate"
   emp2.ToDebug
End Sub
```

Свойства "только для чтения" и "только для записи"

VBA позволяет создавать как свойства "только для чтения", т. е. те, которые могут только считывать данные, так и свойства "только для записи", т. е. те, которые могут только устанавливать значения. При описании свойства "только для чтения" не нужно объявлять процедуру `Property Let` или `Property Set`, а при описании свойства "только для записи" — блок `Property Get`.

В следующем примере (листинг 2.41, *а* и *б*) создается класс `Point`. Свойства `GetX` и `GetY` являются свойствами "только для чтения" и позволяют считывать значения полей. Свойства же `SetX` и `SetY` являются свойствами "только для записи" и предназначены для задания значений полям. Кроме того, в классе объявлен метод `ToString()` для вывода общей информации об экземпляре класса в виде строки.

Листинг 2.41, *а*. Свойства "только для чтения" и "только для записи". Модуль класса `Point`

```
Private x As Integer
Private y As Integer

Public Property Get GetX() As String
   GetX = x
End Property
Public Property Let SetX(ByRef valX As String)
   x = valX
End Property
Public Property Get GetY() As String
   GetY = y
End Property
Public Property Let SetY(ByRef valY As String)
   y = valY
```

```
End Property
Public Function ToString() As String
   ToString = "(" & x & "," & y & ")"
End Function
```

Листинг 2.41, *б*. Свойства "только для чтения" и "только для записи". Стандартный модуль

```
Sub DemoOnlyProperties()
   Dim p As New Point
   p.SetX = 1   :   p.SetY = 4
   Debug.Print p.ToString
   Debug.Print p.GetX & vbTab & p.GetY
End Sub
```

События

До сих пор объекты, которые мы создавали, не получали сообщений от других объектов и не передавали их, что может навести на мысль, что в VBA программы представляют собой последовательность выполняемых друг за другом процедур. Это совсем не так. Зачастую момент выполнения процедуры в программе обусловлен внешним фактором — *событием* (event). Например, нажатием кнопки, выбором переключателя, сворачиванием формы и т. д. Таким образом, событие играет в приложении роль сигнала, информирующего о том, что в системе произошло что-то важное, требующее дополнительного внимания.

Событие объявляется в пределах класса при помощи ключевого слова `Event`. Объявление должно включать идентификатор события и список его параметров, например у следующего события — `MoneyWithdrawn`, которое может генерироваться при снятии денег со счета, имеются два параметра: `Money` и `Cancel`. Первый из них возвращает снимаемую сумму, а второй определяет, надо ли отменить проводимую денежную операцию:

```
Public Event MoneyWithdrawn(ByVal Money As Long, ByRef Cancel As Boolean)
```

Сами по себе события не могут возвращать данные. Данные возвращаются через параметры события.

Объявление события означает только то, что оно может быть сгенерировано, в то время как сам процесс генерации события производится оператором `RaiseEvent`. Например, следующий оператор, помещенный в код, говорит о том, что при выполнении программы по его достижении сгенерируется событие `MoneyWithdrawn`.

```
RaiseEvent MoneyWithdrawn(Money, Cancel)
```

При объявлении переменной, которой будет присвоено значение экземпляра того класса, который может генерировать событие, надо дополнительно добавить ключевое слово `WithEvents`. Например, в приводимом далее коде событие будет генерироваться у переменной `sc` типа `SumListener`.

```
Private WithEvents sc As SumListener
```

Для того чтобы отреагировать на событие, надо его ассоциировать со специальной процедурой, которая называется *обработчиком события* (event handlers).

Такое ассоциирование производится с использованием шаблона процедуры обработки события.

```
Sub Объект_Событие(списокАргументов)
...
End Sub
```

В приводимом далее примере это будет следующая процедура.

```
Sub sc_SumDone(ByVal s As Double)
   Debug.Print CStr("Sum done " & s)
End Sub
```

Рассмотрим простой пример (листинг 2.42, *а* и *б*, файл *12-Пример класса с событием.xlsm* на компакт-диске). В классе `SumListener` имеются два открытых поля `a` и `b` и закрытое — `s`. Метод `Sum()` суммирует значения из полей `a` и `b` и помещает результат в поле `s`. Метод `GetRes()` возвращает значение найденной суммы. При выполнении метода `Sum()` по завершении вычисления суммы генерируется событие `SumDone(ByVal s As Double)`, параметр которого возвращает значение суммы. Таким образом, в процедуре `DemoEvent` результат в окно **Immediate** сначала выводится за счет обработки события `SumDone`, которое генерируется при использовании метода `Sum()`, а уж затем результат повторно явным образом выводится методом `GetRes()`.

Листинг 2.42, *а*. Класс с событием. Модуль класса `SumListener`

```
Public a As Double
Public b As Double
Private s As Double
Public Event SumDone(ByVal s As Double)
Public Sub Sum()
   s = a + b
   RaiseEvent SumDone(s)
End Sub
Public Function GetRes() As Double
   GetRes = s
End Function
```

Листинг 2.42, *б*. Класс с событием. Модуль листа или `ЭтаКнига`

```
Private WithEvents sc As SumListener
Sub DemoEvent()
   Set sc = New SumListener
   sc.a = 1   :   sc.b = 3
   sc.Sum
   Debug.Print sc.GetRes()
End Sub
Sub sc_SumDone(ByVal s As Double)
   Debug.Print CStr("Sum done " & s)
End Sub
```

Объект *Collection*

Объект `Collection` позволяет динамически группировать семейство объектов, идентифицированных по индексу. Объект `Collection` имеет лишь одно свойство `Count`, возвращающее число элементов набора, и три метода, приведенные в табл. 2.11.

Таблица 2.11. *Методы объекта* `Collection`

Метод	Описание
`Add`	Добавляет новый элемент в семейство. `Add item [, key, before, after]` Здесь: • *item* — обязательный параметр, специфицирующий добавляемый элемент; • *key* — необязательный параметр, идентифицирующий элемент. Может использоваться вместо его порядкового номера; • *before* — необязательный параметр, указывающий элемент, перед которым размещается добавляемый элемент; • *after* — необязательный параметр, определяющий элемент, после которого размещается добавляемый элемент
`Item`	Возвращает специфицированный элемент семейства. `Item(index)` Здесь *index* — порядковый номер элемента в семействе или его идентификатор, заданный параметром *key* метода `Add`. Нумерация элементов в объекте `Collection` начинается с *единицы*
`Remove`	Удаляет элемент из семейства. `Remove index` Здесь *index* — порядковый номер элемента в семействе или его идентификатор, заданный параметром *key* метода `Add`

Наши итоги

Изучив тщательно материал данной главы, вы научились создавать различные небольшие программы на языке VBA. Кроме того, теперь вы знаете:

- ❒ устройство процедур и функций;
- ❒ как объявить нужную переменную;
- ❒ основные типы данных и их использование;
- ❒ управляющие конструкции языка VBA;
- ❒ о возможностях встроенных диалоговых окон;
- ❒ основные подходы при объявлении класса и создании его экземпляра.

Глава 3

Обрабатываем данные при помощи формул и функций рабочего листа

В предыдущих двух главах мы познакомились с основами языка VBA, структурой программ на языке VBA, а также научились создавать различные простые программы с использованием управляющих конструкций. В этой главе мы разберем некоторые элементы автоматизации, которые доступны при обработке различных данных с использованием формул и функций рабочего листа, ссылок на ячейки и возможностей форматирования данных. Использование приемов, излагаемых далее, а также всех доступных средств VBA, позволит вам в полной мере создавать удобные приложения, которые ускорят как обработку данных различного типа, так и сократят время пользователей, сталкивающихся с различными однотипными вычислениями и операциями в различных сферах деятельности.

Примечание

Файлы рассмотренных в данной главе примеров можно найти в папке Glava_3 на компакт-диске.

Немного об адресации ячейки

На активном рабочем листе одна ячейка является активной (выделена черной рамкой). Перемещение по ячейкам осуществляется мышью или клавишами со стрелками. Каждая ячейка на активном рабочем листе определяется своим *адресом* (или *ссылкой на ячейку*), состоящим из имени столбца и номера строки, например, **A1**. Этот стиль ссылок так и называется — A1.

MS Excel поддерживает и другую систему адресации (стиль ссылок) — R1C1, когда нумеруются как строки, так и столбцы. В этой системе адресации, например, активная ячейка с адресом **R4C3** означает "четвертая строка, третий столбец. Отметим, что именно такая адресация достаточно часто используется при написании программ на VBA.

Чтобы изменить стиль адресации перейдите на вкладку **Файл** и выберите команду **Параметры**. В открывшемся окне **Параметры Excel** выберите слева категорию **Формулы**, а справа в разделе **Работа с формулами** установите или снимите флажок **Стиль ссылок R1C1**.

Существует еще один способ адресации ячейки — *по имени* (рис. 3.1). Имя или адрес активной ячейки вводится в *поле имен* (расположено у левого края строки формул). Для присвоения имени активной ячейки выделите требуемый диапазон,

перейдите на вкладку ленты **Формулы** и в группе команд **Определение имени** выберите из списка **Определение имени** команду **Определение имени**. При создании имен следует учесть:

- ❒ имена начинаются с буквы или подчеркивания;
- ❒ в имени вместо пробела или дефиса (-) используют подчеркивание (_) или точку (.);
- ❒ имена следует давать короткими и избегать аналогии со ссылками типа A1 или R1C1.

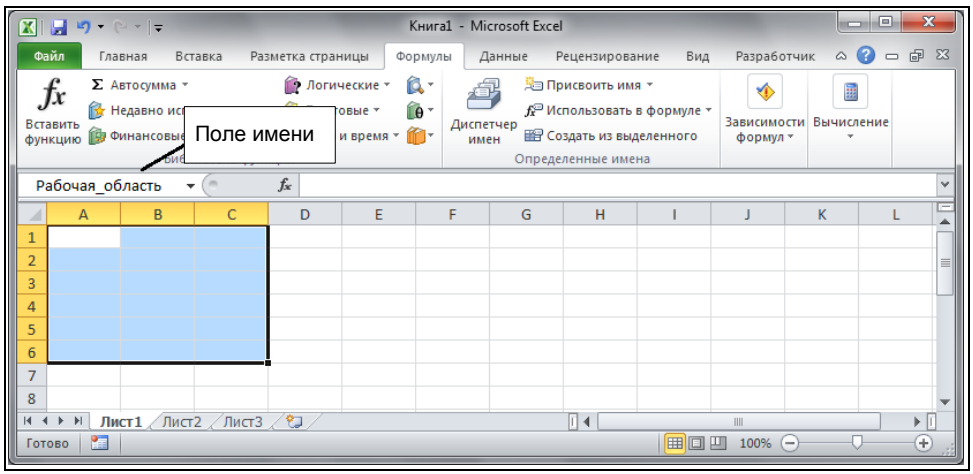


Рис. 3.1. Адресация по имени

Ячейка на неактивном рабочем листе идентифицируется именем листа и ее адресом на листе, например, **Лист2!A1** (восклицательный знак обязателен). Однако следует учесть, что адресация по имени абсолютна, поэтому при ссылке на ячейку по имени на неактивном рабочем листе не нужно указывать имя этого листа.

В ячейку электронной таблицы может быть введена информация различного типа: текст, числовые значения и формулы. Кроме того, каждая ячейка может быть *отформатирована* (т. е. оформлена) по-своему, причем параметры форматирования не влияют на содержимое ячейки.

При вводе данных MS Excel автоматически распознает их тип. Ввод выполняется в позицию активной ячейки. Как только в ячейку вводится хотя бы один символ, содержимое немедленно отражается в строке формул, и сразу же в этой строке появляется изображение трех кнопок, которые используются при обработке содержимого ячейки (рис. 3.2).

Для завершения ввода данных следует нажать клавишу <Enter>, или кнопку в строке формул с изображением галочки ✓, или клавишу управления курсором.

Если длина введенного в ячейку текста превышает ширину ячейки, то после ввода текст будет полностью представлен в таблице, закрывая собой незаполненные ячейки, либо будет урезан по правому краю.

Содержимое ячейки может отличаться от изображения на экране — фактическое содержание ячейки всегда представлено в строке формул.

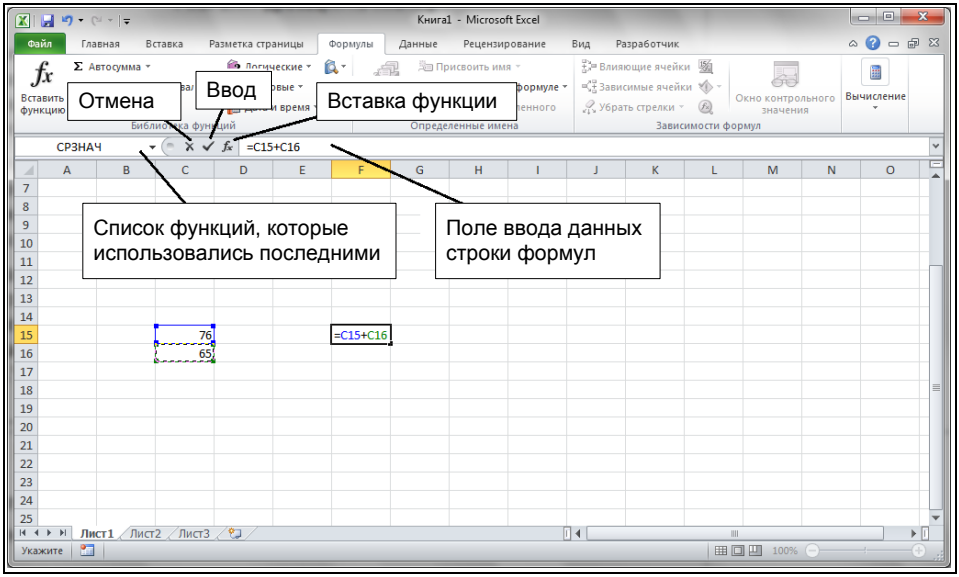


Рис. 3.2. Строка формул в режиме ввода/редактирования формулы

Для редактирования данных в ячейке следует сделать ее активной и нажать клавишу <F2> либо щелкнуть мышью в строке формул.

Для того чтобы вводимым данным не был автоматически присвоен один из заданных в MS Excel форматов, перед вводимой информацией следует ставить апостроф (').

Методы объекта *Range*

Объект `Range` (диапазон) обладает большой коллекцией методов, предоставляющих в распоряжение разработчика возможность программировать целый спектр действий от копирования диапазона в буфер обмена до нахождения корня нелинейного уравнения. Наиболее используемые методы для объекта `Range`:

- `Activate`;
- `AddComment`;
- `AutoFill`;
- `AutoFit`;
- `BorderAround`;
- `Clear`;
- `ClearComments`;
- `ClearComments`;
- `ClearContents`;
- `ClearFormats`;
- `ClearNotes`;
- `Copy`;
- `CopyPicture`;
- `Cut`;
- `DataSeries`;
- `Delete`;
- `FillDown`;
- `FillLeft`;
- `FillRight`;
- `FillUp`;
- `Find`;
- `FindNext`;
- `FindPrevious`;
- `FunctionWizard`;
- `GoalSeek`;
- `Insert`;
- `PasteSpecial`;
- `Replace`;
- `Select`;
- `Show`.

Необходимую информацию об использовании методов для объекта `Range` можно найти в справочной системе VBA. Остановимся на нескольких основных примерах.

Активизация и выбор диапазона

Метод `Activate` объекта `Range` активизирует диапазон, а метод `Select` выбирает диапазон, т. е. возвращает объект `Selection`. Например, в следующем коде сначала активизируется ячейка **A2**, затем в активную ячейку вводится число `1`, после чего выбирается диапазон **A3:A4** и в выбранный диапазон вводится число `3`.

```
Range("A2").Activate
ActiveCell.Value =  1
Range("A3:A4").Select
Selection.Value = 3
```

Автоматический подбор размеров диапазона так, чтобы в нем помещались введенные данные

Метод `AutoFit` объекта `Range` автоматически подбирает ширину столбца и высоту строки так, чтобы в ней помещались введенные данные. В следующем коде (листинг 3.1, файл *1-Методы объекта Range.xlsm* на компакт-диске) продемонстрирована техника использования метода `AutoFit` при создании заголовка некоей отчетной таблицы.

Листинг 3.1. Автоматический подбор размеров диапазона

```
Sub DemoAutoFit()
   Range("A1").Value = "Июнь"
   Range("B1").Value = " Июль "
   Range("C1").Value = "Август"
   Columns("A:C").AutoFit
   Range("D1").Value = "Суммарный объем продаж"
   Range("D1").Columns.AutoFit
End Sub
```

Заполнение диапазона по одному значению

Метод `FillDown` объекта `Range` заполняет сверху вниз диапазон на основе значений, расположенных в верхней строчке этого диапазона, причем данные значения копируются во все остальные ячейки диапазона.

Метод `FillUp` объекта `Range` заполняет снизу вверх диапазон на основе значений, расположенных в нижней строчке этого диапазона.

Метод `FillLeft` объекта `Range` заполняет справа налево диапазон на основе значений, расположенных в крайнем правом столбце этого диапазона.

Метод `FillRight` объекта `Range` заполняет слева направо диапазон на основе значений, расположенных в крайнем левом столбце этого диапазона.

Например, в данной инструкции содержимое ячейки **A10** копируется в каждую из ячеек диапазона **A1:A9**.

```
Range("A1:A10").FillUp
```

Обрамление диапазона границей

Метод `BorderAround` объекта `Range` обрамляет диапазон границей.

```
BorderAround(LineStyle, Weight, ColorIndex, Color)
```

- *`LineStyle`* — необязательный параметр, задающий стиль границы. Допустимыми значениями являются следующие константы `XlLineStyle`: `xlContinuous`, `xlDash`, `xlDashDot`, `xlDashDotDot`, `xlDot`, `xlDouble`, `xlLineStlyeNone`, `xlSlantDashDot` и `xlLineStlyeNone`.
- *`Weight`* — необязательный параметр, определяющий толщину границы. Допустимыми значениями являются следующие константы `XlBorderWeight`: `xlHairline`, `xlMedium`, `xlThick` и `xlThin`.
- *`ColorIndex`* — необязательный параметр, задающий цвет в соответствии с текущей палитрой цветов. Также допустимы следующие константы `XlColorIndex`: `xlColorIndexAutomatic` и `xlColorIndexNone`.
- *`Color`* — необязательный параметр, определяющий цвет в соответствии с RGB-моделью.

Например, следующая инструкция создает вокруг диапазона **A1:B2** двойную толстую границу зеленого цвета.

```
Range("A1:B2").BorderAround LineStyle:=xlDouble, Weight:=xlThick, _
                            Color:=RGB(0, 255, 0)
```

Очистка ячейки

Методы `Clear`, `ClearComments`, `ClearContents`, `ClearFormats` и `ClearNotes` объекта `Range` очищают диапазон и в диапазоне комментарии, содержание, форматы и примечания. Например, следующая инструкция очищает диапазон **A1:G37**.

```
Range("A1:G37").Clear
```

Копирование, вырезание и удаление данных из диапазона

Метод `Copy` объекта `Range` копирует диапазон в другой диапазон или в буфер обмена.

```
Copy(Destination)
```

Здесь *`Destination`* — необязательный параметр, определяющий диапазон, в который копируется данный диапазон. Если этот параметр опущен, то копирование происходит в буфер обмена. В данном примере диапазон **A1:D4** активного рабочего листа копируется в диапазон **E5:H8** рабочего листа **Лист2**.

```
Range("A1:D4").Copy Worksheets("Лист2").Range("E5")
```

Метод `Cut` объекта `Range` вырезает, т. е. копирует с удалением диапазон в указанный диапазон или в буфер обмена.

```
Cut(Destination)
```

Здесь *`Destination`* — необязательный параметр, задающий диапазон, в который копируется данный диапазон. Если этот параметр опущен, то диапазон копируется в буфер обмена. В данном примере диапазон **A1:D4** рабочего листа **Лист1** копируется с удалением в буфер обмена.

```
Worksheets("Лист1").Range("A1:D4").Cut
```

Метод `Delete` объекта `Range` удаляет диапазон. В данном примере удаляется третья строка активной рабочей страницы.

```
Rows(3).Delete
```

Специальная вставка

Метод `PasteSpecial` объекта `Range` производит специальную вставку (special paste) из буфера обмена. Этот метод программирует выполнение на рабочем листе команды **Специальная вставка**, которая запускается соответствующим выбором из списка при щелчке по кнопке **Вставка**, расположенной в группе команд **Буфер обмена** на вкладке **Главная** ленты.

```
expression.PasteSpecial(Paste, Operation, SkipBlanks, Transpose)
```

- *expression* — объект типа `Range`, задающий диапазон или ссылку на верхнюю левую ячейку диапазона, в который вставляются данные из буфера обмена.
- *Paste* — необязательный параметр. Определяет ту часть содержания буфера обмена, которая должна быть вставлена в диапазон. Допустимыми значениями являются следующие константы `XlPasteType`: `xlPasteAll`, `xlPasteAllExceptBorders`, `xlPasteColumnWidths`, `xlPasteComments`, `xlPasteFormats`, `xlPasteFormulas`, `xlPasteFormulasAndNumberFormats`, `xlPasteValidation`, `xlPasteValues` и `xlPasteValuesAndNumberFormats`.
- *Operation* — необязательный параметр. Определяет операцию над вставляемыми данными и теми, которые содержатся в диапазоне. Допустимыми значениями являются следующие константы `XlPasteSpecialOperation`: *xlPasteSp*`ecialOperationAdd`, `xlPasteSpecialOperationDivide`, `xlPasteSpecialOperationMultiply`, `xlPasteSpecialOperationNone` и `xlPasteSpecialOperationSubtract`.
- *SkipBlanks* — необязательный параметр. Принимает логические значения и устанавливает, учитываются ли пустые ячейки при вставке.
- *Transpose* — необязательный параметр. Принимает логические значения и устанавливает, вставляется ли диапазон транспонированным.

В приведенном далее коде данные из диапазона **C1:C5** рабочего листа **Лист1** вставляются в диапазон **D1:D5** того же листа, причем они не заменяют уже существующие данные, а прибавляют к ним данные из диапазона **C1:C5**.

```
With Worksheets("Лист1")
    .Range("C1:C5").Copy
    .Range("D1:D5").PasteSpecial Operation:=xlPasteSpecialOperationAdd
End With
```

Тот же результат можно получить при помощи следующих инструкций, где метод применяется не ко всему диапазону, а только к его левой верхней ячейке.

```
With Worksheets("Лист1")
    .Range("C1:C5").Copy
    .Range("D1").PasteSpecial Operation:=xlPasteSpecialOperationAdd
End With
```

Следующий код копирует данные из диапазона **A1:C1** в буфер обмена и затем вставляет только значения в диапазон **A5:C5**.

```
Range("A1:C1").Copy
Range("A5").PasteSpecial Paste:=xlPasteValues, Operation:=xlNone
```

Вставка диапазона с транспонированием

Вставка диапазона с транспонированием осуществляется методом `PasteSpecial` при значении его параметра `Transpose` равным `True`. Например, в следующем коде (листинг 3.2, см. также файл *1-Методы объекта Range.xlsm* на компакт-диске) значения из диапазона **A1:C2** копируются с транспонированием в диапазон **E1:F3**.

Листинг 3.2. Вставка диапазона с транспонированием

```
Sub Transp()
  Range("A1:C2").Copy
  Range("E1").PasteSpecial Paste:=xlPasteAll, Operation:=xlNone, _
                           Transpose:=True
End Sub
```

Снятие выделения после специальной вставки

После копирования или вставки данных в буфер обмена исходный диапазон получает выделение, и оно не исчезает после проведения специальной вставки. Для того чтобы это выделение все же удалить, надо установить свойство `CutCopyMode` объекта `Application` равным значению `False`, как показано в следующем коде:.

```
Range("C1:C5").Copy
Range("D1").PasteSpecial Operation:=xlPasteSpecialOperationAdd
Application.CutCopyMode = False
```

Добавление ячейки, строки или столбца

Метод `Insert` объекта `Range` добавляет в рабочий лист ячейку, строку или столбец.

`Insert(`*`Shift`*`, `*`CopyOrigin`*`)`

- *`Shift`* — необязательный параметр, специфицирующий способ смещения ячеек. Допустимыми значениями являются следующие константы `XlInsertShiftDirection`: `xlShiftToRight` и `xlShiftDown`.
- *`CopyOrigin`* — необязательный параметр, который задает правило копирования источника данных.

В следующем примере первая инструкция вставляет новую строку перед четвертой строкой рабочего листа, а вторая — ячейку слева от ячейки **G9**.

```
Rows(4).Insert
Range("G9").Insert Shift:=xlShiftToRight
```

Что дает автозаполнение?

Ячейки можно заполнять некоторой информацией автоматически. Для этого предназначена функция *автозаполнения*, которая вызывается с помощью *маркера автозаполнения* (черный крест возле правого нижнего угла выделенной ячейки или ячеек) при подведении к нему указателя мыши (рис. 3.3, файл *2-Автозаполнение.xlsx*

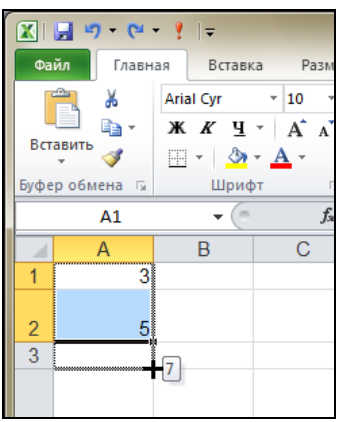


Рис. 3.3. Использование маркера автозаполнения

на компакт-диске). Можно также воспользоваться командой **Заполнить | Прогрессия**, нажав кнопку **Заполнить** в группе команд **Редактирование** на вкладке **Главная** ленты.

Автозаполнение удобно использовать, если необходимо:

- ввести одну и ту же информацию в расположенные рядом ячейки;
- ввести некоторые списки (например, дни недели). Сами списки можно сформировать следующим образом: задать последовательность чисел или дат.

Заполнение диапазона прогрессией

Метод `DataSeries` объекта `Range` заполняет диапазон прогрессией. Метод `DataSeries` программирует выполнение команды **Заполнить | Прогрессия** при нажатии кнопки **Заполнить** в группе команд **Редактирование** на вкладке **Главная** ленты.

```
DataSeries(RowCol, Type, Date, Step, Stop, Trend)
```

- *RowCol* — необязательный параметр, определяющий направление, в котором создается прогрессия. Допустимые значения: `xlRows` (вдоль строк), `xlColumns` (вдоль столбцов). Если параметр опущен, то для задания направления используются размеры данного диапазона.
- *Type* — необязательный параметр, специфицирующий тип прогрессии. Допустимыми значениями являются следующие константы `XlDataSeriesType`: `xlDataSeriesLinear` (линейная, используется по умолчанию), `xlGrowth` (геометрическая), `xlChronological` (даты), `xlAutoFill` (автозаполнение).
- *Date* — необязательный параметр, задающий тип последовательности дат, если параметр *Type* принимает значение `xlChronological`. Допустимыми значениями являются следующие константы `XlDataSeriesDate`: `xlDay` (дни, используется по умолчанию), `xlWeekday` (дни недели), `xlMonth` (месяцы), `xlYear` (годы).
- *Step* — необязательный параметр, определяющий шаг изменения прогрессии. По умолчанию полагается равным 1.
- *Stop* — необязательный параметр, задающий предельное значение прогрессии. По умолчанию строится прогрессия во всем выделенном диапазоне.
- *Trend* — необязательный параметр, принимает логические значения. Если значение параметра равно `True`, то создается арифметическая или геометрическая прогрессия, а если `False`, то создается список.

Например, следующая инструкция (листинг 3.3, *а*, см. также файл *3-Заполнение диапазона прогрессией.xlsm* на компакт-диске) выведут членов арифметической прогрессии, шаг которой равен 2, 0 является ее первым членом, а 10 — последним, т. е. в диапазон **A1:A6** будут выведены следующие числа: 0, 2, 4, 6, 8 и 10 (рис. 3.4).

Листинг 3.3, *а*. Арифметическая прогрессия

```
Sub Progr1()
  Range("A1").Value = 0
  Range("A1").DataSeries Rowcol:=xlColumns, Type:=xlDataSeriesLinear, _
                         Step:=2, Stop:=10
End Sub
```

Инструкции листинга 3.3, *б* (файл *3-Заполнение диапазона прогрессией.xlsm* на компакт-диске) отобразят в диапазон **B1:B5** членов геометрической прогрессии с коэффициентом — 3, первый член которой — 1. Таким образом, в диапазон **B1:B5** будут выведены значения 1, 3, 9, 27 и 81 (рис. 3.4).

Листинг 3.3, *б*. Геометрическая прогрессия

```
Sub Rrogr2()
  Range("B1").Value = 1
  Range("B1:B5").DataSeries Rowcol:=xlColumns, Type:=xlGrowth, Step:=3
End Sub
```

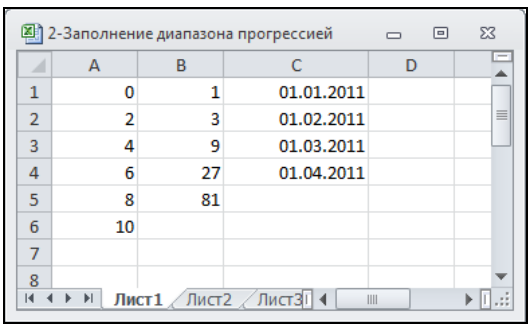


Рис. 3.4. Прогрессии

И, наконец, листинг 3.3, *в* (см. также файл *3-Заполнение диапазона прогрессией.xlsm* на компакт-диске) демонстрирует размещение в диапазоне **C1:C4** последовательность дат, члены которой разнятся ровно на месяц, т. е. значения `01.01.2011`, `01.02.2011`, `01.03.2011`, `01.04.2011` (рис. 3.4).

Листинг 3.3, *в*. Прогрессия дат

```
Sub Progr3()
  Range("C1").Value = "1/01/2011"
  Range("C1:C4").DataSeries Rowcol:=xlColumns, Type:=xlChronological, _
                            Date:=xlMonth
End Sub
```

Автозаполнение ячеек диапазона элементами последовательности

Метод `AutoFill` объекта `Range` производит автозаполнение ячеек диапазона элементами последовательности. Метод `AutoFill` отличается от метода `DataSeries` тем, что явно указывается диапазон, в котором будет располагаться прогрессия. Метод `AutoFill` программирует выполнение копирования данных на диапазон, когда пользователь располагает указатель мыши на маркере заполнения исходного

диапазона и перемещает его вниз и вправо, выделяя весь диапазон, в который переносятся исходные данные.

```
expression.AutoFill(Destination, Type)
```

- *expression* — обязательный элемент, который задает диапазон, с которого начинается заполнение.
- *Destination* — обязательный параметр, определяющий диапазон, который заполняется. Этот диапазон должен содержать диапазон, указанный в *expression*.
- *Type* — необязательный параметр, указывающий тип заполнения. Допустимыми значениями являются следующие константы `XlAutoFillType`: `xlFillDefault`, `xlFillSeries`, `xlFillCopy`, `xlFillFormats`, `xlFillValues`, `xlFillDays`, `xlFillWeekdays`, `xlFillMonths`, `xlFillYears`, `xlLinearTrend`, `xlGrowthTrend`. По умолчанию используется тот тип заполнения, который наиболее подходит к данным из диапазона, указанного в *expression*.

Например, следующие инструкции (листинг 3.4, *а,* см. также файл *3-Заполнение диапазона прогрессией.xlsm* на компакт-диске) заполняют диапазон **A1:A5** членами арифметической прогрессии, причем ее первыми двумя членами являются 1 и 3, т. е. те значения, которые предварительно были введены в ячейки **A1** и **A2** (рис. 3.5).

Листинг 3.4, *а*. Последовательности. Арифметическая последовательность

```
Sub Progr4()
  Range("A1").Value = 1
  Range("A2").Value = 3
  Range("A1:A2").AutoFill Destination:=Range("A1:A5"), Type:=xlLinearTrend
End Sub
```

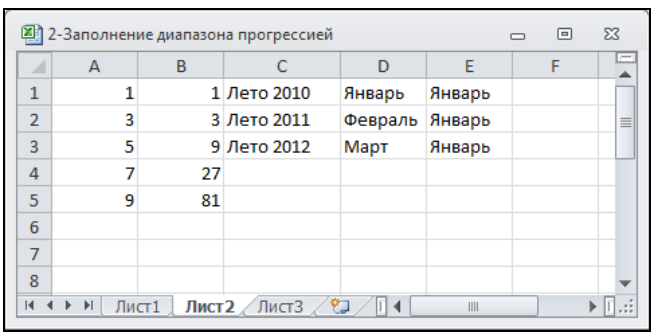


Рис. 3.5. Последовательности

Листинг 3.4, *б* (см. также файл *3-Заполнение диапазона прогрессией.xlsm* на компакт-диске) демонстрирует вывод на рабочем листе нескольких членов геометрической прогрессии с теми же двумя начальными значениями в диапазон **B1:B5** (см. рис. 3.5).

Листинг 3.4, *б*. Последовательности. Геометрическая последовательность

```
Sub Progr5()
  Range("B1").Value = 1
  Range("B2").Value = 3
  Range("B1:B2").AutoFill Destination:=Range("B1:B5"), Type:=xlGrowthTrend
End Sub
```

Следующие инструкции (листинг 3.4, *в,* см. также файл *3-Заполнение диапазона прогрессией.xlsm* на компакт-диске) выведут в диапазон **C1:C3** последовательность значений `Лето 2010`, `Лето 2011` и `Лето 2012` с шагом 1, определенным по умолчанию методом `AutoFill` (см. рис. 3.5).

Листинг 3.4, *в*. Последовательности. Автозаполнение

```
Sub Progr6()
  Range("C1").Value = "Лето 2010"
  Range("C1").AutoFill Destination:=Range("C1:C3"), Type:=xlFillSeries
End Sub
```

Приведенный ниже листинг 3.4, *г (*см. также файл *3-Заполнение диапазона прогрессией.xlsm* на компакт-диске) выводит в диапазон **D1:D3** первые три члена списка — имена месяцев, начиная с `Январь` (см. рис. 3.5).

Листинг 3.4, *г*. Последовательности. Месяцы

```
Sub Progr7()
  Range("D1").Value = "Январь"
  Range("D1").AutoFill Destination:=Range("D1:D3"), Type:=xlFillSeries
End Sub
```

Следующие инструкции (листинг 3.4, *д*, см. также файл *3-Заполнение диапазона прогрессией.xlsm* на компакт-диске) копируют содержимое ячейки **E1** на все ячейки диапазона **E1:E3** (см. рис. 3.5).

Листинг 3.4, *д*. Последовательности. Копирование

```
Sub Progr8()
  Range("E1").Value = "Январь"
  Range("E1").AutoFill Destination:=Range("E1:E3"), Type:=xlCopy
End Sub
```

Табуляция функции

Метод `AutoFill` позволяет решить задачу табуляции функции, т. е. вывода ее значений при изменении значения ее параметра. Например, требуется найти значения функции sin(*x*) при значении параметра *x*, изменяющегося от 0 до 2 с шагом 0,2.

Это можно сделать так, как показано в листинге 3.5 (см. также файл *3-Заполнение диапазона прогрессией.xlsm* на компакт-диске). Сначала в ячейку **A1** ввести первый член арифметической последовательности требуемых значений параметра, а потом с помощью метода `DataSeries` построить и всю последовательность вдоль столбца **A**. Затем определить как текущий диапазон с этими значениями. Диапазон, в котором расположатся соответствующие значению функции, разместится в столбце **B**, и его можно найти при помощи свойства `Offset`. Далее остается самая малость. В ячейку **B1** ввести формулу `=SIN(A1)` для нахождения значения функции при значении параметра, равном 0, а затем скопировать данную формулу на весь диапазон, отводимый под искомые значения функции (рис. 3.6).

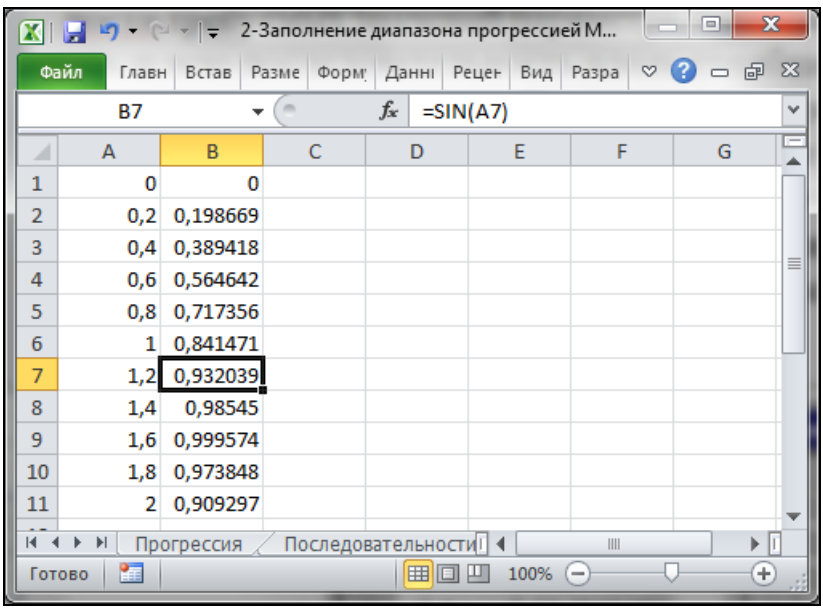


Рис. 3.6. Табуляция функции

Листинг 3.5. Табуляция функции

```
Sub DemoDataSeries()
   Range("A1").Value = 0
   Range("A1").DataSeries Rowcol:=xlColumns, Type:=xlDataSeriesLinear, _
                          Step:=0.2, Stop:=2
   Dim rgn As Range
   Set rgn = Range("A1").CurrentRegion
   Set rgn = rgn.Offset(0, 1)
   Range("B1").Formula = "=SIN(A1)"
   Range("B1").AutoFill Destination:=rgn, Type:=xlCopy
End Sub
```

Используем автозамену

Автозамена позволяет автоматически заменять какие-либо вводимые символы (слова) или сокращения, предварительно определенные в диалоговом окне **Автозамена**.

Свойство `AutoCorrect` объекта `Application` возвращает объект `AutoCorrect`, который позволяет управлять автозаменой на рабочем листе. Свойства этого объекта программируют значение параметров, устанавливаемых в окне **Автозамена** на вкладке **Автозамена** в группе **Заменять при вводе** (рис. 3.7): перейдите на вкладку **Файл** ленты, щелкните по кнопке **Параметры**, в открывшемся окне **Параметры Excel** выберите слева категорию **Правописание**, а справа в группе **Параметры автозамены** нажмите кнопку **Параметры автозамены** возле параметра **Настройка исправления и форматирования текста при вводе**.

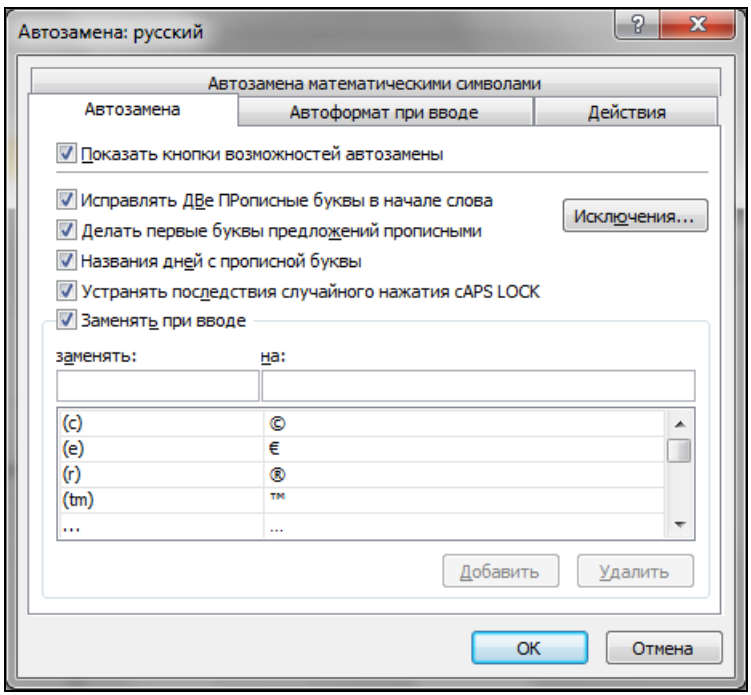


Рис. 3.7. Окно **Автозамена**, вкладка **Автозамена**

Например, в следующем коде (см. модуль `ЭтаКнига` в файле *4-Автозамена.xlsm* на компакт-диске) первая процедура обрабатывает событие `Open` рабочей книги и обеспечивает добавление в список автозамены трех новых элементов. А именно `спб` будет при вводе автоматически заменяться на `Санкт-Петербург`, `мск` — на `Москва`, а `гр` — на `Гродно`. Вторая процедура обрабатывает событие `BeforeClose`, генерируемое при закрытии рабочей книги. Она реализует исключение этих трех элементов из списка автозамены.

Ищем значения

Команды из списка **Найти и выделить**, расположенного на вкладке **Главная** ленты в группе команд **Редактирование**, позволяют быстро найти и заменить содержимое ячейки в соответствии с заданными критериями или же просто осуществит требуемый поиск. С другой стороны, с использованием VBA вы можете также указать необходимые критерии для поиска данных в конкретном диапазоне, осуществить замену и т. п. Рассмотрим некоторые примеры.

Поиск значения в диапазоне

Метод `Find` объекта `Range` производит поиск специфицированной информации в указанном диапазоне и возвращает ссылку на первую найденную ячейку, в которой требуемая найдена. В случае отсутствия искомых данных метод возвращает значение `Nothing`.

```
Find(What, After, LookIn, LookAt, SearchOrder, SearchDirection,
↳MatchCase, MatchByte, SearchFormat)
```

- *What* — обязательный параметр, задающий искомые данные.
- *After* — необязательный параметр, указывающий на ячейку, после которой надо производить поиск.
- *LookIn* — необязательный параметр, специфицирующий, где производить поиск. Допустимыми значениями являются следующие константы `XlFindLookIn`: `xlComments`, `xlFormulas` и `xlValues`.
- *LookAt* — необязательный параметр, указывающий, как надо искать. Допустимыми значениями являются следующие константы `XlLookAt`: `xlWhole` и `xlPart`.
- *SearchOrder* — необязательный параметр, задающий порядок просмотра диапазона. Допустимыми значениями являются следующие константы `XlSearchOrder`: `xlByRows` и `xlByColumns`.
- *SearchDirection* — необязательный параметр, определяющий направление просмотра. Допустимыми значениями являются следующие константы `XlSearchDirection`: `xlNext` и `xlPrevious`.
- *MatchCase* — необязательный параметр, указывающий, надо ли при поиске учитывать регистр букв.
- *MatchByte* — необязательный параметр, применяется редко.
- *SearchFormat* — необязательный параметр, задающий формат поиска.

Например, следующий код (листинг 3.6, см. также файл *5-Поиск значений и др.xlsm* на компакт-диске) производит поиск значения 17 в диапазоне **A1:A10**. В случае его обнаружения на экране отображается окно с адресом первой обнаруженной ячейки.

Листинг 3.6. Поиск значения

```
Sub Find1()
  Dim rng As Range
  Set rng = Range("A1:A10").Find(What:=17, LookIn:=xlValues)
  If Not (rng Is Nothing) Then
```

```
    MsgBox rng.Address
  Else
    MsgBox "Не найдено значение"
  End If
End Sub
```

Код из листинга 3.7 (см. также файл *5-Поиск значений и др.xlsm* на компакт-диске) производит поиск подстроки `"BHV"` без учета регистра в диапазоне **A1:A10**. В случае успешного поиска на экране отображается окно со значением свойства `Value` обнаруженной ячейки.

Листинг 3.7. Поиск подстроки без учета регистра

```
Sub DemoFindNoMatchCase ()
   Dim rng As Range
   Set rng = Range("A1:A20").Find(What:="BHV", LookIn:=xlValues, _
                              LookAt:=xlPart, MatchCase:=False)
   If Not (rng Is Nothing) Then
      MsgBox rng.Value
   Else
      MsgBox "Не найдено подходящее значение"
   End If
End Sub
```

Повторный поиск и поиск всех значений

Методы `FindNext` и `FindPrevious` объекта `Range` реализуют повторный вызов метода `Find` для продолжения специфицированного поиска. Первый из методов производит поиск следующей ячейки, а второй — поиск предыдущей, удовлетворяющей объявленным критериям поиска.

```
FindNext(After)
FindPrevious(After)
```

Здесь *`After`* — необязательный параметр, указывающий на ячейку, после которой надо производить поиск.

В качестве примера приведем код (листинг 3.8, см. также файл *5-Поиск значений и др.xlsm* на компакт-диске), который производит поиск подстроки `"BHV"` без учета регистра в диапазоне **A1:A10**. Все найденные ячейки заливаются желтым цветом.

Листинг 3.8. Нахождение всех вхождений подстроки в данный диапазон

```
Sub Find2()
   Dim firstAddress As String
   Dim rng As Range
   Set rng = Range("A1:A10").Find(What:="BHV", LookIn:=xlValues, _
                                  LookAt:=xlPart, MatchCase:=False)
   If Not (rng Is Nothing) Then
      firstAddress = rng.Address
```

```
      Do
         rng.Interior.Color = RGB(255, 255, 0)
         Set rng = Range("a1:a10").FindNext(rng)
      Loop While Not (rng Is Nothing) And rng.Address <> firstAddress
   End If
End Sub
```

Замена значений

Метод `Replace` объекта `Range` производит замену в указанном диапазоне.

```
Replace(What, Replacement, LookAt, SearchOrder, SearchDirection,
↳ MatchCase, MatchByte, SearchFormat, ReplaceFormat)
```

- *What* — обязательный параметр, задающий строку, которая будет заменяться.
- *Replacement* — обязательный параметр, задающий строку, которой будет производиться замещение.
- *LookAt* — необязательный параметр, указывающий, как надо искать. Допустимыми значениями являются следующие константы `XlLookAt`: `xlWhole` и `xlPart`.
- *SearchOrder* — необязательный параметр, задающий порядок просмотра диапазона. Допустимыми значениями являются следующие константы `XlSearchOrder`: `xlByRows` и `xlByColumns`.
- *SearchDirection* — необязательный параметр, определяющий направление просмотра. Допустимыми значениями являются следующие константы `XlSearchDirection`: `xlNext` и `xlPrevious`.
- *MatchCase* — необязательный параметр, указывающий, надо ли при поиске учитывать регистр букв.
- *MatchByte* — необязательный параметр, применяется редко.
- *SearchFormat* — необязательный параметр, задающий формат поиска.
- *ReplaceFormat* — необязательный параметр, задающий формат замены.

Например, в следующем коде в столбце **A** строка `"MS"` заменяется строкой `"Microsoft"`.

```
Columns("A").Replace What:="MS", Replacement:="Microsoft", _
                     SearchOrder:=xlByColumns, MatchCase:=True
```

Как отобразить примечания?

При работе удобно использовать *примечания*, это упрощает просмотр текста, присоединенного к ячейкам. Для создания примечания и работы с примечаниями в MS Excel 2010 имеется группа команд **Примечания** на вкладке **Рецензирование** ленты. С другой стороны, вы также можете воспользоваться свойством `DisplayCommentIndicator` объекта `Application` для работы с примечаниями.

Свойство `DisplayCommentIndicator` объекта `Application` позволяет управлять стилем отображения примечаний. Допустимыми значениями этого свойства являются следующие константы `XlCommentDisplayMode`:

- `xlNoIndicator` — нет индикатора;
- `xlCommentIndicatorOnly` — только индикатор;
- `xlCommentAndIndicator` — примечание и индикатор.

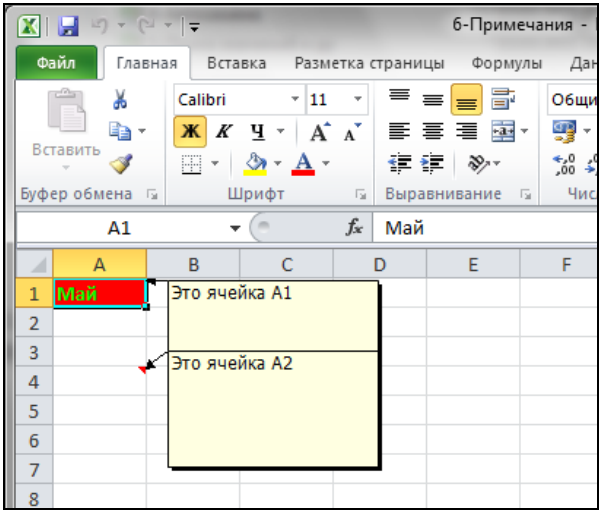


Рис. 3.8. Управление отображением примечаний и их индикаторов

В демонстрационном примере (листинг 3.9, см. также файл *6-Примечания.xlsm* на компакт-диске) при открытии рабочей книги к ячейкам **A1** и **A4** добавляется по примечанию. При выборе ячейки **A1** отображаются как примечания, так и их индикаторы, а при выборе любой другой ячейки они скрываются (рис. 3.8).

Листинг 3.9. Управление отображением примечаний и их индикаторов. Модуль `ЭтаКнига`

```
Private Sub Workbook_Open()
   Worksheets(1).Range("A1").ClearComments
   Worksheets(1).Range("A1").AddComment
   Worksheets(1).Range("A1").Comment.Visible = True
   Worksheets(1).Range("A1").Comment.Text Text:="Это ячейка A1"
   Worksheets(1).Range("A4").ClearComments
   Worksheets(1).Range("A4").AddComment
   Worksheets(1).Range("A4").Comment.Visible = True
   Worksheets(1).Range("A4").Comment.Text Text:="Это ячейка A2"
End Sub
Private Sub Workbook_SheetSelectionChange(ByVal Sh As Object, _
                                          ByVal Target As Range)
   If Sh.Name = Worksheets(1).Name Then
      If Target.Address = "$A$1" Then
         Application.DisplayCommentIndicator = xlCommentAndIndicator
      Else
         Application.DisplayCommentIndicator = xlNoIndicator
      End If
   End If
End Sub
```

Проверяем данные

При заполнении рабочего листа часто полезно контролировать соответствие вводимых данных определенным требованиям. Кроме того, часто приходится вводить определенные значения. *Проверка данных* MS Excel позволяет установить для ячейки или диапазона ячеек допустимый тип значений или другие условия проверки.

Чтобы установить параметры проверки, следует:

- выделить диапазон ячеек;
- перейти на вкладку **Данные** ленты и в группе команд **Работа с данными** выбрать из списка **Проверка данных** команду **Проверка данных**.

Для появления сообщений при вводе следует после установки необходимых параметров выбрать в открывшемся окне **Проверка вводимых значений** вкладку **Сообщение для ввода**, а для установки сообщений об ошибке — вкладку **Сообщение об ошибке**.

Что нужно знать о форматах данных?

Числовые значения, которые вводятся или вычисляются, представляют собой последовательности цифр. Для наглядности представления данные желательно отформатировать, выполнив одно из следующих действий:

- выделите ячейку (ячейки) и воспользуйтесь командой **Формат | Формат ячейки |** вкладка **Число**, которая расположена в группе команд **Ячейки** на вкладке **Главная** ленты;
- правой кнопкой мыши щелкните на выделенной ячейке или группе ячеек и выберите команду **Формат ячеек** (также вкладка **Число**).

На указанной вкладке откроется перечень основных типов числовых форматов, доступных пользователю.

Форматирование числа на VBA

Чтобы представить числовое значение как дату, время, денежное значение или в специальном формате на VBA, следует использовать функцию `Format()`, которая возвращает значение типа `Variant` (`String`), содержащее выражение, отформатированное согласно инструкциям, заданным в описании формата.

```
Format(Expression[, Format[, FirstDayOfWeek [, FirstWeekOfYear]]])
```

- *`Expression`* — любое допустимое выражение.
- *`Format`* — любое допустимое именованное или определяемое пользователем выражение формата. Примером именованного формата является `Fixed` — формат действительного числа с двумя значащими цифрами после десятичной точки. Примеры именованных форматов даны в табл. 3.1 и 3.2.
- *`FirstDayOfWeek`* — константа, определяющая первый день недели.
- *`FirstWeekOfYear`* — константа, определяющая первую неделю года.

***Таблица 3.1.** Именованные числовые форматы*

Имя формата	Описание
`General Number`	Число без разделителя тысяч
`Currency`	Использует установки страны из Панели управления. Отображает две цифры справа от десятичного разделителя
`Fixed`	Отображает по крайней мере одну цифру слева и две справа от десятичного разделителя
`Standard`	Отображает по крайней мере одну цифру слева и две справа от десятичного разделителя и выводит разделитель тысяч
`Percent`	Отображает число в виде процентов и выводит две цифры справа от десятичного разделителя
`Scientific`	Использует формат с плавающим десятичным разделителем
`Yes/No`	Отображает `No`, если число равно 0, и `Yes` — в противном случае
`True/False`	Отображает `False`, если число равно 0, и `True` — в противном случае
`On/Off`	Отображает `Off`, если число равно 0, и `On` — в противном случае

***Таблица 3.2.** Именованные форматы даты и времени*

Имя формата	Описание
`General Date`	Выводит дату или время. Если нет дробной части, то выводит только дату
`Long Date`	Выводит дату в соответствии с полным форматом Windows для даты
`Medium Date`	Выводит дату в соответствии с обычным форматом Windows для даты
`Short Date`	Выводит дату в соответствии с сокращенным форматом Windows для даты
`Long Time`	Выводит часы, минуты и секунды
`Medium Time`	Выводит часы и минуты в 12-часовом формате
`Short Time`	Выводит часы и минуты в 24-часовом формате

Например, в результате действия приводимого далее кода (листинг 3.10, см. также файл *7-Форматирование числа.xlsm* на компакт-диске) в окно **Immediate** будут выведены отформатированные значения (рис. 3.9).

Листинг 3.10. Примеры именованных форматов

```
Sub Frm()
  Dim x As Double
  x = 4654646.544564
```

```
  Debug.Print "General Number", Format(x, "General Number")
  Debug.Print "Currency",, Format(x, "Currency")
  Debug.Print "Fixed",, Format(x, "Fixed")
  Debug.Print "Standard",, Format(x, "Standard")
  Debug.Print "Percent",, Format(x, "Percent")
  Debug.Print "Scientific",, Format(x, "Scientific")
  Debug.Print "Yes/No",, Format(x, "Yes/No")
  Debug.Print "True/False",, Format(x, "True/False")
  Debug.Print "On/Off",, Format(x, "On/Off")
  Debug.Print "General Date",, Format(Now, "General Date")
  Debug.Print "Long Date",, Format(Now, "Long Date")
  Debug.Print "Medium Date",, Format(Now, "Medium Date")
  Debug.Print "Short Date",, Format(Now, "Short Date")
  Debug.Print "Long Time",, Format(Now, "Long Time")
  Debug.Print "Medium Time",, Format(Now, "Medium Time")
  Debug.Print "Short Time",, Format(Now, "Short Time")
End Sub
```

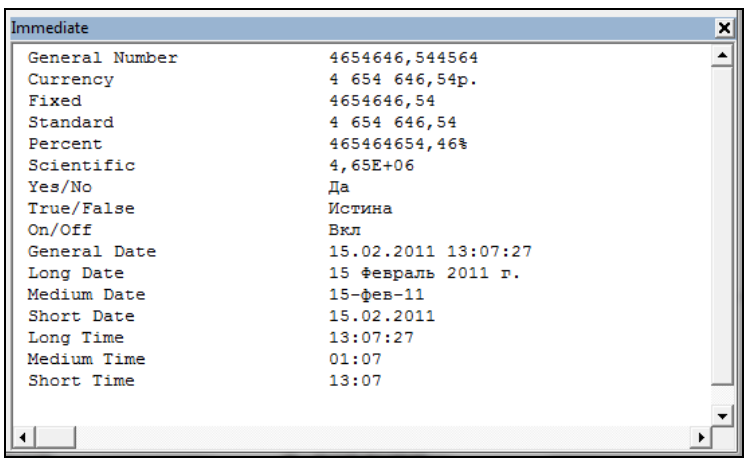


Рис. 3.9. Отформатированные значения в окне **Immediate**

Пользовательский формат

В MS Excel имеется возможность самому определить необходимый формат представления чисел. *Пользовательский формат* можно задать следующим образом:

- выделить необходимый диапазон ячеек;
- воспользоваться командой **Формат | Формат ячейки |** вкладка **Число**, которая расположена в группе команд **Ячейки** на вкладке **Главная** ленты (либо контекстным меню выделенной области);
- в открывшемся окне **Формат ячеек** в списке **Числовые форматы** выбрать тип **Все форматы**, в появившемся поле **Тип** можно задать необходимый пользовательский формат (рис. 3.10).

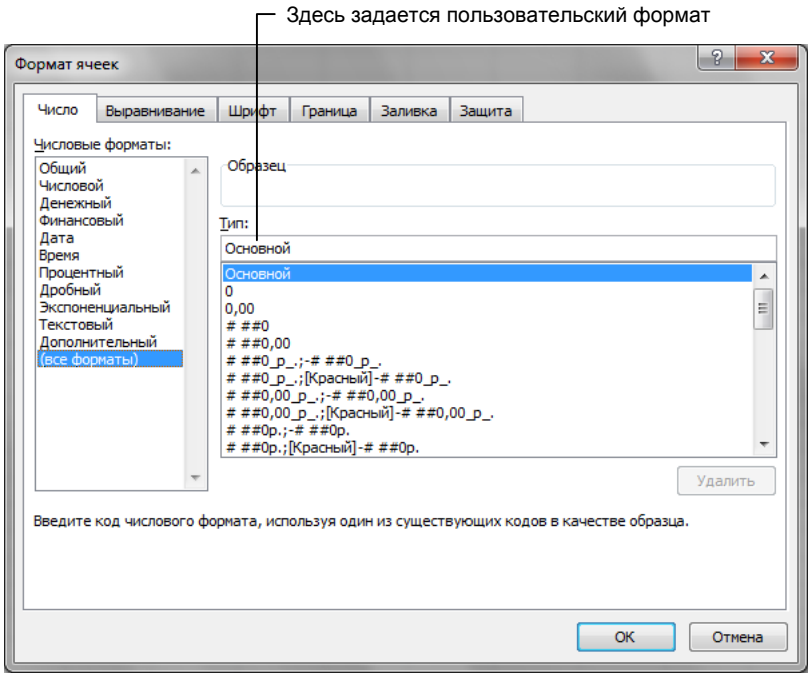


Рис. 3.10. Задание пользовательского формата данных

Пользовательские форматы могут состоять из 4-х секций, разделенных точкой с запятой (`;`):

- ❒ *положительный формат* (для положительных чисел);
- ❒ *отрицательный формат* (для отрицательных чисел);
- ❒ *нулевой формат* (для нуля);
- ❒ *формат текста* (для текста).

При задании пользовательских форматов можно использовать символы, указанные в табл. 3.3.

Таблица 3.3. *Символы пользовательских форматов*

Символ форматирования	Функция
Основной	Использует формат по умолчанию в неформатируемых ячейках
`#`	Для указания цифр. Незначащие нули не отображаются. Десятичная дробь округляется до числа символов `#` справа от запятой. Например, 7,8 в формате `#_###,#р.` отобразится как `7,8р.`
`0`	Используется как заполнитель для цифр. Отображает 0 при отсутствии цифры. Десятичная дробь округляется до заданного числа нулей справа от запятой

Таблица 3.3 *(окончание)*

Символ форматирования	Функция
?	Действует как заполнитель для цифр таким же способом, как 0. Незначащие нули замещаются пробелами, так что числа выравниваются правильно. Этот символ используется в дробях с меняющимся числом цифр, чтобы разделить выравнивание. 10,25 в формате `#''??/??` отобразится как `10 1/4`, а 10,3 отобразится как `10 1/3`, но при вводе в столбец разделители в этих значениях будут один под другим
_ (символ подчеркивания)	Делает пропуск шириной в символ справа от него. В комбинации с правой скобкой `_)` используется в форматах положительных чисел для выравнивания их с отрицательными, которые также заключаются в скобки
Десятичный разделитель `.` или `,` (устанавливается в Панели управления Windows, команда **Язык и региональные стандарты**)	Отличает положение запятой в десятичном числе, используйте 0 перед запятой для отображения 0 целых
Разделитель групп разрядов (задается в Панели управления Windows, значок **Язык и региональные стандарты —** в последних версиях ОС Windows)	Разделяет группы разрядов в числе. Необходимо отметить только первое его положение. Часто используется пробел
`%`	Умножает число на 100 и отображает его как процент от единицы со знаком `%`. При вводе в заранее отформатированную ячейку умножения нет
`E-E+e+e-e+` (латинский шрифт)	Отображает число в экспоненциальной форме. Число 0 или # от `E` (или `e`) определяет число знаков степени
`:p.-+()`	Отображает эти символы в том же месте форматируемого числа
/	Разделитель в простых дробях. Вводится целое с последующей дробью 1 1/5, чтобы получить отображение в таком же виде
\	Отображает как текст один следующий за ней специальный символ или одну цифру
""	Отображает текст, заданный в кавычках
*	Заполняет остаток ширины ячейки символом, следующим за * (одна * на формат)
@	Указывает место в формате, где будет отображен введенный текст
[цвет]	Форматирует содержимое ячеек заданным цветом
[значение условия]	Задает внутри числового формата условие, при котором будет применяться данный формат: <, >, =, <=, >= и <>. Значением может быть любое число

Чтобы скрыть числа с помощью формата пользователя, не следует указывать никакого формата между знаками точки с запятой. Скрытые числа присутствуют на рабочем листе и могут быть использованы другими формулами. При выделении ячейки они отображаются в строке формул.

Для скрытия нулей можно поступить следующим образом:

- создать пользовательский формат, установив белый цвет шрифта при выводе нуля;
- либо применить функцию `ЕСЛИ()`;
- либо скрыть нули на всем листе (перейдите на вкладку **Файл** ленты и выберите команду **Параметры**; в открывшемся окне **Параметры Excel** выберите слева категорию **Дополнительно**, а справа в группе **Показывать параметры для следующего листа** снимите флажок **Показывать нули в ячейках, которые содержат нулевые значения**). Например:

```
=ЕСЛИ(A1+B3=0; " "; A1+B3).
```

Рекомендации по созданию пользовательских форматов даты и времени можно найти в табл. 3.4.

Таблица 3.4. *Создание пользовательского формата даты и времени*

Тип/Символ	Результат отображения
Дни	
Д	Число от 1 до 31 без нуля впереди
ДД	Число от 1 до 31 с нулем
ДДД	Дни недели в сокращенном отображении (пн, ..., вс)
ДДДД	Дни недели в полном отображении
Месяцы	
М	Номер месяца без нуля
ММ	Номер месяца с нулем
МММ	Сокращенное название месяца (янв, ..., дек)
ММММ	Полное название месяца
Годы	
ГГ	От 00 до 99
ГГГГ	Полное число лет
Часы	
Ч	Число часов от 0 до 24 без нуля
ЧЧ	Число часов от 0 до 24 с нулем
Секунды	
С	Число секунд от 0 до 59 без нуля
СС	Число секунд от 0 до 59 с нулем впереди

Таблица 3.4 *(окончание)*

Тип/Символ	Результат отображения
Секунды	
`[]`	Часы, превышающие 24, минуты, превышающие 59, или секунды, превышающие 59
`AM/PM, A/P`	Отображает часы в 12-часовой системе
Разделители	
`-, *, /, :`	Помещает между элементами даты нужный разделитель

Например:

- `ДДДД` — Понедельник;
- `ММММ Д, ГГГГ` — Август 16, 2004;
- `[Синий] Д МММ, ГГ` — 16 Авг, 04 (синим цветом).

Заголовки, включающие текущую дату, можно создавать, используя конкатенацию текста с функцией `ТЕКСТ()` (рис. 3.11):

```
="Сегодня " & ТЕКСТ(ТДАТА();"Д МММ ГГГГ")
```

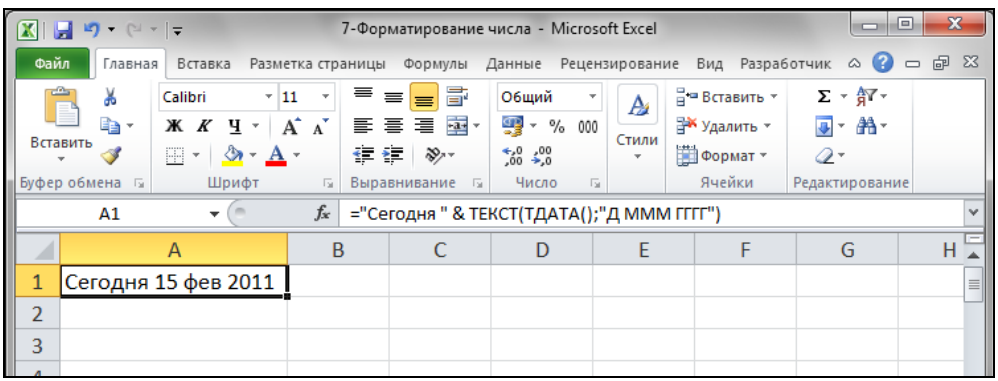


Рис. 3.11. Отображение текущей даты

Функция `ТДАТА()` обновляет текущую дату, когда бы ни открывался рабочий лист.

Итак:

- при задании формата для указания цифр применяются символы `#` и `0`;
- указанием `0` на экране отображаются незначащие нули, а при `#` незначащие нули отбрасываются;
- пробел при задании формата служит для представления разделителей тысяч;
- с помощью запятой (`,`) задается количество знаков до и после десятичной запятой;
- минус (-) перед числом означает ввод отрицательных чисел;
- для выделения числовых значений можно использовать цвет: [черный], [синий], [циан] [голубой], [фиолетовый], [красный], [белый], [желтый];
- в качестве разделителей можно применять дефис (-), косую черту и, как уже указывалось, пробел;

- кавычки ("") отображают текст, заданный в кавычках;
- для задания условия используются условные операторы: <, >, =, <=, >=, <>. Условные операторы, которые содержат код условия и числовое значение, указываются в квадратных скобках;
- для разделения формата используются точка с запятой (;);
- аналогично представляются денежные форматы (с дополнительными символами валюты).

В табл. 3.5 приведены примеры пользовательских форматов.

Таблица 3.5. *Примеры пользовательских форматов*

Формат	Форма представления содержимого ячейки
`0`	Отбрасывается дробная часть числа, т. е. производится округление до ближайшего целого
`# ##0`	Отбрасывается дробная часть числа и есть разделение тысяч
`0,00`	2 знака после запятой в любом случае
`# ##0,00; [красный] - # ##0,00`	Число с разделителем тысяч и два знака после запятой в любом случае; отрицательные числа — красные
`#,##'C`	Формат для отображения чисел в градусах Цельсия
`"N"####-###; "Минус запрещен"; "Введите число"`	Формат для отображения дополнительного текста. В этом формате число 7893,152 отобразится как `№7893-152`. Ввод отрицательного числа отобразит текст: "Минус запрещен", а ввод нуля — "Введите число"

Что касается использования пользовательских форматов при написании программ на VBA, то вы также можете использовать специальные символы, представленные в табл. 3.3. В табл. 3.6 приведен ряд примеров применения пользовательских форматов с функцией `Format()`.

Таблица 3.6. *Примеры пользовательских форматов*

Формат	Результат
`Format(1.2 ^ 2, "##.###")`	`1.44`
`Format(1.2 ^ 2, "##.000")`	`1.440`
`Format(Sin(1) * Exp(5), "#.###e+##")`	`1.249e+2`
`Format(Now, "hh:mm:ss")`	`18:57:23`
`Format(Now, "dd/mm/yyyy")`	`20.01.2002`

Форматирование чисел

Для форматирования чисел в VBA имеется специализированная функция `FormatNumber()`.

```
FormatNumber(Expression[, NumDigitsAfterDecimal [,IncludeLeadingDigit
[,UseParensForNegativeNumbers [,GroupDigits]]]])
```

- *Expression* — обязательный параметр, указывающий форматируемое числовое выражение.
- *NumDigitsAfterDecimal* — необязательный параметр, задающий количество знаков, отображаемых после десятичного разделителя. Допустимыми значениями являются следующие константы: `vbTrue`, `vbFalse` и `vbUseDefault`.
- *IncludeLeadingDigit* — необязательный параметр, указывающий, надо ли отображать нулевую целую часть. Допустимыми значениями являются следующие константы: `vbTrue`, `vbFalse` и `vbUseDefault`.
- *UseParensForNegativeNumbers* — необязательный параметр, определяющий, надо ли отрицательные числа отображать в скобках. Допустимыми значениями являются следующие константы: `vbTrue`, `vbFalse` и `vbUseDefault`.
- *GroupDigits* — необязательный параметр, определяющий, надо ли цифры группировать. Допустимыми значениями являются следующие константы: `vbTrue`, `vbFalse` и `vbUseDefault`.

В табл. 3.7 приведен ряд примеров использования функции `FormatNumber()`.

***Таблица 3.7.** Примеры использования функции `FormatNumber()`*

Формат	Результат
`FormatNumber(Sin(4), 3)`	`-0.757`
`FormatNumber(Sin(4), 3, vbTrue)`	`-0.757`
`Debug.Print (FormatNumber(Sin(4), 3, vbFalse))`	`-.757`
`FormatNumber(Sin(4), 3, vbFalse, vbTrue)`	`(.757)`
`FormatNumber(Sin(4), 3, vbFalse, vbFalse)`	`-.757`

Форматирование процентов

Для форматирования процентов в VBA служит специализированная функция `FormatPercent()`, которая имеет тот же самый синтаксис, что и функция `FormatNumber()`.

Например, следующие две инструкции

```
x = 0.2342
Debug.Print (FormatPercent(x, 2))
```

выведут в окно **Immediate** `23.42%`, т. е. число, записанное в формате процентов, у которого после десятичной точки отображаются две цифры.

Денежный формат

Для вывода данных в денежном формате в VBA служит специализированная функция `FormatCurrency()`, которая имеет тот же самый синтаксис, что и функция `FormatNumber()`. Например: `FormatCurrency(12312.3453, 2)` возвращает `12 312.35р.`

Форматирование даты и времени

Для форматирования даты и времени в VBA имеется специализированная функция `FormatDateTime()`.

`FormatDateTime(Date[, NamedFormat])`

- *Date* — обязательный параметр, задающий форматируемую дату.
- *NamedFormat* — необязательный параметр, указывающий тип форматирования. Допустимыми значениями являются следующие константы: `vbGeneralDate`, `vbLongDate`, `vbShortDate`, `vbLongTime`, `vbShortTime`.

Например, данная строка была написана 15 апреля 2011 года в 18 часов 15 минут. Поэтому:

- `FormatDateTime(Now, vbShortTime)` возвращает `18:15`;
- `FormatDateTime(Now, vbShortDate)` возвращает `15.04.2011`.

Условное форматирование

В Excel 2010 для пользователя предоставлены широкие возможности по осуществлению *форматирования с учетом условий* (ограничений на число, процент, формула, процентиль), которое предлагает пользователю различные возможности по цветовому отображению вводимых значений, а также использованию цветовых шкал и набора значков. Окно **Диспетчер правил условного форматирования** (рис. 3.12) для задания ограничений и управления правилами форматирования можно открыть следующим образом: перейдите на вкладку **Главная** ленты и в группе команд **Стили** выберите из выпадающего списка **Условное форматирование** команду **Управление правилами**. Условное форматирование предлагает следующие возможности:

- упрощенную процедуру создания пользовательских форматов;
- большой выбор элементов форматирования;
- возможность указать в формате необходимое количество условий;
- в качестве условий можно использовать собственные формулы, принимающие логическое значение `ИСТИНА`/`ЛОЖЬ`;
- можно указать, что подвергается проверке — значение в ячейке или в формуле;
- для сравнения со значениями в ячейке можно задавать как числа, так и ссылки на другие ячейки.

Условные форматы особенно ценны для контроля ошибок в результате анализа.

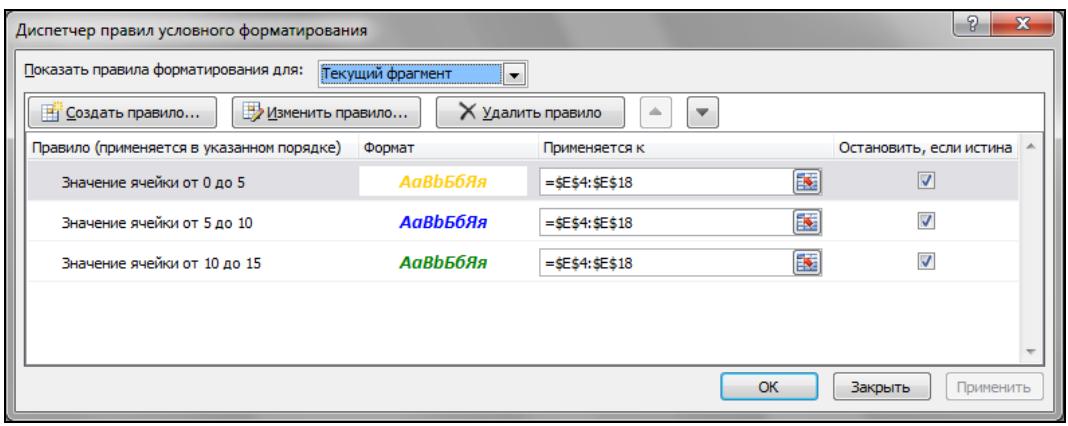


Рис. 3.12. Окно **Диспетчер правил условного форматирования**

Форматирование рабочих листов

Кроме форматирования числовых данных, MS Excel предоставляет возможности общего форматирования данных, находящихся в ячейках или таблицах — выравнивание текста, выбор шрифтов, рамок и цвета фона и т. п. Для оформления диапазона ячеек удобно использовать команды, которые расположены в группе команд **Стили** на вкладке **Главная** ленты. С другой стороны, конечно, можно и "вручную" отформатировать каждую требуемую ячейку, используя различные команды вкладки **Главная** ленты.

Рассмотрим несколько примеров, связанных с форматированием данных и элементов рабочего листа средствами VBA.

Автоматическое переоформление таблицы при изменении в ней значений

Событие `Change` объекта `Worksheet` генерируется при изменении значений в диапазоне рабочего листа. Это событие позволяет автоматизировать переоформление таблицы при вводе в таблицу новых значений. Рассмотрим небольшой пример. Пусть в диапазоне **B2:B13** расположены данные по объему продаж картофеля фирмой "Родные просторы". Необходимо выделить полужирным красным шрифтом те данные, которые соответствуют максимальному объему продаж, синим же цветом — минимальному. Все остальные данные выводятся черным шрифтом. Кроме того, ячейки, объем продаж в которых превышает средний объем, надо залить желтым цветом. Причем требуется обеспечить автоматическое переформатирование таблицы при изменении значений в ее ячейках (рис. 3.13). Для решения этой задачи мы и будем использовать событие `Change` объекта `Worksheet` (модуль рабочего листа `Лист1` файла *8-Автоматическое переоформление таблицы при изменении в ней значений.xlsm* на компакт-диске).

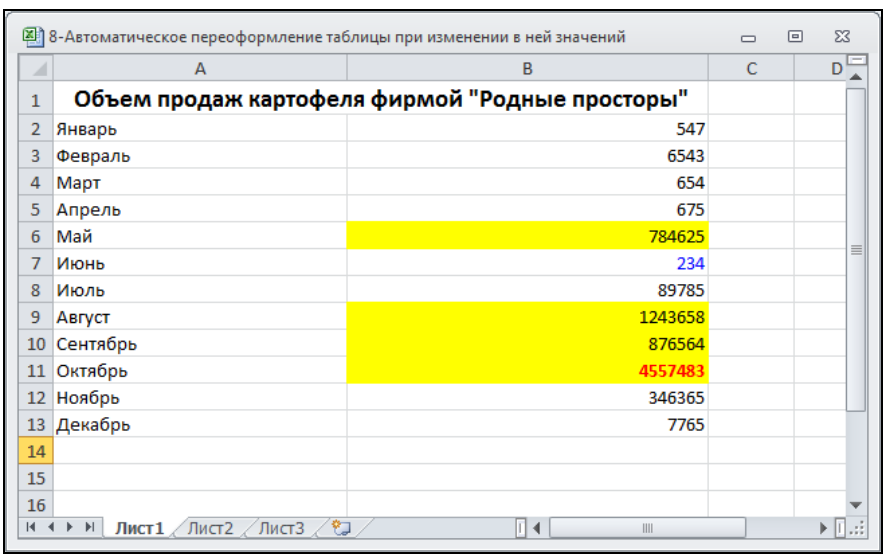


Рис. 3.13. Автоматическое переоформление таблицы при изменении в ней значений

Управление стилем границы диапазона и объекта *Border*

Свойство `Borders` объекта `Range` возвращает семейство `Borders`, элементы которого инкапсулируют данные об одной из граничных или диагональных линий данного диапазона. Допустимыми значениями индекса семейства `Borders` могут быть следующие константы `XlBordersIndex`: `xlDiagonalDown`, `xlDiagonalUp`, `xlEdgeBottom`, `xlEdgeLeft`, `xlEdgeRight`, `xlEdgeTop`, `xlInsideHorizontal` и `xlInsideVertical`. Каждая из этих границ представляет собой объект `Border`, свойства которого перечислены в табл. 3.8.

***Таблица 3.8.** Свойства объекта `Border`*

Свойство	Описание
`Color`	Цвет границы, заданный при помощи RGB-модели
`ColorIndex`	Цвет границы, заданный индексом соответствующего элемента цветовой палитры
`LineStyle`	Задает стиль границы. Допустимыми значениями являются следующие константы `XlLineStyle`: `xlContinuous`, `xlDash`, `xlDashDot`, `xlDashDotDot`, `xlDot`, `xlDouble`, `xlSlantDashDot` и `xlLineStyleNone`
`Weight`	Устанавливает толщину границы. Допустимыми значениями являются следующие константы `XlBorderWeight`: `xlHairline`, `xlThin`, `xlMedium` и `xlThick`

Например, следующий код (листинг 3.11, см. также файл *9-Примеры форматирования на VBA.xlsm* на компакт-диске) задает верхнюю границу диапазона **A2:E2**, расположенного на листе 3, в виде толстой линии красного цвета, а его нижнюю границу — в виде зеленой пунктирной линии средней толщины.

Листинг 3.11. Конструирование границы специфицированного диапазона. Стандартный модуль

```
Sub DemoBorders()
   Dim rgn As Range
   Set rgn = Range("Лист3!A2:E2")
   With rgn.Borders(xlEdgeTop)
      .LineStyle = xlContinuous
      .Weight = xlThick
      .Color = RGB(255, 0, 0)
   End With
   With rgn.Borders(xlEdgeBottom)
      .LineStyle = xlDash
      .Weight = xlMedium
      .Color = RGB(0, 255, 0)
   End With
End Sub
```

Если все компоненты границы имеют одни и те же параметры, то для установки их значения можно воспользоваться не элементами, а всем семейством `Borders`, как это, например, делается в следующей инструкции для создания границы синего цвета у выделенной области.

```
Selection.Borders.Color = RGB(0, 0, 255)
```

Функции *RGB()* и *QBColor()*

Коды цветов в VBA часто задаются числами в форме шестнадцатеричной системы счисления. Вместо прямого указания шестнадцатеричного кода цвета, довольно часто цвет удобнее задавать, используя функции `RGB()` и `QBColor()`. Функция `RGB()` позволяет получить любой цвет смешением красного, зеленого и синего компонентов различной интенсивности.

```
RGB(Red, Green, Blue)
```

- *Red* — целое число из диапазона от 0 до 255, указывающее красный компонент цвета.
- *Green* — целое число из диапазона от 0 до 255, указывающее зеленый компонент цвета.
- *Blue* — целое число из диапазона от 0 до 255, указывающее синий компонент цвета.

В табл. 3.9 приведены значения параметров функции `RGB()` для получения стандартных цветов.

Таблица 3.9. *Значения параметров функции `RGB()` для получения стандартных цветов*

Цвет	Red	Green	Blue
Черный	0	0	0
Синий	0	0	255
Зеленый	0	255	0
Голубой	0	255	255
Красный	255	0	0
Розовый	255	0	255
Желтый	255	255	0
Белый	255	255	255

Функция `QBColor()` возвращает шестнадцать основных цветов в зависимости от значения параметра (табл. 3.10).

```
QBColor(color)
```

Здесь параметр *color* принимает целые числа из диапазона от 0 до 15.

Таблица 3.10. *Соответствие между цветами и значением параметра функции* `QBColor()`

Число	Цвет	Число	Цвет
0	Черный	8	Серый
1	Синий	9	Светло-синий
2	Зеленый	10	Светло-зеленый
3	Голубой	11	Светло-голубой
4	Красный	12	Светло-красный
5	Розовый	13	Светло-розовый
6	Желтый	14	Светло-желтый
7	Белый	15	Насыщенный белый

Объект *Characters* (как форматировать часть содержимого ячейки)

Свойство `Characters` объекта `Range` возвращает объект `Characters`, представляющий собой строку указанной длины от указанного символа. Часто применяется, когда надо форматировать содержимое не всей ячейки, а только ее часть.

`Characters(`*`Start`*`, `*`Length`*`)`

- *`Start`* — необязательный параметр, задающий номер первого возвращаемого символа из данной строки.
- *`Length`* — необязательный параметр, указывающий число возвращаемых символов.

В следующем примере (листинг 3.12, см. также файл *9-Примеры форматирования на VBA.xlsm* на компакт-диске) в ячейку **A1** выводится строка `"Андрей Гарнаев and Лада Рудикова"`, причем первая часть этой строки (`"Андрей Гарнаев"`) выводится полужирным шрифтом зеленого цвета высотой 16 пт, вторая ее часть (`"and"`) — курсивным шрифтом черного цвета высотой 12 пт, а третья (`"Лада Рудикова"`) отображается полужирным шрифтом красного цвета высотой 16 пт (рис. 3.14).

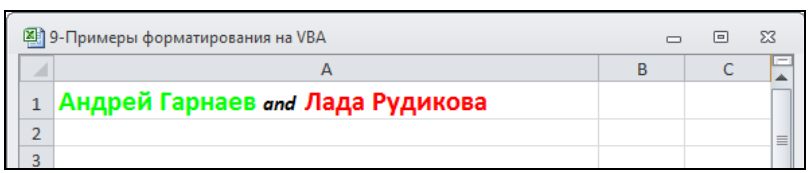


Рис. 3.14. Форматирование части содержимого ячейки

Листинг 3.12. Форматирование части содержимого ячейки

```
Sub CharColor()
  With Range("A1")
   .Value = "Андрей Гарнаев and Лада Рудикова"
```

```
    .Characters(1, 14).Font.Bold = True
    .Characters(1, 14).Font.Size = 16
    .Characters(1, 14).Font.Color = RGB(0, 255, 0)

    .Characters(16, 18).Font.Italic = True
    .Characters(16, 18).Font.Size = 12
    .Characters(16, 18).Font.Color = RGB(0, 0, 0)

    .Characters(20, 32).Font.Bold = True
    .Characters(20, 32).Font.Italic = False
    .Characters(20, 32).Font.Size = 16
    .Characters(20, 32).Font.Color = RGB(255, 0, 0)
  End With
End Sub
```

Объект *Font* (задание шрифта)

Свойство `Font` объекта `Range` возвращает объект `Font`, представляющий собой шрифт. В табл. 3.11 приведены свойства объекта `Font`.

Таблица 3.11. *Свойства объекта* `Font`

Свойство	Описание
`Bold`	Определяет, является ли шрифт полужирным
`Color`	Задает цвет шрифта в соответствии с RGB-моделью
`ColorIndex`	Задает индексированный цвет в соответствии с текущей палитрой цветов
`FontStyle`	Определяет стиль шрифта, заданный в словесной форме. Допустимы значения: `Regular` (обычный), `Bold` (полужирный), `Italic` (курсив), `Bold Italic` (полужирный курсив)
`Italic`	Определяет, является ли шрифт курсивным
`Name`	Строка, указывающая имя шрифта, например `"Arial Cyr"`
`Size`	Размер шрифта
`Strikethrough`	Устанавливает, имеется ли линия по центру, как будто текст перечеркнут
`Superscript`	Устанавливает, используется ли текст как верхний индекс
`Subscript`	Устанавливает, используется ли текст как нижний индекс
`Underline`	Задает тип подчеркивания. Допустимы значения: `xlNone` (нет подчеркивания), `xlSingle` (одинарное, по значению), `xlDouble` (двойное, по значению), `xlSingleAccounting` (одинарное, по ячейке), `xlDoubleAccounting` (двойное, по ячейке)

Например, в следующем коде устанавливается для диапазона **A1:B2** полужирный шрифт красного цвета с высотой символов 14 пт.

```
With Range("A1:B2").Font
    .Size = 14
    .Bold = True
    .Color = RGB(255, 0, 0)
End With
```

Объект *Interior* (заливка диапазона)

Свойство `Interior` объекта `Range` возвращает объект `Interior`, инкапсулирующий данные о заливке диапазона. В табл. 3.12 приведены свойства объекта `Interior`.

***Таблица 3.12.** Свойства объекта `Interior`*

Свойство	Описание
`Color`	Задает цвет заливки в соответствии с RGB-моделью
`ColorIndex`	Задает индексированный цвет заливки в соответствии с текущей палитрой цветов
`Pattern`	Задает узор заливки. Допустимыми являются следующие значения: `xlPatternAutomatic`, `xlPatternChecker`, `xlPatternCrissCross`, `xlPatternDown`, `xlPatternGray16`, `xlPatternGray25`, `xlPatternGray50`, `xlPatternGray75`, `xlPatternGray8`, `xlPatternGrid`, `xlPatternHorizontal`, `xlPatternLightDown`, `xlPatternLightHorizontal`, `xlPatternLightUp`, `xlPatternLightVertical`, `xlPatternNone`, `xlPatternSemiGray75`, `xlPatternSolid`, `xlPatternUp`, `xlPatternVertical`
`PatternColor`	Задает цвет узора в соответствии с RGB-моделью
`PatternColorIndex`	Задает индексированный цвет узора в соответствии с текущей палитрой цветов

В следующем примере (листинг 3.13, см. также файл *9-Примеры форматирования на VBA.xlsm* на компакт-диске) устанавливается красная заливка с синим клетчатым узором для диапазона **A1:D5** на листе 2 (рис. 3.15).

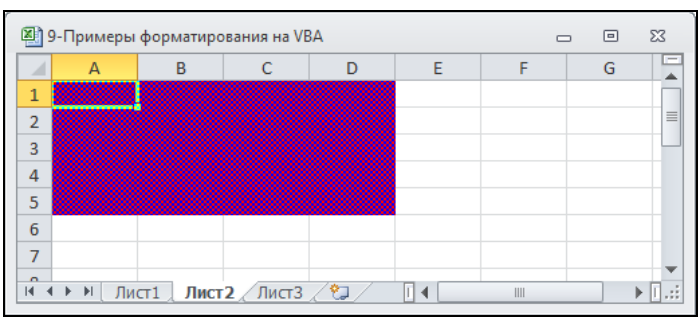


Рис. 3.15. Заливка диапазона

Листинг 3.13. Заливка диапазона

```
Sub Inter()
With Range("Лист2!A1:D5")
   .Interior.Color = RGB(255, 0, 0)
   .Interior.Pattern = xlPatternChecker
   .Interior.PatternColor = RGB(0, 0, 255)
End With
End Sub
```

Отмена заливки диапазона

Для отмены заливки диапазона надо установить значение свойства `ColorIndex` объекта `Interior` равным `xlNone`, например, как это делается в следующей инструкции для ячейки **A1**:

```
Range("A1").Interior.ColorIndex = xlNone
```

Установка числового формата

Свойство `NumberFormat` объекта `Range` устанавливает числовой формат. Например, представленные далее инструкции устанавливают:

- в ячейке **A1** — общий формат,
- в ячейке **A2** — числовой формат, отображающий три знака после десятичной точки, например `12.000`;
- в ячейке **A3** — формат времени с двоеточием в качестве разделителя и по два знака, отведенных под часы, минуты и секунды, например `02:12:55`;
- в ячейке **A4** — формат даты, причем два знака отводится под день, три буквы — под месяц и четыре цифры — под год, например `01 фев 2011`.

```
Range("A1").NumberFormat = "General"
Range("A2").NumberFormat = "0.000"
Range("A3").NumberFormat = "hh:mm:ss"
Range("A4").NumberFormat = "d mmm yyyy"
```

Задание угла, под которым выводится текст в диапазоне

Свойство `Orientation` объекта `Range` устанавливает угол, под которым выводится текст в диапазоне. Допустимыми значениями являются либо угол поворота текста в градусах от –90 до 90°, либо одна из следующих постоянных:

- `xlDownward` — выравнивание по левому краю сверху вниз, соответствует углу –90°;
- `xlHorizontal` — выравнивание по горизонтали, соответствует нулевому углу;
- `xlUpward` — выравнивание по правому краю снизу вверх, соответствует углу 90°;
- `xlVertical` — выравнивание по вертикали, нет соответствия в градусах.

Например, в следующем коде (листинг 3.14, см. также файл *9-Примеры форматирования на VBA.xlsm* на компакт-диске) в ячейке **A1** текст выводится под углом 45°, а в ячейке **B1** — под углом –45° (рис. 3.16).

Листинг 3.14. Задание угла вывода текста

```
Sub Orient()
  Range("Лист4!A1:B1").Font.Size = 16
  Range("Лист4!A1").Orientation = 45
  Range("Лист4!A1").Value = "Вверх"
  Range("Лист4!B1").Orientation = -45
  Range("Лист4!B1").Value = "Вниз"
End Sub
```

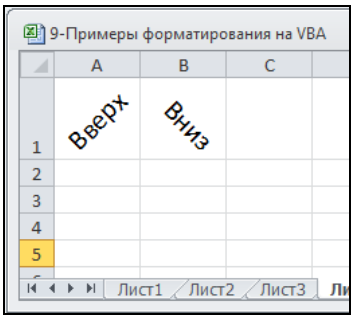


Рис. 3.16. Задание угла вывода текста

Работаем с формулами

Формулы в MS Excel предназначены для выполнения вычислений и анализа данных. Существует несколько основных характеристик для любой формулы:

- первым символом обязательно является знак равенства (=);
- результат вычисления формулы выводится в ячейке таблицы;
- в строке формул отображается формула, содержащаяся в активной ячейке;
- результат обновляется автоматически при изменении значений в ячейках, на которые ссылается формула (если работать в режиме автоматических вычислений).

При работе с формулами в MS Excel можно выбирать один из трех режимов:

- автоматический (по умолчанию);
- автоматический, кроме таблиц;
- вручную.

Для переключения режимов вычислений следует воспользоваться необходимой командой, выбрав ее из списка **Параметры вычислений**, расположенного в группе **Вычисление** на вкладке **Формулы** ленты. Для проверки установки режима вычислений *вручную* следует открыть новую рабочую книгу, выделить, например, ячейку **A3**, затем ввести в нее формулу `=A1+A2` и нажать клавишу <Enter> — в ячейке **A3** должен появиться 0. Если же 0 не появляется, то установлен режим вычислений вручную.

Ссылки на ячейки в формулах

Ссылки делают формулы более удобными, т. к. дают возможность использовать данные, находящиеся в нескольких ячейках, таблицах и рабочих книгах. Ссылки могут быть использованы для идентификации как отдельных ячеек, так и групп ячеек.

Ранее мы рассмотрели два стиля ссылок на ячейки: A1 и R1C1. При использовании ссылок в формулах их имена можно вводить с клавиатуры или выделять с помощью мыши нужные ячейки.

Что касается объектов `Range` и `Selection`, заметим, что в иерархии MS Excel объект `Range` *(диапазон)* идет сразу после объекта `Worksheet`. Объект `Range` является одним из ключевых объектов VBA. Объект `Selection` возникает в VBA двояко — либо как результат работы метода `Select`, либо при вызове свойства `Selection`. Тип получаемого объекта зависит от типа выделенного объекта. Чаще всего объект `Selection` принадлежит классу `Range`, и при работе с ним можно использовать свойства и методы объекта `Range`. Объект `Range` возвращается либо как элемент семейств `Range` или `Cells`, либо свойствами `Range`, `Cells` и `Offset`, либо методами `ActiveCell`, `Intersect` и `Union`.

При обращении к ячейке возможны относительная, абсолютная адресация и их комбинации (смешанная адресация).

- *Относительная адресация* основана на том, что ссылки на ячейки создаются с учетом позиции ячейки, содержащей формулу, т. е. при копировании формулы в созданную ячейку ссылки в каждой копии изменяются таким образом, чтобы сохранялись те же соотношения, что и в исходной формуле. Например, либо **A1** или **C2** — если стиль адресации A1, либо **R1C1** или **R2C3** — если стиль адресации R1C1.
- При копировании формул с *абсолютной адресацией* ссылки сохраняются (ссылка всегда указывает на одну и ту же ячейку). Ссылка на ячейку при абсолютной адресации содержит номер строки и букву столбца, перед которыми стоит знак доллара. Например, `$A10`, `A$10` и `$A$10` задают абсолютную адресацию на столбец **A**, строку **10** и ячейку **A10** соответственно. С другой стороны, если используется стиль ссылок R1C1, то абсолютная адресация определяется следующим образом: если активной ячейкой является **R2C3**, то `R[1]C[-1]` дает ссылку на ячейку **R3C2**.
- Иногда бывает необходимо, чтобы при копировании не менялась только строка или только столбец. В этих случаях используется *смешанная адресация*, которая содержит как абсолютные, так и относительные ссылки.

Клавиша <F4> при редактировании в формулах позволяет делать шаг в цикле всех комбинаций относительных и абсолютных ссылок.

Ссылка на другие листы рабочей книги или на другие рабочие книги

Ссылка на другие листы данной рабочей книги осуществляется путем включения в формулу ссылки на лист:

```
Лист5!A1
```

причем `!` (восклицательный знак) обязателен. Если имя листа содержит пробелы, нужно заключить ссылку на лист *в кавычки*.

Внешние ссылки — это ссылки на ячейки, находящиеся в других рабочих книгах, которые обязательно включают имя рабочей книги, заключенное в *прямоугольные скобки*:

```
[Книга1]Лист3!$B$4
```

Трехмерные ссылки (3D) состоят из диапазона листов с указанием первого и последнего и диапазона ячеек с указанием тех из них, на которые делается ссылка:

```
=СУММ(Лист1:Лист6!$E$1:$E$6)
```

В этой формуле суммируются значения в диапазоне ячеек **$E$1:$E$6** на каждом из листов с **Лист1** по **Лист6**.

Трехмерные ссылки можно использовать в следующих встроенных функциях MS Excel:

- ДИСП (VAR);
- ДИСПР (VARP);
- МАКС (MAX);
- МИН (MIN);
- ПРОИЗВЕД (PRODUCT);
- СРЗНАЧ (AVERAGE);
- СТАНДОТКЛ (STDEV);
- СТАНДОТКЛОНП (STDEVP);
- СУММ (SUM);
- СЧЁТ(COUNT);
- СЧЁТА (COUNTA).

В формулах удобно в качестве адресов применять имена (как на отдельные ячейки, так и на диапазоны ячеек).

Задание групп строк и столбцов

Если в диапазоне указываются только имена столбцов или строк, то объект `Range` задает диапазон, состоящий из указанных столбцов или строк. Например, `Range("A:C")` задает диапазон, состоящий из столбцов **A**, **B** и **C**, а `Range("2:2")` — из второй строки. Другим способом работы со строками и столбцами являются свойства рабочего листа `Rows` и `Columns`, возвращающие семейства строк и столбцов. Например, столбцом **A** является `Columns(1)`, а второй строкой `Rows(2)`.

Связь объекта *Range* и свойства *Cells* объекта *Worksheet*

Ячейка — это частный случай диапазона, который состоит из единственной ячейки. Поэтому естественно, что объект `Range` позволяет работать как с диапазоном ячеек, так и с одной ячейкой.

Альтернативным способом работы с ячейкой является свойство `Cells` объекта `Worksheet`. Например, ячейку **A2** как объект можно описать двумя равносильными способами: `Range("A2")` и `Cells(1, 2)`.

В свою очередь ячейка, возвращаемая свойством `Cells`, используемым как параметр объекта `Range`, позволяет записывать диапазон в альтернативном виде, который иногда удобен для работы. В качестве примера этой формы записи диапазона приведем следующие две инструкции, возвращающие один и тот же диапазон:

```
Range("A2:C3")
Range(Cells(1,2), Cells(3,3))
```

Примечание

Диапазон так же, как и рабочий лист, обладает свойством `Cells`, которое, если используется без параметров, возвращает множество всех ячеек, входящих в диапазон. Если же оно используется с параметрами, то возвращает конкретную ячейку из диапазона. В следующем примере значение `2` вводится в ячейку **C3**:

```
Range("B2:D4").Select
Selection.Cells(2, 2).Value = 2
```

Свойства объекта *Range*

Объект `Range` позволяет сочетать гибкость VBA и мощь рабочего листа. Огромное число встроенных функций рабочего листа существенно упрощают и делают

более наглядным программирование на VBA. Свойства объекта `Range` позволяют управлять им от внешнего вида до автоматизации вычислений. Основными свойствами объекта `Range` являются следующие:

- `Address`;
- `AllowEdit`;
- `Areas`;
- `Borders`;
- `Cells`;
- `Characters`;
- `Column`;
- `Columns`;
- `ColumnWidth`;
- `Comment`;
- `Count`;
- `CurrentRegion`;
- `End`;
- `EntireColumn`;
- `EntireRow`;
- `Font`;
- `Formula`;
- `FormulaArray`;
- `FormulaHidden`;
- `FormulaLocal`;
- `FormulaR1C1`;
- `FormulaR1C1Local`;
- `HasFormula`;
- `Height`;
- `Hidden`;
- `HorizontalAlignment`;
- `Hyperlinks`;
- `Interior`;
- `Left`;
- `Locked`;
- `Name`;
- `NumberFormat`;
- `Offset`;
- `Orientation`;
- `Resize`;
- `Row`;
- `RowHeight`;
- `Rows`;
- `ShrinkToFit`;
- `Top`;
- `UseStandardHeight`;
- `UseStandardWidth`;
- `Value`;
- `VerticalAlignment`;
- `Width`;
- `Worksheet`;
- `WrapText`.

Ввод или считывание значения из диапазона

Свойство `Value` объекта `Range` возвращает или устанавливает значение в ячейках диапазона. В первой инструкции данного примера переменной `x` присваивается значение из ячейки **C1**, во второй — в ячейку **C3** вводится строка `"Отчет"`, а в третьей — в каждую из ячеек диапазона **A1:B2** вводится `1`.

```
x = Range("C1").Value
Range("C3").Value = "Отчет"
Range("A1:B2").Value = 1
```

Ввод в диапазон массива значений

Диапазон можно заполнять не только поячеечно, но и за одну операцию, присваивая свойству `Value` диапазона либо переменную типа `Variant`, как показано в первом примере (листинг 3.15, файл *10-Примеры некоторых свойств объекта Range.xlsm* на компакт-диске), либо непосредственно массив значений, как продемонстрировано во втором примере (листинг 3.16, файл *10-Примеры некоторых свойств объекта Range.xlsm* на компакт-диске).

Листинг 3.15. Ввод в диапазон массива значений. Первый пример

```
Sub DemoInput1()
   Dim s As Variant
   s = Array("1", "2")
   Range("Лист1!A1:B1").Value = s
End Sub
```

Листинг 3.16. Ввод в диапазон массива значений. Второй пример

```
Sub DemoInput2()
   Dim t(8, 8) As Integer
   Dim i As Integer
   Dim j As Integer
   For i = 1 To 9
      For j = 1 To 9
         t(i — 1, j — 1) = i * j
      Next
   Next
   Range(Cells(1, 1), Cells(9, 9)).Value = t
End Sub
```

Поиск по шаблону подобных значений в диапазоне

Последовательный перебор ячеек диапазона и сравнение с помощью оператора `Like` возвращаемых значений свойства `Value` с шаблоном позволяет реализовывать поиск подобных значений в диапазоне. Например, в следующем коде (листинг 3.17, см. также файл *10-Примеры некоторых свойств объекта Range.xlsm* на компакт-диске) последовательно просматриваются все ячейки диапазона **A1:A100**. В тех из них, в которые входит значение `MS`, содержимое ячейки заменяется словом `Microsoft`, сама же ячейка заливается желтым цветом, в то время как все остальные ячейки — белым цветом.

Листинг 3.17. Поиск в диапазоне подобных значений

```
Dim c As Range
For Each c In [A1:A100]
   If c.Value Like "*MS*" Then
      c.Value = "Microsoft"
      c.Interior.Color = RGB(255, 255, 0)
   Else
      c.Interior.Color = RGB(255, 255, 255)
   End If
Next
```

Ввод или считывание формулы в ячейку в формате A1

Свойство `Formula` объекта `Range` возвращает или устанавливает формулу в диапазон в формате A1. Например, в следующем примере первая инструкция вводит в ячейку **C1** формулу `=$A$1+$B$1`, а вторая — в ячейку **C2** формулу `=SIN(A2)^2`:

```
Range("C1").Formula = "=$A$1+$B$1"
Range("C2").Formula = "=SIN(A2)^2"
```

Ввод или считывание формулы в ячейку в формате R1C1

Свойство `FormulaR1C1` объекта `Range` возвращает формулу в формате R1C1. Например, следующая инструкция вводит в ячейку **B1** формулу `=2*R3C2` в формате R1C1, или, что эквивалентно, формулу `=2*$B$3` в формате A1:

```
Range("B1").FormulaR1C1 = "=2*R3C2"
```

Ввод или считывание формулы локальной версии в ячейку в формате A1

Свойство `FormulaLocal` объекта `Range` возвращает формулу локальной версии в формате A1. Например, следующая инструкция вводит в ячейку **B2** формулу `=СУММ(C1:C4)`:

```
Range("B2"). FormulaLocal = "=СУММ(C1:C4)"
```

Ввод или считывание формулы локальной версии в ячейку в формате R1C1

Свойство `FormulaR1C1Local` объекта `Range` возвращает формулу локальной версии в формате R1C1. Например, следующая инструкция вводит в ячейку **B2** формулу `=СУММ(C1:C4)` в формате R1C1:

```
Range("B2").FormulaR1C1Local = "=СУММ(R1C3:R4C3)"
```

Ввод формулы массива в диапазон

Свойство `FormulaArray` объекта `Range` возвращает формулу диапазона в формате A1. В отличие от обыкновенной формулы рабочего листа, формула массива вводится на рабочем листе не нажатием клавиши <Enter>, а комбинацией клавиш <Ctrl>+<Shift>+<Enter>. Например, следующая инструкция вводит в диапазон **E1:E3** формулу `{=A1:A3*3}`:

```
Range("E1:E3").FormulaArray = "=A1:A3*3"
```

Ввод формулы массива локальной версии в диапазон

При вводе формулы массива с функциями рабочего листа локальной версии формулу надо представить в формате R1C1, и вместо формулы локальной версии следует использовать формулу базовой версии. Например, следующая инструкция вводит в ячейку **D1** формулу `{=СУММ($A$1:$B$1*3)}`:

```
Range("D1").FormulaArray = "=SUM(R1C1:R1C2*3)"
```

Ввод формулы массива в диапазон с относительными ссылками на ячейки

Для ввода формулы с относительными ссылками на ячейки необходимо использовать относительную адресацию в формате R1C1. Например, ввод формулы `{=СУММ(A1:B1*3)}` в ячейку **D1** производится следующей инструкцией:

```
Range("D1").FormulaArray = "=SUM(RC[-3]:RC[-1]*3)"
```

Как узнать, спрятана ли формула на защищенном листе?

Свойство `FormulaHidden` объекта `Range` возвращает значение `True`, если формула спрятана на защищенном листе.

Как узнать, имеется ли в ячейке формула?

Свойство `HasFormula` объекта `Range` возвращает значение `True`, если во всех ячейках диапазона имеется по формуле, значение `False`, если формулы нет ни в одной из них, и значение `Null` — в оставшихся случаях. Например, следующий код проверяет наличие формулы в ячейке **C1** и, если таковой нет, вводит в эту ячейку формулу `=1`.

```
If Not Range("C1").HasFormula Then Range("C1").Formula = "=1"
```

Определение адреса ячейки

Свойство `Address` объекта `Range` возвращает адрес диапазона.

```
Address(RowAbsolute, ColumnAbsolute, ReferenceStyle, External, RelativeTo)
```

- *`RowAbsolute`* — необязательный параметр, принимающий логические значения. Если значение параметра равно `True` или параметр опущен, то возвращается абсолютная ссылка на строку.
- *`ColumnAbsolute`* — необязательный параметр, принимающий логические значения. Если его значение равно `True` или параметр опущен, то возвращается абсолютная ссылка на столбец.
- *`ReferenceStyle`* — необязательный параметр. Допустимы два значения — `xlA1` и `xlR1C1`, если `xlA1` или `xlR1C1` опущены, то возвращается ссылка в формате A1.
- *`External`* — необязательный параметр, принимающий логические значения и определяющий, является ли ссылка внешней.
- *`RelativeTo`* — необязательный параметр. В случае если значения параметров *`rowAbsolute`* и *`columnAbsolute`* равны `False`, а *`referenceStyle`* — `xlR1C1`, то данный параметр определяет начальную ячейку диапазона, относительно которой производится адресация.

Следующий код (листинг 3.18, см. также файл *10-Примеры некоторых свойств объекта Range.xlsm* на компакт-диске), обрабатывающий событие `SelectionChange` объекта `Worksheet`, демонстрирует возвращаемые свойством `Address` значения при различных установках его параметров. Например, если на рабочем листе будет выбрана ячейка **A1**, то на экране отобразится окно со следующим сообщением:

```
$A$1
$A1
R1C1
R[-1]C[-1]
```

Листинг 3.18. Свойство `Address`. Модуль рабочего листа

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
   MsgBox Target.Address() & vbCr & _
          Target.Address(RowAbsolute:=False) & vbCr & _
```

```
        Target.Address(ReferenceStyle:=xlR1C1) & vbCr & _
        Target.Address(ReferenceStyle:=xlR1C1, RowAbsolute:=False, _
                       ColumnAbsolute:=False, _
                       RelativeTo:=Worksheets(1).Cells(2, 2))
End Sub
```

Может ли ячейка быть отредактирована на рабочем листе?

Свойство "только для чтения" `AllowEdit` объекта `Range` возвращает значение `True`, если допустимо редактирование значений в указанном диапазоне, даже если на листе установлена защита.

Определения числа областей, из которых состоит данный диапазон

Свойство `Areas` объекта `Range` возвращает семейство `Areas` диапазонов, из которых состоит данный диапазон. Элементами семейства `Areas` является объект `Range`. Основное свойство этого семейства — `Count`, возвращающее число элементов семейства. Например, следующий код (листинг 3.19, файл *10-Примеры некоторых свойств объекта Range.xlsm* на компакт-диске), обрабатывающий событие `SelectionChange` объекта `Worksheet`, выводит в строку состояния число выделенных областей (рис. 3.17).

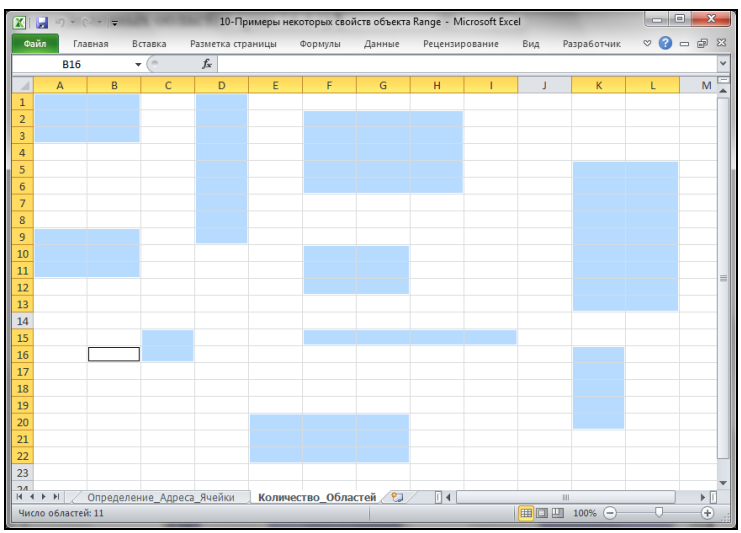


Рис. 3.17. Отображение в строке состояния числа выделенных областей

Листинг 3.19. Отображение в строке состояния числа выделенных областей. Модуль рабочего листа

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
   Application.StatusBar = "Число областей: " & Target.Areas.Count
End Sub
```

Операторы

Формула может содержать функции и математические операторы, порядок вычисления которых соответствует принятому в математике. Результатом вычисления формул, включающих арифметические операторы, являются числовые значения, а в случае операторов сравнения — логические значения ИСТИНА или ЛОЖЬ. В табл. 3.13 приведены математические операторы в формулах Excel.

Таблица 3.13. *Математические операторы в формулах Excel*

Оператор	Значение	Оператор	Значение
(	Открыть скобку	=	Равно
)	Закрыть скобку	<	Меньше
*	Умножение	<=	Меньше или равно
/	Деление	>	Больше
+	Сложение	>=	Больше или равно
-	Вычитание	<>	Не равно
^	Возведение в степень	%	Определение процента

ПРИМЕЧАНИЕ

Символ процента — оператор, который в формулах MS Excel делит предшествующее ему число на 100. Например, формула `=5%` дает результат `0,05`, а формула `=12781193%%%` — результат `12,781193`.

MS Excel может обрабатывать не только арифметические формулы, но и производить операции с текстом, сравнивать и соотносить различные диапазоны и ячейки в рабочей книге.

Операции с текстом и датами

Конкатенация — соединение текста, числа и даты внутри одной ячейки. Оператором конкатенации служит знак &, который соединяет текст, числа и даты в одну длинную текстовую строку.

Пример. Объединить в ячейку данные, находящиеся в различных ячейках рабочего листа MS Excel.

Решение приведено на рис. 3.18. В ячейку **A3** введена следующая формула:

```
=A1&ТЕКСТ(B1;" Д ММММ ГГГГ ")&C1&ТЕКСТ(D1;" # ##0р.")
```

Здесь функция `ТЕКСТ()` применяет новый формат даты и денежный формат к содержимому ячеек **B1** и **D1** и преобразует их в текст.

Текст, *даты* и *время* вводятся в формулы с помощью кавычек. Например, в результате действия формулы:

```
= "Итого" & ИТОГИ
```

появится текст:

```
Итого 1 500 000 р.
```

если в ячейке с именем **ИТОГИ** находится число `1500000 р.`

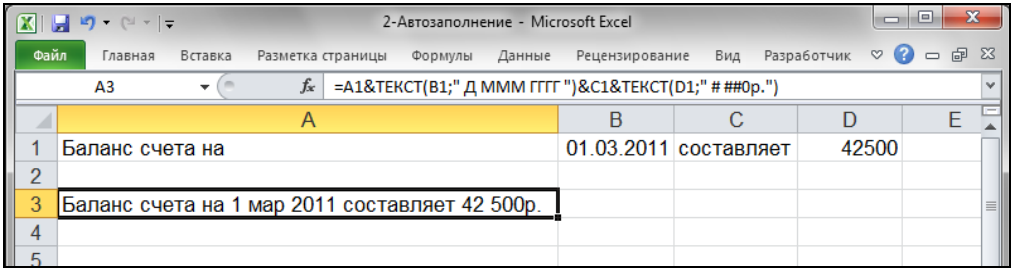


Рис. 3.18. Использование конкатенации

Для выполнения действий с явными датами, т. е. с такими, которые явно указываются в формулах, используются формулы вида:

```
= "15/02/11" - "11/02/11"
```

или

```
= "24 февраля 2011" - "26 мая 2010"
```

Эти формулы возвращают количество дней между двумя датами.

Операции сравнения и адресные операции

Примеры операций сравнения в формулах:

- `=A1<10` — ИСТИНА, если содержимое ячейки **A1** меньше 10; ЛОЖЬ, если больше или равно 10.
- `=B7>=15` — ИСТИНА, если содержимое ячейки **B7** больше или равно 15; ЛОЖЬ, если меньше 15.

В табл. 3.14 приведены знаки адресных операций, а в табл. 3.15 — приоритеты операций MS Excel.

Таблица 3.14. *Знаки адресных операций в MS Excel*

Знак операции	Пример	Операция	Результат
: (двоеточие)	`СУММ(A1:A7)`	Диапазон	Ссылка на все ячейки, в прямоугольном диапазоне, заключенном между двумя углами
, (запятая)	`СУММ(A1:A7,B8)`	Объединение	Объединение двух диапазонов: все ячейки из того и другого диапазона
Пробел	`СУММ(A1:A7 A16:B300)`	Пересечение	Пересечение двух диапазонов: все ячейки, общие для обоих диапазонов (если нет общих, возвращает `#ПУСТО` (`#NULL`))
Пробел	`=Y78 Кредит`	Пересечение	Содержимое ячейки на пересечении столбца с именем **Y78** и строки с именем **Кредит**

Таблица 3.15. *Таблица приоритетов операций (по убыванию) в MS Excel*

Знак операции	Операция	Знак операции	Операция
Пробел	Пересечение	* и /	Умножение и деление
/	Объединение	+ и –	Сложение и вычитание
–	Отрицание	&	Конкатенация текста
%	Процент	=<, <= и т. д.	Сравнения
^	Возведение в степень		

Автоматическое вычисление

В MS Excel имеется возможность автоматически проводить наиболее часто встречающиеся расчеты для выделенного диапазона данных (среднее значение, количество значений, количество чисел, максимум, минимум, сумму). Для этого в строке состояния в области автовычислений необходимо выбрать из контекстного меню (при щелчке правой кнопкой мыши) необходимую функцию (рис. 3.19).

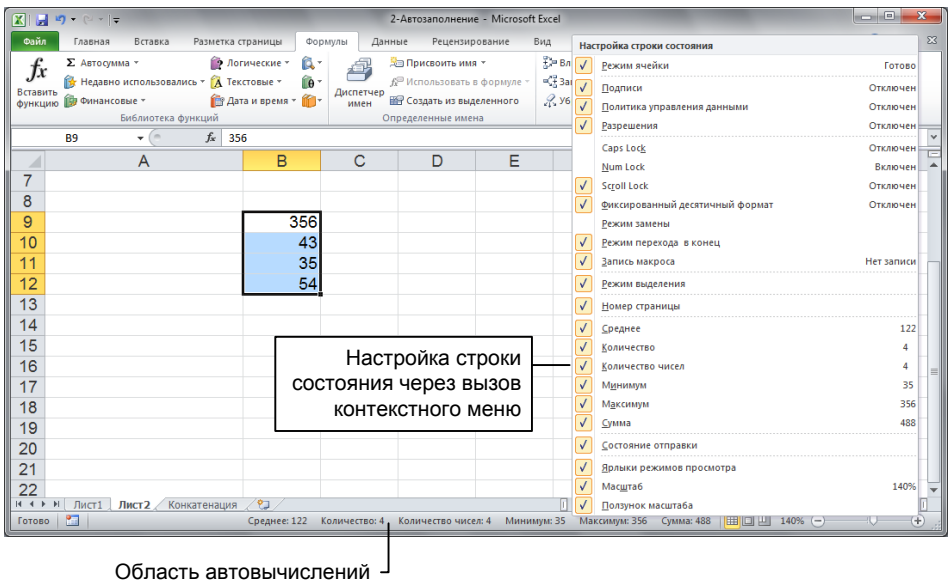


Рис. 3.19. Добавление в строку состояний функции для автовычислений

Используем функции

В процессе вычислений в MS Excel используются различные формулы, причем в качестве аргумента могут выступать константа, ссылка на ячейку или имя диапазона ячеек. В MS Excel существует множество *специальных функций*, в которые эти

формулы уже встроены. Значения, к которым должна применяться функция, задаются в качестве аргументов функций:

=ИМЯ_ФУНКЦИИ(Аргументы)

На формулы, содержащие функцию, не накладывается никаких ограничений по сравнению с другими формулами, в том числе их допускается копировать, учитывая тип ссылки (относительная или абсолютная).

Список всех функций MS Excel можно найти на вкладке **Формулы** ленты в группе команд **Библиотека функций**. С другой стороны, нажав в данной группе команд кнопку **Вставить функцию**, вы переходите в окно **Мастер функций**, в котором также можно выбрать необходимую функцию и получить по ней соответствующую справку.

В общем случае формулы могут включать различные ссылки, операторы и функции. Допускается задание в качестве аргументов ссылок на диапазоны ячеек из других листов и книг:

```
=СУММ(C7:C9;Лист3!D8:D15;[Книга1]Лист5!$E$8:$E$23)
```

При задании в качестве аргумента диапазона ячеек можно передвинуть окно мастера функций (если оно мешает выделению) и выделить мышью нужный диапазон.

При указании адреса диапазона ячеек в качестве аргумента речь может идти как о смежных, так и о несмежных диапазонах. Адрес смежного диапазона ячеек задается посредством указания адресов первой и последней ячеек, разделенных двоеточием. Три и более несмежных диапазонов отделяются точкой с запятой.

Иногда сама функция служит аргументом другой функции. Такие функции называются *вложенными*. Например:

```
=СУММ(A1, СУММ(A5,A6))
```

MS Excel допускает до 64 уровней вложения функций в формулах листа.

Логические функции

Создание сложных формул связано, как правило, с использованием встроенных логических функций MS Excel (табл. 3.16).

Таблица 3.16. *Логические функции MS Excel*

Функция	Описание
ЕСЛИ(*логич_выражение*; *значение_если_истина*; *значение_если_ложь*) IF( )	Логическое ветвление (допускает до 64 вложений): • *логич_выражение* — любое значение или выражение, принимающее значение ИСТИНА или ЛОЖЬ; • *значение_если_истина* — значение, которое возвращается, если *логич_выражение* равно ИСТИНА; • *значение_если_ложь* — значение, которое возвращается, если *логич_выражение* равно ЛОЖЬ
И(*логич_значение 1*; *логич_значение 2*;...) AND( )	Логическое умножение: возвращает значение ИСТИНА, если все аргументы имеют значение ИСТИНА; возвращает значение ЛОЖЬ, если хотя бы один аргумент имеет значение ЛОЖЬ

Таблица 3.16 *(окончание)*

Функция	Описание
ИЛИ(*логич_значение 1; логич_значение 2;...*) OR( )	Логическое сложение: возвращает значение ИСТИНА, если хотя бы один из аргументов имеет значение ИСТИНА; возвращает значение ЛОЖЬ, если все аргументы имеют значение ЛОЖЬ
НЕ(*логич_значение*) NOT( )	Логическое отрицание: изменяет на противоположное значение своего аргумента

Рассмотрим подробнее логическую функцию ЕСЛИ():

ЕСЛИ(*проверяемое логическое условие; значение если истина; значение если ложь*)

Данное выражение можно расширить за счет вложенной функции ЕСЛИ() в последней:

ЕСЛИ(*проверяемое логическое условие; значение если истина;* ЕСЛИ(*проверяемое логическое условие; значение если истина;* ЕСЛИ(*проверяемое логическое условие; значение если истина; значение если ложь*))

Примечание

Как отмечалось ранее, формулы можно копировать и перемещать. Однако нужно учитывать, что относительные ссылки, которые содержатся в формулах, будут изменяться. Перемещение формул — довольно опасная операция, поэтому следует быть внимательными. Во избежание проблем рекомендуется присвоить ячейкам имена, т. к. имена всегда ссылаются на одни и те же значения, независимо от того, куда и как их переместили.

Встроенные функции VBA

В VBA также имеется большой набор встроенных функций и процедур, использование которых существенно упрощает программирование. Как и функции рабочего листа, функции, имеющиеся в VBA, можно разделить на несколько основных категорий. Отметим, что всю необходимую информацию, связанную с использованием имеющихся функций, можно найти в справочной системе по VBA.

- Перейдите к окну редактора **Visual Basic for Applications** и выберите команду **Help | Visual Basic for Applications Help** (или нажмите клавишу <F1>).
- В окне **Справка Excel** выберите в оглавлении слева раздел **Visual Basic for Applications Language Reference | Visual Basic Language Reference | Functions**. Далее, можно увидеть вес список функций, который имеется в VBA, и примеры их использования.

Ошибки в формулах и отслеживание зависимостей

Если при задании формулы были допущены ошибки, результатом ее вычисления в ячейке будет значение ошибки (рис. 3.20). Первый символ ошибки в MS Excel представляет собой символ #, за которым следует текст. Текст значения ошибки может завершаться восклицательным знаком или знаком вопроса. Однако так распознать можно не все ошибки.

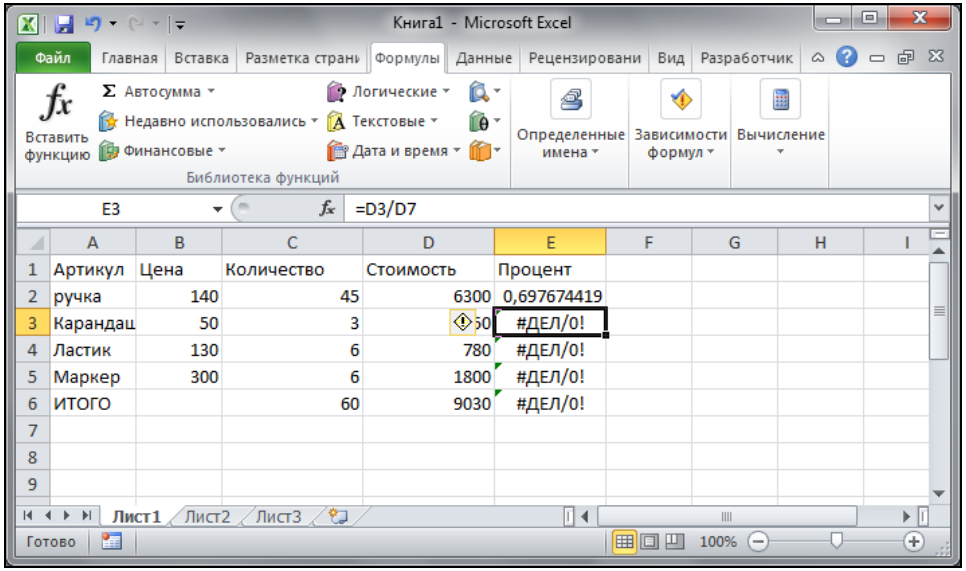


Рис. 3.20. Пример ошибки при задании формулы — деление на ноль

Для облегчения поиска можно включить режим отображения в ячейках формул вместо результата. Для этого следует перейти на вкладку **Файл** и выбрать команду **Параметры**. Далее в открывшемся окне **Параметры Excel** необходимо слева указать категорию **Дополнительно**, а справа в группе **Показать параметры для следующего листа** установить флажок возле опции **Показывать формулы, а не их значения**.

Можно также воспользоваться командой **Показать формулы**, расположенной в группе команд **Зависимости формул** на вкладке **Формулы** ленты.

Для поиска ошибок в MS Excel существует вспомогательная функция — *отслеживание зависимостей*, с помощью которой можно графически представить на экране связи между влияющими и зависимыми ячейками. Ячейка является *зависимой*, если она содержит формулу со ссылкой на активную ячейку. *Влияющей* называется та ячейка, на которую ссылается формула в активной ячейке.

Для отслеживания зависимостей, т. е. для графического представления влияющих и зависимых ячеек, следует использовать команды из группы **Зависимости формул** на вкладке **Формулы** ленты (рис. 3.21).

В случае если в ячейке появилось значение ошибки, можно попробовать установить вероятную причину с помощью команды **Источник ошибки**, которую следует выбрать из списка команд **Проверка наличия ошибок**, расположенного в группе **Зависимости формул** на вкладке **Формулы** ленты: стрелки укажут ячейки с ошибкой.

Поиск ошибок может занять много времени. При этом существенную помощь оказывает команда **Выделение группы ячеек**, которую следует выбрать из списка команд **Найти и выделить**, расположенного в группе команд **Редактирование** на вкладке **Главная** ленты. В открывшемся окне **Выделение группы ячеек** можно выбирать отдельные части рабочего листа, которые удовлетворяют требуемым критериям.

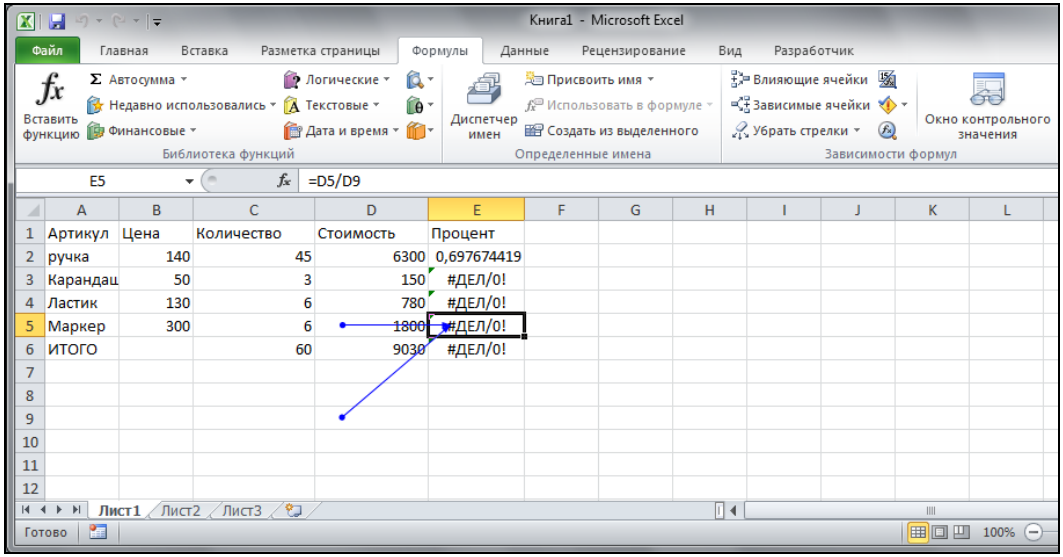


Рис. 3.21. Использование инструментов группы команд **Зависимости формул** на вкладке **Формулы** ленты

Для перемещения активной ячейки среди ранее выделенных с сохранением выделения необходимо пользоваться клавишами <Tab> (движение вперед) или <Shift>+<Tab> (движение назад).

Совет

Для поиска ошибок в формулах:

1. Выделите ячейку, дающую неверный результат или значение ошибки.
2. В строке формул выделите вызывающий сомнения элемент формулы.
3. Нажмите клавишу <F9> — для вычисления выделенной части: если появляется ЛОЖЬ, имеем ошибку.
4. Выделяйте таким же образом и вычисляйте другие части формулы до тех пор, пока не найдется ошибка.
5. Чтобы вернуть формулу к первоначальному виду (т. е. без сообщения ЛОЖЬ или без вычисленной части), нажмите клавишу <Esc> либо кнопку **Отмена** в строке формул.
6. Исправьте ошибочную часть формулы.

Примеры использования различных функций в Microsoft Office Excel

А сейчас мы рассмотрим некоторые примеры использования формул и встроенных функций Excel. В примерах, которые приводятся далее, даются пошаговые инструкции для выполнения, а также иллюстрируются различные варианты использования формул и функций — либо их добавление осуществляется через мастер функций, либо демонстрируется использование в программах на языке VBA.

Подготовка различных ведомостей

Ведомость о продаже квартир

Итак, пусть для начала требуется подготовить ведомость о продаже квартир согласно образцу (рис. 3.22, см. лист **Сцепление** в файле *11-Объявления.xlsx* на компакт-диске), для чего необходимо выполнить последовательно следующие действия.

1. Подготовьте необходимые данные о квартирах в виде списка (рис. 3.23).

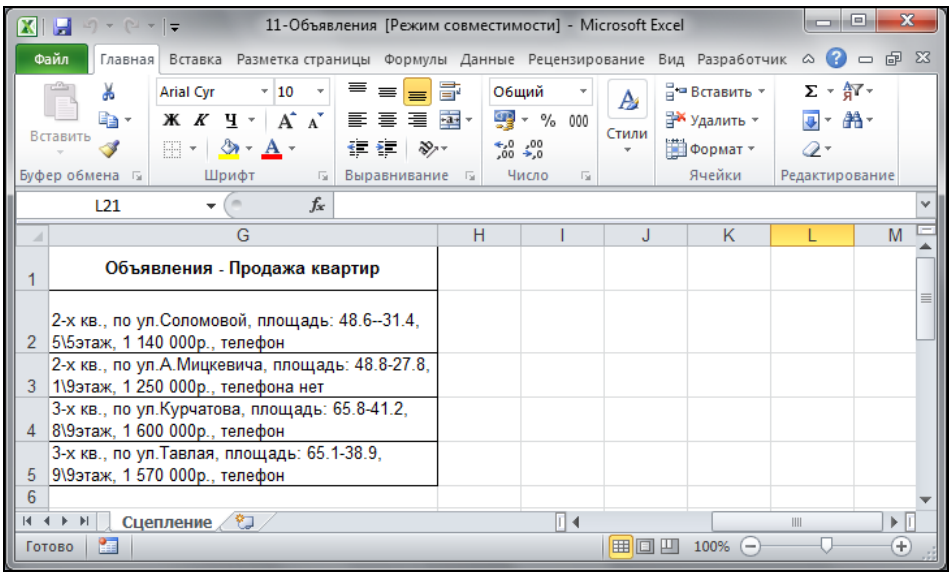


Рис. 3.22. Объявления по продаже квартир

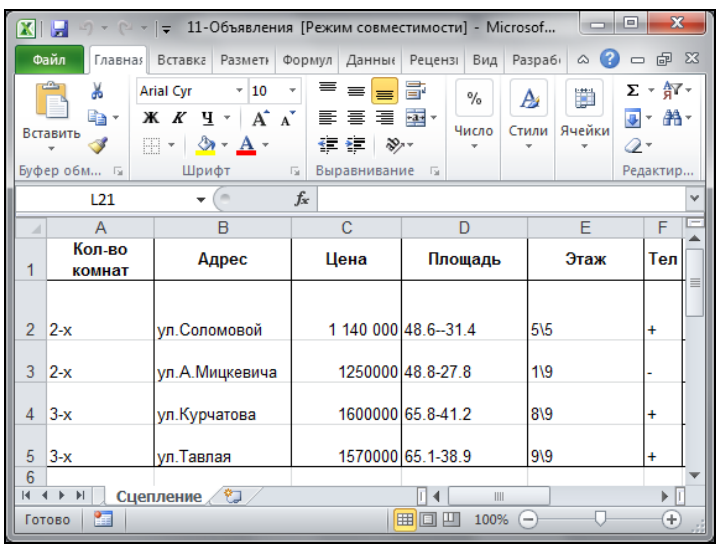
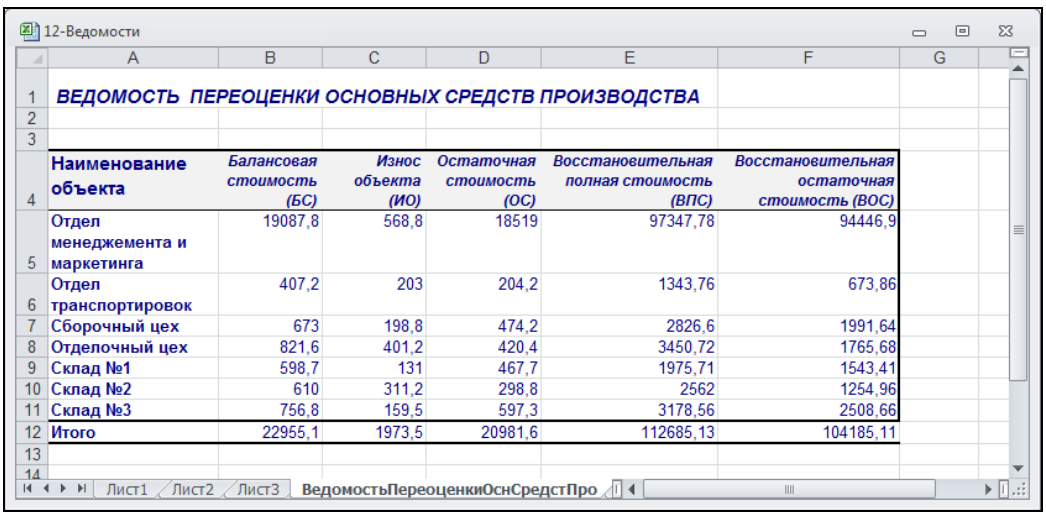


Рис. 3.23. Данные о квартирах, выставленных на продажу

2. В ячейку **G2** введите формулу:
```
=A2&" кв., по "&B2&", площадь: "&D2&", "&E2&"этаж, "&ТЕКСТ(C2;"# ##0р.")&",
"&ЕСЛИ(F2="+";"телефон"; "телефона нет")
```
3. Для диапазона **G3:G5** либо воспользуйтесь маркером автозаполнения, либо скопируйте данную формулу.
4. При необходимости отформатируйте полученный список объявления, используя возможности из коллекции команды **Форматировать как таблицу**, которая расположена в группе **Стили** на вкладке **Главная** ленты.

Ведомость, связанная с переоценкой основных средств производства

Теперь рассмотрим пример подготовки ведомости, связанной с переоценкой основных средств производства по форме, приведенной на рис. 3.24 (см. также файл *12-Ведомости.xlsx* на компакт-диске).



Наименование объекта	Балансовая стоимость (БС)	Износ объекта (ИО)	Остаточная стоимость (ОС)	Восстановительная полная стоимость (ВПС)	Восстановительная остаточная стоимость (ВОС)
Отдел менеджемента и маркетинга	19087,8	568,8	18519	97347,78	94446,9
Отдел транспортировок	407,2	203	204,2	1343,76	673,86
Сборочный цех	673	198,8	474,2	2826,6	1991,64
Отделочный цех	821,6	401,2	420,4	3450,72	1765,68
Склад №1	598,7	131	467,7	1975,71	1543,41
Склад №2	610	311,2	298,8	2562	1254,96
Склад №3	756,8	159,5	597,3	3178,56	2508,66
Итого	22955,1	1973,5	20981,6	112685,13	104185,11

Рис. 3.24. Ведомость переоценки основных средств производства

1. В ячейку **A1** введите название ведомости: `Ведомость переоценки основных средств производства`.
2. В ячейки **A4:F4** введите названия полей ведомости: `Наименование объекта`, `Балансовая стоимость (БС)`, `Износ объекта (ИО)`, `Остаточная стоимость (ОС)`, `Восстановительная полная стоимость (ВП)`, `Восстановительная остаточная стоимость (ВУО)`. Поле `Наименование объекта` включает следующие строки: `Отдел менеджмента и маркетинга`, `Отдел транспортировок`, `Сборочный цех`, `Отделочный цех`, `Склад №1`, `Склад №2`, `Склад №3`, `Итого`.
3. Формулы для расчетов:
```
ОС = БС - ИО
ВПС = БС * К
ВОС = ОС * К
```

где К — коэффициент, равный:

- 3,3 — если ВС меньше либо равен 650 млн руб.;
- 4,2 — если ВС больше 650 млн руб., но меньше 1000 млн руб.;
- 5,1 — если ВС равен 1000 млн руб. или более.

4. Для формирования автоматических расчетов используйте следующие формулы:
   - для ячейки **D5**:
     ```
     =B5-C5
     ```
   - для ячейки **E5**:
     ```
     =B5*ЕСЛИ(B5<=600;3,3;ЕСЛИ(И(B5>600;B5<1000);4,2;5,1))
     ```
   - для ячейки **F5**:
     ```
     =D5*ЕСЛИ(B5<=600;3,3;ЕСЛИ(И(B5>600;B5<1000);4,2;5,1))
     ```
5. Результирующую строку Итого, например, для ячейки **B12** получите с помощью формулы:
   ```
   =СУММ(B5:B11)
   ```
   либо выделите диапазон ячеек **B12:F12** и воспользуйтесь возможностью автосуммирования (нажмите кнопку **Автосумма** Σ Автосумма, которая расположена в группе команд **Библиотека функций** на вкладке **Формулы** ленты).

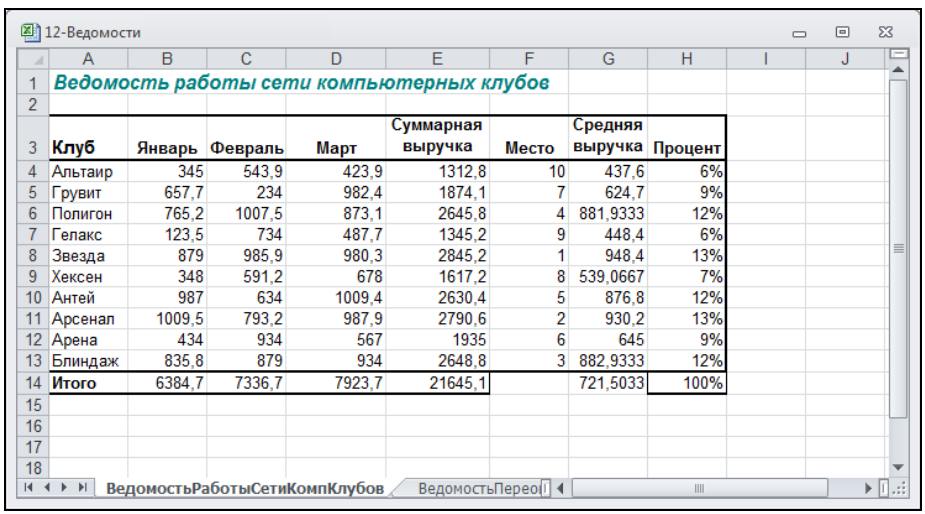
   Примечание

   Стрелка возле кнопки **Автосумма** позволяет производить автоматические вычисления с использованием других функций (например, среднее, максимум, минимум и т. д.).

6. Отформатируйте полученные в таблице результаты, а также название ведомости.

Отчетная ведомость по работе сети компьютерных клубов

В следующем примере демонстрируется подготовка отчетной ведомости по работе сети компьютерных клубов (рис. 3.25, см. также файл *12-Ведомости.xlsx* на компакт-диске).



Клуб	Январь	Февраль	Март	Суммарная выручка	Место	Средняя выручка	Процент
Альтаир	345	543,9	423,9	1312,8	10	437,6	6%
Грувит	657,7	234	982,4	1874,1	7	624,7	9%
Полигон	765,2	1007,5	873,1	2645,8	4	881,9333	12%
Гелакс	123,5	734	487,7	1345,2	9	448,4	6%
Звезда	879	985,9	980,3	2845,2	1	948,4	13%
Хексен	348	591,2	678	1617,2	8	539,0667	7%
Антей	987	634	1009,4	2630,4	5	876,8	12%
Арсенал	1009,5	793,2	987,9	2790,6	2	930,2	13%
Арена	434	934	567	1935	6	645	9%
Блиндаж	835,8	879	934	2648,8	3	882,9333	12%
Итого	6384,7	7336,7	7923,7	21645,1		721,5033	100%

Рис. 3.25. Ведомость работы сети компьютерных клубов

1. В ячейку **A1** введите название ведомости: `Ведомость работы сети компьютерных клубов`.
2. В ячейки **A3:H3** введите названия полей ведомости: `Клуб`, `Январь`, `Февраль`, `Март`, `Суммарная выручка`, `Место`, `Средняя выручка`, `Процент`. Поле `Клуб` включает следующие строки: `Альтаир`, `Грувит`, `Полигон`, `Гелакс`, `Звезда`, `Хексен`, `Антей`, `Арсенал`, `Арена`, `Блиндаж`, `Итого`.
3. Основные формулы для вычислений, которые копируются для аналогичных вычислений по строкам, представлены в табл. 3.17.
4. Отформатируйте полученную ведомость.

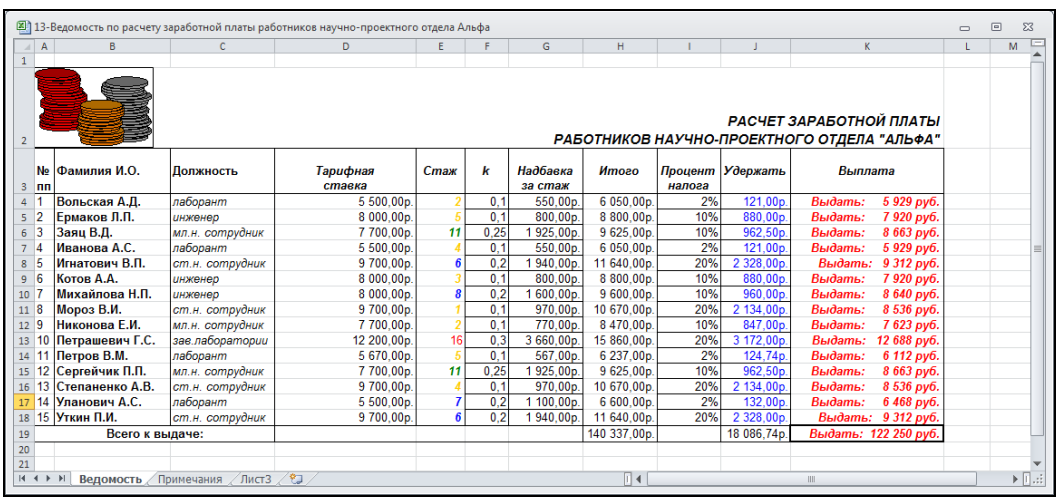
***Таблица 3.17.** Формулы для расчета*

Ячейка	Формула	Ячейка	Формула
E4	`=СУММ(B4:D4)`	**G4**	`=СРЗНАЧ(B4:D4)`
B14	`=СУММ(B4:B13)`	**G14**	`=СРЗНАЧ(G4:G13)`
F4	`=РАНГ(E4;$E$4:$E$13)`	**H4**	`=E4/$E$14`

Ведомость по расчету заработной платы

А сейчас мы рассмотрим подготовку ведомости "Расчет заработной платы работников научно-проектного отдела "Альфа"" (рис. 3.26, см. также файл *13-Ведомость по расчету заработной платы работников научно-проектного отдела Альфа.xlsx* на компакт-диске).

1. В ячейку **A2** поместите название ведомости — `Расчет заработной платы работников научно-проектного отдела "Альфа"`, отцентрируйте по правому краю (например, кнопкой **Выровнять текст по правому краю**  , расположенной в группе команд **Выравнивание** на вкладке **Главная** ленты).



13-Ведомость по расчету заработной платы работников научно-проектного отдела Альфа

РАСЧЕТ ЗАРАБОТНОЙ ПЛАТЫ
РАБОТНИКОВ НАУЧНО-ПРОЕКТНОГО ОТДЕЛА "АЛЬФА"

№ пп	Фамилия И.О.	Должность	Тарифная ставка	Стаж	k	Надбавка за стаж	Итого	Процент налога	Удержать	Выплата
1	Вольская А.Д.	лаборант	5 500,00р.	2	0,1	550,00р.	6 050,00р.	2%	121,00р.	Выдать: 5 929 руб.
2	Ермаков Л.П.	инженер	8 000,00р.	5	0,1	800,00р.	8 800,00р.	10%	880,00р.	Выдать: 7 920 руб.
3	Заяц В.Д.	мл.н. сотрудник	7 700,00р.	11	0,25	1 925,00р.	9 625,00р.	10%	962,50р.	Выдать: 8 663 руб.
4	Иванова А.С.	лаборант	5 500,00р.	4	0,1	550,00р.	6 050,00р.	2%	121,00р.	Выдать: 5 929 руб.
5	Игнатович В.П.	ст.н. сотрудник	9 700,00р.	6	0,2	1 940,00р.	11 640,00р.	20%	2 328,00р.	Выдать: 9 312 руб.
6	Котов А.А.	инженер	8 000,00р.	3	0,1	800,00р.	8 800,00р.	10%	880,00р.	Выдать: 7 920 руб.
7	Михайлова Н.П.	инженер	8 000,00р.	8	0,2	1 600,00р.	9 600,00р.	10%	960,00р.	Выдать: 8 640 руб.
8	Мороз В.И.	ст.н. сотрудник	9 700,00р.	1	0,1	970,00р.	10 670,00р.	20%	2 134,00р.	Выдать: 8 536 руб.
9	Никонова Е.И.	мл.н. сотрудник	7 700,00р.	2	0,1	770,00р.	8 470,00р.	10%	847,00р.	Выдать: 7 623 руб.
10	Петрашевич Г.С.	зав.лаборатории	12 200,00р.	16	0,3	3 660,00р.	15 860,00р.	20%	3 172,00р.	Выдать: 12 688 руб.
11	Петров В.М.	лаборант	5 670,00р.	5	0,1	567,00р.	6 237,00р.	2%	124,74р.	Выдать: 6 112 руб.
12	Сергейчик П.П.	мл.н. сотрудник	7 700,00р.	11	0,25	1 925,00р.	9 625,00р.	10%	962,50р.	Выдать: 8 663 руб.
13	Степаненко А.В.	ст.н. сотрудник	9 700,00р.	4	0,1	970,00р.	10 670,00р.	20%	2 134,00р.	Выдать: 8 536 руб.
14	Уланович А.С.	лаборант	5 500,00р.	7	0,2	1 100,00р.	6 600,00р.	2%	132,00р.	Выдать: 6 468 руб.
15	Уткин П.И.	ст.н. сотрудник	9 700,00р.	6	0,2	1 940,00р.	11 640,00р.	20%	2 328,00р.	Выдать: 9 312 руб.
	Всего к выдаче:						140 337,00р.		18 086,74р.	Выдать: 122 250 руб.

Ведомость | Примечания | Лист3

Рис. 3.26. Ведомость по расчету заработной платы работников научно-проектного отдела "Альфа"

2. В ячейки **A3:K3** введите названия полей ведомости: `№ пп`, `Фамилия И.О.`, `Должность`, `Тарифная ставка`, `Стаж`, `k`, `Надбавка за стаж`, `Итого`, `Процент налога`, `Удержать`, `Выдать`.
3. К шапке ведомости (к каждому столбцу) создайте скрытые примечания (рис. 3.27):
   - `№ пп` — номер работника отдела;
   - `Фамилия И.О.` — заносятся все фамилии работающих в научно-проектном отделе;
   - `Должность` — занимаемая должность на момент заполнения ведомости;

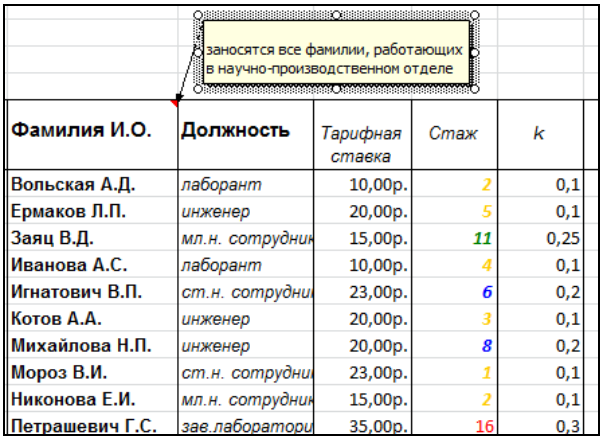


Рис. 3.27. Создание примечаний к ведомости

   - `Тарифная ставка` — денежный эквивалент занимаемой должности;
   - `Стаж` — вносится целое число отработанных лет на момент заполнения ведомости;
   - `k` — коэффициент за стаж работы;
   - `Надбавка за стаж` — денежный эквивалент за стаж работы;
   - `Итого` — начисление заработной платы с учетом тарифной ставки и стажа работы;
   - `Процент налога` — определяет процент отчислений в бюджет;
   - `Удержать` — денежный эквивалент отчислений в бюджет;
   - `Выдать` — сумма, предназначенная к выдаче.

   Примечания создаются так: перейдите на вкладку **Рецензирование** ленты и в группе команд **Примечания** нажмите кнопку **Создать примечание** (другие инструменты на данной вкладке **Примечания** позволяют организовать работу с примечаниями, находящимися или создаваемыми в рабочей книге).
4. При расчетах в ведомости учтите следующее.
   - `k`, `Надбавка за стаж`, `Итого`, `Процент налога`, `Удержать`, `Выдать` вычисляются с помощью соответствующих формул, с использованием автозаполнения или копирования формулы.
   - Коэффициент `k` присваивается из следующего расчета: 0,1 — отработано до 5 лет включительно, 0,2 — от 5 до 10 лет включительно, 0,25 — от 10 до 15 лет включительно, 0,3 — свыше 15 лет. Формула для ячейки **F4**:
     `=ЕСЛИ(E4<=5;0,1;ЕСЛИ(И(E4>5;E4<=10);0,2;ЕСЛИ(И(E4>10;E4<=15);0,25;0,3)))`
   - `Надбавка за стаж` — денежный эквивалент за стаж работы. Формула для ячейки **G4**:
     `=D4*F4`

     Пользовательский формат числа для ячейки **G4**:
     `# ##0,00р.;`

     (вводится в окне **Формат ячеек**: перейдите на вкладку **Главная** ленты и в группе **Ячейки**, щелкнув по кнопке со списком **Формат**, выберите команду

Формат ячеек; далее в открывшемся окне **Формат ячеек** перейдите на вкладку **Число**, выберите из списка **Числовые форматы** вариант **Все форматы** и в поле **Тип** добавьте указанный выше формат).

- `Итого` — тарифная ставка с учетом стажа. Формула для ячейки **H4**:
  `=D4+G4`

  Пользовательский формат числа для ячейки **H4**:
  `# ##0,00р.;`
- `Процент налога` — учитывает, что: 2% — начисление (по `Итого`) составляет до 7000 руб. включительно, 10% — более 7000 до 10 000 руб. включительно, 20% — более 10 000 до 25 000 руб. включительно, 35% — превышающие 25 000 руб. Формула для ячейки **I4**:
  `=ЕСЛИ(H4<=7000;0,02;ЕСЛИ(И(H4>7000;H4<=10000);0,1;ЕСЛИ(И(H4>10000;H4<=25000);0,2;0,35)))`

  Формат числа для ячейки **I4** — **Процентный**.
- `Удержать` — денежный эквивалент налогов. Формула для ячейки **J4**:
  `=H4*I4`

  Пользовательский формат числа для ячейки **J4**:
  `# ##0,00р.;`
- `Выдать` — сумма к выдаче: `Итого` без `Удержать`.

5. Требования к столбцу `Стаж`.
   - Создайте пользовательский формат данных, учитывающий стаж работы: до 5 лет — данные представлены желтым цветом, от 5 до 10 — синим, от 10 до 15 — зеленым, свыше 15 — красным.

     Для создания такого пользовательского формата можно, например, воспользоваться окном **Формат ячеек**: перейдите на вкладку **Главная** ленты и в группе **Ячейки**, щелкнув по кнопке со списком **Формат**, выберите команду **Формат ячеек**; далее в открывшемся окне **Формат ячеек** перейдите на вкладку **Число**, выберите из списка **Числовые форматы** вариант **Все форматы** и в поле **Тип** добавьте следующий формат для диапазона **E4:E18**:
     `[Красный]# ##0;`

     А также обязательно используйте возможности кнопки со списком **Условное форматирование**, которая расположена в группе команд **Стили** на вкладке **Главная** ленты (рис. 3.28).
   - В случае ввода отрицательного числа лет должно появляться соответствующее окно (рис. 3.29). Для проверки ввода чисел используйте правило, которое задается в окне **Проверка вводимых значений** на вкладке **Сообщение об ошибке**: перейдите на вкладку **Данные** ленты и в группе **Работа с данными** выберите из списка **Проверка данных** команду **Проверка данных**.
6. Для поля `Тарифная ставка` выведите постоянное сообщение: "Тарифная ставка. БУДЬТЕ ВНИМАТЕЛЬНЫ ПРИ ВВОДЕ ТАРИФНОЙ СТАВКИ" (рис. 3.30), для получения которого используйте окно **Проверка вводимых значений** (вкладка **Сообщение для ввода**): перейдите на вкладку **Данные** ленты и в группе **Работа с данными** выберите из списка **Проверка данных** команду **Проверка данных**.

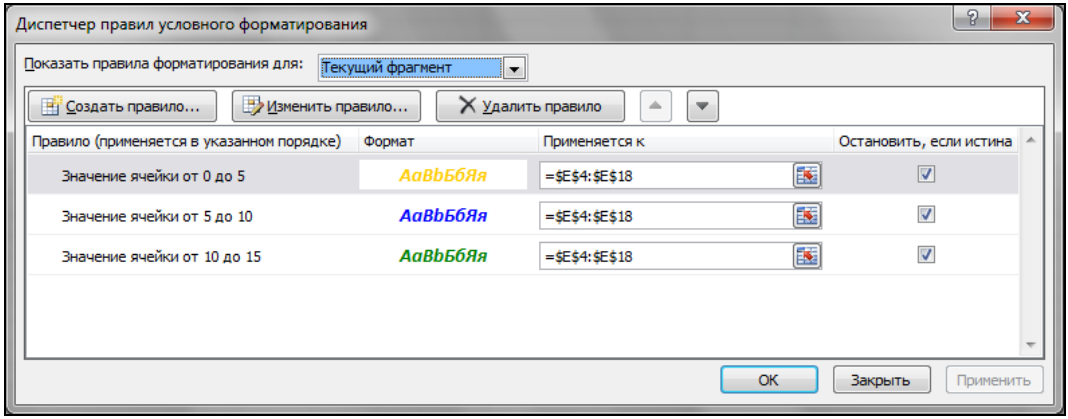


Рис. 3.28. Отображение критериев для условного форматирования данных столбца `Стаж` в окне **Диспетчер правил условного форматирования**

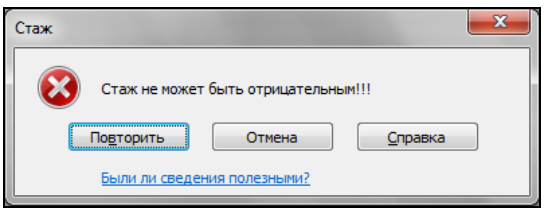


Рис. 3.29. Сообщение о неправильном вводе в поле `Стаж`

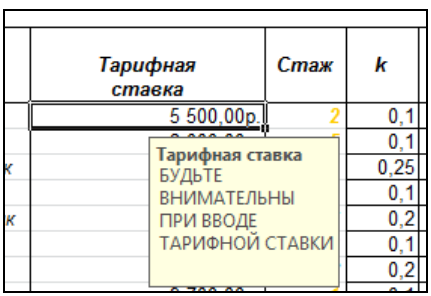


Рис. 3.30. Сообщение для поля `Тарифная ставка`

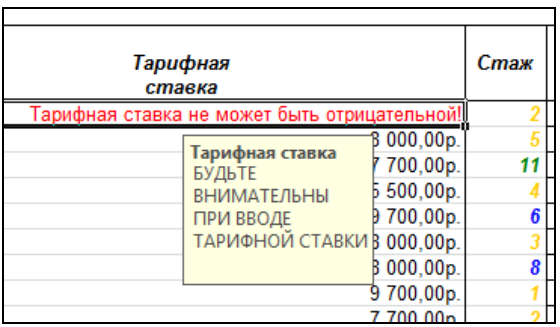


Рис. 3.31. Сообщение при вводе отрицательной тарифной ставки

В случае ввода отрицательных значений в столбце `Тарифная ставка` появляется соответствующее предупреждение "Тарифная ставка не может быть отрицательной" (рис. 3.31), которое формируется через пользовательский формат:

```
# ##0,00р.;[Красный]"Тарифная ставка не может быть отрицательной!"
```

Использование встроенных функций для решения различных задач

Как мы уже отмечали ранее, возможности MS Excel удобно использовать для решения различных математических, физических, экономических и других задач. Достаточно правильно расположить информацию на рабочем листе, т. е. подготовить начальные данные и определиться с местом расположения результата, а также ввести необходимые формулы для расчетов.

Имеющиеся функции Excel можно найти на вкладке **Формулы** ленты в группе **Библиотека функций**. Также доступ ко всем имеющимся функциям и работу с ними непосредственно можно организовать в окне **Мастер функций.**

В MS Excel имеется большой выбор встроенных функций для обработки как числовых значений, так и данных другого типа, содержащихся в ячейках. Просмотреть имеющиеся категории функций и конкретное назначение отдельной функции можно с помощью мастера функций: нажмите кнопку **Вставить функцию**, которая расположена в группе **Библиотека функций** на вкладке **Формулы** ленты, и вы перейдете к окну **Мастер функций.**

Рассмотрим несколько примеров использования функций и формул для решения конкретных задач.

Принадлежность точек плоскости

На плоскости заданы координаты точек. Определить, сколько заданных точек принадлежит области, определенной системой неравенств:

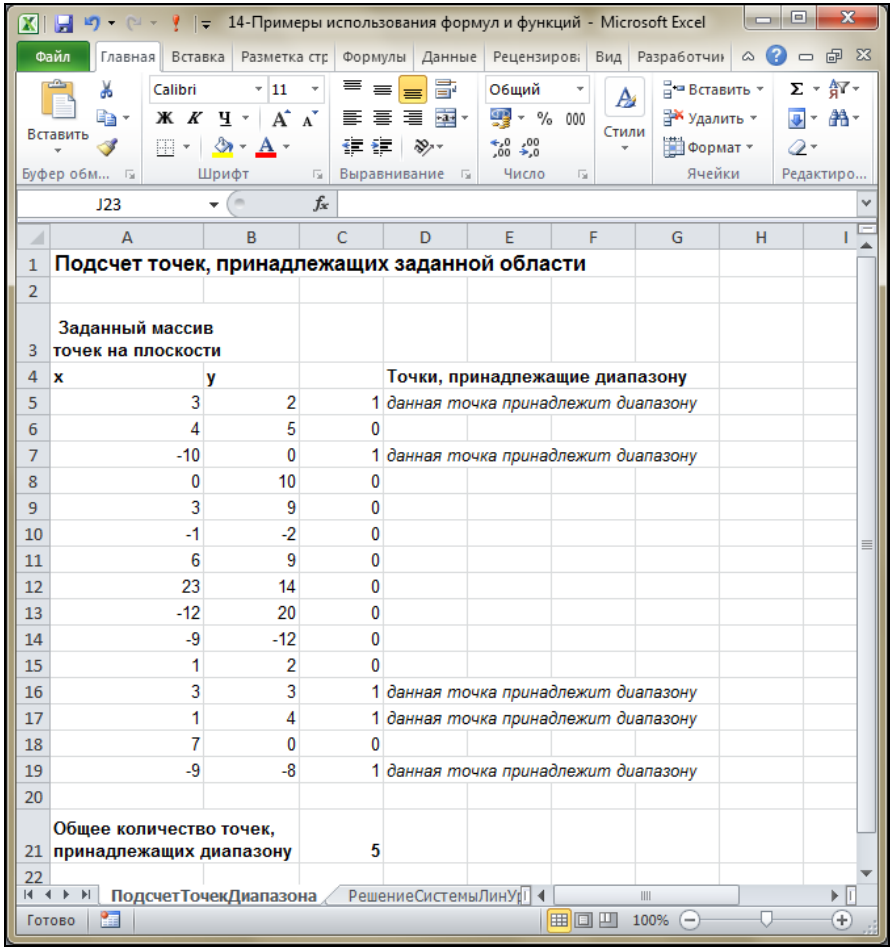
$$\left[\begin{cases} x^2 + y^2 <= 25, \\ x^2 + y^2 >= 9, \end{cases} \right. \left. \begin{cases} y <= -x - 10, \\ x >= -10, \\ y >= -10. \end{cases} \right.$$

Результат определения принадлежности точек представлен на рис. 3.32 (см. также файл *14-Примеры использования формул и функций.xlsx* на компакт-диске).

Для выполнения задания:

1. В ячейки **A1**, **A3**, **A4**, **B4**, **D4**, **A5:B19** (диапазон зависит от количества точек, координаты заданных точек), **A21** рабочего листа введите необходимые подписи для данных (см. рис. 3.32).

> ***Совет***
>
> Используйте комбинацию клавиш <Alt>+<Enter> для перехода к следующей строке в активной ячейке, затем выделите ячейку **A3** (с текстом) и **B3** (соседнюю) и выберите из списка команду **Объединить и поместить в центре**, нажав на кнопку  **Объединить и поместить в центре**, которая расположена в группе **Выравнивание** на вкладке **Главная** ленты.



	A	B	C	D
1	**Подсчет точек, принадлежащих заданной области**			
2				
3	**Заданный массив точек на плоскости**			
4	**x**	**y**		**Точки, принадлежащие диапазону**
5	3	2	1	*данная точка принадлежит диапазону*
6	4	5	0	
7	-10	0	1	*данная точка принадлежит диапазону*
8	0	10	0	
9	3	9	0	
10	-1	-2	0	
11	6	9	0	
12	23	14	0	
13	-12	20	0	
14	-9	-12	0	
15	1	2	0	
16	3	3	1	*данная точка принадлежит диапазону*
17	1	4	1	*данная точка принадлежит диапазону*
18	7	0	0	
19	-9	-8	1	*данная точка принадлежит диапазону*
20				
21	**Общее количество точек, принадлежащих диапазону**		**5**	
22				

Рис. 3.32. Принадлежность точек заданной области

2. В соответствии с заданными неравенствами формируется формула для определения принадлежности точки, удовлетворяющей хотя бы первой или второй группе неравенств, причем в случае положительного результата значению ячейки присвоится 1.

   Итак, в ячейку **C5** добавьте следующую формулу (используя встроенные функции MS Excel):

   ```
   =ЕСЛИ(ИЛИ(И(A5^2+B5^2<=25;A5^2+B5^2>=9);И(B5<=(-A5)-10;A5>=-10;B5>=-10));1;0)
   ```

   которая далее копируется на диапазон **C6:C19**.
3. Для организации текстовой подписи принадлежности добавьте в ячейку **D5** (и, соответственно, в диапазон **D5:D19**) формулу:

   ```
   =ЕСЛИ(C5=1;"данная точка принадлежит диапазону";)
   ```

   причем пользовательский формат для отображения чисел в диапазоне ячеек **D5:D19** следующий:

   ```
   ;;[Белый]
   ```

4. В ячейку **C21** введите формулу:
   `=СУММ(C5:C19)`
5. При необходимости можно отформатировать ячейки с данными и текстом, используя возможности группы команд **Стили**, расположенной на вкладке **Главная** ленты.

Пример решения системы линейных уравнений

В общем случае решение линейной системы $\mathbf{AX}=\mathbf{B}$, где **A** — матрица коэффициентов, **B** — вектор-столбец свободных членов, **X** — вектор-столбец неизвестных, имеет вид $\mathbf{X}=\mathbf{A}^{-1}\mathbf{B}$, где $\mathbf{A}^{-1}$ — матрица, обратная к матрице **A**. Это вытекает из того, что при решении матричных уравнений при **X** должна остаться единичная матрица **E**. Умножая слева обе части уравнения $\mathbf{AX}=\mathbf{B}$ на $\mathbf{A}^{-1}$, получаем решение линейной системы уравнений.

Рассмотрим решение системы линейных уравнений $\mathbf{AX}=\mathbf{B}$, где значения соответствующих матрицы и вектора-столбца имеют вид:

$$\mathbf{A}=\begin{bmatrix}23 & 7\\ 11 & 4\end{bmatrix},\ \mathbf{B}=\begin{bmatrix}3\\ 2\end{bmatrix}.$$

Результат для этого примера представлен на рис. 3.33 (см. также файл *14-Примеры использования формул и функций.xlsx* на компакт-диске).

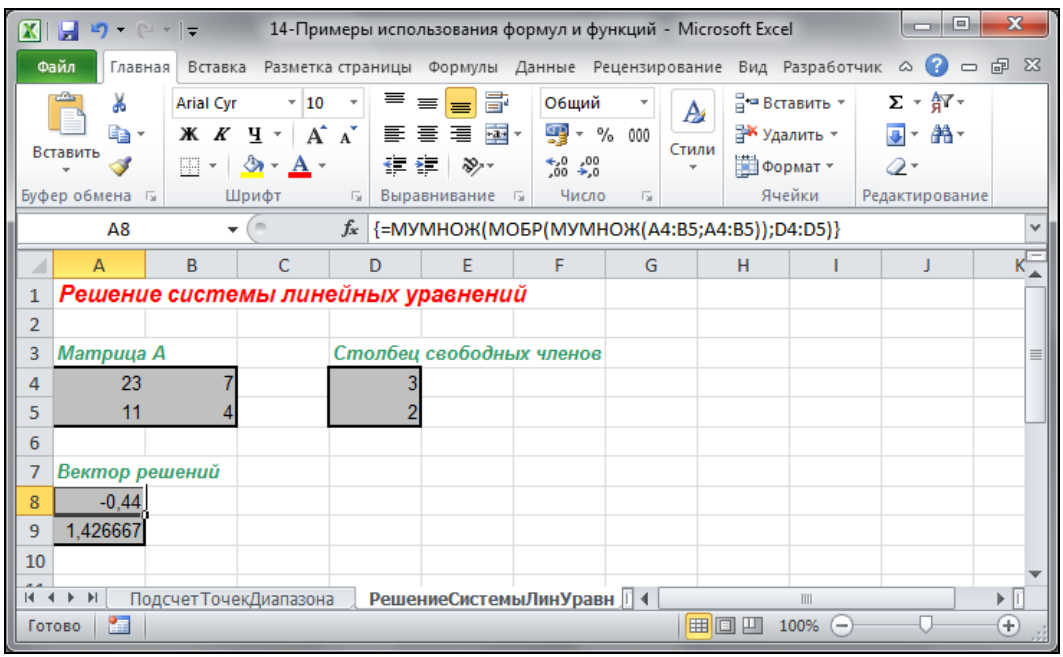


Рис. 3.33. Решение системы линейных уравнений

Для решения системы линейных уравнений:

1. Значения матрицы **A** поместите в ячейки **A4:B5**.
2. Значения столбца свободных членов поместите в ячейки **D4:D5**.
3. В ячейку **A8** введите формулу:

```
=МУМНОЖ(МОБР(МУМНОЖ(A4:B5;A4:B5));D4:D5)
```

При вводе формулы рекомендуется использовать мастер функций.

4. Для получения численных результатов решения системы линейных уравнений выделите диапазон **A8:A9**, затем установите указатель мыши в строку формул и нажмите одновременно клавиши <Ctrl>+<Shift>+<Enter>.
5. При необходимости отформатируйте данные задачи.

Пример создания итоговой конструкции по заданному образцу

Пусть требуется по имеющейся информации:

Ф.И.О.	Телефон	Улица, дом

создать конструкцию:

```
Ф.И.О., Телефон {формат: #00-00-00}, ул. Улица, Дом
```

Результат для этого примера представлен на рис. 3.34 (см. также файл *14-Примеры использования формул и функций.xlsx* на компакт-диске).

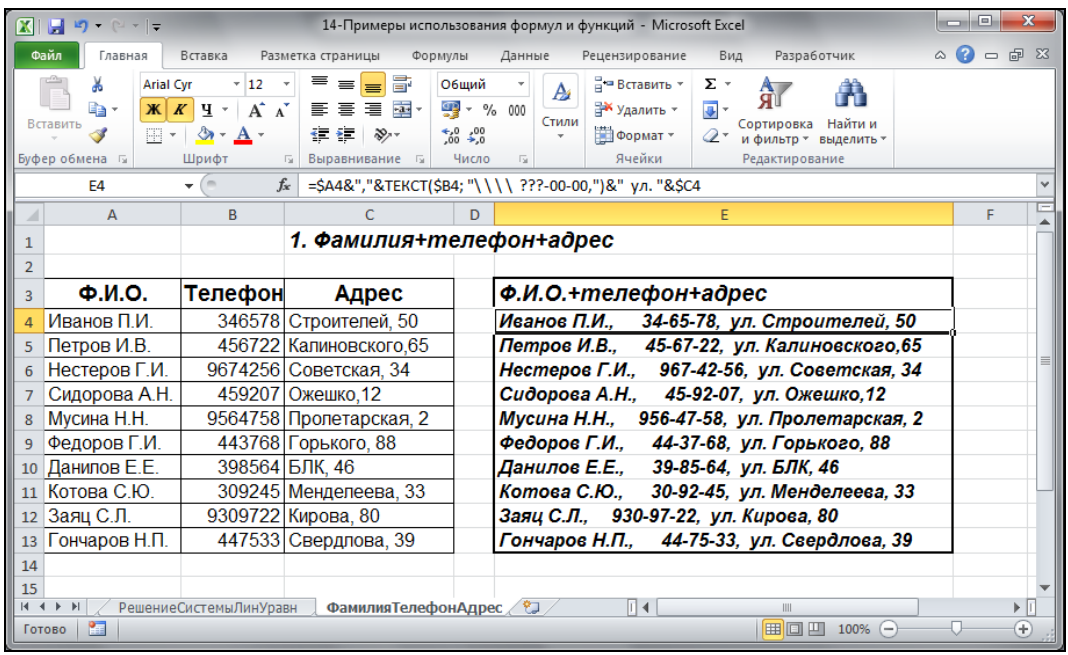


Рис. 3.34. Конструкции **Фамилия+телефон+адрес**

Для выполнения задания:

1. Введите в ячейки **A3:C13** данные в соответствии с заданной конструкцией (см. рис. 3.34).
2. В ячейку **E3** поместите текст заголовка конструкции: `Ф.И.О.+телефон+адрес`.
3. В ячейку **E4** введите формулу:
   `=$A4&","&ТЕКСТ($B4; "\ \ \ \  ???-00-00,")&"  ул. "&$C4`
4. Скопируйте формулу конструкции на диапазон ячеек **E5:E13** (можно использовать маркер автозаполнения).
5. Отформатируйте данные.

Пример разделения информации, находящейся в одной ячейке

Разделить следующую информацию, находящуюся в одной ячейке:

Город!Учреждение!Руководитель!Число занятых

Результат получить в виде:

Город	Учреждение	Руководитель	Число занятых

Результат для этого примера представлен на рис. 3.35 (см. также файл *14-Примеры использования формул и функций.xlsx* на компакт-диске).

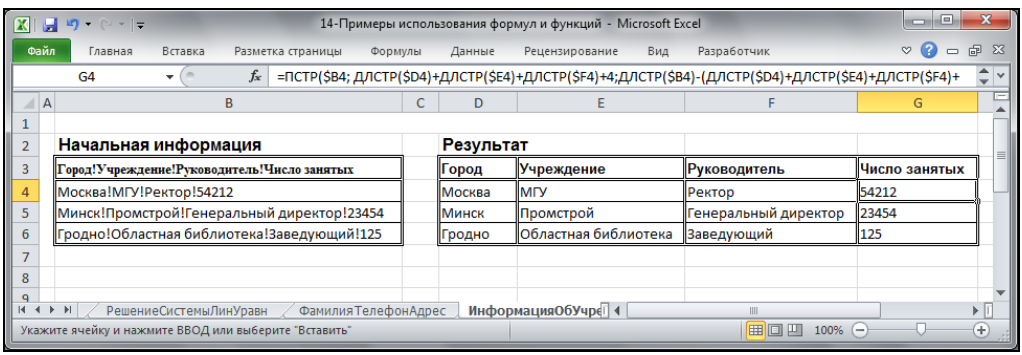


Рис. 3.35. Разделение информации об учреждениях

Для выполнения задания:

1. В ячейки **B2** и **D2** введите, соответственно, подписи: `Начальная информация`, `Результат`.
2. В ячейки **B3:B6** введите заголовок и необходимые данные начальной конструкции.
3. Формулы для расчетов возьмите из табл. 3.18.
4. Отформатируйте данные и результаты, представленные на рабочем листе.

Таблица 3.18. *Формулы для расчета задачи по определению информации об учреждениях*

Ячейка	Формула	Описание	Копируется на диапазон
D3	`=ПСТР(B3;1;НАЙТИ("!";B3)-1)`	Определение города	**D4:D6**
E3	`=ПСТР($B3;ДЛСТР($D3)+2;НАЙТИ("!";$B3; ДЛСТР($D3)+2) - ДЛСТР($D3)-2)`	Определение учреждения	**E4:E6**
F3	`=ПСТР($B3;ДЛСТР($D3)+ДЛСТР($E3)+3;НАЙТИ("!";$B3;ДЛСТР($D3)+ДЛСТР($E3)+3)- (ДЛСТР($D3)+ДЛСТР($E3)+3))`	Определение руководителя	**F4:F6**
G3	`=ПСТР($B3; ДЛСТР($D3)+ДЛСТР($E3)+ДЛСТР($F3)+4;ДЛСТР($B3)- (ДЛСТР($D3)+ДЛСТР($E3)+ДЛСТР($F3)+3))`	Определение числа занятых	**G4:G6**

Пример создания ведомости для учета проката фильмов

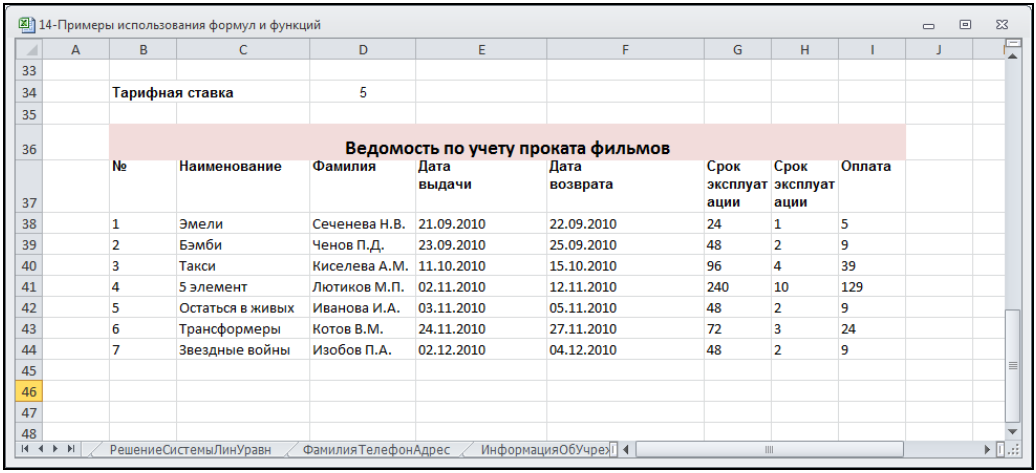
Сформировать ведомость для учета проката фильмов со следующими графами: **№ пп**, **Наименование фильма**, **Фамилия**, **Дата выдачи**, **Дата возврата**, **Срок эксплуатации (в часах)**, **Срок эксплуатации (в днях)**, **Оплата**. Произвести необходимые вычисления. Оплату начислять, исходя из следующих положений:

- если срок эксплуатации меньше либо равен 24 ч, то `ОПЛАТА = тарифная ставка` (определить произвольно);
- если срок эксплуатации больше 24 ч и меньше либо равен 48 ч, то `ОПЛАТА = тарифная ставка + 0,8*тарифная ставка`;
- если срок эксплуатации больше 48 ч, то за каждый просроченный день взимаются 3 тарифные ставки;
- если видеокассета или CD утеряны, то взимается штраф в размере 30 тарифных ставок.

Результат для этого примера представлен на рис. 3.36 (см. также файл *14-Примеры использования формул и функций.xlsx* на компакт-диске).

Для выполнения задания:

1. Сформируйте строку заголовка ведомости и внесите необходимые данные в столбцы: `№ пп`, `Наименование`, `Фамилия`, `Дата выдачи`, `Дата возврата`.
2. В ячейку **D34** введите величину тарифной ставки — `5`.
3. В ячейку **G38** введите формулу для учета времени проката (в часах):
   `=ГОД(F38-E38)-1900+МЕСЯЦ(F38-E38)+ДЕНЬ(F38-E38)*24-1`
4. В ячейку **H38** введите формулу для учета времени проката (в днях):
   `=ГОД(F38-E38)-1900+МЕСЯЦ(F38-E38)+ДЕНЬ(F38-E38)-1`
5. В ячейку **I38** введите формулу для расчета оплаты:
   `=(ЕСЛИ(G38<=24;$D$34;ЕСЛИ(И(G38>24;G38<=48);$D$34+$D$34*0,8;ЕСЛИ(G38>48;$D$34+$D$34*0,8+(G38-48)/24*3*$D$34))))`



	A	B	C	D	E	F	G	H	I	J
33										
34		Тарифная ставка		5						
35										
36		Ведомость по учету проката фильмов								
37		№	Наименование	Фамилия	Дата выдачи	Дата возврата	Срок эксплуатации	Срок эксплуатации	Оплата	
38		1	Эмели	Сеченева Н.В.	21.09.2010	22.09.2010	24	1	5	
39		2	Бэмби	Ченов П.Д.	23.09.2010	25.09.2010	48	2	9	
40		3	Такси	Киселева А.М.	11.10.2010	15.10.2010	96	4	39	
41		4	5 элемент	Лютиков М.П.	02.11.2010	12.11.2010	240	10	129	
42		5	Остаться в живых	Иванова И.А.	03.11.2010	05.11.2010	48	2	9	
43		6	Трансформеры	Котов В.М.	24.11.2010	27.11.2010	72	3	24	
44		7	Звездные войны	Изобов П.А.	02.12.2010	04.12.2010	48	2	9	
45										
46										
47										
48										

Рис. 3.36. Ведомость по учету проката фильмов

6. Скопируйте формулы в соответствующие однотипные диапазоны — **G39:G44**; **H39:H44** и **I39:I44**.
7. Отформатируйте все результаты, представленные на рабочем листе.

Использование функций в программах на языке VBA

Получение случайного числа из целочисленного интервала

Если требуется получить случайные числа не из интервала [0, 1], а из целочисленного интервала, например [1, 6], можно воспользоваться функцией `RndInt()`, определенной в следующем коде (листинг 3.20, см. также файл *15-Примеры решения задач на VBA с использованием функций.xlsm* на компакт-диске).

Листинг 3.20. Получение случайного числа из целочисленного интервала

```
Sub DemoIntegerRnd ()
   Dim i As Integer
   For i = 0 To 10
      Debug.Print RndInt(1, 6)
   Next
End Sub
Function RndInt(ByVal lowerbound As Integer, _
                ByVal upperbound As Integer) As Integer
   Randomize
   RndInt = (upperbound - lowerbound) * Rnd() + lowerbound
End Function
```

Вывод строки посимвольно в окно *Immediate*

Функция `Len()` определяет длину строки. Например, следующий код (листинг 3.21, см. также файл *15-Примеры решения задач на VBA с использованием функций.xlsm* на компакт-диске) выводит данную строку посимвольно в окно **Immediate**.

Листинг 3.21. Вывод строки посимвольно в окно *Immediate*

```
Sub Lengs()
   Dim i As Integer
   Dim s As String
   s = "Hello, World"
   For i = 1 To Len(s)
      Debug.Print Mid(s, i, 1)
   Next
End Sub
```

Строка, состоящая из указанного числа пробелов

Функция `Space()` возвращает строку, состоящую из указанного числа пробелов. Например, в следующем коде (листинг 3.22, см. также файл *15-Примеры решения задач на VBA с использованием функций.xlsm* на компакт-диске) за счет последовательного добавления перед данной строкой другой, состоящей каждый раз из меньшего числа пробелов, создается эффект бегущей строки. Для проверки "работоспособности" бегущей строки запускайте на выполнение процедуру `LetsGo()`.

Листинг 3.22. Бегущая строка

```
Private n As Integer
Sub LetsGo()
   n = 10
   RunString
End Sub
Sub RunString()
   If n >= 0 Then
      Range("Лист2!A1").Value = Space(n) & "Hello, World!"
      n = n - 1
      Application.OnTime Now + TimeValue("00:00:01"), "RunString"
   End If
End Sub
```

Определение числа секунд, прошедших с полуночи

Функция `Timer()` возвращает значение типа `Single`, представляющее число секунд, прошедших после полуночи. В Windows, в отличие от Mac OS X, функция `Timer()` возвращает даже не число секунд, а число долей секунд, прошедших после полуночи. Например, следующий код (листинг 3.23, см. также файл *15-Примеры решения задач на VBA с использованием функций.xlsm* на компакт-диске) создает

мигающую ячейку, у которой цвет фона в течение 20 секунд попеременно становится то красным, то зеленым. Для того чтобы во время выполнения цикла компьютер не зависал, и приложение могло реагировать на различные события, происходящие в системе, в циклы добавлены операторы `DoEvents`.

Листинг 3.23. Мигающая ячейка

```
Sub LetsGo2()
   Dim fl As Boolean
   Dim old As Long
   old = Timer
   Do
      fl = Not fl
      If fl Then
         Range("Лист3!A1").Interior.Color = RGB(255, 0, 0)
      Else
         Range("Лист3!A1").Interior.Color = RGB(0, 255, 0)
      End If
      Delay
      DoEvents
   Loop While Timer — old <= 20
   Range("Лист3!A1").Interior.ColorIndex = xlNone
End Sub
Sub Delay()
   Dim old As Long
   old = Timer
   Do
     DoEvents
   Loop While Timer — old <= 0.5
End Sub
```

Наши итоги

Создание необходимых формул, включающих имеющиеся в Microsoft Office Excel 2010 функции, несомненно, ускорит вашу работу по подготовке различных ведомостей, проведению расчетов и обработке данных различного плана. Материал изученной вами главы и разнообразные примеры продемонстрировали возможности по использованию:

- адресации ячеек и работе с диапазонами рабочего листа Excel;
- автозаполнения и табуляции;
- автозамены и поиска значений;
- примечаний и проверки данных;
- различных видов форматирования;
- создаваемых формул и встроенных функций Excel как на рабочем листе, так и в программах на VBA.

Глава 4

Как создаются пользовательские формы

Вы хотите, чтобы ваши приложения были удобны и легки в использовании? В этой главе мы изучим элементы, которые позволят создавать пользовательский интерфейс для ваших проектов. Это позволит приложениям быть гибкими, максимально отображать бизнес-логику проекта и предоставлять оптимальные удобства в работе.

Для создания автоматизированных приложений можно использовать два вида элементов управления. Ну, во-первых, это элементы управления, которые могут быть размещены на рабочем листе, а, во-вторых, в своих проектах вы можете также использовать формы, которые создаются в редакторе Visual Basic.

Элементы управления рабочего листа помогут создать индивидуальный интерфейс проекта, внедренный непосредственно в рабочий лист, т. е. интерфейс, который максимально приближен к пользователю. Они позволят автоматизировать и алгоритмизировать работу пользователя, тем самым упрощая ее и повышая эффективность работы. Элементы управления такого типа также создадут необходимую защиту ваших данных, с их помощью можно достичь того, чтобы пользователь мог действовать только в рамках бизнес-логики проекта, не нарушая его целостности.

Другое дело, использование формы, которую вы создали в редакторе Visual Basic. Такая форма представляет собой диалоговое окно, в котором можно размещать различные элементы управления. В приложении может быть как одна, так и несколько форм. Используя комбинацию форм и элементов управления, вы можете реализовать любой пользовательский интерфейс, что, несомненно, придаст вашему приложению законченный вид.

Примечание

Файлы рассмотренных в данной главе примеров можно найти в папке Glava_4 на компакт-диске.

Используем элементы управления на рабочем листе

Сначала мы познакомимся с использованием элементов управления, которые можно разместить на рабочем листе Excel. С их помощью у вас получится создавать приложения, которые облегчат ввод и изменение данных, позволят производить необходимые функции автоматизации, а также устанавливать защиту на диапазон ячеек рабочего листа.

О панели инструментов *Элементы управления*

Элементы управления представляют собой составные части графического интерфейса Windows. Примерами элементов управления являются кнопки, поля ввода, списки, полосы прокрутки и другие элементы интерфейса, с помощью которых можно ввести число, выбрать значение или совершить какое-нибудь другое действие. На рабочем листе Excel можно размещать различные элементы управления. Элементы управления доступны при нажатии кнопки **Вставить**, которая расположена на вкладке **Разработчик** ленты в группе **Элементы управления**. Следует заметить, что при нажатии кнопки **Вставить** доступны две группы элементов управления: **Элементы управления формы** и **Элементы ActiveX** (рис. 4.1).

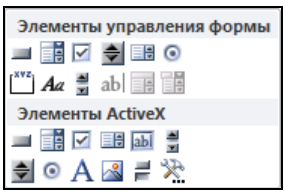


Рис. 4.1. Панель инструментов для рабочего листа

Группа **Элементы управления формы** предназначена, прежде всего, для обеспечения совместимости с документами ранних версий Excel (до Excel 97), использующими соответствующие элементы управления. Они обладают намного меньшими возможностями по сравнению с элементами управления, которые размещены на панели **Элементы ActiveX**. Некоторые из этих элементов вообще не могут быть использованы в документах Excel последних версий — это **Поле**, **Поле со списком** и **Поле с раскрывающимся списком**. Однако они обладают рядом возможностей, которые отсутствуют у элементов управления, расположенных на панели **Элементы ActiveX**, — например, их можно поместить на листы диаграмм.

Элементы управления группы **Элементы ActiveX** (ActiveX Controls) являются независимыми компонентами различных приложений и, в том числе, могут использоваться в Excel. В табл. 4.1 приведен список основных элементов управления и соответствующих кнопок панели инструментов.

Таблица 4.1. *Элементы управления из панели инструментов*

Элемент управления	Имя	Префикс	Кнопка, его создающая
Поле	TextBox	`txt`	
Надпись	Label	`lbl`	
Кнопка	CommandButton	`cmd`	
Список	ListBox	`lst`	
Поле со списком	ComboBox	`cbo`	
Полоса прокрутки	ScrollBar	`scr`	
Счетчик	SpinButton	`spn`	

Таблица 4.1 *(окончание)*

Элемент управления	Имя	Префикс	Кнопка, его создающая
Переключатель	OptionButton	`opt`	
Флажок	CheckBox	`chk`	
Выключатель	ToggleButton	`tgl`	
Рисунок	Image	`img`	
Другие элементы управления	More Controls		

Как расположить элемент управления на рабочем листе и написать код?

Как было указано ранее, элементы управления добавляются на лист Excel с использованием кнопки **Вставить**, которая находится на вкладке **Разработчик** в группе **Элементы управления**.

Создание элемента управления на рабочем листе включает в себя два этапа: размещение элемента управления на рабочем листе и его настройку.

Настройка подразумевает задание свойств элемента управления, в частности, можно связать элемент управления и некоторые ячейки рабочего листа, настроить внешний вид элемента управления и другие его параметры.

Элементу управления можно сопоставить макрос, который будет выполняться при наступлении некоторого *события*, например, при нажатии кнопки — для элемента управления **Кнопка** (Button или CommandButton) или же быть задана соответствующая процедура, написанная на языке VBA.

> ***Примечание***
>
> Элемент управления **Кнопка** имеет одно и то же название на панелях инструментов **Элементы управления формы** (Form Controls) и **Элементы ActiveX** (ActiveX Controls) в локализованной версии, но разные названия (Button или Command Button) в нелокализованной версии.

Чтобы поместить элемент управления на рабочий лист, выполните следующие действия:

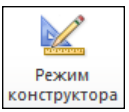
1. Нажмите кнопку **Вставить** на вкладке **Разработчик** ленты в группе **Элементы управления** и выберите в выпадающей панели соответствующий элемент управления в одной из групп: **Элементы управления формы** (Form Controls) или **Элементы ActiveX** (ActiveX Controls). Указатель мыши примет вид крестика.
2. Поместите указатель мыши в то место рабочего листа, где необходимо расположить элемент управления, и щелкните левой кнопкой мыши. Элемент управления появится на рабочем листе.
3. Перемещая белые квадратики — маркеры границы элемента управления (или просто маркеры) — измените его размер по своему усмотрению.

Если нужно выбрать один из дополнительных элементов управления диалогового окна:

1. Нажмите кнопку **Другие элементы** управления на панели **Элементы ActiveX**.
2. В появившемся списке выберите нужный элемент управления. Указатель примет вид крестика.
3. Переместите указатель в то место рабочего листа, где требуется расположить элемент управления, и щелкните кнопкой мыши. Элемент появится на рабочем листе.
4. Если необходимо, измените размер элемента управления.

Элемент управления не привязан к какому-то месту на рабочем листе и может быть свободно перемещен в любое другое место. Для перемещения объекта используется мышь или клавиатура. Мышь удобнее использовать для перемещения на большие расстояния.

Чтобы переместить элемент управления с помощью мыши:



1. Выделите нужный элемент управления. Чтобы выделить элемент управления **Элементы ActiveX**, нажмите кнопку **Режим конструктора** в группе **Элементы управления** на вкладке **Разработчик** и затем выберите элемент управления. Выделенный элемент имеет границу, на которой расположены маркеры.
2. Чтобы выделить элемент управления с панели **Элементы управления формы** (Form Controls), переместите на него указатель и щелкните левой кнопкой мыши при нажатой клавише <Ctrl>. Выделенный элемент имеет широкую серую границу, на которой расположены маркеры.
3. Перетащите элемент управления с помощью мыши на новое место. Перетаскивать элемент следует либо за границу элемента, либо за его графическое изображение. Если попытаться захватить название элемента, то может произойти переход в режим редактирования названия, и перетащить элемент не удастся.

> ***Совет***
>
> Чтобы выделить несколько элементов управления, выберите каждый из них, удерживая нажатыми клавиши <Ctrl>+<Shift>.
>
> Чтобы элемент управления перемещался строго по вертикали или горизонтали, во время перетаскивания удерживайте клавишу <Shift>. Чтобы при перетаскивании элемент управления выравнивался по линиям сетки, удерживайте клавишу <Alt>. Эффекты можно совместить, удерживая при перетаскивании обе эти клавиши.

Перемещение элемента управления с помощью клавиатуры удобнее в том случае, если необходимо точно позиционировать элемент на листе. Чтобы переместить элемент управления при помощи клавиатуры:

1. Выделите элемент управления, который нужно переместить.
2. Передвигайте его с помощью клавиш <←>, <↑>, <→> и <↓>.

Иногда требуется сделать несколько копий одного и того же элемента. Чтобы скопировать элемент управления:

1. Выделите нужный элемент управления.
2. Удерживая нажатой клавишу <Ctrl>, перетащите объект на то место, куда необходимо поместить копию. После того как кнопка мыши будет отпущена, копия объекта появится в указанном месте.

> **СОВЕТ**
>
> При копировании можно пользоваться клавишами <Alt> и <Shift> так же, как и при перемещении элементов управления.

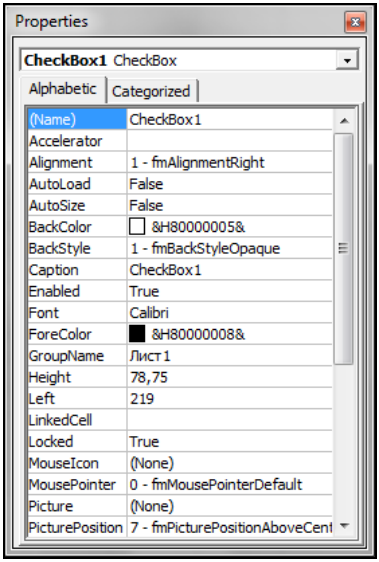
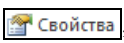
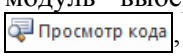


Рис. 4.2. Окно **Properties**

С элементами управления можно выполнять различные операции: их можно группировать, помещать на задний или передний планы, привязывать к объектам, выравнивать и т. д. С элементами управления эти действия выполняются так же, как и с рисунками, за исключением особенностей, которые имеются при осуществлении операции выделения.

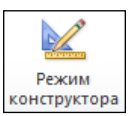
Элементы управления являются объектами. Как и любые другие объекты, они обладают свойствами, методами и событиями. Значения свойств элементов управления устанавливаются как в коде, так и на этапе их конструирования. Для того чтобы установить значения свойств на этапе конструирования, надо выделить элемент управления и нажать кнопку **Свойства** 

, которая расположена на вкладке **Разработчик** в группе **Элементы управления**. На экране отобразится окно **Properties** (рис. 4.2). В левой половине этого окна перечисляются свойства элемента, а в правой — имеются либо поля ввода, либо раскрывающиеся списки для установки значений этих свойств.

Как указывалось ранее, с элементом управления при наступлении какого-либо события можно связать макрос или процедуру VBA. Код процедуры, обрабатывающий события, связанные с элементом управления, набирается в модуле рабочего листа, на котором расположен данный элемент управления. Для перехода в этот модуль выберите элемент управления и нажмите кнопку **Просмотр кода** 

, которая расположена на вкладке **Разработчик** в группе **Элементы управления**.

По завершении конструирования элемента управления не забудьте выйти из режима конструирования, повторно нажав кнопку **Режим конструктора**.



Ваш первый проект с элементом управления

Создадим ваш первый проект с элементом управления: на рабочем листе расположим кнопку, нажатие которой приведет к отображению на экране диалогового окна с приветствием "Hello, World!".

Итак, выполните следующие действия.

1. Нажмите элемент управления **Кнопка**  панели инструментов **Элементы ActiveX**, которая вызывается нажатием кнопки **Вставить** в группе **Элементы управления** на вкладке **Разработчик** ленты.
2. Нарисуйте на рабочем листе элемент управления **Кнопка** необходимого размера.

Примечание

На поверхности первого созданного элемента управления **Кнопка** автоматически будет отображена надпись **CommandButton1**. Если вы создадите второй элемент управления **Кнопка**, то на его поверхности отобразится надпись **CommandButton2** и т. д. Текст, отображаемый на поверхности элемента управления, устанавливается значением свойства `Caption`. Кроме того, VBA устанавливает значение свойства `Name` объекта (т. е. его имя), используемое по умолчанию. Для первой созданной кнопки значение свойства `Name` устанавливается VBA равным `CommandButton1`, для второй — `CommandButton2` и т. д.

3. Выделите созданную кнопку. Нажмите кнопку **Свойства**, которая расположена на вкладке **Разработчик** в группе **Элементы управления**. На экране отобразится окно **Properties**. Установите в этом окне значение свойства `Name` равным `cmdHello` вместо безликого `CommandButton1`. Установите значение свойства `Caption` равным `Hello`.

   Совет

   Установка смыслового значения свойства `Name` элемента управления упрощает чтение кода, т. к. помогает по имени элемента управления распознать функцию, выполняемую им в проекте.

4. Еще раз выделите созданную кнопку. Нажмите кнопку **Просмотр кода** 

, расположенную на вкладке **Разработчик** в группе **Элементы управления**. В результате откроется редактор Visual Basic, причем в редакторе кода автоматически будет создана первая и последняя инструкции обработки события `Click` кнопки, генерируемого при ее нажатии (листинг 4.1, *а*, см. файл *1-Hello World.xlsm* на компакт-диске).

Листинг 4.1, *а*. "Hello, World!". Заготовка кода

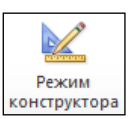
```
Private Sub cmdHello_Click()
End Sub
```

5. В процедуру обработки события `Click` кнопки добавьте инструкцию, которая отобразит на экране диалоговое окно с приветствием (листинг 4.1, *б*, см. также файл *1-Hello World.xlsm* на компакт-диске).

Листинг 4.1, *б*. "Hello, World!"

```
Private Sub cmdHello_Click()
   Msgbox "Hello, World!", vbExclamation
End Sub
```

6. Нажмите кнопку **Режим конструктора**, расположенную на вкладке **Разработчик** в группе **Элементы управления**, для того, чтобы выйти из режима конструирования.



Проект готов. Теперь можно и протестировать созданную кнопку **Hello**. Нажмите ее, и если на экране отобразится диалоговое окно с приветствием, то вы сделали все правильно (рис. 4.3).

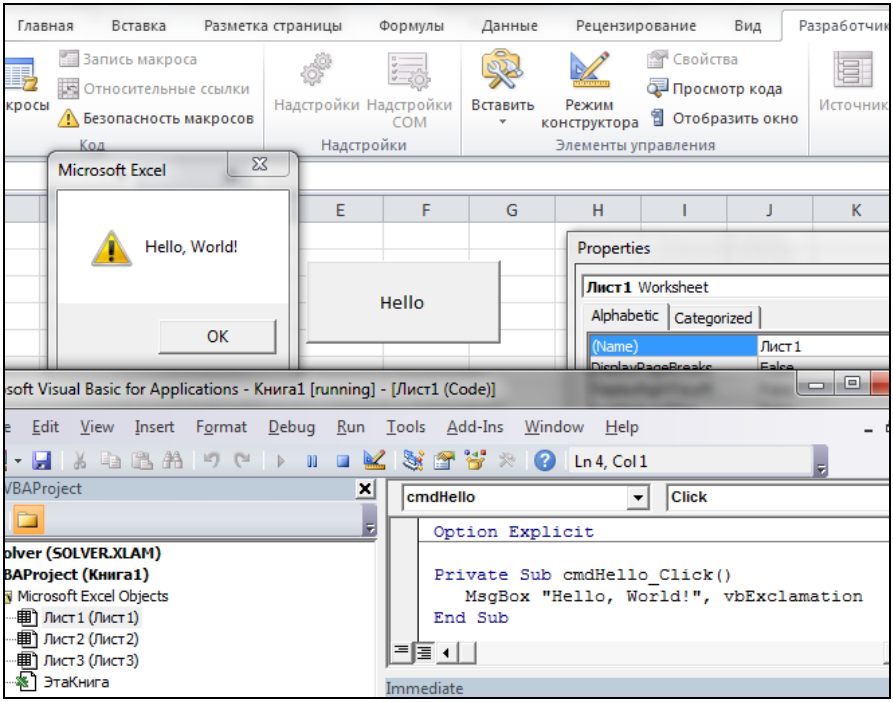


Рис. 4.3. Проект "Hello, World!". Рабочий лист и редактор Visual Basic

Общие свойства элементов управления

Элементы управления обладают большой коллекцией свойств, позволяющих устанавливать различные параметры объекта от его местоположения и размеров до отображаемого в нем текста и рисунка. В табл. 4.2 перечислены общие свойства элементов управления.

***Таблица 4.2.** Общие свойства элементов управления*

Свойство	Описание
`AutoSize`	Специфицирует, надо ли автоматически изменять размеры элемента управления с тем, чтобы в нем умещалась вся выводимая информация
`BackColor`	Задает цвет фона
`BackStyle`	Устанавливает прозрачность фона. Допустимыми значениями являются следующие константы: `fmBackStyleTransparent` и `fmBackStyleOpaque`
`BottomRightCell`, `TopLeftCell`	Возвращают ссылки на ячейки, расположенные под левым верхним и правым нижним углами элемента управления
`Caption`	Задает строку текста, отображаемую на элементе управления
`ControlTipText`	Возвращает текст всплывающей подсказки

Таблица 4.2 *(окончание)*

Свойство	Описание
Enabled	Устанавливает, достижим ли для пользователя элемент управления
Font	Возвращает объект Font для задания параметров шрифта
ForeColor	Задает цвет шрифта
Height, Width	Устанавливают высоту и ширину элемента управления
Left, Top	Координаты левого верхнего угла элемента управления
MouseIcon	Назначает пользовательский указатель мыши
MousePointer	Специфицирует тип указателя мыши
Name	Назначает имя объекта
OldHeight, OldWidth	Возвращают высоту и ширину элемента управления **Кнопка** до изменения его размеров
OldLeft, OldTop	Возвращают координаты левого верхнего угла кнопки до ее перемещения
Parent	Специфицирует ссылку на объект-контейнер для данного элемента управления
Picture	Задает ссылку на файл с растровым изображением, используемым в качестве фона элемента управления
PicturePosition	Устанавливает местоположение рисунка по отношению к надписи. Допустимыми значениями являются следующие константы: fmPicturePositionLeftTop, fmPicturePositionLeftCenter, fmPicturePositionLeftBottom, fmPicturePositionRightTop, fmPicturePositionRightCenter, fmPicturePositionRightBottom, fmPicturePositionAboveLeft, fmPicturePositionAboveCenter, fmPicturePositionAboveRight, fmPicturePositionBelowLeft, fmPicturePositionBelowCenter, fmPicturePositionBelowRight, fmPicturePositionCenter
PrintObject	Определяет, надо ли выводить элемент управления при печати
Tag	Задает параметр, используемый для идентификации конкретного элемента управления
TakeFocusOnClick	Устанавливает, получает ли элемент управления фокус после щелчка на нем
Visible	Задает видимость элемента управления
WordWrap	Устанавливает, надо ли переносить слова, если они не помещаются в строке, отображаемой в элементе управления

Общие методы элементов управления

Элементы управления обладают несколькими общими методами, позволяющими перемещать, располагать их и управлять фокусом. В табл. 4.3 перечислены эти методы.

Таблица 4.3. *Общие методы элементов управления*

Методы	Описание
`Move`	Перемещает элемент управления
`SetFocus`	Устанавливает фокус на элементе управления
`BringToFront`, `SendToBack`	Располагают элемент управления на переднем или заднем плане
`ZOrder`	Располагает элемент управления на переднем или заднем плане по отношению к другим элементам управления в зависимости от значения параметра, допустимыми значениями которого являются следующие константы: `fmTop` и `fmBottom`

Общие события элементов управления

Элементы управления имеют большую коллекцию процедур, способных перехватывать и обрабатывать различные события, происходящие в системе, и действия, произведенные пользователем, от щелчка мышью до обнаружения ошибки. В табл. 4.4 перечислены общие для элементов управления события.

Таблица 4.4. *Общие события элементов управления*

Событие	Описание
`BeforeDragOver`	Происходит при буксировке данных
`BeforeDropOrPaste`	Происходит перед вставкой данных, производимой буксировкой
`Click`	Происходит, когда пользователь щелкает на элементе управления
`DblClick`	Происходит, когда пользователь дважды щелкает мышью на элементе управления
`Enter`, `Exit`	Происходят, когда элемент управления получает или теряет фокус
`Error`	Происходит, когда элемент управления обнаружил ошибку, но не может передать сообщение
`KeyDown`, `KeyUp`	Происходят, когда пользователь нажимает и отпускает любую клавишу на клавиатуре, а элемент управления имеет фокус
`KeyPress`	Происходит, когда пользователь нажимает любую клавишу, кроме функциональных клавиш, клавиш управлением курсором и служебных клавиш, а элемент управления имеет фокус
`MouseDown`, `MouseUp`	Происходят, когда пользователь нажимает и отпускает любую кнопку мыши
`MouseMove`	Происходит, когда пользователь передвигает указатель мыши над элементом управления

Кнопка (CommandButton)

Элемент управления **Кнопка** (CommandButton) в основном используется для инициирования выполнения некоторых действий, вызываемых нажатием кнопки, например запуск программы или остановка ее выполнения, печать результатов и т. д. Таким образом, основным событием, связанным с кнопкой, является событие `Click`. Основным свойством кнопки является свойство `Caption`, возвращающее и устанавливающее текст, отображаемый на поверхности кнопки.

Кнопочное меню

Рассмотрим бизнес-ситуацию создания кнопочного меню. В книге имеются три рабочих листа. На первом из них расположены две кнопки с именами двух других листов. Нажатие кнопки приводит к активизации одноименного листа, причем при нажатии второй кнопки производится не только активизация, но и прокрутка листа с тем, чтобы указанная ячейка отображалась в левом верхнем углу окна листа.

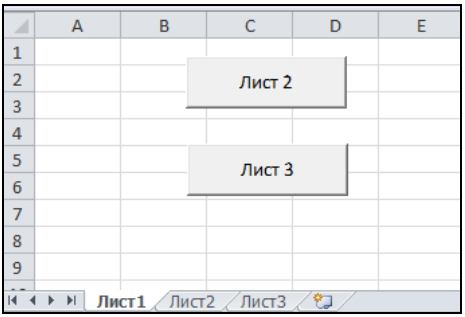


Рис. 4.4. Кнопочное меню

Итак, создайте рабочую книгу с листами **Лист1**, **Лист2** и **Лист3**. На рабочем листе **Лист1** расположите две кнопки (рис. 4.4). При помощи окна **Properties** установите им значения свойств, как показано в табл. 4.5.

Таблица 4.5. *Значения свойств, установленные в окне* ***Properties***

Объект	Свойство	Значение
Кнопка	`Name`	`cmdSheet2`
	`Caption`	`Лист 2`
Кнопка	`Name`	`cmdSheet3`
	`Caption`	`Лист 3`

В модуле рабочего листа **Лист1** наберите необходимый код (см. файл *2-Кнопочное меню.xlsm* на компакт-диске). Проект готов. Активизация листа производится, конечно, методом `Activate` объекта `Worksheet`. Прокрутка же листа с тем, чтобы указанная ячейка (в данном случае **T30**) отображалась в левом верхнем углу окна листа, осуществляется свойствами `ScrollColumn` и `ScrollRow` объекта `Window`.

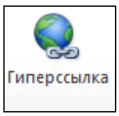
Примечание

Для прокрутки экрана с тем, чтобы отобразился искомый диапазон, можно также применить метод `Show` объекта `Range`. Например, `Cells(30, 20).Show`.

Навигация по книге при помощи гиперссылок

Конечно, навигацию по книге можно производить при помощи кнопочного меню, но можно и классическим способом, без написания кода, средствами гиперссылок (см. файл *3-Навигация гиперссылками.xlsm* на компакт-диске). Итак, в рабочей книге с тремя рабочими листами **Лист1**, **Лист2** и **Лист3**:

1. Выберите лист **Лист1**.
2. В ячейку **B1** введите `Лист2`.
3. Нажмите кнопку **Вставить гиперссылку**, которая расположена в группе **Ссылки** на вкладке **Вставка** ленты.



4. В открывшемся окне **Вставка гиперссылки** можно установить необходимые параметры для гиперссылки. Переключатели **файлом**, **веб-страницей**, **местом в документе**, **новым документом**, **электронной почтой** группы **Связать с** указывают на документ, в который будет дана ссылка в гиперссылке. Поле **Текст** задает текст всплывающей подсказки. В поле **Введите адрес ячейки** дается ссылка на ячейку в документе, в которую происходит переход. В поле **Или выберите место в документе** приводится список листов и именованных объектов книги.
5. Выберите в окне **Вставка гиперссылки** следующие параметры: установите переключатель в положение **местом в документе**, в поле **Введите адрес ячейки** укажите `A1`, далее в поле **Или выберите место в документе** выберите **Лист2**. Нажмите кнопку **OK**.
6. В ячейку **B2** введите `Лист3`. Нажмите кнопку **Вставить гиперссылку**, которая расположена в группе **Ссылки** на вкладке **Вставка** ленты. На экране отобразится окно **Вставка гиперссылки** (рис. 4.5).

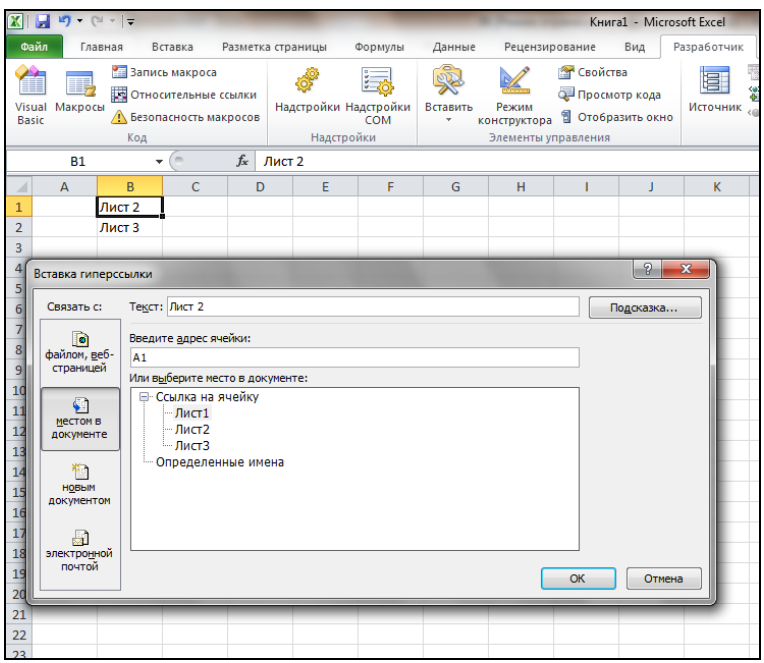


Рис. 4.5. Окно **Добавление гиперссылки**

7. Выберите переключатель **местом в документе** группы **Связать с**. В поле **Введите адрес ячейки** введите `T30`. В поле **Или выберите место в документе** выберите **Лист3**. Нажмите кнопку **OK**.

Кнопочный сценарий

Кнопки позволяют производить не только навигацию, но и вводить данные в ячейки, тем самым создавая кнопочные сценарии, например, если необходимо отобразить итоговую сумму, скажем, расходов или прибыли при различных сочетаниях ее компонентов. В качестве примера рассмотрим задачу нахождения итоговой суммы $a + b + c$ трех переменных при двух возможных вариантах их значений, приведенных в табл. 4.6.

***Таблица 4.6.** Варианты значений переменных*

	I	II
a	3	2
b	4	3
c	5	4

Итак:

1. На рабочем листе ячейки **B2**, **B3** и **B4** отведите под переменные *a*, *b*, *c*.
2. В ячейку **B5** введите формулу нахождения искомой суммы $a + b + c$: `=СУММ(B2:B4)`.
3. В диапазон **D3:D5** введите первый вариант значений переменных *a*, *b*, *c*.
4. В диапазон **E3:E5** введите второй вариант значений переменных *a*, *b*, *c*.
5. Создайте две кнопки и, используя окно **Properties**, установите им значения свойств, как показано в табл. 4.7 (рис. 4.6).
6. В модуле рабочего листа **Лист1** наберите код (см. файл *4-Кнопочный сценарий.xlsm* на компакт-диске).

Таблица 4.7.** Значения свойств, установленные в окне **Properties

Объект	Свойство	Значение
Кнопка	`Name`	`cmdVar1`
	`Caption`	`Вариант 1`
Кнопка	`Name`	`cmdVar1`
	`Caption`	`Вариант 2`

Нажатие кнопки **Вариант 1** произведет считывание значений при помощи свойства `Value` из ячеек диапазона **D3:D5** и ввод их с помощью свойства `Value` в ячейки диапазона **B2:B4**. Нажатие же кнопки **Вариант 2** вызовет считывание значений из ячеек диапазона **E3:E5** и ввод их в ячейки диапазона **B2:B4**. Проект кнопочного сценария готов.

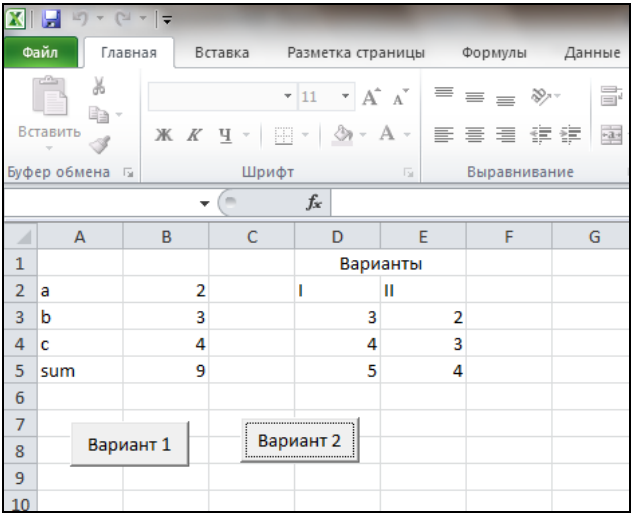


Рис. 4.6. Кнопочный сценарий

Кнопочный сценарий для ввода формул с кнопками, украшенными рисунками, и пользовательским указателем мыши

Кнопки, за счет внедрения в них рисунков, могут принять более презентабельный вид. Кроме того, изменение указателя мыши, отображаемого над кнопкой, с указывающей стрелки, например, на нажимающий палец также может придать интерфейсу дополнительную визуальную привлекательность.

Рисунок загружается на поверхность кнопки при помощи свойства `Picture`. Свойство `PicturePosition` позволяет установить взаимное расположение текста и рисунка, отображаемых на кнопке. Свойство `MousePointer` устанавливает указатель мыши. Допустимые значения этого свойства перечислены в табл. 4.8. Если значение свойства `MousePointer` равно `fmMousePointerCustom`, то курсор создается на основе cur-файла, ссылка на который устанавливается свойством `MouseIcon`. В коде значения свойств `Picture` и `MouseIcon` задаются функцией `LoadPicture`, в качестве значения параметра которой надо указать имя соответствующего графического файла.

Таблица 4.8. *Допустимые значения свойства* `MousePointer`

Константа	Значение	Описание
`fmMousePointerDefault`	0	По умолчанию
`fmMousePointerArrow`	1	
`fmMousePointerCross`	2	
`fmMousePointerIBeam`	3	
`fmMousePointerSizeNESW`	6	
`fmMousePointerSizeNS`	7	

Таблица 4.8 *(окончание)*

Константа	Значение	Описание
fmMousePointerSizeNWSE	8	
fmMousePointerSizeWE	9	
fmMousePointerUpArrow	10	
fmMousePointerHourglass	11	
fmMousePointerNoDrop	12	
fmMousePointerAppStarting	13	
fmMousePointerHelp	14	
fmMousePointerSizeAll	15	
fmMousePointerCustom	99	Определенный пользователем

В качестве примера использования кнопок с рисунками сконструируем еще один кнопочный сценарий, но в этот раз в ячейки рабочего листа будем вводить не числа, а формулы, по которым производятся расчеты. Абстрагируясь, рассмотрим задачу нахождения значения выражений $x + y$ и $x - y$ (рис. 4.7, см. также файл *5-Кнопочный сценарий украшенный картинками.xlsm* на компакт-диске).

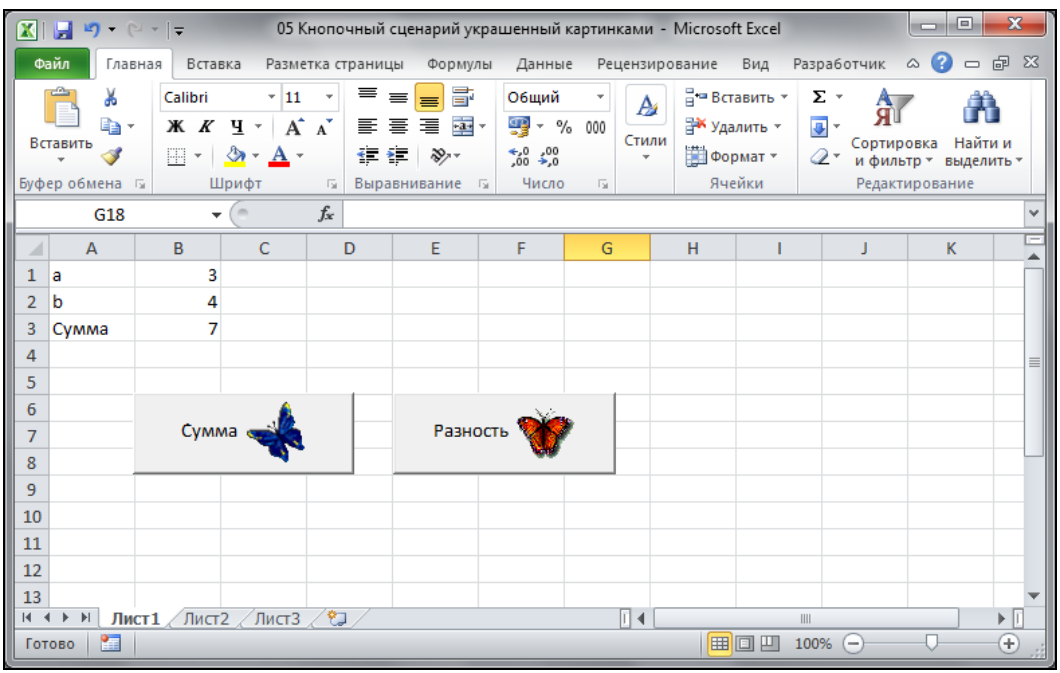


Рис. 4.7. Кнопочный сценарий для ввода формул

Итак:

1. На рабочем листе отведите ячейки **B1** и **B2** под переменные *x* и *y*. В ячейку **B3** в соответствии с выбранным сценарием будет вводиться из кода либо формула `=B1+B2`, либо `=B1-B2`.
2. Создайте две кнопки и, используя окно **Properties**, установите им значения свойств, как показано в табл. 4.9.
3. В модуле рабочего листа наберите код, показанный в листинге 4.2.

Таблица 4.9. *Значения свойств, установленные в окне* ***Properties***

Объект	Свойство	Значение
Кнопка	`Name`	`cmdSum`
	`Caption`	`Сумма`
	`Picture`	Ссылка на файл с рисунком. В данном случае была произведена ссылка на файл с изображением бабочки
	`PicturePosition`	`fmPicturePositionRightCenter`
	`MousePointer`	`fmMousePointerAppStarting`
Кнопка	`Name`	`cmdSub`
	`Caption`	`Разность`
	`Picture`	Ссылка на файл с рисунком. В данном случае была произведена ссылка на файл с изображением бабочки
	`PicturePosition`	`fmPicturePositionAboveCenter`
	`MousePointer`	`fmMousePointerCustom`
	`MouseIcon`	Ссылка на файл с курсором. В данном случае была произведена ссылка на файл Inodrop.cur, который можно найти в каталоге C:\Windows\Cursors

Нажатие кнопки **Сумма** произведет ввод в ячейку **B3** формулы `=B1+B2`, а в ячейку **A3** строки `Сумма`. Нажатие же кнопки **Разность** вызовет ввод в ячейку **B3** формулы `=B1-B2`, а в ячейку **A3** строки `Разность`.

Проект кнопочного сценария готов.

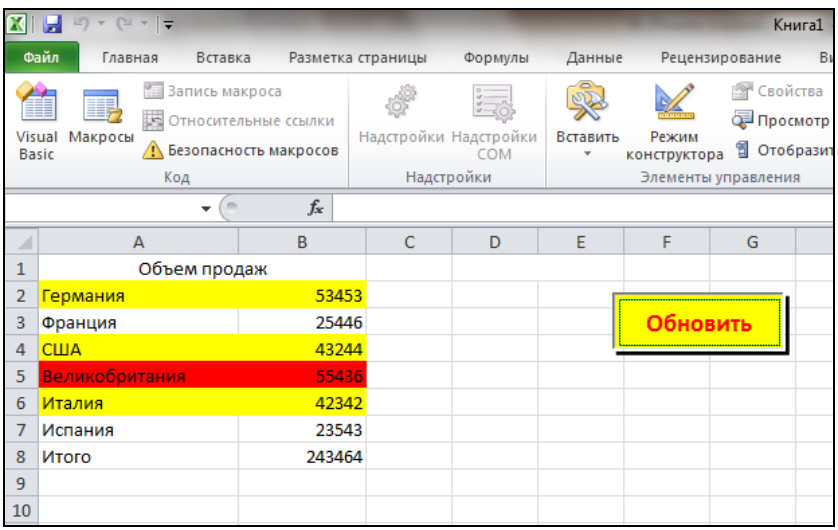
Листинг 4.2. Кнопочный сценарий для ввода формул

```
Private Sub cmdSum_Click()
   Range("B3").Formula = "=B1+B2"
   Range("A3").Value = "Сумма"
End Sub

Private Sub cmdSub_Click()
   Range("B3").Formula = "=B1-B2"
   Range("A3").Value = "Разность"
End Sub
```

Интерактивная кнопка и определение среднего объема продаж

Управляя событиями `MouseDown` и `MouseUp`, можно придать кнопке большую интерактивность. Продемонстрируем, как это делается, на примере простого приложения. Пусть имеются некоторые данные по продажам фирмы "Альматеус", которая поставляет на экспорт различные резинотехнические изделия. Для более наглядного представления данных в таблице объема продаж необходимо выделить красным цветом строку с максимальным объемом продаж и желтым — строки, в которых объем продаж выше среднего (рис. 4.8). Данные о продажах постоянно меняются, поэтому вам необходимо создать такое приложение, которое бы автоматизировало процесс выделения цветом нужных строк.



	A	B	C	D	E	F	G
1	Объем продаж						
2	Германия	53453					
3	Франция	25446					
4	США	43244					
5	Великобритания	55436					
6	Италия	42342					
7	Испания	23543					
8	Итого	243464					
9							
10							

Рис. 4.8. Интерактивная кнопка и определение суммарных продаж

Для реализации описанного примера выполните следующие действия.

1. На рабочем листе отведите диапазон **B2:B7** под объемы продаж.
2. Введите в ячейку **B8** формулу определения суммарных продаж `=СУММ(B2:B7)`.
3. Создайте кнопку и, используя окно **Properties**, установите ей значения свойств, как показано в табл. 4.10.
4. В модуле рабочего листа **Лист1** наберите код (см. файл *6-Интерактивная кнопка.xlsm* на компакт-диске).

Таблица 4.10. *Значения свойств, установленные в окне* ***Properties***

Объект	Свойство	Значение
Кнопка	`Name`	`cmdRefresh`
	`Caption`	`Обновить`

Нажатие кнопки **Обновить** и вызовет пересчет и переоформление таблицы (см. также рис. 4.8). Процедуры обработки событий `MouseDown` и `MouseUp` программируют различный внешний вид кнопок, а именно стиль, размер и цвет шрифта (свойства `Bold`, `Size` объекта `Font` и свойство `ForeColor` кнопки), цвет фона и вывод тени (свойства `BackColor` и `Shadow` кнопки). Изменение этих свойств кнопки и вызывает эффект дополнительной интерактивности. Пересчет таблицы производится в процедуре `DoRefresh`. Максимальное и среднее значения находятся при помощи функций рабочего листа, инкапсулированных в объект `WorksheetFunction`. Цвет ячеек устанавливается свойством `Color` объекта `Interior`. Удаление заливки ячейки реализуется установкой значения свойства `ColorIndex` равным `xlColorIndexNone`.

Обмен значений между двумя выбранными ячейками

В качестве последнего примера по работе с кнопками на рабочем листе приведем следующий проект, способный обменивать значения между любыми двумя выделенными ячейками. Итак, на рабочем листе создайте кнопку, а, например, в ячейки **A2**, **B22**, **H17**, **J3** введите слова `Огонь`, `Воздух`, `Вода`, `Земля`. В модуле рабочего листа **Лист1** наберите код (см. файл *7-Обмен значений.xlsm* на компакт-диске), который обрабатывает событие `Click` кнопки. Вот и все, теперь остается выбрать какие-то две из этих ячеек и нажать кнопку, а значения в них поменяются местами автоматически. Кроме того, наша программа осуществляет контроль за выделенной областью: она должна состоять ровно из двух ячеек.

Переключатель (OptionButton)

Элемент управления **Переключатель** (OptionButton) позволяет выбрать одну из нескольких взаимоисключающих альтернатив. Переключатели обычно отображаются группами по выбираемым альтернативам. Группировка производится при помощи элемента управления **Рамка** или свойства `GroupName` объекта `OptionButton`. Основными событиями переключателя являются события `Click` и `Change`. Основным свойством переключателя является свойство `Value`, возвращающее или устанавливающее его состояние. Если значение этого свойства равно `True`, то переключатель установлен, а если `False` — то сброшен.

Переключатели и объемы продаж

Для рассмотрения примера использования переключателей вернемся к отчету о продажах фирмы "Альматеус". Пусть нам необходимо составить список тех стран, объем продаж в которых является либо максимальным, либо минимальным (рис. 4.9).

Для решения данной задачи выполните следующие действия:

1. На рабочем листе отведите диапазон **B2:B7** под объемы продаж, а диапазон **A2:A7** под названия стран, в которые эти продажи были произведены.
2. Введите в ячейку **B8** формулу определения суммарных продаж `=СУММ(B2:B7)`.
3. Создайте два переключателя и, используя окно **Properties**, установите им значения свойств, как показано в табл. 4.11.
4. В модуле рабочего листа **Лист1** наберите необходимый код (см. файл *8-Переключатель.xlsm* на компакт-диске).

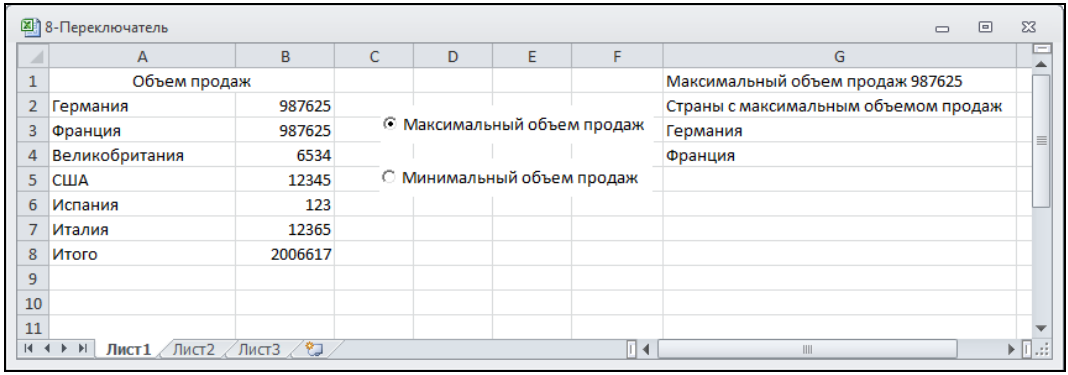


Рис. 4.9. Переключатели и объемы продаж

Таблица 4.11. *Значения свойств, установленные в окне* ***Properties***

Объект	Свойство	Значение
Переключатель	`Name`	`optMax`
	`Caption`	`Максимальный объем продаж`
Переключатель	`Name`	`optMin`
	`Caption`	`Минимальный объем продаж`

Установка переключателя **Максимальный объем продаж** или **Минимальный объем продаж** приведет к определению искомого объема продаж и составлению списка соответствующих стран. Величина искомого объема продаж выводится в ячейку **G1**, пояснительная надпись — в ячейку **G2**, и далее, начиная с ячейки **G3**, выводится составленный список стран. Страны с соответствующими объемами продаж находятся методом `Find` объекта `Range`. Все найденные ячейки методом `Find` объединяются в один диапазон. На основе этого диапазона методом `Offset` объекта `Range` создается диапазон уже с названиями стран, который методом `Copy` объекта `Range` копируется в указанное место. Обратите внимание на то, что программа сама определяет диапазон, в котором надо производить поиск. Существенным является только то, что код учитывает, что первая и последняя строки столбца **B** с данными содержат не исходные данные, а заголовок поля таблицы и итоговую сумму, и поэтому их не надо учитывать, что и делается последовательным применением методов `Resize` и `Offset`. Метод `AutoFit` автоматически изменяет ширину столбца **G** с тем, чтобы в нем уместилась вся выводимая информация.

Флажок (CheckBox) и *Выключатель* (ToggleButton)

Элемент управления **Флажок** (CheckBox) предоставляет пользователю возможность выбора. Флажок обычно имеет два состояния: установлен и сброшен, но может настраиваться на выбор из трех альтернатив. Флажок имеет те же основные свойства `Value` и `Capture`, что и переключатель. Кроме того, флажок обладает уни-

кальным свойством `TripleState`, позволяющим производить выбор из трех альтернатив. Допустимыми значениями свойства `TripleState` являются:

- `False` (выбор из двух альтернатив `True` и `False`, т. е. флажок может находиться только в двух состояниях — установлен и сброшен);
- `True` (выбор из трех альтернатив `True`, `False` и `Null`, т. е. флажок может находиться в трех состояниях — установлен, сброшен и нейтрален).

Элемент управления **Выключатель** (ToggleButton) предоставляет пользователю ровно те же возможности, что и флажок, но визуально он выглядит как кнопка.

Основным событием элементов управления **Флажок** и **Выключатель** является событие `Change`.

Флажок и управление отображением элементов диаграммы

Возвращаясь к отчету о продажах фирмы "Альматеус", создадим простое приложение, позволяющее управлять внешним видом диаграммы, а именно — при установленном флажке диаграмма выводится с тенью, а при снятом — без таковой (рис. 4.10).

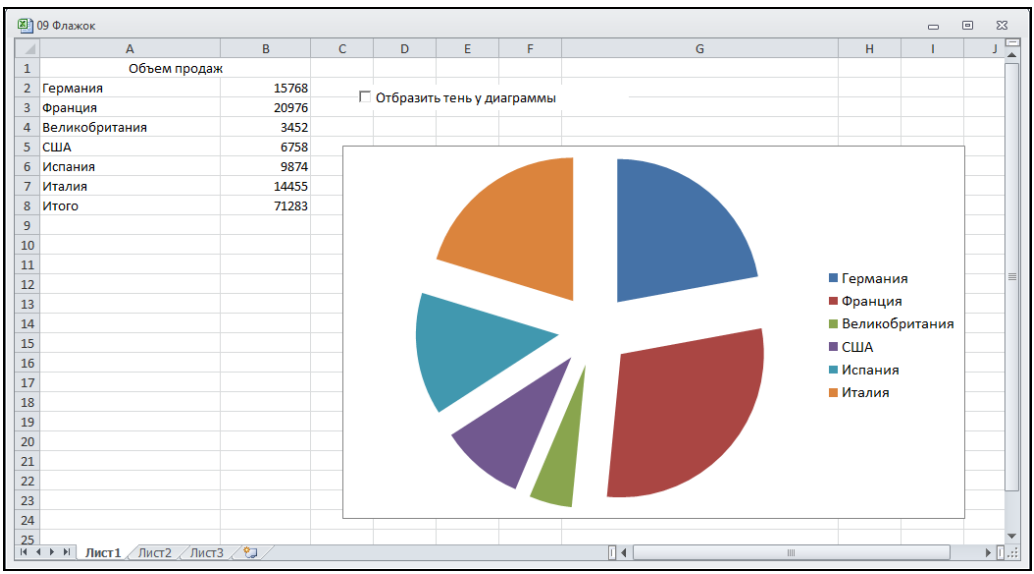


Рис. 4.10. Флажок и управление отображением элементов диаграммы

Итак, пошагово выполните следующие действия.

1. На рабочем листе отведите диапазон **B2:B7** под объемы продаж, а диапазон **A2:A7** — под названия стран, в которые эти продажи были произведены.
2. На основе диапазона **A2:B7** постройте диаграмму.
3. Создайте флажок и, используя окно **Properties**, установите ему значения свойств, как показано в табл. 4.12.
4. В модуле рабочего листа **Лист1** наберите код (см. файл *9-Флажок.xlsm* на компакт-диске).

Таблица 4.12. *Значения свойств, установленные в окне* ***Properties***

Объект	Свойство	Значение
Флажок	`Name`	`chkGraph`
	`Caption`	`Отобразить тень у диаграммы`

Установка флажка **Отобразить тень у диаграммы** вызовет ее отображение с тенью, а при снятом — без таковой. Все встроенные диаграммы образуют семейство `ChartObjects`. Так как в нашем случае имеется единственная диаграмма, то ее можно идентифицировать по номеру, который, само собой разумеется, будет 1. Отображением тени управляет свойство `Shadow` объекта `Chart`.

Выключатель и отображение примечаний

Продемонстрируем работу выключателя также на примере отчета о продажах фирмы "Альматеус". В этот проект теперь будут внедрены примечания. Установка выключателя приведет к одновременному отображению всех примечаний, а снятие выключателя — к их скрытию (рис. 4.11).

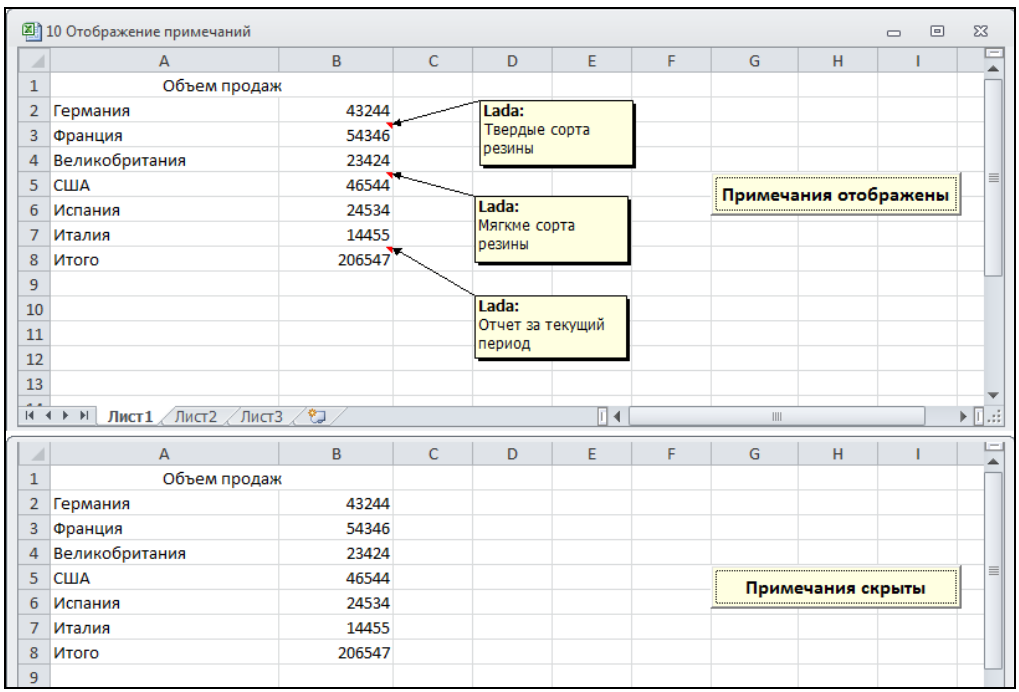


Рис. 4.11. Выключатель и отображение примечаний

Итак:

1. На рабочем листе отведите диапазон **B2:B7** под объемы продаж, а диапазон **A2:A7** под названия стран, в которые эти продажи были произведены.

2. В некоторые из ячеек диапазона **B2:B7**, используя кнопку **Создать примечание**, расположенную в группе **Примечание** на вкладке **Рецензирование** ленты, введите примечания.
3. Создайте выключатели и, используя окно **Properties**, установите ему значения свойств, как показано в табл. 4.13.
4. В модуле рабочего листа **Лист1** наберите код (см. файл *10-Отображение примечаний.xlsm* на компакт-диске).

Таблица 4.13. *Значения свойств, установленные в окне* ***Properties***

Объект	Свойство	Значение
Выключатель	`Name`	`tglCom`
	`Caption`	`Примечания отображены`

Установка выключателя вызовет отображение всех примечаний, его снятие приведет к скрытию как примечаний, так и их индикаторов, и, кроме того, надпись на выключателе сменится на **Примечания скрыты**. Управление отображением примечаний и их индикаторов реализуется свойством `DisplayCommentIndicator` объекта `Application`.

Полоса прокрутки (ScrollBar) и *Счетчик* (SpinButton)

Элемент управления **Полоса прокрутки** (ScrollBar) применяется для установки числового значения, причем этот элемент может устанавливать только целые неотрицательные значения. Основными событиями элемента управления **Полоса прокрутки** являются `Change`, `SpinUp` и `SpinDown`. В табл. 4.14 перечислены наиболее часто используемые свойства полосы прокрутки.

Таблица 4.14. *Свойства полосы прокрутки*

Свойство	Описание
`Value`	Возвращает или устанавливает текущее значение полосы прокрутки, которое может быть только целочисленным
`Min`	Минимальное значение полосы прокрутки
`Max`	Максимальное значение полосы прокрутки
`SmallChange`	Устанавливает шаг изменения значения при щелчке на одной из стрелок полосы прокрутки
`LargeChange`	Устанавливает шаг изменения значения при щелчке между ползунком и одной из стрелок полосы прокрутки
`LinkedCell`	Ссылка на ячейку значения свойства `Value`, синхронизированного со значением аналогичного свойства полосы прокрутки
`Orientation`	Устанавливает ориентацию полосы прокрутки. Допустимыми значениями являются следующие константы: `fmOrientationAuto` (ориентация зависит от размера элемента управления. Используется по умолчанию), `fmOrientationVertical` (вертикальное расположение), `fmOrientationHorizontal` (горизонтальное расположение)

Элемент управления **Счетчик** (SpinButton) по своим функциональным возможностям аналогичен полосе прокрутки. Если не быть чрезмерным буквоедом, то можно сказать, что счетчик — это полоса прокрутки без ползунка. Счетчик имеет те же свойства `Value`, `Min`, `Max` и `SmallChange`, что и полоса прокрутки.

Ввод значений в ячейку и управление цветом

В качестве примера использования полос прокрутки создадим приложение, демонстрирующее RGB-цветовую модель. Итак, на рабочем листе расположите три полосы прокрутки (рис. 4.12). Используя окно **Properties**, установите им значения свойств, как показано в табл. 4.15.

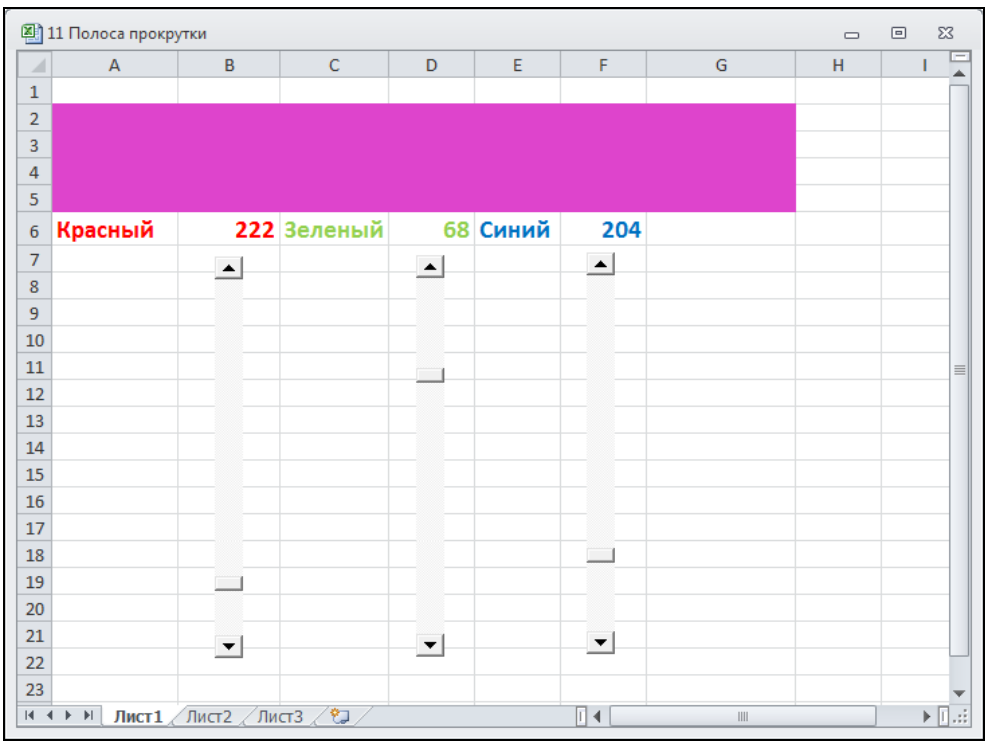


Рис. 4.12. Ввод значений в ячейку и управление цветом

Таблица 4.15. *Значения свойств, установленные в окне* ***Properties***

Объект	Свойство	Значение
Полоса прокрутки	`Name`	`scrRed`
	`Min`	`0`
	`Max`	`255`
	`LinkedCell`	`B6`

Таблица 4.15 *(окончание)*

Объект	Свойство	Значение
Полоса прокрутки	Name	ScrGreen
	Min	0
	Max	255
	LinkedCell	D6
Полоса прокрутки	Name	scrBlue
	Min	0
	Max	255
	LinkedCell	F6

В модуле рабочего листа **Лист1** наберите код (см. файл *11-Полоса прокрутки.xlsm* на компакт-диске). Каждая из полос прокрутки может изменять свое значение в пределах от 0 до 255, причем текущее значение отображается в той ячейке, на которую приведена ссылка в свойстве `LinkedCell`. Более того, эти ячейки работают согласованно с полосами прокрутки. Если пользователь изменит значение в ячейке, то одновременно изменится и значение свойства `Value` полосы прокрутки. Каждая из полос прокрутки отвечает за один из трех компонентов (красного, зеленого и синего) RGB-цветовой модели, в соответствии с которой устанавливается цвет фона диапазона **A2:G5**.

Ввод в ячейку с помощью полосы прокрутки и счетчика нецелочисленных значений

Свойство `Value` полосы прокрутки и счетчика может принимать только целочисленные значения. Если же надо управлять нецелочисленными значениями, то их просто следует масштабировать. Как это делается, покажем на следующем демонстрационном проекте, в котором построен график функции $F(x) = \cos(ax) \times \sin(bx)$. Значения же параметров a и b лежат в диапазоне от 0 до 10 и могут изменяться с шагом 0,1. Данные значения вводятся в ячейки с помощью счетчиков, а график при этом автоматически перестраивается (рис. 4.13).

Для выполнения данного примера проделайте следующие шаги.

1. На рабочем листе отведите под значения параметров *a* и *b* ячейки **B2** и **B4**.
2. В диапазон ячеек **E2:E12** введите результат табулирования переменной *x* от значения 0 до 1 с шагом 0,1.
3. В ячейку **F2** введите формулу `=COS($B$2*E2)*SIN($B$4*E2)`.
4. Выберите ячейку **F2**, расположите указатель мыши на маркере заполнения и протяните его вниз на диапазон **F2:F12**.
5. На основе диапазона **E2:F12** с помощью мастера диаграмм постройте график.
6. Создайте два счетчика и, используя окно **Properties**, установите им значения свойств, как показано в табл. 4.16.
7. В модуле рабочего листа **Лист1** наберите код (см. файл *12-Счетчик.xlsm* на компакт-диске).

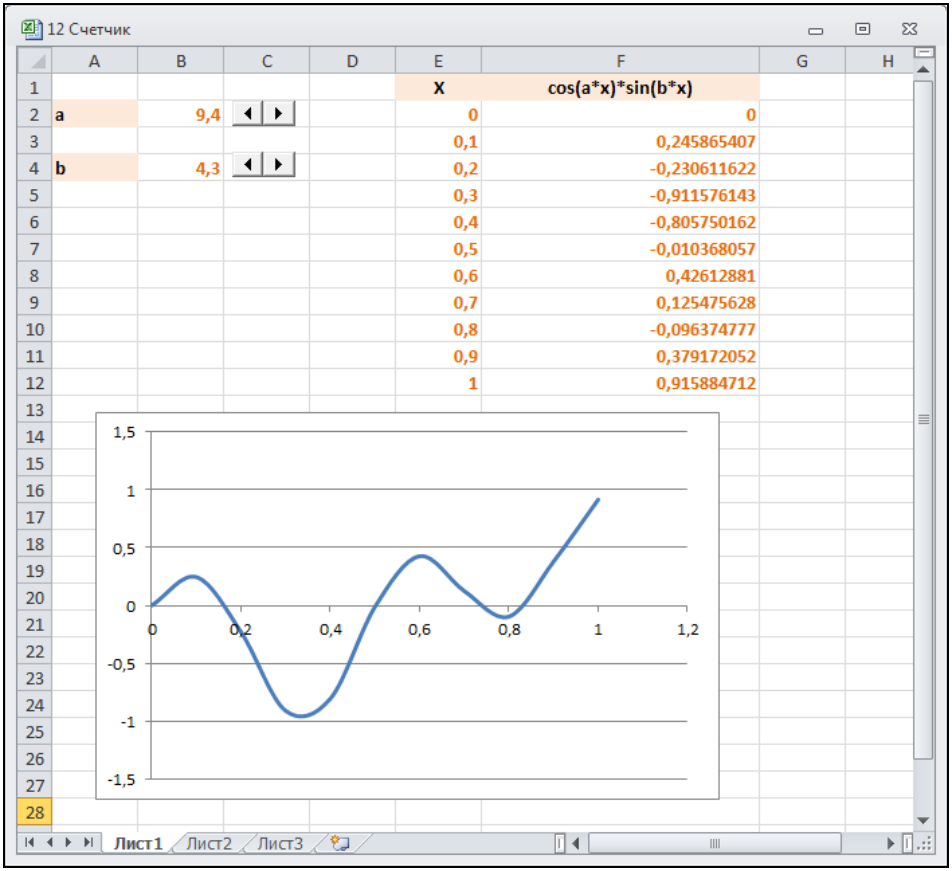


Рис. 4.13. Ввод в ячейку счетчиком нецелочисленных значений

Таблица 4.16. *Значения свойств, установленные в окне* ***Properties***

Объект	**Свойство**	**Значение**
Счетчик	`Name`	`spnA`
	`Min`	`0`
	`Max`	`100`
Счетчик	`Name`	`spnB`
	`Min`	`0`
	`Max`	`100`

Процедуры обработки события `Change` обеспечивают ввод масштабированных значений счетчиков в ячейки рабочего листа. Процедура же обработки события `Change` рабочего листа применяется для синхронизации работы ячеек и счетчиков, а именно изменение значений в ячейках **B2** и **B4** приводит к изменению значений счетчиков.

Список (ListBox)

Элемент управления **Список** (ListBox) применяется для хранения списка значений. В списке пользователь может выбрать одно или несколько значений, которые в последующем используются в тексте программы. Обратите внимание, что на этапе конструирования визуально список похож на поле ввода. Обычно выбор элемента из списка производится щелчком на элементе. Двойной же щелчок на элементе применяется для выполнения каких-то действий в программе, связанных с этим элементом. Событие `Change` генерируется при смене выбранного элемента.

В табл. 4.17 и 4.18 приведены наиболее часто используемые свойства и методы списка.

Таблица 4.17. *Свойства списка*

Свойство	Описание
`ListIndex`	Возвращает номер выбранного элемента списка. Нумерация элементов списка начинается с нуля. Если ни один элемент списка не выбран, то возвращает –1
`ListCount`	Возвращает число элементов списка
`TopIndex`	Возвращает элемент списка с наибольшим номером
`ColumnCount`	Устанавливает число столбцов в списке
`TextColumn`	Устанавливает столбец в списке, элементы из которого возвращаются в качестве значения свойства `Text`
`Text`	Возвращает выбранный в списке элемент
`List`	Возвращает элемент списка, стоящий на пересечении указанной строки и столбца
`ListFillRange`	Ссылка на диапазон, из которого заполняется список
`RowSource`	Устанавливает диапазон, содержащий элементы списка
`ControlSource`	Устанавливает диапазон (ячейку), куда возвращается выбранный элемент из списка
`MultiSelect`	Устанавливает способ выбора элементов списка. Допустимые значения: `fmMultiSelectSingle` (выбор только одного элемента), `fmMultiSelectMulti` (разрешен выбор нескольких элементов, выбор осуществляется либо щелчком, либо нажатием клавиши <Пробел>), `fmMultiSelectExtended` (разрешено использование клавиши <Shift> при выборе ряда последовательных элементов списка)
`Selected`	Логическое свойство, которое возвращает значение `True`, если элемент списка выбран, и `False` — в противном случае. Используется для определения выбранного элемента, когда значение свойства `MultiSelect` установлено равным `fmMultiSelectMulti` или `fmMultiSelectExtended`
`ColumnWidths`	Устанавливает ширину столбцов списка
`ColumnHeads`	Логическое свойство, определяющее, выводить ли в списке заголовки столбцов

Таблица 4.17 *(окончание)*

Свойство	Описание
`ListStyle`	Устанавливает способ выделения выбранных элементов. Допустимые значения: `fmListStylePlain` (выбранный элемент из списка выделяется цветом), `fmListStyleOption` (перед каждым элементом в списке располагается флажок, и выбор элемента из списка соответствует установке этого флажка)
`MatchEntry`	Выводит первый подходящий элемент из списка при наборе его имени с клавиатуры. Допустимые значения: `fmMatchEntryNone` (режим вывода подходящего элемента в списке отключен), `fmMatchEntryFirstLetter` (выводит подходящий элемент по набранной первой букве. В этом случае предпочтительно, чтобы элементы списка были упорядочены в алфавитном порядке), `fmMatchEntryComplete` (выводит подходящий элемент по полному набранному имени)
`BoundColumn`	Устанавливает данные, возвращаемые свойством `Value`. Допустимые значения: 0 (свойством `Value` возвращается индекс выбранной строки, т. е. в этом случае оно действует как свойство `ListIndex`), от 1 до количества столбцов в списке (свойством `Value` возвращается элемент из выбранной строки, стоящий в столбце, заданном значением свойства `BoundColumn`)

Таблица 4.18. *Методы списка*

Метод	Описание
`Clear`	Удаляет все элементы из списка
`RemoveItem`	Удаляет элемент из списка с указанным значением индекса
`AddItem`	Добавляет элемент в список

Сценарии со списком

В качестве примера использования списков вернемся к сценарию нахождения суммы $a + b + c$ при различных вариантах значений слагаемых. Только в этот раз управлять сценариями будем не кнопками, а списком (рис. 4.14).

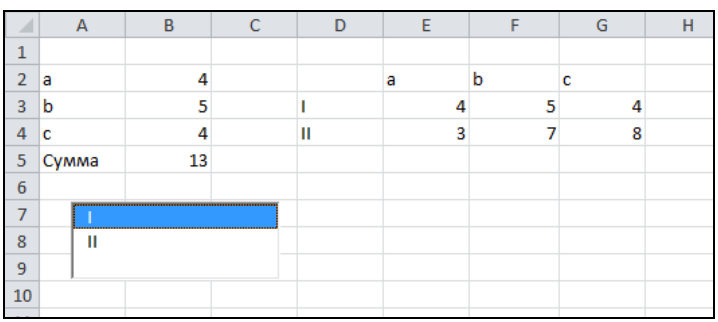


Рис. 4.14. Сценарии со списком

Итак:

1. На рабочем листе ячейки **B2**, **B3** и **B4** отведите под значения слагаемых *a*, *b* и *c*.
2. В ячейку **B5** введите формулу `=СУММ(B2:B4)`.
3. В диапазон **E3:G3** введите первый вариант возможных значений слагаемых *a*, *b* и *c*, а в ячейку **D3** — название первого варианта.
4. В диапазон **E4:G4** введите второй вариант возможных значений слагаемых *a*, *b* и *c*, а в ячейку **D4** — название второго варианта.
5. Создайте список и, используя окно **Properties**, установите ему значения свойств, как показано в табл. 4.19.

Таблица 4.19. *Значения свойств, установленные в окне* ***Properties***

Объект	Свойство	Значение
Список	`Name`	`lstVar`
	`ListFillRange`	`D3:D4`

В модуле рабочего листа **Лист1** наберите код (см. файл *13-Список.xlsm* на компакт-диске). Свойство `ListFillRange` заполняет список именами вариантов на основе тех значений, которые введены в указанный диапазон. Процедура обработки события `Change` списка при изменении выбранного элемента определяет его индекс, который возвращается свойством `ListIndex`. Этот индекс позволяет также идентифицировать диапазон, в котором хранятся значения слагаемых выбранного варианта. Специфицированный диапазон сначала копируется в буфер обмена, а затем с помощью специальной вставки с транспонированием вставляется в диапазон **B2:B4**, отведенный под слагаемые.

Защита ячеек рабочего листа

В проектах с пользовательским интерфейсом часто необходима поддержка целостности, чтобы пользователь не мог нарушать его структуру и производил только позволенные в соответствии с бизнес-сценарием действия, а именно чтобы ему был разрешен ввод данных только в специфицированные ячейки, а попытки ввода в остальные ячейки блокировались программно. Подобный эффект достигается использованием метода `Protect`, устанавливающего защиту на рабочий лист со значением параметра `UserInterfaceOnly`, равным `True`. Эта установка запрещает ввод пользователем данных во все незаблокированные ячейки, в то же время, позволяя это делать программно. Свойство же `Locked` контролирует блокировку ячеек.

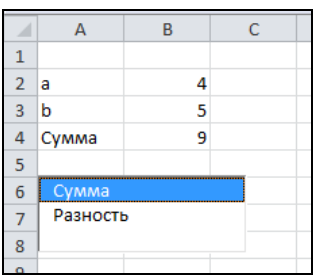


Рис. 4.15. Защита ячеек рабочего листа

Следующий простой проект (см. модуль **ЭтаКнига** и модуль рабочего листа **Лист1** в файле *14-Защита ячеек.xlsm* на компакт-диске) демонстрирует работу с методом `Protect` и свойством `Locked`. Итак, по сценарию задачи требуется найти либо сумму, либо разность двух чисел, введенных в ячейки **B1** и **B2**. Выбор операции пользователь производит из списка. Ему также разрешен

ввод чисел в ячейки **B1** и **B2**, а все остальные ячейки для него заблокированы. Ввод же названия операции и формулы в ячейки **A3** и **B3** реализуется программно по сделанному пользователем в списке выбору (рис. 4.15).

Управление печатью элементов управления

В выводимом на печать отчете иногда требуется отображать элементы управления, а иногда это делать не нужно, если они выполняют вспомогательную роль. В этом случае на помощь приходит свойство `PrintObject` элементов управления, которое устанавливает, надо ли выводить элемент управления при печати. Если значение этого свойства равно `True`, то объект печатается, а если `False`, то не печатается. Для демонстрации работы этого свойства модифицируем проект из предыдущего раздела. Добавьте на рабочий лист две кнопки и флажок. Если флажок **Печатать элементы управления** установлен, то эти элементы и список выводятся на печать, а если снят, то не выводятся. Кнопка **Печать** отвечает за печать, а кнопка **Предварительный просмотр** — за предварительный просмотр рабочего листа перед печатью. На рис. 4.16 показан результат просмотра рабочего листа при условии, что флажок **Печатать элементы управления установлен**.

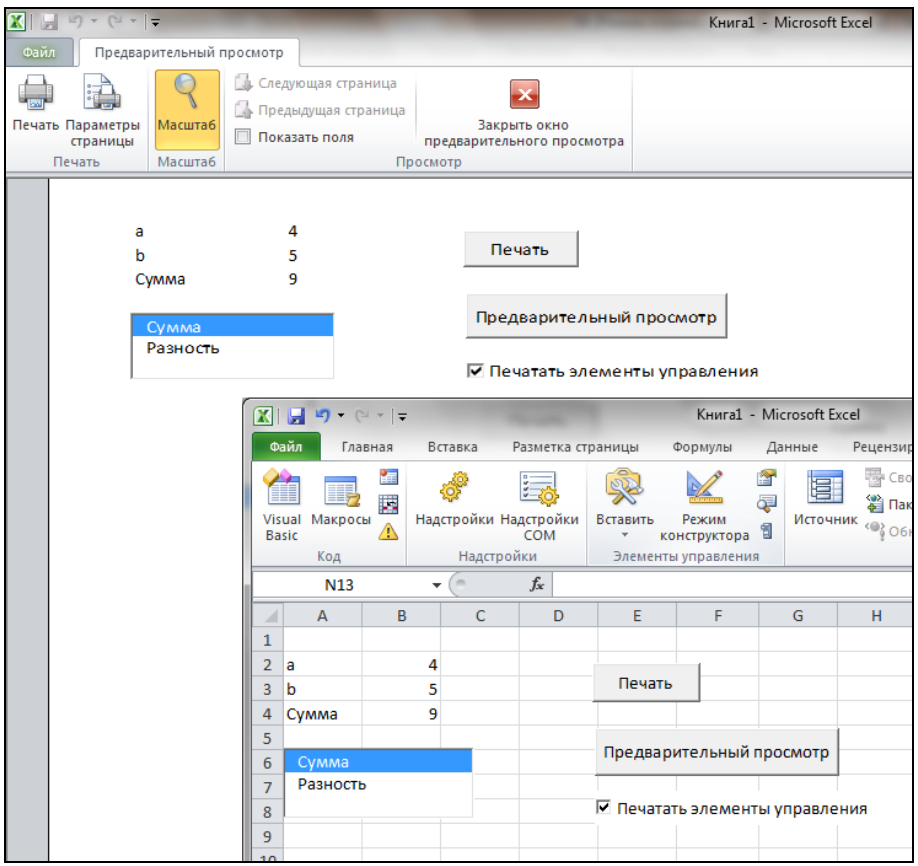


Рис. 4.16. Управление печатью элементов управления

Создайте список и, используя окно **Properties**, установите ему значения свойств, как показано в табл. 4.20.

В модуль рабочего листа **Лист1** добавьте необходимый код (см. файл *15-Управление печатью.xlsm* на компакт-диске). Проект готов.

Таблица 4.20. *Значения свойств, установленные в окне* ***Properties***

Объект	Свойство	Значение
Кнопка	`Name`	`cmdPrint`
	`Caption`	`Печать`
Кнопка	`Name`	`cmdPreview`
	`Caption`	`Предварительный просмотр`
Флажок	`Name`	`chkPrint`
	`Caption`	`Печатать элементы управления`

Создаем пользовательские формы с помощью VBA

Как отмечалось ранее, *пользовательская форма* — это диалоговое окно, в котором вы располагаете различные, необходимые в вашем приложении, элементы управления. Вы можете так продумать интерфейс для своего проекта, что в нем будет не одна, а несколько форм. Кроме того, набор элементов управления будет служить только для выполнения данной конкретной задачи. В любом случае, использование форм придаст вашему проекту индивидуальный вид, позволит максимально просто обращаться с различными данными приложения, а также сократит время на выполнение требуемых операций по их обработке.

Добавление формы в проект

Итак, для добавления пользовательской формы в проект выполните следующее:

1. Перейдите в редактор Visual Basic.
2. Выберите команду **Insert | UserForm**.

Новая форма добавлена в проект (рис. 4.17).

Примечание

Размеры формы можно изменять при помощи маркеров изменения размеров.

Семейство форм

Семейство `UserForms` является семейством, компоненты которого представляют все загруженные формы в приложении. Как и у всех семейств, у семейства `UserForms` имеются свойства `Count` (возвращает число компонентов в семействе) и `Item` (возвращает определенный компонент семейства), а также метод `Add` (добавляет к семейству новый компонент).

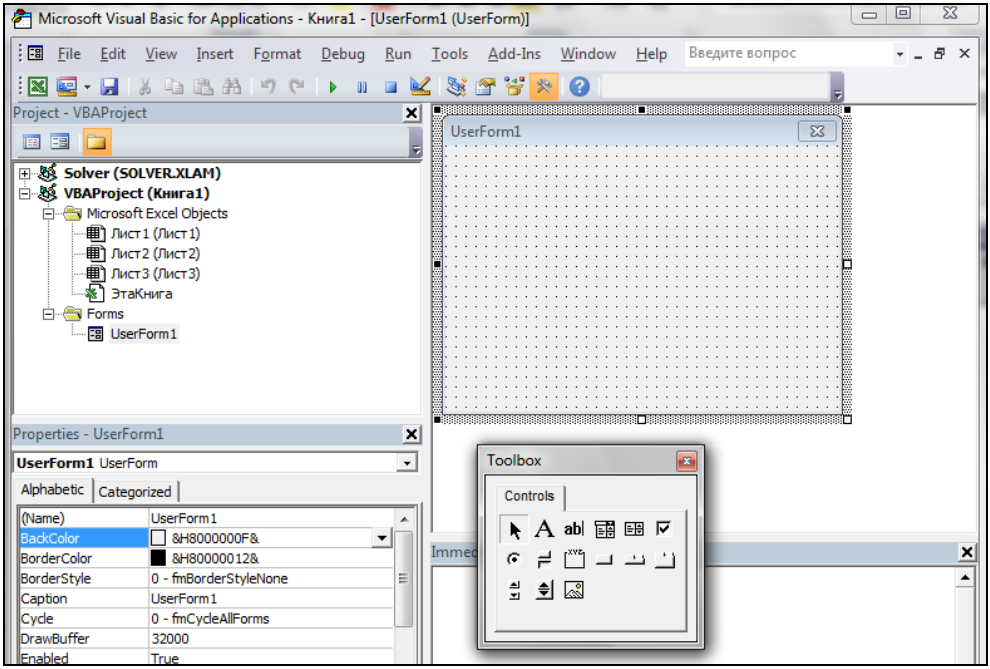


Рис. 4.17. Новая форма в редакторе Visual Basic

Свойства формы

Форма обладает широким спектром свойств, позволяющих контролировать как ее внешний вид, так и параметры функционирования. Конечно, наиболее часто используемыми свойствами формы являются те, которые задают имя формы и текст, отображаемый в заголовке окна. В табл. 4.21 перечислены основные свойства формы.

***Таблица 4.21.** Свойства формы*

Свойство	Описание
`Name`	Имя формы
`ActiveControl`	Возвращает ссылку на элемент управления, получивший фокус
`BackColor`	Цвет фона
`BorderColor`	Цвет границы
`BorderStyle`	Стиль границы. Допустимыми значениями являются следующие константы: `fmBorderStyleNone` и `fmBorderStyleSingle`
`CanPaste`	Устанавливает, возможна ли вставка объекта из буфера обмена
`CanRedo`	Устанавливает, возможна ли операция **Повторить**
`CanUndo`	Устанавливает, возможна ли операция **Отменить**

Таблица 4.21 *(окончание)*

Свойство	Описание
`Caption`	Заголовок формы
`Cycle`	Специфицирует действия элементов, расположенных в объектах-контейнерах **Frame** и **Page**, выполняемые при потере фокуса
`DrawBuffer`	Задает размер памяти, используемой при перерисовке изображения
`Enabled`	Определяет, доступна ли для пользователя форма
`ForeColor`	Цвет заголовка
`Height`, `Width`	Высота и ширина формы
`HelpContextID`	Ссылка на главу справочного файла
`InsideHeight`, `InsideWidth`	Высота и ширина пользовательской части формы, т. е. без строки заголовка и толщины границы
`KeepScrollBarsVisible`	Отображение полос прокрутки. Допустимыми значениями являются следующие константы: `fmScrollBarsNone`, `fmScrollBarsHorizontal`, `fmScrollBarsVertical` и `fmScrollBarsBoth`
`Left`, `Top`	Координаты левого верхнего угла формы
`MouseIcon`	Назначает пользовательский указатель мыши
`MousePointer`	Специфицирует тип указателя мыши
`Picture`	Задает ссылку на файл с растровым изображением, используемым в качестве фона
`PictureAlignment`	Специфицирует выравнивание растрового изображения, используемого в качестве фона
`PictureSizeMode`	Определяет, надо ли изменять масштаб изображения
`ScrollHeight`, `ScrollWidth`	Задают высоту и ширину прокручиваемой области
`ScrollLeft`, `ScrollTop`	Устанавливают координату верхнего левого угла прокручиваемой области
`SpecialEffect`	Определяет внешний вид формы
`StartUpPosition`	Специфицирует начальное местоположение формы
`Tag`	Задает параметр, используемый для идентификации конкретной формы
`VerticalScrollbarSide`	Определяет, на какой стороне формы отображаются полосы прокрутки
`Visible`	Устанавливает видимость формы
`WhatsThisButton`	Специфицирует отображение кнопки с вопросительным знаком
`Zoom`	Указывает, на сколько надо изменить размер отображаемого объекта

Методы формы

Форма имеет множество методов, позволяющих реализовывать широкий спектр операций от ее отображения или скрытия до перерисовки изображения. В табл. 4.22 перечислены методы формы.

Таблица 4.22. *Методы формы*

Метод	Описание
`Copy`	Копирует содержание объекта в буфер обмена
`Cut`	Копирует с удалением содержание объекта в буфер обмена
`Hide`	Скрывает форму без удаления ее из памяти
`Load`	Загружает объект в память без его отображения
`Move`	Перемещает форму
`Paste`	Вставляет содержимое буфера обмена
`PrintForm`	Печатает изображение формы
`RedoAction`	Повторяет последнюю команду **Повторить**
`Repaint`	Обновляет изображение формы
`Scroll`	Прокручивает изображение
`SetDefaultTabOrder`	Устанавливает используемый по умолчанию порядок обхода элементов управления клавишей <Tab>
`Show`	Отображает форму
`UndoAction`	Повторяет последнюю команду **Отменить**
`Unload`	Удаляет объект из памяти
`WhatsThisMode`	Отображает указатель с вопросительным знаком

События формы

Процедуры обработки событий позволяют создавать программы, дающие возможность контролировать весь жизненный цикл формы от ее инициализации до закрытия. В табл. 4.23 перечислены события формы.

Таблица 4.23. *События формы*

Событие	Описание
`Activate`, `Deactivate`	Происходят при активизации и деактивизации формы
`AddControl`	Происходит при добавлении элемента управления
`BeforeDragOver`	Происходит при буксировке данных
`BeforeDropOrPaste`	Происходит перед вставкой данных, производимой буксировкой

Таблица 4.23 *(окончание)*

Событие	Описание
`Click`	Происходит, когда пользователь щелкает мышью на форме
`DblClick`	Происходит, когда пользователь дважды щелкает мышью на форме
`Error`	Происходит, когда форма обнаружила ошибку, но не может передать сообщение
`Initialize`	Происходит при инициализации формы
`Layout`	Происходит при изменении местоположения формы
`KeyDown`, `KeyUp`	Происходят, когда пользователь нажимает и отпускает любую клавишу, а форма имеет фокус
`KeyPress`	Происходит, когда пользователь нажимает любую клавишу, кроме функциональных клавиш, клавиш управлением курсором и служебных клавиш, а форма имеет фокус
`MouseDown`, `MouseUp`	Происходят, когда пользователь нажимает и отпускает любую кнопку мыши
`MouseMove`	Происходит, когда пользователь передвигает указатель мыши над формой
`QueryClose`	Происходит перед закрытием окна
`RemoveControl`	Происходит при удалении элемента управления
`Resize`	Происходит при изменении размеров элемента управления
`Scroll`	Происходит при прокрутке
`Terminate`	Происходит при закрытии формы
`Zoom`	Происходит при изменении масштабирования формы

Отображение и скрытие формы

При работе с формами особое место занимают метод и два оператора, которые управляют процессами начала и завершения работы с формой:

- метод `Show` загружает форму в память и отображает ее;
- оператор `Unload` выгружает форму с экрана и из памяти;
- оператор `End` завершает выполнение кода без генерации событий `Unload` или `Terminate`. Поэтому завершение работы приложения по инструкции `End` игнорирует код, написанный в процедурах, обрабатывающих перечисленные события.

Отображать и скрывать форму можно с помощью методов `Show` и `Hide`.

Первый проект с формой

В качестве первого проекта с формой создадим пользовательскую форму, которая отображается на экране при нажатии кнопки, расположенной на рабочем листе (рис. 4.18, см. также файл *16-Первая форма.xlsm* на компакт-диске).

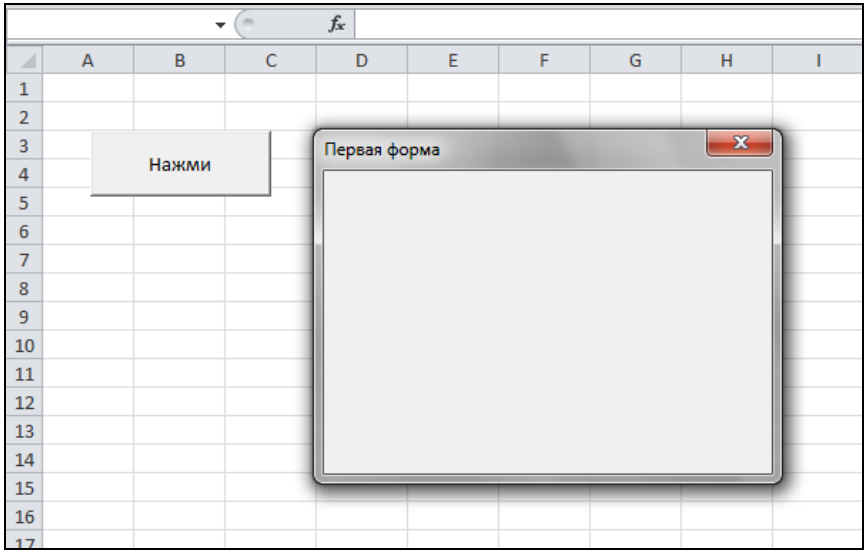


Рис. 4.18. Первый проект с формой

Итак, создайте форму и, используя окно **Properties**, установите ей значения свойств, как показано в табл. 4.24.

Таблица 4.24. *Значения свойств формы, установленные в окне* ***Properties***

Объект	Свойство	Значение
Форма	`Name`	`frmFirst`
	`Caption`	`Первая форма`

Перейдите на рабочий лист, создайте кнопку и, используя окно **Properties**, установите ей значения свойств, как показано в табл. 4.25.

Таблица 4.25. *Значения свойств кнопки, установленные в окне* ***Properties***

Объект	Свойство	Значение
Кнопка	`Name`	`cmdDemoForm`
	`Caption`	`Нажми`

В модуле рабочего листа **Лист1** наберите код (листинг 4.3). Теперь при нажатии кнопки на экране отобразится форма.

Листинг 4.3. Первая форма. Модуль рабочего листа

```
Private Sub cmdDemoForm_Click()
   frmFirst.Show
End Sub
```

Как запустить проект на исполнение?

Форму можно связать с любым элементом управления, размещенным на рабочем листе. Как это делается, было показано в предыдущем примере. В последующих примерах, где основными объектами являются сама форма и функциональные возможности размещенных на ней элементов управления, мы не будем повторять то, как форма интегрируется в рабочий лист, оставляя составление соответствующего кода читателю.

Для проверки работы кода, связанного с формой, на самом деле нет необходимости создавать элементы управления на рабочем листе и с ними связывать форму. Достаточно после конструирования формы и написания кода в модуле формы выбрать команду **Run | Run Sub/UserForm**, либо нажать клавишу <F5>, либо кнопку **Run Macro**  на панели инструментов **Standard**, и форма отобразится поверх активного рабочего листа.

Ключевое слово *Me*

В коде часто используется ключевое слово `Me`, которое возвращает имя активного окна. Например, вместо кода

```
UserForm1.Caption = "Пример"
Unload UserForm1
```

обычно используют следующий код:

```
Me.Caption = "Пример"
Unload Me
```

Форма с обновляемым фоновым рисунком

Рисунок в качестве фона можно внедрить в форму при помощи свойства `Picture`. Это свойство отображает рисунок в его оригинальных размерах. Если же требуется, чтобы рисунок занимал либо всю клиентскую часть формы, либо всю ее ширину или высоту, надо воспользоваться свойством `PictureSizeMode`. Свойство `PictureAlignment` размещает рисунок в клиентской области формы, например, по ее центру или прижатым к специфицированным сторонам формы.

Сконструируем проект формы, в которой в качестве фона отображается рисунок. При щелчках же на форме данный и еще один рисунок будут поочередно заменять друг друга.

Для реализации проекта необходимо иметь два растровых рисунка. В данном случае это D:\1.jpg и D:\2.jpg. Итак, сконструируйте форму и, используя окно **Properties**, установите ей значения свойств, как показано в табл. 4.26.

Таблица 4.26.** Значения свойств, установленные в окне **Properties

Объект	Свойство	Значение
Форма	`Picture`	Ссылка на растровый файл D:\1.jpg
	`PictureSizeMode`	`fmPictureSizeModeStretch`
	`Caption`	`Фоновые сменяемые рисунки`

Выполните двойной щелчок левой кнопкой мыши по форме и в открывшемся модуле формы наберите необходимый код (листинг 4.4, см. также файл *17-Фоновый рисунок.xlsm* на компакт-диске). Теперь при щелчке рисунки D:\1.jpg и D:\2.jpg будут на форме поочередно друг друга заменять. Загрузка изображения из файла в коде производится функцией `LoadPicture`, в качестве значения параметра которой указывается имя исходного файла. Так как значение свойства `PictureSizeMode` для рисунка D:\1.jpg установлено равным `fmPictureSizeModeStretch`, то он растягивается или сжимается с нарушением пропорций с тем, чтобы занять всю клиентскую область формы (рис. 4.19, *а*). Значение свойства `PictureSizeMode` для рисунка D:\2.jpg установлено равным `fmPictureSizeModeZoom`, поэтому он растягивается или сжимается с сохранением пропорций с тем, чтобы занять либо всю ширину, либо высоту клиентской области формы (рис. 4.19, *б*). Свойство `PictureAlignment`, установленное равным `fmPictureAlignmentTopLeft`, приводит к тому, что левые верхние углы рисунка и клиентской области формы совпадают.

а *б*

Рис. 4.19. Форма с обновляемым фоновым рисунком

Листинг 4.4. Форма с обновляемым фоновым рисунком

```
Private Sub UserForm_Click()
   Static flag As Boolean
   Dim filename As String
   If Not flag Then
      filename = "D:\1.jpg"
      Me.Picture = LoadPicture(filename)
      Me.PictureSizeMode = fmPictureSizeModeStretch
      Me.Caption = "Фоновые сменяемые рисунки " & filename
   Else
      filename = "D:\2.jpg"
      Me.Picture = LoadPicture(filename)
      Me.PictureSizeMode = fmPictureSizeModeZoom
```

```
      Me.PictureAlignment = fmPictureAlignmentTopLeft
      Me.Caption = "Фоновые сменяемые рисунки " & filename
   End If
   Me.Repaint
   flag = Not flag
End Sub
```

Примечание

Для загрузки рисунка совсем не обязательно было устанавливать значения свойств `Picture` и `PictureSizeMode` формы. Достаточно было в модуль формы добавить код, на этапе инициализации формы вызывающий процедуру обработки события `Click` формы, в которой и происходит отображение:

```
Private Sub UserForm_Initialize()
   UserForm_Click
End Sub
```

Удаление рисунка

В окне **Properties** картинка удаляется размещением курсора в поле **Picture** и нажатием клавиши <Delete>. В коде это достигается присвоением значения свойства `Picture` равным `LoadPicture("")`. Например:

```
Me.Picture = LoadPicture("")
```

Форма с мозаичным фоном и установкой свойств на этапе инициализации

Изображение в форму можно выводить не только в виде целой картинки, но и как мозаику. В этом случае значение свойства `PictureTiling` надо установить равным `True`. Конечно, следует позаботиться и о свойстве `PictureAlignment`, задающем расположение первоначальной картинки, от которой все мозаичное покрытие и строится (рис. 4.20).

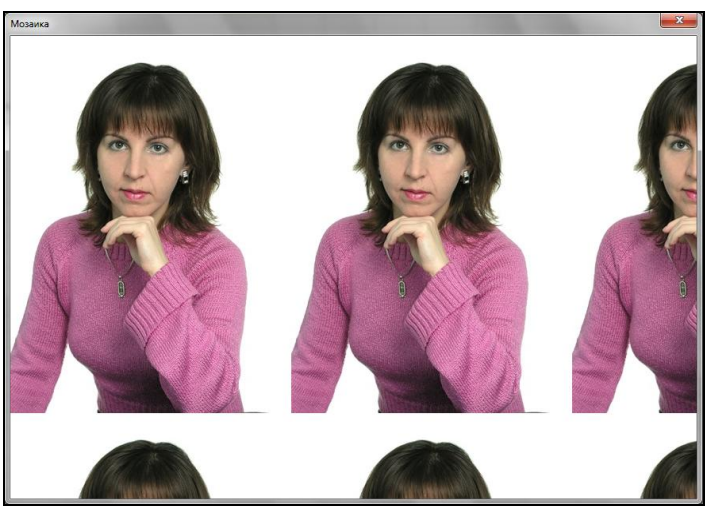


Рис. 4.20. Форма с мозаичным фоном

Значения свойств формы можно устанавливать как при помощи окна **Properties**, так и в коде. В последнем случае это, как правило, делается в процедуре обработки события `Initialize` формы, которое генерируется при инициализации формы, но до ее отображения на экране.

В качестве примера построим форму с мозаичным фоном и заданием ее свойств в коде на этапе инициализации формы. Итак, создайте форму и в модуле формы наберите код (см. файл *18-Мозаика.xlsm* на компакт-диске). Кроме того, убедитесь, что в каталоге, который используется MS Excel по умолчанию, находится необходимый графический файл, который вы хотите отобразить в виде мозаичного фона.

> ***Примечание***
>
> Чтобы узнать, какой каталог задан у вас по умолчанию, перейдите на вкладку **Файл** ленты и нажмите кнопку **Параметры**. В открывшемся окне **Параметры Excel** выберите слева категорию **Сохранение**, а справа в группе **Сохранение книг** в поле **Расположение файлов по умолчанию** будет отображен текущий каталог. При необходимости его можно изменить.

Данное приложение проверяет наличие файла с растровым изображением в данном каталоге при помощи функции `Dir()`. Если его нет в рабочем каталоге, то при запуске приложения форма будет отображаться без мозаичного фона. Функция `Dir()` возвращает имя каталога или файла, соответствующего образцу, указанному в виде аргумента этой функции. В образце можно использовать символы `*` и `?`. Если подходящего каталога или файла нет, то функция `Dir()` возвращает пустую строку. Поэтому проверка наличия файла просто сводится к проверке длины возвращаемой функцией `Dir()` строки. Если она имеет нулевую длину, то файла нет.

Закрытие формы при нажатии клавиши <Esc>

Нажатие кнопки **Закрыть**, расположенной в правом верхнем углу формы, приводит к ее закрытию. Возникает естественный вопрос: "Возможно ли закрытие формы при нажатии каких-либо клавиш, например <Esc>?" Ответ, конечно, положителен. Для этого надо всего лишь написать код, обрабатывающий событие `KeyDown`, явно идентифицировать требуемую клавишу и при помощи оператора `Unload` или `End` закрыть форму. У процедуры обработки события `KeyDown` имеются два параметра. Первый из них возвращает код нажатой клавиши, а второй идентифицирует нажатую клавишу-модификатор. Для кода клавиши <Esc> в VBA имеется специальная константа `vbKeyEscape`. Следующий код (листинг 4.5, см. также файл *19-Закрытие на ESC.xlsm* на компакт-диске) из модуля формы закрывает окно при нажатии клавиши <Esc>.

Листинг 4.5. Закрытие формы при нажатии клавиши <Esc>

```
Private Sub UserForm_KeyDown(ByVal KeyCode As MSForms.ReturnInteger, _
                             ByVal Shift As Integer)
   If KeyCode = vbKeyEscape Then
      Unload Me
   End If
End Sub
```

Подтверждение закрытия окна

В проектах часто возникает ситуация, когда перед закрытием окна требуется получить подтверждение со стороны пользователя. Подобного эффекта можно избежать, воспользовавшись процедурой обработки события `QueryClose`, которое генерируется непосредственно перед закрытием окна. У этой процедуры имеются два параметра. Если первый из них принимает значение –1, то закрытие не происходит, если же 0, то окно закрывается. Второй параметр идентифицирует причину, вызвавшую закрытие окна. Например, в следующем коде (листинг 4.6, см. также файл *20-Подтверждение на закрытие.xlsm* на компакт-диске) при закрытии пользовательского окна появляется диалоговое окно с двумя кнопками: **Да** и **Нет**, требующее подтверждения со стороны пользователя. Если он нажмет кнопку **Да**, то окно закрывается, а если **Нет** — то не закрывается.

Листинг 4.6. Подтверждение закрытия окна

```
Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
   Select Case MsgBox("Закрыть окно?", vbYesNo + vbQuestion)
      Case vbYes  :   Cancel = 0
      Case vbNo   :   Cancel = -1
   End Select
End Sub
```

Задание местоположения формы

Первоначальное местоположение формы устанавливается свойством `StartUpPosition`. Допустимые его значения перечислены в табл. 4.27.

***Таблица 4.27.** Значения свойства `StartUpPosition`*

Значение	Описание
0	Координата левого верхнего угла формы устанавливается при помощи свойств `Top` и `Left`
1	По центру окна
2	Центрируется по окну экрана
3	В верхнем левом углу экрана

Например, следующий код (листинг 4.7) отображает форму так, чтобы ее левый верхний угол располагался в точке (100, 100).

Листинг 4.7. Задание местоположения формы

```
Private Sub UserForm_Initialize()
    Me.StartUpPosition = 0
    Me.Top = 100
    Me.Left = 100
End Sub
```

Модальная форма

Мономодальное окно представляет собой окно, не закрыв которое нельзя отобразить на экране другое окно. По умолчанию пользовательское окно в VBA является мономодальным. Задать тип окна (мономодальный или полимодальный) можно при помощи необязательного параметра *style* метода `Show`.

```
Show style
```

Здесь параметр *style* имеет следующие допустимые значения: `vbModal` или `1` — мономодальная форма, `vbModeless` или `0` — полимодальная форма.

Например, с помощью следующей инструкции окно формы отображается на рабочем листе в полимодальном режиме, и поэтому при открытом окне пользователь имеет доступ к ячейкам рабочего листа.

```
UserForm1.Show vbModeless
```

При запуске же окна следующей инструкцией оно находится в мономодальном режиме, и поэтому ячейки рабочего листа недоступны для пользователя до тех пор, пока он не закроет окно.

```
UserForm1.Show vbModal
```

Использование нескольких форм

В проекте может быть несколько форм. При замене одной формы другой надо учитывать, в каком режиме (мономодальном или полимодальном) она открыта. Например, добавьте в проект две формы `UserForm1` и `UserForm2`. На рабочем листе создайте кнопку. Установите значение ее свойства `Name` равным `cmdForm1`. При нажатии этой кнопки на экране будет отображаться окно первой формы.

Листинги 4.8 и 4.9 демонстрируют, как должны различаться коды для вызова второй формы при щелчке на первой, в зависимости от того, какой тип окна задан: мономодальный или полимодальный.

В мономодальном режиме, прежде чем отображать вторую форму, в коде приходится закрыть первую. При этом на экране всегда отображается только одна форма — либо первая, либо вторая.

В полимодальном режиме нет необходимости закрывать первую форму, и после щелчка на первой форме на экране отображаются обе формы. Для того чтобы обе формы были видны одновременно, вторую форму выводят с небольшим сдвигом относительно первой.

Листинг 4.8, *а*. Мономодальный режим. Код из модуля рабочего листа

```
Private Sub cmdForm1_Click()
   UserForm1.Show vbModal
End Sub
```

Листинг 4.8, *б*. Мономодальный режим. Код из модуля `UserForm1`

```
Private Sub UserForm_Click()
   Unload UserForm1
   UserForm2.Show
End Sub
```

Листинг 4.9, *а*. Полимодальный режим. Код из модуля рабочего листа

```
Private Sub cmdForm1_Click()
   UserForm1.Show vbModeless
End Sub
```

Листинг 4.9, *б*. Полимодальный режим. Код из модуля `UserForm1`

```
Private Sub UserForm_Click()
   UserForm2.StartUpPosition = 0
   UserForm2.Top = UserForm1.Top + 20
   UserForm2.Left = UserForm1.Left + 20
   UserForm2.Show
End Sub
```

"Пасхальное яйцо"

"Пасхальное яйцо" (easter egg) представляет собой скрытое диалоговое окно в приложении и является прораммистской шуткой, которая наиболее часто используется в компьютерных играх. В приводимом далее примере (см. файл *21-Пасхальное яйцо.xlsm* на компакт-диске) "пасхальное яйцо" отображается на экране только в случае, если пользователь щелкнет правой кнопкой мыши в области формы, лежащей в ее правом нижнем углу и занимающей одну девятую часть пользовательской области. Естественно, что до "пасхального яйца" сможет докопаться только тот, кто создавал приложение (рис. 4.21).

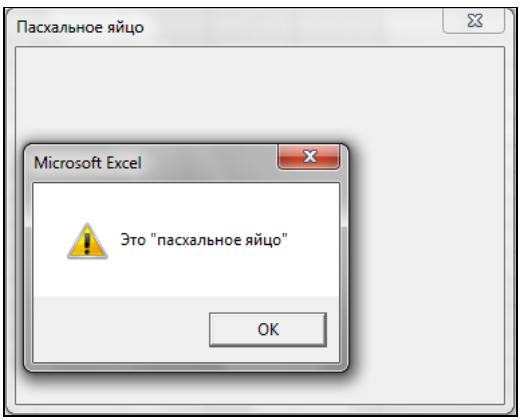


Рис. 4.21. "Пасхальное яйцо"

Для идентификации точки щелчка воспользуемся процедурой обработки события `MouseDown` (нажатие кнопки мыши) формы, которая имеет следующий синтаксис:

```
Private Sub object_MouseDown(ByVal Button As Long, ByVal Shift As Long, _
                             ByVal X As Long, ByVal Y As Long)
```

- *`Button`* — идентифицирует нажатую кнопку мыши. Допустимыми значениями могут быть следующие константы `XlMouseButton`: `xlNoButton`, `xlPrimaryButton`, `xlSecondaryButton` и `xlMiddleButton`.
- *`Shift`* — определяет нажатую клавишу-модификатор (<Shift>, <Ctrl> и <Alt>). Возвращает 0, если ни одна клавиша-модификатор не была нажата; возвращает 1, если была нажата клавиша <Shift>; возвращает 2, если была нажата клавиша <Ctrl>; возвращает 4, если была нажата клавиша <Alt>. Если была нажата комбинация клавиш, то возвращается сумма чисел-идентификаторов этих клавиш.
- *`X`*, *`Y`* — координаты точки, в которой была нажата кнопка мыши.

Элементы управления

В VBA имеется обширный набор встроенных элементов управления, которые можно использовать в форме. Применяя их, нетрудно создать любой пользовательский интерфейс, который будет удовлетворять всем требованиям, предъявляемым к интерфейсу в среде Windows. Элементы управления создаются при помощи панели элементов **Toolbox** (рис. 4.22), которая отображается на экране, либо выбором команды **View | Toolbox**, либо нажатием кнопки  панели инструментов **Standard**. На панели **Toolbox** представлены кнопки, при помощи которых конструируются элементы управления. Все кнопки панели элементов, за исключением кнопки **Select Objects** (Выбор объекта)  , служат для создания элементов управления. Щелкнув на кнопке **Select Objects** (Выбор объекта), можно выбрать уже созданный в форме элемент управления для последующего его редактирования (изменения размеров или перемещения).

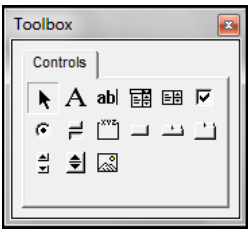


Рис. 4.22. Панель элементов **Toolbox**

Примечание

Панель инструментов **Элементы управления**, которая отображается MS Excel для создания элементов управления на рабочем листе, имеет меньший набор объектов, чем панель элементов из редактора Visual Basic. В ней нет рамок, набора вкладок и страниц.

Создание элементов управления на рабочем листе или форме, как правило, происходит на начальном этапе конструирования приложения. Иногда используется программное их создание в процессе работы приложения. Но этот подход применяется реже. В табл. 4.28 приведен список основных элементов управления и соответствующих кнопок панели элементов **Toolbox.**

Таблица 4.28. *Элементы управления из панели элементов* ***Toolbox***

Элемент управления	**Кнопка, его создающая**
TextBox (Поле)	
Label (Надпись)	
CommandButton (Кнопка)	
ListBox (Список)	
ComboBox (Поле со списком)	
ScrollBar (Полоса прокрутки)	
SpinButton (Счетчик)	
OptionButton (Переключатель)	
CheckBox (Флажок)	

Таблица 4.28 *(окончание)*

Элемент управления	Кнопка, его создающая
ToggleButton (Выключатель)	
Frame (Рамка)	
Image (Рисунок)	
MultiPage (Набор страниц)	
TabStrip (Набор вкладок)	

Размещение элемента управления на форме

Для размещения элемента управления на форме нажмите соответствующую кнопку панели инструментов **Toolbox** и с помощью мыши перетащите рамку элемента управления в нужное место. После этого элемент управления можно перемещать, изменять его размеры, копировать в буфер обмена, вставлять из буфера обмена и удалять из формы.

Label (Надпись)

Элемент управления **Label** (Надпись)  используется для отображения надписей, например заголовков элементов управления, не имеющих свойства `Caption`. Пользователь не может изменять текст, отображаемый в надписи во время выполнения программы. Основным свойством надписи является свойство `Caption`, устанавливающее отображаемый в ней текст.

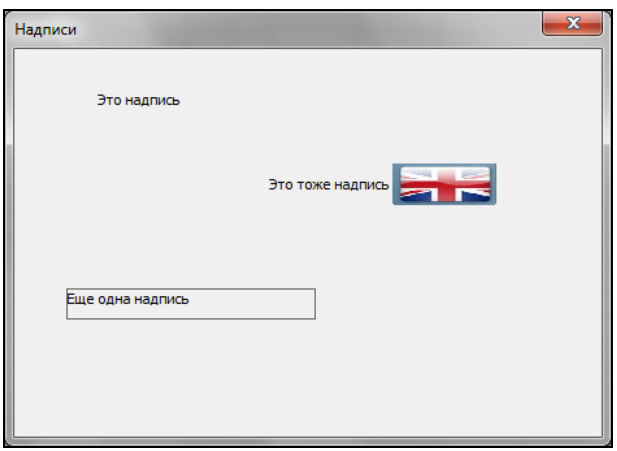


Рис. 4.23. Примеры надписей

Следующий пример демонстрирует различные типы надписей: простую, с рисунком, с рамкой и переносом слов (рис. 4.23). Для реализации этого проекта создайте форму, в которой расположите три надписи. Кроме того, необходим файл

с рисунком. В данном случае это D:\flags.jpg с изображением английского флага. В модуле формы наберите код (см. файл *22-Надписи.xlsm* на компакт-диске), который на этапе инициализации формы и задаст ее параметры. Рисунок в надпись загружается свойством `Picture`. Свойство `PicturePosition` определяет взаимное расположение рисунка и текста. Свойство `BorderStyle` устанавливает, отображается надпись с границей или без нее. Свойство же `WordWrap` задает автоматический перенос слов, если они не умещаются в одну строку.

TextBox (Поле)

Элемент управления **TextBox** (Поле)  , или поле ввода, в основном применяется для ввода пользователем текста, который в последующем используется в программе, или для вывода в него результатов расчетов в программе. Текст, введенный в поле, в коде может быть преобразован либо в числа, либо в формулы. Основным событием, связанным с полем ввода, является событие `Change`. В табл. 4.29 перечислены основные свойства поля ввода.

Таблица 4.29. *Основные свойства поля ввода*

Свойство	Описание
`Text`	Возвращает текст, содержащийся в поле
`Multiline`	Параметр, принимающий логические значения, который устанавливает многострочный режим ввода текста в поле
`ScrollBars`	Устанавливает режим отображения в поле полос прокрутки. Допустимые значения: `fmScrollBarsNone` (не выводить полос прокрутки), `fmScrollBarsHorizontal` (выводить горизонтальную полосу прокрутки), `fmScrollBarsVertical` (выводить вертикальную полосу прокрутки), `fmScrollBarsBoth` (выводить горизонтальную и вертикальную полосы прокрутки)
`SelLenght`, `SelStart`, `SelText`	Эти свойства характеризуют выделенный в поле фрагмент текста (длина, начало и сам фрагмент текста)
`MaxLength`	Устанавливает максимально допустимое количество вводимых в поле символов. Если это свойство равно 0, то нет ограничений на вводимое количество символов
`PasswordChar`	Устанавливает символ, отображаемый при вводе пароля. Если это свойство определено, то вместо вводимых символов в поле будет отображаться установленный символ

Сложение двух чисел

В качестве примера работы с полями ввода создадим демонстрационный проект, в котором по введенным в поля двум числам находится их сумма, а результат отображается в третьем поле. Итак, создайте форму, на которой расположите три надписи, три поля ввода и две кнопки (рис. 4.24) и, используя окно **Properties**, установите им значения свойств, как показано в табл. 4.30.

Таблица 4.30. *Значения свойств, установленные в окне* ***Properties***

Объект	Свойство	Значение
Форма	`Caption`	`c=a+b`
Надпись	`Caption`	`a`
Поле ввода	`Name`	`txtA`
Надпись	`Caption`	`b`
Поле ввода	`Name`	`txtB`
Надпись	`Caption`	`c`
Поле ввода	`Name`	`txtC`
Кнопка	`Name`	`cmdOK`
	`Caption`	`OK`
Кнопка	`Name`	`cmdCancel`
	`Caption`	`Cancel`

В модуле формы наберите код (см. файл *23-Поле-Сложение двух чисел.xlsm* на компакт-диске). Он обеспечит нахождение суммы двух чисел, введенных в поля **a** и **b**, и вывод результата в поле **c** при нажатии кнопки **OK**. Нажатие же кнопки **Cancel** приведет к закрытию окна.

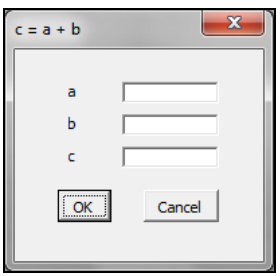


Рис. 4.24. Сложение двух чисел

Кнопка с "горячей" клавишей

Свойство `Accelerator` элемента управления назначает буквенную или цифровую клавишу, при нажатии которой одновременно с клавишей <Alt> выполняется процедура обработки события — щелчок на элементе управления. При этом буква или цифра на этой клавише должна входить в строку, задающую значение свойства `Caption` элемента управления, и эта буква или цифра будет отображаться подчеркнутой. Например, если в код из предыдущего раздела добавить следующую процедуру инициализации формы (листинг 4.10, см. также файл *24-Поле-Сложение двух чисел-Перемещение фокуса между полями.xlsm* на компакт-диске), то кнопки **OK** и **Cancel** превратятся в **OK** и **Cancel**, а нажатие комбинаций клавиш <Alt>+<O> и <Alt>+<C> будет равносильно нажатию этих кнопок.

Листинг 4.10. Кнопка с "горячей" клавишей

```
Private Sub UserForm_Initialize()
   cmdOK.Accelerator = "O"
   cmdCancel.Accelerator = "C"
End Sub
```

Клавиши <Enter> и <Esc>

Свойство `Default` при значении, равном `True`, назначает кнопку, для которой генерируется событие `Click`, если пользователь нажмет клавишу <Enter>. Свойство же `Cancel` при значении, равном `True`, назначает кнопку, для которой генерируется событие `Click`, если пользователь нажмет клавишу <Esc>. Так, в следующем листинге 4.11 для проекта из *разд. "Сложение двух чисел" ранее в этой главе* клавишам <Enter> и <Esc> назначаются функции кнопок **OK** и **Cancel**, т. е. при нажатии клавиши <Enter> будет найдена искомая сумма, а клавиши <Esc> — закрыто окно.

Листинг 4.11. Кнопки с ассициированными клавишами <Enter> и <Esc>

```
Private Sub UserForm_Initialize()
   cmdOK.Default = True
   cmdCancel.Cancel = True
End Sub
```

Суммирование с блокировкой результата для пользователя

Свойство `Enabled` при значении `False` полностью блокирует для пользователя элемент управления, так что элемент управления даже не может получать фокус. Свойство `Lock` при значении `True` запрещает пользователю производить любые изменения в поле ввода, но, тем не менее, поле ввода может получать фокус, так что пользователь теперь имеет возможность копировать содержимое поля ввода в буфер обмена. Следующий код (листинг 4.12) полностью блокирует для пользователя поле **c**, в которое выводится найденная сумма в проекте из *разд. "Сложение двух чисел" ранее в этой главе*.

Листинг 4.12. Суммирование с блокировкой результата для пользователя

```
Private Sub UserForm_Initialize()
   txtC.Enabled = False
End Sub
```

Как сделать, чтобы при нажатии кнопки она не получала фокус?

При нажатии кнопки она получает фокус, и поэтому для повторного ввода значения в поле ввода оно должно сначала этот фокус получить. Можно сделать так, чтобы при нажатии кнопки она не получала фокус, т. е. фокус оставался бы на том же элементе, который имел его до нажатия. Свойство `TakeFocusOnClick` при значении, равном `False`, блокирует передачу фокуса. Например, для проекта из *разд. "Сложение двух чисел" ранее в этой главе* такую блокировку можно сделать процедурой из листинга 4.13.

Листинг 4.13. Кнопка не получает фокус даже при ее нажатии

```
Private Sub UserForm_Initialize()
   cmdOK.TakeFocusOnClick = False
End Sub
```

Перемещение фокуса между полями при нажатии клавиши <Enter>

Для перемещения фокуса между полями при нажатии клавиши <Enter> у полей ввода надо обработать событие `KeyDown` для того, чтобы идентифицировать нажатую клавишу, и если таковой является клавиша <Enter>, то методом `SetFocus` следует установить фокус на требуемом поле ввода. В следующем примере (листинг 4.14) для проекта из *разд. "Сложение двух чисел" (см. ранее в этой главе)* при нажатии клавиши <Enter>, когда фокус имеет поле **a**, фокус перемещается на поле **b**. При нажатии же клавиши <Enter>, когда фокус имеет поле **b**, фокус перемещается на поле **a**, и, кроме того, выполняется процедура обработки события `Click` кнопки **OK**, т. е. находится искомая сумма. Поле **c** заблокировано для пользователя, но оно может получать фокус.

Листинг 4.14. Перемещение фокуса между полями при нажатии клавиши <Enter>

```
Private Sub txtA_KeyDown(ByVal KeyCode As MSForms.ReturnInteger, _
                         ByVal Shift As Integer)
   If KeyCode = vbKeyReturn Then
       txtB.SetFocus
   End If
End Sub
Private Sub txtB_KeyDown(ByVal KeyCode As MSForms.ReturnInteger, _
                         ByVal Shift As Integer)
   If KeyCode = vbKeyReturn Then
      cmdOK_Click
      txtA.SetFocus
   End If
End Sub
Private Sub UserForm_Initialize()
   txtC.Locked = True
End Sub
```

Всплывающая подсказка

Элементам управления можно устанавливать всплывающую подсказку при помощи свойства `ControlTipText`, которая появляется в виде окна с пояснительным текстом, отображаемым рядом с элементом управления, на который нацелен указатель мыши. В следующем примере (см. файл *25-Подсказки.xlsm* на компакт-диске) для проекта из *разд. "Сложение двух чисел" ранее в этой главе* трем полям ввода и двум кнопкам задаются всплывающие подсказки (рис. 4.25).

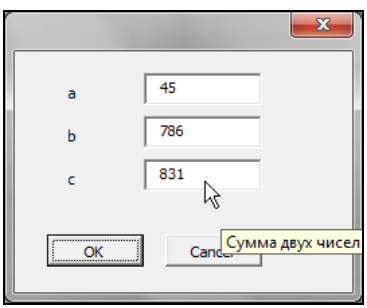


Рис. 4.25. Всплывающая подсказка

Поле ввода пароля

Элемент управления **TextBox** позволяет создавать поле ввода пароля (password). Символы, которые отображаются в нем как набираемые, называются *эхо-символами* и устанавливаются свойством `PasswordChar`.

Рассмотрим пример простого приложения ввода пароля. Имеется поле ввода пароля и кнопка. Кнопка при инициализации окна заблокирована. Если пользователь введет в поле ввода правильный пароль (в данном случае `laru`), то кнопка разблокируется, если же неправильный, то она повторно блокируется (рис. 4.26).

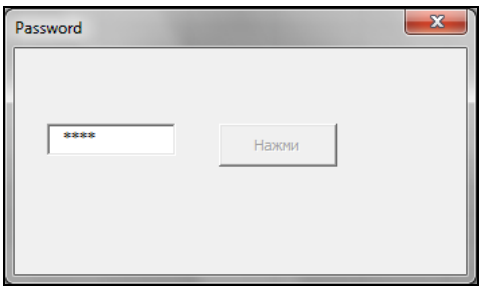


Рис. 4.26. Окно **Password**

Итак, создайте форму, на которой расположите поле ввода и кнопку. Используя окно **Properties**, установите им значения свойств, как показано в табл. 4.31.

Таблица 4.31. *Значения свойств, установленные в окне* ***Properties***

Объект	Свойство	Значение
Форма	`Caption`	`Password`
Поле ввода	`Name`	`txtPass`
Кнопка	`Name`	`cmdMsg`
	`Caption`	`Нажми`

В модуле формы наберите код (см. файл *26-Пароль.xlsm* на компакт-диске). Для упрощения ввода пароля в нем не учитывается различие между регистрами букв. Это обеспечивается переводом всех введенных букв в нижний регистр функцией `LCase()`.

Многострочное поле ввода

Поле ввода можно использовать как многострочное. Для этого надо установить значение свойства `MultiLine` равным `True`. В качестве примера сконструируем простой проект, который производит пересчет рублей в доллары. В этом проекте имеются два поля ввода: простое и многострочное. В простое поле вводится исходная сумма, которая при нажатии клавиши <Enter> переводится в доллары, а результат отображается в многострочном поле (рис. 4.27).

Итак, создайте форму, на которой расположите два поля ввода и, используя окно **Properties**, установите им значения свойств, как показано в табл. 4.32.

Таблица 4.32.** Значения свойств, установленные в окне **Properties

Объект	Свойство	Значение
Поле ввода	Name	txtMoney
Поле ввода	Name	txtResult

В модуле формы наберите код (см. файл *27-Многострочность-Пересчет валюты.xlsm* на компакт-диске). С тем, чтобы пользователь не мог подкорректировать результаты расчетов, многострочное поле для него в проекте блокируется на этапе инициализации формы.

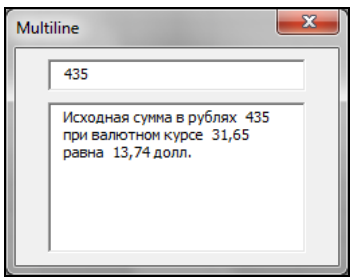


Рис. 4.27. Многострочное поле ввода

Обмен значениями между формами

Если в проекте присутствует несколько форм, то они могут обмениваться значениями посредством открытых переменных, причем переменные должны быть объявлены в стандартном модуле. Пример обмена значениями дается в листинге 4.15. Создайте две формы. В первой из них разместите два поля ввода и кнопку. Во второй — поле ввода. При нажатии кнопки в первой форме считываются значения из ее полей, затем они складываются, первая форма закрывается, вторая отображается, а найденная сумма выводится в ее поле ввода.

Листинг 4.15, *а*. Модуль первой формы

```
Private Sub CommandButton1_Click()
   res = CDbl(TextBox1.Text) + CDbl(TextBox2.Text)
   Unload Me
   UserForm2.Show
End Sub
```

Листинг 4.15, *б*. Модуль второй формы

```
Private Sub UserForm_Initialize()
   TextBox1.Text = res
End Sub
```

Листинг 4.15, *в*. Стандартный модуль

```
Public res As Double
```

Таймер как пример класса, генерирующего события

В качестве еще одного примера создадим проект, который включает класс `TimerState`, моделирующий таймер. У этого класса есть два открытых поля — `Duration` и `Enabled`, устанавливающие продолжительность работы таймера и его возможность выполнять задание. Кроме того, в классе имеется метод `TimerTask`, который запускает таймер. По завершении каждой 1/100 доли секунды в таймере генерируется событие `Tick`, а по завершении задания — событие `Collapse`.

Для испытания этого класса создайте форму, в которой расположите два поля ввода и две кнопки (**Start** и **Stop**), а в модуле формы наберите требуемый код (см. модуль класса `TimerState` и модуль формы в файле *28-Пример таймера.xlsm* на компакт-диске). Нажатие кнопки **Start** обеспечит включение таймера на 5 с. В первое поле ввода будет выводиться число прошедших с момента включения 1/100 долей секунды. Во второе поле ввода — сообщение о том, что либо таймер еще работает, либо прекратил работу, если такое событие случилось. Нажатие кнопки **Stop** вызовет остановку работы таймера. Кроме того, добавьте на рабочий лист кнопку, нажатие которой будет открывать окно (созданную форму) с таймером.

CheckBox (Флажок) и *ToggleButton* (Выключатель)

Флажок ☑ и выключатель ⇄ предоставляют пользователю возможность выбора. Основным свойством этих элементов управления является свойство `Value`, возвращающее их состояние. Эти элементы управления обычно имеют два состояния: установлен (значение свойства `Value` равно `True`) и сброшен (значение свойства `Value` равно `False`), но могут настраиваться на выбор из трех альтернатив при помощи свойства `TripleState`.

Управление видимостью элементов управления

Свойство `Visible` контролирует видимость элемента управления. Если его значение равно `True`, то он видим, а если `False`, то невидим. Следующий пример демонстрирует, как с помощью флажка можно контролировать видимость элемента управления (в данном случае поля ввода). Итак, создайте форму и в модуле кода формы наберите код (см. файл *29-Флажок.xlsm* на компакт-диске). Если флажок **Показать** установлен, то поле ввода отображается, а если снят, то оно скрыто (рис. 4.28).

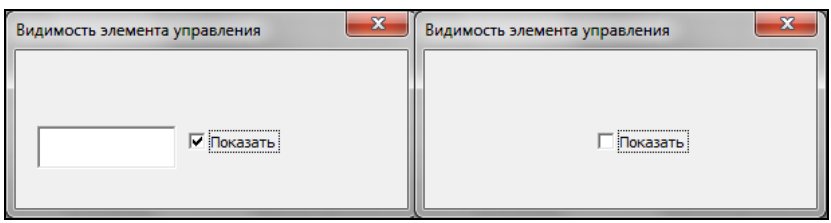


Рис. 4.28. Управление видимостью элементов управления

Управление доступностью для пользователя элементов управления

Свойство `Enabled` управляет доступностью для пользователя элементов управления. Если значение свойства равно `True`, то элемент управления может получить фокус и быть доступным для пользователя, а если `False`, то не может. Следующий пример демонстрирует, как с помощью флажка можно контролировать доступность элемента управления (в данном случае, кнопки). Итак, создайте форму и в модуле кода наберите необходимый код (см. файл *30-Достижимость элемента управления.xlsm* на компакт-диске). Если флажок **Блокировка** установлен, то кнопка **Нажми** заблокирована, а если снят, то доступна (рис. 4.29).

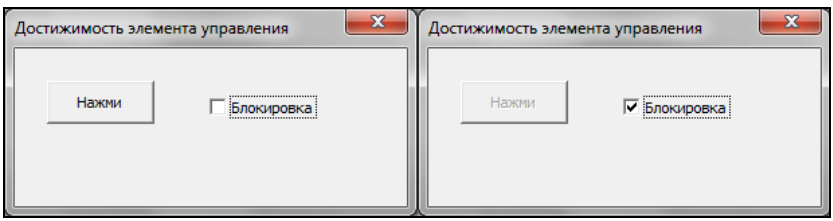


Рис. 4.29. Управление доступностью для пользователя элементов управления

Frame (Рамка)

Элемент управления **Frame** (Рамка) используется для визуальной группировки элементов управления. Основным свойством рамки является `Capture`, задающее надпись при рамке. Флажки и переключатели также в коде можно группировать при помощи свойства `GroupName`.

OptionButton (Переключатель)

Элемент управления **OptionButton** (Переключатель) позволяет выбрать одну из нескольких взаимоисключающих альтернатив. Переключатели обычно отображаются группами по выбираемым альтернативам. Группировка производится при помощи элемента управления **Frame** (Рамка) или свойства `GroupName` объекта `OptionButton`. Основными событиями переключателя являются события `Click` и `Change`, а основным свойством — свойство `Value`, возвращающее или устанавливающее его состояние. Если значение этого свойства равно `True`, то переключатель установлен, а если `False` — то сброшен.

Переключатель и выбор результирующей операции

В качестве примера использования переключателей слегка видоизменим проект из *разд. "Сложение двух чисел" ранее в этой главе*, а именно теперь будем находить не сумму, а результат по выбранной операции: сложения или вычитания. Исполняемая операция устанавливается за счет выбора соответствующего переключателя (рис. 4.30).

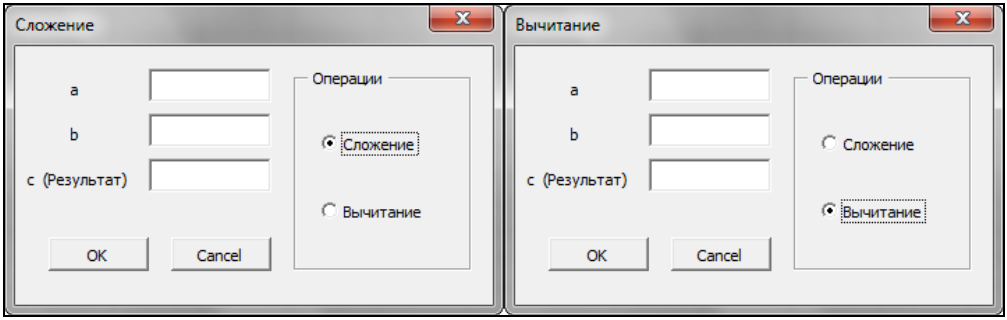


Рис. 4.30. Выбор результирующей операции

Итак, создайте форму, на которой расположите три надписи, три поля ввода и кнопку, а также рамку, в которой как в контейнере разместите два переключателя, и, используя окно **Properties**, установите им значения свойств, как показано в табл. 4.33.

Таблица 4.33. *Значения свойств, установленные в окне* ***Properties***

Объект	**Свойство**	**Значение**
Надпись	`Caption`	`a`
Поле ввода	`Name`	`txtA`
Надпись	`Caption`	`b`
Поле ввода	`Name`	`txtB`
Надпись	`Caption`	`c`
Поле ввода	`Name`	`txtC`
Кнопка	`Name`	`cmdOK`
	`Caption`	`OK`
Кнопка	`Name`	`cmdCancel`
	`Caption`	`Cancel`
Рамка	`Caption`	`Операции`
Переключатель	`Name`	`optAdd`
	`Caption`	`Сложение`
Переключатель	`Name`	`optSub`
	`Caption`	`Вычитание`

В модуле формы наберите код (см. файл *31-Переключатель-Выбор результирующей операции.xlsm* на компакт-диске). В процедуре обработки события `Click` кнопки происходит идентификация установленного переключателя, и на основе этого производится расчет по соответствующей формуле. Процедуры обработки события `Click` переключателя выводят название выбранной операции в заголовок формы.

ScrollBar (Полоса прокрутки) и *SpinButton* (Счетчик)

Элемент управления **ScrollBar** (Полоса прокрутки) применяется для установки числового значения, причем этот элемент может устанавливать только целые неотрицательные значения. Основным событием элемента управления **ScrollBar** (Полоса прокрутки) является `Change`, а основными свойствами — свойства `Value`, `Min` и `Max`, устанавливающие соответственно текущее, минимальное и максимальное значения. Элемент управления **SpinButton** (Счетчик) по своим функциональным возможностям аналогичен полосе прокрутки, но у него нет ползунка.

Синхронизированная работа поля ввода и счетчика

Счетчик позволяет установить целое значение, которое можно затем вывести в поле ввода. Как сделать работу поля ввода и счетчика синхронизированными, чтобы текущее значение счетчика выводилось в поле, а число, введенное в поле ввода, становилось значением счетчика (рис. 4.31)? В этом случае, как у счетчика, так и у поля надо согласованно обработать события `Change`.

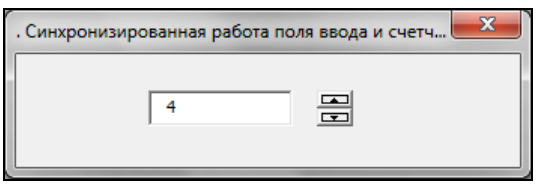


Рис. 4.31. Синхронизированная работа поля ввода и счетчика

Итак, создайте форму, в которой расположите поле и счетчик. В модуле формы наберите необходимый код (см. файл *32-Синхронизированная работа поля ввода и счетчика.xlsm* на компакт-диске). При считывании данных из поля, прежде чем их присваивать значению `Value` счетчика, надо убедиться, являются ли эти данные числом, что можно сделать при помощи функции `IsNumeric()`. Кроме того, необходима проверка, принадлежит ли число интервалу допустимых значений для счетчика, т. е. лежит ли оно в интервале, определяемом свойствами `Min` и `Max` счетчика. В данном примере значения этих свойств устанавливаются равными 1 и 5 при инициализации формы.

ListBox (Список)

Элемент управления **ListBox** (Список) применяется для хранения списка значений. В списке пользователь может выбрать одно или несколько значений, которые в последующем используются в тексте программы. Обратите внимание, что на этапе конструирования визуально список похож на поле ввода. Обычно выбор элемента из списка производится щелчком на элементе. Двойной же щелчок на элементе применяется для выполнения каких-то действий в программе, связанных с этим элементом.

Поэлементное заполнение списка

Заполнение списка поэлементно осуществляется методом `AddItem`. Для примера, создадим простой проект (см. файл *33-Поэлементное заполнение списка.xlsm* на компакт-диске), в котором имеется форма со списком. В список выведены названия городов. Выбор элемента из списка ни к чему не приводит, двойной же щелчок на элементе списка выводит его в очередную свободную ячейку первого столбца активного рабочего листа (рис. 4.32). Выбранное значение из списка возвращается свойством `Text`. Число же заполненных ячеек столбца возвращается свойством `CountA` объекта `WorksheetFunction`.

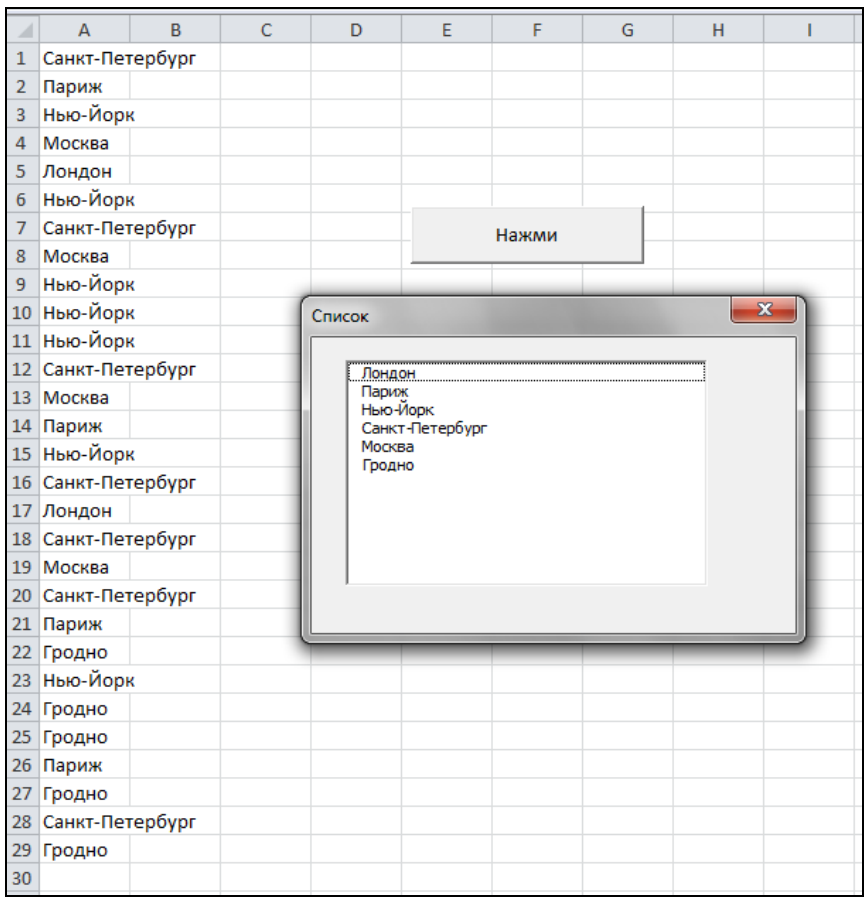


Рис. 4.32. Поэлементное заполнение списка и вывод значений на рабочий лист

Заполнение списка из массива и выбор операции

Заполнять список можно поэлементно, а можно и за одну операцию. Для этого надо свойству `List` списка присвоить значение переменной, содержащей в себе массив значений. В качестве примера еще раз вернемся к проекту из

разд. "Сложение двух чисел" ранее в той главе. В этот раз будем находить не сумму двух чисел, а совершать над ними одну из четырех арифметических операций, перечисленных в списке (рис. 4.33).

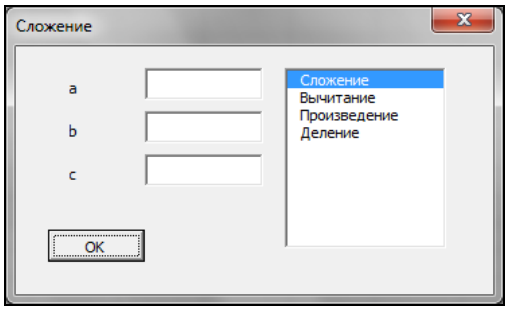


Рис. 4.33. Заполнение списка из массива и выбор операции

Итак, создайте форму, на которой расположите три надписи, три поля ввода, кнопку и список. У всех элементов управления, кроме списка, установите значения свойств при помощи окна **Properties**, как было описано в *разд. "Сложение двух чисел" ранее в этой главе*. У списка установите значения свойства `Name` равным `lstOp`. В модуле формы наберите требуемый код (см. файл *34-Заполнение списка из массива и выбор операции.xlsm* на компакт-диске). Элемент в списке идентифицируется по индексу, который возвращается свойством `ListIndex`. Индексация элементов начинается с 0. В данном коде по выбору элемента (другими словами, операции) из списка этот элемент выводится в заголовке формы. Это обеспечивается обработкой события `Click` списка.

Заполнение списка из диапазона

Свойство `RowSource` позволяет заполнять список из диапазона. Значением этого свойства является строка с именем диапазона. Продемонстрируем работу этого свойства на примере, причем рассмотрим как случай, когда диапазон, из которого берутся данные, известен заранее, так и ситуацию, когда известна единственная его ячейка (см. файл *35-Заполнение списка из диапазона.xlsm* на компакт-диске).

Создадим простой проект по учету поездок сотрудников некоторой фирмы. Итак, имеется книга с тремя рабочими листами: **Города**, **Сотрудники** и **Поездки**.

- На листе **Города** в диапазоны **A1:A5** введен список городов, в которые намечены поездки сотрудников.
- На листе **Сотрудники** в столбец **A**, начиная с ячейки **A1**, введены фамилии сотрудников. Размер этого списка не фиксирован и может изменяться по усмотрению пользователя.
- На листе **Поездки** в два столбца выводится список поездок сотрудников по городам. Заполнение списка на листе **Поездки** производится с помощью окна **Заполнение списка из массива** (рис. 4.34). В этом окне имеются два списка (с именами сотрудников и названиями городов) и кнопка. Списки заполняются из диапазонов соответствующих рабочих листов. Нажатие кнопки **OK** вызывает ввод выбранных данных в очередную пустую строку списка, расположенного на листе **Поездки**.

Создайте форму, на которой расположите два списка и кнопку, и, используя окно **Properties**, установите им значения свойств, как показано в табл. 4.34.

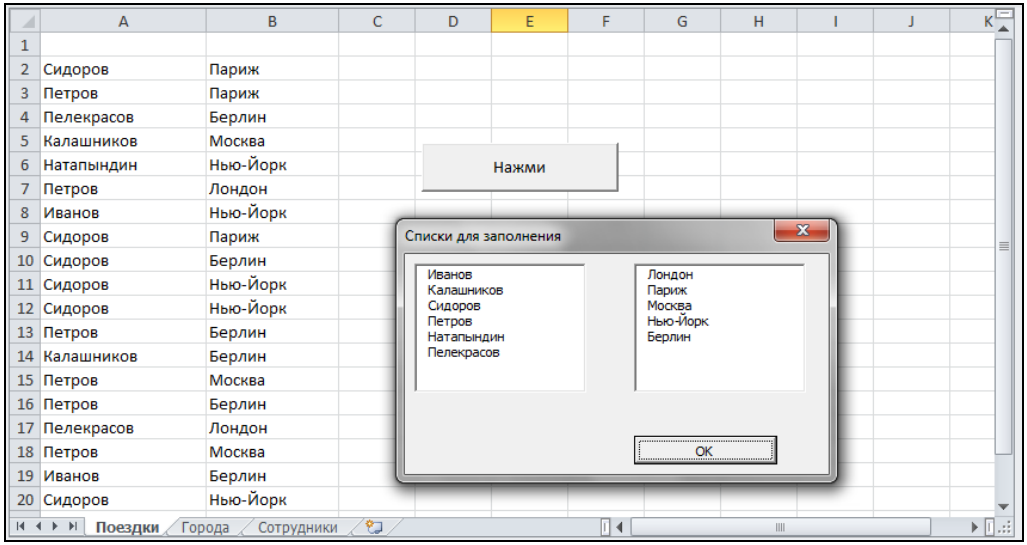


Рис. 4.34. Заполнение списка из диапазона

Таблица 4.34. *Значения свойств, установленные в окне* ***Properties***

Объект	Свойство	Значение
Список	`Name`	`lstName`
Список	`Name`	`lstCity`
Кнопка	`Name`	`cmdOK`
	`Caption`	`OK`

В модуле формы наберите требуемый код (см. файл *35-Заполнение списка из диапазона.xlsm* на компакт-диске). При инициализации формы производится заполнение списков. Список городов заполняется с явным указанием адреса диапазона, а прежде чем заполнить список сотрудников, т. к. по сценарию он может иметь переменный размер, происходит сначала идентификация диапазона, потом определение его адреса, а уже затем заполнение списка. При нажатии кнопки **OK** в первую очередь проверяется, были ли выбраны данные из списков, далее определяется номер первой свободной строки в списке из листа **Поездки**, и в нее выводятся выбранные данные из формы.

Выбор нескольких элементов из списка

В списке можно выбирать как один, так и несколько элементов. Свойство `MultiSelect` устанавливает режим, при котором допустим такой выбор.

Допустимыми значениями свойства `MultiSelect` являются константы:

- `fmMultiSelectSingle` — разрешен выбор только одного элемента;
- `fmMultiSelectMulti` — щелчок на элементе или нажатие клавиши <Пробел> выделяет элемент или снимает с него выделение;

- `fmMultiSelectExtended` — щелчок на элементе при нажатой клавише <Ctrl> выделяет элемент или снимает с него выделение. Щелчок на элементе при нажатой клавише <Shift> добавляет в выделение диапазон элементов от предыдущего выделенного до текущего.

Свойство `Selected` используется для идентификации выбранных элементов. Если его значение равно `True`, то элемент выбран, если равно `False`, то не выбран. Свойство `List` возвращает элемент с указанным индексом. При этом надо помнить, что индексация элементов производится с 0. Общее количество элементов списка возвращается свойством `ListCount`.

Продемонстрируем, как можно выбирать несколько элементов из списка на следующем проекте (рис. 4.35, см. также файл *36-Выбор нескольких элементов из списка.xlsm* на компакт-диске). Создайте форму, на которой расположите список и кнопку, и, используя окно **Properties**, установите им значения свойств, как показано в табл. 4.35.

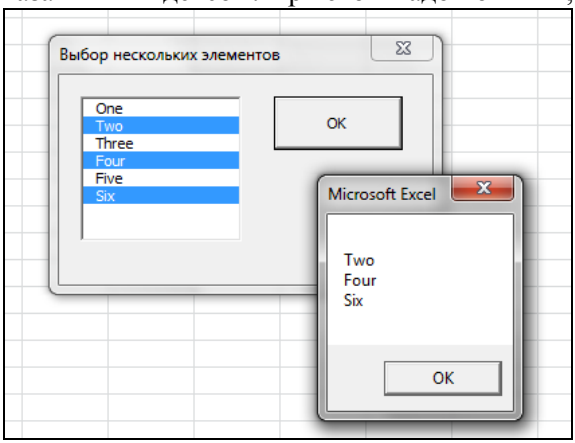


Рис. 4.35. Выбор нескольких элементов из списка

Таблица 4.35. *Значения свойств, установленные в окне* ***Properties***

Объект	Свойство	Значение
Список	`Name`	`lstNum`
Кнопка	`Name`	`cmdOK`
	`Caption`	`OK`

При инициализации формы производится ее заполнение и установка режима, позволяющего выбирать несколько элементов. Нажатие кнопки **OK** вызывает последовательное считывание выбранных элементов и составление на их основе информационной строки, которая отображается в диалоговом окне.

Согласованная работа двух списков

Часто необходимо обеспечить согласованную работу двух списков. Например, некоторое издательство сотрудничает с определенными магазинами в разных городах (рис. 4.36). В один список выводится перечень городов. При выборе же элемента из этого списка во втором списке отображается перечень магазинов. Как этого добиться? Очень просто. Элементы массива могут иметь тип `Variant`, а переменной типа `Variant` может быть все, что угодно, в частности другой массив. Поэтому в VBA допустимо создание массива массивов, а дальше — элементарно, можно, например, поступить, как сделано в коде см. файл *37-Согласованная работа двух списков.xlsm* на компакт-диске.

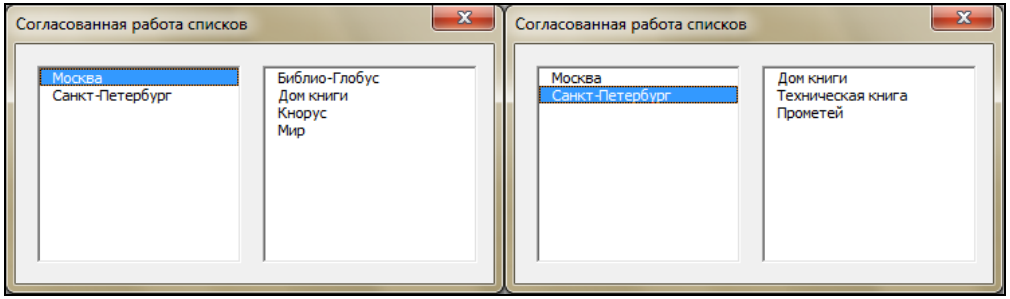


Рис. 4.36. Согласованная работа двух списков

Многостолбцовый список

Списки могут быть не только одностолбцовыми, но и многостолбцовыми. Для этого надо установить значение свойства `ColumnCount`, указывающее, сколько столбцов имеется в списке.

Кроме того, при помощи свойства `ColumnWidths` можно установить ширину каждого из столбцов списка.

```
ColumnWidths = String
```

Здесь параметр *`String`* представляет собой строку, образованную из чисел, которые равны ширине соответствующего столбца. Разделителем является точка с запятой. Например, строка `"90;80"` говорит о том, что под первый столбец отводится 90 пунктов, а под второй — 80.

Доступ к элементам списка производится свойством `List`, первый параметр которого указывает его индекс, а второй — номер столбца. Нумерация как индексов, так и столбцов начинается с 0.

```
List(row, column)
```

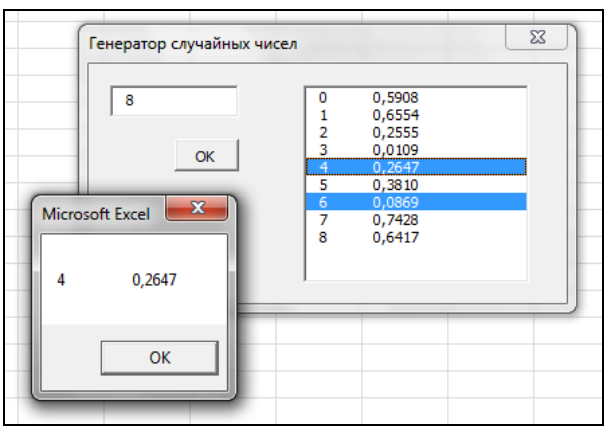


Рис. 4.37. Многостолбцовый список

В качестве примера сконструируем демонстрационный проект, в котором в двухстолбцовый список выводится заданное количество случайных чисел (рис. 4.37, см. также файл *38-Многостолбцовый список.xlsm* на компакт-диске). Создайте форму, на которой расположите список, поле ввода и кнопку, и, используя окно **Properties**, установите им значения свойств, как показано в табл. 4.36.

При инициализации формы устанавливается двухстолбцовый режим списка, под первый столбец отводится 30 пунктов, а под второй — 50. При нажатии кнопки **OK** список сначала очищается методом `Clear`, а затем заполняется указанным количе-

ством случайных чисел с их номерами. При выборе элемента из списка отображается окно с сообщением о сделанном выборе.

Таблица 4.36.** Значения свойств, установленные в окне **Properties

Объект	Свойство	Значение
Список	`Name`	`lstRnd`
Поле ввода	`Name`	`txtNum`
Кнопка	`Name`	`cmdOK`
	`Caption`	`OK`

Заполнение многостолбцового списка из диапазона и нахождение среднего значения выбранных чисел

Многостолбцовый список, так же как и одностолбцовый, можно заполнять из диапазона данных. Продемонстрируем это на примере приложения, вычисляющего средний балл у выбранной группы студентов из списка. Список заполняется данными из диапазона, правым верхним углом которого служит ячейка **A1**. Первый столбец содержит фамилии студентов, а второй — их оценки (рис. 4.38, см. также файл *39-Заполнение многостолбцового списка из диапазона_Средний балл.xlsm* на компакт-диске).

Перейдем к конструированию приложения. Создайте форму с надписью, полем ввода, списком и кнопкой. В модуле формы наберите необходимый код. Заполнение списка и задание значения свойства `Caption` элементов управления и формы производится на этапе инициализации формы. При нажатии кнопки **OK** сначала проверяется, имеются ли выбранные элементы. Это делается потому, что средний балл равен сумме баллов, деленной на их количество. Если в списке не выбран ни один элемент, то данная формула приведет к генерации ошибки деления на ноль.

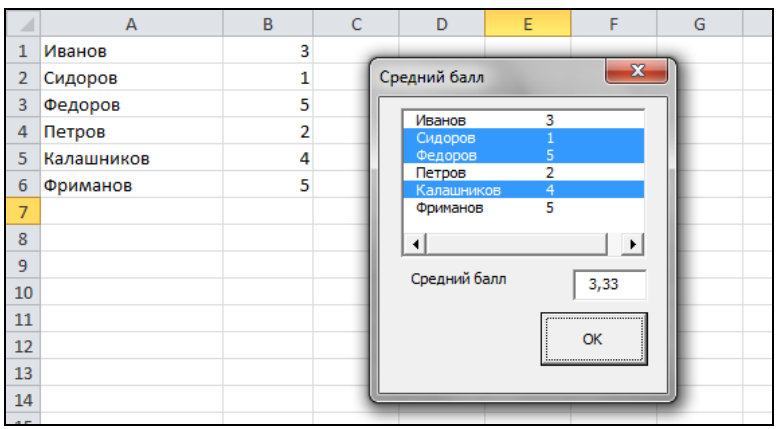


Рис. 4.38. Нахождение среднего балла

Скрытие данных в многостолбцовом списке

Если в значении свойства `ColumnWidths` какой-либо компонент указан нулевым, то соответствующий столбец не отображается. Это можно делать для скрытия некоей вспомогательной информации. Например, создадим следующий демонстрационный проект — отдел кадров. В нем в список на основе диапазона **A2:C9** рабочего листа **Сотрудники** вводится информация о сотрудниках фирмы: фамилия, должность и стаж работы (рис. 4.39). Хотя все эти данные загружаются в список, визуально в нем отображаются только фамилии. При щелчке на элементе списка отображается диалоговое окно с фамилией выбранного сотрудника. Установив требуемую комбинацию флажков **Должность** и **Стаж**, отображаемую о выбранном сотруднике, информацию можно расширить. Для реализации этого проекта создайте книгу с рабочим листом **Сотрудники**, в диапазон **A2:C9** которого введены данные о сотрудниках. Сконструируйте форму, на которой расположите список и два флажка. В модуле формы наберите код (см. файл *40-Скрытие данных в многостолбцовом списке_Отдел кадров.xlsm* на компакт-диске).

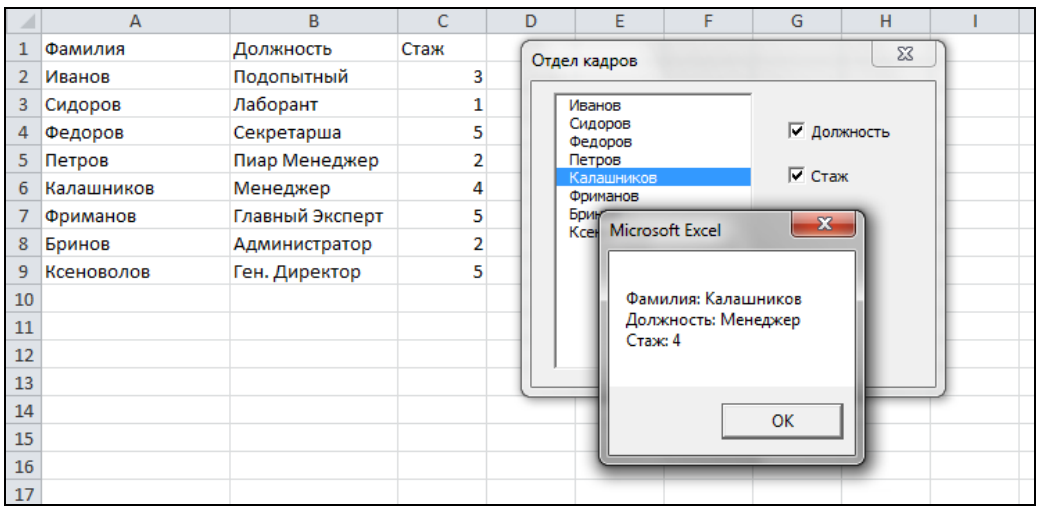


Рис. 4.39. Отдел кадров

Заполнение списка и задание значения свойства `Caption` элементов управления и формы производится на этапе инициализации формы. При выборе элемента списка генерируется зависящая от состояния флажков строка, отображаемая в диалоговом окне. Причем элемент первого столбца возвращается свойством `Text`, а второго и третьего — свойством `List`.

Изменение состояния флажка также будет вызывать выполнение процедуры обработки события `Click` списка при условии, что в нем имеются выбранные элементы. Последнее проверяется значением, возвращаемым свойством `ListIndex`. Дело в том, что это свойство возвращает индекс выбранного элемента. Если же такового нет, то оно возвращает значение –1.

Вывод в многостолбцовом списке выбранного значения при помощи свойств *Text* и *Value*

Свойство `Text` по умолчанию возвращает выбранный элемент из первого столбца. Но эту установку можно изменить. Свойство `TextColumn` задает номер столбца, выбранный элемент из которого возвращается свойством `Text`. Кроме свойства `Text`, возвращающего выбранный элемент, в списке имеется свойство `Value`, действующее аналогично. Столбец, из которого оно возвращает элемент, задается свойством `BoundColumn`. При этом надо учитывать — как в свойстве `TextColumn`, так и в `BoundColumn` нумерация столбцов начинается с 1. Следующий код (листинг 4.16) является модификацией проекта предыдущего раздела, только теперь данные из второго и третьего столбцов считываются при помощи свойств `Text` и `Value`. Так как в списке три столбца (хотя последние два и скрыты), то значение из первого столбца считывается свойством `List`.

Листинг 4.16. Вывод в многостолбцовом списке выбранного значения при помощи свойств `Text` и `Value`

```
Private Sub UserForm_Initialize()
   ListBox1.ColumnCount = 3
   ListBox1.ColumnWidths = "80;0;0"
   Me.Caption = "Отдел кадров"
   ListBox1.RowSource = "Сотрудники!A2:C9"
   CheckBox1.Caption = "Должность"
   CheckBox2.Caption = "Стаж"
   ListBox1.TextColumn = 2
   ListBox1.BoundColumn = 3
End Sub

Private Sub ListBox1_Click()
   Dim msg As String
   msg = "Фамилия: " & ListBox1.List(ListBox1.ListIndex, 0) & vbCr
   If CheckBox1.Value Then
      msg = msg & "Должность: " & ListBox1.Text & vbCr
   End If
   If CheckBox2.Value Then
      msg = msg & "Стаж: " & ListBox1.Value & vbCr
   End If
   MsgBox msg
End Sub
```

Буксировка элементов из одного списка в другой

Объект `DataObject` может служить транспортером данных при программировании операции буксировки. Методы объекта `DataObject` позволяют контролировать процесс буксировки от ее начала до конца (табл. 4.37).

Таблица 4.37. *Методы объекта* `DataObject`

Метод	Описание
`Clear`	Удаляет переносимые данные из объекта
`GetFormat`	Возвращает 1, если переносимые данные имеют строковый формат
`GetFromClipboard`	Копирует содержание буфера обмена в объект
`GetText`	Возвращает строку, переносимую в объекте
`PutInClipboard`	Перемещает данные из объекта в буфер обмена
`SetText`	Копирует строку в объект
`StartDrag`	Инициализирует операцию буксировки. Допустимыми возвращаемыми значениями являются следующие константы: `fmDropEffectNone`, `fmDropEffectCopy`, `fmDropEffectMove` и `fmDropEffectCopyOrMove`

При кодировании операции буксировки надо обрабатывать два события:

- `BeforeDragOver` — генерируется во время операции буксировки;
- `BeforeDropOrPaste` — генерируется непосредственно перед вставкой объекта.

Итак, для демонстрации того, как операция буксировки может быть запрограммирована, создайте форму, расположите в ней два списка и наберите соответствующий код (см. файл *41-Буксировка элементов из одного списка в другой.xlsm* на компакт-диске). Проект готов.

ComboBox (Поле со списком)

Элемент управления **ComboBox** (Поле со списком)  применяется для хранения списка значений. Он сочетает в себе функциональные возможности списка и поля ввода. В отличие от списка, в поле со списком отображается только один элемент списка. У него отсутствует режим выделения нескольких элементов списка. Элемент управления **ComboBox** позволяет пользователю вводить значение через поле ввода, как это делает элемент управления **TextBox** (Поле ввода). Свойства элемента управления **ComboBox**, такие как `ListIndex`, `ListCount`, `List`, и методы `Clear`, `RemoveItem` и `AddItem` аналогичны соответствующим свойствам и методам элемента управления **ListBox** (Список). Кроме того, у него есть ряд уникальных свойств, которые перечислены в табл. 4.38.

Таблица 4.38. *Свойства поля со списком*

Свойство	Описание
`DropButtonStyle`	Устанавливает вид поля со списком. Допустимыми значениями являются следующие константы: `fmDropButtonStylePlain` (кнопка без символов), `fmDropButtonStyleArrowDisplays` (кнопка со стрелкой), `fmDropButtonStyleEllipsis` (кнопка с эллипсом), `fmDropButtonStyleReduce` (кнопка с линией)
`ListRows`	Устанавливает число элементов, отображаемых в поле со списком

Таблица 4.38 *(окончание)*

Свойство	Описание
`MatchRequired`	Допустимые значения: `True` (в поле ввода нельзя ввести значения, отличные от перечисленных в списке, т. е. в поле со списком отключается функция поля ввода) и `False` (в противном случае)
`MatchFound`	Допустимые значения: `True` (среди элементов поля со списком имеется элемент, который совпадает с элементом, вводимым в поле ввода) и `False` (в противном случае)

Поле со списком, ввод данных в алфавитном порядке и объект *Collection*

Создадим простое приложение, которое при вводе данных, например фамилий, через поле ввода поля со списком автоматически сортирует их в алфавитном порядке. При этом вводимые данные сразу же после нажатия клавиши <Enter> отображаются в поле со списком по алфавиту. При закрытии окна все данные из поля со списком вводятся в ячейки рабочего листа (рис. 4.40).

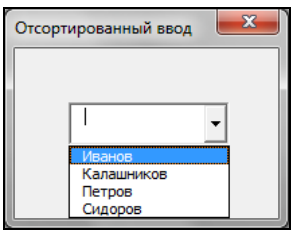


Рис. 4.40. Окно **Отсортированный ввод**

Для реализации данного приложения воспользуемся объектом `Collection`, который представляет собой удобное динамическое хранилище других объектов. Для работы с объектом `Collection` имеется только одно свойство `Count`, возвращающее число элементов, и три метода: `Item` (возвращающий элемент), `Add` (добавляющий новый элемент) и `Remove` (удаляющий элемент). Приводимый далее код, с одной стороны, является хорошим примером, демонстрирующим работу с объектом `Collection` (место в семействе, куда вставляется новый элемент, определяется при помощи алгоритма бинарного поиска), а с другой стороны, показывающим, как из поля ввода поля со списком вводятся данные в сам список.

Итак, создайте форму, разместите на ней поле со списком, а в модуле формы наберите необходимый код (см. файл *42-Поле со списком.xlsm* на компакт-диске).

Добавление и удаление данных в поле со списком

Рассмотрим простое приложение, в котором через поле ввода поля со списком возможно добавление новых элементов, *отличных* от элементов, уже имеющихся в списке. Таким образом, не допускается ввод элемента, совпадающего с одним из элементов списка. Этим рассматриваемый способ ввода данных отличается от того способа, который был рассмотрен в предыдущем разделе. Кроме того, из списка допускается удаление выбранных элементов и очистка всего списка. Первоначально список заполняется данными из диапазона ячеек. Если через поле ввода списка будет введен новый элемент, то он также вводится в диапазон, на основе которого заполнялся список. При удалении элемента из списка он удаляется и из диапазона. В обоих случаях у этого диапазона изменяются размеры, т. к. та или иная строка либо добавляется, либо удаляется. Поэтому приходится постоянно переопределять диапазон, на основе которого заполняется список.

Перейдем к конструированию приложения (см. файл *43-Добавление и удаление в поле со списком.xlsm* на компакт-диске). Создайте форму с полем со списком и тремя кнопками. Установите значения свойства `Name` формы и элементов управления, используя окно **Properties**, как показано в табл. 4.39.

Таблица 4.39. *Значения свойств, установленные в окне* ***Properties***

Элемент управления	Значение свойства `Name`
Список	`cboNames`
Кнопка	`cmdAdd`
Кнопка	`cmdDelete`
Кнопка	`cmdClear`

Список будет заполняться данными, например фамилиями, из диапазона, расположенного в одном столбце с верхней ячейкой **A1**. Нажатие кнопки **Добавить** вызывает добавление в список нового элемента из поля ввода списка, нажатие кнопки **Удалить** приводит к удалению выбранного элемента из списка, а нажатие кнопки **Очистить** инициирует очистку списка (рис. 4.41).

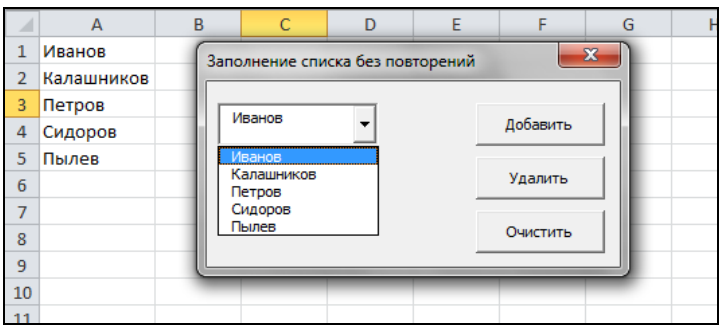


Рис. 4.41. Заполнение списка без повторений

Image (Рисунок)

Элемент управления **Image** (Рисунок)  используется для отображения графических файлов в формате BMP, CUR, GIF, ICO, JPG и WMF. В табл. 4.40 перечислены основные свойства элемента управления **Image**.

Таблица 4.40. *Свойства элемента управления* ***Image***

Свойство	Описание
`AutoSize`	Принимает логические значения и устанавливает, должен ли объект автоматически изменять размер для того, чтобы поместить изображение целиком

Таблица 4.40 *(окончание)*

Свойство	Описание
`Picture`	Задает отображаемый графический файл. Используется с функцией `LoadPicture`
`PictureSizeMode`	Устанавливает масштабирование рисунка. Допустимые значения: `fmPictureSizeModeClip` (не помещающиеся в границах объекта части рисунка обрезаются), `fmPictureSizeModeStretch` (рисунок масштабируется так, чтобы он занимал всю поверхность объекта), `fmPictureSizeModeZoom` (рисунок масштабируется с сохранением относительных размеров так, чтобы он помещался целиком внутри объекта)
`PictureAlignment`	Устанавливает расположение изображения внутри элемента управления. Допустимые значения: `fmPictureAlignmentTopLeft` (в левом верхнем углу), `fmPictureAlignmentTopRight` (в правом верхнем углу), `fmPictureAlignmentCenter` (в центре), `fmPictureAlignmentBottomLeft` (в левом нижнем углу), `fmPictureAlignmentBottomRight` (в правом нижнем углу)
`PictureTiling`	Принимает логические значения и устанавливает, надо ли покрывать объект мозаикой из рисунка

Окно *О программе*

Рисунок часто используется при создании окон типа **О программе** или **Об авторе** для вставки в окно растровых изображений. Следующий проект является демонстрацией подобного применения рисунка (рис. 4.42, см. также файл *44-Рисунок.xlsm* на компакт-диске). Для его реализации создайте окно, в котором расположите рисунок и две надписи. Кроме того, потребуется растровый файл (в данном случае Рудикова.jpg) с вашим или каким-либо другим изображением.

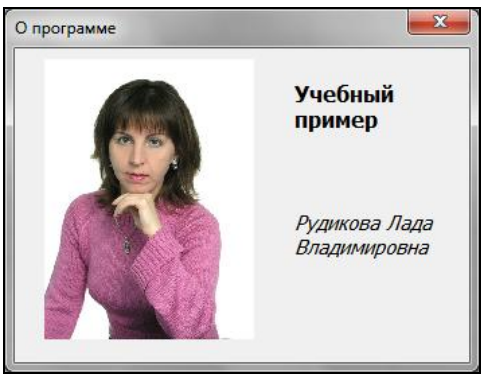


Рис. 4.42. Окно **О программе**

Итак, убедитесь, что в каталоге, который используется MS Excel по умолчанию, находится необходимый графический файл, который вы хотите отобразить в виде мозаичного фона. В модуле формы наберите приводимый необходимый код. Обратите внимание на то, что в различных надписях для придания окну большей презентабельности используются разные установки параметров шрифта.

Просмотр слайдов

Рисунок позволяет создавать простое средство для просмотра слайдов. Как это делается, продемонстрируем на простом примере. Здесь мы реализуем приложение, позволяющее просматривать несколько картинок (рис. 4.43, см. также файл *45-Показ слайдов.xlsm* на компакт-диске). Итак, создайте форму, в которой располо-

жите рисунок и список. Кроме того, потребуются файлы с соответствующими растровыми изображениями (в данном случае 1.jpg, 2.jpg, 3.jpg), которые необходимо поместить в тот же каталоге, который используется MS Excel по умолчанию. В модуле формы наберите соответствующий код. Вот и все. Проект готов.

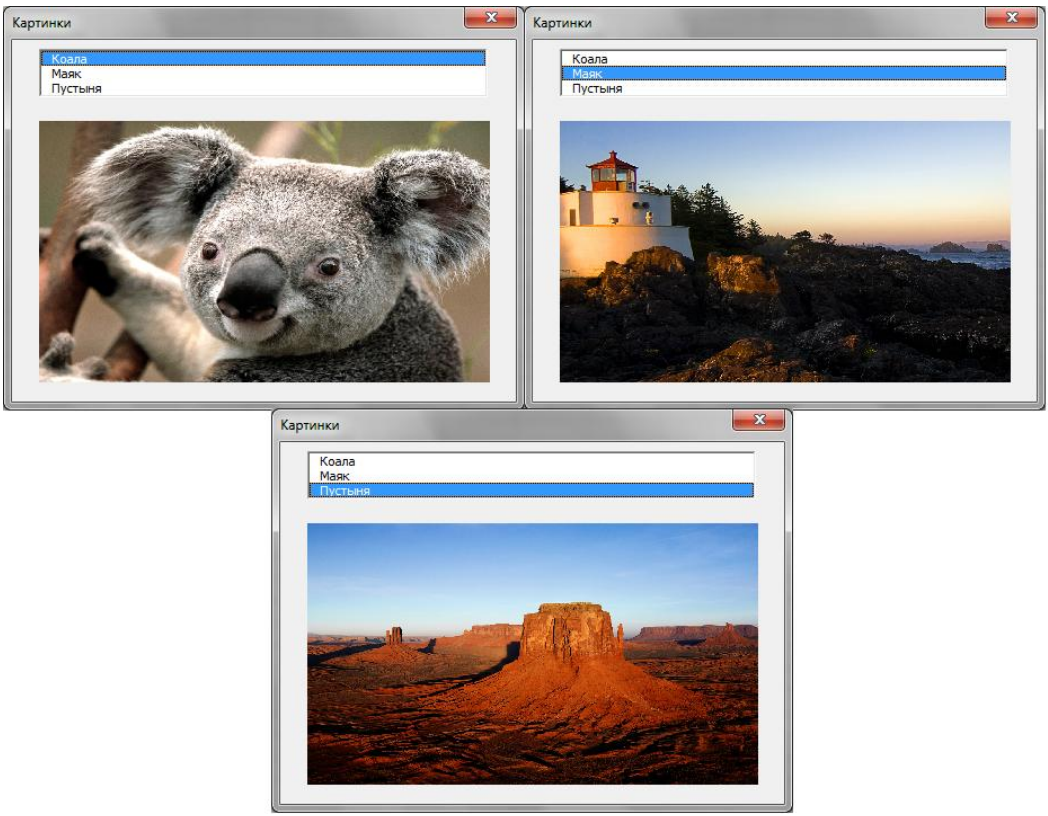


Рис. 4.43. Просмотр слайдов

Модифицированный мастер диаграмм

Построение графика с помощью мастера диаграмм предполагает довольно большой объем подготовительной работы: заполнение диапазона значениями аргумента, заполнение другого диапазона значениями функции и непосредственное построение мастером диаграмм графика в четыре шага.

Создадим приложение, которое будет заполнять диапазоны со значениями аргумента и функции, строить диаграмму и отображать диаграмму не только на рабочем листе, но и в форме. В этом приложении от пользователя будет требоваться только ввод границ интервала, на котором строится график, и самой функции, причем функцию необходимо вводить не в виде формулы рабочего листа, а в привычном виде, в качестве аргумента которой нужно использовать символ "z" (рис. 4.44, см. также файл *46-Модифицированный мастер диаграмм.xlsm* на компакт-диске).

Использование символа "z" в качестве аргумента функции связано с целью упрощения кода.

Перейдем к разработке приложения. Создайте форму с тремя надписями, тремя полями ввода, рисунком и кнопкой. Установите значения свойства `Name` элементов управления в окне **Properties**, как показано в табл. 4.41.

Таблица 4.41. *Значения свойства* `Name` *элементов управления, установленные в окне* ***Properties***

Элемент управления	**Значение свойства `Name`**	**Описание**
Поле ввода	`txtBegin`	Ввод левого конца интервала, на котором строится график
Надпись	`lblBegin`	Надпись, соответствующая полю ввода `txtBegin`
Поле ввода	`txtEnd`	Ввод правого конца интервала, на котором строится график
Надпись	`lblEnd`	Надпись, соответствующая полю ввода `txtEnd`
Поле ввода	`txtFun`	Ввод формулы функции, для которой строится график. Формула должна быть записана по правилам программирования, в качестве аргумента использован символ "z"
Надпись	`lblFun`	Надпись, соответствующая полю ввода `txtFun`
Рисунок	`imgFun`	Отображается растровое изображение из файла Graph.gif, который содержит графический образ диаграммы
Кнопка	`cmdReady`	График функции строится по 101 точке. Первоначально определяется шаг табуляции аргумента функции как результат деления разности между начальным и конечным значениями аргумента на 100. После этого начальное значение вводится в ячейку **A2**, и при помощи метода `DataSeries` производится табуляция аргумента функции вниз по столбцу. В формуле функции заменяется аргумент `z` ссылкой на ячейку **A2**, а перед формулой помещается знак равенства. Полученная таким образом формула рабочего листа вводится в ячейку **B2**. Методом `AutoFill` формула буксируется вниз по столбцу так, чтобы найти значения функции для всех протабулированных значений аргумента. По протабулированным значениям аргумента и функции строится диаграмма, графический образ которой экспортируется в файл Graph.gif. Отображается растровое изображение из файла Graph.gif в рисунке `imgFun`

Для завершения создания приложения в модуле формы наберите соответствующий код. В качестве параметра функции будем использовать символ "z", т. к. данный символ не входит в имена встроенных функций рабочего листа. Это позволяет, используя только одну инструкцию

```
f = Replace(LCase(f), "z", "A2")
```

заменить все вхождения параметра в функцию ссылкой на ячейку **A2**. Если бы в качестве аргумента использовался символ "x", то здесь в общем случае не обойтись одной инструкцией. Дело в том, что символ "x" входит в имя функции `Exp()`. Поэтому применение данной инструкции будет заменять ссылкой на ячейку **A2** не только аргумент, но и встречаемые упоминания имени функции `Exp()` на `EA2p`, что, конечно, приведет к генерации ошибки.

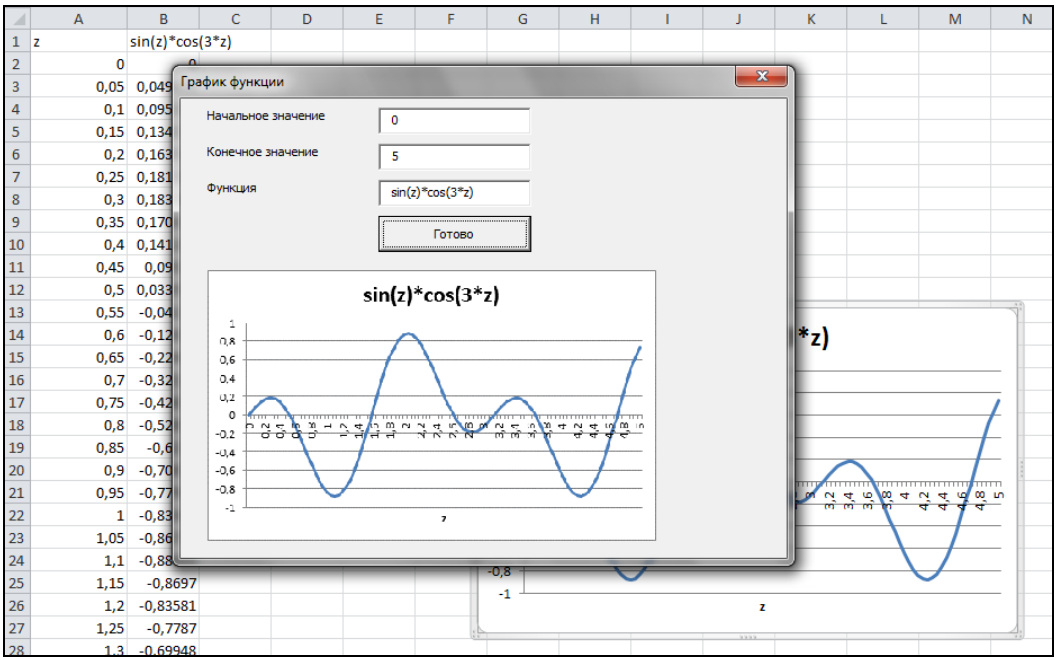


Рис. 4.44. Построение графика модифицированным мастером диаграмм

Элемент управления *RefEdit*

Элемент управления **RefEdit** аналогичен полю ввода, но позволяет вводить в него ссылку на диапазон выбором на рабочем листе. Свойство `Value` возвращает эту ссылку.

> ***Совет***
>
> Чтобы добавить элемент управления **RefEdit** на панель инструментов **Toolbox**, щелкните по ней правой кнопкой мыши и выберите команду **Aditional Controls**. В открывшемся окне **Aditional Controls** выберите из списка **Available Controls** элемент **ReEditCtrl** и нажмите кнопку **OK**.

Определение статистических параметров диапазона

В качестве примера использования элемента управления **RefEdit** сконструируем простой проект, определяющий некоторые статистические параметры диапазона, а именно — максимальное, минимальное значения, а также сумму всех значений ячеек этого диапазона. Итак, создайте форму, на которой расположите кнопку

и элемент управления **RefEdit** (рис. 4.45). В модуле формы наберите необходимый код (см. файл *47-Статистика.xlsm* на компакт-диске). Проект готов. Для вычисления максимального, минимального значений, а также суммы всех значений ячеек диапазона используются свойства `Min`, `Max` и `Sum` объекта `WorksheetFunction`, представляющие собой одноименные функции рабочего листа.

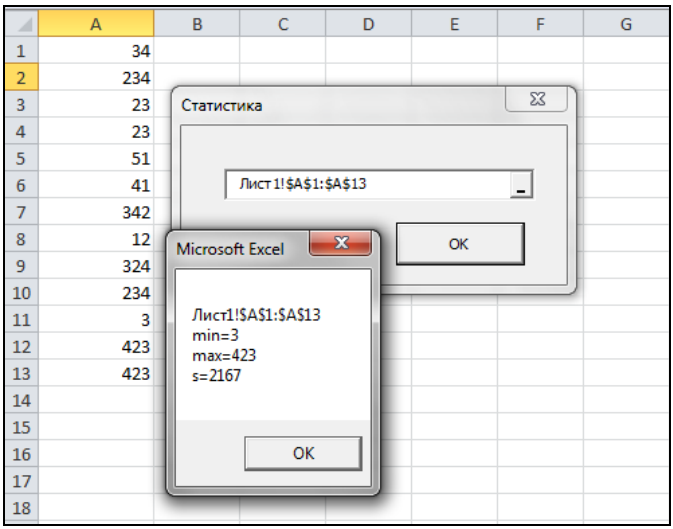


Рис. 4.45. Статистика

Решение системы линейных уравнений

В качестве еще одного примера использования элемента управления **RefEdit** создадим проект по решению систем линейных уравнений $\mathbf{AX}=\mathbf{B}$ (листинг 4.56). Здесь **A** — квадратная $n\times n$ матрица, **B** — столбец свободных членов, а **X** — столбец неизвестных. При решении системы уравнений используются функции рабочего листа `МОБР()` и `МУМНОЖ()`, возвращающие матрицу, обратную к данной, а также результат перемножения двух матриц. Кроме того, для проверки существования решения системы применяется функция `МОПР()`, возвращающая определитель квадратной матрицы.

В окне формы имеется элемент управления **RefEdit** и кнопка. Пользователь вводит в элемент управления **RefEdit** ссылку на диапазон размера $n\times(n+1)$, первые n столбцов которого содержат матрицу коэффициентов, а последний столбец — столбец свободных членов. Нажатие кнопки вызывает нахождение решения системы и вывод его в диапазон, справа смежный с выделенным.

Например, на рис. 4.46 (см. также файл *48-Решение системы.xlsm* на компакт-диске) решена следующая система линейных уравнений:

$$\begin{cases}12x_1+3x_2=2,\\23x_1+45x_2=3.\end{cases}$$

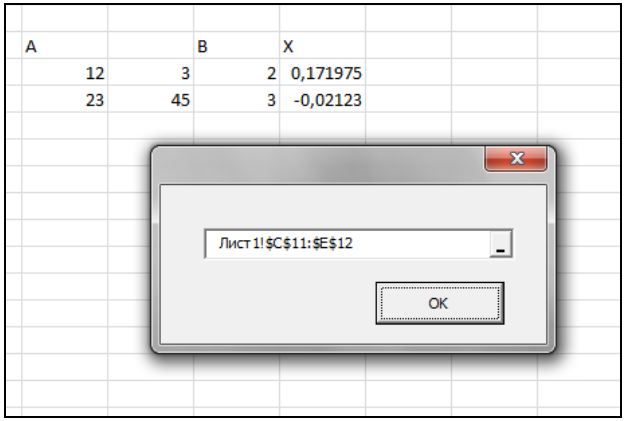


Рис. 4.46. Решение системы линейных уравнений

Матрица коэффициентов расположена в диапазоне **B3:C4**, столбец свободных членов — в диапазоне **D3:D4**, решение системы выводится в диапазон **E3:E4**.

MultiPage (Набор страниц) и *TabStrip* (Набор вкладок)

Элемент управления **MultiPage** (Набор страниц) позволяет реализовать многостраничные диалоговые окна. Заголовки страниц отображаются на вкладках. Переход от страницы к странице осуществляется щелчком на вкладке. Создать, переименовать, удалить или переместить страницу элемента управления **MultiPage** (Набор страниц) можно вручную, выбрав ярлык соответствующего листа и вызвав щелчком правой кнопки мыши контекстное меню. Объект `MultiPage` содержит в себе семейство `Pages`, являющееся набором всех страниц, входящих в этот объект. В табл. 4.42 перечислены основные свойства объекта `MultiPage`.

Таблица 4.42. *Основные свойства объекта* `MultiPage`

Свойство	Описание
`Value`	Возвращает или устанавливает номер активной страницы. Нумерация производится с нуля
`MultiRow`	Свойство, принимающее логические значения. Если его значение равно `True`, то ярлыки, не помещающиеся в одну строчку, выводятся в несколько строчек. Если его значение равно `False`, то в случае, когда ярлыки не помещаются в одну строчку, появляется полоса прокрутки, позволяющая переходить от страницы к странице
`SelectedItem`	Возвращает выбранную страницу

Семейство `Pages` состоит из всех объектов `Page`, т. е. страниц объекта `MultiPage`. Семейство `Pages` имеет единственное свойство `Count`, возвращающее число элементов семейства. Методы семейства `Pages` перечислены в табл. 4.43.

Таблица 4.43. *Методы семейства* `Pages`

Метод	Описание
`Add`	Создает новую страницу
`Clear`	Удаляет все страницы из семейства `Pages`
`Remove`	Удаляет страницу из семейства `Pages`
`Item`	Возвращает страницу с указанным в качестве индексом

Элемент управления **TabStrip** (Набор вкладок) создает несколько вкладок в диалоговом окне и по своим функциональным возможностям эквивалентен набору страниц. Объект `TabStrip` содержит в себе семейство `Tabs`, представляющее собой набор всех вкладок. Объект `TabStrip` и семейство `Tabs` обладают теми же свойствами и методами, что и объект `MultiPage` и семейство `Pages`.

Статистика и набор страниц

В качестве примера приложения с набором страниц улучшим интерфейс проекта из *разд. "Определение статистических параметров диапазона" ранее в этой главе* (рис. 4.47, см. также файл *49-Статистика и набор страниц.xlsm* на компакт-диске). Создайте форму, на которой расположите набор страниц. На первой странице, значение `Caption` которой установите равным `Вычисления`, разместите элемент управления **RefEdit**, поле ввода и кнопку. На второй странице, значение `Caption` которой установите равным `Параметры`, разместите три флажка.

Используя окно **Properties**, установите значения свойств элементов управления, как показано в табл. 4.44.

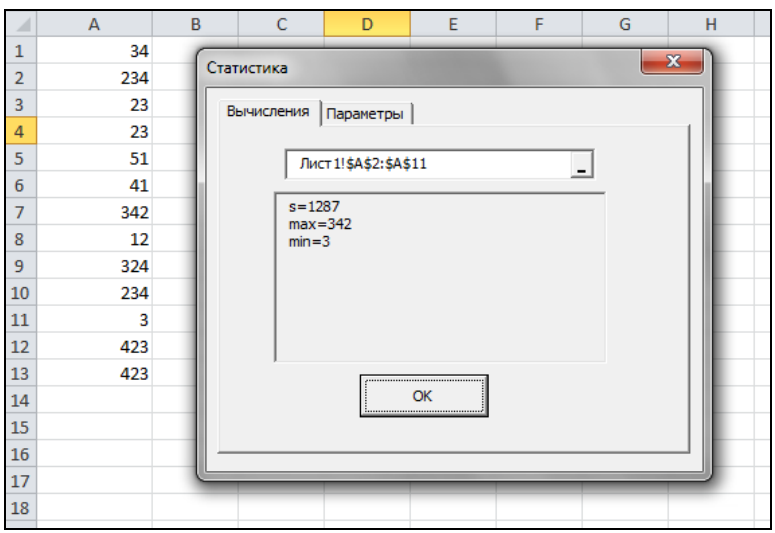


Рис. 4.47. Статистические параметры диапазона и набор страниц

Таблица 4.44. *Значения свойств, установленные в окне* ***Properties***

Объект	Свойство	Значение
Поле ввода	Name	txtStat
Кнопка	Name	cmdOK
	Caption	OK
Флажок	Name	chkSum
	Caption	Сумма
Флажок	Name	chkMax
	Caption	Максимум
Флажок	Name	chkMin
	Caption	Минимум

В модуле формы наберите необходимый код. Проект готов. В поле ввода выводится в несколько строчек интегрированная статистика, параметры которой устанавливаются флажками страницы **Параметры**. Поле ввода заблокировано для корректировки результатов со стороны пользователя. Кроме того, цвет фона поля установлен равным цвету фона формы. Поэтому визуально поле ввода воспринимается как вдавленная в форму надпись.

Последовательность перехода элементов управления

Последовательность перехода определяет порядок, в котором получают фокус элементы управления при нажатии клавиши <Tab>. При нажатии комбинации клавиш <Shift>+<Tab> элементы управления получают фокус в обратном порядке. Для установки последовательности перехода формы необходимо:

1. Находясь в редакторе Visual Basic, выбрать команду **View | Tab Order**.
2. В появившемся диалоговом окне **Tab Order** (Последовательность перехода) с помощью кнопок **Move Up** (Вниз) и **Move Down** (Вверх) изменить последовательность перехода выделенного элемента управления в зависимости от потребности.

Другим способом задания последовательности перехода является определение значения свойства `TabIndex` элемента управления. При этом надо помнить, что начальному элементу соответствует значение свойства `TabIndex`, равное 0, второму — 1, третьему — 2 и т. д. Если значение свойства `TabStop` равно `False`, то элемент управления не получает фокус при обходе элементов управления нажатием клавиши <Tab>. Если же значение свойства `TabStop` равно `True` (значение, используемое по умолчанию), то элемент управления получает фокус.

Отображение встроенных диалоговых окон

VBA позволяет программно отображать на экране встроенные в MS Excel диалоговые окна наряду с пользовательскими диалоговыми окнами. Все встроенные диалоговые окна в MS Excel образуют семейство `Dialogs`, параметр которого специфицирует активизируемое диалоговое окно (табл. 4.45). Отображение встроен-

ного диалогового окна на экране осуществляется методом `Show`. Например, процедура из листинга 4.17 (см. также файл *50-Кнопка для открытия документа.xlsm* на компакт-диске) при щелчке на кнопке активизирует диалоговое окно **Открытие документа** (`Open`) (рис. 4.48).

Листинг 4.17. Отображение диалогового окна Открытие документа

```
Private Sub CommandButton1_Click()
   Application.Dialogs(xlDialogOpen).Show
End Sub
```

Таблица 4.45. *Значения параметра семейства* `Dialogs`

Значение параметра	Диалоговое окно
`xlDialogFindFile`	Открытие документа при поиске файла
`xlDialogFileDelete`	Удалить файл
`xlDialogGoalSeek`	Подбор параметра
`xlDialogSaveAs`	Сохранить как
`xlDialogSaveWorkbook`	Сохранить
`xlDialogPrint`	Печать
`xlDialogPrintPreview`	Предварительный просмотр

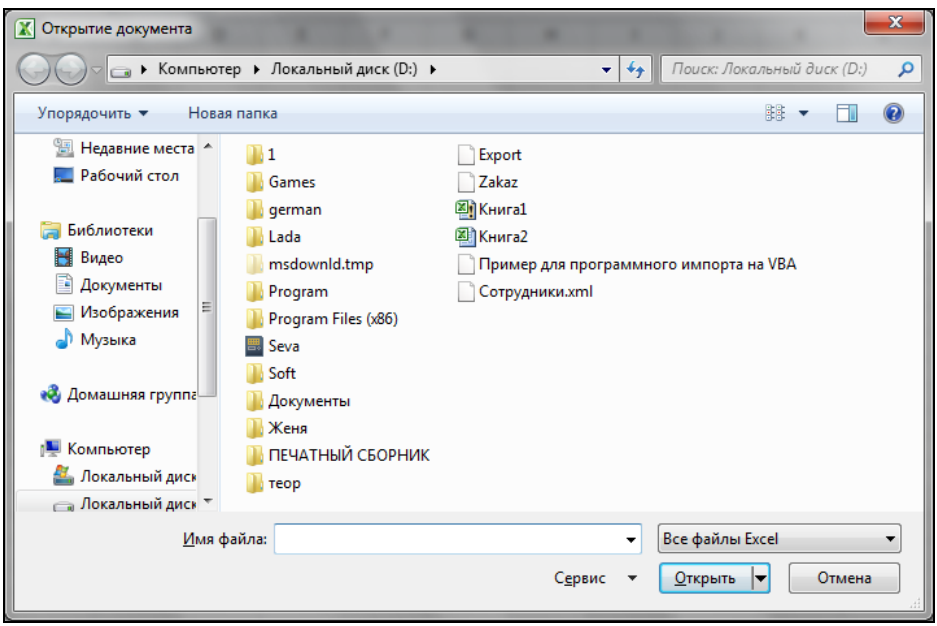


Рис. 4.48. Диалоговое окно **Открытие документа**

В методе `Show` можно указывать значения параметров, управляющих выводом в диалоговом окне специфицированной информации. Например, при отображении диалогового окна **Подбор параметра** можно указать значения параметров `target_cell` (поле ввода **Установить в ячейке**), `target_value` (поле ввода **Значение**), `variable_cell` (поле ввода **Изменяя значение ячейки**), в результате чего на экране отобразится окно с заполненными полями ввода (листинг 4.18).

Листинг 4.18. Отображение диалогового окна *Подбор параметра* с введенными значениями

```
Private Sub DoGoalSeek ()
  Dim fl As Boolean
  fl = Application.Dialogs(xlDialogGoalSeek).Show(Range("A1"), 0, Range("A2"))
  If fl Then
     MsgBox "Решение найдено"
  Else
     MsgBox "Решение не найдено"
  End If
End Sub
```

Открытие документа и метод *GetOpenFilename*

Существует еще один способ отображения окна **Открытие документа**, а именно методом `GetOpenFilename` объекта `Application`. Этот метод приводит к отображению на экране окна **Открытие документа**, но не к открытию выбранного при помощи этого окна документа. Метод просто возвращает имя выбранного файла или значение `False`, если файл не был выбран. Для открытия же выбранного файла надо дополнительно воспользоваться методом `Open` семейства `Workbooks` (листинг 4.19, см. также файл *51-Кнопки для открытия и удаления документа.xlsm* на компакт-диске).

Листинг 4.19. Открытие файла

```
Sub OpenDoc()
   Dim FName As Variant
   FName = Application.GetOpenFilename( _
             "Книга Microsoft Excel (*.xlsx), *.xlx")
   If FName <> False Then
      Workbooks.Open FName
   Else
      MsgBox "Файл не выбран"
   End If
End Sub
```

Устанавливая значения параметров метода `GetOpenFilename`, можно управлять видом диалогового окна. В общем случае этот метод имеет следующий синтаксис:

`GetOpenFilename(`*`FileFilter`*`, `*`FilterIndex`*`, `*`Title`*`, `*`ButtonText`*`, `*`MultiSelect`*`)`

- *FileFilter* — необязательный парамер, являющийся строкой, задающей фильтр для отображаемых файлов. Если этот параметр опущен, то его значение по умолчанию полагается равным `"Все файлы (*.*),*.*"`. Еще одним примером фильтра может быть, например, строка `"Книга Microsoft Excel (*.xlsx), *.xlsx, Растровые файлы (*.bmp), *.bmp"`. (Здесь через запятую указываются те типы файлов, которые будут выводиться в списке **Тип файлов**.)
- *FilterIndex* — необязательный параметр, задающий индекс фильтра из списка **Тип файлов**, который используется по умолчанию.
- *Title* — необязательный параметр, определяющий заголовок окна.
- *ButtonText* — используется только в Mac OS X.
- *MultiSelect* — необязательный параметр, принимающий логические значения. Если его значение равно `True`, то допустим выбор нескольких файлов. В последнем случае метод возвращает не строку, а массив строк.

Так как метод `GetOpenFilename` не совершает действий над файлами, а только возвращает имена выбранных файлов, то отображаемое окно можно использовать для функций, отличных от открытия выбранного файла — например, удаления выбранных файлов (листинг 4.20, см. также файл *51-Кнопки для открытия и удаления документа.xlsm* на компакт-диске). При этом, конечно, изменяется заголовок окна в соответствии с его новыми функциями.

Листинг 4.20. Удаление файлов

```
Sub DeleteFile()
   Dim FName As Variant
   FName = Application.GetOpenFilename( _
               FileFilter:="Книга Microsoft Excel (*.xls), *.xls", _
               MultiSelect:=True, Title:="Удаление файла")
   If Not IsArray(FName) Then
      MsgBox "Файл не выбран"
      Exit Sub
   End If
   Dim i As Integer
   For i = LBound(FName) To UBound(FName)
       Kill FName(i)
   Next
End Sub
```

Простейший браузер для графических файлов

Метод `GetOpenFilename` возвращает имя выбранного файла любой природы. В частности, он может вернуть и растровый файл. С другой стороны, в рисунок может быть загружено изображение любого растрового файла. Вместе эти два

Рис. 4.49. Простейший браузер для графических файлов

факта обеспечивают возможность построения простого средства просмотра растровых файлов (рис. 4.49). Итак, создайте форму, расположите в ней кнопку и рисунок, а в модуле формы наберите необходимый код (см. файл *52-Графический браузер.xlsm* на компакт-диске). Проект готов.

Сохранение документа и метод *GetSaveAsFilename*

Метод `GetSaveAsFilename` объекта `Application` отображает на экране окно **Сохранение документа**. Этот метод так же не выполняет сохранение файла, а только возвращает имя сохраняемого файла, выбранного в этом окне. Сохранить же файл можно, дополнив код инструкцией, в которой применяется один из методов `SaveAs` или `Save` объекта `Workbook`.

Синтаксис метода `GetSaveAsFilename` похож на синтаксис метода `GetOpenFilename`.

```
GetSaveAsFilename(InitialFilename, FileFilter, FilterIndex, Title, ButtonText)
```

- *`InitialFilename`* — необязательный параметр, задающий любое допустимое имя файла, которое будет появляться в поле **Имя файла** (по умолчанию, там появляется имя файла `Книга`).
- *`FileFilter`* — необязательный параметр, устанавливающий фильтр для отображаемых файлов. Если вы хотите, чтобы к имени файла автоматически добавлялось расширение, то использование этого параметра обязательно.
- *`FilterIndex`* — необязательный параметр, указывающий, какой фильтр из списка **Тип файлов** будет использоваться по умолчанию.
- *`Title`* — необязательный параметр, определяющий заголовок окна.
- *`ButtonText`* — используется только в Mac OS X.

Например, следующий код (листинг 4.21, см. также файл *53-Кнопка для сохранения документа.xlsm* на компакт-диске) можно использовать при сохранении рабочей книги, причем по умолчанию предлагается имя файла — Отчет.

Листинг 4.21. Сохранение файла

```
Sub SaveDoc()
   Dim FName As Variant
   FName = Application.GetSaveAsFilename(InitialFilename:="Отчет", _
           FileFilter:="Книга Microsoft Excel (*.xlsx), *.xls")
   If FName <> False Then Application.ThisWorkbook.SaveAs FName
End Sub
```

Дополнительные элементы управления

В VBA кроме перечисленных стандартных элементов управления имеется ряд дополнительных. Дополнительные элементы управления являются самостоятельными объектами, обладающими как общими для всех элементов управления свойствами и методами, так и присущими только им свойствами и методами.

Добавление дополнительного элемента управления

Добавляются элементы управления в панель элементов (новая форма должна быть добавлена в проект) следующим образом:

1. Выберите команду **Tools | Additional Controls**.
2. В отобразившемся окне **Additional Controls** (рис. 4.50) в списке **Available Controls** установите флажок напротив добавляемого элемента управления.
3. Нажмите кнопку **OK**.

В результате значок выбранного дополнительного элемента управления появится в панели элементов **Toolbox**.

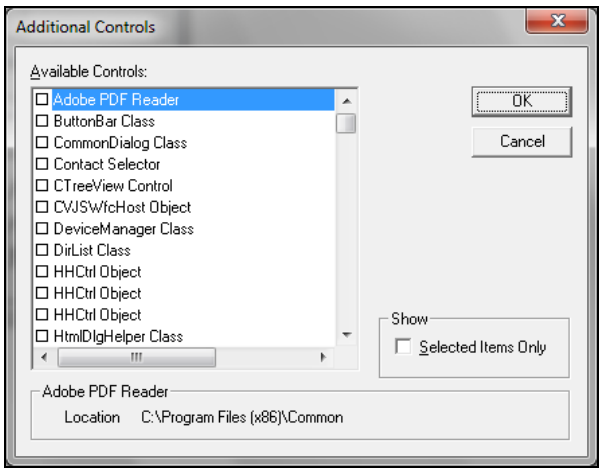


Рис. 4.50. Окно **Additional Controls**

> ***Примечание***
>
> Распространять приложение, имеющее дополнительный элемент управления, необходимо совместно с соответствующим OCX-файлом. Имя OCX-файла и его местоположение можно узнать из надписи **Location** окна **Additional Controls**.

Удаление дополнительного элемента управления

Удаляется ненужный элемент управления из панели элементов почти аналогично тому, как добавляется в нее, а именно:

1. Выберите команду **Tools | Additional Controls**.
2. В отобразившемся на экране окне **Additional Controls** в списке **Available Controls** снимите флажок напротив удаляемого элемента управления.
3. Нажмите кнопку **OK**.

Разрабатываем пользовательские приложения

А сейчас мы рассмотрим некоторые примеры создания пользовательских приложений, которые помогут в значительной мере облегчить работу с данными. С учетом того что в данной главе достаточно подробно рассматривались общие приемы работы с формами и элементами управления, приведем лишь некоторые рекомендации по разработке создаваемых бизнес-приложений.

Заполнение табличного списка данных

Одно из основных преимуществ MS Excel — возможность работы с однотипными массивами данных, которые получли название списки.

Списки в MS Excel — это таблицы, строки которых содержат однородную информацию. Строки таблицы называются *записями*, а столбцы — *полями записей*.

Столбцам присваиваются уникальные имена полей, которые заносятся в первую строку списка — *строку заголовка*.

В Microsoft Office Excel 2010 существуют, например, следующие способы ввода данных в список:

- использование формы данных, которая автоматически создается после определения заголовка списка с помощью команды **Форма**;

> ***Примечание***
>
> В предыдущих версиях MS Excel (включая версию 2003), вызвать окно формы данных можно было с использованием команды линейки меню **Данные | Форма**. В версиях MS Excel 2007 и MS Excel 2010 вам необходимо добавить соответствующую команду **Форма** на панель быстрого доступа или на соответствующую вкладку с использованием окна **Параметры Excel**, выбрав в нем (MS Excel 2010) слева категорию **Настройка ленты** или **Панель быстрого доступа**.

- ввод данных во вставляемые в список пустые строки; в этом случае имя диапазона списка переопределяется автоматически (непосредственно ввод данных);
- использование средства **Автоввода, Прогрессии** и команды **Выбрать из списка** для ускорения работы;
- использование форм MS Access и дальнейший перенос данных на лист MS Excel;
- применение VBA — написанная вами соответствующая программа будет предоставлять форму или окно диалога для ввода данных и их последующего помещения в определенные ячейки рабочего листа MS Excel.

Отметим, что встроенное средство *форма*, диалоговое окно которой отображается выбором соответствующей команды **Форма**, позволяет заполнять и редактировать записи в таблице списка данных. Большим неудобством этого средства является то, что каждому полю записи в нем соответствует только поле ввода.

При заполнении же данных списка значения некоторых полей часто выбираются из конечной группы альтернатив, например, пол может быть мужским или женским, список фамилий сотрудников в отделе продаж ограничен и т. д. Для ввода подобных альтернативных значений удобнее использовать списки, переключатели, флажки, а не ограничиваться полями ввода, как это имеет место в форме. При этом использование списков, переключателей и флажков, во-первых, ускорит процесс заполнения таблицы, а во-вторых, избавит от опечаток, которые неизбежны при ручном вводе данных.

Создадим приложение, которое будет учитывать указанные ранее недостатки формы. Итак, предположим, что вы — менеджер туристической фирмы "Сквозь пространство и время!" и заносите информацию о каждом клиенте в список данных (см. файл *54-Список данных.xlsm* на компакт-диске). Для ускорения ввода данных вы решили создать приложение с диалоговым окном. На рис. 4.51 приведена таблица списка данных и диалоговое окно, которое будет создано. Во избежание ошибок при вводе данных повторяющаяся информация, такая как направление тура и вид транспорта, вводится из списков. Списки же заполняются на основе данных, собранных на рабочих листах **Тур** и **Транспорт**.

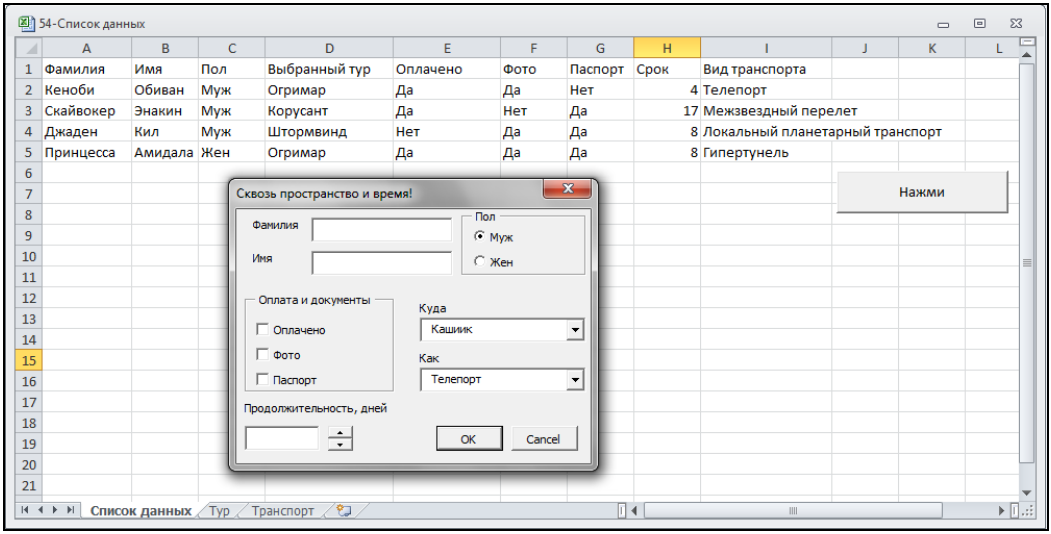


Рис. 4.51. Заполнение табличного списка данных

Итак, перейдем к конструированию приложения. Создайте форму, а на ней расположите три поля ввода, пять надписей, счетчик, две кнопки, два поля со списком, одну рамку, содержащую два переключателя, а другую — три флажка. При помощи окна **Properties** установите им значения свойств, как показано в табл. 4.46.

Таблица 4.46. *Значения свойств, установленные в окне* ***Properties***

Объект	**Свойство**	**Значение**
Форма	`Caption`	`Регистрация туристов фирмы "Сквозь пространство и время!"`
Надпись	`Caption`	`Фамилия`
Поле ввода	`Name`	`txtLName`
Надпись	`Caption`	`Имя`
Поле ввода	`Name`	`txtFName`
Рамка	`Caption`	`Оплата и документы`
Флажок	`Caption`	`Оплачено`
	`Name`	`chkPayment`
Флажок	`Caption`	`Фото`
	`Name`	`chkPhoto`
Флажок	`Caption`	`Паспорт`
	`Name`	`chkPassport`
Надпись	`Caption`	`Продолжительность, дней`
Поле ввода	`Name`	`txtDays`

Таблица 4.46 *(окончание)*

Объект	Свойство	Значение
Счетчик	Name	SpnDays
Рамка	Caption	Пол
Переключатель	Name	optMale
	Caption	Муж
Переключатель	Name	optFemale
	Caption	Жен
Надпись	Caption	Куда *(направление тура)*
Поле со списком	Name	cmbTour
Надпись	Caption	Как *(вид транспорта)*
Поле со списком	Name	cmbTrans
Кнопка	Name	cmdOK
	Caption	OK
Кнопка	Name	cmdCancel
	Caption	Отмена

В модуле формы наберите необходимый код (см. файл *54-Список данных.xlsm* на компакт-диске), который обеспечивает:

- считывание данных из диалогового окна и ввод записи в первую пустую строку таблицы;
- ввод продолжительности тура либо с клавиатуры, либо из окна счетчика, которые работают синхронно;
- защиту всех данных на рабочих листах от изменений со стороны пользователя.

Сервисные возможности для рабочей книги

Приведем пример создания пользовательской формы, которая поддерживает следующие сервисные возможности для рабочей книги Excel: изменяет цвет сетки на рабочих листах, добавляет в книгу необходимое количество рабочих листов, а также заполняет заданный массив на выбранном рабочем листе нулями или единицами.

Итак, приведем вкратце рекомендации по созданию такого проекта (см. файл *55-Добавление листов, изменение цвета сетки, заполнение массива.xlsm* на компакт-диске).

1. Перейдите в окно редактора кода VBA.
2. Добавьте пользовательскую форму, например, командой **Insert | UserForm**.
3. Поместите на форму необходимые элементы управления и установите им соответствующие свойства так, чтобы они приняли вид, аналогичный рис. 4.52.
4. В модуле формы наберите соответствующий код.
5. Расположите на первом рабочем листе кнопку, в процедуру обработки события `Click` которой также добавьте необходимый код.

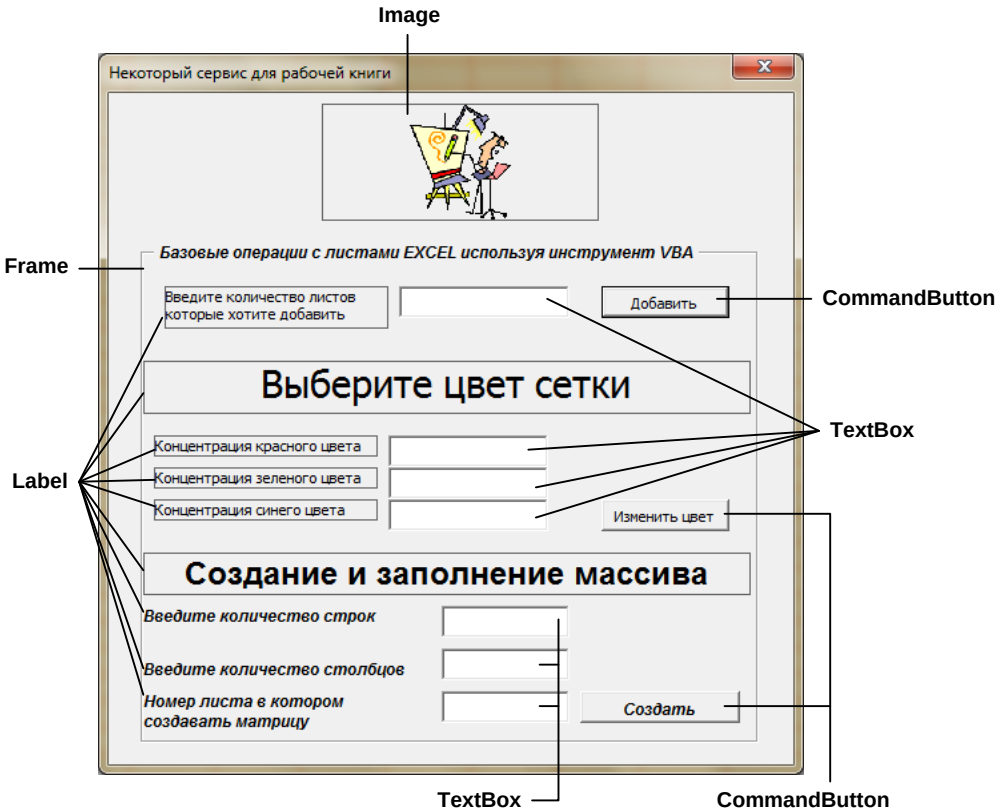


Рис. 4.52. Конструирование пользовательской формы

Разработка модели склада

В примере *56-Модель склада*, который расположен на прилагаемом компакт-диске, демонстрируется приложение, позволяющее управлять складом товаров.

Приведем основные этапы создания данного приложения.

Сначала разместим необходимые данные, касающиеся организации склада, на листах Excel.

- Создадим 6 листов в рабочей книге Excel — **prod_pit**, **Storeage**, **shv_izd**, **aud_vid_texn**, **mebel**, **kanc_tov**.
- На листе **Storeage** разместим служебную информацию для работы программы (количество видов товаров, названия видов товаров (продукты питания, швейные изделия, аудиовидеотехника, мебель, канцелярские товары), количество на складе товаров каждого вида, количество кладовщиков, фамилии кладовщиков).
- Остальные пять листов будут хранить непосредственные данные о товарах на складе.
- Для каждого вида товара хранится следующая информация: название товара, вес/количество товара, дата поступления на склад, дата отгрузки товара со склада, кладовщик, который принял товар на склад, кладовщик, который отпустил

товар со склада, стоимость хранения товара, подтверждение оплаты (да, нет).

Далее приступаем к разработке основной формы в нашем приложении, которое поддерживает работу склада.

1. Перейдите в окно редактора кода VBA.
2. Добавьте пользовательскую форму, например, командой **Insert | UserForm**. Свойство `Name` для данной формы — `MainForm`.
3. Поместите на форму необходимые элементы управления и установите им соответствующие свойства так (табл. 4.47), чтобы они приняли вид, аналогичный рис. 4.53.
4. Напишите в модуле формы код, который будет поддерживать обработку событий для имеющихся элементов управления: из раскрывающегося поля со списком вы должны выбирать вид товара; соответствующие товары, которые расположены на разных рабочих листах Excel, будут отображаться в расположенном ниже списке; кнопки **Принять товар** и **Сдать товар** должны открывать формы, которые будут поддерживать действия, связанные с принятием и отгрузкой товаров.

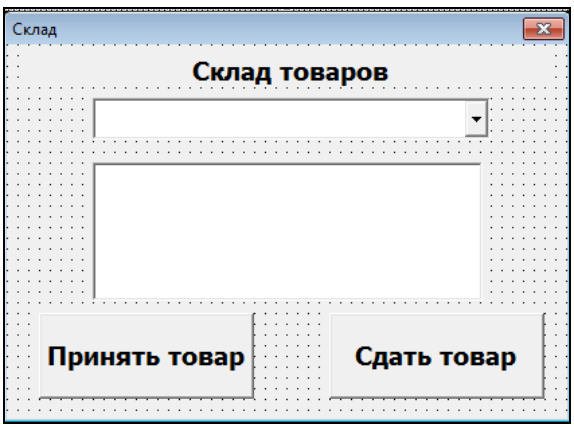


Рис. 4.53. Разработка основной формы `MainForm` для модели склада

Таблица 4.47. *Значения свойств для формы* `MainForm` *и ее элементов управления, установленные в окне* ***Properties***

Объект	Свойство	Значение
Форма	`Name`	`MainForm`
	`Caption`	`Склад`
Надпись	`Name`	`Label1`
	`Caption`	`Склад товаров`
Поле со списком	`Name`	`ComboBox1`
	`Caption`	`fmStyleDropDownList`

Таблица 4.47 *(окончание)*

Объект	Свойство	Значение
Кнопка	Name	CommandButton1
	Caption	Принять товар
Кнопка	Name	CommandButton2
	Caption	Сдать товар
Список	Name	ListBox1
	BoundColumn	1
	ColumnCount	1
	ColumnHeads	False

Следующий этап — разработка формы для приема товара (рис. 4.54) и формы для отгрузки товара (рис. 4.55). В данном случае наши действия аналогичны описанным при разработке основной формы. Обратите внимание на установку соответствующих свойств для элементов управления и написание соответствующего кода.

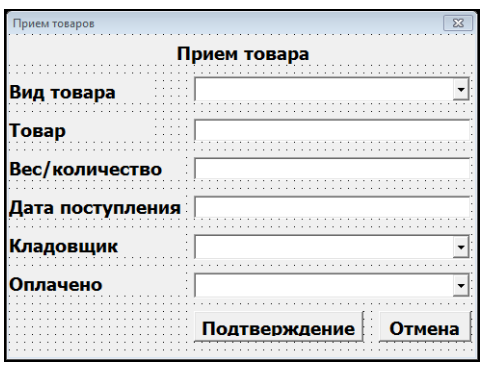


Рис. 4.54. Разразработка формы приема товара `UserForm1` для модели склада

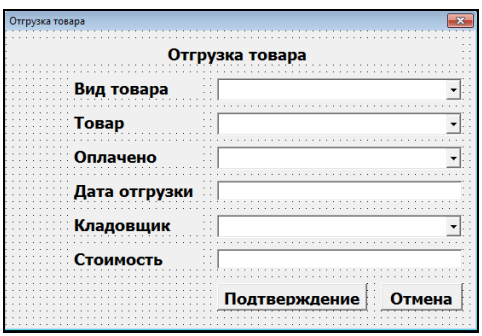


Рис. 4.55. Разработка формы отгрузки товара `UserForm2` для модели склада

И в завершении разработки нашего приложения разместите на рабочем листе **Storeage** кнопку, которая будет открывать основную форму `MainForm`. Позаботьтесь также о том, чтобы в модуль рабочего листа был введен соответствующий код, обрабатывающий нажатие кнопки.

Примечание

Размещенный на диске пример **56-Модель склада** содержит все необходимые комментарии для размещенных в соответствующих модулях листингах для приведенной модели склада.

Наши итоги

Как вы убедились, разработка полноценных приложений в Microsoft Office Excel 2010 невозможна без использования элементов управления и пользовательских форм. На подробных примерах вы изучили основные приемы работы с каждым элементом управления, а также основные варианты их применения. Все изученное вами, несомненно, позволит вам создавать:

- удобный пользовательский интерфейс на рабочем листе с применением элементов управления панели **Элементы ActiveX**;
- формы в редакторе VBA с использованием подходящих элементов управления панели элементов **Toolbox;**
- приложения, которые ограничивают действия нежелательных пользователей;
- разнообразные проекты, охватывающие обработку данных различного типа;
- завершенные бизнес-приложения, предназначенные только для решения конкретной задачи предметной области.

Глава 5

Настройка ленты — это так просто!

В Microsoft Office Excel 2010 часть интерфейса, на котором сосредоточены различные команды, занимает *лента* (ribbon). Понятно, что при разработке пользовательских приложений одним из важных моментов является создание собственного интерфейса, в частности, настройка ленты: часть вкладок, возможно, вы удалите, возможно, вам необходимо создать собственные вкладки, а, может быть, некоторые команды вы разместите на ленте, но именно в том месте, которое вам покажется более удобным для вызова этих команд.

Итак, по сравнению с предыдущей версией Microsoft Office Excel 2007, когда ленту можно было настроить только программным способом, используя язык XML, Microsoft Office Excel 2010 предлагает пользователю два варианта: либо вы настраиваете пользовательский интерфейс имеющимися средствами настройки ленты, либо делаете это, используя язык XML и процедуры VBA.

В этой главе мы рассмотрим оба варианта настройки интерфейса пользователя, включая настройку панели быстрого доступа (Customize Quick Access Toolbar).

> ***Примечание***
>
> Файлы рассмотренных в данной главе примеров можно найти в папке Glava_5 на компакт-диске.

Как настроить панель быстрого доступа?

Достаточно часто панель быстрого доступа является более удобной при выполнении расчетов и оформлении рабочих книг Microsoft Office Excel 2010. Изначально на ней располагаются следующие стандартные кнопки: **Сохранить**, **Отменить ввод**, **Повторить ввод** и **Настройка панели быстрого доступа**. Если вам приходится обращаться к каким-то командам гораздо чаще, чем к другим, вынесите необходимую кнопку на панель быстрого доступа (рис. 5.1), щелкнув по кнопке **Настройка панели быстрого доступа** и выбрав либо команду из имеющегося списка, либо команду **Другие команды**.

В последнем случае открывается окно **Параметры Excel** в категории **Панель быстрого доступа**, где можно выбрать необходимые кнопки и добавить их на панель, используя возможности данного окна (рис. 5.2).

С другой стороны, открыть окно **Параметры Excel** можно и следующим образом: щелкните по вкладке **Файл** окна Microsoft Office Excel 2010 и выберите команду **Параметры**. В открывшемся окне **Параметры Excel** перейдите в области переходов к категории **Панель быстрого доступа**.

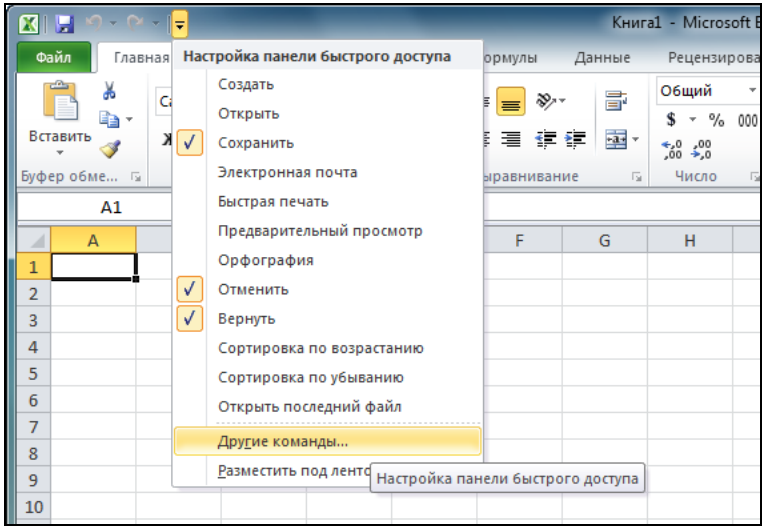


Рис. 5.1. Меню кнопки **Настройка панели быстрого доступа**

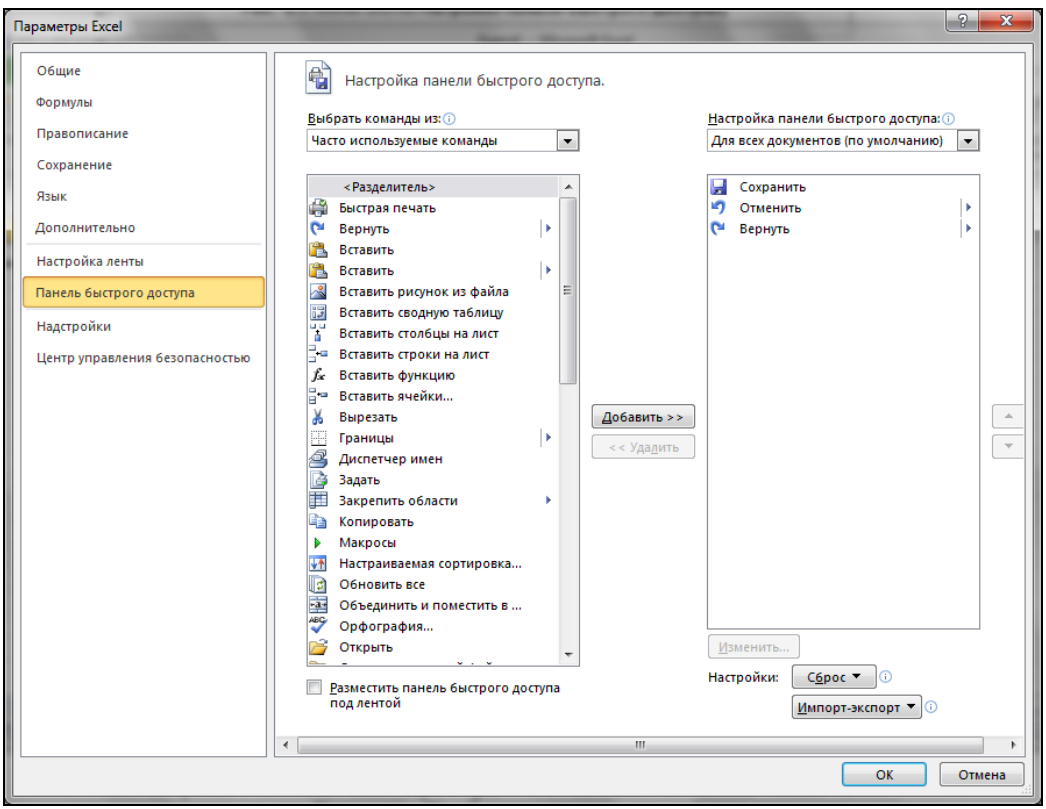


Рис. 5.2. Окно **Параметры Excel**, категория **Панель быстрого доступа**

Примечание

Еще один способ открыть окно **Параметры Excel** в категории **Панель быстрого доступа** таков: вызовите контекстное меню для панели быстрого доступа и выберите из него команду **Настройка панели быстрого доступа**.

Теперь вы можете добавить на панель быстрого доступа другие кнопки, используя соответствующие поля со списками и управляющие кнопки: выберите команду из левого списка и добавьте ее в правый список, нажав кнопку **Добавить**, расположенную между двумя списками. С другой стороны, для удаления кнопки с панели быстрого доступа, выделите ее в правом столбце и воспользуйтесь кнопкой **Удалить** (также расположена между списками). Кнопки со стрелками, расположенные справа от правого столбца, позволят вам настроить порядок следования кнопок на панели быстрого доступа.

По умолчанию в левом окне отображаются лишь часто используемые команды. Для изменения списка команд укажите в поле **Выбрать команды из** с помощью выпадающего списка нужный объект: все команды, все команды на ленте, макросы, команды из меню **Office** (меню, открывающееся при нажатии кнопки **Microsoft Office**), любая вкладка ленты.

Выбирая необходимую команду из списка **Настройка панели быстрого доступа**, вы можете указать, где будут использоваться добавленные вами кнопки: в данной рабочей книге или же — для всех рабочих книг Excel.

Кроме добавления команд, в окне **Параметры Excel** в категории **Панель быстрого доступа** вы можете также указать опцию для размещения панели быстрого доступа под лентой: установите соответствующий флажок в нижней части окна.

Группа **Настройки** поможет вам произвести сброс всех настроек панели быстрого доступа или осуществить соответствующий импорт/экспорт настроек.

Совет

Для добавления на панель быстрого доступа любой команды, находящейся на ленте, используйте контекстное меню: щелкните по требуемой кнопке ленты правой кнопкой мыши и выберите из контекстного меню команду **Добавить на панель быстрого доступа** (рис. 5.3). Команда **Настройка панели быстрого доступа** вызванного контекстного меню позволяет также открыть категорию **Панель быстрого доступа** окна **Параметры Excel**. Для удаления кнопки с панели быстрого доступа вызовите ее контекстное меню и воспользуйтесь командой **Удалить с панели быстрого доступа**.

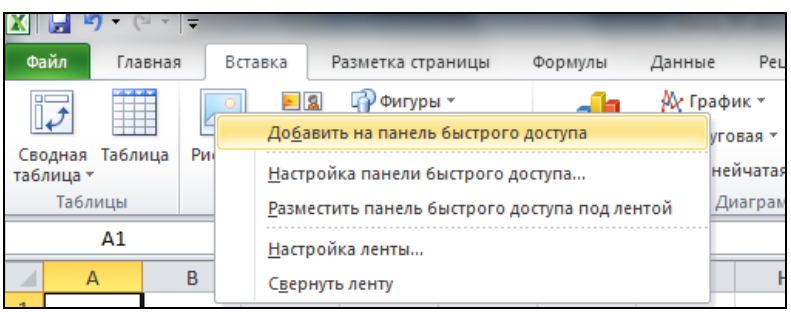


Рис. 5.3. Контекстное меню для кнопки ленты

Записываем макрос и назначаем его кнопке

С другой стороны, если вы разрабатываете свое приложение, то вам, конечно же, необходимо создавать свои кнопки, которые выполняют действия, записанные вами с помощью макроса или же определенные процедурой языка VBA.

Рассмотри пример, демонстрирующий запись макроса и назначение его кнопке, которая будет размещена на панели быстрого доступа.

Пусть нам необходимо создать макрос, который выделяет на активном рабочем листе книги Excel диапазон **A1:K20** и устанавливает для этого диапазона следующие параметры: цвет заливки — желтый, границы диапазона — все границы; вид шрифта — Bookman Old Style, размер — 14 пт; цвет шрифта — красный; начертание — полужирный. После выполнения данных действий указатель устанавливается в ячейку **A1**.

Итак, откройте рабочую книгу Microsoft Office Excel 2010 и убедитесь, что указатель стоит в ячейке **A1**.

1. Перейдите на вкладку **Разработчик** ленты и в группе **Код** щелкните по кнопке **Запись макроса**.
2. В открывшемся окне **Запись макроса** установите необходимые параметры записываемой процедуры (рис. 5.4). Помните, что в имени макроса не должно быть пробелов.
3. В режиме записи макроса (рис. 5.5) выполните необходимые действия в такой последовательности:
   - перейдите на вкладку **Главная** ленты и выделите диапазон ячеек **A1:K20**;
   - используя группу **Шрифт** вкладки **Главная** ленты с помощью соответствующих инструментов установите: цвет заливки — желтый, границы диапазона — все границы; вид шрифта — Bookman Old Style, размер — 14 пт; цвет шрифта — красный; начертание — полужирный;
   - установите указатель в ячейку **A1**.

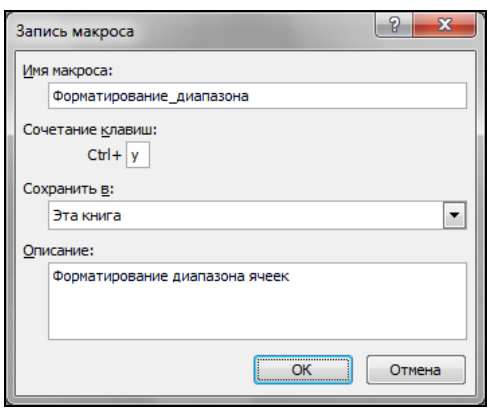


Рис. 5.4. Окно **Запись макроса**

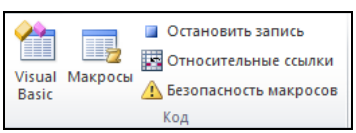


Рис. 5.5. Группа **Код** на вкладке **Разработчик** в режиме записи макроса

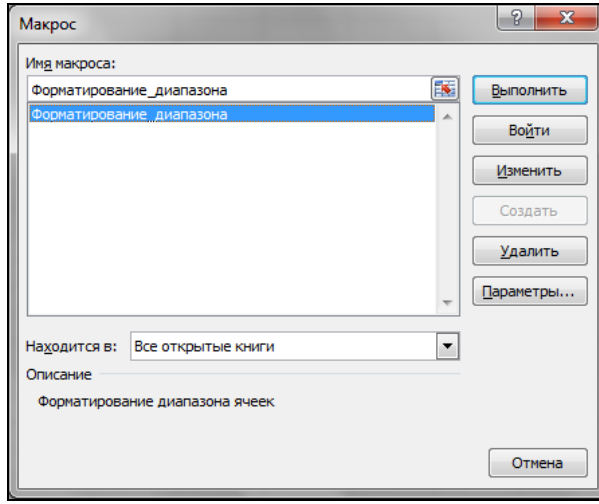


Рис. 5.6. Окно **Макрос**

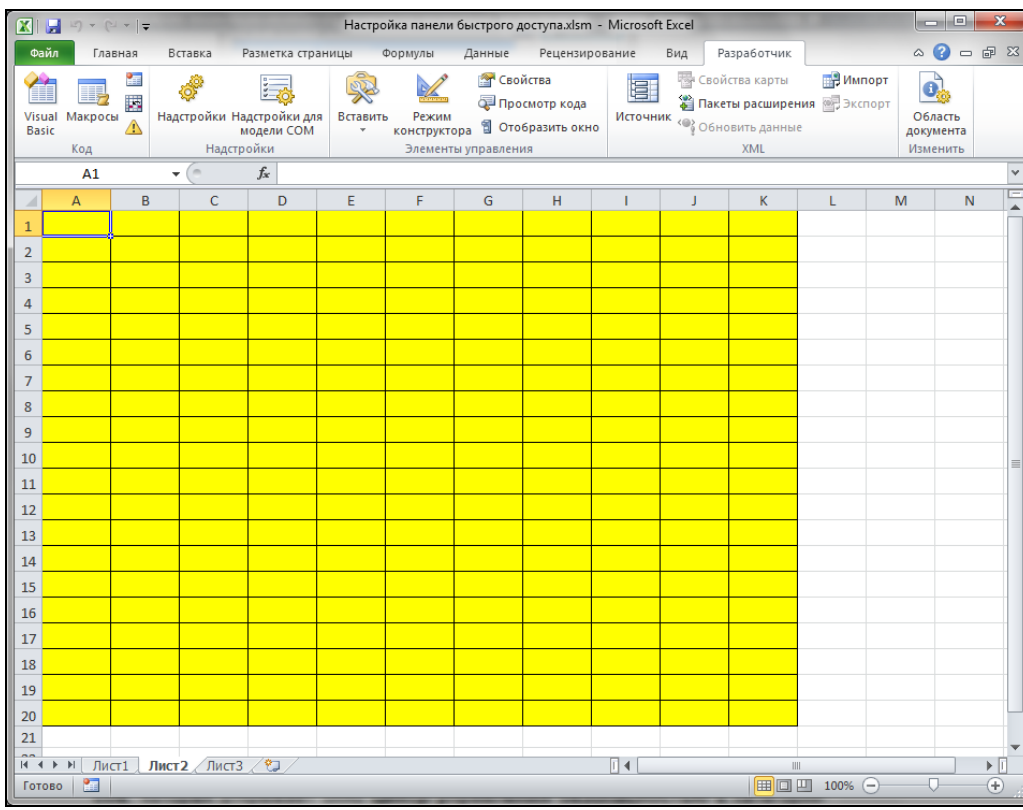


Рис. 5.7. Результат выполнения макроса

4. Нажмите кнопку **Остановить запись**, расположенную в группе **Код** на вкладке **Разработчик** для остановки записи макроса.
5. Сохраните вашу рабочую книгу под именем *Настройка панели быстрого доступа* с поддержкой макросов (см. файл *1-Настройка панели быстрого доступа.xlsm* на компакт-диске).
6. Теперь перейдите, например, на **Лист2** в вашей рабочей книге.
7. Щелкните по кнопке **Макросы** в группе **Код** на вкладке **Разработчик** ленты.
8. В открывшемся окне **Макрос** (рис. 5.6) укажите имя созданного макроса и нажмите кнопку **Выполнить**. Убедитесь, что в вашей рабочей книге данный макрос произвел все необходимые изменения (рис. 5.7).

Примечание

На вкладке **Разработчик** в группе **Код** находится кнопка **Безопасность макросов**, которая открывает окно **Центр управления безопасностью** в категории **Параметры макросов** (рис. 5.8). Вы всегда можете выбрать требуемый параметр, чтобы предотвратить нежелательное выполнение вредоносного кода, который может содержаться в макросах, полученных из неизвестных источников.

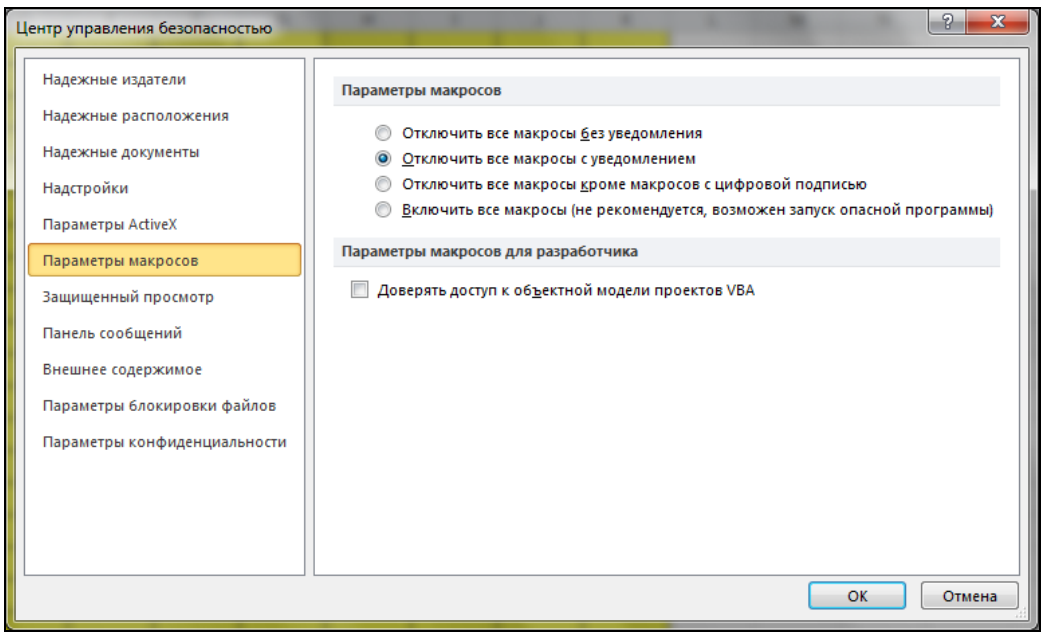


Рис. 5.8. Окно **Центр управления безопасностью** в категории **Параметры макросов**

Ну, а теперь назначим наш созданный макрос кнопке на панели быстрого доступа.

1. Щелкните правой кнопкой мыши по панели быстрого доступа и выберите команду **Настройка панели быстрого доступа**.
2. В открывшемся окне **Параметры Excel** в области переходов категории **Панель быстрого доступа** выберите в поле со списком **Выбрать команды из** объект **Макросы** (рис. 5.9).

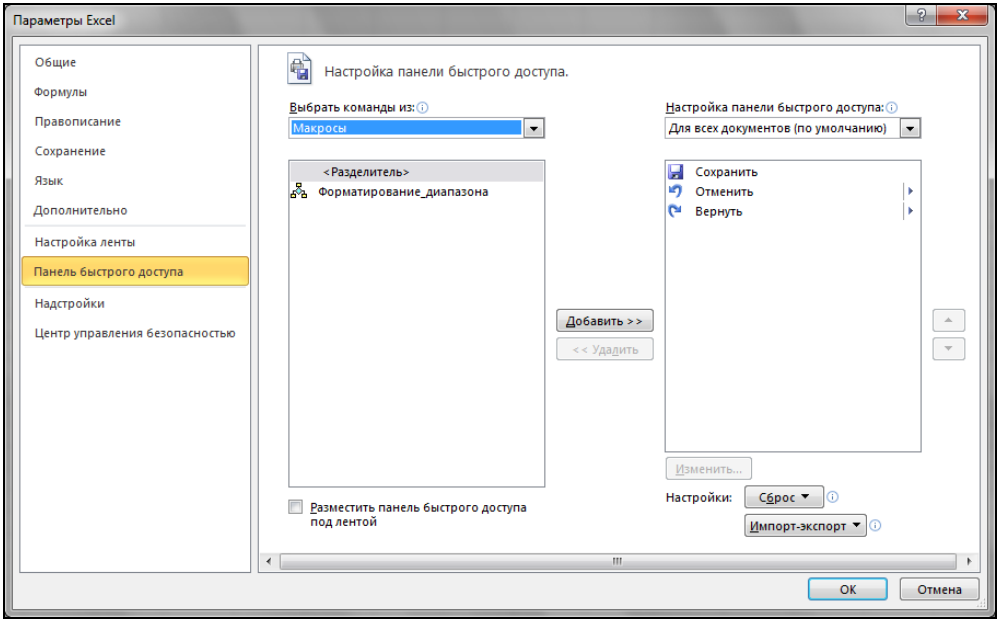


Рис. 5.9. Выбор объекта **Макросы** в окне **Параметры Excel** категории **Панель быстрого доступа**

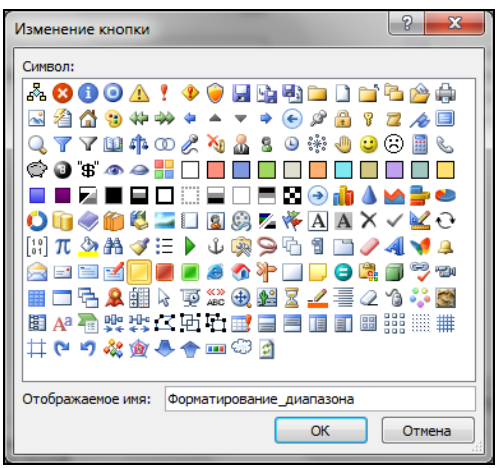


Рис. 5.10. Окно **Изменение кнопки**

3. Выберите в левом столбце созданный вами макрос и с помощью кнопки **Добавить** перенесите его в правый столбец. Обратите внимание на то, что внизу правого столбца стала доступной кнопка **Изменить**: она предназначена для назначения кнопки соответствующему макросу. Нажмите кнопку **Изменить**.
4. В открывшемся окне **Изменение кнопки** (рис. 5.10) укажите мышью символ для кнопки и введите в поле **Отображаемое имя** необходимое имя для макроса, которое будет всплывающей подсказкой на панели быстрого доступа. Нажмите кнопку **OK**.

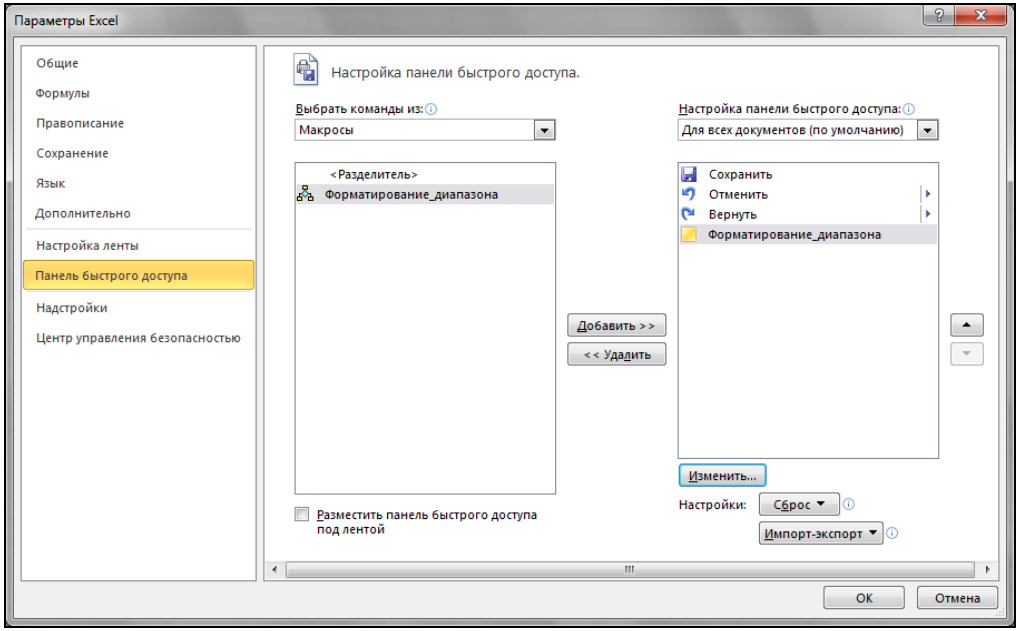


Рис. 5.11. Кнопка и новое название для макроса в окне **Параметры Excel**

Рис. 5.12. Кнопка макроса на панели быстрого доступа

5. В окне **Параметры Excel** кнопка и новое название для макроса отображаются в правом столбце (рис. 5.11). Нажмите кнопку **OK**.
6. Убедитесь, что кнопка с записанным макросом появилась на панели быстрого доступа (рис. 5.12) и ее нажатие приводит к выполнению записанных вами действий.

Назначаем кнопкам процедуры VBA

А теперь рассмотрим пример, позволяющий запускать с панели быстрого доступа вашу процедуру, записанную на языке VBA.

Отметим, что последовательность действий по подключению процедуры VBA выбранной кнопке аналогична той, что мы делали при назначении кнопке макроса.

Пусть, например, нам надо добавить четыре кнопки на панель быстрого доступа, которые соответственно будут производить следующие действия: определять попадание в круг заданного числа точек с центром в начале координат, рассчитывать комиссионные служащим по определенному правилу, заменять значения ячеек из выделенного диапазона по определенному правилу и производить форматирование чисел. Итак, рассмотрим подробнее условия и процедуры на языке VBA, которые реализуют требуемые действия (см. файл *2-Процедуры VBA на Панели быстрого доступа.xlsm* на компакт-диске).

Процедура на языке VBA, определяющая попадание заданного числа точек в круг заданного числа точек с центром в начале координат, приведена в стандартном модуле **Module1** (файл *2-Процедуры VBA на Панели быстрого доступа.xlsm* на компакт-диске).

Теперь приведем пример расчета комиссионных служащим по следующему правилу:

- если продукции продано не меньше, чем на 1 000 000 у. е., то комиссионные составляют 4% от стоимости реализованной продукции;
- если продукции продано меньше, чем на 1 000 000 у. е., то комиссионные составляют 2% от стоимости реализованной продукции;
- если стаж работы в фирме не менее 5 лет, то выплачивается доплата в размере 1,5% от стоимости реализованной продукции.

Соответствующая запись процедуры на языке VBA приведена в стандартном модуле **Module2** (файл *2-Процедуры VBA на Панели быстрого доступа.xlsm* на компакт-диске).

Стандартный модуль **Module3** (файл *2-Процедуры VBA на Панели быстрого доступа.xlsm* на компакт-диске) содержит процедуру замены значений ячеек из выделенного диапазона по следующему правилу: положительные числа заменяются "+", отрицательные — "–", нули — "нуль".

В стандартном модуле **Module4** (файл *2-Процедуры VBA на Панели быстрого доступа.xlsm* на компакт-диске) демонстрируется форматирование чисел, находящихся в выделенном диапазоне рабочего листа по следующему правилу:

- от 0 до 1000 — числа будут отформатированы красным цветом;
- от 1000 до 10 000 — числа будут отформатированы зеленым цветом;
- свыше 10 000 — числа будут отформатированы черным цветом;
- формат данных для положительных чисел: `#000,000`;
- ноль прописывается фиолетовым цветом — "Нуль!!!";
- отрицательные числа выделяются синим цветом с форматом: `#000,000`.

Итак, для того чтобы добавить кнопки на панель быстрого доступа, выполняющие действия, прописанные выше, выполните следующие шаги:

1. Перейдите в окна редактора VBA, выбрав на вкладке **Разработчик** ленты группу **Код** и щелкнув по кнопке **Visual Basic**.
2. В окне редактора VBA добавьте последовательно 4 модуля, используя команды **Insert | Module**, в каждый из которых вставьте соответственно код требуемой процедуры VBA (см. см. стандартные модули файла *2-Процедуры VBA на Панели быстрого доступа.xlsm* на компакт-диске).
3. Перейдите в окно рабочей книги Microsoft Excel.
4. Щелкните правой кнопкой мыши по панели быстрого доступа и выберите команду **Настройка панели быстрого доступа**.
5. В открывшемся окне **Параметры Excel** в категории **Панель быстрого доступа** выберите в поле со списком **Выбрать команды из** объект **Макросы** (см. рис. 5.9).
6. Последовательно выбирайте в левом столбце созданную вами процедуру и с помощью кнопки **Добавить** переносите ее в правый столбец. Используя кнопку **Изменить**, смените значок для каждой кнопки.
7. После того как будут добавлены все ваши созданные процедуры в правый столбец окна **Параметры Excel** категории **Панель быстрого доступа**, нажмите кнопку **OK**.

Рис. 5.13. Кнопки процедур VBA на панели быстрого доступа

8. Убедитесь, что требуемые кнопки появились на панели быстрого доступа (рис. 5.13) и их нажатие приводит к выполнению записанных вами действий.

Очень быстро настраиваем ленту

В Microsoft Office Excel 2010 можно также настраивать вкладки, группы и отдельные команды ленты. По умолчанию в Excel 2010 файлы рабочих книг сохраняются в формате Microsoft Office Open XML и имеют расширение xlsx (или xlsm для книг с поддержкой макросов). Это важно помнить, т. к. для настройки ленты окна Microsoft Excel 2010 используются знания языка XML.

Отметим, что в Excel 2010 появился удобный интерфейс, который позволяет пользователю производить настройку ленты.

Далее в этой главе мы разберем возможности программной настройки ленты. Однако сначала укажем возможности пользовательского интерфейса: вероятно, при разработке вашего интерфейса пользователя определенную часть вы будете делать именно таким образом, а необходимые процедуры писать на языке VBA.

Итак, для быстрой настройки ленты выполните следующие действия.

1. Перейдите на вкладку **Файл** и нажмите кнопку **Параметры**.
2. В открывшемся окне **Параметры Excel** выберите в области переходов категорию **Настройка ленты** (рис. 5.14).
3. С использованием инструментов, которые предоставляет пользовательский интерфейс окна **Параметры Excel**, вы можете: создать новую вкладку, удалить или модифицировать имеющуюся, включая добавление/удаление групп и команд.

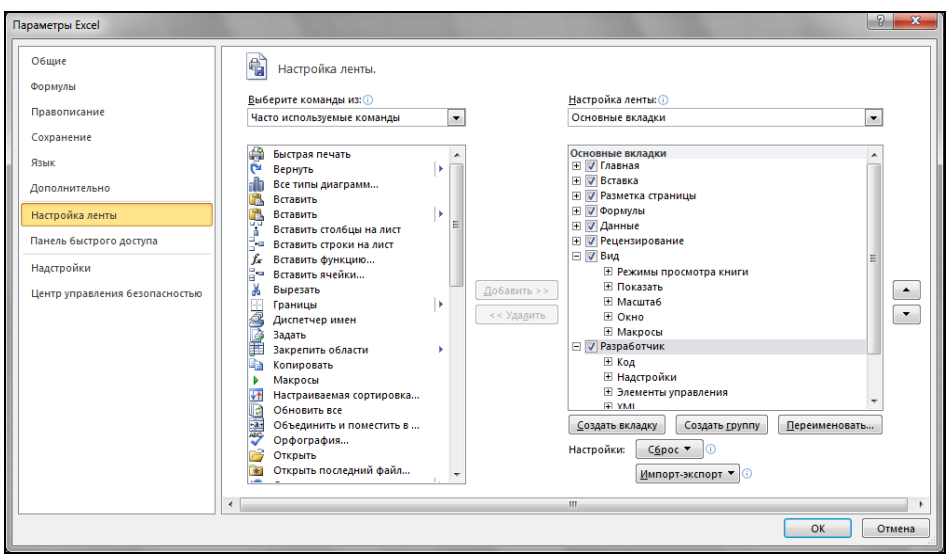


Рис. 5.14. Окно **Параметры Excel** в области переходов категории **Настройка ленты**

Настраиваем ленту с использованием формата Microsoft Office Open XML

Формат Microsoft Office Open XML в рабочих книгах Microsoft Office Excel 2010

В Microsoft Office Excel 2010 поддерживается формат файлов Microsoft Office Open XML, введенный в версии Excel 2007. Как вы уже знаете, расширение файлов, которыми вы обычно пользуетесь, — это xlsx (формат файла, используемый по умолчанию) или xlsm (для книг с поддержкой макросов).

Файлы, сохраненные в таких форматах, реально состоят из нескольких XML-файлов *(частей)*, упакованных в один ZIP-архив, расширение zip которого заменяется расширением xlsx или xlsm. Вы должны знать, что архив Office Open XML должен обязательно соответствовать требованиям Open Packaging Convention, в частности он должен включать файл *(часть)* с названием [Content_Types].xml и файл связей *(relationships)* .rels в папке _rels. Иногда в архив могут быть добавлены файлы других типов, например BMP, AVI, PDF (в дальнейшем вы увидите, для чего это делается). Для того чтобы просмотреть структуру текущей рабочей книги, выполните следующие действия.

1. Запустите Microsoft Office Excel 2010 и сохраните рабочую книгу в формате по умолчанию, например, First.xlsx. Закройте сохраненную рабочую книгу.
2. Перейдите, например, в Проводнике к сохраненной книге и измените ее расширение на zip.
3. Войдите в содержимое архива и просмотрите его структуру (рис. 5.15).

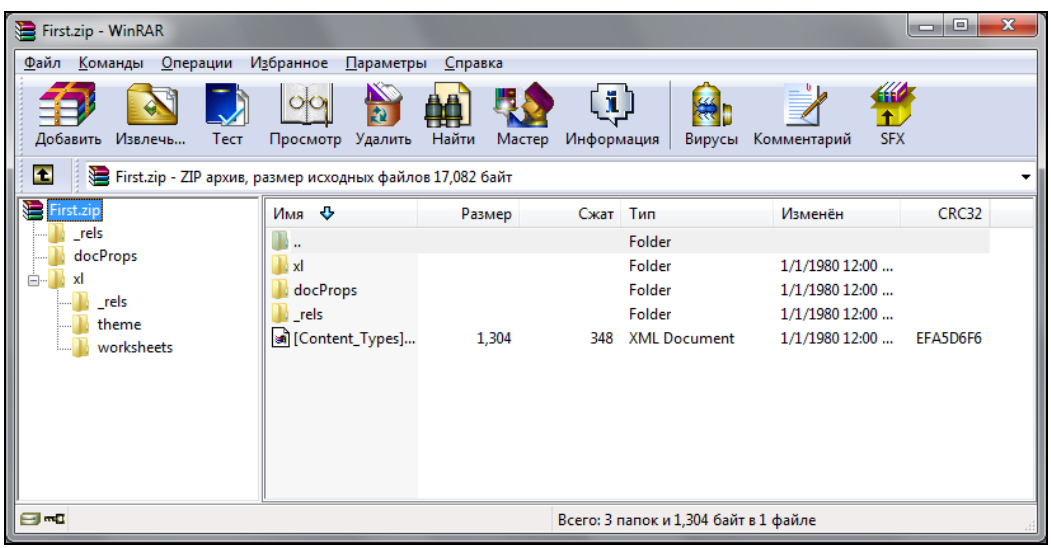


Рис. 5.15. Внутренняя структура файла рабочей книги First.xlsx (в архиве)

Как вы уже убедились, Excel 2010 предлагает простой пользовательский интерфейс для настройки ленты. Однако существует также возможность и программным способом произвести настройку ленты с учетом требований вашего приложения: добавить, модифицировать или удалить вкладки, группы и отдельные команды.

В версии Excel 2010 существует возможность настраивать ленту, используя прямое редактирование XML-файлов рабочей книги Excel и процедуры VBA. Как правило, для настройки ленты, необходимо создать XML-файл, задающий конкретные параметры настройки, поместить его внутрь как часть xlsx-архива (или xlsm-архива) и добавить в текст базового XML-файла _rels/.rel ссылку с указанием расположения добавленного файла (части) в архиве.

В общем случае можно привести следующие рекомендации по настройке ленты.

1. Продумайте, какой именно ленту вы хотите видеть в своем приложении: какие у вас будут вкладки — какие из имеющихся хотите скрыть, какие новые вкладки необходимо добавить; все ли группы команд на вкладках необходимы в приложении, возможно, вам следует добавить собственную группу на стандартной вкладке; какие элементы управления и команды необходимы вам в ваших создаваемых группах и вкладках.
2. Определите, какие стандартные действия будут выполнять элементы управления на ваших вкладках. Возможно, вам необходимо также написать процедуры на языке VBA, выполняющие действия, отличные от стандартных, или производящие некоторые специфические расчеты. Заранее продумайте логику процедур и их реализацию на языке VBA.
3. Опишите пользовательский интерфейс, т. е. вашу ленту, с помощью языка XML, например, используя Блокнот и сохраняя этот файл, соответственно, с именем и расширением: customUI.xml в отдельной папке CustomUI.
4. Откройте рабочую книгу Microsoft Excel 2010 и в окне редактора VBA введите (при необходимости) на листе модуля требуемый код для выполнения действий тех элементов управления, которые вы размещаете на вкладках ленты. Сохраните ваш файл с расширением xlsm. Если вы добавляете стандартные элементы управления и вам не требуется писать процедуры на VBA, достаточно просто сохранить рабочую книгу с расширением xlsx.
5. Измените расширение вашей рабочей книги на zip и добавьте в структуру архива папку CustomUI.
6. Откройте в архиве базовый XML-файл (часть) _rels/.rel и добавьте перед последним тегом `</Relationships>` ссылку с указанием расположения файла (части) CustomUI.xml в архиве, т. е. такой тег, как в листинге 5.1.

Листинг 5.1. Тег, содержащий ссылку с указанием расположения CustomUI.xml-файла (части) в архиве

```
<Relationship Id="customUI_Relation"
Type="http://schemas.microsoft.com/office/2006/relationships/ui/extensibili
ty" Target="customUI/customUI.xml" />
</Relationships>
```

7. Сохраните изменения в файле и а архиве. Закройте архив.
8. Замените расширение архива zip на первоначальное, т. е. на xlsx или xlsm.

9. Откройте файл измененной рабочей книги и убедитесь в изменениях, которые произошли на ленте.

> ***Примечание***
>
> Для редактирования xml-файлов (частей) непосредственно внутри исходного xlsx-файла можно использовать редактор Custom UI Editor (который можно найти по адресу **http://openxmldeveloper.org/articles/customuieditor.aspx**).

> ***Примечание***
>
> При настройке ленты скрывать можно стандартные вкладки и группы команд. Скрыть конкретный элемент управления, расположенный в некоторой группе команд, нельзя.

Чтобы правильно писать в файле customUI.xml теги для настройки ленты на языке XML (RibbonX-код), окинем быстрым взглядом ленту, ее структуру и объекты, которые на ней имеются.

Так, основными вкладками ленты являются **Главная** (Home), **Вставка** (Insert), **Разметка страницы** (PageLayoutExcel), **Формулы** (Formulas), **Данные** (Data), **Рецензирование** (Review), **Вид** (View), **Разработчик** (Developer) и **Надстройки** (Add-Ins). На каждой вкладке имеются группы команд, на которых собраны (сгруппированы) соответствующие элементы управления и команды. Все эти объекты входят в стандартную библиотеку Microsoft Office.

Для работы со стандартными (имеющимися) вкладками достаточно указать соответствующую вкладку и описать ее свойства. Аналогично поступаем и с группой команд, которые хотим добавить на ленту.

Следует заметить еще раз, что ваш подход к настройке ленты полностью зависит от того, что вы хотите оставить, а что добавить в ваше приложение. Далее мы рассмотрим несколько вариантов ее настройки.

Настройка ленты прямым редактированием XML-файлов рабочей книги Excel

Приведем пример настройки ленты, используя для ее формирования лишь те элементы управления, которые имеются в библиотеке Microsoft Office, и непосредственное редактирование XML-файла.

Так, пусть нам необходимо в своем приложении иметь измененную ленту, которая учитывает следующие требования. Лента должна скрывать вкладки **Рецензирование** (Review), **Вид** (View), **Разработчик** (Developer); скрывать группы команд **Стили** (Styles) на вкладке **Главная** (Home), **Число** (Number) на вкладке **Главная** (Home), **Параметры страницы** (PageSetup) на вкладке **Разметка страницы** (PageLayoutExcel), **Упорядочить** (Arrange) на вкладке **Разметка страницы** (PageLayoutExcel), **Получить внешние данные** (GetExternalData) на вкладке **Данные** (Data). Кроме того, на ленту необходимо добавить одну вкладку — **Новая**, на которой будут две группы команд — **Новая группа 1** и **Новая группа 2**. В группе **Новая группа 1** разместить элементы управления **Копировать** (Copy) и **Вставить** (Paste). В группе **Новая группа 2** разместить элементы управления для форматирования текста — **Жирный** (Bold), **Курсив** (Italic), **Подчеркнутый** (Underline),

Двойное подчеркивание (UnderlineDouble), **Подчеркивание слов** (UnderlineWords), а также элементы **Все границы** (BorderOutside) и **Автосумма** (AutoSum).

1. Запустите Блокнот (Notepad) и опишите ленту, с использованием языка XML (листинг 5.2, см. также файл *customUI.xml* в папке *CustomUI*, расположенной в папке *1-Пример настройки Ленты прямым редактированием XML-файла*, на компакт-диске).

Листинг 5.2. Описание ленты с использованием языка XML

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon>
    <tabs>
      <tab idMso="TabReview"  visible="false" />
      <tab idMso="TabView"  visible="false" />
      <tab idMso="TabDeveloper"  visible="false" />
      <tab idMso="TabHome">
        <group idMso="GroupStyles" visible="false" />
        <group idMso="GroupNumber" visible="false" />
      </tab>
      <tab idMso="TabPageLayoutExcel">
        <group idMso="GroupPageSetup" visible="false" />
        <group idMso="GroupArrange" visible="false" />
      </tab>
      <tab idMso="TabData">
        <group idMso="GroupGetExternalData" visible="false" />
      </tab>
      <tab id="CustomTab" label="Новая" visible="true">
        <group id="Group1" label="Новая группа 1" >
          <control idMso="Copy" label="Копировать" enabled="true" />
          <control idMso="Paste" label="Вставить" enabled="true" />
        </group>
        <group id="Group2" label="Новая группа 2" >
          <control idMso="Bold" label="Полужирный" enabled="true" />
          <control idMso="Italic" label="Курсив" enabled="true" />
          <control idMso="Underline" label="Подчеркнутый" enabled="true" />
          <control idMso="UnderlineDouble" label="Двойное подчеркивание"
                   enabled="true" />
          <control idMso="UnderlineWords" label="Подчеркивание слов"
                   enabled="true" />
          <control idMso="AutoSum"  label="Автосумма" enabled="true" />
          <control idMso="BorderOutside"  label="Все границы"
                   enabled="true" />
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

2. Сохраните этот файл соответственно с именем и расширением: customUI.xml в отдельной папке CustomUI.

> ***Совет***
>
> При сохранении в блокноте описания ленты проверьте, чтобы для кодировки символов был выбран параметр **UTF-8** (рис. 5.16).

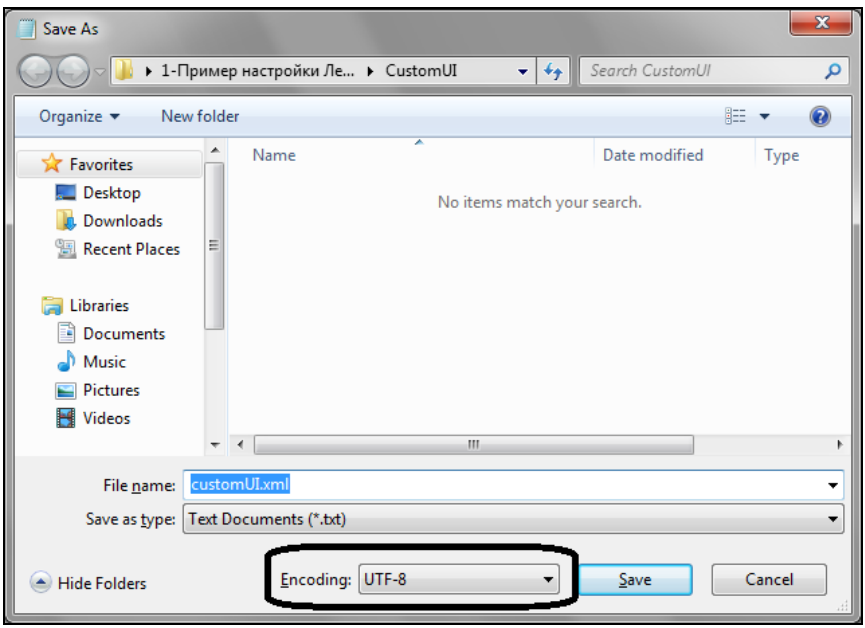


Рис. 5.16. Сохранение файла описания ленты в Блокноте

3. Запустите Microsoft Excel 2010 и сохраните рабочую книгу соответственно с именем и расширением *1-Пример настройки Ленты.xlsx*. Закройте рабочую книгу.
4. Измените расширение рабочей книги *1-Пример настройки Ленты.xlsx* на zip и добавьте, например, перетаскиваем в структуру архива папку CustomUI.
5. Откройте в архиве базовый XML-файл (часть) _rels/.rel и добавьте перед последним тегом `</Relationships>` ссылку с указанием расположения файла (части) CustomUI.xml в архиве (см. листинг 5.1). Таким образом, содержимое rel-файла соответствует листингу 5.3.

Листинг 5.3. Содержимое XML-файла ссылок _rels/.rel (в архиве)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships
xmlns="http://schemas.openxmlformats.org/package/2006/relationships"><Relat
ionship Id="rId3"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/e
xtended-properties" Target="docProps/app.xml"/><Relationship
Id="customUI_Relation"
Type="http://schemas.microsoft.com/office/2006/relationships/ui/extensibili
```

```
ty" Target="CustomUI/customUI.xml"/><Relationship Id="rId2"
Type="http://schemas.openxmlformats.org/package/2006/relationships/metadata
/core-properties" Target="docProps/core.xml"/><Relationship Id="rId1"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/o
fficeDocument" Target="xl/workbook.xml"/></Relationships>
```

6. Сохраните изменения в файле и в архиве. Закройте архив.
7. Замените расширение архива *1-Пример настройки Ленты.zip* на xlsx.
8. Откройте файл измененной рабочей книги и убедитесь в изменениях, которые произошли на ленте (рис. 5.17, см. также файл *1-Пример настройки Ленты.xlsx*, расположенный в папке *1-Пример настройки Ленты прямым редактированием XML-файла*, на компакт-диске).

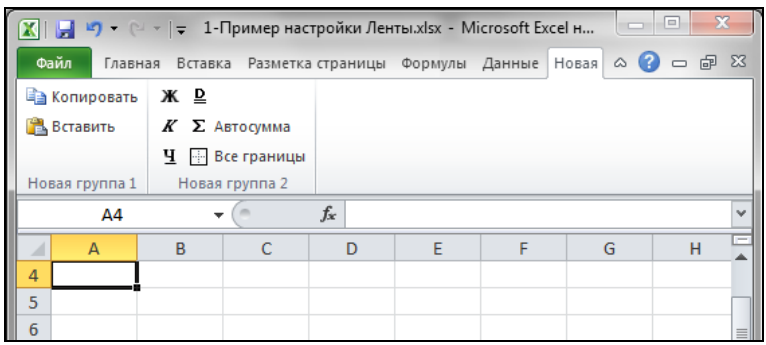


Рис. 5.17. Окно Excel с измененной лентой

Примечание

Убедитесь, что в файле *1-Пример настройки Ленты.xlsx* все добавленные элементы управления выполняют необходимые действия.

Настройка ленты с использованием XML и VBA

А сейчас мы приведем пример настройки ленты, элементы управления которой использует процедуры, написанные на языке VBA.

Как и в предыдущем примере, все элементы управления, вкладки и группы команд ленты вы описываете внутри XML-файла (RibbonX-код), а все необходимые действия — внутри процедур VBA, которые размещаете в модуле рабочей книги. Следует помнить, что аргументом процедуры VBA является элемент управления `control` как объект `IRibbonControl`. В свою очередь, параметр `onAction`, который встречается в XML-коде ленты, связывает внешний файл настройки ленты с процедурой VBA.

Рассмотрим пример, который позволяет добавить на ленту вкладку **ЛИЧНАЯ** с двумя группами (рис. 5.18, на рисунке вкладка разделена на две части только для удобства чтения). Группа **Аксессуары** содержит следующие элементы управления: надпись **Текущая дата** (рядом с ней выводится текущая дата); надпись **День недели**,

после которой выводится текущий день недели, надпись **Время открытия рабочей книги** (выводится время открытия рабочей книги). Далее находится разделитель группы, а затем следующие элементы управления: флажок **Страницы**, который позволяет разбить на страницы текущий рабочий лист; стандартный элемент управления **Открыть**, позволяющий открывать необходимые файлы рабочих книг; кнопка **Расчет**, которая проверяет попадание точек в круг с заданным радиусом в начале координат (см. листинг 5.1); кнопка **Художник**, открывающая соответствующее стандартное Windows-приложение mspaint.exe и галерею **Выбор дня недели**, которая позволяет выбрать необходимый день недели и получить подтверждение в соответствующем диалоговом окне. Группа **Мои данные** включает следующие элементы управления: надпись **Дата моего рождения**; поля со списками **День**, **Месяц**, **Год**, позволяющие выбрать необходимую дату рождения; разделитель группы; надпись **Мой рост (в см)**, после которой располагается поле, предназначенное для ввода данных о росте с целью дальнейшего расчета вашего идеального веса; галерея **Мои фотографии**, которая позволяет выбрать одну из имеющихся фотографий и получить подтверждение в соответствующем окне диалога.

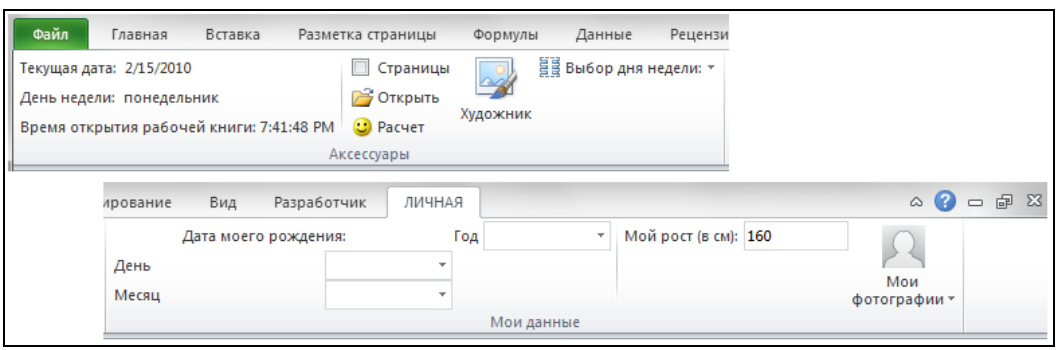


Рис. 5.18. Окно Excel с добавленной вкладкой **ЛИЧНАЯ** на ленте

Итак, приведем рекомендации по созданию вкладки **ЛИЧНАЯ** и добавлению необходимых элементов управления.

1. Запустите Блокнот (Notepad) и опишите вкладку **ЛИЧНАЯ** на ленте, с использованием языка XML (см. файл *customUI.xml* в папке *CustomUI*, расположенной в папке *2-Пример настройки Ленты с использованием XML и VBA*, на компакт-диске).
2. Сохраните этот файл соответственно с именем и расширением: customUI.xml в отдельной папке CustomUI.
3. Создайте в папке CustomUI папку images, куда поместите соответствующие графические файлы с расширением jpg. В данном случае это файлы: Lada1.jpg, Lada2.jpg, Lada3.jpg и Lada4.jpg.
4. Создайте в программе Блокнот соответствующий файл ссылок (листинг 5.4), который сохраните под именем CustomUI.xml.rels в папке _rels, входящей соответственно в папку CustomUI.

Листинг 5.4. Содержимое XML-файла ссылок _rels/.rel на графические файлы

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships
xmlns="http://schemas.openxmlformats.org/package/2006/relationships"><Relat
ionship Id="Lada1"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/i
mage" Target="images/Lada1.jpg"/> <Relationship Id="Lada2"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/i
mage" Target="images/Lada2.jpg"/><Relationship Id="Lada3"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/i
mage" Target="images/Lada3.jpg"/><Relationship Id="Lada4"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/i
mage" Target="images/Lada4.jpg"/></Relationships> </Relationships>
```

5. Запустите Microsoft Excel 2010 и создайте рабочую книгу с именем и расширением *2-Пример настройки ленты с использованием XML и VBA.xlsm* (см. также одноименный файл в папке *2-Пример настройки Ленты с использованием XML и VBA* на компакт-диске).
6. Перейдите в окно редактора VBA и создайте новый модуль, используя команду **Insert | Module**.
7. Добавьте в лист модуля соответствующие процедуры, обрабатывающие события, связанные с элементами управления вкладки **ЛИЧНАЯ** (см. файл *2-Пример настройки Ленты с использованием XML и VBA.xlsm,* расположенный в папке *2-Пример настройки Ленты с использованием XML и VBA*, на компакт-диске).
8. Закройте рабочую книгу, сохраняя в ней сделанные изменения.
9. Измените расширение рабочей книги *2-Пример настройки ленты с использованием XML и VBA.xlsm* на zip и добавьте, например, перетаскиванием в структуру архива папку CustomUI.
10. Откройте в архиве базовый XML-файл (часть) _rels/.rel и добавьте перед последним тегом </Relationships> ссылку с указанием расположения файла (части) CustomUI.xml в архиве (см. листинг 5.1).
11. Сохраните изменения в файле и в архиве. Закройте архив.
12. Замените расширение архива *2-Пример настройки ленты с использованием XML и VBA.zip* на xlsm.
13. Откройте файл измененной рабочей книги и убедитесь, что на ленте появилась новая вкладка **ЛИЧНАЯ** (см. рис. 5.18).
14. Убедитесь, что в файле *2-Пример настройки ленты с использованием XML и VBA.xlsm* все добавленные элементы управления выполняют необходимые действия.

Пример создания динамического меню ленты

Еще одним интересным примером является создание динамического меню ленты, которое изменяется при переходе на различные листы рабочей книги Excel. Настройка элемента управления `dynamicMenu`, который описывается в XML-файле, является достаточно сложной задачей. Необходимо учесть, что кроме его описания, вам требуется добавить соответствующий код меню для каждого листа в рабочую

книгу Excel, модифицировать файл CustomUI с учетом загрузки ленты через процедуруVBA и т. д.

На прилагаемом компакт-диске вы найдете пример реализации динамического меню для ленты (рис. 5.19) — файл *3-Пример настройки динамического меню ленты.xlsm* в папке *3-Динамическое меню Ленты*. Здесь же мы дадим лишь небольшие пояснения, связанные с общим подходом к описанию элемента управления `dynamicMenu`.

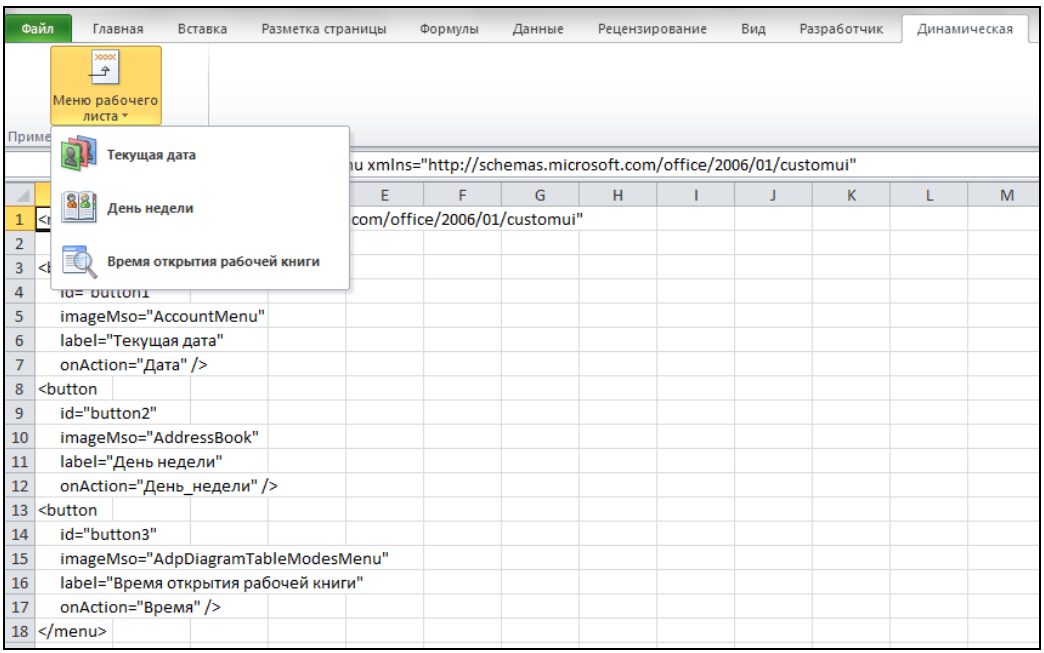


Рис. 5.19. Элемент управления `dynamicMenu` на ленте на первом листе рабочей книги

1. В Блокноте (Notepad) с использованием языка XML опишите вкладку **Динамическая**, содержащую элемент управления `dynamicMenu` (листинг 5.5). Обратите внимание на то, что тег `customUI` содержит параметр `onLoad`, который инициализируется процедурой `ribbonLoaded` на VBA (листинг 5.6). Данная процедура создает объект `MyRibbon` как `IRibbonUI`-объект и считывает файл CustomUI. Переменная `MyRibbon` объявлена как переменная уровня модуля, т. е. она является открытой (или `Public`-переменной), и доступна для всех процедур проекта VBA.

Листинг 5.5. Описание вкладки *Динамическая* ленты, включающей элемент управления `dynamicMenu`

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui"
          onLoad="ribbonLoaded">
  <ribbon>
```

```
  <tabs>
    <tab id="CustTab" label="Динамическая">
      <group id="Group1" label="Пример динамического меню">
        <dynamicMenu id="DynamicMenu" getContent="dynamicMenuContent"
                     imageMso="ControlTitle" size = "large"
                     label="Меню рабочего листа"/>
      </group>
    </tab>
  </tabs>
  </ribbon>
</customUI>
```

Листинг 5.6. Процедура `ribbonLoaded` на листе модуля VBA

```
' Объявление переменной MyRibbon на уровне модуля как IRibbonUI-объект
Public MyRibbon As IRibbonUI

Sub ribbonLoaded(ribbon As IRibbonUI)
' Процедура загрузки файла CustomUI
    Set MyRibbon = ribbon
End Sub
```

2. На листе модуля `ЭтаКнига` расположите процедуру `Workbook_SheetActivate`, вызывающую обновление динамического меню ленты при активизации рабочего листа (листинг 5.7).

Листинг 5.7. Процедура `Workbook_SheetActivate` на листе модуля `ЭтаКнига`

```
Private Sub Workbook_SheetActivate(ByVal Sh As Object)
    Call UpdateDynamicRibbon
End Sub
```

3. Соответствующая процедура модуля VBA `UpdateDynamicRibbon` (листинг 5.8) обновляет объекты ленты и включает необходимую обработку ошибок в случае их потери.

Листинг 5.8. Процедура `UpdateDynamicRibbon` на листе модуля VBA

```
Sub UpdateDynamicRibbon()
    On Error Resume Next
    MyRibbon.Invalidate
    If Err.Number <> 0 Then
       MsgBox "Потерян объект ленты. Сохраните рабочую книгу " & _
              "и выполните заново ее открытие"
    End If
End Sub
```

4. Следующая процедура `dynamicMenuContent` (листинг 5.9) модуля VBA считывает XML-код, расположенный в ячейках активного рабочего листа и хранит его в переменной `XMLcode`. Когда весь XML-код считан, он передается как аргумент типа `returnedVal`. Таким образом, у элемента управления `dynamicMenu` появляется новый код, что отражается на наборе вариантов меню текущего рабочего листа.

Листинг 5.9. Процедура `dynamicMenuContent` на листе модуля VBA

```
Sub dynamicMenuContent(control As IRibbonControl, ByRef returnedVal)
' Процедура считывания XML-кода с листов рабочей книги (текущей)
  Dim r As Long
  Dim XMLcode As String
  ' Считывание XML-кода с активного рабочего листа
  For r = 1 To Application.CountA(Range("A:A"))
    XMLcode = XMLcode & ActiveSheet.Cells(r, 1) & " "
  Next r
  returnedVal = XMLcode
End Sub
```

Примечание

В прилагаемом файле примера вы найдете необходимые дополнительные процедуры на языке VBA, а также XML-код, расположенный в ячейках столбца **A** на листах рабочей книги. Кроме того, на компакт-диске расположены также сопутствующие файлы для создания динамического меню ленты.

Дополнительные замечания по настройке ленты

Итак, после того как вы проделали предлагаемые в этой главе примеры и освоили основные приемы настройки ленты, следует привести основные замечания, которые помогут вам в дальнейшей работе.

1. Отображение ошибок, которые возникают при настройке ленты, можно произвести следующим образом. Перейдите на вкладку **Файл** ленты и выберите команду **Параметры**. В открывшемся окне **Параметры Excel** выберите слева категорию **Дополнительно**, а справа — раздел **Общие**, в котором установите флажок **Показывать ошибки интерфейса пользователя надстроек**.
2. Теги, связанные с настройкой ленты (RibbonX-код), вводятся с учетом регистра.
3. Все уникальные идентификаторы (ID) элементов управления прописываются на английском языке, поэтому модификация ленты не зависит от языковой версии Excel.
4. Все модификации ленты доступны лишь в той рабочей книге, для которой они создавались. Если же нам необходимо, чтобы некоторые элементы управления были доступны для любой рабочей книги, их следует добавить на вкладку **Надстройки**.
5. Для элементов управления, помещаемых на вкладки ленты, невозможно изменять размеры.

6. На стандартные вкладки ленты невозможно добавить или удалить элементы управления.
7. Для стандартной ленты возможно скрытие вкладок и групп команд (листинг 5.10). Если мы хотим скрыть все стандартные вкладки ленты и оставить лишь собственные, достаточно установить для элемента `ribbon` свойство `startFromScratch` равным `true` (листинг 5.11, см. также соответствующие файлы в папке 4-*Пример скрытия всех стандартных вкладок Ленты*).

Листинг 5.10. Пример скрытия стандартных вкладок и групп команд в описании ленты на языке XML

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon>
    <tabs>
     <tab idMso="TabInsert">
       <group idMso="GroupInsertIllustrations" visible="false" />
       <group idMso="GroupInsertLinks" visible="false" />
       <group idMso="GroupInsertText" visible="false" />
     </tab>
     <tab idMso="TabReview"  visible="false" />
     <tab idMso="TabView"  visible="false" />
     <tab idMso="TabData" visible="false" />
     <tab idMso="TabDeveloper"  visible="false" />
     <tab idMso="TabHome">
       <group idMso="GroupStyles" visible="false" />
       <group idMso="GroupNumber" visible="false" />
     </tab>
    </tabs>
  </ribbon>
</customUI>
```

Листинг 5.11. Пример скрытия всех стандартных вкладок и описания пользовательской вкладки в описании ленты на языке XML

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="true">
    <tabs>
     <tab id="CustomTab" label="Первая" visible="true">
      <group id="Group1" label="Группа 1" >
        <control idMso="Copy" label="Копировать" enabled="true" />
        <control idMso="Paste" label="Вставить" enabled="true" />
      </group>
      <group id="Group2" label="Группа 2" >
        <control idMso="Bold" label="Полужирный" enabled="true" />
        <control idMso="Italic" label="Курсив" enabled="true" />
```

```
        <control idMso="Underline" label="Подчеркнутый" enabled="true" />
      </group>
     </tab>
    </tabs>
  </ribbon>
</customUI>
```

8. Можно назначить собственный макрос (листинг 5.12) встроенному элементу управления (repurposing the control).

Листинг 5.12. Пример назначения собственных макросов встроенным элементам управления в описании ленты на языке XML

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
<commands>
  <command idMso="FileSave" onAction="Сохранение"/>
  <command idMso="FilePrint" onAction="Распечатка"/>
</commands>
</customUI>
```

9. В описании ленты на языке XML возможно также отменить действие встроенного элемента управления (листинг 5.13).

Листинг 5.13. Пример отмены действия, позволяющего вставить картинку для элемента управления *Картинка* (ClipArtInsert) в описании ленты на языке XML

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
<commands>
  <command idMso="ClipArtInsert" enabled="false"/>
</commands>
</customUI>
```

Создаем панели инструментов из ранних версий MS Excel

Следует еще раз заметить, что панели инструментов предыдущих версий, если вы все-таки решили их создавать, имеют ряд существенных ограничений. Во-первых, они не могут свободно позиционироваться по пространству открытой рабочей книги, во-вторых, они будут размещаться на вкладке **Надстройки** ленты, а в-третьих, многие свойства и методы объекта `CommanBar`, который инкапсулирует данные о меню, контекстном меню или панели инструментов, могут просто игнорироваться в версии Microsoft Office Excel 2010. Кроме того, в отличие от модификаций ленты пользовательские панели инструментов будут доступны для всех рабочих книг Excel.

Приведем пример создания панели инструментов (см. файл *3-Пример создания Панели инструментов ранних версий.xlsm* на компакт-диске), которая будет содер-

жать несколько кнопок, которые соответственно: выводят приветствие и текущую дату, а также производят запуск Блокнота.

Итак, выполните следующие действия.

1. В окне модуля ЭтаКнига введите две процедуры: первая будет создавать панель инструментов, когда рабочая книга открыта, а вторая — удалять панель инструментов при закрытии рабочей книги (листинг 5.14).

Листинг 5.14. Процедуры для создания и удаления панели инструментов в рабочей книге. Модуль ЭтаКнига

```
Private Sub Workbook_Open()
    Call CreateToolbar
End Sub

Private Sub Workbook_BeforeClose(Cancel As Boolean)
    Call DeleteToolbar
End Sub
```

2. В окне стандартного модуля VBA **Module1** разместите код процедуры для создания панели инструментов (при открытии рабочей книги), включающей описание необходимых элементов управления, процедуры удаления панели инструментов при закрытии рабочей книги, а также для процедур, связанных с обработкой событий элементов управления (т. е. в нашем случае — с нажатием соответствующей кнопки панели инструментов).

Примечание

Обращаем еще раз внимание на то, что пользовательские панели инструментов будут доступны для всех рабочих книг Excel. Если у вас при открытии рабочей книги загружаются пользовательские панели, можно, во-первых, отключить вкладку **Надстройки**, а во-вторых, вообще их удалить: достаточно на листе модуля ЭтаКнига расположить процедуру удаления панели инструментов DeleteToolbar, например, при открытии рабочей книги, а на листе модуля — непосредственно описание процедуры удаления: какие именно панели следует удалять.

Конструируем контекстное меню

Контекстное меню представляется одним классом объектов — CommandBar. Такие объекты, как вы уже поняли, сохранились из предыдущих версий Excel. Они дают возможность одновременно содержать кнопки с пиктограммами и раскрывающимися списками.

Чтобы построить собственное контекстное меню в рабочей книге, следует выполнить такие действия.

1. Написать процедуры, обеспечивающие добавление собственного контекстного меню при открытии рабочей книги и его удаление при закрытии рабочей книги.
2. Создать новое контекстное меню.
3. Добавить в контекстное меню элементы управления и связать с ним макросы или процедуры VBA.

4. Задать момент (место) вывода контекстного меню.
5. Позаботиться о том, чтобы после вывода пользовательского контекстного меню не отображалось встроенное в Excel контекстное меню.

Созданное контекстное меню отображается на экране методом `ShowPopup` объекта `CommandBar`. Для задания момента отображения контекстного меню лучше всего подходит процедура обработки события `BeforeRightClick` объекта `Worksheet`. Параметр этой процедуры `Target` позволяет указать диапазон рабочего листа, в котором это меню должно отображаться. Если необходимо отображение контекстного меню при нажатии правой кнопки мыши в произвольном месте выбранного рабочего листа, то значение параметра `Target` можно не задавать. Установив значение параметра `Cancel` процедуры указанного события `SheetBeforeRightClick` объекта `Workbook` равным `True`, можно отключить выполнения тех функций, которые по умолчанию связаны с нажатием правой кнопки мыши.

Файл *4-Пример контекстного меню - На первом листе.xlsm*, расположенный на компакт-диске, демонстрирует создание контекстного меню на примере меню, состоящего из следующих элементов: команд **Формат числа** (открывает окно **Формат ячеек** на вкладке **Число**), **Шрифт** (открывает окно **Формат ячеек** на вкладке **Шрифт**), черта — начало группы, команд **Открыть**, **Сохранить как** и **Выход** (данные элементы выполняют функции одноименных кнопок на вкладке **Файл**. Пользовательское контекстное меню отображается при щелчке правой кнопкой мыши в диапазоне **A1:L25** на листе **Лист1** (диапазону ячеек присвоено имя — **menu**). Все необходимые листинги вы найдете в соответствующих модулях: **ЭтаКнига**, **Лист1**, **Module1** указанного файла примера.

> ***Примечание***
>
> На прилагаемом компакт-диске вы найдете еще один пример создания контекстного меню *5-Пример контекстного меню - Со списком.xlsm*, которое действует на весь рабочий лист и содержит элемент управления — раскрывающийся список.

Наши итоги

В данной главе вы познакомились с настройкой ленты и панели быстрого доступа. Используя широкие возможности модификации ленты, вы сможете теперь созвать необходимый пользовательский интерфейс ваших рабочих книг. Итак, вы можете производить модификации с помощью:

- пользовательских средств настройки ленты и панели быстрого доступа;
- записи макроса (или процедуры VBA) и назначении его кнопке на ленте и на панели быстрого доступа;
- прямого редактирования XML-файла;
- с использованием XML и VBA.

Кроме того, вы познакомились также с возможностью конструирования собственного контекстного меню и создания панелей инструментов более ранних версий Microsoft Excel.

Глава 6

Строим диаграммы

Для графической визуализации числовых данных в Microsoft Excel имеется достаточно широкий набор диаграмм различного вида. Естественно, для того чтобы построить диаграмму, вам необходимо предварительно подготовить диапазон данных для построения, определиться с типом диаграммы, продумать все элементы, которые хотите отобразить на диаграмме и т. п. Данная глава посвящена демонстрации возможностей построения диаграмм средствами Microsoft Excel 2010, а также объектов `Chart` и `ChartObject`, которые позволяют автоматизировать как процесс построения диаграмм, так и их настройки, в зависимости от бизнес-логики разрабатываемого проекта.

> ***Примечание***
>
> Файлы рассмотренных в данной главе примеров можно найти в папке Glava_6 на компакт-диске.

Что нужно знать о диаграммах?

В MS Excel можно строить два типа диаграмм: *внедренные* и диаграммы *на отдельных листах*. Внедренные диаграммы создают на рабочем листе рядом с таблицами, данными и текстом, и их используют при создании отчетов. Диаграммы на отдельном листе удобны для подготовки слайдов или для вывода на печать.

В новой версии Excel 2010 построение диаграмм осуществляется, практически, одним щелчком мыши: выделите подготовленные данные для диаграммы, перейдите на вкладку ленты **Вставка** и в группе **Диаграммы** выберите необходимый тип диаграммы. По умолчанию построенная диаграмма размещается рядом с данными на рабочем листе. При ее активизации становятся доступными еще три контекстные вкладки ленты: **Конструктор**, **Макет**, **Формат**. Инструменты, размещенные на этих вкладках, позволяют отформатировать полученную диаграмму, изменить ее тип, стиль, месторасположение и т. д.

Диаграммы MS Excel содержат различные объекты (рис. 6.1), каждый из которых можно изменять и форматировать. Кроме того, MS Excel предлагает использование различных типов диаграмм (табл. 6.1).

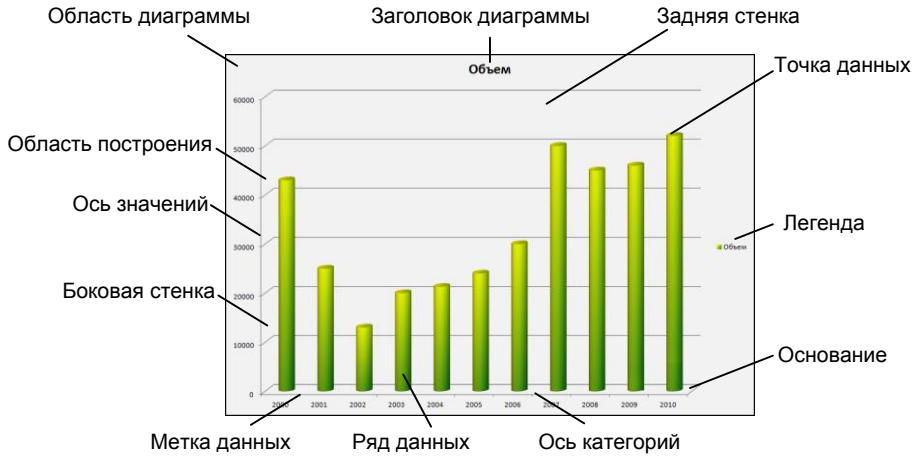


Рис. 6.1. Элементы диаграммы MS Excel

Таблица 6.1. *Типы диаграмм MS Excel*

Название	Описание
Гистограмма	Используется для сравнения отдельных величин или их изменений в течение некоторого периода времени. Удобна для отображения дискретных данных
График	График отображает зависимость данных (ось *y*) от величины, которая меняется с постоянным шагом (ось *x*). Метки оси категорий должны располагаться по возрастанию или убыванию. Графики чаще используют для коммерческих или финансовых данных, равномерно распределенных во времени (отображение непрерывных данных), или таких категорий, как продажи, цены и т. п. Данные по оси категорий должны применяться с постоянным шагом
Круговая	Круговая диаграмма отображает соотношение частей и целого и строится только по одному ряду данных, первому в выделенном диапазоне. Эту диаграмму можно использовать, когда компоненты в сумме составляют 100%
Линейчатая	Похожи на гистограммы (отличие — повернуты на 90° по часовой стрелке). Используются для сопоставления отдельных значений в определенный момент времени, не дают представления об изменении объектов во времени. Горизонтальное расположение полос позволяет подчеркнуть положительные или отрицательные отклонения от некоторой величины. Линейчатые диаграммы можно использовать для отображения отклонений по разным статьям бюджета в определенный момент времени. Можно перетаскивать точки в любое положение
С областями	Позволяют отслеживать непрерывное изменение суммы значений всех рядов данных и вклад каждого ряда в эту сумму. Этот тип применяется для отображения процесса производства или продажи изделий (с равно отстающими интервалами)
Точечная	Хорошо демонстрируют тенденции изменения данных при неравных интервалах времени или других интервалах измерения, отложенных по оси категорий. Можно использовать для представления дискретных измерений по осям *x* и *y*. В точечной диаграмме деления на оси категорий наносятся равномерно между самым низким и самым высоким значением *X*

Таблица 6.1 *(окончание)*

Название	Описание
Биржевая	Используется для отображения изменения информации о ценах на бирже. Отображает наборы данных из трех значений
Поверхность	Показывает низкие и высокие точки поверхности. Эти диаграммы используются для набора данных, который зависит от двух переменных. Диаграмму можно поворачивать и видеть с разных точек зрения
Кольцевая	Сравнивает вклад частей в целое. В отличие от круговой, на кольцевой диаграмме могут быть представлены два и более ряда данных
Лепестковая	Используют обычно, чтобы показать соотношения отдельных рядов данных, а также — одного определенного ряда данных и всех остальных рядов. Каждая категория лепестковой диаграммы имеет собственную ось координат (луч). Точки данных располагаются вдоль луча. Линии, соединяющие точки данных одного ряда, охватывают площадь, характеризующую совокупность значений в этом ряду. На лепестковой диаграмме можно отобразить, например, динамику затрат времени на проект, включающий несколько задач. В этом случае каждой категории (лучу) соответствует определенная задача проекта, а точке на луче — затраты времени на нее к какому-то сроку
Пузырьковая	Позволяют отображать на плоскости наборы из трех значений. Первые два значения откладываются по осям *x*, *y*. Третье значение представляется размером пузырька

С диаграммами можно также производить следующие операции:

- ❒ добавлять и удалять ряды данных;
- ❒ редактировать, форматировать и добавлять различные элементы диаграмм; изменять пространственную ориентацию трехмерных диаграмм;
- ❒ добавлять различные графические объекты (например, стрелки, выноски и т. д.);
- ❒ настраивать оси и выбирать шкалу;
- ❒ изменять типы диаграмм;
- ❒ создавать рисованные диаграммы (вместо цветовой заливки — рисунки);
- ❒ связывать текст на диаграмме с ячейками рабочего листа;
- ❒ создавать диаграммы на основе структурированных данных;
- ❒ применять диаграммы для анализа данных, т. е. строить различные линии тренда и делать прогнозы.

Примечание

В Microsoft Office Excel 2010 появилась новая возможность: теперь в ячейки рабочего листа вы можете поместить так называемые спарклайны. Спарклайны (или инфокривые) представляют собой микродиаграммы, которые располагаются в ячейке рабочего листа и визуализируют данные, представленные рядом в строке таблицы. Используя спарклайны, вы можете отобразить тенденции в рядах значений (например, тенденции на валютном рынке, экономические циклы, продажи по регионам и т. д.) и выделять максимальные и минимальные значения. В отличие от диаграмм на листе Excel, спарклайны не являются объектами: фактически, спарклайн — это фон ячейки. Добавить спарклайн можно в группе **Спарклайны** на вкладке **Вставка** ленты. В дальнейшем работа со спарклайнами осуществляется с помощью команд, расположенных на вкладке **Конструктор** ленты в режиме **Работа со спарклайнами**.

Создаем шаблон отчета с диаграммой

Создадим на основе числовых данных о мировом товарном экспорте диаграмму с использованием возможностей Microsoft Excel 2010 (см. файл *1-Диаграмма_Мировой товарный экспорт.xlsm* на компакт-диске), выполняя следующие действия.

1. Подготовьте на рабочем листе Microsoft Excel данные, связанные с мировым товарным экспортом (рис. 6.2).

	A	B	C	D	E	F	G	H	I	J	K
1	**Мировой товарный экспорт, в ценах и по ППС 2000 г., млрд. долл.**										
2											
3		**1900**	**1913**	**1929**	**1938**	**1950**	**1960**	**1970**	**1980**	**1990**	**2000**
4	**Германия**	21,5	54	58	64,1	36,5	87,5	185	385	600	710
5	**Франция**	22	28,5	40,5	40	31,5	62,5	140	235	330	420
6	**Великобритания**	38,5	54,5	73	76	66	105	160	235	320	400
7	**Бельгия**	12,2	15,5	18,4	16,8	12,3	27,5	63	112	176	214
8											

Рис. 6.2. Данные на рабочем листе Excel для построения диаграммы

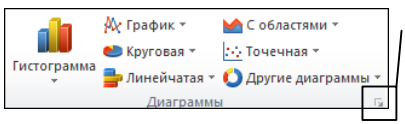


Рис. 6.3. Группа **Диаграммы** на вкладке **Вставка** ленты

2. Выделите подготовленный диапазон с данными.
3. Перейдите на вкладку **Вставка** ленты и в группе **Диаграммы** выберите необходимый тип диаграммы (рис. 6.3) — в нашем случае щелкните по кнопке с выпадающим списком **Гистограмма** и выберите тип диаграммы **Объемная гистограмма**. Следует отметить, что, нажав кнопку открытия диалогового окна **Создать диаграмму**, которая расположена в правом нижнем углу группы **Диаграммы**, вы сможете просмотреть все доступные типы диаграммы в соответствующем окне (рис. 6.4).
4. Итак, на рабочем листе, рядом с данными появилась диаграмма гистограммного типа, а на ленте — три дополнительные контекстные вкладки **Работа с диаграммами**: **Конструктор**, **Макет**, **Формат**.
5. Отформатируем полученную диаграмму. Прежде всего, увеличьте размер области диаграммы, потянув границу диаграммы с помощью указателя мыши (маркера перемещения). Обратите внимание на то, что для данного типа диаграммы при увеличении ее размеров, на трех осях появились все подписи данных. Далее, удалите легенду, щелкнув по ней мышью и нажав клавишу <Delete>. Перейдите на контекстную вкладку **Конструктор** и в группе **Данные** щелкните при необходимости по кнопке **Строка/столбец**, чтобы поменять на диаграмме их расположение. В группе **Данные** находится также кнопка **Выбрать данные**, которая позволяет изменить диапазон данных, представленных на диаграмме.

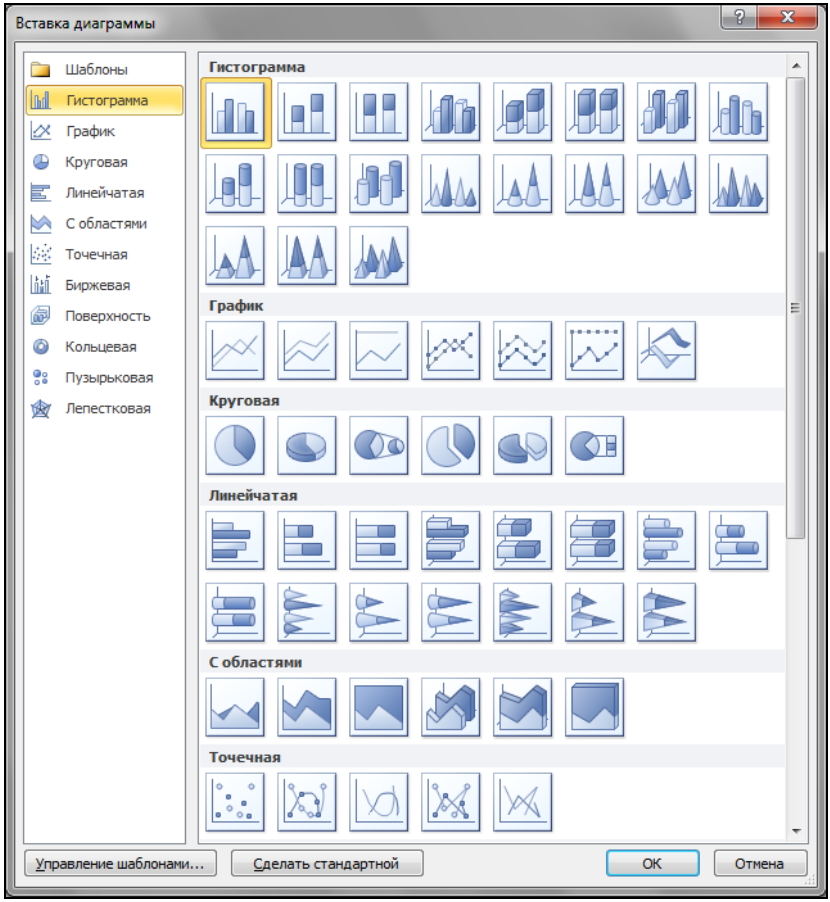


Рис. 6.4. Окно **Вставка диаграммы**

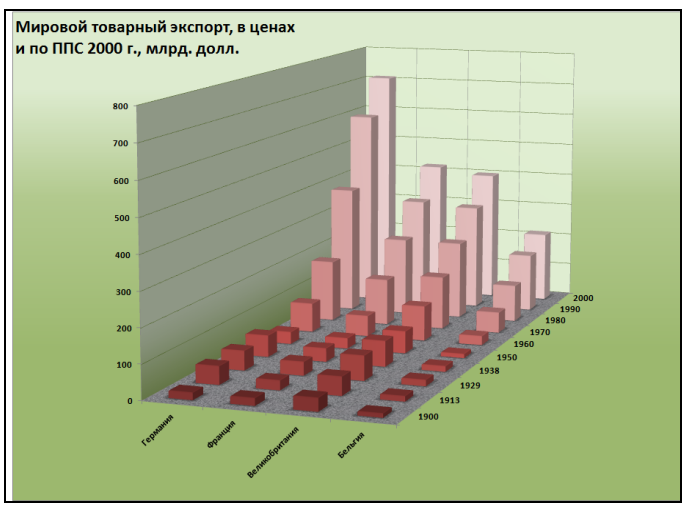


Рис. 6.5. Диаграмма, полученная с использованием средств Microsoft Excel 2010

6. Добавьте название диаграммы: на контекстной вкладке ленты **Макет** перейдите к группе **Подписи** и, щелкнув по кнопке **Название диаграммы**, выберите команду **Название по центру с перекрытием**. В появившейся области названия введите требуемую подпись для названия диаграммы — **Мировой товарный экспорт, в ценах и по ППС 2000 г., млрд. долл.**.
7. Используя другие возможности, предоставляемые инструментами контекстных вкладок или контекстным меню каждого элемента диаграммы, измените стиль диаграммы, заливку области диаграммы, форматирование различных элементов диаграммы. Отметим, что, выбрав кнопку **Переместить диаграмму**, расположенную в группе команд **Расположение** на контекстной вкладке **Конструктор**, можно расположить диаграмму на отдельном листе диаграмм в книге Microsoft Excel.
8. Таким образом, полученная диаграмма может выглядеть, например, в соответствии с рис. 6.5.

Что представляют собой семейства *ChartObjects*, *Charts* и объекты *ChartObject*, *Chart*?

Итак, в MS Excel можно создавать различные типы диаграммы и форматировать их надлежащим способом. С точки зрения VBA семейство листов рабочей книги `Sheets` включает в себя два семейства листов: `Worksheets` (рабочих листов) и `Charts` (листы диаграмм). Объектами семейства `Charts` являются диаграммы, созданные на листах диаграмм. Это семейство не включает в себя диаграммы, непосредственно внедренные в рабочие листы. Такие диаграммы принадлежат семейству `ChartObjects`. Таким образом, объект `ChartObject` встроен в объект `Worksheet`, а `Chart` — в `Workbook`. У объектов `Workbook` и `Application` имеется общее свойство `ActiveChart`, которое возвращает активную диаграмму рабочей книги, независимо от того, какому семейству она принадлежит. У объекта `Chart` имеется ряд дочерних по отношению к нему объектов, которые перечислены в табл. 6.2 (см. также рис. 6.1).

Таблица 6.2. *Объекты, подчиненные объекту* `Chart`

Объект	Описание
`ChartArea`	Область, в которой нарисована диаграмма
`PlotArea`	Область диаграммы, где нарисована диаграмма
`Floor`	Горизонтальная плоскость трехмерной диаграммы (основание)
`Walls` (`BackWall`, `Walls`)	Вертикальные плоскости трехмерной диаграммы
`Corners`	Углы трехмерной диаграммы
`PageSetup`	Параметры страницы
`ChartTitle`	Заголовок диаграммы

Таблица 6.2 *(окончание)*

Объект	Описание
SeriesCollection	Диапазон данных, откладываемых по оси ординат
Tredlines	Линия тренда
Axis	Оси координат
AxisTitle	Заголовки осей
DisplayUnitLabel	Единица масштабирования осей
Gridlines	Координатная сетка
TickLabels	Значки, откладываемые по осям
DataTable	Таблица данных диаграммы
Legend	Легенда
Shapes	Область построения диаграммы
SeriesCollection	Ряды данных
DataLabels	Подписи данных
Points	Точки данных

Если диаграмма располагается на рабочем листе, то иерархию объектов при обращении, например, к заголовку диаграммы можно представить в виде:

```
Application
    Workbook
        Worksheet
            ChartObject
                Chart
                    ChartTitle
```

Для диаграмм, выполненных на листах диаграмм, иерархия объектов будет уже немного другой:

```
Application
    Workbook
        Chart
             ChartTitle
```

Добавление нового элемента в семейства *ChartObjects* и *Charts*

Семейства `ChartObjects` и `Charts` имеют методы `Add` (создание нового элемента семейства), `Delete` (удаление элемента семейства) и свойство `Count` (возвращение числа элементов семейства).

Метод `Add` семейства `ChartObjects` включает следующие параметры:

```
Add(Left, Top, Width, Height)
```

- *`Left`*, *`Top`* — задают координаты на рабочем листе левого верхнего угла диаграммы.

- *Width*, *Height* — устанавливают ширину и высоту диаграммы.

Все параметры метода необязательные.

В свою очередь, метод Add семейства Charts имеет следующие параметры:

Add(*Before*, *After*, *Count*)

- *Before* — задает, перед каким листом добавляется диаграмма.
- *After* — указывает, после какого листа добавляется диаграмма.
- *Count* — определяет число добавляемых диаграмм.

Все параметры метода также не являются обязательными.

Свойства объекта *Chart*

Объект Chart имеет более 50 свойств, определяющих внешний вид диаграммы (подробное описание этих свойств см. в справочной системе MS Excel). Основные свойства объекта Chart представлены в табл. 6.3, а основные типы диаграмм (значения свойства ChartType) — в табл. 6.4.

Таблица 6.3. *Основные свойства объекта* Chart

Свойство	Описание
Area3DGroup	Возвращает объект ChartGroup, инкапсулирующий информацию об области, в которой выводится трехмерная диаграмма
AutoScaling	Устанавливает автоматическое масштабирование трехмерных диаграмм
Bar3DGroup	Возвращает объект ChartGroup, инкапсулирующий информацию о трехмерной диаграмме
ChartArea	Возвращает объект ChartArea
ChartTitle	Возвращает объект ChartTitle
ChartType	Задает тип диаграммы. Допустимые значения приведены в табл. 6.4
Column3DGroup	Возвращает объект ChartGroup, инкапсулирующий информацию о столбцах трехмерной диаграммы
Corners	Возвращает объект Corners
DataTable	Возвращает объект DataTable
DepthPercent	Устанавливает коэффициент глубины для трехмерной диаграммы
DisplayBlanksAs	Задает, как пустые ячейки интерпретируются при построении диаграммы. Допустимые значения: xlNotPlotted, xlInterpolated и xlZero
Elevation	Задает угол, под которым смотрят на трехмерную диаграмму
Floor	Возвращает объект Floor
GapDepth	Задает промежуток между рядами для трехмерной диаграммы

Таблица 6.3 *(окончание)*

Свойство	Описание
HasAxis	Устанавливает, имеются ли у диаграммы оси
HasDataTable	Устанавливает, имеется ли таблица данных
HasLegend	Проверяет, имеется ли легенда
HasTitle	Проверяет, имеется ли заголовок
HeightPercent	Задает высоту диаграммы как процент по отношению к ее ширине
Hyperlinks	Возвращает семейство Hyperlinks
Index	Возвращает значение индекса в семейства Charts
Legend	Возвращает объект Legend
PageSetup	Возвращает объект PageSetup
Perspective	Устанавливает перспективу для трехмерной диаграммы
PlotArea	Возвращает объект PlotArea
PlotBy	Определяет, как данные располагаются в диапазоне. Допустимые значения: xlColumns и xlRows
PlotVisibleOnly	Задает, надо ли учитывать невидимые ячейки
ProtectContents, ProtectData, ProtectDrawingObjects, ProtectFormatting, ProtectionSelection, ProtectGoalSeek	Логические свойства, которые определяют, установлена ли защита на соответствующий элемент диаграммы
Rotation	Возвращает угол поворота трехмерной диаграммы вокруг оси *z*
Visible	Управляет видимостью диаграммы
Walls	Возвращает объект Walls

Таблица 6.4. *Допустимые значения свойства* `ChartType`

Тип диаграммы	Константы
Гистограмма	xlColumnClustered, xl3DcolumnClustered, xlColumnStacked, xl3DcolumnStacked, xlColumnStacked100, xl3DColumnStacked100, xl3Dcolumn
Линейчатая	xlBarClustered, xl3DbarClustered, xlBarStacked, xl3DbarStacked, xlBarStacked100, xl3DbarStacked100
График	xlLine, xlLineMarkers, xlLineStacked, xlLineMarkersStacked, xlLineStacked100, xlLineMarkersStacked100, xl3Dline

***Таблица 6.4** (окончание)*

Тип диаграммы	Константы
Круговая	`xlPie`, `xlPieExploded`, `xl3Dpie`, `xl3DpieExploded`, `xlPieOfPie`, `xlBarOfPie`
Точечная	`xlXYScatter`, `xlXYScatterSmooth`, `xlXYScatterSmoothNoMarkers`, `xlXYScatterLines`, `xlXYScatterLinesNoMarkers`
С областями	`xlArea`, `xl3Darea`, `xlAreaStacked`, `xl3DareaStacked`, `xlAreaStacked100`, `xl3DAreaStacked100`
Кольцевая	`xlDoughnut`, `xlDoughnutExploded`
Лепестковая	`xlRadar`, `xlRadarMarkers`, `xlRadarFilled`
Поверхность	`xlSurface`, `xlSurfaceTopView`, `xlSurfaceWireframe`, `xlSurfaceTopViewWireframe`
Пузырьковая	`xlBubble`, `xlBubble3Deffect`
Биржевая	`xlStockHLC`, `xlStockVHLC`, `xlStockOHLC`, `xlStockVOHLC`
Цилиндрическая	`xlCylinderColClustered`, `xlCylinderBarClustered`, `xlCylinderColStacked`, `xlCylinderBarStacked`, `xlCylinderColStacked100`, `xlCylinderBarStacked100`, `xlCylinderCol`
Коническая	`xlConeColClustered`, `xlConeBarClustered`, `xlConeColStacked`, `xlConeBarStacked`, `xlConeColStacked100`, `xlConeBarStacked100`, `xlConeCol`
Пирамидальная	`xlPyramidColClustered`, `xlPyramidBarClustered`, `xlPyramidColStacked`, `xlPyramidBarStacked`, `xlPyramidColStacked100`, `xlPyramidBarStacked100`, `xlPyramidCol`

Методы объекта *Chart*

Объект `Chart`, как и любой другой объект, имеет не только свойства, но и методы, позволяющие управлять его внешним видом. В табл. 6.5 приводятся методы объекта `Chart`, которые будут полезны вам при работе с данным объектом.

***Таблица 6.5.** Методы объекта `Chart`*

Метод	Описание
`Activate`	Активизация диаграммы
`ApplyDataLabels`	Применение специфицированных меток
`AutoFormat`	Автоформат
`Axes`	Возвращает семейство `Axes`, используемое для установки различных свойств осей
`ChartObjects`	Возвращает семейство `ChartObjects`

Таблица 6.5 *(окончание)*

Метод	Описание
ChartWizard	Построение диаграммы
CheckSpelling	Проверка орфографии
Copy	Копирует диаграмму в указанное местоположение
CopyPicture	Копирует диаграмму в буфер обмена как рисунок
Delete	Удаляет диаграмму
Deselect	Снимает выделение с диаграммы
Export	Экспортирует диаграмму в файл графического формата
GetChartElement	Возвращает информацию об элементе диаграммы, расположенном в указанной точке
Location	Задает местоположение диаграммы
Move	Перемещает диаграмму
Paste	Вставляет данные диаграммы из буфера обмена
PrintOut	Выводит диаграмму на печать
SendToBack	Отображает диаграмму на заднем плане
Protect	Устанавливает защиту
Refresh	Обновляет диаграмму
SaveAs	Сохраняет измененную диаграмму в новом файле
Select	Выбирает диаграмму
SeriesCollection	Возвращает ряды данных
SetBackgroundPicture	Задает фоновый рисунок
SetSourceData	Задает диапазон, на основе которого строится диаграмма
Unprotect	Снимает защиту с диаграммы

События объекта *Chart*

У объекта `Chart` имеется также целый ряд событий (табл. 6.6), позволяющих отслеживать различные действия пользователя.

Таблица 6.6. *События объекта* `Chart`

Событие	Описание
Activate	Происходит при активизации диаграммы
BeforeDoubleClick	Происходит перед двойным щелчком
BeforeRightClick	Происходит перед щелчком правой кнопкой
Calculate	Происходит при изменении данных

Таблица 6.6 *(окончание)*

Событие	Описание
`Deactivate`	Происходит при деактивизации диаграммы
`DragOver`	Происходит при буксировке диапазона над диаграммой
`DragPlot`	Происходит при буксировке и вставке диапазона в диаграмму
`MouseDown`, `MouseUp`	Происходят, когда пользователь нажимает и отпускает любую кнопку мыши
`MouseMove`	Происходит, когда пользователь передвигает указатель мыши над диаграммой
`Resize`	Происходит при изменении размеров диаграммы
`Select`	Происходит при выборе элемента диаграммы
`SeriesChange`	Происходит при изменении указателя на ряд данных

Строим диаграмму с помощью VBA

А теперь рассмотрим пример построения диаграммы на VBA. На рабочем листе книги Excel располагаются 4 кнопки: **Построение диаграммы**, **Удаление диаграммы**, **Добавление подписей данных**, **Удаление подписей данных**. При нажатии кнопки **Построение диаграммы** производится построение диаграммы на активном рабочем листе, причем ее заголовок будет совпадать с содержимым ячейки **B1** (рис. 6.6, см. также файл *2-Построение простой диаграммы гистограммного типа.xlsm* на компакт-диске).

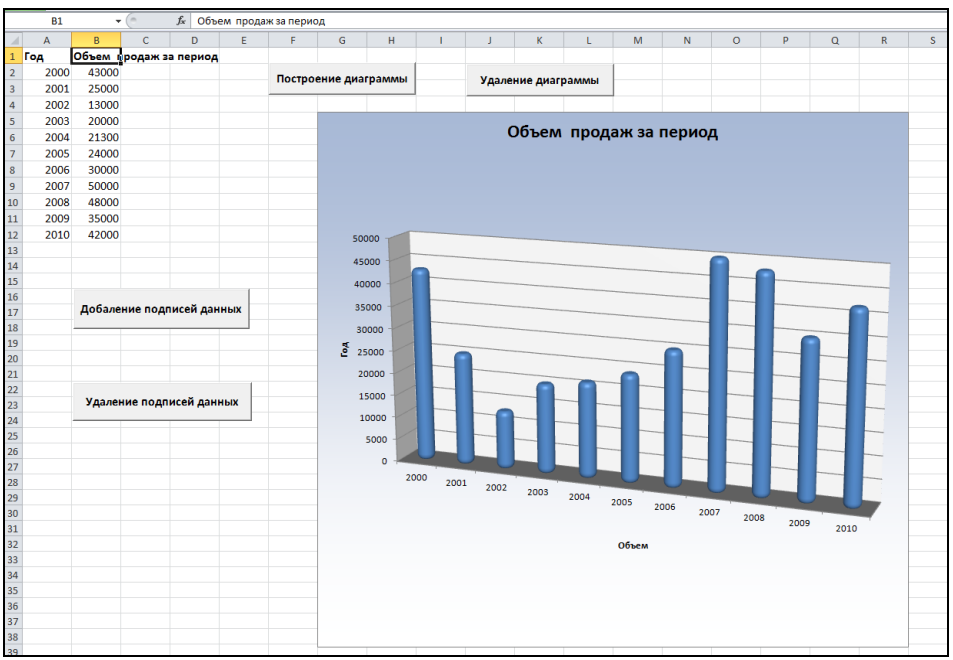


Рис. 6.6. Построение диаграммы

Кнопкой **Удаление диаграммы** удаляется диаграмма. Оставшиеся две кнопки позволяют соответственно добавить или удалить ряд подписей данных на построенной диаграмме.

Разместите на рабочем листе четыре кнопки CommandButton. Обратите внимание на то, что на вкладке **Разработчик** ленты в группе команд **Элементы управления** кнопка **Режим конструктора** является активной.

Введите соответственно для свойства `Caption` добавленных кнопок (выделите кнопку и воспользуйтесь соответственно кнопкой **Свойства** группы команд **Элементы управления** на вкладке **Разработчик** ленты) следующие значения: `Построение диаграммы`, `Удаление диаграмм`, `Добавление подписей данных` и `Удаление подписей данных`. При желании измените также значения для свойства `Font`.

Выполните последовательно щелчки мышью по добавленным кнопкам и добавьте в модуле **Лист1** программный код в соответствии с листингом 6.1.

Листинг 6.1. Построение диаграммы. Модуль `Лист1`

```
Private Sub CommandButton1_Click()
  ChartCreate
End Sub

Private Sub CommandButton2_Click()
  ChartDelete
End Sub

Private Sub CommandButton3_Click()
  DataLabAdd
End Sub

Private Sub CommandButton4_Click()
  DataLabDelete
End Sub
```

Таким образом, щелчки мыши по соответствующим кнопкам должны выполнять следующие процедуры: `ChartCreate` — добавлять диаграмму на рабочий лист, `ChartDelete` — удалять диаграмму с рабочего листа, `DataLabAdd` — добавлять подписи данных на диаграмму, `DataLabDelete` — удалять подписи данных на диаграмме.

Для реализации данных процедур следует добавить в редакторе VBA стандартный модуль (команда **Insert | Module**) и ввести соответствующий программный код (листинг 6.2).

Отметим, что процедура `ChartCreate` добавления диаграммы реализуется следующим образом. Методом `Add` создается новая диаграмма. Свойство `ChartType` задает тип диаграммы. Методом `SetSourceData` дается ссылка на диапазон, значения из которого откладываются по оси ординат. В данном случае это диапазон **B2:B12** активного рабочего листа. Метод `SeriesCollection` задает ссылку на диапазон, значения из которого откладываются по оси абсцисс. В нашем случае это

диапазон **A2:A12** активного рабочего листа. Затем методом `Location` указывается местоположение диаграммы. В данном случае она будет внедрена в рабочий лист со специфицированным именем, которое совпадает с содержимым ячейки **B1**. После этого для диаграммы описываются ее элементы. Так, свойством `ChartTitle` и методом `Axes` задаются заголовок (который совпадает с именем рабочего листа) и названия осей (обратите внимание на то, что методом `.Axes(xlSeries).Delete` удаляются подписи данных по второй оси, расположенной в основании диаграммы). Далее свойством `HasLegend` удаляется легенда, после чего идет задание свойств для форматирования стенок, основания диаграммы, ряда данных и области построения диаграммы. Свойствами `Top`, `Left`, `Width` и `Height` диаграмма размещается в специфицированном месте рабочего листа. Таким образом, данная процедура позволяет строить диаграмму на любом рабочем листе.

Удаление диаграммы производится методом `ChartObjects.Delete`.

Процедура `DataLabAdd` добавления подписей данных на диаграмму базируется на использовании метода `ApplyDataLabels` для ряда данных `SeriesCollection(1)`. В свою очередь, процедура удаления подписей данных `DataLabDelete` устанавливает значение `False` для свойства `HasDataLabels`.

Листинг 6.2. Построение диаграммы. Стандартный модуль

```
'   Процедура построения диаграммы
Sub ChartCreate()
   Dim rx As Range
   Dim ry As Range
   Dim nameX As String
   Dim nameY As String
   Dim title As String
   Dim nameSh As String
   nameX = "Объем"
   nameY = "Год"
   nameSh = ActiveSheet.Name
   title = Sheets(nameSh).Range("B1")
   Set ry = Sheets(nameSh).Range("B2:B12")
   Set rx = Sheets(nameSh).Range("A2:A12")

'  Добавление диаграммы
   Charts.Add
   ActiveChart.ChartType = xlCylinderCol
   ActiveChart.SetSourceData Source:=ry, PlotBy:=xlColumns
   ActiveChart.SeriesCollection(1).XValues = _
       "=" & rx.Address(ReferenceStyle:=xlR1C1, external:=True)
   ActiveChart.Location Where:=xlLocationAsObject, Name:=nameSh

'  Определение элементов диаграммы
   With ActiveChart
      .HasTitle = True
```

```
      .ChartTitle.Characters.Text = title
      .Axes(xlCategory, xlPrimary).HasTitle = True
      .Axes(xlCategory, xlPrimary).AxisTitle.Characters.Text = nameX
      .Axes(xlValue, xlPrimary).HasTitle = True
      .Axes(xlValue, xlPrimary).AxisTitle.Characters.Text = nameY
      .Axes(xlSeries).Delete
   End With
   ActiveChart.HasLegend = False

'  Форматирование задней стенки диаграммы
   With ActiveChart.BackWall.Format.Fill
        .Visible = msoTrue
        .ForeColor.ObjectThemeColor = msoThemeColorBackground1
        .ForeColor.TintAndShade = 0
        .ForeColor.Brightness = -0.150000006
        .Transparency = 0
        .Solid
   End With

'  Форматирование боковых стенок диаграммы
   With ActiveChart.Walls.Format.Fill
        .Visible = msoTrue
        .ForeColor.ObjectThemeColor = msoThemeColorBackground1
        .ForeColor.TintAndShade = 0
        .ForeColor.Brightness = -0.050000007
        .Transparency = 0
        .Solid
    End With

'   Форматирование основания диаграммы
    With ActiveChart.Floor.Format.Fill
        .Visible = msoTrue
        .ForeColor.ObjectThemeColor = msoThemeColorBackground1
        .ForeColor.TintAndShade = 0
        .ForeColor.Brightness = -0.5
        .Transparency = 0
        .Solid
    End With

'   Форматирование ряда данных
    With ActiveChart.SeriesCollection(1).Format.ThreeD
        .BevelTopType = msoBevelCoolSlant
        .BevelTopInset = 13
        .BevelTopDepth = 6
    End With
```

```
'   Форматирование области построения диаграммы
    With Worksheets(nameSh).Shapes("Диаграмма 1").Fill
        .Visible = msoTrue
        .ForeColor.ObjectThemeColor = msoThemeColorAccent1
        .ForeColor.TintAndShade = 0.3399999738
        .ForeColor.Brightness = 0
        .BackColor.ObjectThemeColor = msoThemeColorAccent1
        .BackColor.TintAndShade = 0.7649999857
        .BackColor.Brightness = 0
        .TwoColorGradient msoGradientHorizontal, 1
    End With

'   Размещение диаграммы на рабочем листе Excel
    With Worksheets(nameSh).ChartObjects(1)
      .Top = Range("G5").Top
      .Left = Range("G5").Left
      .Width = Range("G1:R34").Width
      .Height = Range("C1:R34").Height
    End With
End Sub

'   Процедура удаления диаграммы
Sub ChartDelete()
   ActiveSheet.ChartObjects.Delete
End Sub

'   Процедура добавления подписи данных на диаграмму
Sub DataLabAdd()
    Dim Rng As Range
    Dim Ct As Chart
    Dim i As Integer, K As Integer

'   Идентификация диаграммы
    Set Ct= ActiveSheet.ChartObjects(1).Chart

'   Запрос ввода ряда данных для подписи
    On Error Resume Next
    Set Rng = Application.InputBox _
      (prompt:="Введите диапазон для подписей ряда данных", Type:=8)
    If Rng Is Nothing Then Exit Sub
    On Error GoTo 0

'   Добавление подписей данных
      Ct.SeriesCollection(1).ApplyDataLabels _
      Type:=xlDataLabelsShowValue, AutoText:=True, LegendKey:=False

'   Идентификация точки и подписи данных
```

```
    K = Ct.SeriesCollection(1).Points.Count
    For i = 1 To Pts
      Ct.SeriesCollection(1).Points(i).DataLabel.Text = _
          "=" & "'" & Rng.Parent.Name & "'!" & _
            Rng(i).Address(ReferenceStyle:=xlR1C1)
End Sub

'   Процедура удаления подписей данных на диаграмме
Sub DataLabDelete()
    Dim Ct As Chart
    Dim x As Series
    Set Ct= ActiveSheet.ChartObjects(1).Chart
    Ct.SeriesCollection(1).HasDataLabels = False
End Sub
```

Изменяем диапазон, по которому строится диаграмма

Рассмотрим отчетную ведомость по итогам работы сети компьютерных клубов, которая размещается на рабочем листе **Ведомость** в книге MS Excel. При открытии рабочей книги рядом с данными должна располагаться диаграмма, дающая визуальное представление динамики работы конкретного клуба. Последнее должно достигаться за счет возможности выбора конкретного клуба из списка клубов (рис. 6.7). Более того, выбор клуба из списка должен приводить к изменению диапазона, по которому строится диаграмма, и соответствующей перестройке диаграммы.

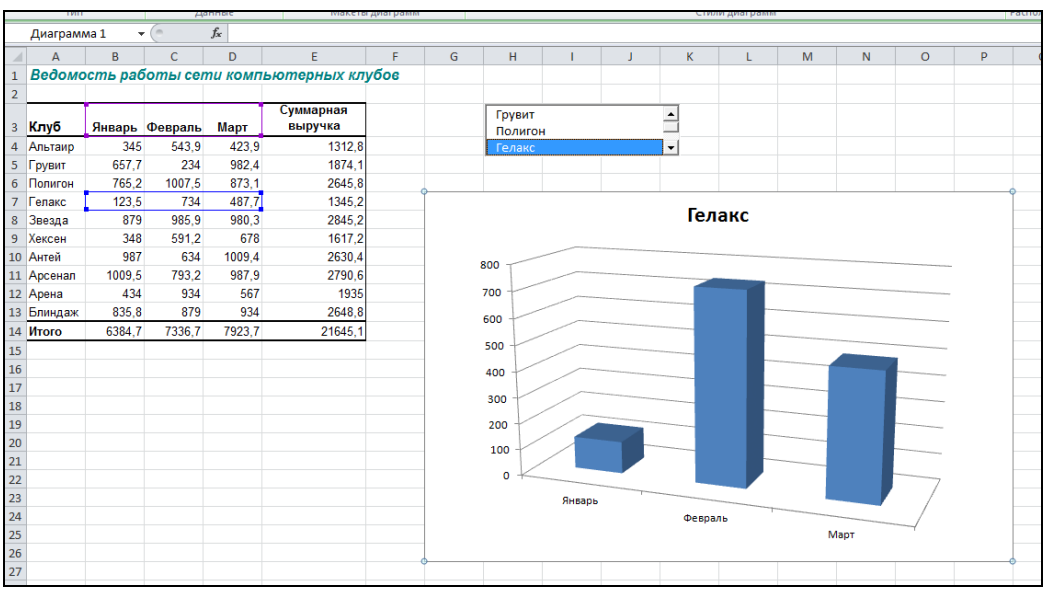


Рис. 6.7. Изменение диапазона, по которому строится диаграмма

Для реализации данного проекта выполните следующие действия.

1. Присвойте, например, первому рабочему листу имя **Ведомость**.
2. Подготовьте на данном рабочем листе табличную ведомость по итогам работы компьютерных клубов.
3. Создайте на рабочем листе список и, используя окно **Properties,** установите значение его свойства `Name` равным `Club`.
4. В модуле `ЭтаКнига` наберите необходимый код (см. файл *3-Изменение диапазона, по которому строится диаграмма.xlsm* на компакт-диске): при открытии рабочей книги выполняется процедура `Workbook_Open`, которая удаляет все диаграммы с рабочего листа **Ведомость**, строит объемную гистограмму по результатам работы компьютерного клуба "Альтаир" и располагает ее так, чтобы она занимала диапазон **G7:P26**. Это гарантирует неналожение диаграммы на таблицу с данными. В процедуре также заполняется список на основе заголовков записей таблицы данных и происходит выбор первого элемента из этого списка.
5. В модуле рабочего листа **Ведомость** наберите процедуру обработки события `Click` списка, которая при выборе клуба из списка перестраивает диаграмму (см. также файл *3-Изменение диапазона, по которому строится диаграмма.xlsm* на компакт-диске).

Изменяем тип диаграммы

А теперь мы внесем изменения в предыдущий пример и добавим на рабочий лист еще один список с целью, чтобы пользователь мог управлять как типом диаграммы, так и отображением легенды (рис. 6.8, см. также файл *4-Изменение диапазона и типа диаграммы.xlsm* на компакт-диске).

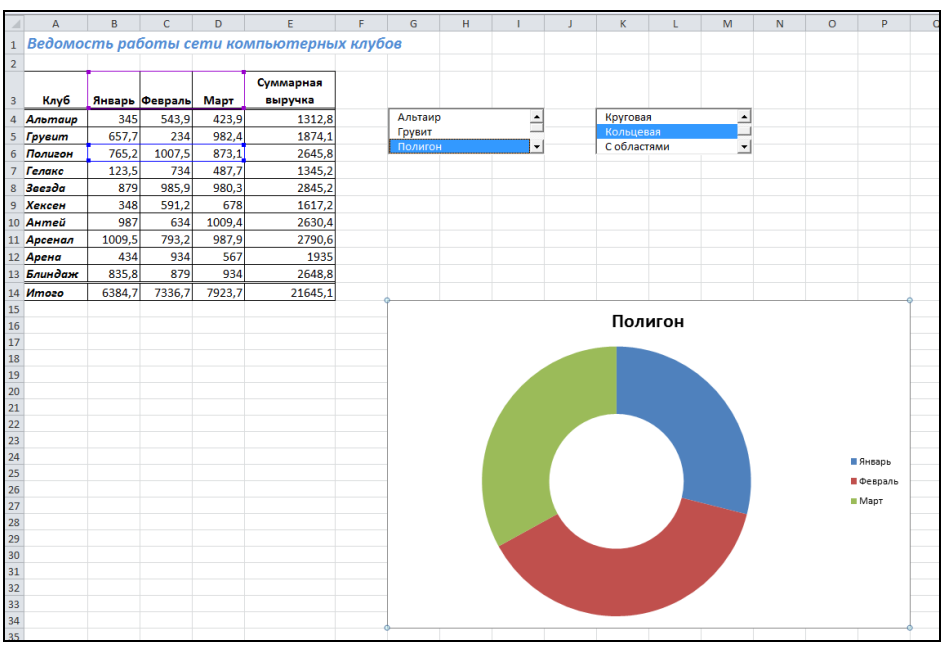


Рис. 6.8. Управление типом и легендой диаграммы

1. На рабочем листе **Ведомость** создайте второй список. При помощи окна **Properties** установите значение его свойства `Name` равным `DType`.
2. В модуле рабочего листа **Ведомость** дополнительно введите код процедуры `FillDType` (листинг 6.3), а в процедуру обработки события `Open` рабочей книги соответственно добавьте вызов этой процедуры `FillDType`.

Листинг 6.3. Управление типом и легендой диаграммы. Модуль `ЭтаКнига`

```
Private Sub Workbook_Open()
   DeleteCharts
   ChartBuilder
   FilllstCategory
   FillDType
End Sub
Private Sub FillDType()
   Dim tb(6, 1) As Variant
       tb(0, 0) = "Гистограмма": tb(0, 1) = xlColumnClustered
       tb(1, 0) = "График":      tb(1, 1) = xlLine
       tb(2, 0) = "Круговая":    tb(2, 1) = xlPie
       tb(3, 0) = "Кольцевая":   tb(3, 1) = xlDoughnut
       tb(4, 0) = "С областями": tb(4, 1) = xlArea
       tb(5, 0) = "Линейчатая":  tb(5, 1) = xlBarClustered
       tb(6, 0) = "Коническая":  tb(6, 1) = xlConeColStacked

   With Worksheets("Ведомость").DType
      .ColumnCount = 2
      .TextColumn = 2
      .ColumnWidths = "70;0"
      .List = tb
      .ListIndex = 0
   End With
End Sub
```

3. В модуле рабочего листа **Ведомость** наберите следующую процедуру обработки события `Click` списка, которая при выборе типа диаграммы перестраивает ее (листинг 6.4).

Листинг 6.4. Выбор категории расходов. Модуль рабочего листа *Ведомость*

```
Private Sub DType_Click()
    ActiveSheet.ChartObjects(1).Activate
    ActiveChart.ChartType = DType.Text
    If DType.Text = xlPie Or DType.Text = xlDoughnut Then
      ActiveChart.HasLegend = True
    Else
      ActiveChart.HasLegend = False
    End If
End Sub
```

Примечание

Обратите внимание на одну особенность, которая использована в данном примере. У диаграмм есть название типов, а также константы, задающие типы диаграмм. В списке необходимо отобразить только названия типов диаграмм, а результатом выбора элемента из списка должна быть константа, задающая этот тип. Для решения этой проблемы в программном коде создается двухстолбцовый список. Первый столбец состоит из названий типов, а второй — из констант, задающих эти типы. Второй столбец нам не нужен, поэтому для него устанавливается нулевая ширина, что приводит к тому, что элементы второго столбца не отображаются в списке. Но т. к. значение свойства `TextColumn` списка установлено равным 2, то при выборе элемента из списка в качестве значения свойства `Text` возвращается элемент второго, а не первого столбца.

Примечание

В зависимости от типа диаграммы легенда может быть необходима, а может оказаться и лишней, т. к. дублирует подписи, приведенные в диаграмме. Например, для гистограммы легенда в построенном приложении совсем не нужна, но для круговой или кольцевой диаграмм без легенды не обойтись. Поэтому в процедуре `lstType_Click` в зависимости от выбранного типа диаграммы легенда либо отображается, либо скрывается.

Автоматически перестраиваем диаграмму при изменении диапазона данных

Продолжая усложнять пример, связанный с деятельностью компьютерных клубов, усовершенствуем автоматическое построение диаграмм. Итак, теперь пользователь в общую таблицу ведомости, расположенную на рабочем листе **Ведомость**, может добавлять или удалять любое число месяцев. После изменения количества месяцев достаточно щелкнуть на списке клубов, и диаграмма будет автоматически перестроена. Для реализации этой задачи изменим процедуру обработки события `Click` списка клубов в соответствии с листингом 6.5 (см. также файл *5-Изменение диапазона и типа диаграммы_Добавление месяцев.xlsm* на компакт-диске).

Листинг 6.5. Управление типом и легендой диаграммы. Модуль рабочего листа *Ведомость*

```
Private Sub Club_Click()
   Dim r As Integer
   ActiveSheet.ChartObjects(1).Activate
   r = Club.ListIndex + 1
   Dim rgn As Range
   Dim rgnTitle As Range
   Set rgn = Range("A3").CurrentRegion

   Set rgnTitle = rgn.Rows(1)
   Set rgnTitle = rgnTitle.Offset(0, 1)
   Set rgnTitle = rgnTitle.Resize(ColumnSize:=rgnTitle.Columns.Count - 2)

   Set rgn = rgn.Offset(1, 1)
```

```
    Set rgn = rgn.Resize(rgn.Rows.Count - 2, rgn.Columns.Count - 2)

    With ActiveChart
        .SetSourceData Source:=rgn.Rows(r), PlotBy:=xlRows
         .SeriesCollection(1).XValues = rgnTitle
    End With
    With ActiveChart
        .HasTitle = True
        .ChartTitle.Characters.Text = Club.Text
    End With
End Sub
```

Последовательно отображаем ряды данных на диаграмме

По умолчанию в Microsoft Excel диаграммы не отображают данные, которые содержатся в скрытых строках или столбцах. В данном примере мы проиллюстрируем легкий способ скрывать и отображать ряды данных на диаграмме. В качестве элементов управления, позволяющих визуализировать или скрывать ряд данных, как на рабочем листе, так и на диаграмме (рис. 6.9), нами будут использованы флажки.

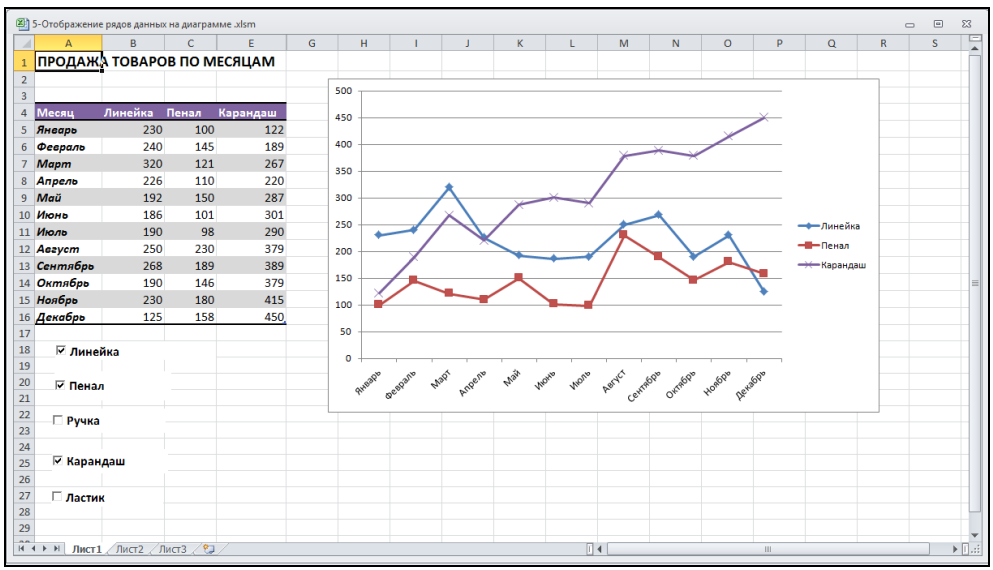


Рис. 6.9. Добавление или удаление ряда данных на диаграмме

Для реализации данного примера выполните следующие действия.

1. На рабочем листе подготовьте таблицу, связанную с продажей товаров по месяцам.
2. Постройте диаграмму для подготовленных данных, для чего перейдите на вкладку **Вставка** ленты и в группе **Диаграммы** выберите из списка **График** тип диаграммы — **График с маркерами**.

3. Отформатируйте полученную диаграмму.
4. Добавьте на рабочий лист последовательно пять элементов управления **Флажок** и установите для них соответствующие параметры для свойства `Caption`: `Линейка`, `Пенал`, `Ручка`, `Карандаш`, `Ластик`.
5. Добавьте на лист модуля рабочего листа **Лист1** процедуры обработки события `Click` для элементов управления **Флажок** (см. файл *6-Отображение рядов данных на диаграмме.xlsm* на компакт-диске).

Создаем проект с линией тренда

Достаточно часто на диаграммах желательно видеть определенные тенденции в представлении данных. В этом случае можно добавить линию тренда, позволяющую делать определенные прогнозы.

Если вы хотите добавить линию тренда средствами Microsoft Excel, необходимо активизировать диаграмму, перейти на контекстную вкладку **Макет** ленты и в группе **Анализ**, щелкнув по списку **Линия тренда**, выбрать требуемый вид линии тренда. Следует помнить, что линия тренда всегда строится для выбранного ряда данных, поэтому, если на диаграмме размещено несколько рядов данных, вам дополнительно придется уточнить, для какого ряда данных строится линия тренда. С другой стороны, вы можете также выделить требуемый ряд на диаграмме и в контекстном меню выбрать команду **Добавить линию тренда**.

С точки зрения VBA все линии тренда, соответствующие конкретному ряду данных, образуют семейство `Trendlines`, элементами которого являются объекты `Trendline` (линия тренда). Семейство `Trendlines` имеет только два метода: `Add`, добавляющий новый элемент в семейство, и `Item`, возвращающий конкретный элемент семейства. Отметим, что объект `Trendline` имеет такие же свойства, как параметры метода `Add` (см. справочную систему Microsoft Visual Basic for Applications).

Приведем описание метода `Add` семейства `Trendlines`.

```
Add(Type, Order, Period, Forward, Backward, Intercept, DisplayEquation,
↳DisplayRSquared, Name)
```

- *`Type`* — устанавливает тип линии тренда. Допустимые значения: `xlLinear` (линейная), `xlLogarithmic` (логарифмическая), `xlExponential` (экспоненциальная), `xlPolynomial` (полиномиальная), `xlMovingAvg` (скользящее среднее) и `xlPower` (степенная).
- *`Order`* — устанавливает порядок линии тренда. Допустимые значения: целые числа из интервала от 2 до 6. Используется, если аргумент *`Type`* принимает значение `xlPolynomial`.
- *`Period`* — период тренда. Допустимые значения: целое из диапазона от 1 до числа точек, по которым строится тренд. Используется, если аргумент *`Type`* принимает значение `xlMovingAvg`.
- *`Forward`* — число точек вперед (в будущее) для предсказания значений в соответствии с трендом.
- *`Backward`* — число точек назад (в прошлое) для предсказания значений в соответствии с трендом.

- `Intercept` — пересечение с осью ординат.
- `DisplayEquation` — параметр, принимающий логические значения, показывающие, надо ли отображать на диаграмме уравнение тренда.
- `DisplayRSquared` — параметр, принимающий логическое значение, показывающее, надо ли отображать квадрат коэффициента корреляции.
- `Name` — строка, задающая имя линии тренда.

Построение линии тренда рассмотрим снова на примере деятельности компьютерных клубов. В наш последний проект добавим группу элементов управления **Флажок**, которые позволят управлять отображением линии тренда на диаграмме (рис. 6.10, см. также файл *7-Построение линии тренда.xlsm* на компакт-диске).

На рабочий лист **Ведомость** добавьте три флажка и при помощи окна **Properties** установите им значения свойств, как показано в табл. 6.7.

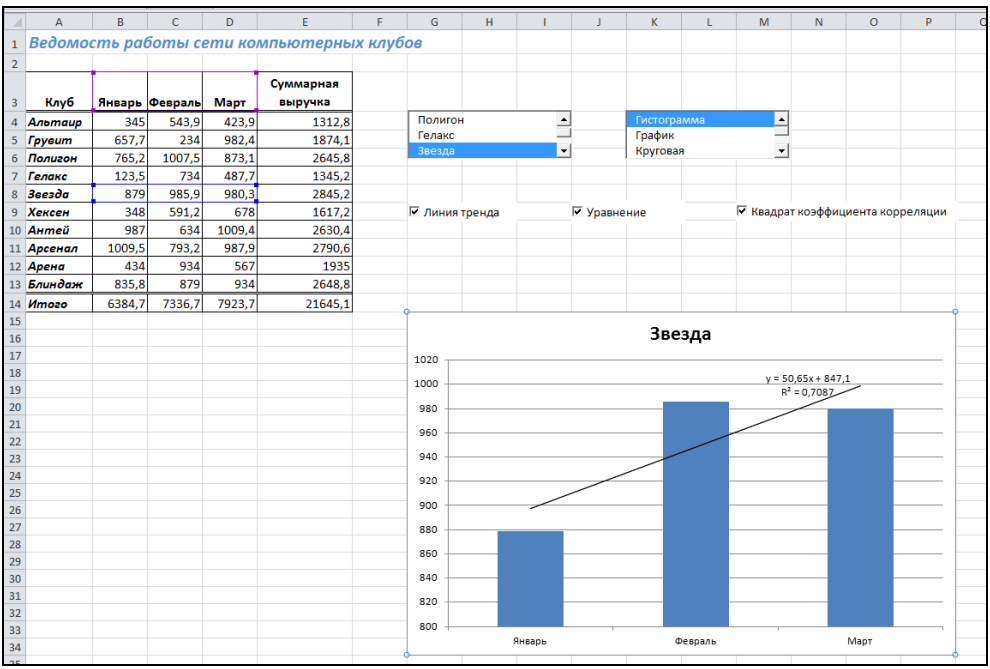


Рис. 6.10. Проект с линией тренда

Таблица 6.7. *Значения свойств, установленные в окне* ***Properties***

Элемент управления	Свойство	Значение
Флажок	`Name`	`Trend`
	`Caption`	`Линия тренда`
Флажок	`Name`	`Equation`
	`Caption`	`Уравнение`
Флажок	`Name`	`Coef`
	`Caption`	`Квадрат коэффициента корреляции`

Откройте предыдущий проект, связанный с деятельностью компьютерных клубов, и внесите следующие дополнения. В модуль рабочего листа **Ведомость** добавьте код (листинг 6.6) приводимых далее трех процедур обработки событий `Click` каждого из флажков, где:

- флажок **Линия тренда** обеспечивает построение или удаление линии тренда. Если линия тренда удалена, то флажки **Уравнение** и **Квадрат коэффициента корреляции** блокируются;
- флажок **Уравнение** отвечает за вывод уравнения на линию тренда;
- флажок **Квадрат коэффициента корреляции** управляет отображением значения одноименного коэффициента.

Листинг 6.6. Линия тренда. Модуль рабочего листа *Ведомость*

```
Private Sub Trend_Click()
   Dim c As Chart
   On Error Resume Next
   If DType.Text = xlPie Or DType.Text = xlDoughnut Then Exit Sub

   ActiveSheet.ChartObjects(1).Activate

   Set c = ActiveChart
   If Trend.Value Then
      Equation.Enabled = True
      Coef.Enabled = True
      c.SeriesCollection(1).Trendlines.Add _
           Type:=xlLinear, Forward:=0, Backward:=0, _
           DisplayEquation:=False, DisplayRSquared:=False
      If Trend.Value Then
          c.SeriesCollection(1).Trendlines(1).DisplayEquation = False
      End If
      If Coef.Value Then
          c.SeriesCollection(1).Trendlines(1).DisplayRSquared = True
      End If
   Else
      c.SeriesCollection(1).Trendlines(1).Delete
      Equation.Enabled = False
      Coef.Enabled = False
   End If
End Sub

Private Sub Equation_Click()
   On Error Resume Next
   If DType.Text = xlPie Or DType.Text = xlDoughnut Then Exit Sub
   If Trend.Value Then
      Dim c As Chart
      ActiveSheet.ChartObjects(1).Activate
```

```
      Set c = ActiveChart
      If Equation.Value Then
         c.SeriesCollection(1).Trendlines(1).DisplayEquation = True
      Else
         c.SeriesCollection(1).Trendlines(1).DisplayEquation = False
      End If
   End If
End Sub

Private Sub Coef_Click()
   On Error Resume Next
   If DType.Text = xlPie Or DType.Text = xlDoughnut Then Exit Sub
   If Trend.Value Then
      Dim c As Chart
      ActiveSheet.ChartObjects(1).Activate
      Set c = ActiveChart
      If Coef.Value Then
        c.SeriesCollection(1).Trendlines(1).DisplayRSquared = True
      Else
        c.SeriesCollection(1).Trendlines(1).DisplayRSquared = False
      End If
   End If
End Sub
```

В модуль `ЭтаКнига` добавьте следующую процедуру, устанавливающую начальные состояния флажков (листинг 6.7). Конечно, вызов процедуры `InitTrend` надо также добавить в процедуру `Workbook_Open`, расположенную в том же модуле.

Листинг 6.7. Линия тренда. Модуль `ЭтаКнига`

```
Private Sub InitTrend()
   With Worksheets("Ведомость")
      .Trend.Value = False
      .Equation.Value = False
      .Coef.Value = False
   End With
End Sub
```

Строим поверхности и управляем ориентацией

Рассмотрим теперь пример с построением поверхности, которая, естественно, имеет пространственную ориентацию. В качестве данных для построения опять будем использовать ведомость работы сети компьютерных клубов. Поверхность будет строиться автоматически по таблице при открытии рабочей книги, причем

по оси абсцисс откладываются названия клубов, по оси ординат — месяцы, а оси *z* — доходы клубов. Понятно, что наглядность представления данных в виде поверхности существенно зависит от того, с какой стороны и под каким углом пользователь на нее смотрит. Поэтому дополнительно к программированию процесса создания поверхности расположим на рабочем листе два элемента управления — два списка. Первый список предназначен для изменения угла подъема, с которого мы смотрим на поверхность (рис. 6.11). Установите значение его свойства `Name` равным `Elev`. Второй — для поворота поверхности вокруг оси *z*. Установите значение его свойства `Name` равным `Rotat`. В списки при активизации рабочего листа выводятся допустимые углы. В стандартном модуле `ЭтаКнига` и модуле рабочего листа **Ведомость** введите код для соответствующих процедур (см. файл *8-Вращение поверхности.xlsm* на компакт-диске).

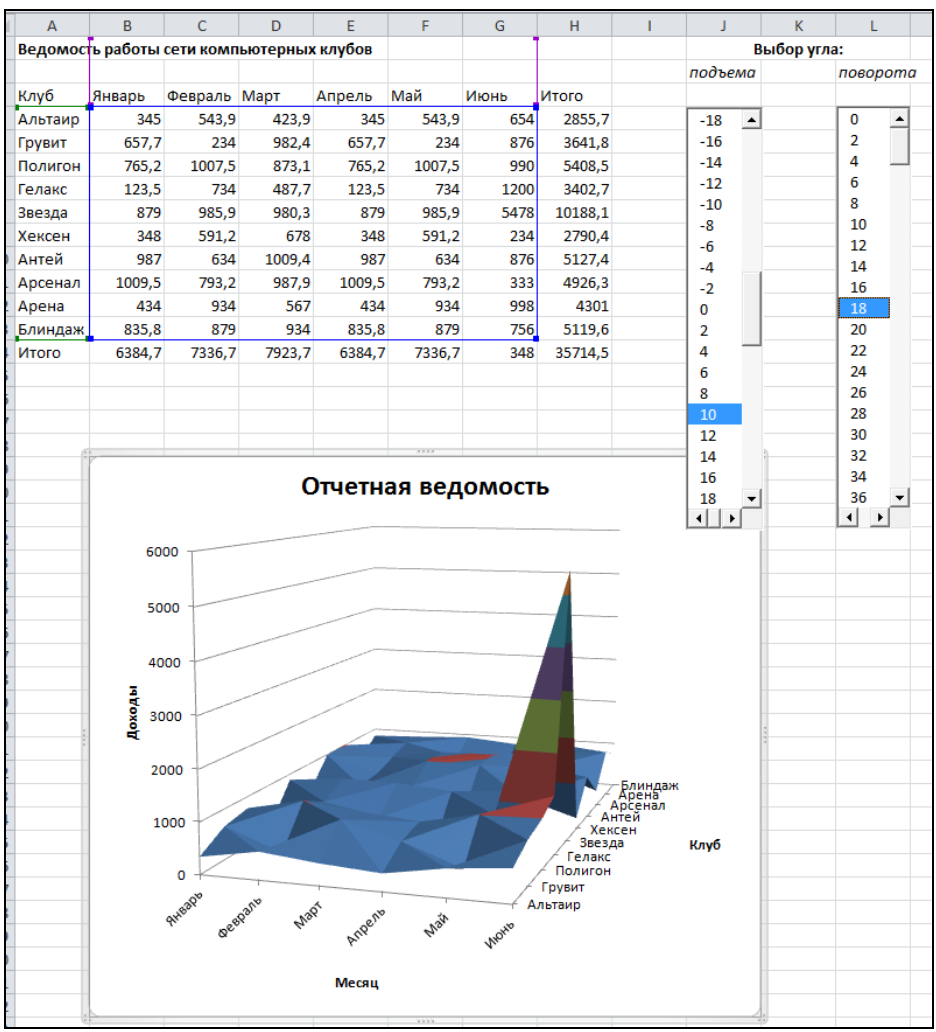


Рис. 6.11. Вращение поверхности

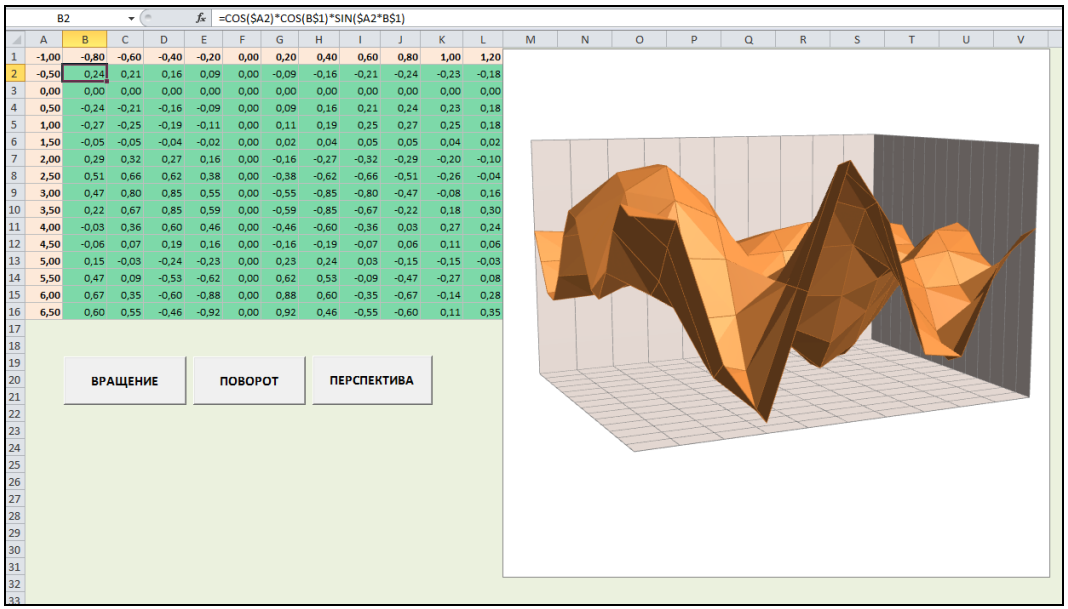


Рис. 6.12. Вращение поверхности по трем измерениям

Приведем в качестве примера еще одну поверхность, у которой вращение происходит уже по трем измерениям (рис. 6.12).

Для реализации данного примера выполните следующие действия.

1. Подготовьте диапазон для построения диаграммы. Введите соответственно в ячейки **A1:A16** и **B1:L1** данные для изменения значений по оси *x* (интервал: от –1 до 6,5; шаг: 0,5) и по оси *y* (интервал: от –1 до 1,2; шаг: 0,2). В ячейку **B2** введите формулу $z = \cos x \cos y \sin(xy)$:

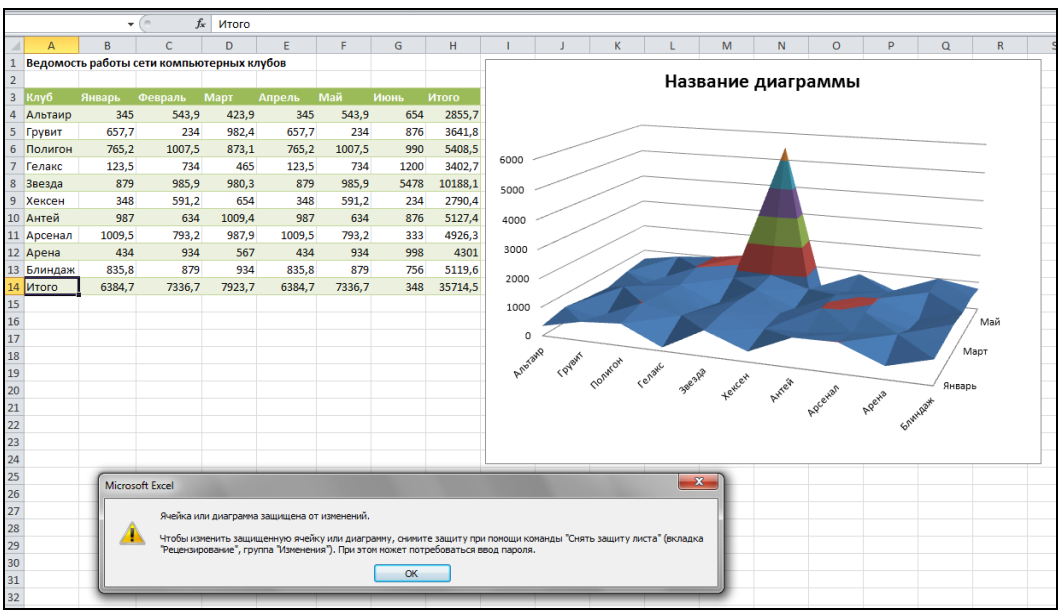
   ```
   =COS($A2)*COS(B$1)*SIN($A2*B$1)
   ```

   Скопируйте формулу на диапазон **B2:L16**.
2. Выделите диапазон **A1:L16** и постройте поверхность с использованием имеющихся средств Microsoft Excel: перейдите на вкладку **Вставка** ленты, далее в группе **Диаграммы** щелкните по кнопке со списком **Другие диаграммы** и выберите тип — **Поверхность**.
3. Отформатируйте полученную поверхность с использованием возможностей контекстных вкладок **Работа с диаграммами**.
4. Разместите на рабочем листе три элемента управления **Кнопка** и установите соответственно значения для свойства `Caption`: `ВРАЩЕНИЕ` (для `CommandButton1`), `ПОВОРОТ` (для `CommandButton2`) и `ПЕРСПЕКТИВА` (для `CommandButton3`).
5. Введите в стандартном модуле и модуле рабочего листа **Вращение_поверхности** код процедур, поддерживающих вращение поверхности по трем измерениям (см. файл *9-Вращение поверхности по трем измерениям.xlsm* на компакт-диске).

Устанавливаем защиту на вложенную в рабочий лист диаграмму

Если вы хотите защитить построенную на рабочем листе диаграмму, а также данные рабочего листа вне некоторого определенного диапазона, воспользуйтесь методом `Password` объекта `Worksheets` со значением параметра `UserInterfaceOnly` равным `True`. Это обеспечит защиту рабочего листа и позволит вводить данные только в указанные ячейки. Так, например, для установки защиты на все объекты рабочего листа кроме диапазона **B4:G13** (рис. 6.13, см. также файл *10-Защита рабочего листа.xlsm* на компакт-диске) добавьте на лист модуля `ЭтаКнига` программный код листинга 6.8.

Чтобы разрешить ввод данных на рабочий лист, необходимо перейти на вкладку **Рецензирование** ленты и в группе **Изменения** щелкнуть по кнопке **Снять защиту листа**. Как правило, от вас потребуется ввод пароля (в нашем случае это "pass").



Клуб	Январь	Февраль	Март	Апрель	Май	Июнь	Итого
Альтаир	345	543,9	423,9	345	543,9	654	2855,7
Грувит	657,7	234	982,4	657,7	234	876	3641,8
Полигон	765,2	1007,5	873,1	765,2	1007,5	990	5408,5
Гелакс	123,5	734	465	123,5	734	1200	3402,7
Звезда	879	985,9	980,3	879	985,9	5478	10188,1
Хексен	348	591,2	654	348	591,2	234	2790,4
Антей	987	634	1009,4	987	634	876	5127,4
Арсенал	1009,5	793,2	987,9	1009,5	793,2	333	4926,3
Арена	434	934	567	434	934	998	4301
Блиндаж	835,8	879	934	835,8	879	756	5119,6
Итого	6384,7	7336,7	7923,7	6384,7	7336,7	348	35714,5

Рис. 6.13. Сообщение, отображаемое при попытке ввести данные в ячейки вне диапазона **B4:G13**

Листинг 6.8. Установка защиты на внедренную диаграмму. Модуль `ЭтаКнига`

```
Private Sub Workbook_Open()
   SetPtotection
End Sub

Private Sub SetPtotection()
```

```
   On Error Resume Next
   Worksheets("Ведомость").Range("B4:G13").Locked = False
   Worksheets("Ведомость").Protect Password:="pass", UserInterfaceOnly:=True
End Sub
```

ПРИМЕЧАНИЕ

Если вы добавили защиту на лист рабочей книги, в которой содержатся элементы управления, попытка воспользоваться элементами управления приводит к ошибке проекта. Снимите защиту листа и выполните обычные действия.

Защита диаграммы, расположенной на отдельном листе

Для защиты диаграммы, расположенной на отдельном листе (объект `Chart`), используется метод `Protect`:

```
Protect(Password, DrawingObjects, Contents, Scenarios, UserInterfaceOnly)
```

- *`Password`* — задает пароль защиты.
- *`DrawingObjects`* — устанавливает защиту на графические объекты.
- *`Contents`* — определяет защиту всей диаграммы.
- *`Scenarios`* — задает защиту на сценарии.
- *`UserInterfaceOnly`* — защита пользовательского интерфейса, но не макросов. Если этот параметр опущен, то защита применяется как к интерфейсу, так и макросам.

Снять защиту можно методом `Unprotect`:

```
Unprotect(Password)
```

Листинг 6.9 демонстрирует, как установить защиту на диаграмму, которая расположена на отдельном листе (рис. 6.14, см. также файл *11-Защита диаграммы на отдельном листе.xlsm* на компакт-диске), а также добавить блокировку для пользователя на все ячейки рабочего листа **Лист1**, за исключением диапазона **B4:G13**.

Листинг 6.9. Установка защиты на диаграмму. Модуль ЭтаКнига

```
Private Sub Workbook_Open()
   SetPtotection
End Sub

Private Sub SetPtotection()
    On Error Resume Next
    Charts(1).Protect Password:="d1", UserInterfaceOnly:=True
    Worksheets("Лист1").Range("B4:G13").Locked = False
    Worksheets("Лист1").Protect Password:="1"
End Sub
```

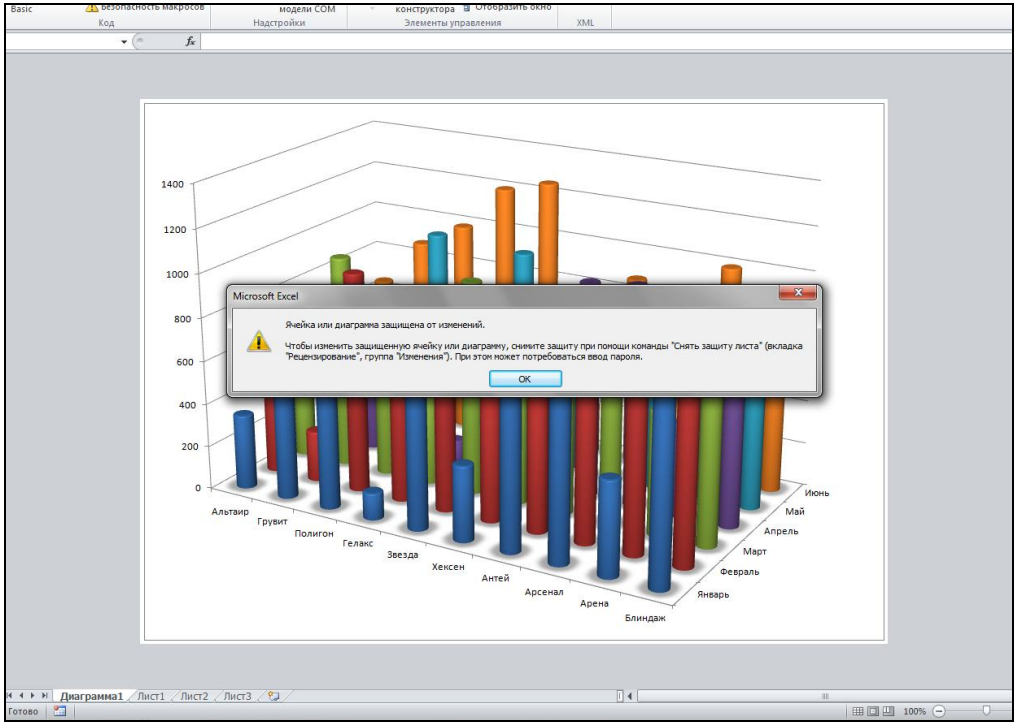


Рис. 6.14. Установка защиты для диаграммы на отдельном листе

Немного о событиях и диаграммах

По умолчанию события ассоциируются с диаграммами, которые созданы на отдельных листах диаграмм. Приведем некоторые примеры обработки событий, связанные с диаграммами.

Пусть нам необходимо, чтобы изменялся цвет области и самой диаграммы, расположенной на отдельном листе, в зависимости от того, где будет произведен щелчок мышью. Для обработки данного события можно использовать программный код, приведенный в модуле `Диаграмма1` (см. файл *12-Привязка Изменения цвета к Листу диаграммы.xlsm* на компакт-диске).

Приведем еще один пример, связанный с обработкой события (перемещение мыши) на листах диаграмм. Пусть нам необходимо в надписи, расположенной на листе диаграмм, выводить дополнительные примечания, связанные с точками данных (рис. 6.15, см. также файл *13-Привязка текста примечаний к Листу диаграммы.xlsm* на компакт-диске).

Для реализации данного примера разместите на рабочем листе **Лист1** соответствующую таблицу данных и таблицу примечаний (рис. 6.16), которые можно взять из файла *13а-Данные для примера_13-Привязка текста примечаний к Листу диаграммы* в папке Glava_6 на прилагаемом компакт-диске, а на листе модуля `Диаграмма1` введите программный код из листинга 6.10.

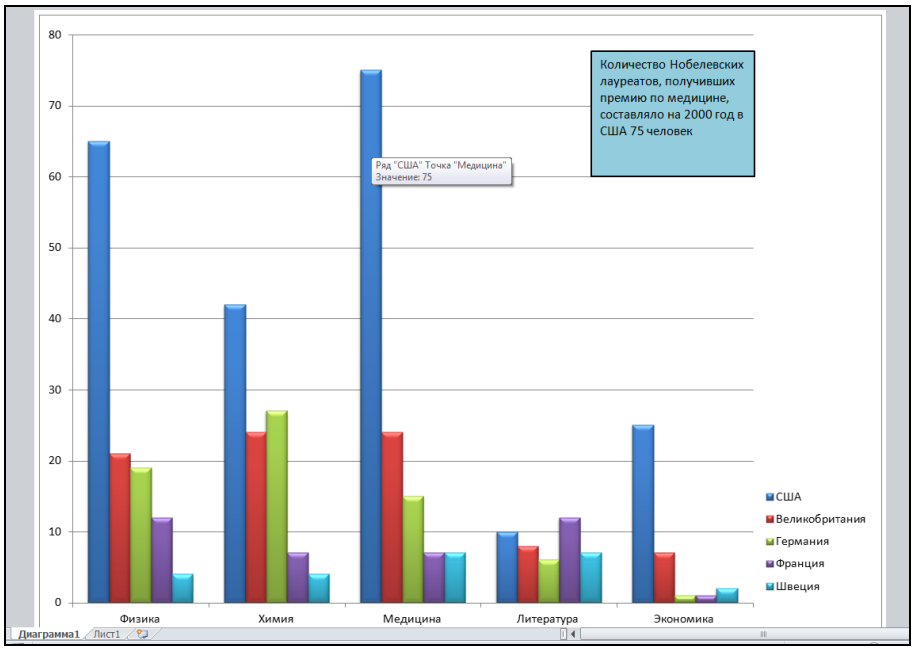


Рис. 6.15. Привязка текста примечаний к листу диаграммы

B9 fx Количество Нобелевских лауреатов, получивших премию по физике, составляло на 2000 год в США 65 человек

	A	B	C	D	E	F	G
1	**Страна**	**Физика**	**Химия**	**Медицина**	**Литература**	**Экономика**	
2	**США**	65	42	75	10	**25**	
3	**Великобритания**	21	24	24	8	**7**	
4	**Германия**	19	27	15	6	**1**	
5	**Франция**	12	7	7	12	**1**	
6	**Швеция**	4	4	7	7	**2**	
7							
8	**Примечание**						
9		Количество Нобелевских лауреатов, получивших премию по физике, составляло на 2000 год в США 65 человек	Количество Нобелевских лауреатов, получивших премию по физике, составляло на 2000 год в Великобритании 21 человек	Количество Нобелевских лауреатов, получивших премию по физике, составляло на 2000 год в Германии 19 человек	Количество Нобелевских лауреатов, получивших премию по физике, составляло на 2000 год во Франции 12 человек	Количество Нобелевских лауреатов, получивших премию по физике, составляло на 2000 год в Швеции 4 человека	
10		Количество Нобелевских лауреатов, получивших премию по химии, составляло на 2000 год в США 42 человека	Количество Нобелевских лауреатов, получивших премию по химии, составляло на 2000 год в Великобритании 24 человека	Количество Нобелевских лауреатов, получивших премию по химии, составляло на 2000 год в Германии 27 человек	Количество Нобелевских лауреатов, получивших премию по химии, составляло на 2000 год во Франции 7 человек	Количество Нобелевских лауреатов, получивших премию по химии, составляло на 2000 год в Швеции 4 человека	
11		Количество Нобелевских лауреатов, получивших премию по медицине, составляло на 2000 год в США 75 человек	Количество Нобелевских лауреатов, получивших премию по медицине, составляло на 2000 год в Великобритании 24 человека	Количество Нобелевских лауреатов, получивших премию по медицине, составляло на 2000 год в Германии 15 человек	Количество Нобелевских лауреатов, получивших премию по медицине, составляло на 2000 год во Франции 7 человек	Количество Нобелевских лауреатов, получивших премию по медицине, составляло на 2000 год в Швеции 7 человек	
12		Количество Нобелевских лауреатов, получивших премию по литературе, составляло на 2000 год в США 10 человек	Количество Нобелевских лауреатов, получивших премию по литературе, составляло на 2000 год в Великобритании 8 человек	Количество Нобелевских лауреатов, получивших премию по литературе, составляло на 2000 год в Германии 6 человек	Количество Нобелевских лауреатов, получивших премию по литературе, составляло на 2000 год во Франции 12 человек	Количество Нобелевских лауреатов, получивших премию по литературе, составляло на 2000 год в Швеции 7 человек	
13		Количество Нобелевских лауреатов, получивших премию по экономике, составляло на 2000 год в США 25 человек	Количество Нобелевских лауреатов, получивших премию по экономике, составляло на 2000 год в Великобритании 7 человек	Количество Нобелевских лауреатов, получивших премию по экономике, составляло на 2000 год в Германии 1 человек	Количество Нобелевских лауреатов, получивших премию по экономике, составляло на 2000 год во Франции 1 человек	Количество Нобелевских лауреатов, получивших премию по экономике, составляло на 2000 год в Швеции 2 человека	

Диаграмма1 Лист1

Рис. 6.16. Исходные данные для диаграммы и текста примечаний

Листинг 6.10. Привязка текста примечаний к листу диаграммы. Модуль Диаграмма 1

```
Option Explicit
Private Sub Chart_MouseMove(ByVal Button As Long, ByVal Shift As Long, _
                            ByVal X As Long, ByVal Y As Long)
    Dim RowId As Long
    Dim rg1 As Long, rg2 As Long
    Dim MyText As String

    On Error Resume Next
    ActiveChart.GetChartElement X, Y, RowId, rg1, rg2
    If RowId = xlSeries Then
        MyText = Sheets("Лист1").Range("Примечание").Offset(rg2, rg1)
    Else
        MyText = "Для получения справки о Нобелевских лауреатах " & _
                 "выберите столбец диаграммы"
    End If
    ActiveChart.Shapes(1).TextFrame.Characters.Text = MyText
End Sub
```

Привязка события к вложенным в рабочий лист диаграммам

Если диаграмма расположена на рабочем листе, напрямую связать с ней события нельзя. В данном случае вам потребуется ряд дополнительных действий, после чего можно будет привязывать требуемые события к диаграммам, находящимся на рабочих листах. В качестве небольшого примера проделайте следующие действия.

1. Перейдите к редактору VBA и создайте модуль класса `MyEventClassModule` (команда **Insert | Class Module**, свойству `Name` присвойте значение `MyEventClassModule`).
2. Объявите в модуле класса `MyEventClassModule` переменную типа `Chart` с ключевым словом `WithEvents` (листинг 6.11). После этого в редакторе кода модуля класса в списке объектов появится объект `MyEventClassModule`, а в списке событий — все связанные с диаграммами события.

Листинг 6.11. Привязка событий к вложенным в рабочий лист диаграммам. Модуль класса MyEventClassModule (вариант 1)

```
Public WithEvents MyChartClass As Chart
```

3. В редакторе кода наберите код обработки тех событий, которые необходимы для бизнес-логики вашего проекта. Например (листинг 6.12), свяжем нажатие кнопки мыши с сообщением "Подумайте о Ваших дальнейших действиях!".

Листинг 6.12. Привязка событий к вложенным в рабочий лист диаграммам. Модуль класса MyEventClassModule (вариант 2)

```
Private Sub MyChartClass_MouseDown(ByVal Button As Long, _
      ByVal Shift As Long, ByVal x As Long, ByVal y As Long)
    MsgBox "Подумайте о Ваших дальнейших действиях!"
End Sub
```

4. Свяжите событие с вложенной диаграммой. Например, это можно сделать на этапе открытия книги, добавив в модуль `ЭтаКнига` код (листинг 6.13, см. также файл *14-Привязка события к вложенной в рабочий лист диаграмме.xlsm* на компакт-диске), который ассоциирует первую из вложенных в первый рабочий лист диаграмм с объектом `MyChartClass`. Теперь при нажатии кнопки мыши на этой диаграмме отобразится сообщение "Подумайте о Ваших дальнейших действиях!".

Листинг 6.13. Привязка событий к вложенным в рабочий лист диаграммам. Модуль `ЭтаКнига`

```
Dim MyClassModule As New MyEventClassModule
Sub ChartInit()
   Set MyClassModule.MyChartClass = Worksheets(1).ChartObjects(1).Chart
End Sub

Private Sub Workbook_Open()
   ChartInit
End Sub
```

Изменение типа диаграммы при помощи контекстного меню

В качестве примера использования событий диаграмм, расположенных на рабочем листе, рассмотрим проект, в котором при щелчке правой кнопкой мыши на диаграмме отображается контекстное меню, состоящее из следующих команд: **Гистограмма**, **График**, **Кольцевая**, **С областями**, **Линейчатая** и **Коническая**. Выбор одной из них приводит к изменению на соответствующий тип диаграммы (рис. 6.17).

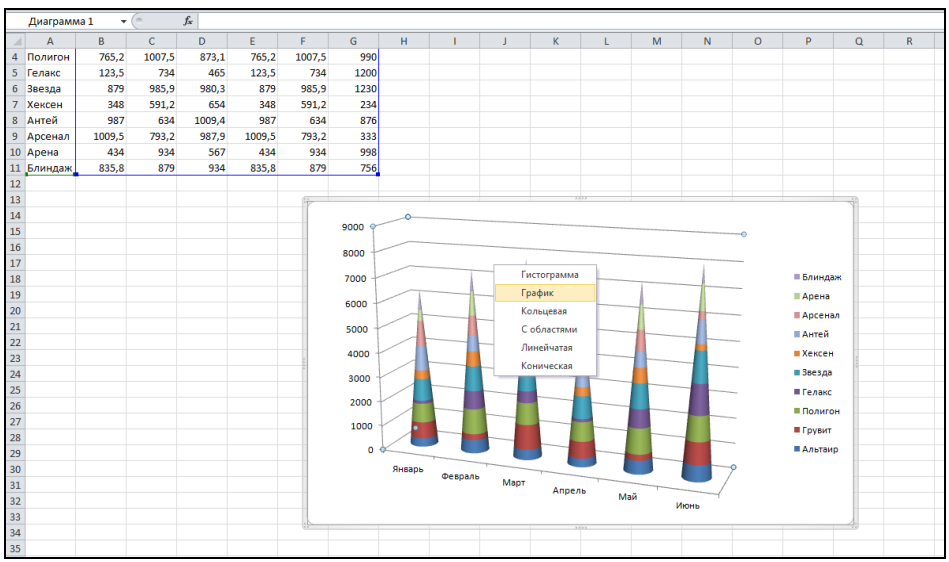


Рис. 6.17. Изменение типа диаграммы при помощи контекстного меню

Прежде всего, у нас происходит обработка события — "щелчок правой кнопкой на внедренной диаграмме". Поэтому необходимо создать класс (в данном случае `ChartEventClass`), в котором реализуется обрабатывающий это событие код. При открытии рабочей книги связываем экземпляр класса `ChartEventClass` с конкретной диаграммой, создаем контекстное меню, а также назначаем командам этого меню соответствующие макросы, которые и осуществляют изменение типа диаграммы (модули `ЭтаКнига`, стандартный модуль и модуль класса `ChartEventClass` в файле *15-Изменение типа диаграммы с помощью контекстного меню.xlsm* на компакт-диске).

Наши итоги

Итак, Microsoft Office Excel 2010 предлагает достаточно большой набор средств для построения и форматирования диаграмм различного вида. Кроме того, разные свойства и методы объектов `Chart` и `ChartObject` позволяют создавать пользователям собственные проекты, включающие как внедренные диаграммы, так и диаграммы на отдельных листах. После того как вами изучен материал данной главы и проделаны все предлагаемые примеры, вы значительно продвинулись в искусстве построения диаграмм. Теперь вы:

- знаете основные элементы диаграмм и типы диаграмм, основные операции, которые можно производить с диаграммами;
- можете быстро визуализировать имеющиеся данные средствами Microsoft Office Excel 2010;
- имеете представление о семействах `ChartObjects`, `Charts` и объектах `ChartObject`, `Chart`;
- ориентируетесь в основных свойствах, методах и событиях объекта `Chart`;
- с помощью VBA можете построить диаграмму, изменить диапазон построения диаграммы, ее тип;
- умеете визуализировать и позиционировать поверхности;
- можете добавлять линию тренда на диаграмму;
- устанавливать защиту диаграммы;
- обрабатывать события, связанные с диаграммами.

Все полученные знания и умения, несомненно, помогут вам при создании собственных проектов, в которых требуется визуализация числовых данных.

Глава 7

Обрабатываем списки в Microsoft Excel

В Microsoft Excel существует возможность обработки больших массив однотипных данных — так называемых *списков*. Различные экономические, финансовые, учетные и многие другие задачи требуют именно такого представления данных. Работа с подготовленным списком в MS Excel может осуществляться по трем направлениям. Прежде всего, это *сортировка*, т. е. выстраивание данных в нужном порядке. *Отбор данных* — извлечение записей данных из списка — требует уже кроме самого списка наличия дополнительных требований (критериев) отбора. И, наконец, *анализ данных* предполагает использование специальных средств, которые позволяют производить определенные манипуляции с данными и получать обработанную информацию. В этой главе мы остановимся на сортировке и отборе данных и продемонстрируем некоторые возможности, связанные с этими направлениями.

> ***Примечание***
>
> Файлы рассмотренных в данной главе примеров можно найти в папке Glava_7 на компакт-диске.

Что нужно знать о списке?

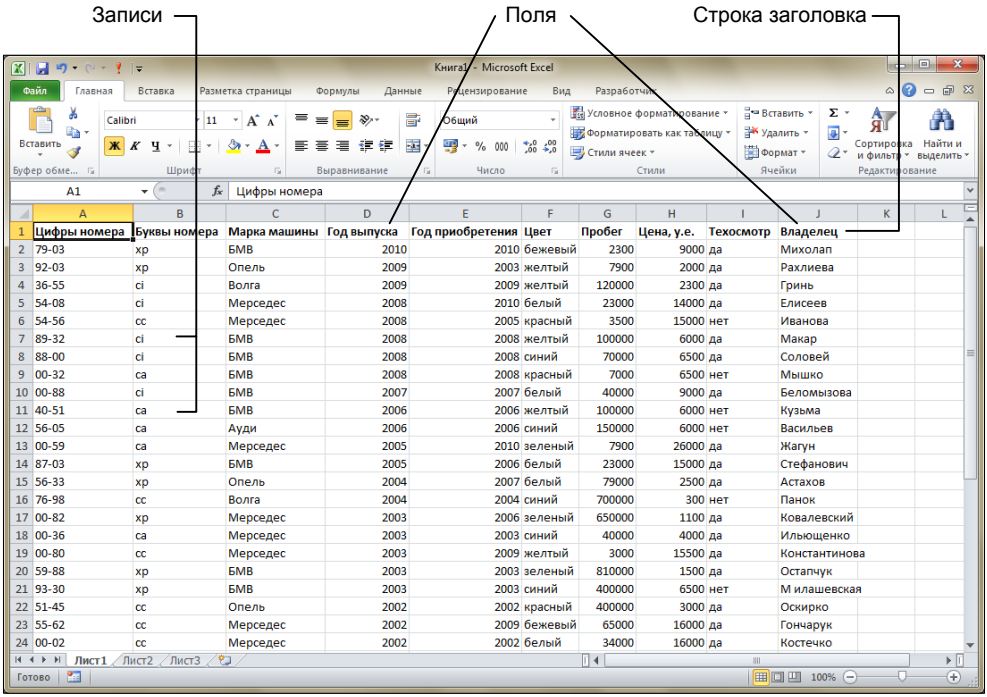
Списки в MS Excel — это таблицы (рис. 7.1), строки которых содержат однородную информацию. Строки таблицы называются *записями*, а столбцы — *полями записей*. Столбцам присваиваются уникальные имена полей, которые заносятся в первую строку списка — *строку заголовка*.

Как правило, при работе со списками приходится сталкиваться со следующими диапазонами.

Диапазон данных — область, где хранятся данные списка. Данные, связанные друг с другом, записываются в отдельные строки, каждому столбцу соответствует свое поле списка с уникальным именем поля.

Диапазон критериев — область на рабочем листе, в которой задаются критерии для поиска информации. В диапазоне критериев указываются имена полей и отводится область для записи условий отбора.

Диапазон для извлечения — область, в которую MS Excel копирует выбранные данные из списка. Этот диапазон может быть расположен на том же листе, что и список, а может — и на другом листе рабочей книги.



Цифры номера	Буквы номера	Марка машины	Год выпуска	Год приобретения	Цвет	Пробег	Цена, у.е.	Техосмотр	Владелец
79-03	хр	БМВ	2010	2010	бежевый	2300	9000	да	Михолап
92-03	хр	Опель	2009	2003	желтый	7900	2000	да	Рахлиева
36-55	ci	Волга	2009	2009	желтый	120000	2300	да	Гринь
54-08	ci	Мерседес	2008	2010	белый	23000	14000	да	Елисеев
54-56	cc	Мерседес	2008	2005	красный	3500	15000	нет	Иванова
89-32	ci	БМВ	2008	2008	желтый	100000	6000	да	Макар
88-00	ci	БМВ	2008	2008	синий	70000	6500	да	Соловей
00-32	ca	БМВ	2008	2008	красный	7000	6500	нет	Мышко
00-88	ci	БМВ	2007	2007	белый	40000	9000	да	Беломызова
40-51	ca	БМВ	2006	2006	желтый	100000	6000	нет	Кузьма
56-05	ca	Ауди	2006	2006	синий	150000	6000	нет	Васильев
00-59	ca	Мерседес	2005	2010	зеленый	7900	26000	да	Жагун
87-03	хр	БМВ	2005	2006	белый	23000	15000	да	Стефанович
56-33	хр	Опель	2004	2007	белый	79000	2500	да	Астахов
76-98	cc	Волга	2004	2004	синий	700000	300	нет	Панок
00-82	хр	Мерседес	2003	2006	зеленый	650000	1100	да	Ковалевский
00-36	ca	Мерседес	2003	2003	синий	40000	4000	да	Ильющенко
00-80	cc	Мерседес	2003	2009	желтый	3000	15500	да	Константинова
59-88	хр	БМВ	2003	2003	зеленый	810000	1500	да	Остапчук
93-30	хр	БМВ	2003	2003	синий	400000	6500	нет	М илашевская
51-45	cc	Опель	2002	2002	красный	400000	3000	да	Оскирко
55-62	cc	Мерседес	2002	2009	бежевый	65000	16000	да	Гончарук
00-02	cc	Мерседес	2002	2002	белый	34000	16000	да	Костечко

Рис. 7.1. Список MS Excel

Ввод данных в список осуществляется, например, непосредственно в ячейки рабочего листа, т. е. в пустые строки под заголовком, либо с использованием формы данных (щелкните по кнопке со списком на панели быстрого доступа и выберите команду **Другие команды**; далее в окне **Параметры Excel** выберите слева категорию **Панель быстрого доступа**, а справа в списке **Выбрать команды из** — вариант **Все команды** и в расположенном ниже списке найдите команду **Формы**, которую и добавьте на панель быстрого доступа).

Как указывалось ранее, с данными, помещенными в список, можно выполнять: сортировку, отбор данных и анализ данных.

Сортируем данные

Итак, сортировка позволяет выстраивать данные в алфавитном или цифровом порядке по возрастанию или убыванию. Microsoft Excel может сортировать строки, а также столбцы списков рабочих листов. При сортировке текста в таблице можно сортировать как один столбец, так и таблицу целиком. Кроме того, можно выполнить сортировку по нескольким словам или полям в одном столбце таблицы, а также — для выделенного диапазона списка.

Для осуществления быстрой сортировки по требуемому полю, вам достаточно установить указатель ячейки в нужный столбец списка с данными, перейти на вклад-

ку **Данные** и в группе **Сортировка и фильтр** щелкнуть либо по кнопке **Сортировка от минимального к максимальному**, либо по кнопке **Сортировка от максимального к минимальному**. С другой стороны, если вы на вкладке **Данные** в группе **Сортировка и фильтр** выберите кнопку **Сортировка**, то откроется окно, в котором можно задать ключи сортировки (столбцы или строки), порядок сортировки и некоторые другие параметры (рис. 7.2).

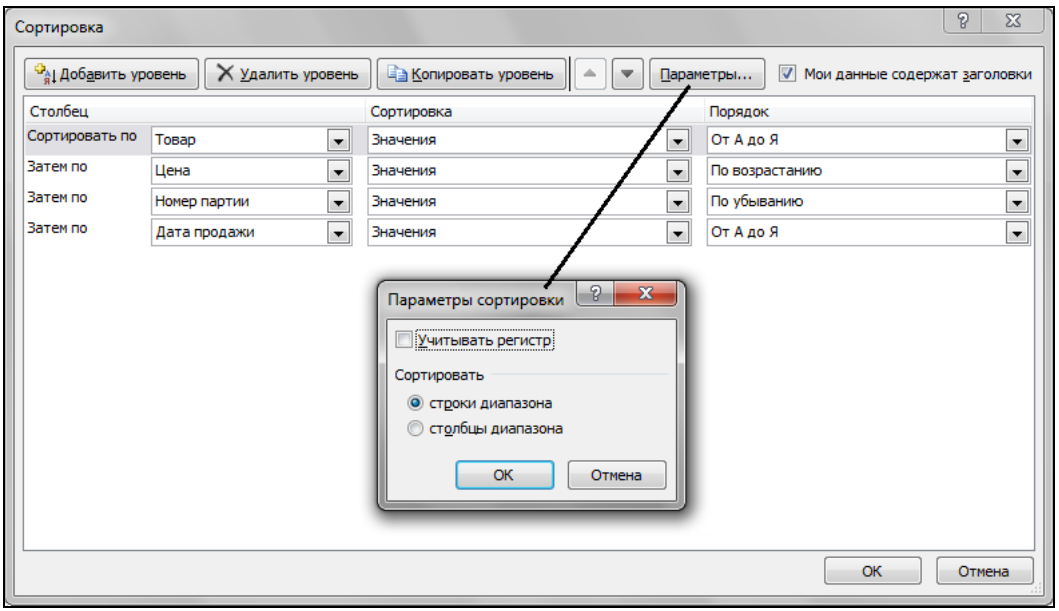


Рис. 7.2. Окно **Сортировка** с открытым окном **Параметры сортировки**

Примечание

На вкладке **Главная** в группе команд **Редактирование** находится кнопка со списком **Сортировка и фильтр**, команды которой также позволяют выполнить сортировку списка данных.

В MS Excel используется следующий порядок сортировки:

1. числа (от –∞ до +∞);
2. текст и формулы;
3. значение ЛОЖЬ;
4. значение ИСТИНА;
5. значения ошибок;
6. пустые значения.

При использовании *сортировки* следует помнить:

1. Порядок сортировки данных в MS Excel зависит от национальных настроек Windows.

2. Если необходимо упорядочить числовые величины в алфавитном порядке, следует перед числовыми величинами ставить апостроф либо отформатировать числа как текст, либо ввести число как формулу (например, `="345"`).
3. При сортировке списков, содержащих формулы, следует помнить, что относительные ссылки в формулах при перемещении записей могут привести к неправильным результатам, поэтому в списках лучше использовать формулы с абсолютными ссылками.
4. Для возврата к первоначальному списку следует ввести перед списком дополнительное индексное поле, содержащее возрастающую с любым шагом числовую последовательность (например, 1, 2, 3, ...). Таким образом, выделив ячейку в данном столбце и отсортировав список по возрастанию, мы вернемся к первоначальному списку.
5. Сортировка в особом порядке позволит упорядочить данные в каком-либо заданном порядке (например, дни недели, месяцы и т. д.). Для этого используется окно **Списки** (рис. 7.3), которое вызывается командой **Настраиваемый список** в окне **Сортировка** (см. рис. 7.2): щелкните по полю со списком в столбце **Порядок** данного окна и выберите команду **Настраиваемый список**.

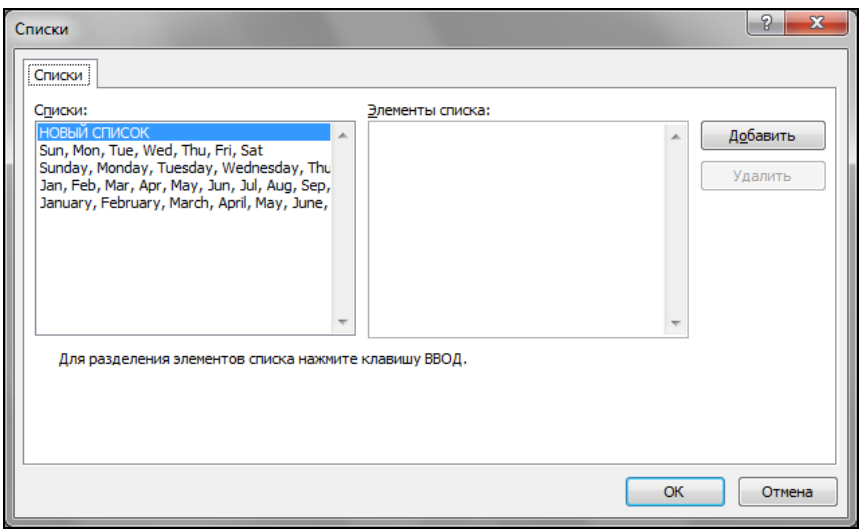


Рис. 7.3. Окно **Списки**

6. Даты и время должны быть введены в соответствующем формате либо с помощью функций даты или времени, т. к. для сортировки таких данных MS Excel использует внутреннее представление этих величин.
7. Сортировка по многим полям задается в окне **Сортировка** (см. рис. 7.2), начиная с самого верхнего уровня. Изменить уровень сортировки можно с использованием кнопок **Вверх** ▲, **Вниз** ▼ указанного окна.
8. MS Excel может сортировать не только строки, но и столбцы, а также выделенный диапазон в списке. Кроме того, сортировку можно проводить с учетом регистра введенных символов (см. рис. 7.2).

Используем VBA для сортировки данных

А теперь рассмотрим несколько примеров, связанных с сортировкой данных с использованием программ на VBA.

Для осуществления сортировки данных с учетом до трех критериев, по которым производится сортировка, применяется метод `Sort`, который позволяет сортировать строки списков, сводных таблиц и баз данных, а также столбцы рабочих листов:

```
expression.Sort(Key1, Order1, Key2, Type, Order2, Key3, Order3, Header,
↳OrderCustom, MatchCase, Orientation, SortMethod, DataOption1,
↳DataOption2, DataOption3)
```

- *expression* — ссылка на ячейку диапазона или на сам диапазон, который будет сортироваться.
- *Key1* — необязательный параметр, задающий ссылку на первое упорядочиваемое поле.
- *Order1* — необязательный параметр, определяющий порядок сортировки поля, заданного параметром *Key1*. Допустимыми значениями являются следующие константы `XlSortOrder`: `xlAscending` (возрастающий порядок), `xlDescending` (убывающий).
- *Key2* — необязательный параметр, задающий ссылку на второе упорядочиваемое поле.
- *Type* — необязательный параметр, специфицирующий элементы, которые должны быть отсортированы. Используется только со сводными таблицами.
- *Order2* — необязательный параметр, определяющий порядок сортировки поля, заданного параметром *Key2*. Допустимыми значениями являются константы `XlSortOrder`.
- *Key3* — необязательный параметр, задающий ссылку на третье упорядочиваемое поле.
- *Order3* — необязательный параметр, определяющий порядок сортировки поля, заданного параметром *Key3*. Допустимыми значениями являются константы `XlSortOrder`.
- *Header* — необязательный параметр, специфицирующий, имеется ли в первой строке списка заголовок. Допустимыми значениями являются следующие константы `XlYesNoGuess`: `xlYes` (первая строка диапазона содержит заголовок, который не сортируется), `xlNo` (первая строка диапазона не содержит заголовок, по умолчанию считается данное значение), `xlGuess` (MS Excel решает сам, имеет ли список заголовок).
- *OrderCustom* — необязательный параметр, задающий пользовательский порядок сортировки. Является целым числом, указывающим порядковый номер списка, используемого в качестве шаблона сортировки.
- *MatchCase* — необязательный параметр, указывающий, надо ли учитывать регистры букв при сортировке.
- *Orientation* — необязательный параметр, специфицирующий ориентацию сортировки. Допустимыми значениями являются следующие константы `XlSortOrientation`: `xlTopToBottom` (сортировка осуществляется сверху вниз, т. е. по строкам), `xlLeftToRight` (сортировка осуществляется слева направо, т. е. по столбцам).

- *SortMethod* — необязательный параметр, указывающий метод сортировки. Применяется для таких языков, как китайский, японский.
- *DataOption1* — необязательный параметр, специфицирующий, как должен сортироваться текст в поле, заданном параметром *Key1*. Допустимыми значениями являются следующие константы `XlSortDataOption`: `xlSortTextAsNumbers` (числовые и текстовые данные сортируются вместе), `xlSortNormal` (числовые и текстовые данные сортируются по отдельности).
- *DataOption2* — необязательный параметр, специфицирующий, как должен сортироваться текст в поле, заданном параметром *Key2*. Допустимыми значениями являются константы `XlSortDataOption`.
- *DataOption3* — необязательный параметр, специфицирующий, как должен сортироваться текст в поле, заданном параметром *Key3*. Допустимыми значениями являются константы `XlSortDataOption`.

Сортировка данных списка по трем полям

Итак, рассмотрим пример, который продемонстрируют использование метода `Sort`. Пусть на листе рабочей книги имеется список данных об автомобилях и их владельцах. Разместите на рабочем листе два элемента управления **Кнопка** (CommandButton), один из которых будет отвечать за сортировку по возрастанию по трем произвольным столбцам, а второй соответственно — за сортировку по убыванию (рис. 7.4, см. также файл *1-Сортировка диапазона по трем полям.xlsm* на компакт-диске).

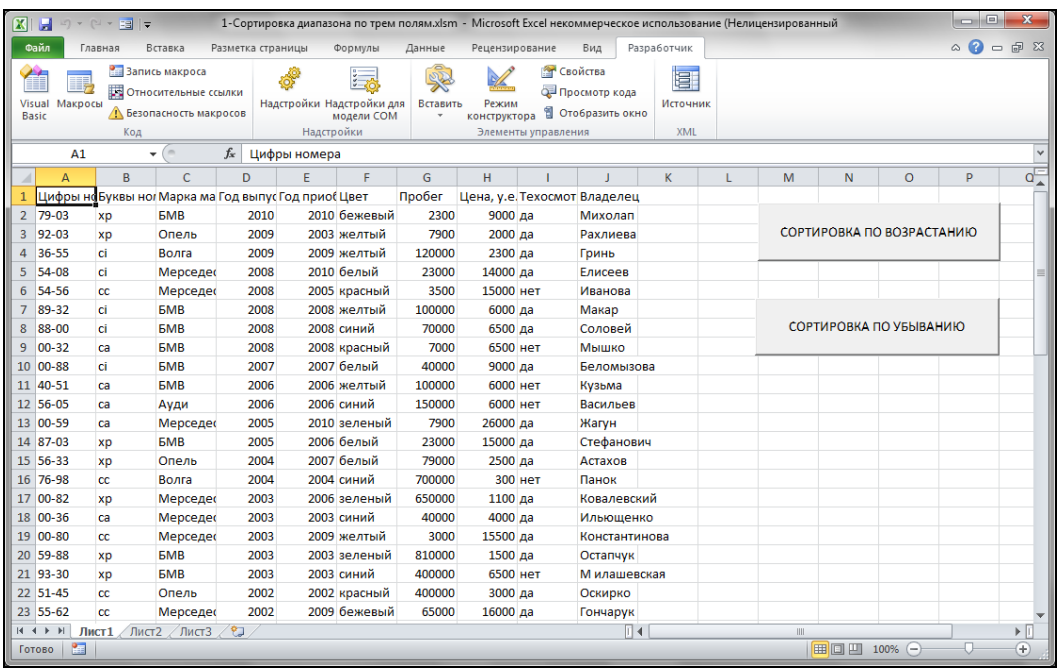


Рис. 7.4. Сортировка списка по трем полям

Измените соответственно значения свойства `Caption` для первой кнопки (`CommandButton1`) на `СОРТИРОВКА ПО ВОЗРАСТАНИЮ`, а для второй кнопки (`CommandButton2`) на `СОРТИРОВКА ПО УБЫВАНИЮ`. Введите на листе стандартного модуля программный код, представленный в листинге 7.1, на листе модуля `Лист1` — программный код, представленный в листинге 7.2.

Листинг 7.1. Сортировка по трем полям по возрастанию и по убыванию. Стандартный модуль

```
Sub Sort_Up()
  Range("A1").Select
  x = InputBox("Введите адрес столбца для сортировки по первому полю", _
               "Ввод диапазона")
  y = InputBox("Введите адрес столбца для сортировки по второму полю", _
               "Ввод диапазона")
  z = InputBox("Введите адрес столбца для сортировки по третьему полю", _
               "Ввод диапазона")
  Selection.Sort Key1:=Range(x), Order1:=xlAscending, _
                 Key2:=Range(y), Order2:=xlAscending, _
                 Key3:=Range(z), Order3:=xlAscending, Header:=xlYes
  Range("A1").Select
End Sub
Sub Sort_Down()
  Range("A1").Select
  x = InputBox("Введите адрес столбца для сортировки по первому полю", _
               "Ввод диапазона")
  y = InputBox("Введите адрес столбца для сортировки по второму полю", _
               "Ввод диапазона")
  z = InputBox("Введите адрес столбца для сортировки по третьему полю", _
               "Ввод диапазона")
  Selection.Sort Key1:=Range(x), Order1:=xlDescending, _
                 Key2:=Range(y), Order2:=xlDescending, _
                 Key3:=Range(z), Order3:=xlDescending, Header:=xlYes
  Range("A1").Select
End Sub
```

Листинг 7.2. Сортировка по трем полям по возрастанию и по убыванию. Модуль `Лист1`

```
Private Sub CommandButton1_Click()
  Sort_Up
End Sub

Private Sub CommandButton2_Click()
  Sort_Down
End Sub
```

Сортировка данных на защищенном листе

Данные на защищенном листе можно сортировать, если они находятся в диапазоне, значение свойства `Locked` которого установлено равным `False`, и если параметр `AllowSorting` метода `Protect` установлен равным `True`. Свойство "только для чтения" `AllowSorting` объекта `Protection` возвращает значение этого параметра. Например, код из листинга 7.3 разрешает сортировку диапазона **A1:A5** на защищенном листе, если она ранее не была установлена.

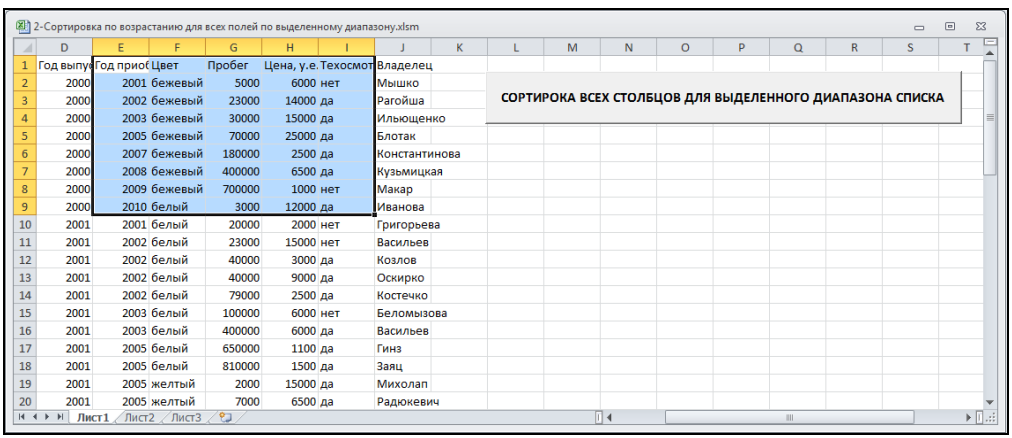
Листинг 7.3. Сортировка на защищенном листе

```
Sub DemoAllowSorting()
   ActiveSheet.Unprotect
   Range("A1:A5").Locked = False
   If ActiveSheet.Protection.AllowSorting Then
        ActiveSheet.Protect AllowSorting:=True
   End If
End Sub
```

Сортировка данных в выделенном диапазоне

А теперь рассмотрим пример сортировки данных списка, но только для предварительно выделенного диапазона списка. В качестве исходных данных возьмем также таблицу об автомобилях и их владельцах. На рабочий лист добавим элемент управления **Кнопка** (CommandButton), для которой заменим значение свойства `Caption` на `СОРТИРОКА ВСЕХ СТОЛБЦОВ ДЛЯ ВЫДЕЛЕННОГО ДИАПАЗОНА СПИСКА` (рис. 7.5, см. также файл *2-Сортировка по возрастанию для всех полей по выделенному диапазону.xlsm* на компакт-диске).

Добавьте на лист стандартного модуля программный код, представленный в листинге 7.4, на лист модуля `Лист1` — программный код, связанный с нажатием кнопки (листинг 7.5).



	D	E	F	G	H	I	J
1	Год выпуска	Год приобр	Цвет	Пробег	Цена, у.е.	Техосмотр	Владелец
2	2000	2001	бежевый	5000	6000	нет	Мышко
3	2000	2002	бежевый	23000	14000	да	Рагойша
4	2000	2003	бежевый	30000	15000	да	Ильющенко
5	2000	2005	бежевый	70000	25000	да	Блотак
6	2000	2007	бежевый	180000	2500	да	Константинова
7	2000	2008	бежевый	400000	6500	да	Кузьмицкая
8	2000	2009	бежевый	700000	1000	нет	Макар
9	2000	2010	белый	3000	12000	да	Иванова
10	2001	2001	белый	20000	2000	нет	Григорьева
11	2001	2002	белый	23000	15000	нет	Васильев
12	2001	2002	белый	40000	3000	да	Козлов
13	2001	2002	белый	40000	9000	да	Оскирко
14	2001	2002	белый	79000	2500	да	Костечко
15	2001	2003	белый	100000	6000	нет	Беломызова
16	2001	2003	белый	400000	6000	да	Васильев
17	2001	2005	белый	650000	1100	да	Гинз
18	2001	2005	белый	810000	1500	да	Заяц
19	2001	2005	желтый	2000	15000	да	Михолап
20	2001	2005	желтый	7000	6500	да	Радюкевич

Рис. 7.5. Сортировка списка по выделенному диапазону

Листинг 7.4. Сортировка данных списка по выделенному диапазону. Стандартный модуль

```
Sub SortColumnUp()
   Dim k As Long
   For k = Selection.Columns.Count To 1 Step -1
       Selection.Sort Key1:=Selection.Cells(2, k), Order1:=xlAscending, _
                      Header:=xlGuess, Orientation:=xlTopToBottom
   Next k
End Sub
```

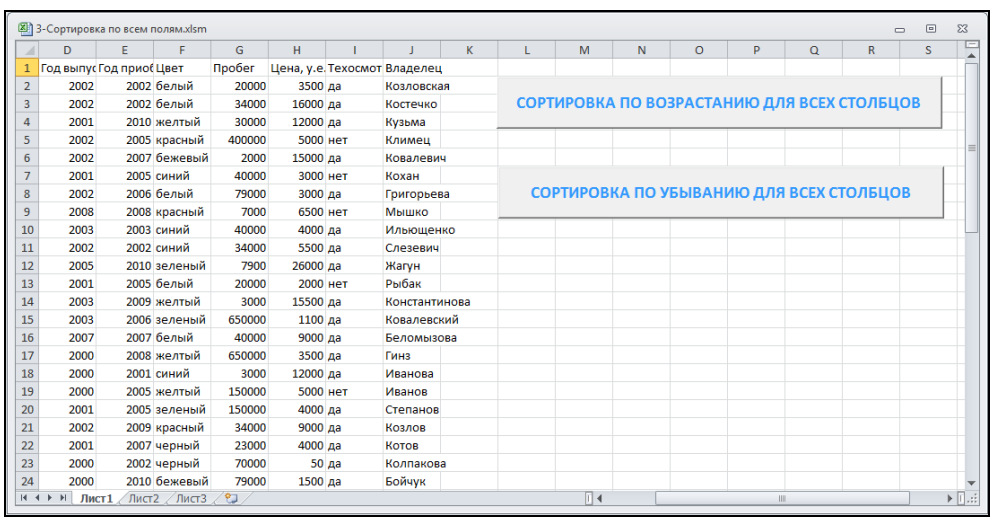
Листинг 7.5. Сортировка данных списка по выделенному диапазону. Модуль Лист1

```
Private Sub CommandButton1_Click()
  SortColumnUp
End Sub
```

Сортировка всех столбцов списка

В Microsoft Office Excel 2010 с использованием окна **Сортировка** (рис. 7.2) вы можете задать последовательно, как и по каким столбцам будет проводиться сортировка данных. Если же вам необходимо отсортировать весь список по возрастанию или убыванию, последовательно учитывая все столбцы, которые принадлежат списку, то проделайте приведенный далее пример.

1. Итак, для начала разместите на рабочем листе список с данными. Пусть, например, это также будет наш список с автовладельцами.
2. Добавьте два элемента управления **Кнопка** (CommandButton). Первая кнопка будет отвечать за сортировку по возрастанию для всех столбцов списка, а вторая — за сортировку по убыванию (рис. 7.6).



	D	E	F	G	H	I	J
1	Год выпус	Год приоб	Цвет	Пробег	Цена, у.е.	Техосмот	Владелец
2	2002	2002	белый	20000	3500	да	Козловская
3	2002	2002	белый	34000	16000	да	Костечко
4	2001	2010	желтый	30000	12000	да	Кузьма
5	2002	2005	красный	400000	5000	нет	Климец
6	2002	2007	бежевый	2000	15000	да	Ковалевич
7	2001	2005	синий	40000	3000	нет	Кохан
8	2002	2006	белый	79000	3000	да	Григорьева
9	2008	2008	красный	7000	6500	нет	Мышко
10	2003	2003	синий	40000	4000	да	Ильющенко
11	2002	2002	синий	34000	5500	да	Слезевич
12	2005	2010	зеленый	7900	26000	да	Жагун
13	2001	2005	белый	20000	2000	нет	Рыбак
14	2003	2009	желтый	3000	15500	да	Константинова
15	2003	2006	зеленый	650000	1100	да	Ковалевский
16	2007	2007	белый	40000	9000	да	Беломызова
17	2000	2008	желтый	650000	3500	да	Гинз
18	2000	2001	синий	3000	12000	да	Иванова
19	2000	2005	желтый	150000	5000	нет	Иванов
20	2001	2005	зеленый	150000	4000	да	Степанов
21	2002	2009	красный	34000	9000	да	Козлов
22	2001	2007	черный	23000	4000	да	Котов
23	2000	2002	черный	70000	50	да	Колпакова
24	2000	2010	бежевый	79000	1500	да	Бойчук

Рис. 7.6. Сортировка всего списка

3. Измените значения свойства `Caption` для кнопок соответственно на `СОРТИРОКА ПО ВОЗРАСТАНИЮ ДЛЯ ВСЕХ СТОЛБЦОВ` и `СОРТИРОКА ПО УБЫВАНИЮ ДЛЯ ВСЕХ СТОЛБЦОВ`.
4. Добавьте в стандартный модуль программный код, осуществляющий сортировку, а в лист модуля `Лист1` — программный код, обрабатывающий нажатие кнопок (см. файл *3-Сортировка по всем полям.xlsm* на компакт-диске).

Фильтруем данные

Для поиска и фильтрации данных в MS Excel 2010 существуют автофильтр и расширенный фильтр. *Автофильтр*, включая фильтр по выделенному, используется для простых условий отбора; а *расширенный фильтр* — для более сложных условий отбора.

В отфильтрованном списке отображаются только строки, отвечающие условиям отбора, заданным для столбца. При фильтрации порядок записей в списке не изменяется, строки, которые не удовлетворяют критерию фильтрации, временно скрываются. Отфильтрованные строки можно редактировать, форматировать и выводить на печать, а также создавать на их основе диаграммы, не перемещая их и не изменяя порядок строк.

Отметим, что бывают следующие *критерии поиска*.

- *По точному соответствию* — отбор данных по точному соответствию. Математические вычисления и логические операции (И, ИЛИ) не используются.
- *На основе сравнения* — используют различные операции сравнения (`=200` (число), `=` (ищут пустые поля), `>`, `>=`, `<`, `<=`, `<>`). Данные операции можно применять к различным форматам данных (числа, текст, т. е. символы, дата, время и др.).
- *По близкому соответствию с использованием образца* — задают образец поиска, используя символы шаблона — `?` или (и) `*`. Для нахождения полей, содержащих просто `?` или `*`, перед ними ставится тильда: `~?` или `~*`.
- *По поиску соответствия с использованием множественного критерия* с операциями И и ИЛИ — поиск данных по нескольким условиям.

Как найти данные с использованием автофильтра?

Если вы пользуетесь автофильтром для поиска данных в списке, то справа от подписей столбцов в фильтруемом списке появляются стрелки автофильтра . У отфильтрованных данных номера строк окрашиваются в синий цвет, а стрелка автофильтра превращается в стрелку с воронкой (фильтром) возле тех полей, по которым произведен отбор данных. В раскрывающемся списке автофильтров можно (рис. 7.7):

- провести сортировку данных, выбрав команду **Сортировка от минимального к максимальному** или **Сортировка от максимального к минимального**;
- провести сортировку данных по цвету, выбрав команду **Сортировка по цвету**; в раскрывающемся списке возле этой команды можно задать пользовательскую сортировку;
- удалить фильтр по полю, используя команду **Удалить фильтр с *"Имя_поля"***;
- установить фильтр по цвету, используя команду **Фильтр по цвету**;

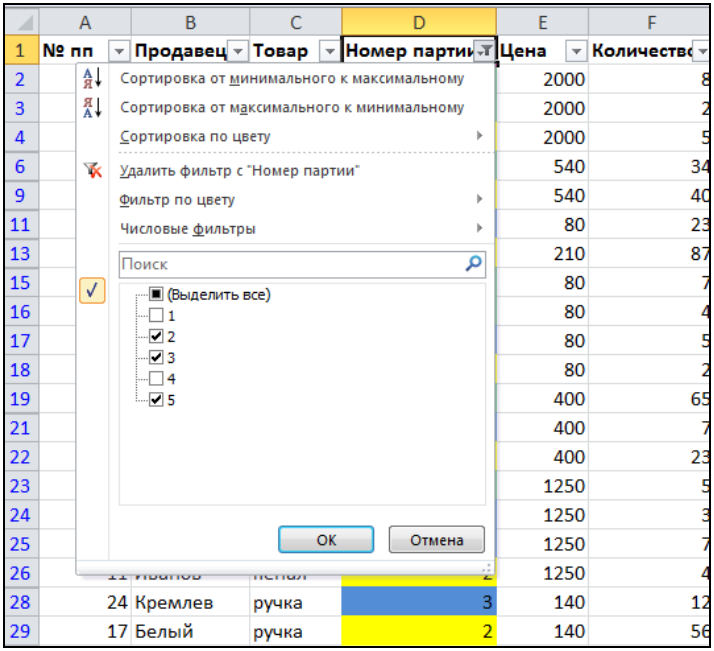


Рис. 7.7. Раскрывающийся список автофильтра

- ❒ указать числовой диапазон для отбора данных, выбрав команду **Числовые фильтры**; здесь же в раскрывающемся списке возле этой команды можно, выбрав команду **Настраиваемый фильтр**, открыть окно **Пользовательский автофильтр**, где задается простой критерий отбора данных, содержащий до двух условий;
- ❒ провести поиск данных с использованием поля **Поиск**, введя критерий по точному соответствию или критерий по близкому соответствию с использованием образца;
- ❒ задать значение поля для поиска точного соответствия.

Поиск с помощью автофильтра производится в следующем порядке.

1. Установить указатель ячейки в список данных.
2. Перейти на вкладку **Главная** ленты, в группе команд **Редактирование** щелкнуть по кнопке со стрелкой **Сортировка и фильтр** и выбрать команду **Фильтр** или же воспользоваться командой **Фильтр**, расположенной в группе **Сортировка и фильтр** на вкладке **Данные** ленты. Возле каждого поля строки заголовка списка появятся стрелки автофильтра ▾.
3. Перейти к необходимому полю.
4. Выбрать требуемый критерий поиска или прибегнуть к пользовательскому автофильтру. Здесь отметим, что диалоговое окно **Пользовательский автофильтр** (рис. 7.8) позволяет быстро задать более сложное условие, чем простое сравнение. В левом верхнем раскрывающемся списке выбирается операция сравнения (в данном случае выбрано **больше или равно**), в правом — выбирается или вводится значение (введено `9000`). Затем, если необходимо, выбирается

один из переключателей **И**, **ИЛИ**, задается вторая операция сравнения и значение. В данном случае выбран переключатель **И**, операция сравнения **меньше** и введено значение `20000`. Далее нажимается кнопка **OK** для получения результата пользовательской фильтрации.

5. Для включения в критерий другого поля, следует возвратиться к п. 3.

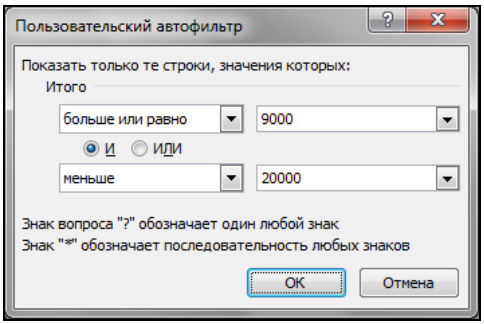


Рис. 7.8. Диалоговое окно **Пользовательский автофильтр**

Как программировать автофильтрацию?

Метод `AutoFilter` объекта `Range` позволяет программным способом задать выполнение автофильтрации списка по критериям, указанным в параметрах:

expression.AutoFilter(*Field*, *Criteria1*, *Operator*, *Criteria2*, *VisibleDropDown*)

- *expression* — ссылка на ячейку диапазона или на сам диапазон, который будет фильтроваться.
- *Field* — необязательный параметр, задающий номер поля списка, в котором производится фильтрация данных. Нумерация производится с крайнего левого поля, причем первое поле имеет номер 1.
- *Criteria1* и *Criteria2* — необязательные параметры, специфицирующие два возможных критерия фильтрации поля. Допускается использование строковой константы, например `"101"`, и знаков отношений >, <, >=, <=, =, <>. Используйте = для фильтрации пустых полей, а <> для фильтрации непустых полей. Если параметр опущен, то критерием является значение `All`. Если значение параметра *Operator* равно `xlTop10Items`, то параметр *Criteria1* определяет число отображаемых элементов (например, `"30"`).
- *Operator* — необязательный параметр, допустимыми значениями которого могут быть следующие константы `XlAutoFilterOperator`: `xlAnd` (для логического объединения первого и второго критериев фильтрации), `xlOr` (для логического сложения первого и второго критериев), `xlTop10Items` (для отображения первых десяти элементов поля), `xlBottom10Items` (для отображения последних десяти элементов поля), `xlTop10Percent` (для отображения первых 10% элементов поля), `xlBottom10Percent` (для отображения последних 10% элементов поля).
- *VisibleDropDown* — необязательный параметр, принимающий логические значения. Определяет, отображаются ли раскрывающиеся списки. По умолчанию параметр принимает значение `True`.

При написании программ на VBA, которые связаны с использованием автофильтра, вам помогут также следующие методы и свойства.

- Метод `AutoFilter` объекта `Range`, примененный без параметров, приводит к отображению или скрытию кнопок со стрелками автофильтра  .
- Метод `ShowAllData` объекта `Worksheet` позволяет отобразить на рабочем листе все данные списка — отфильтрованные и не отфильтрованные, например:
  `Worksheets("Заказы").ShowAllData`
- Свойство "только для чтения" `AutoFilterMode` объекта `Worksheet` принимает логические значения. Оно возвращает значение `True`, если на рабочем листе имеются кнопки со стрелками автофильтра.
- Свойство "только для чтения" `FilterMode` объекта `Worksheet` принимает логические значения. Оно возвращает значение `True`, если на рабочем листе имеются отфильтрованные данные со скрытыми строками.

Объект `AutoFilter` инкапсулирует в себе данные об используемом на рабочем листе автофильтре. Этот объект возвращается свойством `AutoFilter` объекта `Worksheet`. Основные свойства объекта `AutoFilter` приведены в табл. 7.1, а свойства объекта `Filter` — в табл. 7.2.

***Таблица 7.1.** Основные свойства объекта `AutoFilter`*

Свойство	Описание
`Filters`	Возвращает семейство `Filters` объектов `Filter`, т. е. всех фильтров, образующих данный автофильтр. Свойства объекта `Filter` приведены в табл. 7.2. У семейства `Filters` основными свойствами являются `Count` и `Item`, которые возвращают число элементов и конкретный элемент семейства
`Range`	Возвращает диапазон, к которому применен фильтр

***Таблица 7.2.** Свойства объекта `Filter`*

Свойство	Описание
`On`	Возвращает значение `True`, если фильтр установлен
`Criteria1`	Возвращает первый критерий фильтра
`Criteria2`	Возвращает второй критерий фильтра
`Operator`	Возвращает оператор фильтра. Допустимыми значениями являются константы `XlAutoFilterOperator`

Отметим также, что свойство `EnableAutoFilter` рабочего листа позволяет управлять доступом к раскрывающимся спискам автофильтра на защищенном листе. Если его значение равно `True`, то раскрывающиеся списки автофильтра достижимы для пользователя даже в случае установки защиты на рабочем листе. При этом должен быть включен режим защиты только пользовательского интерфейса:

```
ActiveSheet.EnableAutoFilter = True
ActiveSheet.Protect Contents:=True, UserInterfaceOnly:=True
```

Приведем пример, использующий объект `AutoFilter` (см. файл *4-Определение автофильтра.xlsm* на компакт-диске). В стандартном модуле располагается код процедуры, в которой сначала определяется, имеется ли на рабочем листе автофильтр, а затем, в случае его наличия, находится общее количество фильтров и критерии отбора для установленных пользовательских автофильтров. В модуле рабочего листа **Лист1** омещен код для обработки нажатия кнопки.

Пример приложения, фильтрующего данные

Рассмотрим создание приложения, которое производит фильтрацию данных для указанного столбца в списке данных. Для начала создайте рабочую книгу и переименуйте рабочий **Лист1** в **Список**. Удалите остальные листы рабочей книги. Расположите на листе **Список** подготовленные в виде таблицы данные по автомобилям и их владельцам. При щелчке правой кнопкой мыши на заголовке таблицы должно отображаться контекстное меню с единственной командой **Filter**. Выбор этой команды приведет к вызову диалогового окна, запрашивающего ввод адреса для ячейки заголовка списка данных. После указания адреса ячейки должна отобразиться форма со списком, в котором перечислены значения данного столбца, и флажком **Фильтрация**. При установленном флажке на рабочем листе фильтруются данные по выбранному в списке значению. При снятом флажке — фильтрация удаляется (рис. 7.9).

Итак, создайте форму, на ней расположите список и флажок. Используя окно **Properties**, установите значение свойства `Name` формы равным `frmFilter`. В модулях формы, `ЭтаКнига` и стандартном, наберите соответствующий код (см. файл *5-Форма с Автофильтром по требуемому полю.xlsm* на компакт-диске).

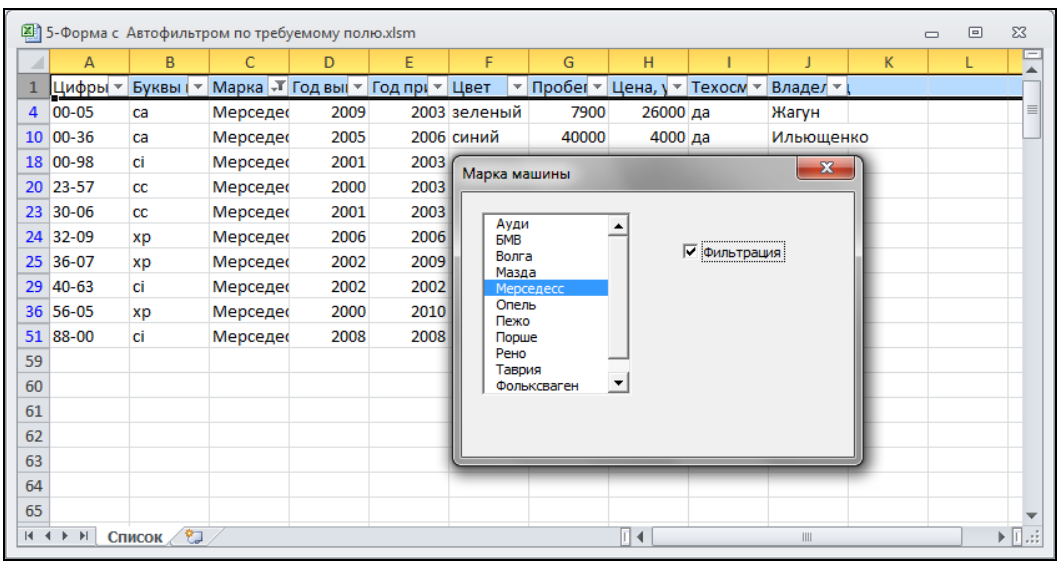


Рис. 7.9. Таблица **автомобили** и окно ***Имя_поля***

В модуле формы имеются три процедуры:

- процедура обработки события `Initialize` формы устанавливает значения свойства `Caption` флажка и формы, а также заполняет список. Список заполняется из того столбца таблицы, адрес первой ячейки которого вводится пользователем, причем в процессе заполнения повторяющиеся данные из этого списка опускаются;
- процедура обработки события `Change` списка при выборе элемента из списка, если флажок **Фильтрация** установлен, фильтрует таблицу, причем в случае если кнопки автофильтра еще не были выведены в таблицу, то эта процедура сначала инициализирует автофильтр, а затем производит фильтрацию;
- процедура обработки события `Change` флажка в зависимости от состояния флажка либо производит фильтрацию, вызывая процедуру обработки события `Change` списка, либо удаляет автофильтр.

В модуле `ЭтаКнига` имеются три процедуры:

- процедура обработки события `Open` объекта `Workbook` создает контекстное меню, в котором имеется единственный элемент **Filter** и с которым связывается процедура `DoFilter`;
- процедура обработки события `SheetBeforeRightClick` объекта `Workbook` отображает контекстное меню при щелчке пользователем правой кнопкой на первой строке листа **Список**;
- процедура обработки события `BeforeClose` объекта `Workbook` удаляет из книги контекстное меню при закрытии книги.

В стандартном модуле имеется единственная процедура `DoFilter`, которая отображает окно **Имя_поля.**

Как использовать расширенный фильтр?

Расширенный фильтр предоставляет пользователю больше возможностей по заданию критериев отбора данных, учитывающих операции И, ИЛИ и вычисляемые критерии. Поиск с помощью расширенного фильтра производится в соответствии со следующими рекомендациями.

- Подготовка диапазона критериев для расширенного фильтра:
  - верхняя строка содержит заголовки полей, по которым будет производиться отбор (точное соответствие заголовкам полей списка);
  - условия критериев поиска записываются в пустые строки под подготовленной строкой заголовка, причем следует учитывать, что:
    - ◊ выполнение условия "И" требует располагать критерии поиска рядом в одной строке;
    - ◊ выполнения условия "ИЛИ" требует располагать критерии в разных строках;
    - ◊ поиск по вычисляемому критерию включает формулы (пользовательские или функции MS Excel), в которых аргументами являются поля списка. Вычисляемый критерий располагается под некоторым заголовком, например, **Условие**, который не должен совпадать ни с одним именем поля списка. Для ссылок на список используются относительные ссылки, которые указывают на верхние записи в диапазоне данных списка. Ссылки на ячейки вне списка берутся абсолютными. Вычисляемый критерий может

включать несколько функций и зависеть от нескольких полей. Результатом вычисления критерия должно быть логическое значение ИСТИНА или ЛОЖЬ (расширенный фильтр отбирает записи с ИСТИНА);

- ◊ в случае сложного условия поиск данных осуществляется по составному критерию с "И" и "ИЛИ". Критерий следует составлять с помощью логических функций И(), ИЛИ(), НЕ().

❒ Указатель ячейки помещается в список (выделяется весь необходимый список или часть диапазона списка).

❒ Перейти на вкладку **Данные** ленты и в группе команд **Сортировка и фильтр** нажать кнопку **Дополнительно**. Работа с окном **Расширенный фильтр** (рис. 7.10):

- указать в группе **Обработка** место, куда будут помещаться результаты выборки данных;
- в поле **Исходный диапазон** пометить весь список, подлежащий фильтрации (как правило, после помещения указателя ячейки в список, данный диапазон выделяется по умолчанию);
- в поле **Диапазон условий** указать подготовленный диапазон условий отбора записей (удобно выделить мышью на рабочем листе);
- если отобранные записи необходимо поместить в другое место, в поле **Поместить результат в диапазон** указать соответствующее место для отобранных данных;
- для отбора уникальных записей (без повторений) необходимо установить флажок **Только уникальные записи**.

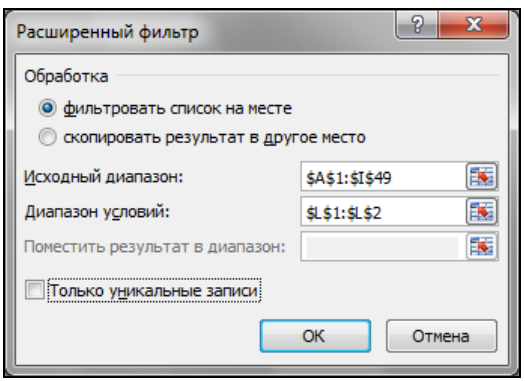


Рис. 7.10. Окно **Расширенный фильтр**

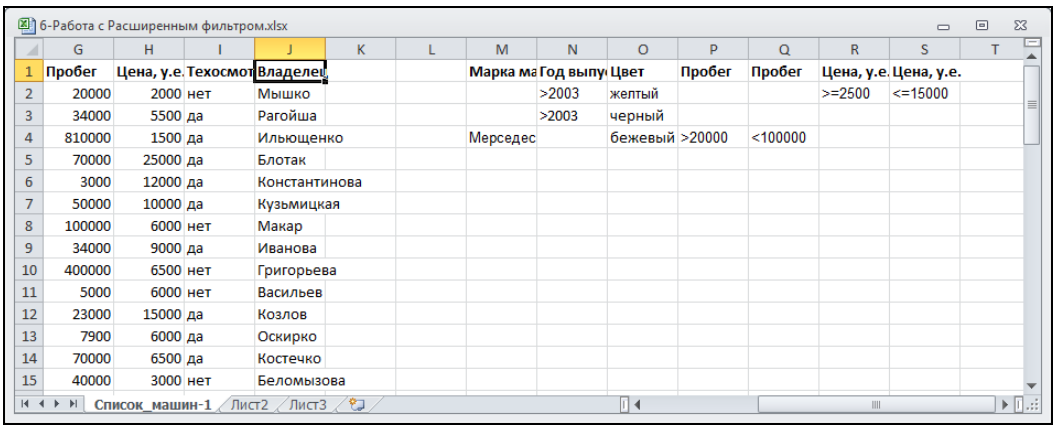
❒ Результаты расширенной фильтрации отображаются на данном рабочем листе или копируются в другое место.

Рассмотрим несколько примеров, которые демонстрируют возможности расширенного фильтра (см. файл *6-Работа с Расширенным фильтром.xlsm* на компакт-диске).

Пример 1. Определить, имеются ли в предложенном списке желтые или черные машины, год выпуска которых больше 2003 и цена находится в диапазоне 2500—

15 000 у. е., или бежевые "Мерседесы", пробег которых более 20 000 км, но менее 100 000 км.

1. Итак, откройте список, подлежащий фильтрации (список располагается в диапазоне **A1:J58**, строка заголовка — в диапазоне **A1:J1**, рабочий лист — **Список_машин-1**).
2. Сформируйте диапазон критериев для расширенного фильтра в соответствии с рис. 7.11.



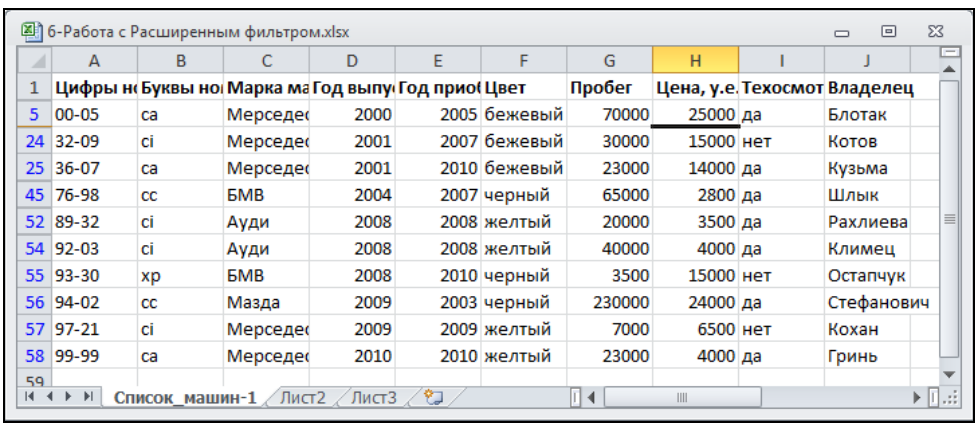
6-Работа с Расширенным фильтром.xlsx

	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Пробег	Цена, у.е.	Техосмот	Владелец			Марка ма	Год выпу	Цвет	Пробег	Пробег	Цена, у.е.	Цена, у.е.	
2	20000	2000	нет	Мышко				>2003	желтый			>=2500	<=15000	
3	34000	5500	да	Рагойша				>2003	черный					
4	810000	1500	да	Ильющенко			Мерседес		бежевый	>20000	<100000			
5	70000	25000	да	Блотак										
6	3000	12000	да	Константинова										
7	50000	10000	да	Кузьмицкая										
8	100000	6000	нет	Макар										
9	34000	9000	да	Иванова										
10	400000	6500	нет	Григорьева										
11	5000	6000	нет	Васильев										
12	23000	15000	да	Козлов										
13	7900	6000	да	Оскирко										
14	70000	6500	да	Костечко										
15	40000	3000	нет	Беломызова										

Список_машин-1 Лист2 Лист3

Рис. 7.11. Диапазон критериев для расширенного фильтра

3. Перейдите на вкладку **Данные** ленты и в группе команд **Сортировка и фильтр** нажмите кнопку **Дополнительно**.
4. В окне **Расширенный фильтр** (рис. 7.10) укажите требуемые диапазоны и параметры отбора записей списка. Нажмите кнопку **OK**.
5. Отфильтрованные данные приведены на рис. 7.12.



6-Работа с Расширенным фильтром.xlsx

	A	B	C	D	E	F	G	H	I	J
1	Цифры но	Буквы но	Марка ма	Год выпу	Год прио	Цвет	Пробег	Цена, у.е.	Техосмот	Владелец
5	00-05	ca	Мерседес	2000	2005	бежевый	70000	25000	да	Блотак
24	32-09	ci	Мерседес	2001	2007	бежевый	30000	15000	нет	Котов
25	36-07	ca	Мерседес	2001	2010	бежевый	23000	14000	да	Кузьма
45	76-98	cc	БМВ	2004	2007	черный	65000	2800	да	Шлык
52	89-32	ci	Ауди	2008	2008	желтый	20000	3500	да	Рахлиева
54	92-03	ci	Ауди	2008	2008	желтый	40000	4000	да	Климец
55	93-30	xp	БМВ	2008	2010	черный	3500	15000	нет	Остапчук
56	94-02	cc	Мазда	2009	2003	черный	230000	24000	да	Стефанович
57	97-21	ci	Мерседес	2009	2009	желтый	7000	6500	нет	Кохан
58	99-99	ca	Мерседес	2010	2010	желтый	23000	4000	да	Гринь
59										

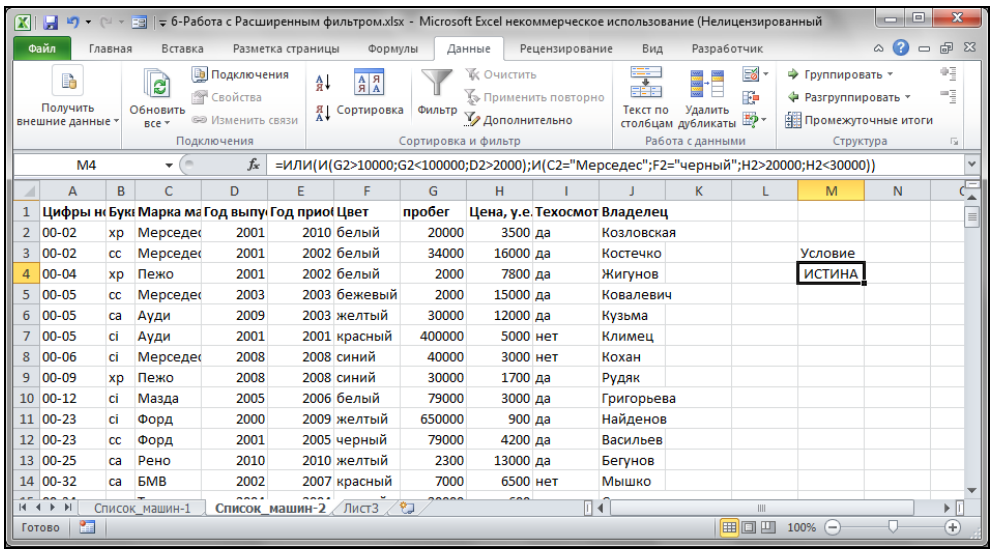
Список_машин-1 Лист2 Лист3

Рис. 7.12. Данные, отобранные расширенным фильтром

Пример 2. Определить, имеются ли в списке машины, год выпуска которых больше 2000 и пробег более 10 000 км, но менее 100 000 км, или черные "Мерседесы", цена которых более 20 000 у. е., но менее 30 000 у. е.

1. Откройте список, подлежащий фильтрации (список располагается в диапазоне **A1:J133**, строка заголовка — в диапазоне **A1:J1**, рабочий лист — **Список_машин-2**).
2. Сформируйте вычисляемый критерий для расширенного фильтра в диапазоне **M3:M4**. В ячейку **M3** добавьте слово **Условие**. В ячейку **M4** введите формулу (рис. 7.13):

```
=ИЛИ(И(G2>10000;G2<100000;D2>2000);И(C2="Мерседес";F2="черный";H2>20000;
H2<30000))
```



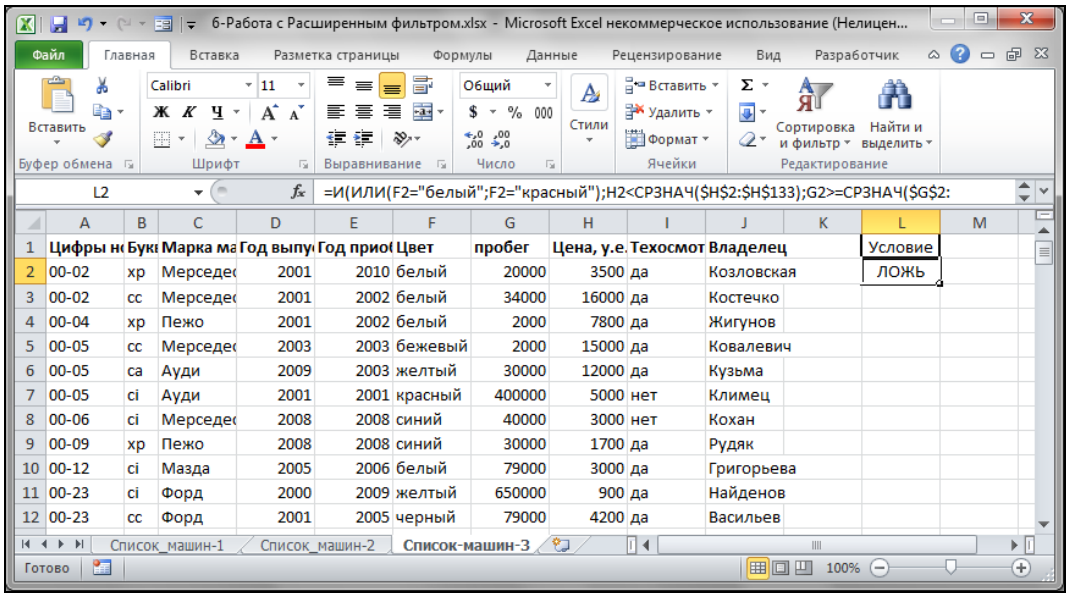
	A	B	C	D	E	F	G	H	I	J
1	Цифры нс	Бук	Марка ма	Год выпу	Год прио	Цвет	пробег	Цена, у.е.	Техосмот	Владелец
2	00-02	хр	Мерседес	2001	2010	белый	20000	3500	да	Козловская
3	00-02	сс	Мерседес	2001	2002	белый	34000	16000	да	Костечко
4	00-04	хр	Пежо	2001	2002	белый	2000	7800	да	Жигунов
5	00-05	сс	Мерседес	2003	2003	бежевый	2000	15000	да	Ковалевич
6	00-05	са	Ауди	2009	2003	желтый	30000	12000	да	Кузьма
7	00-05	сі	Ауди	2001	2001	красный	400000	5000	нет	Климец
8	00-06	сі	Мерседес	2008	2008	синий	40000	3000	нет	Кохан
9	00-09	хр	Пежо	2008	2008	синий	30000	1700	да	Рудяк
10	00-12	сі	Мазда	2005	2006	белый	79000	3000	да	Григорьева
11	00-23	сі	Форд	2000	2009	желтый	650000	900	да	Найденов
12	00-23	сс	Форд	2001	2005	черный	79000	4200	да	Васильев
13	00-25	са	Рено	2010	2010	желтый	2300	13000	да	Бегунов
14	00-32	са	БМВ	2002	2007	красный	7000	6500	нет	Мышко

Рис. 7.13. Вычисляемый критерий для расширенного фильтра

3. Перейдите на вкладку **Данные** ленты и в группе команд **Сортировка и фильтр** нажмите кнопку **Дополнительно**.
4. В окне **Расширенный фильтр** (рис. 7.10) укажите требуемые диапазоны и параметры отбора записей списка. Нажмите кнопку **OK**.
5. Отфильтрованные данные отобразятся на рабочем листе.

Пример 3. Определить автомобили белого или красного цвета, цена которых меньше средней цены для всех автомобилей и пробег больше среднего пробега либо равен ему.

1. Откройте список, подлежащий фильтрации (список располагается в диапазоне **A1:J133**, строка заголовка — в диапазоне **A1:J1**, рабочий лист — **Список_машин-3**).



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Цифры но	Буки	Марка ма	Год выпу	Год прио	Цвет	пробег	Цена, у.е.	Техосмот	Владелец		Условие	
2	00-02	хр	Мерседес	2001	2010	белый	20000	3500	да	Козловская		ЛОЖЬ	
3	00-02	сс	Мерседес	2001	2002	белый	34000	16000	да	Костечко			
4	00-04	хр	Пежо	2001	2002	белый	2000	7800	да	Жигунов			
5	00-05	сс	Мерседес	2003	2003	бежевый	2000	15000	да	Ковалевич			
6	00-05	са	Ауди	2009	2003	желтый	30000	12000	да	Кузьма			
7	00-05	ci	Ауди	2001	2001	красный	400000	5000	нет	Климец			
8	00-06	ci	Мерседес	2008	2008	синий	40000	3000	нет	Кохан			
9	00-09	хр	Пежо	2008	2008	синий	30000	1700	да	Рудяк			
10	00-12	ci	Мазда	2005	2006	белый	79000	3000	да	Григорьева			
11	00-23	ci	Форд	2000	2009	желтый	650000	900	да	Найденов			
12	00-23	сс	Форд	2001	2005	черный	79000	4200	да	Васильев			

Рис. 7.14. Диапазон для вычисляемого критерия, включающего среднюю цену и средний пробег автомобилей

2. Сформируйте вычисляемый критерий для расширенного фильтра в диапазоне **L1:L2** (см. рис. 7.14). Ячейка L1 содержит слово **Условие**. В ячейку **L2** введите формулу:

```
=И(ИЛИ(F2="белый";F2="красный");H2<СРЗНАЧ($H$2:$H$133);G2>=СРЗНАЧ($G$2:$G$133))
```

3. Перейдите на вкладку **Данные** ленты и в группе команд **Сортировка и фильтр** воспользуйтесь кнопкой **Дополнительно**.

Немного о методе *AdvancedFilter*

Метод `AdvancedFilter` для объекта `Range` разрешает программным способом выполнить расширенную фильтрацию списка по критериям, указанным в параметрах:

expression`.AdvancedFilter(`*Action*`, `*CriteriaRange*`, `*CopyToRange*`, `*Unique*`)`

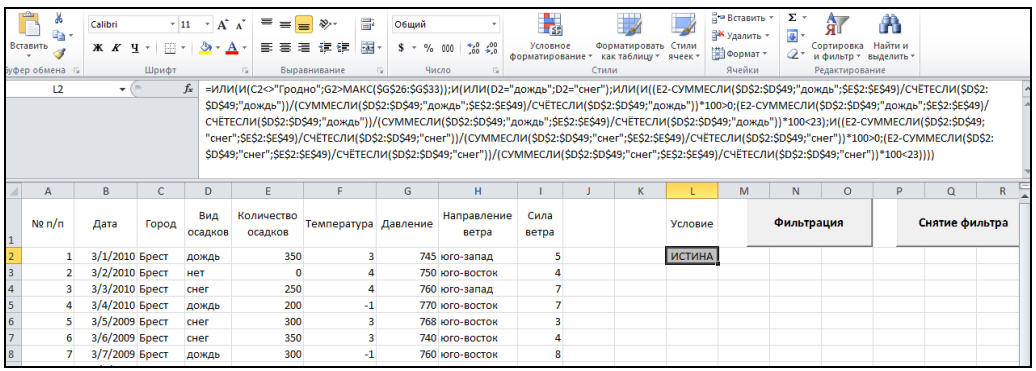
- *expression* — ссылка на ячейку диапазона или на сам диапазон, который будет фильтроваться.
- *Action* — константа, которая указывает, оставить ли отфильтрованные данные на рабочем листе (`xlFilterInPlace`) или же скопировать их в другое место (`xlFilterCopy`).
- *CriteriaRange* — необязательный параметр, указывающий на наличие критериев отбора; если этот параметр опущен, значит, критерии отбора отсутствуют.
- *CopyToRange* — необязательный параметр, указывающий на диапазон, куда будут помещены отфильтрованные данные, если выбрана константа `xlFilterCopy`.
- `Unique` — параметр, указывающий на необходимость оставить в отфильтрованном диапазоне только уникальные значения; может принимать значения `True` или `False`; по умолчанию устанавливается значение `False`.

В качестве демонстрационного примера использования метода `AdvancedFilter` разберем следующую задачу.

Пусть в списке на рабочем листе находятся данные о погоде. Необходимо определить города, давление воздуха в которых больше максимального значения для города Гродно, или города, осадки в которых — дождь или снег, а их количество превышает среднее для всех видов осадков не более, чем на 23%. Итак (см. файл *7-Расширенный фильтр.xlsm* на компакт-диске):

1. Откройте список, подлежащий фильтрации (список располагается в диапазоне **A1:I49**, строка заголовка — в диапазоне **A1:I1**, рабочий лист — **Осадки**).
2. Сформируйте вычисляемый критерий для расширенного фильтра в диапазоне **L1:L2** (рис. 7.15). Ячейка **L1** содержит слово **Условие**. В ячейку **L2** введите формулу:

```
=ИЛИ(И(C2<>"Гродно";G2>МАКС($G$26:$G$33));И(ИЛИ(D2="дождь";D2="снег");
ИЛИ(И((E2-СУММЕСЛИ($D$2:$D$49;"дождь";$E$2:$E$49)/СЧЁТЕСЛИ($D$2:$D$49;
"дождь"))/(СУММЕСЛИ($D$2:$D$49;"дождь";$E$2:$E$49)/СЧЁТЕСЛИ($D$2:$D$49;
"дождь"))*100>0;(E2-СУММЕСЛИ($D$2:$D$49;"дождь";$E$2:$E$49)
/СЧЁТЕСЛИ($D$2:$D$49;"дождь"))/(СУММЕСЛИ($D$2:$D$49;"дождь";$E$2:$E$49)
/СЧЁТЕСЛИ($D$2:$D$49;"дождь"))*100<23);И((E2-СУММЕСЛИ($D$2:$D$49;
"снег";$E$2:$E$49)/СЧЁТЕСЛИ($D$2:$D$49;"снег"))/(СУММЕСЛИ($D$2:$D$49;
"снег";$E$2:$E$49)/СЧЁТЕСЛИ($D$2:$D$49;"снег"))*100>0;
(E2-СУММЕСЛИ($D$2:$D$49;"снег";$E$2:$E$49)/СЧЁТЕСЛИ($D$2:$D$49;"снег"))
/(СУММЕСЛИ($D$2:$D$49;"снег";$E$2:$E$49)/СЧЁТЕСЛИ($D$2:$D$49;"снег"))
*100<23))))
```



№ п/п	Дата	Город	Вид осадков	Количество осадков	Температура	Давление	Направление ветра	Сила ветра
1	3/1/2010	Брест	дождь	350	3	745	юго-запад	5
2	3/2/2010	Брест	нет	0	4	750	юго-восток	4
3	3/3/2010	Брест	снег	250	4	760	юго-запад	7
4	3/4/2010	Брест	дождь	200	-1	770	юго-восток	7
5	3/5/2009	Брест	снег	300	3	768	юго-восток	3
6	3/6/2009	Брест	снег	350	3	740	юго-восток	4
7	3/7/2009	Брест	дождь	300	-1	760	юго-восток	8

Рис. 7.15. Формула для вычисляемого критерия для задачи, содержащей условия по осадкам

3. Расположите на рабочем листе два элемента управления **Кнопка** (CommandButton), одна из кнопок будет отвечать за фильтрацию данных, а вторая — за снятие фильтра.
4. Измените значения свойства `Caption` для первой кнопки (`CommandButton1`) на `Фильтрация`, а для второй кнопки (`CommandButton2`) на `Снятие фильтра`.
5. Введите на листе стандартного модуля программный код, представленный в листинге 7.6, на листе модуля `Лист1` — программный код, представленный в листинге 7.7.

Листинг 7.6. Расширенная фильтрация. Стандартный модуль

```
Sub AdFilt()
    Range("A1:I49").AdvancedFilter Action:=xlFilterInPlace, _
                      CriteriaRange:=Range("L1:L2"), Unique:=True
End Sub
Sub Del()
  Worksheets("Осадки").ShowAllData
End Sub
```

Листинг 7.7. Расширенная фильтрация. Модуль Лист1

```
Private Sub CommandButton1_Click()
  AdFilt
End Sub
Private Sub CommandButton2_Click()
  Del
End Sub
```

6. Теперь, для того чтобы получить данные из списка в соответствии с критериями, вам достаточно лишь нажать кнопку **Фильтрация**, расположенную на рабочем листе **Осадки**, для снятия фильтра — щелкнуть по кнопке **Снятие фильтра**.

Наши итоги

Итак, в этой главе вы познакомились со списками данных, которые могут обрабатываться Microsoft Office Excel. Сортировка и отбор данных в соответствии с некоторыми критериями позволяют, во-первых, перестраивать данные списка в нужном порядке, а во-вторых, быстро извлекать из однотипных массивов данных требуемую информацию. Теперь вы:

- ❒ имеете представление о том, что такое списки;
- ❒ умеете производить различные сортировки данных;
- ❒ использовать VBA для сортировки данных;
- ❒ производить фильтрацию данных с использованием автофильтра и расширенного фильтра;
- ❒ программировать автофильтрацию;
- ❒ использовать метод `AdvancedFilter` для отбора данных в соответствии с пользовательскими условиями.

С использованием данных возможностей ваши приложения по обработке списков станут более функциональными и простыми в использовании.

Глава 8

Обрабатываем данные средствами Microsoft Office Excel

Microsoft Office Excel 2010 предоставляет в распоряжение пользователя и разработчика достаточно эффективные средства по обработке и анализу данных. К средствам анализа данных относят:

- промежуточные итоги — средство для автоматического подведения итогов требуемого уровня вложенности в списке однотипных данных;
- консолидация — средство обобщения однородных данных;
- сводные таблицы — средство для представления и анализа данных в трехмерном виде;
- структуризация рабочих листов — средство представления данных по уровням иерархической организации;
- специальные средства анализа данных — **Сценарии**, **Таблица подстановки**, **Пакет анализа** и др.

В этой главе продемонстрируем использование наиболее используемых перечисленных выше средств. Отметим также, что работа с **Подбором параметра** и **Поиском решения** будет рассмотрена нами в следующей главе.

> ***Примечание***
>
> Файлы рассмотренных в данной главе примеров можно найти в папке Glava_8 на компакт-диске.

Подводим промежуточные итоги

Промежуточные итоги необходимы для создания разнообразных отчетов и для обобщения большого количества однотипной информации. При добавлении автоматических промежуточных итогов Excel изменяет разметку списка данных, что позволяет отображать и скрывать строки каждого промежуточного уровня итогов. С помощью средства **Итоги** можно: указать способ группировки данных; вывести промежуточные и общие итоги для одной группы в списке; вывести промежуточные и общие итоги для нескольких групп в списке, а также выполнить расчеты над данными. Таким образом, **Итоги** позволяют, не составляя формул и не преобразуя таблицу, по предлагаемому MS Excel списку математических выражений найти промежуточные и общие итоги, структурировать данные и вставить в таблицу строки с промежуточными и общими итогами.

При создании итогов при необходимости можно выполнить следующие действия:

- использовать одну операцию для нескольких столбцов данных;
- использовать несколько операций для одного набора данных (например, вычислить среднее и суммарное значения для столбца с числовыми данными);
- подвести итоги по отфильтрованным данным (сначала — отфильтровать, затем — отсортировать по необходимому полю (полям) и, наконец, — подвести итоги).

Рассмотрим несколько примеров создания итогов в Microsoft Office Excel 2010.

Простые промежуточные итоги

Простые промежуточные итоги подводятся следующим образом (методика подведения итогов).

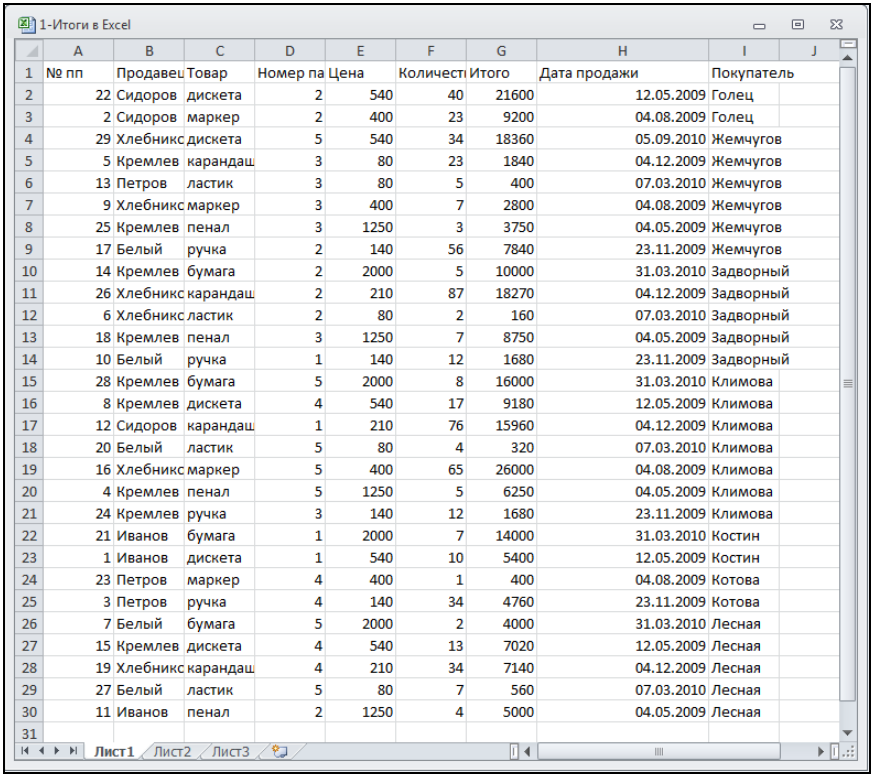
1. Подготовьте список данных и оставьте в нем указатель ячейки. Определитесь с тем, какие нужны итоги.
2. Проведите сортировку по необходимому полю (используйте, например, окно **Сортировка**, которое открывается при выполнении команды **Сортировка**, расположенной в группе **Сортировка и фильтр** на вкладке **Данные** ленты).
3. Подведите итоги с использованием окна **Итоги** (открывается при выполнении команды **Промежуточный итог**, расположенной в группе **Структура** на вкладке **Данные** ленты).

> ***Совет***
>
> Чтобы убрать промежуточные итоги, необходимо установить указатель в список с итогами и воспользоваться кнопкой **Убрать все** в окне **Итоги**.

Итак, рассмотрим пример подведения простых промежуточных итогов. Пусть имеется список данных со следующими полями: (рис. 8.1): `№ пп`, `Продавец`, `Товар`, `Номер партии`, `Цена`, `Количество`, `Итого`, `Дата продажи`, `Покупатель`. Определить количество товаров, проданных конкретным продавцом.

Для подведения простых промежуточных итогов выполните следующие действия.

1. Выделите список (или просто установите в список указатель ячейки) и проведите сортировку (выполните команду **Сортировка**, расположенную в группе **Сортировка и фильтр** на вкладке **Данные** ленты) по полю `Продавец` (рис. 8.2).
2. Воспользуйтесь командой **Промежуточный итог**, расположенной в группе **Структура** на вкладке **Данные** ленты.
3. В открывшемся окне **Промежуточные итоги** установите параметры в соответствии с рис. 8.3 и нажмите кнопку **OK**.
4. Полученные промежуточные итоги представлены на рис. 8.4 (см. также файл *1-Итоги в Excel.xlsm* на компакт-диске).



1-Итоги в Excel

	A	B	C	D	E	F	G	H	I	J
1	№ пп	Продавец	Товар	Номер па	Цена	Количест	Итого	Дата продажи	Покупатель	
2	22	Сидоров	дискета	2	540	40	21600	12.05.2009	Голец	
3	2	Сидоров	маркер	2	400	23	9200	04.08.2009	Голец	
4	29	Хлебникс	дискета	5	540	34	18360	05.09.2010	Жемчугов	
5	5	Кремлев	карандаш	3	80	23	1840	04.12.2009	Жемчугов	
6	13	Петров	ластик	3	80	5	400	07.03.2010	Жемчугов	
7	9	Хлебникс	маркер	3	400	7	2800	04.08.2009	Жемчугов	
8	25	Кремлев	пенал	3	1250	3	3750	04.05.2009	Жемчугов	
9	17	Белый	ручка	2	140	56	7840	23.11.2009	Жемчугов	
10	14	Кремлев	бумага	2	2000	5	10000	31.03.2010	Задворный	
11	26	Хлебникс	карандаш	2	210	87	18270	04.12.2009	Задворный	
12	6	Хлебникс	ластик	2	80	2	160	07.03.2010	Задворный	
13	18	Кремлев	пенал	3	1250	7	8750	04.05.2009	Задворный	
14	10	Белый	ручка	1	140	12	1680	23.11.2009	Задворный	
15	28	Кремлев	бумага	5	2000	8	16000	31.03.2010	Климова	
16	8	Кремлев	дискета	4	540	17	9180	12.05.2009	Климова	
17	12	Сидоров	карандаш	1	210	76	15960	04.12.2009	Климова	
18	20	Белый	ластик	5	80	4	320	07.03.2010	Климова	
19	16	Хлебникс	маркер	5	400	65	26000	04.08.2009	Климова	
20	4	Кремлев	пенал	5	1250	5	6250	04.05.2009	Климова	
21	24	Кремлев	ручка	3	140	12	1680	23.11.2009	Климова	
22	21	Иванов	бумага	1	2000	7	14000	31.03.2010	Костин	
23	1	Иванов	дискета	1	540	10	5400	12.05.2009	Костин	
24	23	Петров	маркер	4	400	1	400	04.08.2009	Котова	
25	3	Петров	ручка	4	140	34	4760	23.11.2009	Котова	
26	7	Белый	бумага	5	2000	2	4000	31.03.2010	Лесная	
27	15	Кремлев	дискета	4	540	13	7020	12.05.2009	Лесная	
28	19	Хлебникс	карандаш	4	210	34	7140	04.12.2009	Лесная	
29	27	Белый	ластик	5	80	7	560	07.03.2010	Лесная	
30	11	Иванов	пенал	2	1250	4	5000	04.05.2009	Лесная	
31										

Лист1 Лист2 Лист3

Рис. 8.1. Список продаж

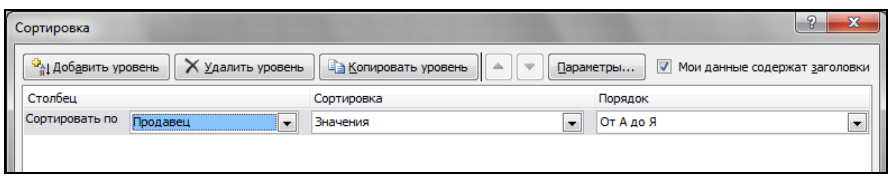


Рис. 8.2. Сортировка списка по полю `Продавец`

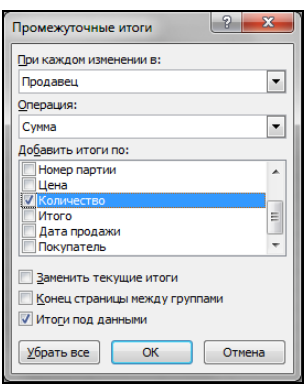
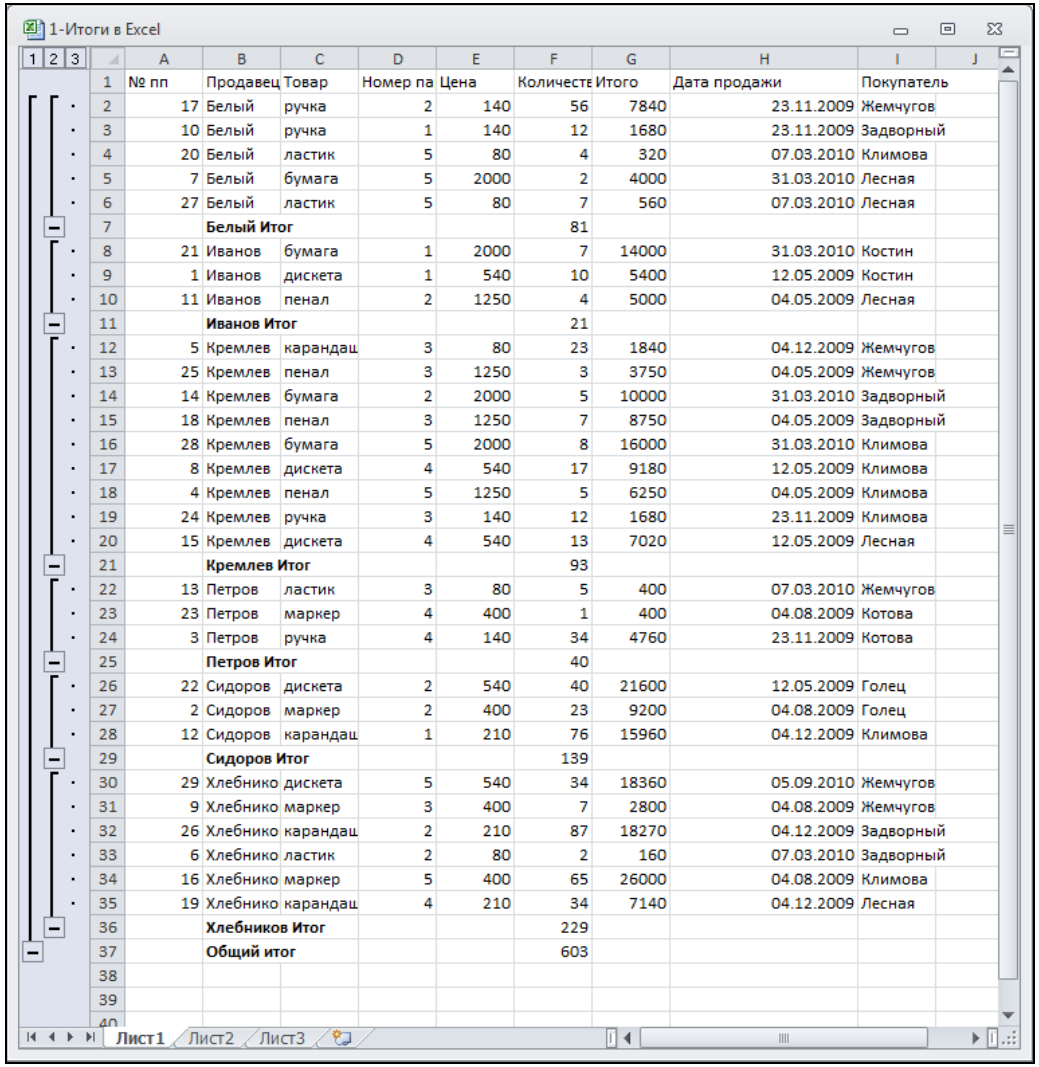


Рис. 8.3. Окно **Промежуточные итоги** для получения итогов по полю `Продавец`



	A	B	C	D	E	F	G	H	I	J
1	№ пп	Продавец	Товар	Номер па	Цена	Количеств	Итого	Дата продажи	Покупатель	
2	17	Белый	ручка	2	140	56	7840	23.11.2009	Жемчугов	
3	10	Белый	ручка	1	140	12	1680	23.11.2009	Задворный	
4	20	Белый	ластик	5	80	4	320	07.03.2010	Климова	
5	7	Белый	бумага	5	2000	2	4000	31.03.2010	Лесная	
6	27	Белый	ластик	5	80	7	560	07.03.2010	Лесная	
7		**Белый Итог**				81				
8	21	Иванов	бумага	1	2000	7	14000	31.03.2010	Костин	
9	1	Иванов	дискета	1	540	10	5400	12.05.2009	Костин	
10	11	Иванов	пенал	2	1250	4	5000	04.05.2009	Лесная	
11		**Иванов Итог**				21				
12	5	Кремлев	карандаш	3	80	23	1840	04.12.2009	Жемчугов	
13	25	Кремлев	пенал	3	1250	3	3750	04.05.2009	Жемчугов	
14	14	Кремлев	бумага	2	2000	5	10000	31.03.2010	Задворный	
15	18	Кремлев	пенал	3	1250	7	8750	04.05.2009	Задворный	
16	28	Кремлев	бумага	5	2000	8	16000	31.03.2010	Климова	
17	8	Кремлев	дискета	4	540	17	9180	12.05.2009	Климова	
18	4	Кремлев	пенал	5	1250	5	6250	04.05.2009	Климова	
19	24	Кремлев	ручка	3	140	12	1680	23.11.2009	Климова	
20	15	Кремлев	дискета	4	540	13	7020	12.05.2009	Лесная	
21		**Кремлев Итог**				93				
22	13	Петров	ластик	3	80	5	400	07.03.2010	Жемчугов	
23	23	Петров	маркер	4	400	1	400	04.08.2009	Котова	
24	3	Петров	ручка	4	140	34	4760	23.11.2009	Котова	
25		**Петров Итог**				40				
26	22	Сидоров	дискета	2	540	40	21600	12.05.2009	Голец	
27	2	Сидоров	маркер	2	400	23	9200	04.08.2009	Голец	
28	12	Сидоров	карандаш	1	210	76	15960	04.12.2009	Климова	
29		**Сидоров Итог**				139				
30	29	Хлебнико	дискета	5	540	34	18360	05.09.2010	Жемчугов	
31	9	Хлебнико	маркер	3	400	7	2800	04.08.2009	Жемчугов	
32	26	Хлебнико	карандаш	2	210	87	18270	04.12.2009	Задворный	
33	6	Хлебнико	ластик	2	80	2	160	07.03.2010	Задворный	
34	16	Хлебнико	маркер	5	400	65	26000	04.08.2009	Климова	
35	19	Хлебнико	карандаш	4	210	34	7140	04.12.2009	Лесная	
36		**Хлебников Итог**				229				
37		**Общий итог**				603				
38										
39										
40										

Рис. 8.4. Промежуточные итоги

Вложенные промежуточные итоги

Вложенные промежуточные итоги предполагают получение нескольких уровней вложенности для одного списка данных.

Вложенные промежуточные итоги подводятся следующим образом.

1. Подготовьте список данных и оставьте в нем указатель ячейки. Определитесь с тем, какие нужны итоги — по уровням вложенности.
2. Проведите сортировку по необходимым полям (используйте окно **Сортировка**, которое открывается при выполнении команды **Сортировка**, расположенной в группе **Сортировка и фильтр** на вкладке **Данные** ленты).

3. Подведите итоги с использованием окна **Итоги** (которое откроется при выполнении команды **Промежуточный итог**, расположенной в группе **Структура** на вкладке **Данные** ленты). При создании вложенных промежуточных итогов следует четко представлять уровни итогов и создавать их в порядке увеличения уровня детализации: сначала — по первому полю сортировки, далее, отключая опцию **Заменить текущие итоги** (в окне **Промежуточные итоги**), — по второму полю и т. д.

А теперь рассмотрим пример вложенных итогов, используя в качестве исходных данных все тот же список продаж (рис. 8.1). Теперь пусть нам необходимо получить общее количество товаров, проданных конкретным продавцом с учетом конкретной даты продажи.

1. Выделите список (или установите в список указатель ячейки) и проведите сортировку (выполните команду **Сортировка**, расположенную в группе **Сортировка и фильтр** на вкладке **Данные** ленты) по полям `Продавец` и `Дата продажи` (рис. 8.5). Для добавления каждого следующего поля в окне **Сортировка** для сортировки нажмите кнопку **Добавить уровень**.

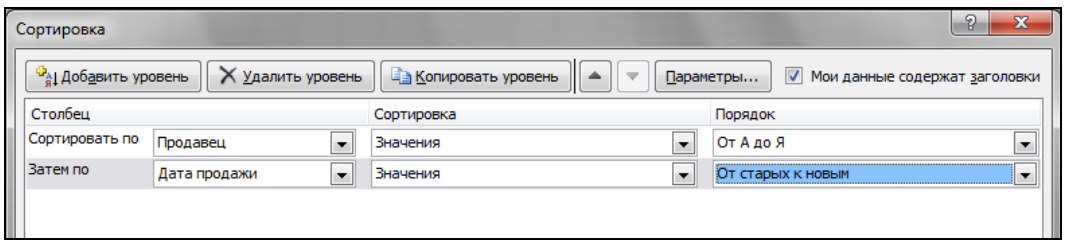


Рис. 8.5. Сортировка списка по полям `Продавец` и `Дата продажи`

2. Воспользуйтесь командой **Промежуточный итог**, расположенной в группе **Структура** на вкладке **Данные** ленты.
3. В открывшемся окне **Промежуточные итоги** установите параметры в соответствии с рис. 8.3 для получения верхнего (первого) уровня итогов — общее количество товаров, проданных конкретным продавцом (см. рис. 8.4).
4. Для получения второго уровня итогов в данный список с полученными итогами снова поместите указатель ячейки, затем воспользуйтесь командой **Промежуточный итог**.
5. В открывшемся окне **Промежуточные итоги** установите параметры в соответствии с рис. 8.6.
6. Полученные промежуточные итоги представлены на рис. 8.7 (см. также файл *1-Итоги в Excel.xlsm* на компакт-диске).

Примечание

При добавлении в список промежуточных итогов разметка списка изменяется таким образом, что становится видна его структура. Нажимая кнопки структуры [+], [−] и [1][2][3][4], можно создать итоговый отчет, скрыв подробности и отобразив только итоги.

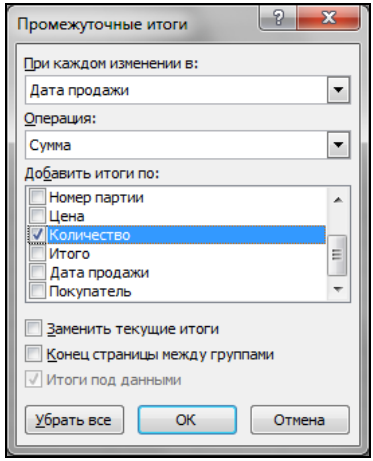


Рис. 8.6. Окно **Промежуточные итоги** для получения итогов по полю `Дата продажи`

1-Итоги в Excel

	A	B	C	D	E	F	G	H	I	J
1	№ пп	Продавец	Товар	Номер па	Цена	Количест	Итого	Дата продажи	Покупатель	
10		**Белый Итог**				81				
17		**Иванов Итог**				21				
20						13		**31.03.2010 Итог**		
21	5	Кремлев	карандаш	3	80	23	1840	04.12.2009	Жемчугов	
22						23		**04.12.2009 Итог**		
23	24	Кремлев	ручка	3	140	12	1680	23.11.2009	Климова	
24						12		**23.11.2009 Итог**		
25	8	Кремлев	дискета	4	540	17	9180	12.05.2009	Климова	
26	15	Кремлев	дискета	4	540	13	7020	12.05.2009	Лесная	
27						30		**12.05.2009 Итог**		
28	25	Кремлев	пенал	3	1250	3	3750	04.05.2009	Жемчугов	
29	18	Кремлев	пенал	3	1250	7	8750	04.05.2009	Задворный	
30	4	Кремлев	пенал	5	1250	5	6250	04.05.2009	Климова	
31						15		**04.05.2009 Итог**		
32		**Кремлев Итог**				93				
39		**Петров Итог**				40				
46		**Сидоров Итог**				139				
47	29	Хлебников	дискета	5	540	34	18360	05.09.2010	Жемчугов	
48						34		**05.09.2010 Итог**		
49	6	Хлебников	ластик	2	80	2	160	07.03.2010	Задворный	
50						2		**07.03.2010 Итог**		
51	26	Хлебников	карандаш	2	210	87	18270	04.12.2009	Задворный	
52	19	Хлебников	карандаш	4	210	34	7140	04.12.2009	Лесная	
53						121		**04.12.2009 Итог**		
54	9	Хлебников	маркер	3	400	7	2800	04.08.2009	Жемчугов	
55	16	Хлебников	маркер	5	400	65	26000	04.08.2009	Климова	
56						72		**04.08.2009 Итог**		
57		**Хлебников Итог**				229				
58		**Общий итог**				603				
59										

Простые итоги | **Вложенные итоги** | Лист3

Рис. 8.7. Вложенные промежуточные итоги

Метод *Subtotal*

Метод `Subtotal` объекта `Range` добавляет промежуточные итоги в список данных, основываясь на изменениях в определенных полях данных. Промежуточные итоги позволяют обобщить данные. Метод `Subtotal` автоматически вставляет содержащие промежуточные итоги строки с формулами, по которым подводятся итоги. Необходимо, чтобы до активизации этого метода данные были правильно отсортированы. В противном случае этот метод может привести к неверному выводу промежуточных итогов. Метод `Subtotal` программирует выполнение команды **Промежуточный итог**, расположенной в группе **Структура** на вкладке **Данные** ленты.

```
expression.Subtotal(GroupBy, Function, TotalList, Replace, PageBreaks,
↳SummaryBelowData)
```

- *expression* — ссылка на ячейку диапазона или на весь диапазон, для которого подводятся промежуточные итоги.
- *GroupBy* — обязательный параметр, задающий номер поля, по которому вычисляются промежуточные итоги.
- *Function* — обязательный параметр, определяющий функцию, по которой производится подсчет промежуточных итогов. Допустимыми значениями являются следующие константы `XlConsolidationFunction`: `xlAverage` (среднее арифметическое), `xlCount` (количество значений), `xlCountNums` (количество чисел), `xlMax` (максимум), `xlMin` (минимум), `xlProduct` (произведение), `xlStDev` (несмещенное отклонение), `xlStDevP` (смещенное отклонение), `xlSum` (сумма), `xlVar` (несмещенная дисперсия) и `xlVarP` (смещенная дисперсия).
- *TotalList* — обязательный параметр, специфицирующий массив целых чисел с номерами полей, по которым вычисляются промежуточные итоги.
- *Replace* — необязательный параметр, принимающий логические значения. Если его значение равно `True`, то существующие промежуточные итоги будут замещены.
- *PageBreaks* — необязательный параметр, принимающий логические значения. Если его значение равно `True`, то после каждой группы будет вставлено по символу разрыва страницы.
- *SummaryBelowData* — необязательный параметр, задающий местоположение вывода промежуточных итогов. Допустимыми значениями являются следующие константы `XlSummaryRow`: `xlSummaryAbove` (промежуточные итоги будут выведены над данными) и `xlSummaryBelow` (промежуточные итоги будут выведены под данными).

Удаление промежуточных итогов

Метод `RemoveSubtotal` объекта `Range` удаляет промежуточные итоги с рабочего листа. Например, следующая инструкция удаляет итоги, связанные с диапазоном **A1:I40**.

```
Range("A1:I40").RemoveSubtotal
```

Обобщаем однородные данные с помощью консолидации

При консолидации данных объединяются значения из нескольких диапазонов данных, происходит обобщение однородных данных. Например, можно обработать сведения, поступающие из различных отделов компании, и, таким образом, получить общую картину. Однако консолидация — это не только суммирование. В ходе этого процесса можно вычислить такие статистические величины, как среднее значение, стандартное отклонение, количество значений. Консолидировать данные в MS Excel можно несколькими способами:

- *консолидация при помощи трехмерных формул*. В этом случае создаются трехмерные формулы, содержащие ссылки на ячейки обобщаемых данных в разных диапазонах, возможно, находящихся на различных листах;
- *консолидация по расположению*. Ее следует использовать в случае, если данные всех исходных областей находятся в одном месте и размещены в одинаковом порядке, например, если имеются данные из нескольких листов, созданных на основе одного шаблона;
- *консолидация по категориям*. Она применяется в случае, если требуется обобщить набор листов, имеющих одинаковые заголовки рядов и столбцов, но различную организацию данных. Этот способ позволяет консолидировать данные с одинаковыми заголовками со всех листов.

Консолидация при помощи трехмерных формул на рабочем листе

В качестве примера консолидации при помощи трехмерных формул рассмотрим бизнес-ситуацию построения консолидирующей таблицы о расходах фирмы ООО "Альянс" за отчетный период с января по март, которые собраны в таблицы, расположенные на рабочих листах **Январь**, **Февраль** и **Март**. Расходы фирмы детализированы поквартально (рис. 8.8, см. также файл *2-Консолидация при помощи трехмерных формул на рабочем листе.xlsm* на компакт-диске).

Итак:

1. Создайте рабочий лист **Итоги**, на котором разместите шаблон отчетной таблицы.
2. Введите в ячейку **B3** формулу, находящую суммарные расходы за телефон в первом квартале с января по март:

```
=СУММ(Январь:Март!B3)
```

или равносильную формулу

```
=СУММ(Январь!B3;Февраль!B3;Март!B3)
```

	A	B	C	D	E
1		Декада			Итого
2		I	II	III	
3	Телефон	3242	3424	423423	430089
4	Аренда	4234	23424	2344	30002
5	Амортиза	423	14123	1321	15867
6	Страховка	213	23	234	470
7	Заработна	5141	3424	334	8899
8	Итого	13253	44418	427656	485327
9					
10					
11					

Январь | **Февраль** | Март

Рис. 8.8. Данные для консолидации

3. Расположите указатель мыши на маркере заполнения и переместите его вниз и вправо на диапазон **B3:E8**. Это позволит найти суммарные расходы по каждой категории расходов с июня по август.

Консолидация при помощи трехмерных формул в коде

Описанную в предыдущем разделе процедуру создания консолидирующей таблицы на основе трехмерных формул можно автоматизировать при помощи следующего кода (листинг 8.1, см. также файл *3-Консолидация при помощи трехмерных формул в коде.xlsm* на компакт-диске). В нем имеется проверка наличия в книге листа с именем **Итоги**. Если такой лист отсутствует, на экране отображается сообщение, а процесс построения консолидирующей таблицы прерывается.

Листинг 8.1. Консолидация при помощи трехмерных формул

```
Sub DemoConsolidate3D()
    Dim rgn As Range
    Dim ws As Worksheet
    Dim str As String
    Dim nm As String
    nm = "Итоги"
    For Each ws In Worksheets
       str = str & ws.Name & "!B3" & ";"
       If ws.Name = nm Then
          MsgBox "Итоговый лист уже существует"
          Exit Sub
       End If
    Next
    str = Left(str, Len(str) - 1)
    Worksheets.Add After:=Worksheets(Worksheets.Count)
    ActiveSheet.Name = nm
    Worksheets("Январь").Range("A1:E8").Copy Worksheets(nm).Range("A1:E8")
    Range("B3:E8").Clear
    Range("B3").FormulaLocal = "=СУММ(" & str & ")"
    Range("B3").AutoFill Destination:=Range("B3:B8"), Type:=xlFillDefault
    Range("B3:B8").AutoFill Destination:=Range("B3:E8"), Type:=xlFillDefault
End Sub
```

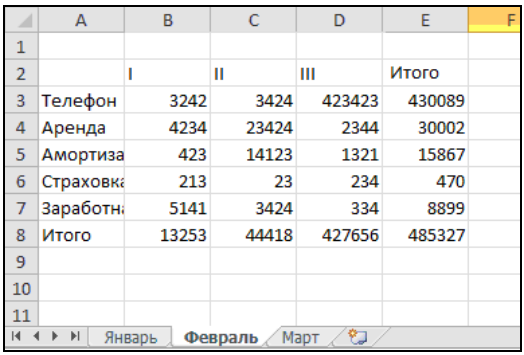
Консолидация данных по положению и категориям

Консолидация данных по положению производится, если планируется объединение данных, находящихся в одинаковых ячейках разных диапазонов. Консолидация по категориям производится, если имеется нескольких диапазонов и планируется объединять эти данные по строкам или столбцам с одинаковыми подписями.

Совместно с консолидацией полезно также использовать структурирование, причем структура может создаваться автоматически. Важно, что предназначенные для консолидации рабочие листы совсем не обязаны иметь одну и ту же структуру.

Опишем процесс консолидации данных на примере построения консолидирующей таблицы о расходах фирмы ООО "Альянс" за отчетный период с января по март.

1. Убедитесь, что все диапазоны консолидируемых данных представлены в формате *списка*. Если консолидация выполняется по положению, убедитесь, что макеты всех диапазонов совпадают. Если консолидация выполняется по категории, убедитесь, что подписи столбцов или строк, которые требуется объединить, совпадают (с учетом регистра букв). Итак, проверьте, чтобы в вашей рабочей книге на трех листах **Январь**, **Февраль**, **Март** располагались данные в виде, представленном на рис. 8.9. Кроме того, в книге также должен иметься лист **Итоги** — для помещения результирующей таблицы после консолидации данных.



	A	B	C	D	E	F
1						
2		I	II	III	Итого	
3	Телефон	3242	3424	423423	430089	
4	Аренда	4234	23424	2344	30002	
5	Амортиза	423	14123	1321	15867	
6	Страховк	213	23	234	470	
7	Заработн	5141	3424	334	8899	
8	Итого	13253	44418	427656	485327	
9						
10						
11						

Январь | Февраль | Март

Рис. 8.9. Данные о расходах фирмы "Альянс"

2. Выберите левую верхнюю ячейку диапазона, в котором требуется разместить консолидированные данные. В нашем случае выберем ячейку **A2** рабочего листа **Итоги**.
3. Выберите команду **Консолидация**, расположенную в группе **Работа с данными** на вкладке **Данные** ленты. На экране отобразится диалоговое окно **Консолидация** (рис. 8.10).

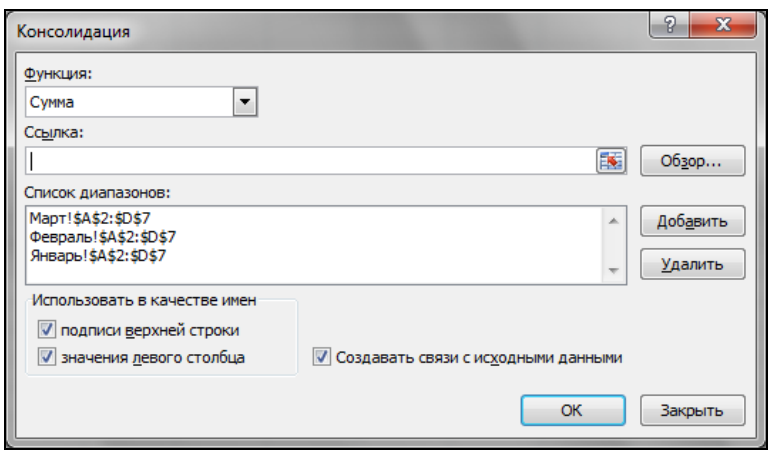
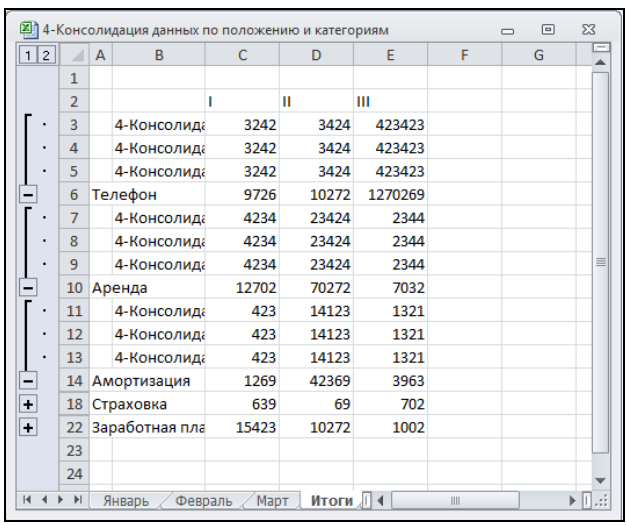


Рис. 8.10. Окно **Консолидация**

4. Выберите из раскрывающегося списка **Функция** так называемую *весовую*, или *итоговую функцию*. Эта функция задает тип вычисления, производимый при объединении данных в таблице консолидации. Допустимыми являются следующие типы: `Сумма`, `Количество`, `Среднее`, `Максимум`, `Минимум`, `Произведение`, `Количество чисел`, `Смещенное отклонение`, `Несмещенное отклонение`, `Смещенная дисперсия`, `Несмещенная дисперсия`. В данном случае выберите `Сумма`.
5. Щелкните в поле **Ссылка**, откройте лист, содержащий первый диапазон данных для консолидации, введите ссылку на этот диапазон (в данном случае `Январь!$A$2:$E$8`) и нажмите кнопку **Добавить**. В результате ссылка на диапазон будет добавлена в **Список диапазонов**. Повторите этот шаг для всех консолидируемых диапазонов. В данном случае для `Февраль!$A$2:$E$8` и `Март!$A$2:$E$8`. При этом:
   - если таблицу консолидации требуется обновлять автоматически при каждом изменении данных в каком-либо исходном диапазоне и позднее точно не потребуется изменять или добавлять диапазоны исходных данных для консолидации, установите флажок **Создавать связи с исходными данными**, что в нашем случае и следует сделать;
   - если консолидация выполняется по положению, оставьте все поля в группе **Использовать в качестве имен** пустыми. В MS Excel подписи исходных строк и столбцов не копируются в консолидированные данные. Если требуется скопировать подписи в консолидированные данные, сделайте это вручную. В нашем случае этот флажок не устанавливается;
   - если консолидация выполняется по категории, в группе **Использовать в качестве имен** установите флажки, соответствующие расположению подписей в исходных диапазонах: в верхней строке, в левом столбце или в верхней строке и в левом столбце одновременно. Все подписи, не совпадающие с подписями в других исходных областях, в консолидированных данных будут расположены в отдельных строках или столбцах. В нашем случае этот флажок устанавливается.
6. Нажмите кнопку **OK**.



	A	B	C	D	E
1					
2			I	II	III
3		4-Консолида	3242	3424	423423
4		4-Консолида	3242	3424	423423
5		4-Консолида	3242	3424	423423
6	Телефон		9726	10272	1270269
7		4-Консолида	4234	23424	2344
8		4-Консолида	4234	23424	2344
9		4-Консолида	4234	23424	2344
10	Аренда		12702	70272	7032
11		4-Консолида	423	14123	1321
12		4-Консолида	423	14123	1321
13		4-Консолида	423	14123	1321
14	Амортизация		1269	42369	3963
18	Страховка		639	69	702
22	Заработная пла		15423	10272	1002
23					
24					

Рис. 8.11. Консолидирующая таблица

В результате будет создана консолидирующая таблица, показанная на рис. 8.11 (см. также файл *4-Консолидация данных по положению и категориям.xlsm* на компакт-диске).

Методы и свойства, используемые при программировании консолидирующей таблицы

Для программного конструирования консолидирующей таблицы используется метод `Consolidate` объекта `Range`. Этот метод позволяет подвести итоги и обобщить однородные данные, размещенные в нескольких диапазонах. На рабочем листе действия, программируемые методом `Consolidate`, выполняют команду **Консолидация**, расположенную в группе **Работа с данными** на вкладке **Данные** ленты.

expression.Consolidate(*Sources*, *Function*, *TopRow*, *LeftColumn*, *CreateLinks*)

- *expression* — ссылка на диапазон или ячейку, расположенную в левом верхнем его углу, где будет построена консолидирующая таблица.
- *Sources* — необязательный параметр, задающий массив ссылок в формате R1C1 на диапазоны, по которым строится консолидирующая таблица. Ссылки должны содержать полные имена диапазонов с указанием имен рабочих листов, на которых они расположены. Например, `Array("'Январь'!R1C1:R5C3", "'Февраль'!R1C1:R5C3")`.
- *Function* — необязательный параметр, специфицирующий функцию, на основе которой строится консолидирующая таблица. Допустимыми значениями являются следующие константы `XlConsilidationFunction`: `xlAverage` (среднее), `xlCount` (количество значений), `xlCountNums` (количество чисел), `xlMax` (максимум), `xlMin` (минимум), `xlProduct` (произведение), `xlStDev` (несмещенная дисперсия), `xlStDevP` (смещенная дисперсия), `xlSum` (сумма), `xlVar` (несмещенное отклонение) и `xlVarP` (смещенное отклонение).
- *TopRow* — необязательный параметр, принимающий логические значения. Показывает, основывается ли консолидация на заголовках столбцов консолидируемых диапазонов.
- *LeftColumn* — необязательный параметр, принимающий логические значения. Показывает, основывается ли консолидация на заголовках строк консолидируемых диапазонов.
- *CreateLinks* — необязательный параметр, принимающий логические значения. Показывает, связана ли консолидируемая таблица с исходными таблицами. Если параметр принимает значения `True`, то консолидируемая таблица выводится в виде структуры.

При консолидации данных также важную роль играют три свойства объекта `Worksheet`, приведенные в табл. 8.1.

Таблица 8.1. *Свойства объекта* `Worksheet`*, используемые при консолидации данных*

Свойство	Описание
`ConsolidationOptions`	Возвращает трехмерный массив. Первый его элемент показывает, основывается ли консолидация на заголовках столбцов. Второй элемент устанавливает, основывается ли консолидация на заголовках строк консолидируемых диапазонов. Третий элемент показывает, связана ли консолидируемая таблица с исходными таблицами
`ConsolidationFunction`	Возвращает константу `XlConsolidationFunction`, идентифицирующую функцию, на основе которой строится консолидирующая таблица
`ConsolidationSources`	Возвращает массив ссылок на диапазоны, на основе которых была построена на рабочем листе консолидирующая таблица. Если на рабочем листе таковой таблицы нет, то это свойство возвращает значение `Empty`

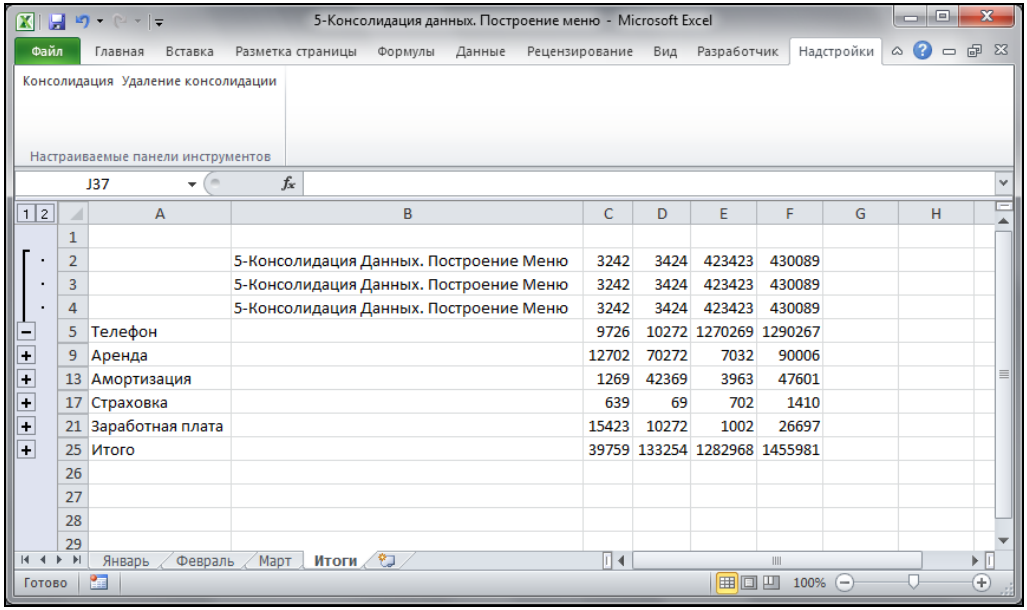
Пример приложения, консолидирующего данные

Продемонстрируем на примере бизнес-ситуации с построением итоговой таблицы расходов фирмы ООО "Альянс" за отчетный период, как в коде можно создавать и удалять консолидирующие таблицы. Для этого создайте рабочую книгу, имеющую несколько листов, скажем, **Январь**, **Февраль**, **Март**, с таблицами, как показано на рис. 8.8. Кроме того, в книге должен быть пустой лист **Итоги**. После этого в стандартный модуль и модуль `ЭтаКнига` надо добавить соответствующий код (см. также файл *5-Консолидация данных. Построение меню.xlsm* на компакт-диске).

> ***Примечание***
>
> В программе производится консолидация заранее не оговоренного числа таблиц. Поэтому в качестве значения параметра `Sources` метода `Consolidate` нельзя приводить массив `Array`, размерность которого заранее не известна. Из данного затруднения в программе выходим очень просто — вводя дополнительную переменную типа `Variant`, ей присваиваем значения динамического массива с адресами консолидированных таблиц. А уж потом, в качестве значения параметра `Sources` используем значение этой вспомогательной переменной.

В стандартном модуле имеются две процедуры, которые и реализуют бизнес-логику проекта:

- процедура `ConsolidationBuilder` осуществляет построение консолидирующей таблицы по любому числу листов с данными, имена которых отличны от имени листа с консолидирующей таблицей, т. е. от имени **Итоги**. Прежде чем производить требуемые построения, эта процедура проверяет наличие на листе **Итоги** какой-либо таблицы (точнее, наличие каких-либо данных в его первом столбце). В случае присутствия таковых, построения не производятся;
- процедура `ConsolidationKiller` удаляет консолидирующую таблицу. Точнее, она методом `ClearOutline` удаляет структуру, созданную этой таблицей, а также методом `Clear` очищает содержимое ячеек. Прежде чем производить удаление, эта процедура проверяет наличие структуры на рабочем листе, и в случае отсутствия таковой, удаление отменяется за его ненадобностью.



		A	B	C	D	E	F	G	H
	1								
	2		5-Консолидация Данных. Построение Меню	3242	3424	423423	430089		
	3		5-Консолидация Данных. Построение Меню	3242	3424	423423	430089		
	4		5-Консолидация Данных. Построение Меню	3242	3424	423423	430089		
−	5	Телефон		9726	10272	1270269	1290267		
+	9	Аренда		12702	70272	7032	90006		
+	13	Амортизация		1269	42369	3963	47601		
+	17	Страховка		639	69	702	1410		
+	21	Заработная плата		15423	10272	1002	26697		
+	25	Итого		39759	133254	1282968	1455981		
	26								
	27								
	28								
	29								

Рис. 8.12. Пример приложения, консолидирующего данные

В модуле `ЭтаКнига` имеются две процедуры, которые помещают на вкладку **Надстройка** ленты необходимые кнопки при открытии книги, и происходит их удаление при ее закрытии (рис. 8.12):

- процедура обработки события `Open` объекта `Workbook` конструирует панель инструментов **Консолидация** старого образца, на которой создается две кнопки: **Консолидация** и **Удаление консолидации**, отображающиеся на вкладке **Надстройка** ленты в группе **Настраиваемые панели инструментов**, которые вызывают выполнение процедур `ConsolidationBuilder` и `ConsolidationKiller`;
- процедура обработки события `BeforeClose` объекта `Workbook` удаляет созданную панель **Консолидация** с кнопками при закрытии книги (и. соответственно, вкладка **Надстройка** не будет отображаться при открытии других рабочих книг).

Структурируем рабочие листы

Цель структуризации заключается в разбиении данных, содержащихся на рабочем листе, на определенные уровни детализации. Используя структуру, легче проводить анализ и сравнение данных.

Если между данными имеется строгая зависимость, то MS Excel позволяет автоматически создать структуру: в этом случае MS Excel ищет ячейки, которые содержат формулы, обобщающие информацию в строках и которые расположены слева. Данные должны быть *согласованы в одном направлении*. Для выполнения автоматической структуризации все детальные столбцы должны стоять по одну сторону от итоговых столбцов, все детальные строки должны находиться по отношению к итоговым либо только снизу, либо — только сверху. Если это условие не соблюдается, то структуру следует создать вручную.

Рабочий лист может содержать только одну структуру, хотя ее можно разделить на несколько частей (рис. 8.13, см. также файл *6-Пример структуризации данных на Рабочем листе.xlsm* на компакт-диске).

Отображение и скрытие данных структуры может отразиться на частях рабочего листа, которые не участвуют в иерархии, т. к. строки сворачиваются и разворачиваются по всей ширине рабочего листа, а столбцы — по всей высоте рабочего листа (рис. 8.14).

	A	B	C	D	E	F	G	H	I	J
1	**Студенты**	**Мат.анали**	**ировани е програм**	**Философи**	**Практик ум на ЭВМ**	**Средний балл за 1 семестр**	**Диф.урав нения**	**Выч.сети**	**Филосо фия**	**Средний балл за 2 семестр**
2	Иванов	3	4	5	5	4,25	4	5	3	
3	Петров	5	5	5	5	5	5	5	5	
4	Федорова	4	3	3	3	3,25	5	4	3	
5	Заяц	5	4	3	5	4,25	3	4	4	3,666666
6	Николаев	4	5	5	4	4,5	5	5	5	
7	**Средний балл (1 группа)**	4,2	4,2	4,2	4,4	4,25	4,4	4,6	4	4,333333
8	Баглык	3	4	5	5	4,25	4	5	3	
9	Струк	5	5	5	5	5	5	5	5	
10	Иванченко	4	3	3	3	3,25	5	4	3	
11	Романов	5	4	3	5	4,25	3	4	4	3,666666
12	Сенько	4	5	5	4	4,5	5	5	5	
13	Кот	3	4	3	3	3,25	5	4	3	
14	Петухов	4	5	4	5	4,5	3	4	4	3,666666
15	Науменко	5	5	5	5	5	5	5	5	
16	**Средний балл (2 группа)**	4,125	4,375	4,125	4,375	4,25	4,375	4,5	4	4,291666
17	Ильяшевский	3	4	5	5	4,25	5	5	5	
18	Стефанович	5	5	5	5	5	5	4	3	

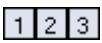
Рис. 8.13. Пример структуризации данных на рабочем листе

	A	F	J	K	L
1	**Студенты**	**Средний балл за 1 семестр**	**Средний балл за 2 семестр**	**Средний бал за учебный год**	
7	**Средний балл (1 группа)**	4,25	4,333333333	4,291666667	
16	**Средний балл (2 группа)**	4,25	4,291666667	4,270833333	
24	**Средний балл (3 группа)**	4,214285714	4,333333333	4,273809524	
25	**Всего (средний балл)**	4,238095238	4,319444444	4,278769841	
26					
27					

Рис. 8.14. Скрытие низших уровней в структуре данных

При выводе структуры по левому и верхнему краям рабочего листа отображаются специальные символы, которые служат для вывода и скрытия уровней детализации (табл. 8.2).

Таблица 8.2. *Символы структуры*

Символ структуры	Назначение
	Кнопка для показа детальных данных
	Кнопка для скрытия соответствующих детальных данных
 Номера уровней	Последовательные уровни для строк и столбцов
 Уровень структуры	Все детальные строки или детальные столбцы одного уровня

Для автоматического создания структуры следует:

- проверить, что в итоговых формулах содержатся ссылки на детальные данные, расположенные в одном направлении относительно итоговых;
- для структуризации части рабочего листа необходимо выделить нужный диапазон ячеек; для структуризации всего рабочего листа — выбрать одну ячейку;
- воспользоваться командой **Создание структуры**, выбрав ее из списка **Группировать**, который расположен в группе **Структура** на вкладке **Данные** ленты.

При структуризации рабочего листа "вручную" необходимо:

- выделить нужные ячейки строк и столбцов, которые подлежат объединению в структуру, за исключением ячейки с итоговой формулой;
- воспользоваться командой **Группировать**, выбрав ее из списка **Группировать,** который расположен в группе **Структура** на вкладке **Данные** ленты;
- в случае ошибочных действий или для разгруппировки данных выбрать команду **Разгруппировать** из списка **Разгруппировать**, который расположен в группе **Структура** на вкладке **Данные** ленты;
- для отображения или скрытия данных структуры следует использовать команды **Отобразить детали** и **Скрыть детали**, расположенные также в группе команд **Структура** на вкладке **Данные** ленты;
- для возврата рабочего листа в исходное состояние следует использовать команду **Удалить структуру**, выбрав ее из списка **Разгруппировать**, который расположен в группе **Структура** на вкладке **Данные** ленты.

Для структурированных данных имеется возможность создавать диаграммы с заданных уровней структуры.

Структура и объект *Outline*

Объект `Outline` инкапсулирует в себе данные о структуре. Свойство `Outline` рабочего листа возвращает объект `Outline`. В табл. 8.3 представлены основные свойства объекта `Outline`.

Таблица 8.3. *Основные свойства объекта* `Outline`

Свойство	Описание
`AutomaticStyles`	Принимает логические значения. Если значение этого свойства равно `True`, то структура строится на основе автоматических стилей
`SummaryColumn`	Возвращает местоположение итоговых столбцов. Допустимыми значениями являются следующие константы `XlSummaryColumn`: `xlLeft` (итоговые столбцы располагаются слева от столбцов, по которым подводятся итоги), `xlRight` (итоговые столбцы находятся справа)
`SummaryRow`	Возвращает местоположение итоговых строк. Допустимыми значениями являются следующие константы `XlSummaryRow`: `xlAbove` (итоговые строки располагаются выше строк, по которым подводятся итоги), `xlBelow` (итоговые строки располагаются ниже)

Отображение указанного числа уровней структуры

Объект `Outline` имеет единственный метод `ShowLevels`, который отображает указанное число уровней структуры по строкам и столбцам.

```
ShowLevels(RowLevels, ColumnLevels)
```

- *`RowLevels`* — необязательный параметр, устанавливающий число отображаемых уровней структуры по строкам.
- *`ColumnLevels`* — необязательный параметр, задающий число отображаемых уровней структуры по столбцам.

Удаление структуры

Метод `ClearOutline` объекта `Range` удаляет структуру. Например, следующая инструкция удаляет структуру, связанную с диапазоном **A1:I40**:

```
Range("A1:I40").ClearOutline
```

Отображение значков структуры

Метод `DisplayOutline` объекта `Window` принимает логические значения и управляет отображением значков структуры. Например, данная инструкция скрывает значки структуры:

```
ActiveWindow.DisplayOutline = False
```

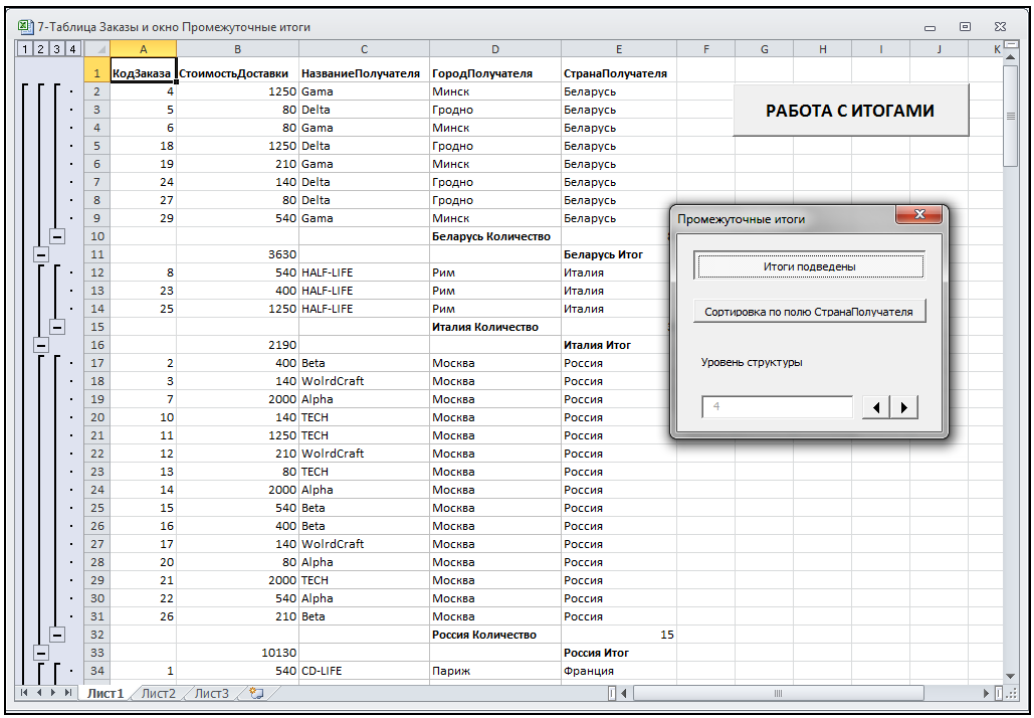
Автоматическое создание структуры

Метод `AutoOutline` объекта `Range` автоматически создает структуру, которая заменяет уже существующую. Если объект `Range` является ячейкой, то структура создается для всего листа. Например, следующая инструкция создает структуру для диапазона **A1:I40**:

```
Range("A1:I40").AutoOutline
```

Пример приложения, подводящего промежуточные итоги и управляющего структурой

Применим метод `Subtotal` и объект `Outline` для решения несложной задачи. Воспользуемся списком данных со следующими полями: `КодЗаказа`, `СтоимостьДоставки`, `НазваниеПолучателя`, `ГородПолучателя`, `СтранаПолучателя`, который отражает необходимые расходы по доставке заказов конкретным заказчикам (рис. 8.15). Нам необходимо получить итоговые данные по количеству заказов, сделанных каждым из клиентов, и суммарные почтовые расходы по доставке этих заказов для каждой страны.



	КодЗаказа	СтоимостьДоставки	НазваниеПолучателя	ГородПолучателя	СтранаПолучателя
2	4	1250	Gama	Минск	Беларусь
3	5	80	Delta	Гродно	Беларусь
4	6	80	Gama	Минск	Беларусь
5	18	1250	Delta	Гродно	Беларусь
6	19	210	Gama	Минск	Беларусь
7	24	140	Delta	Гродно	Беларусь
8	27	80	Delta	Гродно	Беларусь
9	29	540	Gama	Минск	Беларусь
10				**Беларусь Количество**	
11		3630			**Беларусь Итог**
12	8	540	HALF-LIFE	Рим	Италия
13	23	400	HALF-LIFE	Рим	Италия
14	25	1250	HALF-LIFE	Рим	Италия
15				**Италия Количество**	
16		2190			**Италия Итог**
17	2	400	Beta	Москва	Россия
18	3	140	WolrdCraft	Москва	Россия
19	7	2000	Alpha	Москва	Россия
20	10	140	TECH	Москва	Россия
21	11	1250	TECH	Москва	Россия
22	12	210	WolrdCraft	Москва	Россия
23	13	80	TECH	Москва	Россия
24	14	2000	Alpha	Москва	Россия
25	15	540	Beta	Москва	Россия
26	16	400	Beta	Москва	Россия
27	17	140	WolrdCraft	Москва	Россия
28	20	80	Alpha	Москва	Россия
29	21	2000	TECH	Москва	Россия
30	22	540	Alpha	Москва	Россия
31	26	210	Beta	Москва	Россия
32				**Россия Количество**	15
33		10130			**Россия Итог**
34	1	540	CD-LIFE	Париж	Франция

Рис. 8.15. Список **Заказы** и окно **Промежуточные итоги**

Создайте форму, на ней расположите выключатель, кнопку, счетчик, поле ввода и надпись. При установленном выключателе на нем будет отображаться надпись **Итоги подведены**, а на рабочем листе будут создаваться промежуточные итоги, подсчитывающие количество заказов, сделанных каждым из клиентов, и суммарные почтовые расходы по доставке этих заказов.

Нажатие кнопки **Сортировка по полю СтранаПолучателя** обеспечивает сортировку по полю **СтранаПолучателя** списка, что необходимо выполнить перед тем, как создавать итоги.

Счетчик позволит управлять отображением различных уровней структуры промежуточных итогов. При снятом выключателе на нем отображается надпись **Итоги удалены**, а промежуточные итоги удаляются с рабочего листа. Они также удаля-

ются при закрытии диалогового окна. Итак, для завершения создания приложения в модуле формы наберите соответствующий код (см. файл *7-Таблица Заказы и окно Промежуточные итоги.xlsm* на компакт-диске).

Используем сценарии

Каждое уникальное значение в ячейке или каждая уникальная группа значений для группы ячеек называется *сценарием*. Сценарии позволяют проводить так называемый анализ данных "что, если". В ключевые ячейки можно вводить различные значения и смотреть, что при этом происходит. Довольно часто необходимо иметь под рукой различные варианты решения, а сценарии как раз и предоставляют эту возможность пользователю.

Диспетчер сценариев в MS Excel позволяет автоматически выполнить анализ "что, если" для различных моделей. Можно создать несколько входных наборов данных (изменяемых ячеек) для любого количества переменных и присвоить имя каждому набору. По имени выбранного набора данных MS Excel сформирует результаты анализа на рабочем листе. Кроме этого, диспетчер сценариев позволяет создать итоговый отчет по сценариям, в котором отображаются результаты подстановки различных комбинаций входных параметров.

Диспетчер сценариев открывается командой **Диспетчер сценариев**, которая выбирается из списка **Анализ "что-если"**, расположенном в группе **Работа с данными** на вкладке **Данные** ленты (рис. 8.16).

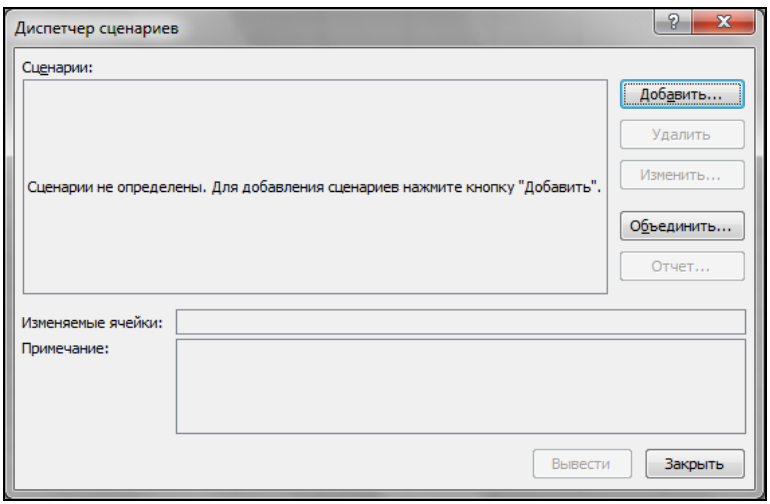


Рис. 8.16. Окно **Диспетчер сценариев**

В появившемся окне с помощью соответствующих кнопок можно добавить новый сценарий, изменить, удалить или вывести существующий, а также объединить несколько различных сценариев и получить итоговый отчет для существующих сценариев.

Расчет внутренней скорости оборота инвестиций

Рассмотрим пример (см. файл *8-Расчет внутренней скорости оборота инвестиций.xlsm* на компакт-диске). Пусть затраты по проекту составят 700 млн руб. Ожидаемые доходы в течение последующих 5 лет составят соответственно 70 млн руб., 90 млн руб., 300 млн руб., 250 млн руб., 300 млн руб. Оценить экономическую целесообразность проекта по скорости оборота инвестиции, если рыночная норма дохода 12%. Рассмотреть также следующие варианты (затраты на проект — число со знаком "минус"): (–600; 50; 100; 200; 200; 300), (–650; 90; 120; 200; 250; 250), (–500, 100, 100, 200, 250, 250).

Для вычисления внутренней скорости оборота инвестиции (внутренней нормы доходности) используется функция:

`ВСД(`*`Значения; Предположения`*`)`

В данном случае функция для решения задачи использует только аргумент *`Значения`*, один из которых обязательно отрицателен (затраты по проекту). Если внутренняя скорость оборота инвестиций будет больше рыночной нормы доходности, то проект считается экономически целесообразным. В противном случае проект должен быть отвергнут.

Решение для данного примера приведено на рис. 8.17. Формулы для расчета:

- в ячейке **B84**: `=ВСД(B75:B80)`;
- в ячейке **C84**: `=ЕСЛИ(B84>B82;"Проект экономически целесообразен";"Проект необходимо отвергнуть")`.

	A	B	C	D	E
70					
71					
72	**Расчет внутренней скорости оборота инвестиций**				
73					
74	Ожидаемые доходы в течение	5	лет		
75	Затраты по проекту	-600 000 000,00р.			
76	Первый год	50 000 000,00р.			
77	Второй год	100 000 000,00р.			
78	Третий год	300 000 000,00р.			
79	Четвертый год	200 000 000,00р.			
80	Пятый год	300 000 000,00р.			
81					
82	Рыночная норма дохода	12%			
83					
84	Внутренняя скорость оборота инвестиций	14%	Проект экономически целесообразен		
85					

Рис. 8.17. Расчет внутренней скорости оборота инвестиций

Рассмотрим данный пример для всех комбинаций исходных данных. Для создания (или изменения) сценария следует использовать команду **Диспетчер сценариев**, которая выбирается из списка **Анализ "что-если"**, расположенном в группе **Работа с данными** на вкладке **Данные** ленты. В открывшемся окне **Диспетчер сценариев** (см. рис. 8.16) нажмите кнопку **Добавить** для добавления нового сценария.

В окне **Добавление сценария** (рис. 8.18) введите новое название для сценария и установите другие необходимые параметры. После нажатия кнопки **OK** появляется возможность внесения новых значений для изменяемых ячеек (рис. 8.19). Для

сохранения результатов по первому сценарию, нет необходимости редактировать значения ячеек, достаточно нажать кнопку **OK** для подтверждения значений, появившихся по умолчанию, и выхода в окно **Диспетчер сценариев** (рис. 8.20).

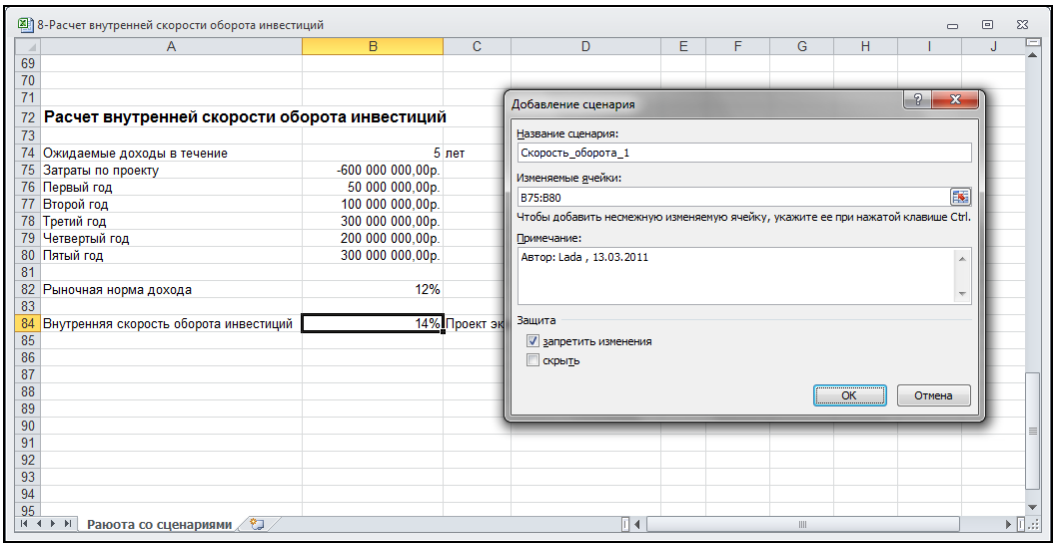


Рис. 8.18. Добавление сценария для первой комбинации исходных данных

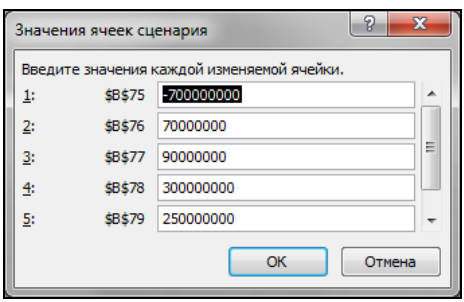


Рис. 8.19. Окно для изменения значений ячеек сценария

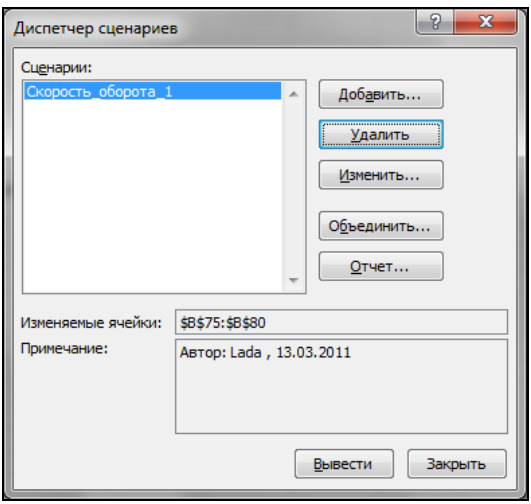


Рис. 8.20. Окно **Диспетчер сценариев** с первым сохраненным сценарием

Для добавления новых сценариев для рассматриваемой задачи достаточно нажать кнопку **Добавить** в окне **Диспетчер сценариев** и повторить вышеописанные действия, изменив значения в ячейках исходных данных. На рис. 8.21 сценарий *Скорость_оборота_1* соответствует данным (–700; 70; 90; 300; 250; 300), сценарий

Скорость_оборота_2 — данным (–600; 50; 100; 200; 200; 300), сценарий *Скорость_оборота_3* — данным (–650; 90; 120; 200; 250; 250), сценарий *Скорость_оборота_4* — данным (–500, 100, 100, 200, 250, 250). Нажав кнопку **Вывести**, можно просмотреть на рабочем листе результаты расчета для соответствующей комбинации исходных значений.

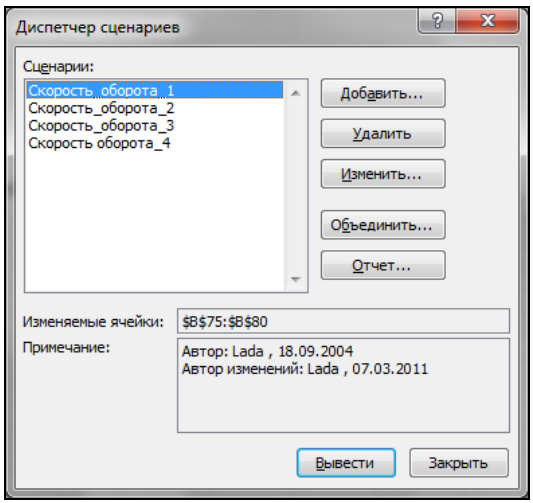


Рис. 8.21. Окно **Диспетчер сценариев** с добавленными сценариям по расчету скорости оборота инвестиций

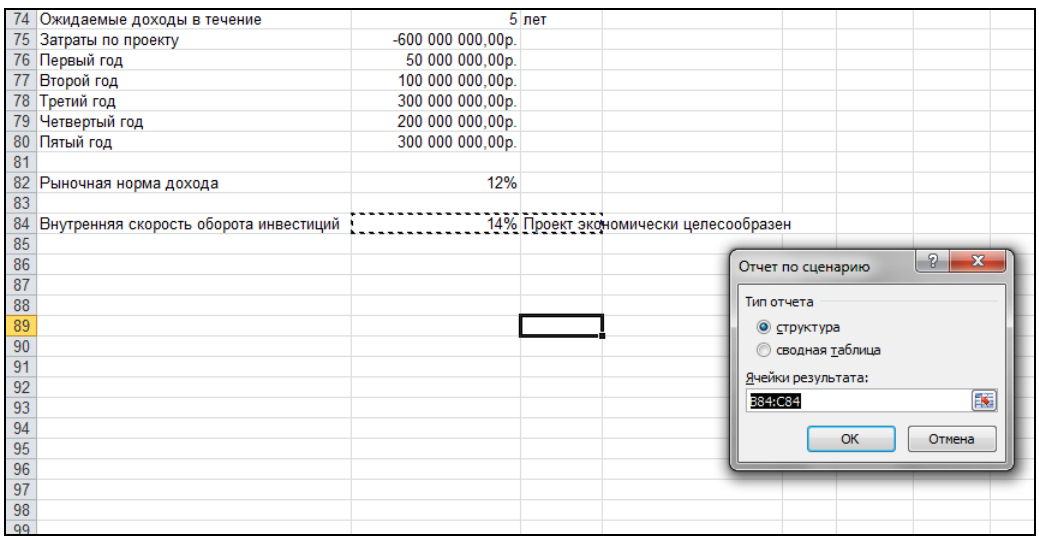
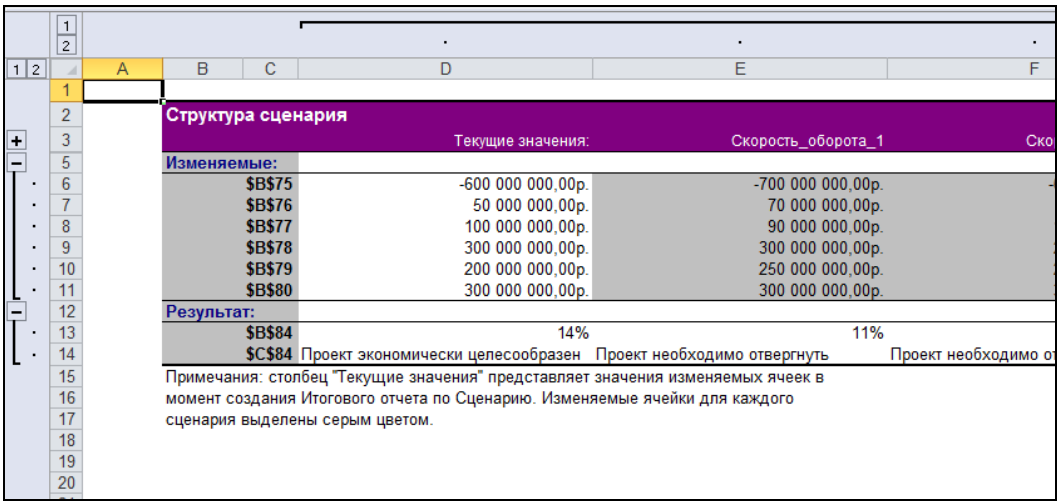


Рис. 8.22. Добавление ячеек результата в окно **Отчет по сценарию**

Для получения итогового отчета по всем добавленным сценариям достаточно нажать кнопку **Отчет** в окне **Диспетчера сценариев**. В появившемся окне **Отчет по сценарию** (рис. 8.22) следует выбрать требуемый тип отчета и дать ссылки

на ячейки, в которых вычисляются результирующие функции. При нажатии кнопки **OK**, на соответствующий лист рабочей книги выводится отчет по сценариям (рис. 8.23 и 8.24).



Структура сценария			
	Текущие значения:	Скорость_оборота_1	Ско
Изменяемые:			
$B$75	-600 000 000,00р.	-700 000 000,00р.	-
$B$76	50 000 000,00р.	70 000 000,00р.	
$B$77	100 000 000,00р.	90 000 000,00р.	
$B$78	300 000 000,00р.	300 000 000,00р.	
$B$79	200 000 000,00р.	250 000 000,00р.	
$B$80	300 000 000,00р.	300 000 000,00р.	
Результат:			
$B$84	14%	11%	
$C$84	Проект экономически целесообразен	Проект необходимо отвергнуть	Проект необходимо от

Примечания: столбец "Текущие значения" представляет значения изменяемых ячеек в момент создания Итогового отчета по Сценарию. Изменяемые ячейки для каждого сценария выделены серым цветом.

Рис. 8.23. Отчет типа **Структура**
по сценариям расчета скорости оборота инвестиций

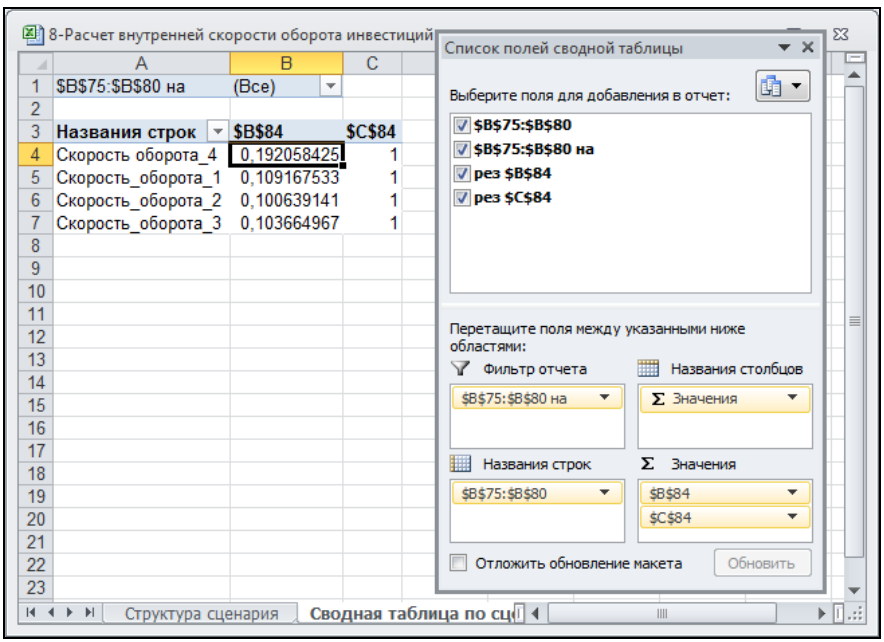


Рис. 8.24. Отчет типа **Сводная таблица**
по сценариям расчета скорости оборота инвестиций

Объект *Scenario*

Объект `Scenario` позволяет хранить несколько значений в одной ячейке и представляет собой сценарий. Семейство `Scenarios` состоит из объектов `Scenario` и содержит в себе все сценарии рабочего листа.

В табл. 8.4 перечислены наиболее важные методы семейства `Scenarios`.

Таблица 8.4. *Методы семейства* `Scenarios`

Метод	Описание
`Add`	Добавляет новый сценарий. • `Add(`*`Name`*`, `*`ChangingCells`*`, `*`Values`*`, `*`Comment`*`, `*`Locked`*`, `*`Hidden`*`)` • *`Name`* — имя сценария; • *`ChangingCells`* — диапазон, отводимый под изменяемые ячейки сценария; • *`Values`* — массив значений, вводимых в изменяемые ячейки; • *`Comment`* — текстовая строка комментариев; • *`Locked`* — свойство, принимающее логические значения. Если оно равно значению `True`, заблокировано изменение сценария; • *`Hidden`* — свойство, принимающее логические значения. Если оно равно значению `True`, сценарий скрыт
`CreateSummary`	Добавляет в книгу новый рабочий лист, в котором создает отчет. `CreateSummary(`*`ReportType`*`, `*`ResultCells`*`)` • *`ReportType`* — тип отчета. Допустимые значения: ◊ `xlStandardSummary` (стандартный отчет типа структуры); ◊ `xlSummaryPivotTable` (отчет в виде сводной таблицы); • *`ResultCells`* — ссылка на ячейку или диапазон ячеек с формулами, которые зависят от значений из ячеек, указанных в параметре `ChangingCells` метода `Add`. Отчет создается на отдельном рабочем листе и не связан с исходными данными. Очень полезно перед составлением отчета присвоить имена ячейкам, заданным в параметре *`ResultCells`*. В противном случае вместо понятных имен в отчете будут помещены малопонятные ссылки на ячейки

В табл. 8.5 и 8.6 перечислены методы и свойства объекта `Scenario`.

Таблица 8.5. *Методы объекта* `Scenario`

Метод	Описание
`Show`	Показывает сценарий, вводя значения сценария в изменяемые ячейки
`Delete`	Удаляет сценарий
`ChangeScenario`	Изменяет группу изменяемых ячеек в сценарии. `ChangeScenario(`*`ChangingCells`*`, `*`Values`*`)` • *`ChangingCells`* — группа ячеек, которая будет играть роль новой группы изменяемых ячеек; • *`Values`* — массив с новыми значениями изменяемых ячеек

Таблица 8.6. *Свойства объекта* `Scenario`

Свойство	Описание
`ChangingCells`	Возвращает диапазон изменяемых ячеек. Например: `ActiveSheet.Scenarios(1).ChangingCells.Select`

`Values`	Возвращает массив текущих значений изменяемых ячеек. Например:
	`ActiveSheet.Scenarios(1).Values = Range("B1:B3")`
	или
	`ActiveSheet.Scenarios(1).Values = Array(1, 3, 5)`

Пример приложения по работе со сценариями

На простом примере покажем, как работает объект `Scenario`. Составим таблицу расходов ООО "Мегатоп" за январь (рис. 8.25) и спрогнозируем расходы на следующий месяц. Свои прогнозы построим на основе предположения, что относительная величина расходов в феврале останется прежней, а их абсолютная величина увеличится с учетом инфляции. Будем считать, что инфляция в феврале будет в лучшем случае 1%, в худшем — 7%, а в наиболее вероятном случае — 3%.

	A	B	C	D	E	F	G	H
1		Расходы, январь	Ожидаемые расходы, февраль		Выбранный вариант			
2	Телефон	3213	3437,91		0,07			
3	Аренда	32131	34380,17					
4	Амортизация	5122	5480,54					
5	Страховка	32543	34821,01					
6	Заработная плата	435435	465915,45					
7	Итого	508444	544035,08					
8								
9			Варианты инфляции		Худший	7%		
10			Худший	7%	Ожидаемый	3%		
11			Ожидаемый	3%	Лучший	1%		
12			Лучший	1%				
13								
14								

Рис. 8.25. Сценарии расходов ООО "Мегатоп"

Оформим работу со сценариями следующим образом (см. файл *9-Пример приложения по работе со сценариями.xlsm* на компакт-диске):

1. На рабочем листе расположите один список, установите при помощи окна **Properties** значение его свойства `Name` равным `lstScenarios`. При открытии книги за счет обработки события `Open` объекта `Workbook` будет происходить заполнение списка названиями возможных сценариев инфляции в следующем месяце.
   Щелчок на элементе списка приведет:
   - к вводу значения уровня выбранного сценария инфляции в ячейку **E2**;
   - к расчету предполагаемых расходов на следующий месяц.

2. В ячейку **B7** введите формулу, определяющую суммарные расходы:
   `=СУММ(B2:B6)`
3. Выделите диапазон **C2:C7** и введите в него следующую формулу:
   `{=B2:B7*(1+E2)}`
   причем ее ввод надо завершить нажатием комбинации клавиш <Ctrl>+<Shift>+ +<Enter>, т. к. это формула массива. Такая формула позволяет определить ожидаемые расходы для целого диапазона значений.
   В коде такие сценарии будут иметь вид, представленный в листинге 8.2.

Листинг 8.2, *а*. Сценарии расходов на основе объекта `Scenario`. Модуль `ЭтаКнига`

```
Private Sub Workbook_Open()
   Dim sc As Scenario
   Dim i As Integer
   Dim V As Variant
   For Each sc In Worksheets("Январь").Scenarios
      sc.Delete
   Next
   With Worksheets("Январь")
      For i = 10 To 12
         V = .Cells(i, 4).Value
        .Scenarios.Add Name:=.Cells(i, 3).Value, _
                       ChangingCells:=.Range("E2"), Values:=V
      Next
      With .lstScenarios
         .ColumnCount = 2
         .ListFillRange = "C10:D12"
         .BoundColumn = 2
         .ListIndex = 0
     End With
   End With
End Sub
```

Листинг 8.2, *б*. Сценарии расходов на основе объекта `Scenario`. Модуль рабочего листа *Январь*

```
Private Sub lstScenarios_Click()
   Worksheets("Январь").Scenarios(lstScenarios.Text).Show
End Sub
```

Создаем сводные таблицы

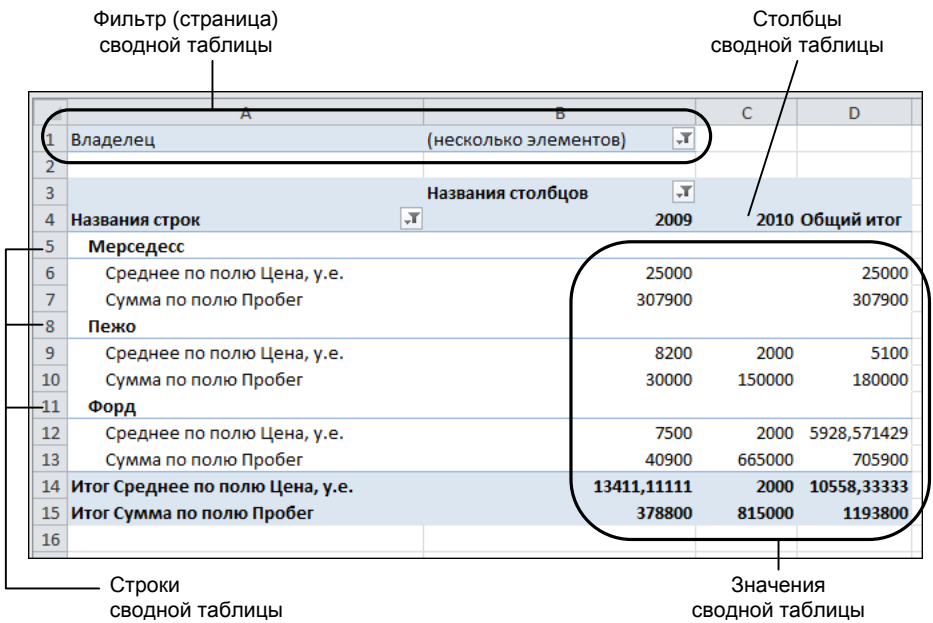
Сводные таблицы представляют собой средство для группировки, обобщения и анализа данных, находящихся в списках MS Excel или в таблицах, созданных в других приложениях. Сводные таблицы могут использоваться для:

- обобщения большого количества однотипных данных;
- реорганизации данных;

- ❒ отбора и группировки данных;
- ❒ построения диаграмм.

Внешне сводные таблицы являются структурой, позволяющей размещать данные в трехмерном виде. До того как вы начнете создавать сводную таблицу, желательно продумать ее логику, ее структуру (рис. 8.26). Так, необходимо определить следующие поля, которые будут использоваться в макете сводной таблице:

- ❒ поля для строк и столбцов таблицы;
- ❒ поля, по которым подводятся итоги (с выбором необходимой операции) — размещаются на пересечении строк и столбцов;
- ❒ поля для фильтра (страницы сводной таблицы) — для осуществления необходимых срезов (фильтров), что позволяет представить информацию в трехмерном виде.



	A	B	C	D
1	Владелец	(несколько элементов)		
2				
3		Названия столбцов		
4	Названия строк	2009	2010	Общий итог
5	Мерседесс			
6	Среднее по полю Цена, у.е.	25000		25000
7	Сумма по полю Пробег	307900		307900
8	Пежо			
9	Среднее по полю Цена, у.е.	8200	2000	5100
10	Сумма по полю Пробег	30000	150000	180000
11	Форд			
12	Среднее по полю Цена, у.е.	7500	2000	5928,571429
13	Сумма по полю Пробег	40900	665000	705900
14	Итог Среднее по полю Цена, у.е.	13411,11111	2000	10558,33333
15	Итог Сумма по полю Пробег	378800	815000	1193800
16				

Рис. 8.26. Основные элементы макета сводной таблицы

Сводные таблицы создаются с использованием команды **Сводная таблица**, которая выбирается из одноименного списка, расположенного в группе **Таблицы** на вкладке **Вставка**.

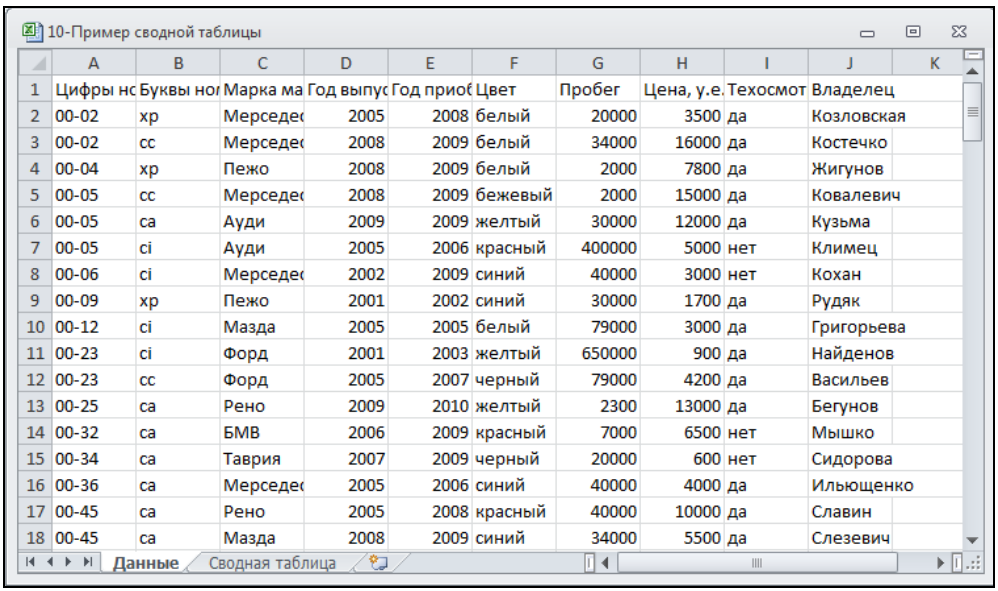
Совет

Располагайте, при возможности, сводную таблицу на отдельном листе, т. к. при обновлении, группировках данной сводной таблицы информация, содержащаяся на рабочих листах рядом со сводной таблицей, может оказаться скрытой.

Пример создания сводной таблицы на рабочем листе Excel

Итак, прежде чем перечислить различные возможности и операции, доступные для сводных таблиц, рассмотрим пример ее подготовки встроенными средствами Microsoft Office Excel 2010.

Пусть на рабочем листе **Данные** имеется список машин с данными. Поля этого списка: `Цифры номера`, `Буквы номера`, `Марка машины`, `Год выпуска`, `Год приобретения`, `Цвет`, `Пробег`, `Цена, у.е.`, `Техосмотр`, `Владелец` (рис. 8.27).



10-Пример сводной таблицы

	A	B	C	D	E	F	G	H	I	J	K
1	Цифры но	Буквы ног	Марка ма	Год выпус	Год прио	Цвет	Пробег	Цена, у.е.	Техосмот	Владелец	
2	00-02	хр	Мерседе	2005	2008	белый	20000	3500	да	Козловская	
3	00-02	сс	Мерседе	2008	2009	белый	34000	16000	да	Костечко	
4	00-04	хр	Пежо	2008	2009	белый	2000	7800	да	Жигунов	
5	00-05	сс	Мерседе	2008	2009	бежевый	2000	15000	да	Ковалевич	
6	00-05	са	Ауди	2009	2009	желтый	30000	12000	да	Кузьма	
7	00-05	сі	Ауди	2005	2006	красный	400000	5000	нет	Климец	
8	00-06	сі	Мерседе	2002	2009	синий	40000	3000	нет	Кохан	
9	00-09	хр	Пежо	2001	2002	синий	30000	1700	да	Рудяк	
10	00-12	сі	Мазда	2005	2005	белый	79000	3000	да	Григорьева	
11	00-23	сі	Форд	2001	2003	желтый	650000	900	да	Найденов	
12	00-23	сс	Форд	2005	2007	черный	79000	4200	да	Васильев	
13	00-25	са	Рено	2009	2010	желтый	2300	13000	да	Бегунов	
14	00-32	са	БМВ	2006	2009	красный	7000	6500	нет	Мышко	
15	00-34	са	Таврия	2007	2009	черный	20000	600	нет	Сидорова	
16	00-36	са	Мерседе	2005	2006	синий	40000	4000	да	Ильющенко	
17	00-45	са	Рено	2005	2008	красный	40000	10000	да	Славин	
18	00-45	са	Мазда	2008	2009	синий	34000	5500	да	Слезевич	

Данные | Сводная таблица

Рис. 8.27. Список автомобилей для создания сводной таблицы

Выполните последовательно следующие действия (см. также файл *10-Пример сводной таблицы.xlsm* на компакт-диске).

1. Перейдите на новый лист в рабочей книге и выполните команду **Сводная таблица**, которая выбирается из списка **Сводная таблица**, расположенного в группе **Таблицы** на вкладке **Вставка**.
2. В открывшемся окне **Создание сводной таблицы** в поле **Таблица или диапазон** введите ссылку на данные о списке машин (рис. 8.28) и нажмите кнопку **OK**.

   Примечание

   В качестве данных для сводной таблицы вы можете использовать данные списка Excel, внешний источник данных (например, данные из таблиц баз данных), диапазоны консолидации, находящиеся в другой сводной таблице.

3. На рабочем листе будет отведено место под создание таблицы и выведены необходимые инструменты, а также контекстные вкладки **Параметры** и **Конструктор** режима **Работа со сводными таблицами** (рис. 8.29).

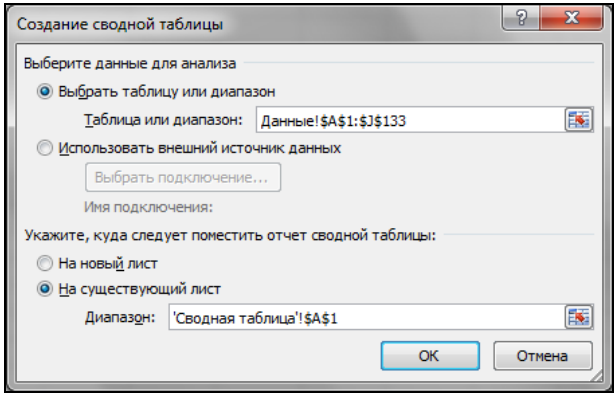


Рис. 8.28. Определение местоположения данных для сводной таблицы

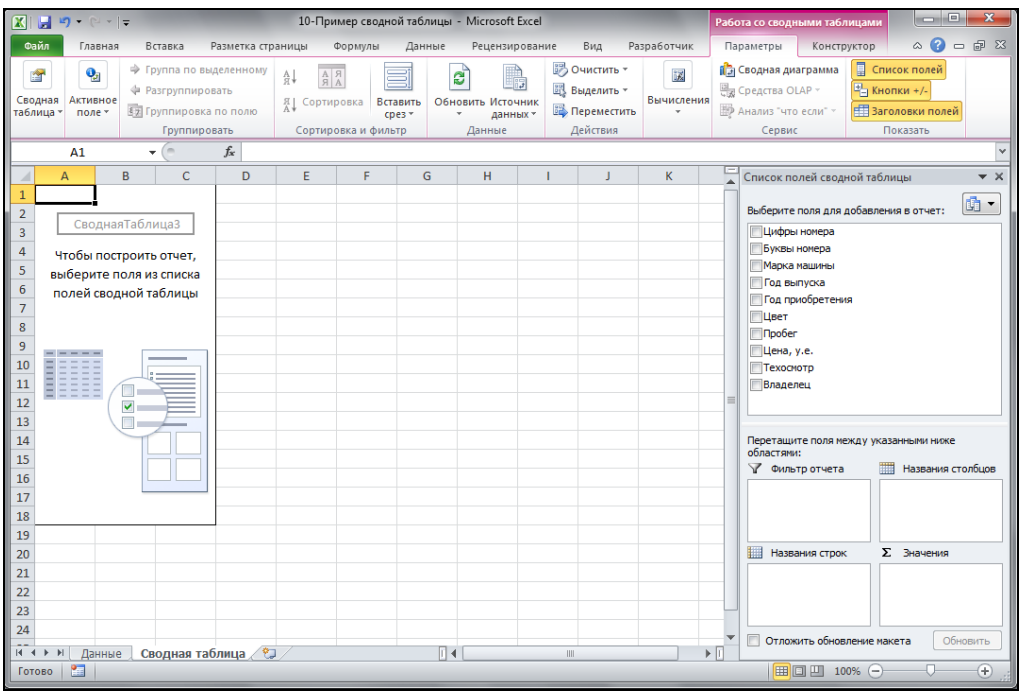


Рис. 8.29. Режим работы со сводными таблицами

4. Определите необходимые элементы сводной таблицы, задав соответствующие поля для фильтра, строк, столбцов и значений в окне **Список полей сводной таблицы** (рис. 8.30). Удобнее всего это сделать перетягиванием соответствующего поля из верхней части окна в нижнюю.

Примечание

Элемент области **Значения** Σ Значения содержит всегда вычисляемые данные. Поэтому по умолчанию в качестве базовой операции добавляется `Сумма`. Чтобы из-

менить итоговую операцию или же задать вычисляемое поле для области **Значения**, щелкните по кнопке с выпадающим списком, которая находится справа от поля в области **Значения**, и выберите команду **Параметры полей значений**. В открывшемся окне **Параметры поля значений** (рис. 8.31) вы можете выбрать необходимую итоговую операцию (на вкладке **Операция**), определить дополнительные вычисления (на вкладке **Дополнительные вычисления**), задать формат отображения данных (воспользовавшись кнопкой **Формат**).

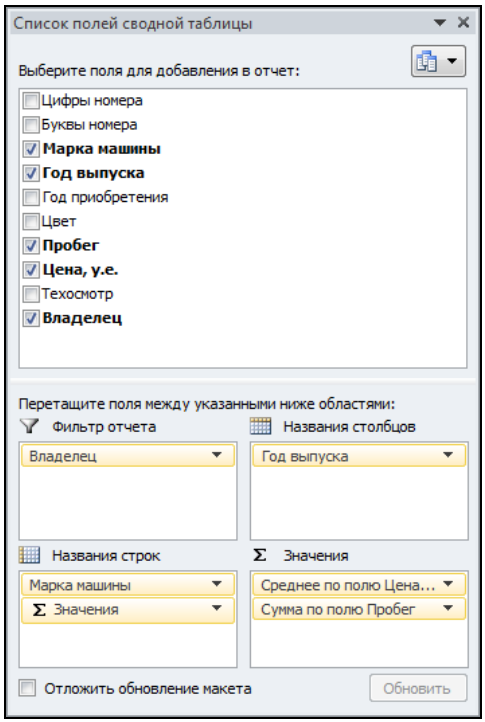


Рис. 8.30. Окно **Список полей сводной таблицы**

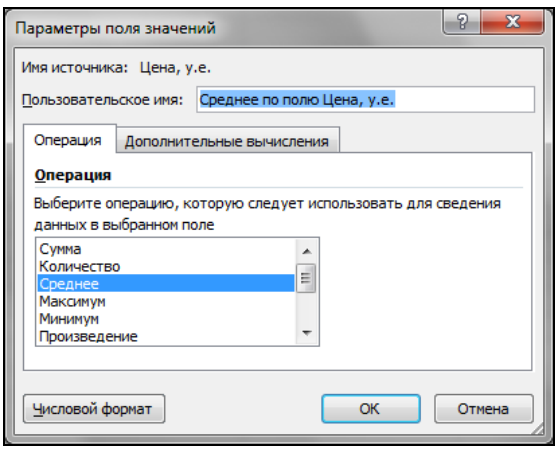


Рис. 8.31. Окно **Параметры поля значений**

5. Создайте вычисляемое поле **Эксплуатация**, которое будет определяться по следующей формуле:

   `='Год приобретения'-'Год выпуска'`

   Для его определения перейдите на контекстную вкладку **Параметры** режима **Работа со сводными таблицами** и в группе **Вычисления** выберите из списка **Поля, вычисления и наборы** команду **Вычисляемое поле**. В открывшемся окне создайте требуемое поле, используя при этом возможности, предоставляемые окном **Вставка вычисляемого поля** (рис. 8.32). После его задания оно появится в верхней части окна **Список полей сводной таблицы**. Добавьте это поле в область значения, выбрав итоговую операцию `Сумма` (рис. 8.33).

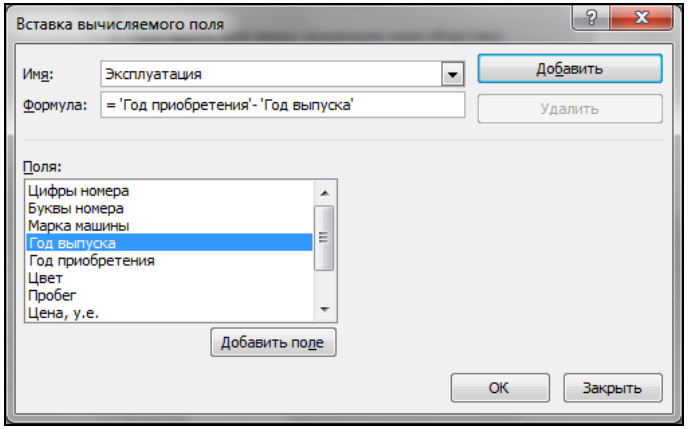


Рис. 8.32. Окно **Вставка вычисляемого поля**

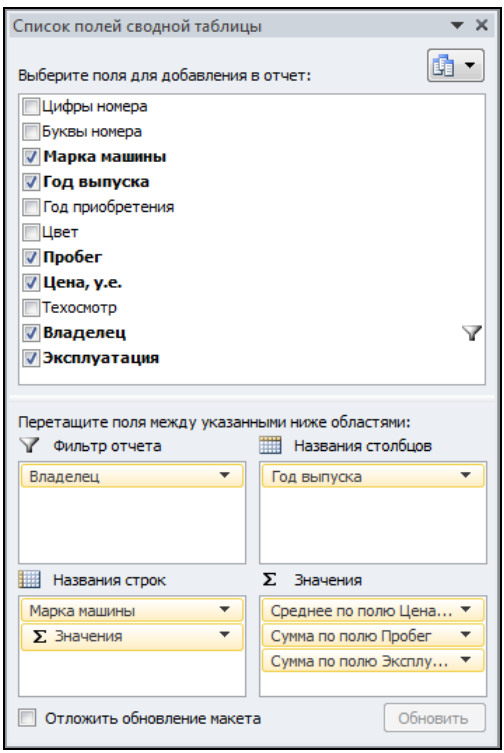


Рис. 8.33. Вычисляемое поле в списке полей сводной таблицы

Совет

Для того чтобы сводная таблица была удобочитаемой, переместите поле **Значения**, которое появляется после добавления полей в область значений, из области **Названия столбцов** в область **Названия строк**.

6. Добавьте еще одно поле **Техосмотр** в качестве фильтра для сводной таблицы.
7. Выполните группировку данных по полю **Год выпуска**: установите указатель ячейки в строку с названиями столбцов и выполните команду **Группировка по полю**, которая расположена в группе **Группировать** на контекстной вкладке **Параметры** режима **Работа со сводными таблицами**. В открывшемся окне **Группирование** задайте начало и конец для дат, а также шаг группирования (рис. 8.34).

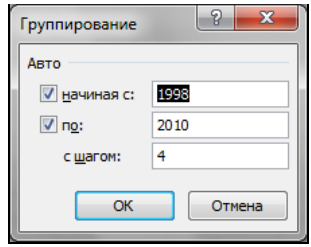


Рис. 8.34. Окно **Группирование**

Примечание

При необходимости выполните группировку также и для строк сводной таблицы, задав произвольно группы: выделите необходимое количество строк и воспользуйтесь командой команду **Группа по выделенному**, которая расположена в группе **Группировать** на контекстной вкладке **Параметры** режима **Работа со сводными таблицами**. Соответствующая кнопка **Разгруппировать** поможет удалить созданную группу.

8. Отформатируйте сводную таблицу, используя один из предложенных стилей, которые располагаются в коллекции в соответствующей группе **Стили сводной таблицы** на контекстной вкладке **Конструктор** режима **Работа со сводными таблицами**.
9. Подготовленная сводная таблица представлена на рис. 8.35.

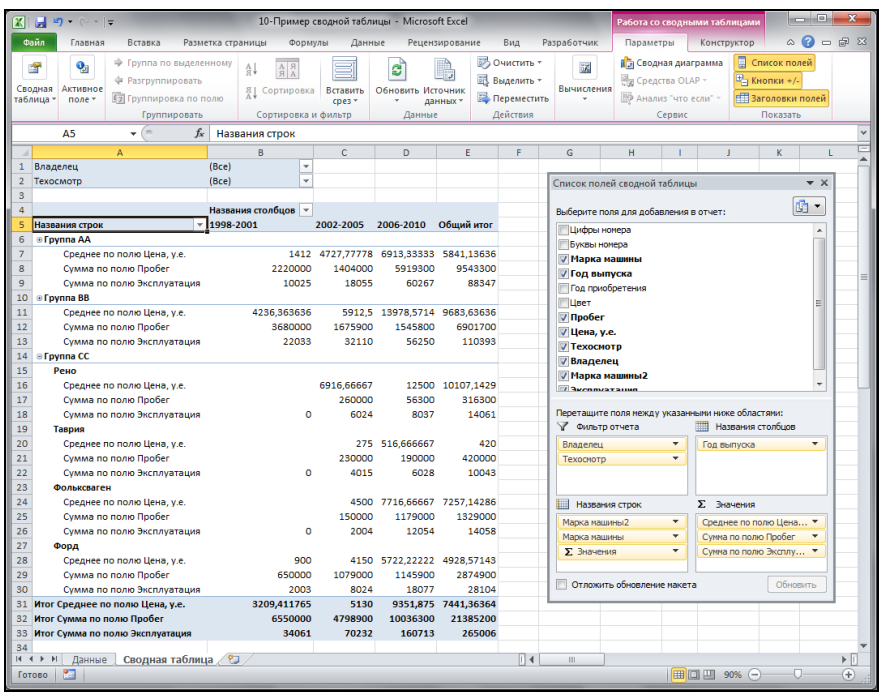


Рис. 8.35. Пример сводной таблицы

ПРИМЕЧАНИЕ

Сводная таблица — это средство только для отображения данных. Поэтому в самой сводной таблице данные редактировать нельзя. Для изменения данных в сводной таблице, необходимо внести изменения в источник данных, а затем обновить сводную таблицу. Для обновления данных можно воспользоваться командами **Обновить** или **Обновить все** из списка **Обновить**, который расположен в группе **Данные** на контекстной вкладке **Параметры** режима **Работа со сводными таблицами**.

В завершении общего обзора, связанного со сводными таблицами, перечислим основные действия, которые можно выполнять со сводными таблицами:

- изменение названия полей (это не влечет изменений в полях исходных данных);
- изменения в перестановках полей для фильтров, столбцов и строк через перетаскивание на рабочем листе;
- группировка элементов полей по различным уровням иерархии путем объединения (выделение данных сводной таблицы осуществляется, например, с помощью мыши) в группы (группы можно переименовывать по желанию);
- скрывать и показывать детали в группе; элементы самого высокого уровня группировки (обобщающие элементы) располагаются по верхней или по крайней левой границе сводной таблицы;
- построение диаграмм на основе сводных таблиц;
- сортировка элементов в сводной таблице;
- размещение страниц сводной таблицы на различных рабочих листах (команда **Отобразить страницы фильтра отчета**, которую можно выбрать из списка **Параметры** в группе **Параметры** контекстной вкладки **Параметры** режима **Работа со сводными таблицами**);
- управление общими и промежуточными итогами;
- использование различных итоговых функций для анализа данных и дополнительных вычислений;
- вставка вычисляемого поля в сводную таблицу;
- использование стилей для форматирования сводной таблицы.

Для выполнения перечисленных действий и некоторых других изучите подробнее команды контекстных вкладок **Параметры** и **Конструктор** режима **Работа со сводными таблицами**, а также возможности контекстного меню для элементов сводной таблицы.

СОВЕТ

Для удаления сводной таблицы установите в нее указатель ячейки, затем воспользуйтесь командой **Всю сводную таблицу**, выбрав ее из списка **Выделить**, который расположен в группе **Действия** на вкладке **Параметры** режима **Работа со сводными таблицами**. Далее перейдите на вкладку **Главная** ленты и в группе **Редактирование** выберите из списка **Очистить** команду **Очистить все**.

Объекты, связанные со сводной таблицей

Со сводной таблицей связан ряд объектов, которые перечислены в табл. 8.7.

Все эти объекты являются членами соответствующих семейств, а именно `PivotTables`, `PivotTables`, `PivotFields`, `PivotFormulas`, `PivotItems` и `PivotItemList`.

Таблица 8.7. *Объекты, связанные со сводной таблицей*

Объект	Описание
PivotTable	Сводная таблица
PivotCache	Кэш-память, отведенная под сводную таблицу. Этот объект возвращается методом PivotCache объекта PivotTable
PivotCell	Ячейка сводной таблицы. Данный объект можно получить при помощи свойства PivotCell объекта Range
PivotField	Поле сводной таблицы. Этот объект возвращается методом PivotFields объекта PivotTable
PivotFormula	Формула, по которой подводится итог в сводной таблице. Этот объект возвращается методом PivotFormulas объекта PivotTable
PivotItem	Элемент поля сводной таблицы. Этот объект возвращается методом PivotItems объекта PivotTable
PivotLayout	Данные о размещении полей и осей сводной таблицы и сводной диаграммы. Данный объект можно получить при помощи свойства PivotLayout объекта Chart

Объект *PivotTable*

Объект PivotTable инкапсулирует в себе данные о сводной таблице. Этот объект является членом семейства PivotTables. В табл. 8.8 приведены методы объекта PivotTable, а в табл. 8.9 — наиболее часто используемые свойства этого объекта.

Таблица 8.8. *Методы объекта* PivotTable

Метод	Описание
AddDataField	Добавляет поля с данными
AddFields	Добавляет строки, столбцы и страницы в сводную таблицу
CalculatedFields	Возвращает семейство CalculatedFields всех вычисляемых полей сводной таблицы
Format	Задает формат отчета сводной таблицы
GetData	Возвращает данные из указанной ячейки сводной таблицы
GetPivotData	Возвращает диапазон с информацией о сводной таблице
ListFormulas	Создает список формул на отдельном листе
PivotCache	Возвращает объект PivotCache
PivotFields	Возвращает семейство PivotFields
PivotSelect	Выбирает часть сводной таблицы
PivotTableWizard	Конструирует сводную таблицу по данной таблице
RefreshTable	Обновляет сводную таблицу. Для перерасчета сводной таблицы вручную надо ее выделить и выбрать команду **Данные | Обновить данные**
ShowPages	Устанавливает содержимое области **Страница**
Update	Обновляет связи в сводной таблице

Таблица 8.9. *Свойства объекта* `PivotTable`

Свойство	Описание
`ColumnFields`, `RowFields`, `DataFields`, `PageFields`	Возвращают объект (либо единичное поле, либо семейство полей), который является столбцом (строкой, данными и страницей) сводной таблицы
`VisibleFields`, `HiddenFields`	Возвращают объект, являющийся либо единичным полем, либо семейством полей, который в данный момент отображается (скрыт) в сводной таблице

Объект *PivotCache*

Объект `PivotCache` представляет собой кэш-память, выделенную под конкретную сводную таблицу. Этот объект является членом семейства `PivotCaches` и возвращается методом `PivotCache` объекта `PivotTable`.

Основным методом семейства `PivotCaches` является метод `Add`. Он имеет следующий синтаксис:

`Add(`*`SourceType`*`, `*`SourceData`*`)`

- *`SourceType`* — обязательный параметр, задающий тип данных, на основе которых строится сводная таблица. Допустимыми значениями являются следующие константы `XlPivotTableSourceType`: `xlConsolidation` (консолидация нескольких диапазонов рабочих листов), `xlDatabase` (список или база данных MS Excel), `xlExternal` (внешняя база данных), `xlPivotTable` (сводная таблица).
- *`SourceData`* — необязательный параметр, определяющий вид источника данных в зависимости от значения параметра *`SourceType`*. Этот параметр является обязательным, если значение параметра *`SourceType`* отлично от `xlExternal`.

Объект `PivotCache` имеет ряд методов, перечисленных в табл. 8.10.

Таблица 8.10. *Методы объекта* `PivotCache`

Метод	Описание
`CreatePivotTable`	Создает объект `PivotTable`. `CreatePivotTable(`*`TableDestination`*`, `*`TableName`*`, `*`ReadData`*`)` • *`TableDestination`* — необязательный параметр, дающий ссылку на ячейку, в которой будет располагаться верхний левый угол сводной таблицы. Если сводная таблица создается на новом рабочем листе, то значение этого параметра должно быть равно пустой строке; • *`TableName`* — необязательный параметр, задающий имя сводной таблицы; • *`ReadData`* — необязательный параметр, принимающий логические значения и определяющий, надо ли сводную таблицу строить по всей внешней базе данных
`MakeConnection`	Устанавливает соединения с кэш-памятью

Таблица 8.10 *(окончание)*

Метод	Описание
`Refresh`	Обновляет кэш-память
`ResetTimer`	Переустанавливает таймер обновления кэш-памяти
`SaveAsODC`	Запись кэш-памяти в файл ODC (Microsoft Office Data Connection)

Объект *PivotField*

Объект `PivotField` представляет собой поле сводной таблицы. Этот объект является членом семейства `PivotFields` и возвращается методом `PivotField` объекта `PivotTable`. В табл. 8.11 перечислены методы, а в табл. 8.12 — основные свойства объекта `PivotField`.

Таблица 8.11. *Методы объекта* `PivotField`

Метод	Описание
`AddPageItem`	Добавляет элемент в поле
`AutoShow`	Задает правило автоматического отображения элементов полей
`AutoSort`	Задает правило автоматической сортировки полей
`CalculatedItems`	Возвращает семейство `CalculatedItems`
`Delete`	Удаляет поле
`PivotItems`	Возвращает семейство `PivotItems` всех элементов поля

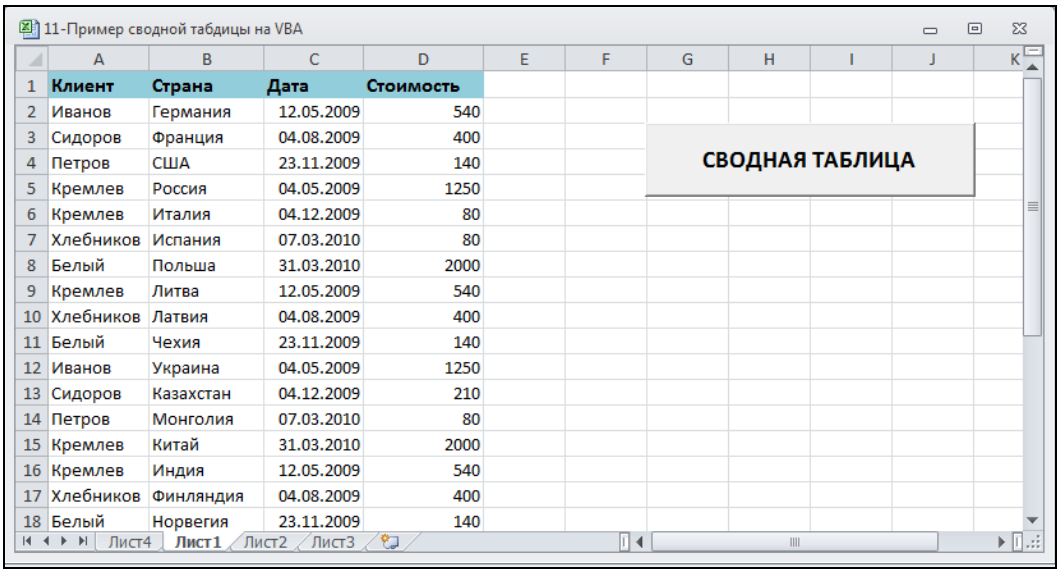
Таблица 8.12. *Основные свойства объекта* `PivotField`

Свойство	Описание
`Orientation`	Возвращает местоположения поля в сводной таблице. Допустимые значения: `xlColumnField`, `xlDataField`, `xlHidden`, `xlPageField` и `xlRowField`
`Name`	Имя поля
`Function`	Функция, по которой в поле подводится итог. Допустимые значения: `xlAverage`, `xlCount`, `xlCountNums`, `xlMax`, `xlMin`, `xlProduct`, `xlStDev`, `xlStDevP`, `xlSum`, `xlVar` и `xlVarP`
`Position`	Возвращает позицию поля (первое, второе и т. д.) среди полей того же местоположения

Пример построения сводной таблицы средствами VBA

Приведем простой пример построения сводной таблицы для списка данных со следующими полями: `Клиент`, `Страна`, `Дата`, `Стоимость`. Расположите на рабочем листе кнопку, нажатие которой вызывает процедуру `CreateStylePivotTable()`

(рис. 8.36). В стандартном модуле поместите код из листинга 8.3. Проверьте, какая сводная таблица получилась у вас на новом листе (рис. 8.37, см. также файл *11-Пример сводной таблицы на VBA.xlsm* на компакт-диске).



	A	B	C	D
1	**Клиент**	**Страна**	**Дата**	**Стоимость**
2	Иванов	Германия	12.05.2009	540
3	Сидоров	Франция	04.08.2009	400
4	Петров	США	23.11.2009	140
5	Кремлев	Россия	04.05.2009	1250
6	Кремлев	Италия	04.12.2009	80
7	Хлебников	Испания	07.03.2010	80
8	Белый	Польша	31.03.2010	2000
9	Кремлев	Литва	12.05.2009	540
10	Хлебников	Латвия	04.08.2009	400
11	Белый	Чехия	23.11.2009	140
12	Иванов	Украина	04.05.2009	1250
13	Сидоров	Казахстан	04.12.2009	210
14	Петров	Монголия	07.03.2010	80
15	Кремлев	Китай	31.03.2010	2000
16	Кремлев	Индия	12.05.2009	540
17	Хлебников	Финляндия	04.08.2009	400
18	Белый	Норвегия	23.11.2009	140

Рис. 8.36. Пример списка для построения сводной таблицы

Листинг 8.3. Создание форматированной сводной таблицы. Стандартный модуль

```
Sub CreateStylePivotTable()
    Dim DCache As PivotCache
    Dim Sales As PivotTable

'   Создаем кэш
    Set DCache = ActiveWorkbook.PivotCaches.Create( _
        SourceType:=xlDatabase, SourceData:=Range("A1").CurrentRegion)

'   Добавляем новый лист в рабочую книгу
    Worksheets.Add

'   Создаем сводную таблицу
    Set Sales = ActiveSheet.PivotTables.Add( _
        PivotCache:= DCache, TableDestination:=Range("A3"))

'   Определяем макет сводной таблицы
    With Sales
        .PivotFields("Страна").Orientation = xlPageField
        .PivotFields("Дата").Orientation = xlColumnField
        .PivotFields("Клиент").Orientation = xlRowField
```

```
        .PivotFields("Стоимость").Orientation = xlDataField
        ' Убираем заголовки полей
        .DisplayFieldCaptions = False
        ' Форматируем сводную таблицу
        .TableStyle2 = "PivotStyleDark5"
    End With
End Sub
```

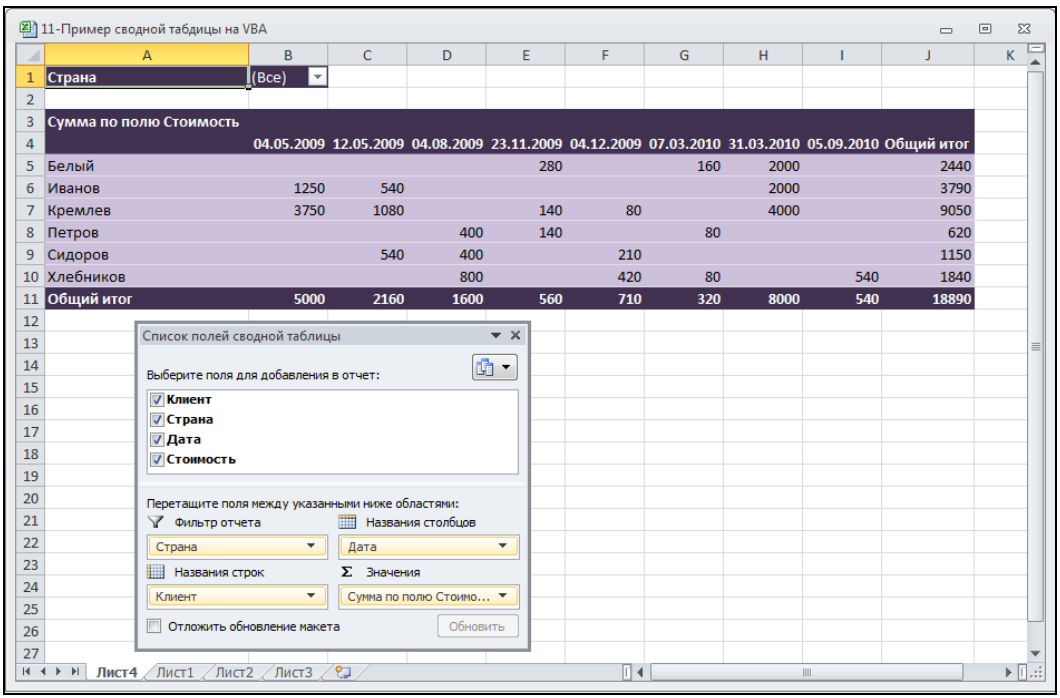


Рис. 8.37. Сводная таблица, построенная с использованием VBA

Наши итоги

В этой главе мы познакомились со многими средствами Microsoft Excel, которые позволяют производить реорганизацию и анализ данных, находящихся в однотипных списках. Все это, несомненно, позволит вам за считанные минуты узнать итоговые результаты. Произвести статистические операции над данными, а также создать требуемую иерархию в своих проектах. Таким образом, вы научились:

- создавать простые и промежуточные итоги;
- обобщать однородные данные с использованием консолидации;
- создавать иерархические структуры, согласуя их как по строкам, так и по столбцам;
- использовать сценарии;
- строить сводные таблицы;
- применять VBA для разработки конкретных приложений с использованием описанных выше средств Microsoft Excel.

Глава 9

Используем поиск решения и подбор параметра

Многие задачи производства, проектирования, прогнозирования сводятся к широкому классу задач оптимизации, для решения которых применяются математические методы. Типовыми задачами такого плана являются, например, следующие:

- ассортимент продукции — максимизация выпуска товаров при ограничениях на сырье для производства этих товаров;
- штатное расписание — составление штатного расписания для достижения наилучших результатов при наименьших расходах;
- планирование перевозок — минимизация затрат на транспортировку товаров;
- составление смеси — достижение заданного качества смеси при наименьших расходах;
- размер емкости — определение размеров некоторой емкости с учетом стоимости материала для достижения максимального объема;
- случайные величины — разнообразные задачи, в которые входят случайные величины;
- разнообразные задачи оптимального распределения ресурсов и оптимального проектирования и т. д.

В MS Excel имеется достаточно мощный инструмент **Поиск решения**, который позволяет легко решать указанные оптимизационные задачи линейного и нелинейного программирования. Кроме того, зачастую при решении задач достаточно воспользоваться другим средством — **Подбор параметра**, которое позволяет найти точное значение некоторого параметра при известном итоговом результате и заданной формуле вычислений. И **Поиск решения**, и **Подбор параметра** также относятся к средствам анализа данных в MS Excel. Отметим, что, как частный случай, **Поиск решения** позволяет решать и те задачи, которые решаются при помощи подбора параметра.

В предлагаемой главе мы рассмотрим указанные инструменты и их использование для решения различных задач.

Примечание

Файлы рассмотренных в данной главе примеров можно найти в папке Glava_9 на компакт-диске.

Поиск решения: как это работает?

Итак, рассмотрим на различных примерах использование надстройки **Поиск решения**. Обычно **Поиск решения** можно найти на вкладке **Данные** ленты в группе **Анализ**. Если данная команда отсутствует в указанном месте, выполните следующие действия: перейдите на вкладку **Файл** ленты и выберите команду **Параметры**. В открывшемся окне **Параметры Excel** выберите слева категорию **Надстройки**, а справа в группе **Управление** выберите из списка **Настройки Excel** и нажмите кнопку **Перейти**. Далее в окне **Надстройки** (рис. 9.1) в списке **Доступные надстройки** установите флажок перед надстройкой **Поиск решения** и нажмите кнопку **OK**.

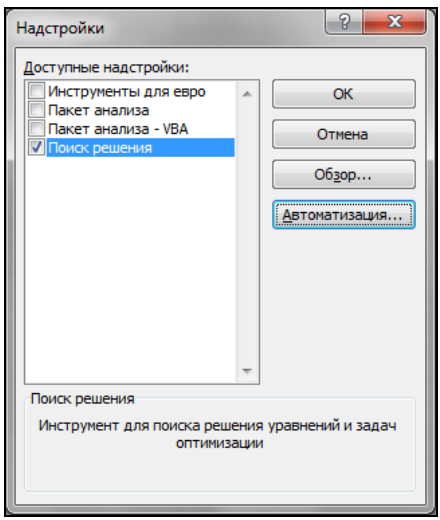


Рис. 9.1. Окно **Надстройки**

Постановка задачи оптимизации в общем случае

Задачу оптимизации в общем виде можно сформулировать так, как представлено в табл. 9.1.

Таблица 9.1. *Постановка задачи оптимизации в общем случае*

№ п/п	Название	Математическая запись	Описание
1.	*Целевая функция* (критерий оптимизации)	$F = f\ x_j \ \rightarrow \max(\min, \text{const})$, $j = \overline{1,n}$	Показывает, в каком смысле решение должно быть оптимальным, т. е. наилучшим. Возможны три вида целевой функции: максимизация, минимизация, назначение заданного значения

Таблица 9.1 *(окончание)*

№ п/п	Название	Математическая запись	Описание
2.	*Ограничения*	$g_i \ x_j \ \le \ =; \ge \ b_i$, $i = \overline{1, m}, \ j = \overline{1, n}$. $x_j = \overline{1, k} \le n$ — целые, для задач целочисленного программирования. $0 \le x_j \le 1, \ \ j = \overline{1, k}$ — для задач с булевыми переменными	Устанавливают зависимости между переменными. Могут быть односторонними и двусторонними. При решении задач двустороннее ограничение записывается в виде двух односторонних
3.	*Граничные условия*	$d_j \le x_j \le D_j, \ \ j = \overline{1, n}$	Показывают, в каких пределах могут быть значения искомых переменных в оптимальном решении

Решение задач (1)—(3), удовлетворяющее всем ограничениям и граничным условиям, называется *допустимым*. Важная характеристика задачи оптимизации — ее размерность, которая определяется числом переменных n и числом ограничений m. При $n < m$ — задачи решения не имеют. *Необходимым требованием задач оптимизации* является условие $n > m$. Систему уравнений, для которых $n = m$ рассматривают, как задачу оптимизации, имеющую одно допустимое решение (ее можно решать как обычную задачу оптимизации, назначая в качестве целевой функции любую переменную).

Итак, задача имеет оптимальное решение, если она удовлетворяет двум требованиям:

- имеет более одного решения, т. е. существуют допустимые решения;
- имеется критерий, показывающий, в каком смысле принимаемое решение должно быть оптимальным, т. е. наилучшим из допустимых.

Надстройка *Поиск решения*

Надстройка **Поиск решения** запускается, как указывалось ранее, соответствующей командой из группы **Анализ** на вкладке **Данные** ленты.

Вид открывшегося окна **Параметры поиска решения** и его опции приведены на рис. 9.2, 9.3 и в табл. 9.2.

При нажатии кнопки **Параметры** (в окне **Параметры поиска решения**) открывается окно **Параметры** (рис. 9.4), характеристика которого приведена в табл. 9.3.

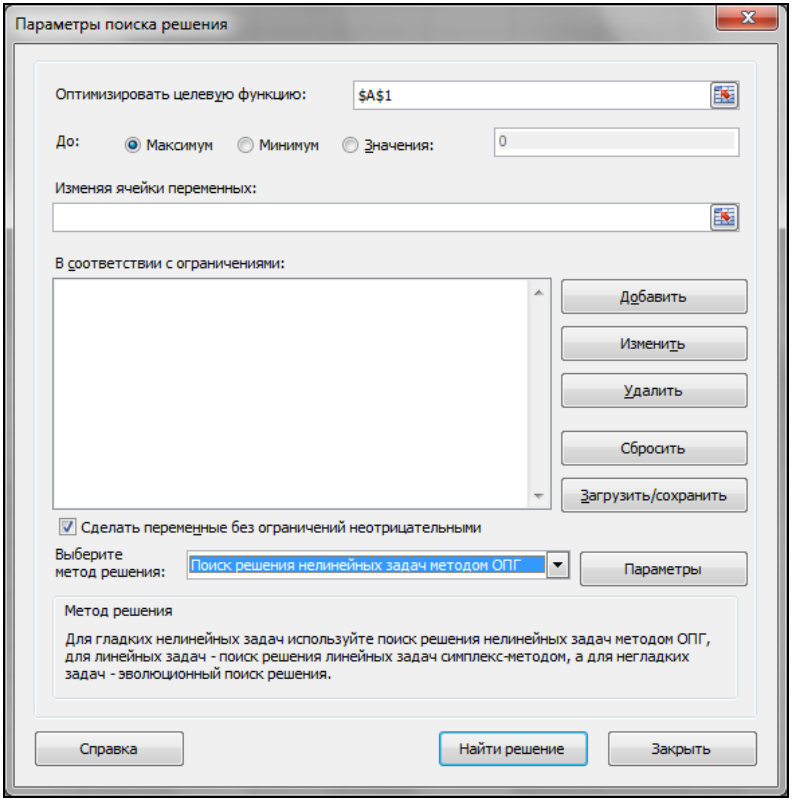


Рис. 9.2. Окно **Параметры поиска решения**

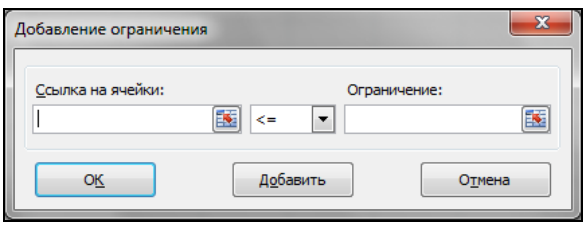


Рис. 9.3. Окно **Добавление ограничения**

Таблица 9.2. *Опции окна* ***Параметры поиска решения***

Название	**Описание**
Оптимизировать целевую функцию	Указывается ячейка, содержащая целевую функцию (критерий оптимизации) рассматриваемой задачи
До	Следует выбрать из трех переключателей (**Максимум**, **Минимум**, **Значения**) тот, который определяет тип взаимосвязи между решением и целевой ячейкой

Таблица 9.2 *(окончание)*

Название	Описание
Изменяя ячейки переменных	Указываются ячейки, которые должны изменяться в процессе поиска решения задачи (т. е. ячейки, которые являются переменными задачи)
В соответствии с ограничениями	Отображаются ограничения, налагаемые на переменные задачи. Допускаются ограничения: в виде равенств, неравенств, требование целочисленности переменных, принятия лишь двух значений: 0 или 1. Для добавления, изменения или удаления ограничения используются соответственно кнопки **Добавить**, **Изменить** или **Удалить**. Ограничения добавляются по одному за один раз и отображаются в окне **Добавление ограничения** (рис. 9.3), вызываемом нажатием кнопки **Добавить**. В поле **Ссылка на ячейку** вводится левая часть ограничений, в поле **Ограничение** — правая часть. Раскрывающийся список позволяет задать тип соотношения между левой и правой частями ограничения. А именно **>=**, **<=**, **=**, **цел**, **бин** и **разн**. Соотношение **цел** подразумевает, что выражение может принимать только целочисленные значения, **бин** — только два значения: 0 и 1, а **разн** — все значения должны быть различны. При нажатии кнопки **Добавить** окна **Добавление ограничения** вводится вторая группа ограничений, налагаемых на переменные, а затем и последующие ограничения. Нажатие кнопки **ОК** завершает ввод ограничений. На экране опять отобразится окно **Параметры поиска решения**, но теперь уже заполненное
Сделать переменные без ограничений неотрицательными	Данные параметр устанавливает требование неотрицательности переменных задачи
Выберите метод решения	Позволяет выбрать метод (алгоритм оптимизации), который будет использоваться надстройкой Поиск решения для нахождения оптимального решения задачи. Из расположенного списка можно выбрать следующие методы: **Поиск решения нелинейных задач методом ОПГ**, **Поиск решения линейных задач симплекс-методом** и **Эволюционный поиск решения**. Внизу, под списком, приводится замечание по использованию указанных методов.
Сбросить	Восстанавливает изначальные параметры **Поиска решения**, т. е. происходит сброс всех настроек окна **Параметры поиска решения**
Загрузить/сохранить	Сохранение (загрузка) различных данных для **Поиска решения** осуществляется с помощью кнопки окна **Параметры поиска решения**
Параметры	Позволяют изменять условия и варианты поиска решений исследуемой задачи. Значения и состояния элементов управления, используемые по умолчанию, подходят для решения большинства задач
Найти решение	Запускает поиск решения при установленных параметрах. По завершении работы на экране отобразится окно **Результаты поиска решения**

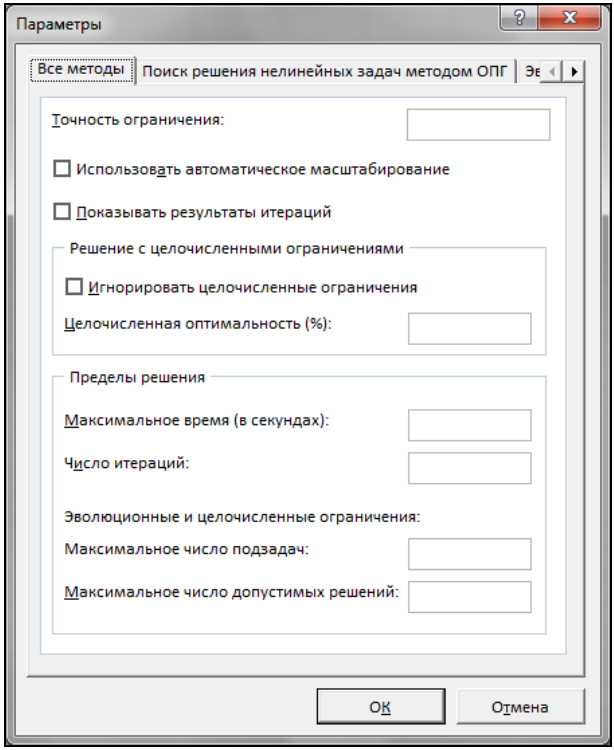


Рис. 9.4. Окно **Параметры**, открытое на вкладке **Все методы**

Таблица 9.3. *Опции окна* ***Параметры***

Название	**Описание**
Вкладка **Все методы**	
Точность ограничения	Устанавливается требуемая точность, с которой ищется решение. Данное ограничение считается выполненным, если разность между значением в ячейке и значением ограничения не превышает указанное число. Чем меньше число, которое вы указали, тем выше точность
Использовать автоматическое масштабирование	Предназначен для включения автоматической нормализации входных и выходных значений, качественно различающихся по величине. Например, при максимизации прибыли в процентах по отношению к вложениям, исчисляемым в миллионах рублей
Показывать результаты итераций	Для приостановки поиска решений и просмотра отдельных итераций с целью получения дополнительной информации
Группа **Решение с целочисленными ограничениями**	Данная группа включает следующие параметры: **Игнорировать целочисленные ограничения** и **Целочисленная оптимальность (%)**. Первый флажок устанавливается для задач с требованием целочисленности, бинарности и различных значений. Поиск решения определит наилучшие значения с указанной точностью

Таблица 9.3 *(продолжение)*

Название	Описание
Вкладка **Все методы**	
Группа **Решение с целочисленными ограничениями**	Значение по умолчанию для параметра **Целочисленная оптимальность (%)** составляет 1%. Установите значение 0% для получения наиболее точного решения задачи с целочисленными или бинарными ограничениями. *Замечание*. Для задач с целочисленными ограничениями на переменные рекомендуется после нахождения решения с величинами данных параметров, заданными по умолчанию, повторить вычисления с большей точностью и меньшим допустимым отклонением и сравнить с первоначальным решением. Параметр **Целочисленная оптимальность (%)** называется также относительной погрешностью
Группа **Пределы решения**	**Максимальное время (в секундах)** ограничивает время, отпускаемое на поиск решения задачи
	Число итераций ограничивает число промежуточных вычислений
	Эволюционные и целочисленные ограничения применяется только для задач, которые включают требования целочисленности переменных или используют эволюционный метод решения. Здесь два параметра: • **Максимальное число подзадач** — устанавливается максимальное число подзадач, которые вы хотите разрешить; • **Максимальное число допустимых решений** — устанавливается максимальное количество возможных решений, которые вы хотите разрешить; если возникают проблемы с целыми ограничения, то это максимальное число целых допустимых решений. *Замечание.* Если процесс решения достигает максимального времени, числа итераций, максимального количества подзадач или максимально возможного решения, то поиск решения находит оптимальное решение и отображает его в диалоговом окне **Результаты поиска решения** (см. рис. 9.5)
Вкладка **Поиск решения нелинейных задач методом ОПГ**	
Сходимость	Устанавливается значение допустимого отклонения для оптимального решения
Группа **Производные**	Служит для выбора метода численного дифференцирования. При установке переключателя в положение **Центральные** при решении задачи используются более точные центральные разности, однако время вычислений значительно возрастает
Группа **Несколько начальных точек**	Данная группа используется для последовательного поиска оптимального решений. Установите флажок **Использовать несколько начальных точек** для того, чтобы процесс поиска решения обрабатывал несколько стартовых вариантов конкретной задачи
	Размер совокупности — устанавливается количество вариантов решения, причем, минимальный размер совокупности составляет 10, а максимальный — 200

Таблица 9.3 *(окончание)*

Название	Описание
Вкладка **Поиск решения нелинейных задач методом ОПГ**	
Группа **Несколько начальных точек**	**Случайное начальное значение** — выберите начальное положительное значение, которое будет использоваться генератором случайных чисел в качестве стартового значения при решении задачи оптимизации. Это ведет к поучению различных итоговых значений. Если это поле оставить пустым, то генератор случайных чисел будет при каждом новом шаге решения задачи генерировать произвольно число
	Обязательные границы для переменных — задается верхняя и нижняя границы для переменных при поиске оптимального решения
Вкладка **Эволюционный поиск решения**	
Сходимость	Устанавливается значение допустимого отклонения для оптимального решения
Скорость изменения	Число между 0 и 1, указывающее относительную частоту изменения в совокупности решений
Размер совокупности	Устанавливается количество вариантов решения, причем минимальный размер совокупности составляет 10, а максимальный — 200
Случайное начальное значение	Выберите начальное положительное значение, которое будет использоваться генератором случайных чисел в качестве стартового значения при решении задачи оптимизации
Максимальное время без улучшения	Вводится максимальное число секунд, допускающее поиск решение без существенного улучшения с использованием метода эволюционного поиска
Обязательные границы для переменных	При установке данного флажка эволюционный метод будет работать лишь тогда, когда вы определите верхнюю и нижнюю границы для переменных в списке ограничений

Рекомендации по решению задач оптимизации с помощью надстройки *Поиск решения*

Построение математической модели задачи

Работа по решению некоторой оптимизационной задачи всегда начинается с построения математической модели, для чего следует ответить на вопросы:

- ❒ Каковы переменные модели (для определения каких величин строится модель)?
- ❒ В чем состоит цель, для достижения которой из множества всех допустимых значений переменных выбираются оптимальные?
- ❒ Каким ограничениям должны удовлетворять неизвестные?

Стоит также учесть, что при конструировании модели формулировка ограничений является самой ответственной частью конструкции. В некоторых случаях огра-

ничения очевидны, например, ограничение на количество сырья. Другие же ограничения могут быть менее очевидны и могут быть указаны неверно. Например:

- в модели с несколькими периодами времени величина материального ресурса на начало следующего периода должна равняться величине этого ресурса на конец предыдущего периода;
- в модели поставок величина запаса на начало периода плюс количество полученного должна равняться величине запаса на конец периода плюс количество отправленного;
- многие величины в модели по своему физическому смыслу не могут быть отрицательными, например, количество полученных единиц товара.

Таким образом, на этом этапе делаются выводы об исходных данных (детерминированные или случайные), искомых переменных (непрерывные или дискретные), о пределах, в которых могут находиться значения искомых величин, о зависимостях между переменными (линейные или нелинейные), о критериях, по которым необходимо находить оптимальное решение. Сюда же входит преодоление несовместности, а также неограниченности целевой функции: при максимизации целевой функции область допустимых решений должна быть ограничена сверху, при минимизации — снизу.

Подготовка рабочего листа MS Excel для решения задачи оптимизации

Рекомендуется: корректно разместить все исходные данные на рабочем листе, грамотно ввести необходимые формулы для целевой функции и для других зависимостей, выбрать место для значения переменных.

Решение задачи с помощью надстройки *Поиск решения*

Следует правильно ввести все ограничения, переменные, целевую функцию и другие значения в окно **Поиск решения.**

Большую часть задач оптимизации представляют задачи линейного программирования, т. е. такие, у которых критерий оптимизации и ограничения — линейные функции. В этом случае для решения задачи надо установить опцию **Линейная модель** в окне **Параметры поиска решения**. Это обеспечивает применение симплекс-метода. В противном случае даже для решения линейной задачи будут использоваться более общие методы (т. е. более медленные).

Поиск решения может работать также и с нелинейными зависимостями и ограничениями. Это, как правило, задачи нелинейного программирования или, например, решение системы нелинейных уравнений. Для успешной работы **Поиска решения** следует стремиться к тому, чтобы зависимости были гладкими или, по крайней мере, непрерывными. Наиболее часто разрывные зависимости возникают при использовании функции `ЕСЛИ()`, среди аргументов которой имеются переменные величины модели. Проблемы могут возникнуть также и при использовании в модели функций типа `ABS()`, `ОКРУГЛ()` и т. д.

В случае нелинейных зависимостей следует:

- ввести предварительно предположительные значения искомых переменных (иногда легко получить графическое представление решения и сделать приблизительные выводы о решении);

- в окне **Параметры поиска решения** отключить (если включена) опцию **Линейная модель**.

В случае задач целочисленного программирования не следует забывать также о требованиях условий целочисленности и булевости.

Анализ решения задачи оптимизации

При необходимости проводится анализ решения. Часто добавляют также представление решения в виде графиков или диаграмм. Можно получить и **Отчет о поиске решения**. Отчеты бывают трех типов*: Результаты*, *Устойчивость*, *Пределы*. Тип отчета выбирается по окончании поиска решения: в окне **Результаты поиска решения** (рис. 9.5) в списке **Отчеты** (можно выбрать сразу два или три типа). Отчет типа **Результаты** содержит окончательные значения параметров задачи целевой функции и ограничений. Отчет типа **Устойчивость** показывает результаты малых изменений параметров **Поиска решения**. Отчет типа **Пределы** демонстрирует изменения решения при поочередной максимизации и минимизации каждой переменной при неизменных других переменных.

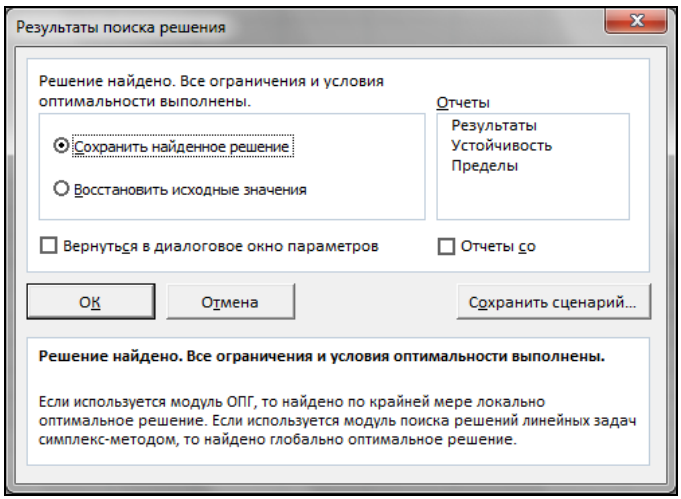


Рис. 9.5. Окно **Результаты поиска решения**

Решаем задачу линейного программирования

Линейное программирование — это раздел математического программирования, посвященный нахождению экстремума линейных функций нескольких переменных при дополнительных линейных ограничениях, которые налагаются на переменные. Методы, с помощью которых решаются задачи, подразделяются на универсальные (например, симплексный метод) и специальные. С помощью универсальных методов решаются любые задачи линейного программирования. Особенностью задач линейного программирования является то, что экстремум целевой функции достигается на границе области допустимых решений.

Планирование производства материалов

Рассмотрим пример, связанный с производством материалов (см. также файл *1-Задачи линейной оптимизации.xlsx* на компакт-диске). Пусть фирма выпускает два типа строительных материалов: *A* и *B*. Продукция обоих видов поступает в продажу. Для производства материалов используются два исходных продукта: I и II. Максимально возможные суточные запасы этих продуктов составляют 7 и 9 тонн соответственно. Расходы продуктов I и II на 1 тонну соответствующих материалов приведены в табл. 9.4.

Таблица 9.4. *Расход продуктов*

Исходный продукт	Расход исходных продуктов (на одну тонну материалов), т		Максимально возможный запас, т
	материал *A*	материал *B*	
I	3	2	7
II	2	3	9

Изучение рынка сбыта показало, что суточный спрос на материал *B* никогда не превышает спроса на материал *A* более чем на 1 т. Кроме того, спрос на материал *A* никогда не превышает 3 т в сутки. Оптовые цены одной тонны материалов равны: 4000 у. е. для *B* и 3000 у. е. для *A*. Какое количество материала каждого вида должна производить фабрика, чтобы доход от реализации был максимальным?

Итак, для решения поставленной задачи вам необходимо выполнить следующие шаги.

1. Формулировка математической модели задачи:
   - переменные для решения задачи. x_1 — суточный объем производства материала *A*, x_2 — суточный объем производства материала *B*;
   - определение функции цели (критерия оптимизации). Суммарная суточная прибыль от производства x_1 материала *A* и x_2 материала *B* равна:

     $$F = 4000x_2 + 3000x_1,$$

     поэтому цель фабрики — среди всех допустимых значений x_2 и x_1 найти такие, которые максимизируют суммарную прибыль от производства материалов *F*:

     $$F = 4000x_2 + 3000x_1 \to \max;$$

   - ограничения на переменные:
     ◊ объем производства красок не может быть отрицательным, т. е.

     $$x_2 \geq 0,\ \ x_1 \geq 0;$$

     ◊ расход исходного продукта для производства обоих видов материалов не может превосходить максимально возможного запаса данного исходного продукта, т. е.

     $$2x_2 + 3x_1 \leq 7,$$
     $$3x_2 + 2x_1 \leq 9;$$

- ограничения на величину спроса на материалы:

$$x_1 - x_2 \le 1,$$
$$x_1 \le 3.$$

Таким образом, получаем математическую модель задачи: найти максимум следующей функции:

$$F = 4000x_2 + 3000x_1 \to \max$$

при ограничениях вида:

$$2x_2 + 3x_1 \le 7,$$
$$3x_2 + 2x_1 \le 9,$$
$$x_1 - x_2 \le 1,$$
$$x_1 \le 3,$$
$$x_1 \ge 0,\ x_2 \ge 0.$$

2. Подготовка листа рабочей книги MS Excel для вычислений. На рабочий лист вводим необходимый текст, данные и формулы в соответствии с рис. 9.6. Переменные задачи x_1 и x_2 находятся соответственно в ячейках **C3** и **C4**. Целевая функция находится в ячейке **C6** и содержит формулу:

```
=4000*C4+3000*C3
```

Ограничения на данную задачу учтены в ячейках **C8:D11** (рис. 9.6).

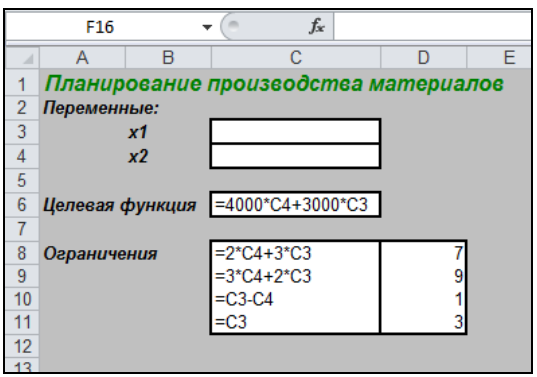


Рис. 9.6. Рабочий лист MS Excel
для решения задачи планирования производства материалов

3. Работа с надстройкой **Поиск решения**. Воспользовавшись командой **Поиск решения**, находящейся на вкладке **Данные** ленты в группе **Анализ**, вводим необходимые данные для рассматриваемой задачи.
   - Установка данных задачи приведена на рис. 9.7.
   - Для добавления ограничений в поле **В соответствии с ограничениями** нажмите кнопку **Добавить** в окне **Параметры поиска решения**. В открывшемся окне **Добавление ограничения** (см. рис. 9.3) введите первую группу ограничений: в поле **Ссылка на ячейку** введите левую часть ограничений — **C3:G4**, в поле **Ограничение** — правую часть, в нашем случае — 0. Раскры-

вающийся список позволяет задать тип соотношения между левой и правой частями ограничения. В нашем случае выберите соотношение >=. Таким образом, требование неотрицательности переменных задано. Нажмите кнопку **Добавить** и с помощью окна **Добавление ограничения** введите вторую группу ограничений, налагаемых на переменные: `C8:C11 <= D8:D11`. Нажмите кнопку **OK** для завершения ввода ограничений. На экране опять отобразится окно **Поиск решения**, но теперь уже заполненное.

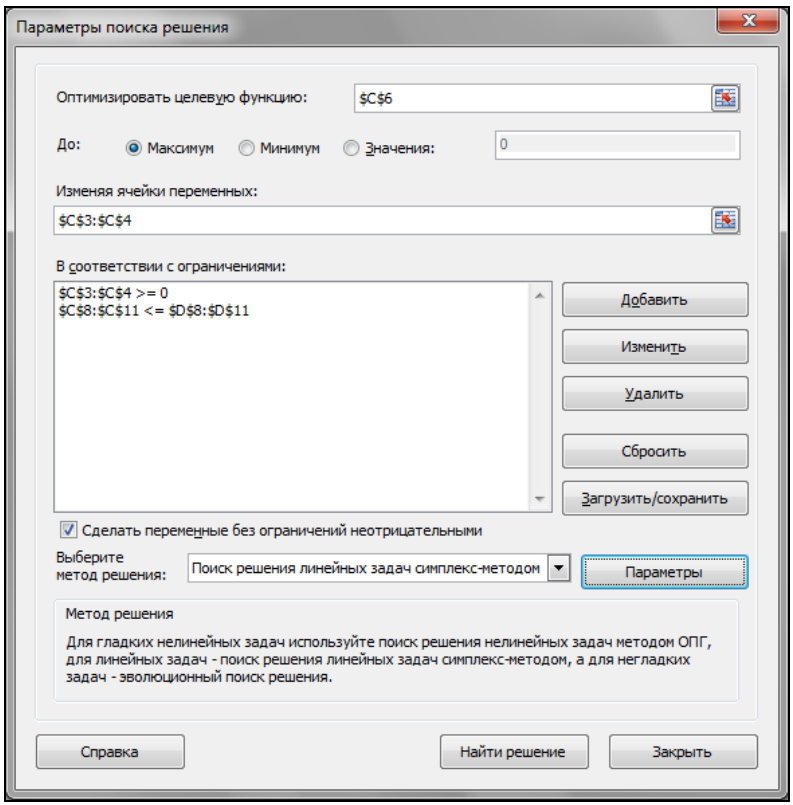
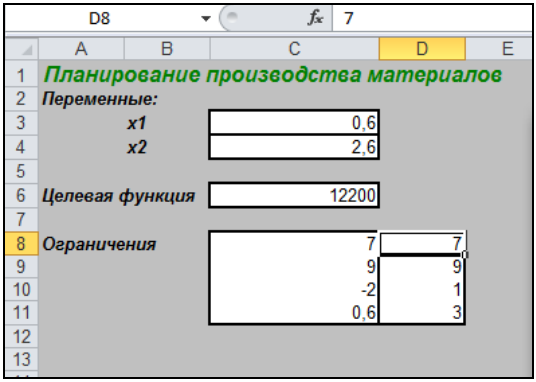


Рис. 9.7. Установка необходимых параметров задачи планирования материалов в окне **Параметры поиска решения**

- Нажмите кнопку **Найти решение** для запуска на выполнение надстройки **Поиск решения**. В открывшемся окне **Результаты поиска решения** (см. рис. 9.5) приведены соответствующие результаты, и есть возможность создать требуемый отчет.
- Результат работы **Поиск решения** приведен на рис. 9.8.

Отчет по результатам (рис. 9.9). Таблица **Ячейка целевой функции** выводит сведения о целевой функции; таблица **Ячейки переменных** показывает значения искомых переменных, полученных в результате решения задачи; таблица **Ограничения** отображает результаты оптимального решения для ограничений и для граничных условий. В поле **Формула** приведены зависимости, которые были введены

в окно **Поиск решения**, в поле **допуск** — величины использованного материала. Если материал используется полностью, то в поле **Состояние** указывается **привязка**, при неполном использовании материала в данном поле указывается **без привязки**. Для граничных условий приводятся аналогичные величины с той лишь разницей, что вместо величины неиспользованного продукта показана разность между значением переменной в найденном оптимальном решении и заданным для нее граничным условием.

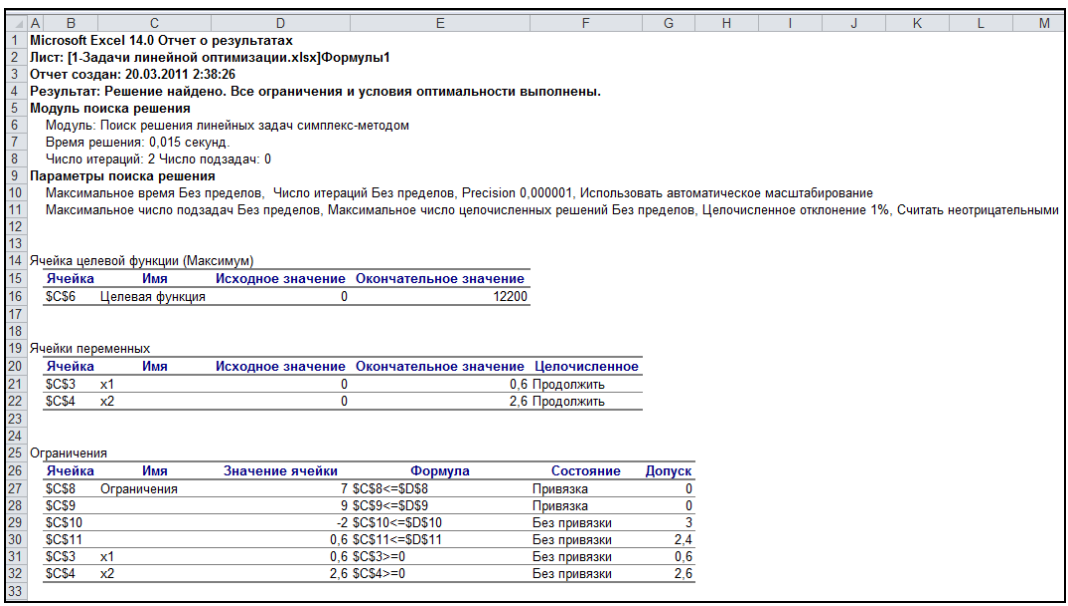


	A	B	C	D	E
1	**Планирование производства материалов**				
2	*Переменные:*				
3		*x1*	0,6		
4		*x2*	2,6		
5					
6	*Целевая функция*		12200		
7					
8	*Ограничения*		7	7	
9			9	9	
10			-2	1	
11			0,6	3	
12					
13					

Рис. 9.8. Результат расчета надстройки **Поиск решения**

Отчет по устойчивости (рис. 9.10). В таблице **Ячейки переменных** приводятся результат решения задачи. В таблице **Ограничения** выводятся значения для ограничений, при которых сохраняется оптимальный набор переменных, входящих в оптимальное решение.

Отчет по пределам (рис. 9.11). В отчете показано, в каких пределах может изменяться количество материалов, вошедших в оптимальное решение, при сохранении структуры оптимального решения: приводятся значения переменных в оптимальном решении, а также нижние и верхние пределы изменения значений переменных, здесь также указаны значения целевой функции при выпуске данного типа продукции на верхнем и нижнем пределах.



Microsoft Excel 14.0 Отчет о результатах
Лист: [1-Задачи линейной оптимизации.xlsx]Формулы1
Отчет создан: 20.03.2011 2:38:26
Результат: Решение найдено. Все ограничения и условия оптимальности выполнены.
Модуль поиска решения
Модуль: Поиск решения линейных задач симплекс-методом
Время решения: 0,015 секунд.
Число итераций: 2 Число подзадач: 0
Параметры поиска решения
Максимальное время Без пределов, Число итераций Без пределов, Precision 0,000001, Использовать автоматическое масштабирование
Максимальное число подзадач Без пределов, Максимальное число целочисленных решений Без пределов, Целочисленное отклонение 1%, Считать неотрицательными

Ячейка целевой функции (Максимум)

Ячейка	Имя	Исходное значение	Окончательное значение
$C$6	Целевая функция	0	12200

Ячейки переменных

Ячейка	Имя	Исходное значение	Окончательное значение	Целочисленное
$C$3	x1	0	0,6	Продолжить
$C$4	x2	0	2,6	Продолжить

Ограничения

Ячейка	Имя	Значение ячейки	Формула	Состояние	Допуск
$C$8	Ограничения	7	$C$8<=$D$8	Привязка	0
$C$9		9	$C$9<=$D$9	Привязка	0
$C$10		-2	$C$10<=$D$10	Без привязки	3
$C$11		0,6	$C$11<=$D$11	Без привязки	2,4
$C$3	x1	0,6	$C$3>=0	Без привязки	0,6
$C$4	x2	2,6	$C$4>=0	Без привязки	2,6

Рис. 9.9. Отчет по результатам поиска решения

Microsoft Excel 14.0 Отчет об устойчивости
Лист: [1-Задачи линейной оптимизации.xlsx]Формулы1
Отчет создан: 20.03.2011 2:38:26

Ячейки переменных

Ячейка	Имя	Окончательное Значение	Приведенн. Стоимость	Целевая функция Коэффициент	Допустимое Увеличение	Допустимое Уменьшение
$C$3	x1	0,6	0	3000	3000	333,3333333
$C$4	x2	2,6	0	4000	500	2000

Ограничения

Ячейка	Имя	Окончательное Значение	Тень Цена	Ограничение Правая сторона	Допустимое Увеличение	Допустимое Уменьшение
$C$8	Ограничения	7	200	7	3	1
$C$9		9	1200	9	1,5	3
$C$10		-2	0	1	1E+30	3
$C$11		0,6	0	3	1E+30	2,4

Рис. 9.10. Отчет по устойчивости поиска решения

Microsoft Excel 14.0 Отчет о пределах
Лист: [1-Задачи линейной оптимизации.xlsx]Формулы1
Отчет создан: 20.03.2011 2:38:27

Целевая функция

Ячейка	Имя	Значение
$C$6	Целевая функ	12200

Переменная			Нижний	Целевая функция	Верхний	Целевая функция
Ячейка	Имя	Значение	Предел	Результат	Предел	Результат
$C$3	x1	0,6	0	10400	0,6	12200
$C$4	x2	2,6	0	1800	2,6	12200

Рис. 9.11. Отчет по пределам поиска решения

Определение состава удобрений

Для получения удобрений видов 1 и 2 используются химические вещества *A*, *B*, *C* и *D*, требования к которым приведены в табл. 9.5.

Таблица 9.5. *Требования по содержанию химических веществ в удобрениях*

Вид удобрения	Требования к содержанию химических веществ
1	Не более 50% вещества *A* Не более 60% вещества *B*
2	От 40 до 70% вещества *B* Не менее 20% вещества *C* Не более 80% вещества *D*

Характеристики и запасы минералов, используемых для производства химических веществ *A*, *B*, *C* и *D*, указаны в табл. 9.6.

Таблица 9.6. *Характеристика и запасы минералов*

Минерал	Максимальный запас, т	Состав, %				Цена, у. е./т
		A	*B*	*C*	*D*	
1	1200	30	20	15	35	40
2	2500	20	30	10	40	50
3	3100	15	15	40	30	60

Цена 1 т удобрения вида 1 равна 320 у. е., а 1 т удобрения вида 2 — 350 у. е. Необходимо максимизировать прибыль от продажи удобрений видов 1 и 2.

Для решения данной задачи вам необходимо выполнить следующие шаги.

1. Математическая модель задачи. Пусть:
   - $x_{A1}, x_{B1}, x_{C1}, x_{D1}$ — количество химических веществ *A*, *B*, *C* и *D*, используемых для получения удобрения вида 1;
   - $x_{A2}, x_{B2}, x_{C2}, x_{D2}$ — количество химических веществ *A*, *B*, *C* и *D*, используемых для получения удобрения вида 2;
   - $y_i, i = \overline{1,3}$ — количество используемого *i*-го минерала.

   Математическая модель данной задачи будет иметь вид: найти максимум функции:

$$F = 320\ \ x_{A1} + x_{B1} + x_{C1} + x_{D1}\ \ + 350\ \ x_{A2} + x_{B2} + x_{C2} + x_{D2}\ \ - 40y_1 - 50y_2 - 60y_3 \to \max$$

   при следующих ограничениях:

   - на состав вида удобрения (см. табл. 9.5):

$$x_{A1} \le 0,5\ \ x_{A1} + x_{B1} + x_{C1} + x_{D1}\ ,$$
$$x_{B1} \le 0,6\ \ x_{A1} + x_{B1} + x_{C1} + x_{D1}\ ,$$
$$x_{B2} \le 0,7\ \ x_{A2} + x_{B2} + x_{C2} + x_{D2}\ ,$$
$$x_{B2} \ge 0,4\ \ x_{A2} + x_{B2} + x_{C2} + x_{D2}\ ,$$
$$x_{C2} \ge 0,2\ \ x_{A2} + x_{B2} + x_{C2} + x_{D2}\ ,$$
$$x_{D2} \le 0,8\ \ x_{A2} + x_{B2} + x_{C2} + x_{D2}\ ,$$

   - на характеристики и минералов (см. табл. 9.6):

$$x_{A1} + x_{A2} \le 0,3y_1 + 0,2y_2 + 0,15y_3,$$
$$x_{B1} + x_{B2} \le 0,2y_1 + 0,3y_2 + 0,15y_3,$$
$$x_{C1} + x_{C2} \le 0,15y_1 + 0,1y_2 + 0,4y_3,$$
$$x_{D1} + x_{D2} \le 0,35y_1 + 0,4y_2 + 0,3y_3,$$

- на диапазоны переменных:

$$x_{i1} \geq 0,\ x_{i2} \geq 0,\ i = \overline{A, D};$$
$$0 \leq y_1 \leq 1200,$$
$$0 \leq y_2 \leq 2500,$$
$$0 \leq y_3 \leq 3100.$$

2. Подготовка листа рабочей книги MS Excel. Разместим данные для решения задачи на рабочем листе в соответствии с рис. 9.12 и табл. 9.7 (см. также файл *1-Задачи линейной оптимизации.xlsx* на компакт-диске).
3. Ввод данных в окно **Параметры поиска решения** осуществляется в соответствии с рис. 9.13. Не следует забывать о заполнении необходимых опций в окне **Параметры**.
4. Результат поиска решения, т. е. решение задачи об определении состава удобрений, представлен на рис. 9.14.

	A	B	C	D	E	F	G	H
1	**Определение состава удобрений**							
2	**Переменнные**							
3			**Удобрение**					
4		**Вещество**	**A**	**B**		**Минерал**		**Запас минерала**
5			1			1		1200
6			2			2		2500
7			3			3		3100
8			4					
9								
10								
11	**Целевая функция**	=320*СУММ(C5:C8)+350*СУММ(D5:D8)-40*F5-50*F6-60*F7						
12								
13	**Ограничения**							
14	=C5-0,5*СУММ(C5:C8)							
15	=C6-0,6*СУММ(C5:C8)							
16	=D6-0,7*СУММ(D5:D8)							
17	=0,4*СУММ(D5:D8)-D6							
18	=0,22*СУММ(D5:D8)-D7							
19	=D8-0,8*СУММ(D5:D8)							
20	=СУММ(C5:D5)-0,3*$F$5-0,2*$F$6-0,15*$F$7							
21	=СУММ(C6:D6)-0,2*$F$5-0,3*$F$6-0,15*$F$7							
22	=СУММ(C7:D7)-0,15*$F$5-0,1*$F$6-0,4*$F$7							
23	=СУММ(C8:D8)-0,35*$F$5-0,4*$F$6-0,3*$F$7							
24								
25								
26								

Рис. 9.12. Лист рабочей книги для решения задачи производства удобрений

Таблица 9.7. *Формулы для расчета, используемые при решении задачи определения состава удобрений*

Описание	Ячейка	Формула
Целевая функция	**D11**	=320*СУММ(C5:C8)+350*СУММ(D5:D8)-40*F5-50*F6-60*F7
Ограничения	**B14**	=C5-0,5*СУММ(C5:C8)
	B15	=C6-0,6*СУММ(C5:C8)
	B16	=D6-0,7*СУММ(D5:D8)
	B17	=0,4*СУММ(D5:D8)-D6
	B18	=0,2*СУММ(D5:D8)-D7
	B19	=D8-0,8*СУММ(D5:D8)
	B20	=СУММ(C5:D5)-0,3*$F$5-0,2*$F$6-0,15*$F$7
	B21	=СУММ(C6:D6)-0,2*$F$5-0,3*$F$6-0,15*$F$7
	B22	=СУММ(C7:D7)-0,15*$F$5-0,1*$F$6-0,4*$F$7
	B23	=СУММ(C8:D8)-0,35*$F$5-0,4*$F$6-0,3*$F$7

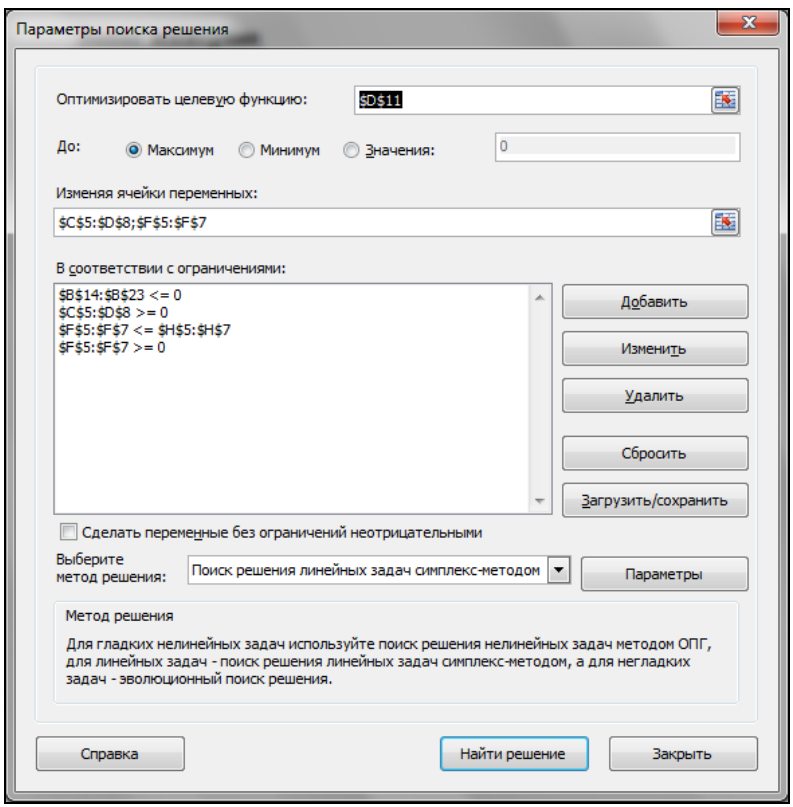


Рис. 9.13. Заполнение окна **Параметры поиска решения** для задачи о производстве удобрений

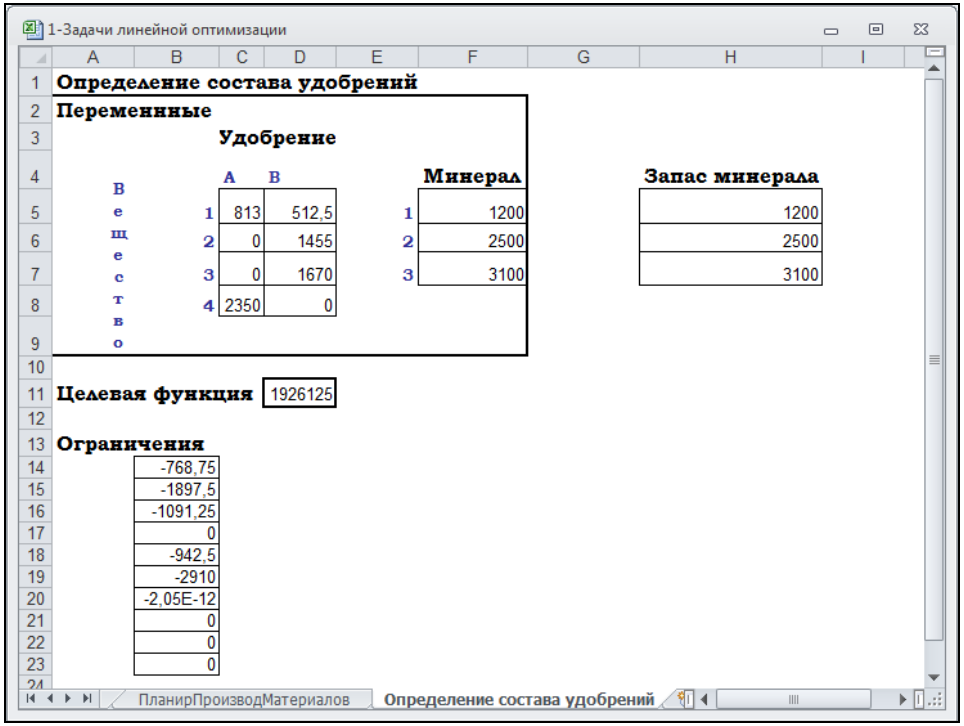


Рис. 9.14. Оптимальное решение задачи о производстве удобрений

Решаем транспортную задачу

В общем случае транспортную задачу можно сформулировать следующим образом: в m пунктах отправления $A_1, ..., A_m$ находится однородный груз, количество которого равно соответственно $a_1, ..., a_m$ единиц. Данный груз необходимо доставить потребителям $B_1, ..., B_n$, спрос которых — $b_1, ..., b_n$. Стоимость перевозки единицы груза из i-го $i = \overline{1, m}$ пункта отправления в j-й $j = \overline{1, n}$ пункт назначения равна c_{ij}. Необходимо составить план перевозок, который полностью удовлетворяет спрос потребителей в грузе, и при этом суммарные транспортные издержки минимальны.

Математически транспортную задачу можно записать следующим образом:

$$F = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \to \min, \tag{9.1}$$

$$\sum_{j=1}^{n} x_{ij} = a_i, \ i = \overline{1, m}, \qquad \sum_{i=1}^{m} x_{ij} = b_j, \ j = \overline{1, n}, \tag{9.2}$$

$$x_{ij} \geq 0, \ i = \overline{1, m}, \ j = \overline{1, n}. \tag{9.3}$$

Таким образом, даны: система ограничений (9.2) при условии (9.3) и линейная функция (9.1). Требуется среди множества решений системы (9.2) найти такое неотрицательное решение, которое доставляет минимум линейной функции (9.1).

Модель транспортной задачи называют *закрытой* (сбалансированной), если суммарный объем груза, имеющегося у поставщиков, равен суммарному спросу потребителей, т. е. выполняется равенство:

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j.$$

Если для транспортной задачи выполняется одно из условий:

$$\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j, \ \sum_{i=1}^{m} a_i < \sum_{j=1}^{n} b_j,$$

то модель задачи называют *открытой* (несбалансированной) .

Для разрешимости транспортной задачи с открытой моделью следует ее преобразовать в закрытую. Так, если выполняется условие $\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j$, то необходимо ввести фиктивный $n+1$ -й пункт назначения B_{n+1}, т. е. в матрице задачи вводится дополнительный столбец. Спрос фиктивного потребителя равен $b_{n+1} = \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j$. Стоимость перевозок продукции полагается одинаковой, чаще всего равной нулю (если не задана стоимость складирования продукции), т. е. $c_{i,\,n+1} = 0, \ i = \overline{1, m}.$

Если выполняется условие $\sum_{i=1}^{m} a_i < \sum_{j=1}^{n} b_j$, то необходимо ввести фиктивного $m+1$ -го поставщика A_{m+1}, т. е. в матрице задачи вводится дополнительная строка. Запас груза данного поставщика равен $a_{m+1} = \sum_{j=1}^{n} b_j - \sum_{i=1}^{m} a_i$. Стоимость перевозок продукции полагается одинаковой, чаще всего равной нулю (если не задана стоимость штрафов за недопоставку продукции), т. е. $c_{m+1,\,j} = 0, \ j = \overline{1, n}.$

При преобразовании открытой задачи в закрытую, целевая функция не меняется, т. к. все слагаемые, соответствующие дополнительным перевозкам, равны нулю.

Пример решения транспортной задачи

Итак, рассмотрим пример решения транспортной задачи с использованием надстройки **Поиск решения** (см. также файл *2-Транспортная задача.xlsx* на компакт-диске). Пусть производство продукции осуществляется на 4-х предприятиях, а затем развозится в 5 пунктов потребления. Предприятия могут выпускать в день 235, 175, 185 и 175 единиц продукции. Пункты потребления готовы принимать ежедневно 125, 160, 60, 250 и 175 единиц продукции. Хранение на предприятии единицы про-

дукции обходится в 2 у. е. в день, штраф за недопоставленную продукцию — 3,5 у. е. в день. Стоимость перевозки единицы продукции (в у. е.) с предприятий в пункты потребления приведена в табл. 9.8.

Таблица 9.8. *Транспортные расходы*

Предприятия	Пункты потребления				
	1	**2**	**3**	**4**	**5**
1	3,2	3	2,35	4	3,65
2	3	2,85	2,5	3,9	3,55
3	3,75	2,5	2,4	3,5	3,4
4	4	2	2,1	4,1	3,4

Необходимо минимизировать суммарные транспортные расходы по перевозке продукции.

Для решения транспортной задачи с использованием надстройки **Поиск решения** выполните следующие шаги.

1. Проверка сбалансированности модели задачи. Модель является сбалансированной, т. к. суммарный объем производимой продукции в день равен суммарному объему потребности в ней:

$$235 + 175 + 185 + 175 = 125 + 160 + 60 + 250 + 175.$$

   Поэтому при решении этой задачи не учитываются издержки, связанные со складированием и недопоставкой продукции.

2. Построение математической модели. Объемы перевозок — это неизвестные задачи. Пусть x_{ij} — объем перевозок с i-го предприятия в j-й пункт потребления.

   Суммарные транспортные расходы — это функционал качества (критерий цели):

$$F = \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij},$$

   где c_{ij} — стоимость перевозки единицы продукции с i-го предприятия в j-й пункт потребления.

   Неизвестные в данной задаче должны удовлетворять следующим ограничениям:

   - объемы перевозок не могут быть отрицательными;
   - т. к. модель сбалансирована, то вся продукция должна быть вывезена с предприятий, а потребности всех пунктов потребления должны быть полностью удовлетворены.

   Итак, имеем следующую задачу: найти минимум функционала:

$$F = \sum_{i=1}^{4} \sum_{j=1}^{5} c_{ij} x_{ij} \to \min,$$

при ограничениях

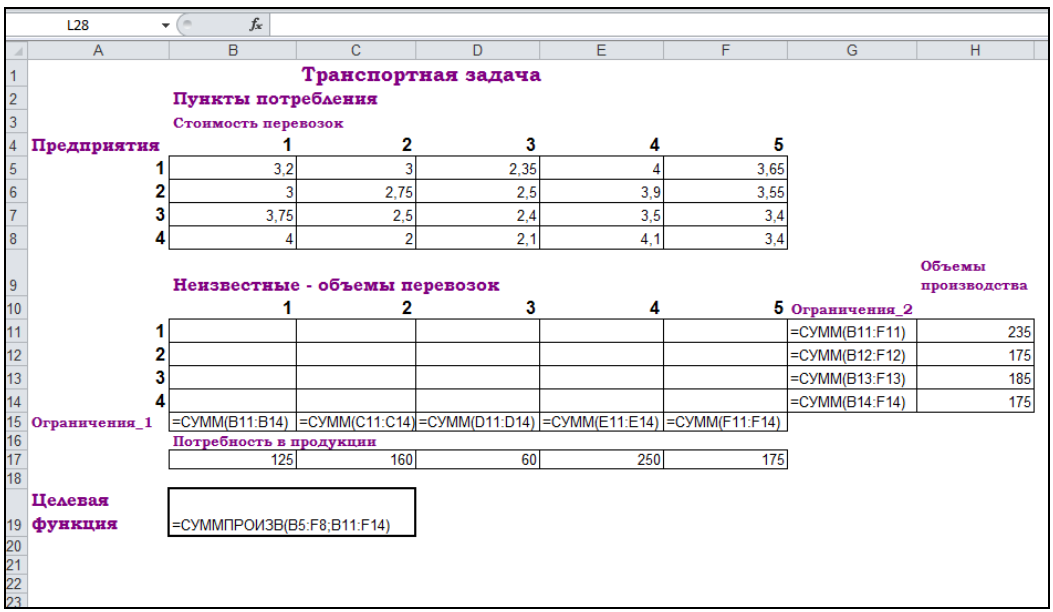
$$\sum_{i=1}^{4} x_{ij} = b_j,\ \ j \in 1,5\ ,$$

$$\sum_{j=1}^{5} x_{ij} = a_i,\ \ i \in 1,4\ ,$$

$$x_{ij} \geq 0,\ \ \ i \in 1,4\ ,\ \ j \in 1,5\ ,$$

где a_i — объем производства на i-м предприятии, b_i — спрос в j-м пункте потребления.

3. Решение задачи с помощью окна **Параметры поиска решения**.
Подготовка рабочего листа для задачи осуществляет в соответствии с рис. 9.15. Формулы для расчета приведены в табл. 9.9.



	A	B	C	D	E	F	G	H
1		**Транспортная задача**						
2		**Пункты потребления**						
3		**Стоимость перевозок**						
4	**Предприятия**	**1**	**2**	**3**	**4**	**5**		
5	**1**	3,2	3	2,35	4	3,65		
6	**2**	3	2,75	2,5	3,9	3,55		
7	**3**	3,75	2,5	2,4	3,5	3,4		
8	**4**	4	2	2,1	4,1	3,4		
9		**Неизвестные - объемы перевозок**						**Объемы производства**
10		**1**	**2**	**3**	**4**	**5**	**Ограничения_2**	
11	**1**						=СУММ(B11:F11)	235
12	**2**						=СУММ(B12:F12)	175
13	**3**						=СУММ(B13:F13)	185
14	**4**						=СУММ(B14:F14)	175
15	**Ограничения_1**	=СУММ(B11:B14)	=СУММ(C11:C14)	=СУММ(D11:D14)	=СУММ(E11:E14)	=СУММ(F11:F14)		
16		**Потребность в продукции**						
17		125	160	60	250	175		
18								
19	**Целевая функция**	=СУММПРОИЗВ(B5:F8;B11:F14)						

Рис. 9.15. Исходные данные для решения транспортной задачи

Таблица 9.9. *Формулы для расчетов в транспортной задаче*

Описание	**Ячейка**	**Формула**
Ограничения_1	**G11**	=СУММ(B11:F11)
	G12	=СУММ(B12:F12)
	G13	=СУММ(B13:F13)
	G14	=СУММ(B14:F14)

Таблица 9.9 *(окончание)*

Описание	Ячейка	Формула
Ограничения_2	**B15**	=СУММ(B11:B14)
	C15	=СУММ(C11:C14)
	D15	=СУММ(D11:D14)
	E15	=СУММ(E11:E14)
	F15	=СУММ(F11:F14)
Целевая функция	**B19**	=СУММПРОИЗВ(B5:F8;B11:F14)

Ввод данных в окно **Параметры поиска решения** производится в соответствии с рис. 9.16. Не следует забывать также об опциях окна **Параметры** (кнопка **Параметры** в окне **Параметры поиска решения**) — **Точность ограничения** (на вкладке **Все методы**).

Полученное оптимальное решение представлено на рис. 9.17.

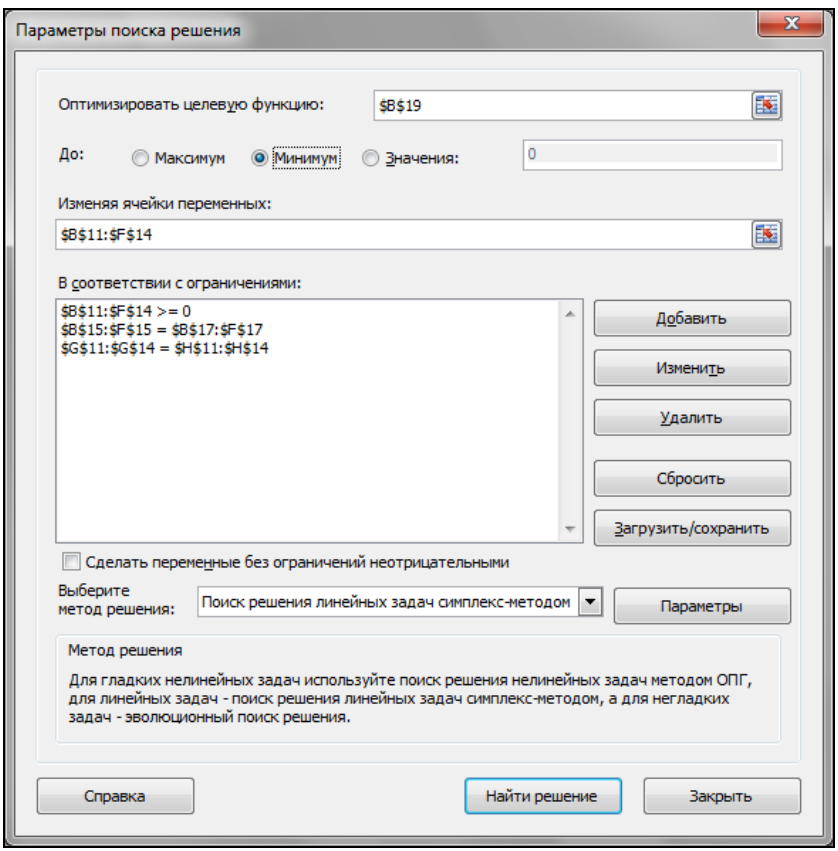
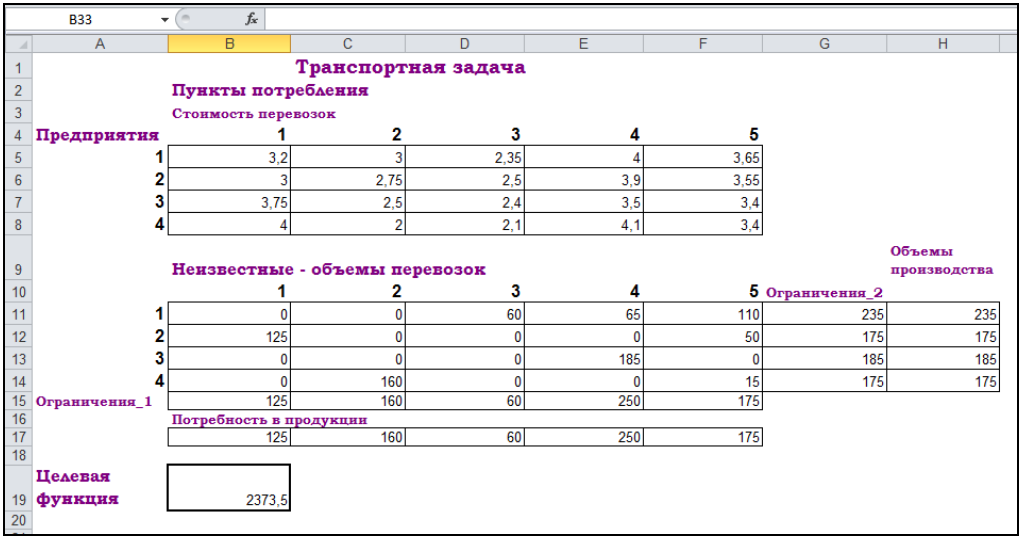


Рис. 9.16. Ввод данных в окно **Параметры поиска решения** для транспортной задачи



	A	B	C	D	E	F	G	H
1		**Транспортная задача**						
2		**Пункты потребления**						
3		**Стоимость перевозок**						
4	**Предприятия**	**1**	**2**	**3**	**4**	**5**		
5	**1**	3,2	3	2,35	4	3,65		
6	**2**	3	2,75	2,5	3,9	3,55		
7	**3**	3,75	2,5	2,4	3,5	3,4		
8	**4**	4	2	2,1	4,1	3,4		
9		**Неизвестные - объемы перевозок**						**Объемы производства**
10		**1**	**2**	**3**	**4**	**5**	**Ограничения_2**	
11	**1**	0	0	60	65	110	235	235
12	**2**	125	0	0	0	50	175	175
13	**3**	0	0	0	185	0	185	185
14	**4**	0	160	0	0	15	175	175
15	**Ограничения_1**	125	160	60	250	175		
16		**Потребность в продукции**						
17		125	160	60	250	175		
18								
19	**Целевая функция**	2373,5						
20								

Рис. 9.17. Оптимальное решение для транспортной задачи

Что такое дискретное программирование?

Дискретное программирование изучает экстремальные задачи, в которых на искомые переменные накладывается условие дискретности, а область допустимых решений конечна. Это, прежде всего, задачи с физической неделимостью многих факторов и объектов расчета. К дискретному программированию относят также ряд задач целочисленного программирования, в которых искомые переменные принимают только целочисленные значения (например, задача о планировании штатного расписания) или логические, булевы, значения 0 или 1 (например, задача о назначениях). Рассмотрим решение задачи о назначениях.

Каждый из преподавателей может провести определенные виды занятий. Почасовая оплата c_{ij} i-му преподавателю по j-му виду занятий приведена в табл. 9.10.

Составить план проведения учебных занятий так, чтобы все виды занятия были проведены, каждый преподаватель проводил занятия только по одному виду, а суммарная стоимость почасовой оплаты была минимальной.

Таблица 9.10. *Стоимости выполнения работы*

Преподаватели	**Почасовая оплата курсов**			
	1	**2**	**3**	**4**
1	350	420	610	200
2	890	130	650	900
3	430	520	600	720
4	830	610	780	470

Для решения данной задачи выполните следующие шаги (см. файл *3-Задача о назначениях.xlsx* на компакт-диске).

1. Проверка задачи на сбалансированность. Задача является сбалансированной, т. к. количество преподавателей соответствует числу возможных видов занятий. В случае несбалансированности задачи необходимо ввести недостающее число фиктивных преподавателей (строчек) или видов занятий (столбцов).
2. Построение математической модели задачи. Пусть $x_{ij} = 1$ в случае выполнения i-м преподавателем j-го вида занятий, и $x_{ij} = 0$ — в случае невыполнения вида занятий. Тогда математическая модель задачи примет вид: найти минимум функционала

$$f = \sum_{i=1}^{4} \sum_{j=1}^{4} c_{ij} x_{ij} \to \min$$

при следующих ограничениях:

$$\sum_{i=1}^{4} x_{ij} = 1, \ j = \overline{1,4},$$

$$\sum_{i=1}^{4} x_{ij} = 1, \ i = \overline{1,4},$$

$$x_{ij} \in \{0,1\}, \ i = \overline{1,4}, \ j = \overline{1,4}.$$

3. Решение задачи с помощью надстройки **Поиск решения**.
   Подготовка рабочего листа может быть произведена в соответствии с рис. 9.18. Формулы для расчета приведены в табл. 9.11.

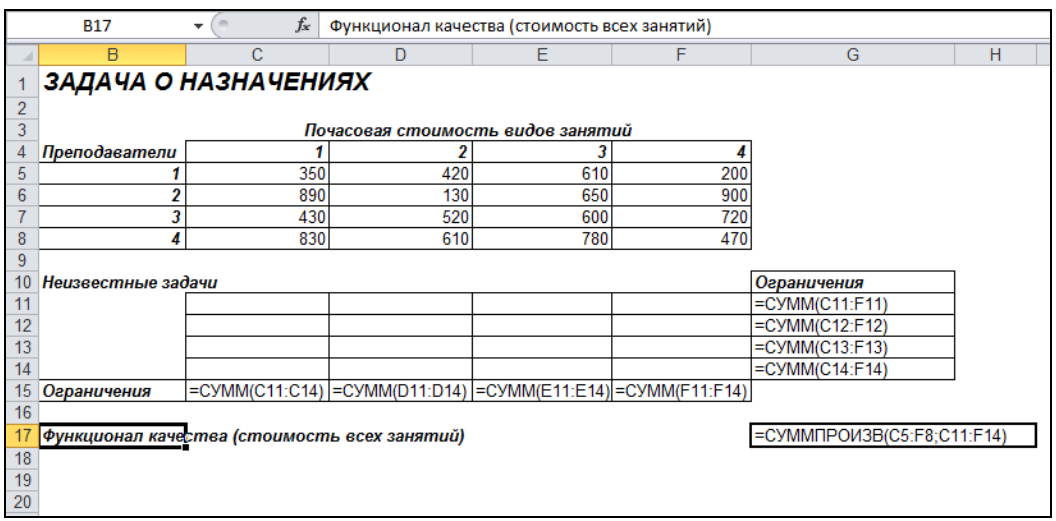


Рис. 9.18. Подготовка рабочего листа для решения задачи о назначениях

Таблица 9.11. *Формулы для расчета в задаче о назначениях*

Описание	Ячейка	Формула
Ограничения	**G11**	=СУММ(C11:F11)
	G12	=СУММ(C12:F12)
	G13	=СУММ(C13:F13)
	G14	=СУММ(C14:F14)
Ограничения	**C15**	=СУММ(C11:C14)
	D15	=СУММ(D11:D14)
	E15	=СУММ(E11:E14)
	F15	=СУММ(F11:F14)
Функционал качества (стоимость работ)	**G17**	=СУММПРОИЗВ(C5:F8;C11:F14)

Установка ограничений в окне **Параметры поиска решения** приведено на рис. 9.19.
Решение задачи получено на рис. 9.20.

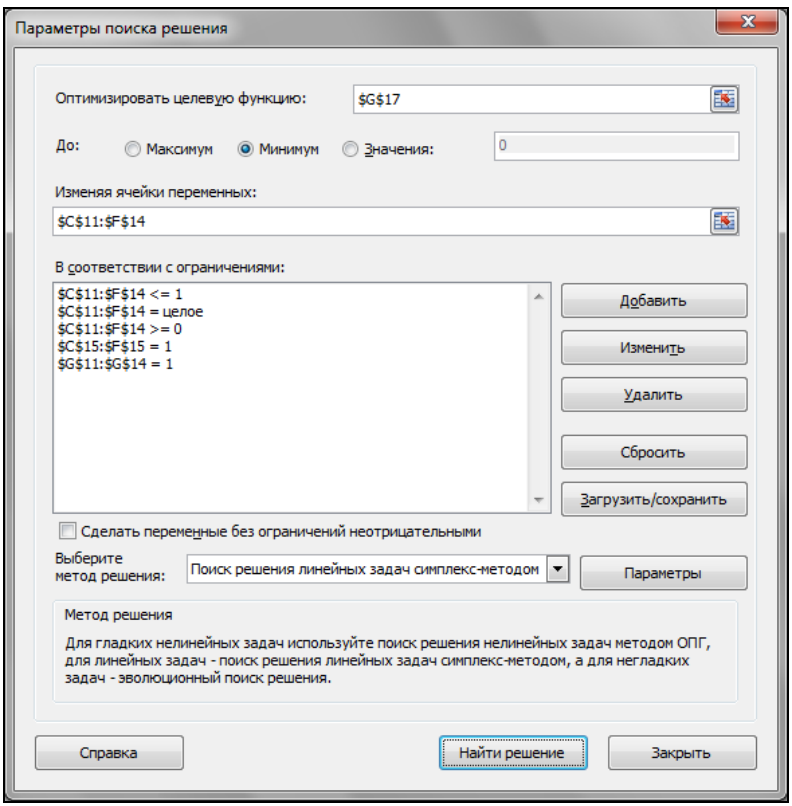
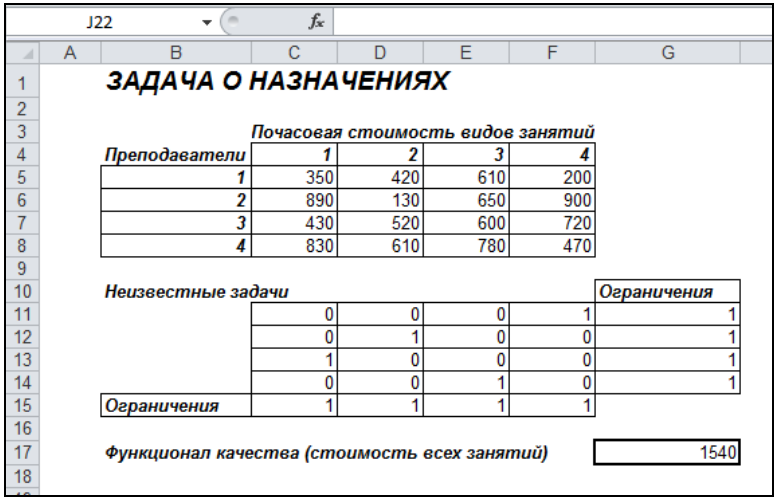


Рис. 9.19. Установка параметров в окне **Параметры поиска решения** для задачи о назначениях



ЗАДАЧА О НАЗНАЧЕНИЯХ

		Почасовая стоимость видов занятий			
Преподаватели	1	2	3	4	
1	350	420	610	200	
2	890	130	650	900	
3	430	520	600	720	
4	830	610	780	470	
Неизвестные задачи					Ограничения
	0	0	0	1	1
	0	1	0	0	1
	1	0	0	0	1
	0	0	1	0	1
Ограничения	1	1	1	1	
Функционал качества (стоимость всех занятий)					1540

Рис. 9.20. Решение задачи о назначениях

Решаем задачу нелинейного программирования

Задача нелинейного программирования формулируется подобно задаче линейного программирования, но с учетом того, что целевая функция или (и) хотя бы одно ограничение являются нелинейными. Вследствие этого задачи нелинейного программирования (НП) сложнее задач линейного программирования (ЛП). И для них не существует общего метода решения аналогичного симплексному методу в ЛП. Следует также заметить, что задачи НП включают также нелинейные целочисленные задачи и задачи дискретного программирования. С учетом методов решения, задачи нелинейной оптимизации делятся на задачи условной оптимизации (поиск экстремума функции с учетом дополнительных условий в виде ограничений и граничных условий) и задачи безусловной оптимизации (поиск экстремума функции без всяких дополнительных условий). Для решения такого типа задач существует много различных методов. Применение того или иного метода решения зависит от типа нелинейности. Надстройка **Поиск решения** помогает облегчить численное решение задач нелинейного программирования.

Разберем решение системы нелинейных уравнений с двумя неизвестными с помощью надстройки **Поиск решения**:

$$\begin{cases} f_1(x, y) = C_1, \\ f_2(x, y) = C_2, \end{cases} \tag{9.4}$$

где $f_i(x, y)$, $i = 1, 2$ — нелинейная функция от переменных x и y, C_i, $i = 1, 2$ — произвольная постоянная.

Известно, что пара (x, y) является решением системы уравнений (9.4) тогда и только тогда, когда она является решением следующего нелинейного уравнения с двумя неизвестными:

$$f_1(x,y)-C_1\ ^2+\ f_2(x,y)-C_2\ ^2=0. \qquad (9.5)$$

С другой стороны, решение системы (9.4) — это точки пересечения двух кривых: $f_1(x,y)=C_1$ и $f_2(x,y)=C_2$ на плоскости *xOy*.

Исходя из этого, метод нахождения корней системы нелинейных уравнений таков.

1. Определить (хотя бы приближенно) интервал существования решения системы уравнений (9.4) или уравнения (9.5). Здесь следует учитывать вид уравнений, входящих в систему, область определения каждого их уравнений и т. п. Иногда применяется подбор начального приближения решения.
2. Протабулировать решение уравнения (9.5) по переменным *x* и *y* на выбранном интервале либо построить графики функций $f_1(x,y)=C_1$ и $f_2(x,y)=C_2$ (система (9.4)).
3. Локализовать предполагаемые корни системы уравнений — найти несколько минимальных значений из таблицы табулирования корней уравнения (9.5) либо определить точки пересечения кривых, входящих в систему (9.4).
4. Найти корни для системы уравнений (9.4) с помощью надстройки **Поиск решения**.

Итак, решим следующую систему нелинейных уравнений (см. также файл *4-Нелинейное программирование.xlsx* на компакт-диске):

$$\begin{cases} x-1\ ^2+\ y+1\ ^2=4, \\ 5x+4y=2. \end{cases}$$

Легко видеть, что решением системы уравнений являются точки пересечения окружности (с радиусом 2 и центром (1, –1)) и прямой.

Данную систему заменяем равносильным уравнением:

$$x-1\ ^2+\ y+1\ ^2-4\ ^2+\ 5x+4y-2\ ^2=0,$$

для которого будем искать решения с помощью надстройки **Поиск решения**.

1. Исходя из графиков уравнений, интервал локализации корней лежит в границах от –3 до 3 (рис. 9.21). Ячейки **B3:B43** содержат значения *X*. Формулы для построения графиков:
   - для ячейки **C3**: `=-1+КОРЕНЬ(4-(B3-1)^2)`;
   - для ячейки **D3**: `=-1-КОРЕНЬ(4-(B3-1)^2)`;
   - для ячейки **E3**: `=(2-5*B3)/4`.
2. Табулируем равносильное уравнение на отрезке [–3; 3] с шагом 0,5 (рис. 9.22).

Диаграмма 1

РЕШЕНИЕ СИСТЕМЫ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

1.	x	окружность	прямая
	-1	-1	-1	1,75
	-0,9	-0,3755	-1,6245	1,625
	-0,8	-0,12822	-1,87178	1,5
	-0,7	0,0535654	-2,053565	1,375
	-0,6	0,2	-2,2	1,25
	-0,5	0,3228757	-2,322876	1,125
	-0,4	0,4282857	-2,428286	1
	-0,3	0,5198684	-2,519868	0,875
	-0,2	0,6	-2,6	0,75
	-0,1	0,6703293	-2,670329	0,625
	0	0,7320508	-2,732051	0,5
	0,1	0,7860571	-2,786057	0,375
	0,2	0,8330303	-2,83303	0,25
	0,3	0,8734994	-2,873499	0,125
	0,4	0,9078784	-2,907878	0
	0,5	0,9364917	-2,936492	-0,125
	0,6	0,9595918	-2,959592	-0,25
	0,7	0,977372	-2,977372	-0,375
	0,8	0,9899749	-2,989975	-0,5
	0,9	0,9974984	-2,997498	-0,625
	1	1	-3	-0,75
	1,1	0,9974984	-2,997498	-0,875
	1,2	0,9899749	-2,989975	-1
	1,3	0,977372	-2,977372	-1,125
	1,4	0,9595918	-2,959592	-1,25
	1,5	0,9364917	-2,936492	-1,375
	1,6	0,9078784	-2,907878	-1,5
	1,7	0,8734994	-2,873499	-1,625
	1,8	0,8330303	-2,83303	-1,75
	1,9	0,7860571	-2,786057	-1,875
	2	0,7320508	-2,732051	-2
	2,1	0,6703293	-2,670329	-2,125
	2,2	0,6	-2,6	-2,25
	2,3	0,5198684	-2,519868	-2,375
	2,4	0,4282857	-2,428286	-2,5
	2,5	0,3228757	-2,322876	-2,625
	2,6	0,2	-2,2	-2,75
	2,7	0,0535654	-2,053565	-2,875

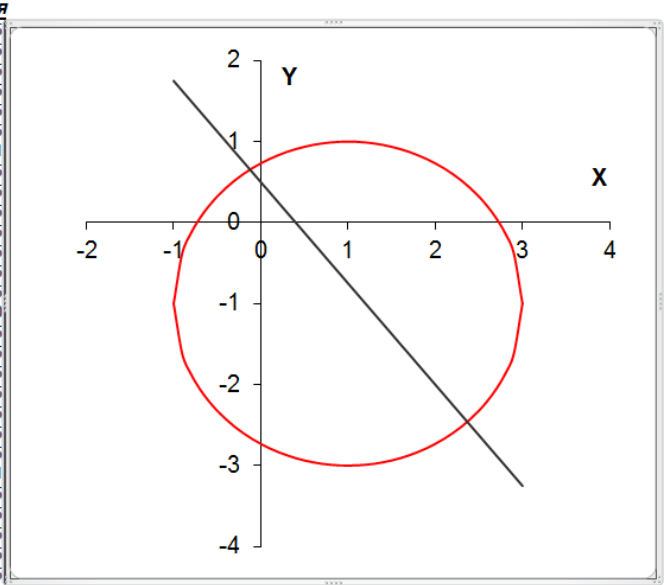


Рис. 9.21. Графическое решение системы линейных уравнений

K85

2.

	-3	-2,5	-2	-1,5	-1	-0,5	0	0,5	1	1,5	2	2,5	3
-3	1097	932,0625	794	679,063	585	511,0625	458	428,0625	425	454,0625	522	637,0625	809
-2,5	852,31	710,5	591,8125	492,5	410,3125	344,5	295,813	266,5	260,3125	282,5	339,8125	440,5	594,3125
-2	657	536,5625	436	351,563	281	223,5625	180	152,5625	145	162,5625	212	301,5625	441
-1,5	501,31	400,5	316,8125	246,5	187,3125	138,5	100,813	76,5	69,3125	84,5	128,8125	210,5	339,3125
-1	377	294,0625	226	169,063	121	81,0625	50	30,0625	25	40,0625	82	159,0625	281
-0,5	277,31	210,5	156,8125	112,5	75,3125	44,5	20,8125	6,5	5,3125	22,5	64,8125	140,5	259,3125
0	197	144,5625	104	71,5625	45	23,5625	8	0,5625	5	26,5625	72	149,5625	269
0,5	132,31	92,5	63,8125	42,5	26,3125	14,5	7,8125	8,5	20,3125	48,5	99,8125	182,5	306,3125
1	81	52,0625	34	23,0625	17	15,0625	18	28,0625	49	86,0625	146	237,0625	369
1,5	42,313	22,5	13,8125	12,5	16,3125	24,5	37,8125	58,5	90,3125	138,5	209,8125	312,5	456,3125
2	17	4,5625	4	11,5625	25	43,5625	68	100,5625	145	206,5625	292	409,5625	569
2,5	7,3125	0,5	6,8125	22,5	45,3125	74,5	110,813	156,5	215,3125	292,5	394,8125	530,5	709,3125
3	17	14,0625	26	49,0625	81	121,0625	170	230,0625	305	400,0625	522	679,0625	881

Рис. 9.22. Табулирование функции для нахождения решения системы уравнений

3. Локализация корней равносильного уравнения (рис. 9.23). Ячейки **A47:A59** содержат значения *X* на отрезке [–3; 3] с шагом 0,5, ячейки **B46:N46** содержат значения *Y* на отрезке [–3; 3] с шагом 0,5. Формула для ячейки **B47** (копируется на диапазон **B47:N59**):

```
=(($A47-1)^2+(B$46+1)^2-4)^2+(5*$A47+4*B$46-2)^2
```

Формула для ячейки **B62** (копируется на диапазон **B62:N62**):

```
=МИН(B47:B60)
```

Исходя из полученных вычислений, можно определить следующие пары предполагаемых корней уравнения: (–2,5; –2,5), (2; –2), (0; 0,5) и (0; 1).

45	2.													
46		-3	-2,5	-2	-1,5	-1	-0,5	0	0,5	1	1,5	2	2,5	3
47	-3	1097	932,0625	794	679,063	585	511,0625	458	428,0625	425	454,0625	522	637,0625	809
48	-2,5	852,31	710,5	591,8125	492,5	410,3125	344,5	295,813	266,5	260,3125	282,5	339,8125	440,5	594,3125
49	-2	657	536,5625	436	351,563	281	223,5625	180	152,5625	145	162,5625	212	301,5625	441
50	-1,5	501,31	400,5	316,8125	246,5	187,3125	138,5	100,813	76,5	69,3125	84,5	128,8125	210,5	339,3125
51	-1	377	294,0625	226	169,063	121	81,0625	50	30,0625	25	40,0625	82	159,0625	281
52	-0,5	277,31	210,5	156,8125	112,5	75,3125	44,5	20,8125	6,5	5,3125	22,5	64,8125	140,5	259,3125
53	0	197	144,5625	104	71,5625	45	23,5625	8	0,5625	5	26,5625	72	149,5625	269
54	0,5	132,31	92,5	63,8125	42,5	26,3125	14,5	7,8125	8,5	20,3125	48,5	99,8125	182,5	306,3125
55	1	81	52,0625	34	23,0625	17	15,0625	18	28,0625	49	86,0625	146	237,0625	369
56	1,5	42,313	22,5	13,8125	12,5	16,3125	24,5	37,8125	58,5	90,3125	138,5	209,8125	312,5	456,3125
57	2	17	4,5625	4	11,5625	25	43,5625	68	100,5625	145	206,5625	292	409,5625	569
58	2,5	7,3125	0,5	6,8125	22,5	45,3125	74,5	110,813	156,5	215,3125	292,5	394,8125	530,5	709,3125
59	3	17	14,0625	26	49,0625	81	121,0625	170	230,0625	305	400,0625	522	679,0625	881

Рис. 9.23. Локализация корней системы уравнений

4. Нахождение корней равносильного уравнения. Для этого следует поместить пары значений для предполагаемых корней в ячейки **D69:E72**. В ячейку **G69** (рис. 9.24) ввести формулу для равносильного уравнения (копируется на диапазон **G69:G72**):

```
=((D69-1)^2+(E69+1)^2-4)^2+(5*D69+4*E69-2)^2
```

	A	B	C	D	E	F	G	H
66	4.							
67	Предполагаемые значения корней уравнения							
68				x	y		Значение уравнения	
69				2,3675746	-2,45934		2,56434E-07	
70				2,367553	-2,45933		2,42523E-07	
71				-0,123564	0,65443		2,10512E-07	
72				-0,123669	0,65458		6,80851E-08	
73								

Рис. 9.24. Подготовка листа рабочей книги для нахождения корней нелинейной системы уравнений

С помощью надстройки **Поиск решения** установить необходимые параметры для поиска корня равносильного уравнения (рис. 9.25), затем выполнить поиск решения. Процедуру повторить для всех имеющихся пар корней.

Результаты поиска решения (рис. 9.26) позволяют сделать вывод о том, что система имеет 2 решения: (2,3675745729901; –2,45934248863711) и (–0,123564081639673; 0,654434224216163).

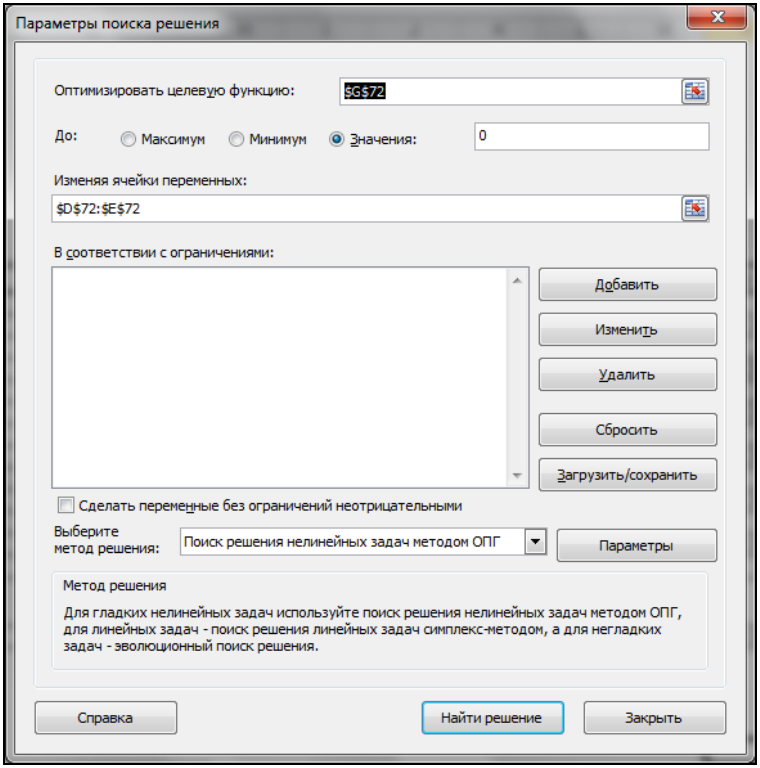
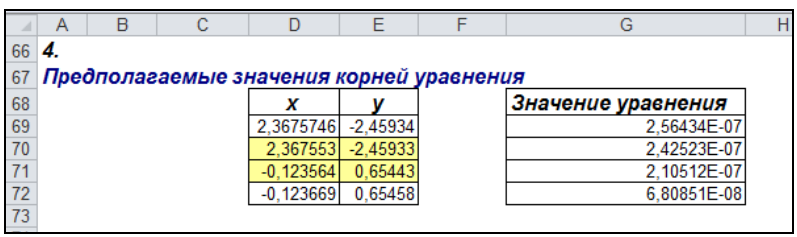


Рис. 9.25. Ввод данных в окно **Поиск решения** для задачи нахождения корней системы уравнений



	A	B	C	D	E	F	G	H
66	***4.***							
67	***Предполагаемые значения корней уравнения***							
68				***x***	***y***		***Значение уравнения***	
69				2,3675746	-2,45934		2,56434E-07	
70				2,367553	-2,45933		2,42523E-07	
71				-0,123564	0,65443		2,10512E-07	
72				-0,123669	0,65458		6,80851E-08	
73								

Рис. 9.26. Результаты поиска решения для нелинейной системы уравнений

Какие функции программируют поиск решения?

После решения транспортной задачи вручную при помощи средства **Поиск решения** рассмотрим примеры, которые демонстрируют его программное использование. Для решения задач оптимизации используются следующие функции:

- `SolverAdd();`
- `SolverChange();`
- `SolverDelete();`
- `SolverFinish();`
- `SolverLoad();`
- `SolverOk();`
- `SolverOptions();`
- `SolverReset();`
- `SolverSave();`
- `SolverSolve().`

Функция `SolverOk()` позволяет поставить задачу оптимизации. Она задает значения тех параметров, которые вручную устанавливаются в окне **Поиск решения**.

`SolverOk(`*`SetCell`*`, `*`MaxMinVal`*`, `*`ValueOf`*`, `*`ByChange`*`)`

- *`SetCell`* — задает ячейку, содержащую формулу с функцией цели.
- *`MaxMinVal`* — задает тип задачи, решаемой для функции цели. Допустимые значения:
  - 1 — для задачи максимизации;
  - 2 — для задачи минимизации;
  - 3 — для задачи нахождения значения.
- *`ValueOf`* — если значение параметра *`MaxMinVal`* равно 3, то этот параметр задает то значение, которое функция цели должна достичь.
- *`ByChange`* — задает диапазон изменяемых ячеек, т. е. тех ячеек, которые отведены под переменные оптимизационной задачи.

Функция `SolverAdd()` позволяет добавлять в модель ограничения. Она задает значения тех параметров, которые вручную устанавливаются в окне **Добавление ограничения**.

`SolverAdd(`*`CellRef`*`, `*`Relation`*`, `*`FormulaText`*`)`

- *`CellRef`* — ссылка на ячейку или диапазон ячеек из левой части ограничений.
- *`Relation`* — целое число от 1 до 5, задающее тип соотношения между левой и правой частями ограничения. Если значение этого параметра равно 4 или 5, то значение параметра `FormulaText` опускается. Допустимые значения:
  - 1 — соответствует соотношению <=;
  - 2 — соответствует соотношению =;
  - 3 — соответствует соотношению >=;
  - 4 — допустимыми значениями диапазона ячеек, заданного параметром *`CellRef`*, являются целые числа;
  - 5 — допустимыми значениями диапазона ячеек, заданного параметром *`CellRef`*, являются только 0 и 1.
- *`FormulaText`* — ссылка на ячейку или диапазон ячеек из правой части ограничений либо значение из правой части ограничений.

Если список ограничений оптимизационной задачи уже задан, то в коде из него можно удалять ограничения функцией `SolverDelete()`, а изменять ограничения — функцией `SolverChange()`. Синтаксис этих функций такой же, как у функции `SolverAdd()`. А именно:

`SolverDelete(`*`CellRef`*`, `*`Relation`*`, `*`FormulaText`*`)`

`SolverChange(`*`CellRef`*`, `*`Relation`*`, `*`FormulaText`*`)`

Функция `SolverOptions()` задает значения тех параметров поиска решения, которые вручную устанавливаются в окне **Параметры поиска решения**. Мы опустим описание этих параметров, т. к., по сути дела, они подробно рассмотрены при описании структуры окна **Параметры поиска решения**.

`SolverOptions(`*`MaxTime`*`, `*`Iterations`*`, `*`Precision`*`, `*`AssumeLinear`*`, `*`StepThru`*`,`
↳*`Estimates`*`, `*`Derivatives`*`, `*`Search`*`, `*`IntTolerance`*`, `*`Scaling`*`, `*`Convergence`*`,`
↳*`AssumeNonNeg`*`)`

Тесно с функцией `SolverOptions()` связана функция `SolverReset()`, которая устанавливает значения параметров поиска решения, используемые по умолчанию. У этой функции нет параметров.

Функция `SolverSolve()` запускает поиск решения, ее действие эквивалентно нажатию кнопки **Выполнить** окна **Поиск решения**. Она возвращает значение 0, если решение найдено.

```
SolverSolve(UserFinish, ShowRef)
```

- *UserFinish* — параметр, принимающий логические значения. Если его значение равно `False` (которое устанавливается по умолчанию), результаты отображаются в диалоговом окне.
- *ShowRef* — ссылка на макрос, который должен выполняться между итерациями поиска решения.

Функция `SolverSave()` сохраняет модель.

```
SolverSave(SaveArea)
```

Здесь параметр *SaveArea* задает диапазон ячеек, в котором сохраняется модель. Например:

```
SolverSave("Лист1!A1:A3")
```

Функция `SolverLoad()` загружает модель.

```
SolverLoad(LoadArea)
```

Здесь параметр *LoadArea* задает диапазон ячеек, в котором записана модель.

> ***Примечание***
>
> До использования перечисленных функций необходимо установить ссылку на Solver в редакторе Visual Basic в окне **References**, отображаемом на экране выбором команды **Tools | References**. Если Solver отсутствует в списке **Available References**, нажмите кнопку **Browse** и откройте Solver.xlam из каталога C:\Program Files\Microsoft Office\Office14\Library\ Solver.

Приложение "Транспортная задача"

Создадим простое приложение, решающее транспортную задачу. В этом приложении пользователь будет указывать ссылки на диапазоны ячеек, в которые введены стоимости перевозок и объемы ввоза продукции в пункты потребления и вывоза продукции из пунктов производства, а также задавать диапазон ячеек, которые отводятся под определяемые объемы перевозок. Все остальные необходимые действия будет производить приложение: ввод в ячейки рабочего листа вспомогательные формулы, которые возвращают суммарную стоимость перевозок и объемы ввозимой и вывозимой продукции в каждом из пунктов; проверка модели на сбалансированность; задание всех необходимых параметров поиска решения; запуск на выполнение. Результат нахождения решения будет выведен в виде соответствующей надписи на рабочем листе.

Перейдем к конструированию приложения. Прежде всего, разместите необходимые данные на рабочем листе и отведите место под целевую ячейку. Разместите также на рабочем листе кнопку, которая будет запускать соответствующую форму для вашего приложения. Далее создайте форму, на ней расположите четыре надписи, четыре элемента управления **RefEdit** и кнопку (рис. 9.27), установите при помощи окна **Properties** значения их свойств, как показано в табл. 9.12.

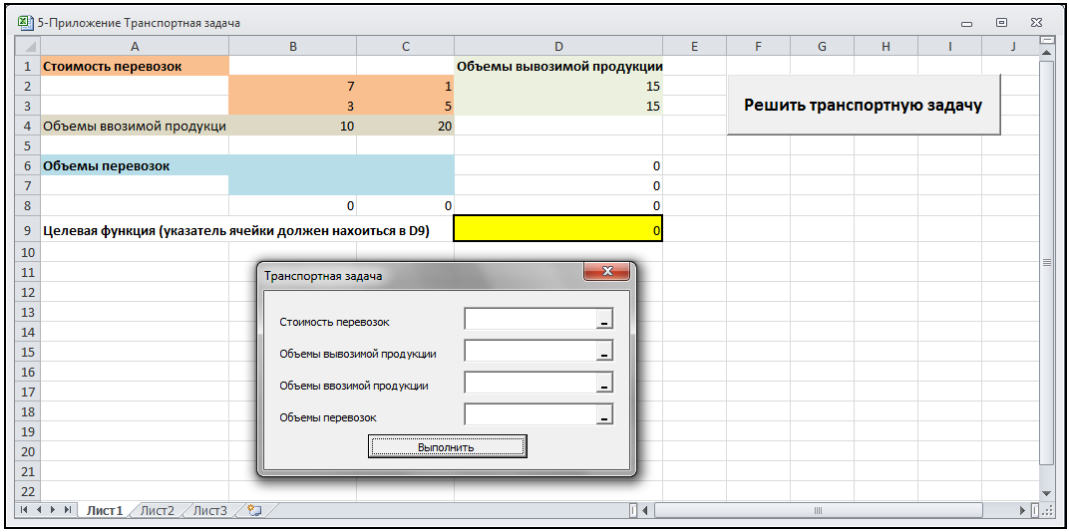


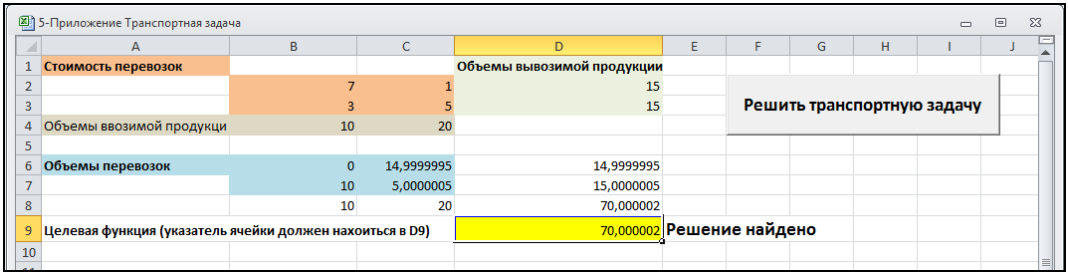
Рис. 9.27. Окно **Транспортная задача** и исходные данные на рабочем листе

Таблица 9.12. *Значения свойств, установленные в окне* ***Properties***

Объект	Свойство	Значение
Форма	Name	frmSolver
	Caption	Транспортная задача
Надпись	Caption	Стоимость перевозок
RefEdit	Name	refCosts
Надпись	Caption	Объемы вывозимой продукции
RefEdit	Name	refOut
Надпись	Caption	Объемы ввозимой продукции
RefEdit	Name	refIn
Надпись	Caption	Объемы перевозок
RefEdit	Name	refVar
Кнопка	Name	cmdSolve
	Caption	Выполнить

Прежде чем набирать код, убедитесь, что установлена ссылка на Solver.xlam в окне **References**, отображаемом на экране выбором команды **Tools | References**. Если Solver.xlam отсутствует в списке **Available References**, нажмите кнопку **Browse** и откройте Solver.xla из каталога C:\Program Files\Microsoft Office\Office14\Library\SOLVER. Необходимый код наберите соответственно в модулях формы и модуле рабочего листа **Лист1** (см. файл *5-Приложение Транспортная задача.xlsm* на компакт-диске).

На рис. 9.28 приведены результаты выполнения приложения "Транспортная задача" (как введенные вспомогательные формулы в ячейки рабочего листа, так и результат расчета).



	A	B	C	D	E	F	G	H	I	J
1	**Стоимость перевозок**			**Объемы вывозимой продукции**						
2		7	1	15						
3		3	5	15						
4	Объемы ввозимой продукци	10	20							
5										
6	**Объемы перевозок**	0	14,9999995	14,9999995						
7		10	5,0000005	15,0000005						
8		10	20	70,000002						
9	**Целевая функция (указатель ячейки должен нахоиться в D9)**			70,000002	**Решение найдено**					
10										

Рис. 9.28. Результат расчета приложения "Транспортная задача"

Примечание

Обратите внимание, как в программе использован метод `Evaluate` для преобразования выражения в значение. Этот прием избавил нас от необходимости добавления в код двух циклов для определения суммарных объемов ввозимой и вывозимой продукции либо ввода в ячейки рабочего листа ненужных дополнительных формул.

Примечание

Использование в программе функции `SolverReset()` до задания параметров поиска решения существенно. Устанавливая значения параметров поиска решения равными тем значениям, которые используются по умолчанию, функция `SolverReset()` тем самым очищает окно **Поиск решения** от всех предыдущих установок. Отсутствие функции `SolverReset()` приведет к добавлению к списку тех ограничений, которые задаются функциями `SolverAdd()` при каждом нажатии кнопки **Выполнить**. Легко представить, что в этом случае произойдет со списком **Ограничения** окна **Поиск решения**, например, через десяток нажатий кнопки **Выполнить**!

Решение оптимизационных задач, зависящих от параметра

Программирование поиска решения может помочь и существенно ускорить процесс обработки данных, когда требуется проанализировать зависимость оптимального решения от параметра. Покажем это на примере простейшей задачи, которая аналогична рассмотренной ранее задаче нелинейного программирования. Предположим, что нам требуется найти решение системы нелинейных уравнений:

$$\begin{cases} x^2 + y^2 - 1 = 0, \\ 2x + 3y - d = 0, \end{cases}$$

где параметр d изменяется в интервале от 0,2 до 1 с шагом 0,1. Очевидно, что решить данную систему равносильно нахождению решения уравнения:

$$x^2 + y^2 - 1^{\ 2} + \ 2x + 3y - d^{\ 2} = 0.$$

Первоначальная система уравнений, конечно, не решается средством **Поиск решения**, а вот равносильное ей уравнение с двумя неизвестными подходит для того, чтобы попытаться его решить.

Итак, на рабочем листе отведите ячейки **A1** и **B1** под неизвестные, ячейку **D1** под значения параметра, а в ячейку **C1** введите формулу левой части уравнения

```
=(A1^2+B1^2-1)^2+(2*A1+3*B1-D1)^2
```

Расположите также на рабочем листе кнопку, которая будет открывать форму нашего приложения. Теперь перейдем к конструированию интерфейса приложения. Создайте форму с тремя надписями, тремя полями ввода и кнопкой (рис. 9.29). Установите значения свойств этих элементов управления с помощью окна **Properties**, как показано в табл. 9.13.

Таблица 9.13. *Значения свойств, установленные в окне* ***Properties***

Объект	Свойство	Значение
Форма	Name	frmSystemSolver
	Caption	Решение системы, зависящей от параметра
Надпись	Caption	Начальное значение
Поле	Name	txtBegin
Надпись	Caption	Конечное значение
Поле	Name	txtEnd
Надпись	Caption	Шаг изменения
Поле	Name	txtStep
Кнопка	Name	cmdOK
	Caption	OK

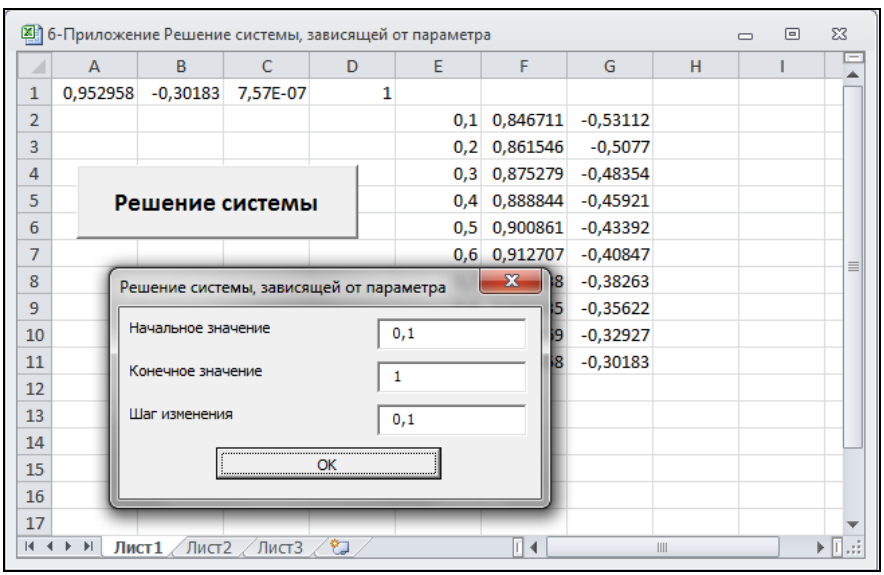


Рис. 9.29. Окно **Решение системы, зависящей от параметра**

Прежде чем набирать необходимый код в модуле формы и в модуле рабочего листа **Лист1** (см. также файл *6-Приложение Решение системы, зависящей от параметра.xlsm* на компакт-диске), убедитесь, что установлена ссылка на Solver.xlam в окне **References**.

Итак, для решения задачи достаточно в поля ввода ввести начальное, конечное значения параметра и шаг его изменения. После нажатия кнопки **OK** приложение найдет решения и выведет в столбец **E** значения параметра, а в столбцы **F** и **G** — значения неизвестных *x* и *y*.

Примечание

С геометрической точки зрения рассмотренная здесь задача также представляет собой нахождение точки пересечения прямой и окружности. Понятно, что у этой задачи либо нет решения (прямая не пересекается с окружностью), либо имеется только одно (прямая касается окружности), либо два решения (прямая пересекает окружность). Наша программа нашла только половину решений. Это связано с тем, что все решения мы находили с помощью одного и того же начального приближения, а именно нулевого. В качестве упражнения модернизируйте приложение так, чтобы оно находило все решения задачи.

Работаем со средством *Подбор параметра*

Средство MS Excel **Подбор параметра** позволяет определить значение одной входной ячейки, которое требуется для получения желаемого результата в зависимой ячейке (ячейке результата). Особенно эффективно применение данного средства при решении различных экономических задач. Рассмотрим несколько примеров использования средства **Подбор параметра** на рабочем листе Excel.

Пример определения затрат на проект

Пусть предполагается, что доходы по проекту в течение 5 лет составят: 120 млн руб., 200 млн руб., 300 млн руб., 250 млн руб., 320 млн руб. Надо определить первоначальные затраты на проект, чтобы обеспечить скорость оборота 12%.

Итак, расчет внутренней скорости оборота инвестиций производится с помощью функции (в ранних версиях — `ВНДОХ()`):

```
ВСД (Значения; Предположения)
```

Ввод исходных данных производится в соответствии с рис. 9.30 (см. также файл *7-Задача на Подбор параметра.xlsm* на компакт-диске).

Первоначально для расчета выбирается произвольное число — затраты на проект (ячейку с величиной данной суммы можно оставить также пустой) и производятся вычисления. В ячейку **B14** вводится формула:

```
=ВСД(B5:B10)
```

Далее, перейдите на вкладку **Данные** ленты и в группе **Работа с данными** выберите из списка **Анализ "что-если"** команду **Подбор параметра**. Установите в окне **Подбор параметра** значения в соответствии с рис. 9.31 и нажмите кнопку **OK**. Результаты подбора параметра выводятся в окне **Результат подбора параметра** (рис. 9.32).

C14 =ЕСЛИ(B14>B12;"Проект экономически целесообразен";"Проект необходимо отвергнуть")

	A	B	C	D	E	F
1						
2	**Расчет внутренней скорости оборота инвестиций**					
3						
4	Ожидаемые доходы в течение	5	лет			
5	Затраты по проекту	-700 000 000,00р.				
6	Первый год	120 000 000,00р.				
7	Второй год	200 000 000,00р.				
8	Третий год	300 000 000,00р.				
9	Четвертый год	250 000 000,00р.				
10	Пятый год	320 000 000,00р.				
11						
12	Рыночная норма дохода	12%				
13						
14	Внутренняя скорость оборота инвестиций	18%	Проект экономически целесообразен			
15						
16						

Рис. 9.30. Рабочий лист для определения первоначальных затрат по проекту

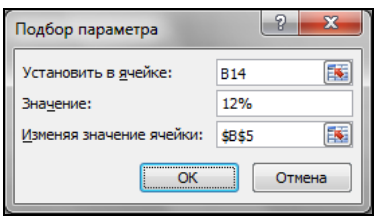


Рис. 9.31. Окно **Подбор параметра**

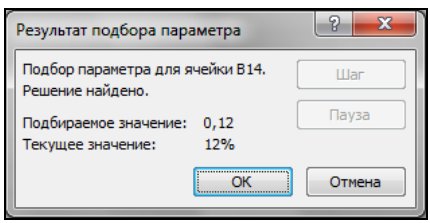


Рис. 9.32. Окно **Результат подбора параметра**

Обратите внимание на то, какие формулы размещены на рабочем листе (рис. 9.33). Поэтому и первоначальные, и итоговые расчеты позволяют получить не только итоговые цифры, но и текстовое резюме об экономической целесообразности проекта.

C14 =ЕСЛИ(B14>B12;"Проект экономически целесообразен";"Проект необходимо отвергнуть")

	A	B	C	D	E	F
4	Ожидаемые доходы в течение	5	лет			
5	Затраты по проекту	-820 389 165,92р.				
6	Первый год	120 000 000,00р.				
7	Второй год	200 000 000,00р.				
8	Третий год	300 000 000,00р.				
9	Четвертый год	250 000 000,00р.				
10	Пятый год	320 000 000,00р.				
11						
12	Рыночная норма дохода	12%				
13						
14	Внутренняя скорость оборота инвестиций	12%	Проект экономически целесообразен			
15						
16						

Рис. 9.33. Рассчитанная величина первоначальных затрат по проекту

Нахождение корней уравнения

Удобным средством отыскания корней уравнений является MS Excel. *Общие рекомендации по нахождению корней уравнений произвольной степени* можно сформулировать следующим образом.

1. Произвести табулирование заданной функции на некотором интервале с целью выявления (локализации) корней уравнения (перемена знака в значении функции). Иногда следует использовать табуляцию неоднократно для более точных оценок.
2. После локализации корней установить предельное число итераций и погрешность для вычисления корней (перейдите на вкладку **Файл** ленты и выберите команду **Параметры**; в открывшемся окне **Параметры Excel** выберите слева категорию **Формулы**, а справа — группу **Параметры вычислений**).
3. Непосредственное вычисление корней уравнения с использованием средства **Подбор параметра** (перейдите на вкладку **Данные** ленты и в группе **Работа с данными** выберите из списка **Анализ "что-если"** команду **Подбор параметра**).

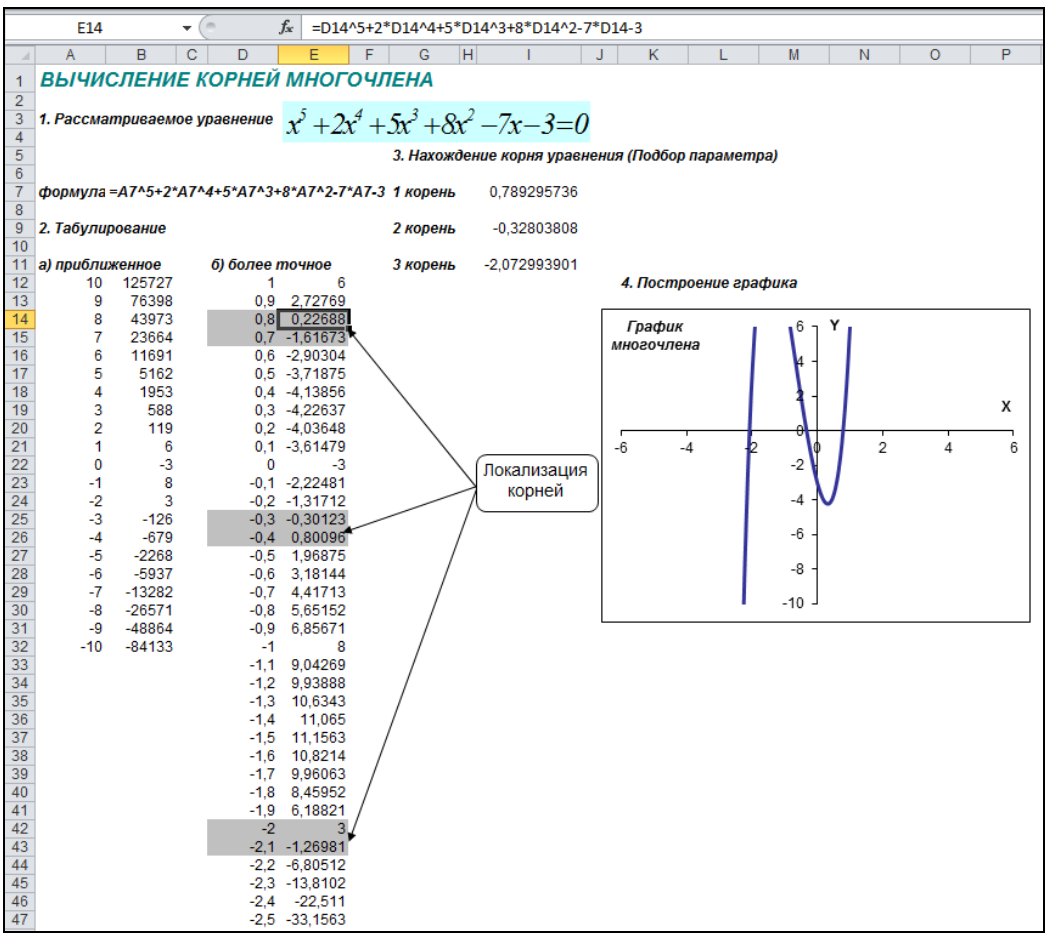


Рис. 9.34. Вычисление корней многочлена

4. Построение для наглядности графика исследуемой функции (выберите из коллекции требуемый **График** в группе **Диаграммы** на вкладке **Вставка** ленты).

Рассмотрим пример. Пусть требуется найти все корни уравнения:

$$x^5 + 2x^4 + 5x^3 + 8x^2 - 7x - 3 = 0 .$$

Решение данного уравнения приведено на рис. 9.34 (см. также файл *8-Решение уравнения с использованием Подбора параметра.xlsm* на компакт-диске).

1. Приближенное табулирование функции

$$f(x) = x^5 + 2x^4 + 5x^3 + 8x^2 - 7x - 3$$

на отрезке [–10; 10]. В ячейки **A12:A32** вводим аргумент функции — значение отрезка [–10; 10] с шагом 1. В ячейках **B12:B32** вычисляем значение функции $f(x)$. Формула для ячейки **B12**:

```
=A12^5+2*A12^4+5*A12^3+8*A12^2-7*A12-3
```

2. Для табулированной функции можно построить график.
3. По результатам вычисления определяем, что значение функции $f(x)$ меняет знак на отрезке [–3; 1]. Проводим более точное табулирование функции на данном отрезке. В ячейки **D12:D52** вводим аргумент функции $f(x)$ — значение отрезка [–3; 1] с шагом 0,1. В ячейках вычисляем значение функции $f(x)$. Формула для ячейки **E12** аналогична формуле для ячейки **B12**:

```
=D12^5+2*D12^4+5*D12^3+8*D12^2-7*D12-3
```

4. Результаты точного табулирования функции дают 3 изменения знака на отрезке [–3; 1], что свидетельствует о наличии корней уравнения $f(x) = 0$.
5. С помощью средства **Подбор параметра** определяем первый корень уравнения. Поместите указатель в ячейку **E14** (либо **E15**), далее перейдите на вкладку **Данные** ленты и в группе **Работа с данными** выберите из списка **Анализ "что-если"** команду **Подбор параметра** (рис. 9.35). Это средство дает первый корень уравнения:

$$x_1 = 0,789295735548989 .$$

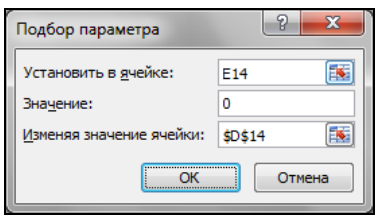


Рис. 9.35. Нахождение корня уравнения с использованием средства **Подбор параметра**

6. Аналогично вычисляем оставшиеся два корня:

$$x_2 = -0,328038079539342,$$

$$x_3 = -2,07299390058983.$$

Подбор параметра и решение уравнения с одним неизвестным с использованием VBA

Метод `GoalSeek` объекта `Range` подбирает значение параметра (неизвестной величины), являющееся решением уравнения с одним неизвестным. Предполагается, что уравнение приведено к виду: правая часть уравнения является постоянной, не зависящей от параметра, параметр входит только в левую часть уравнения, например:

$$x^3 - 3x - 5 = 0.$$

Метод `GoalSeek` программирует **Подбор параметра**. Этот метод вычисляет корень, используя метод последовательных приближений, результат выполнения которого, вообще говоря, зависит от начального приближения. Поэтому для корректности нахождения корня надо позаботиться о корректном указании этого начального приближения.

expression.GoalSeek(*Goal*, *ChangingCell*)

- *expression* — ячейка, в которую введена формула, являющаяся правой частью решаемого уравнения. В этой формуле роль параметра (неизвестной величины) играет ссылка на ячейку, указанную в аргументе *ChangingCell*.
- *Goal* — обязательный параметр, задающий значение левой части решаемого уравнения, не содержащей параметра.
- *ChangingCell* — обязательный параметр, указывающий ссылку на ячейку, отведенную под параметр (неизвестную величину). Значение, введенное в данную ячейку до активизации метода `GoalSeek`, рассматривается как начальное приближение к искомому корню. Значение, возвращаемое в эту ячейку после выполнения метода `GoalSeek`, является найденным приближением к корню.

Точность, с которой находится корень и предельно допустимое число итераций, используемых для нахождения корня, устанавливается свойствами `MaxChange` и `MaxIterations` объекта `Application`. Например, определение корня с точностью до 0,0001 максимум за 1000 итераций устанавливается инструкцией:

```
With Application
   .MaxIterations = 1000
   .MaxChange = 0.0001
End With
```

Метод `GoalSeek` возвращает значение `True`, если решение найдено, и значение `False` — в противном случае.

Например, код из листинга 9.1 ищет корень уравнения $x^3 - 3x - 5 = 0$ при начальном приближении 1 (рис. 9.36, см. также файл *9-Решение уравнения.xlsm* на компакт-диске).

Листинг 9.1, *а*. Решение уравнения. Стандартный модуль

```
Sub DemoGoalSeek()
   Range("A1").Name = "x"
   Range("A1").Value = 1
   Range("B1").Formula = "=x^3-3*x-5"
   If Range("B1").GoalSeek(Goal:=0, ChangingCell:=Range("x")) Then
      MsgBox "Корень: " & Range("A1").Value
   Else
      MsgBox "Корень не найден"
   End If
End Sub
```

Листинг 9.1, *б*. Решение уравнения. Модуль рабочего листа

```
Private Sub CommandButton1_Click()
  DemoGoalSeek
End Sub
```

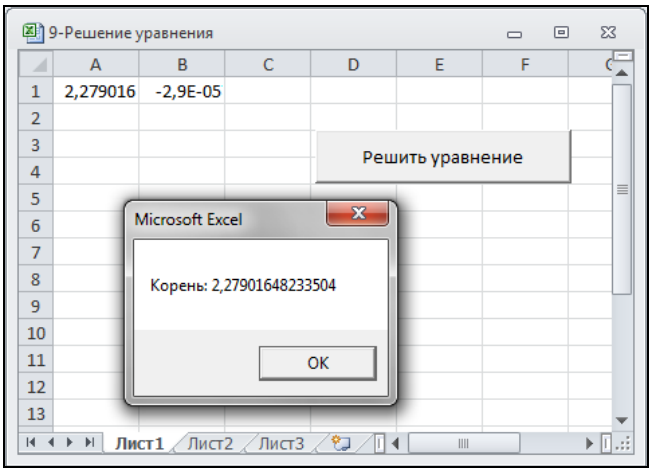


Рис. 9.36. Решение уравнения

Усовершенствование средства *Подбор параметра*

Средство MS Excel **Подбор параметра**, как указывалось ранее, позволяет определить значение одной входной ячейки, которое требуется для получения желаемого результата в зависимой ячейке (ячейке результата). **Подбор параметра** можно использовать для различных целей, например, данное средство позволяет на рабочем листе найти корень уравнения. Однако использование подбора параметра при решении уравнения требует выполнить относительно большой объем работы. Рассмотрим основной алгоритм работы со средством **Подбор параметра** в данном конкретном случае.

1. Представьте уравнение в таком виде, чтобы неизвестное входило только в его левую часть, а правая часть была постоянной. Например, уравнение $x^2 = x + 1$ надо привести к виду $x^2 - x - 1 = 0$.
2. Перейдите на вкладку **Файл** ленты и выберите команду **Параметры**. В открывшемся окне **Параметры Excel** выберите слева категорию **Формулы**, а справа в группе **Параметры вычислений** в поле **Относительная погрешность** укажите погрешность, с которой будет находиться корень уравнения. Например, 10^{-5}.
3. На рабочем листе отведите одну ячейку, например **B1**, под неизвестное. Введите в эту ячейку начальное приближение к корню, например 1.
4. Выберите на рабочем листе другую ячейку, например **B2**, под левую часть уравнения. Введите в эту ячейку формулу левой части уравнения. В рассматриваемом случае — формулу `=B1^2-B1-1`.

5. Перейдите на вкладку **Данные** ленты и в группе **Работа с данными** выберите из выпадающего списка **Анализ "что-если"** команду **Подбор параметра**.
6. В открывшемся диалоговом окне **Подбор параметра** в поле **Установить в ячейке** дать ссылку на ячейку, отведенную под левую часть уравнения. В нашем конкретном случае, **B2**. В поле **Значение** введите число, стоящее в правой части уравнения. В данном случае надо ввести `0`. В поле **Изменяя значение ячейки** дать ссылку на ячейку, отведенную под неизвестное, в рассматриваемом примере **B1**. Нажать кнопку **OK**.
7. На экране отобразится диалоговое окно **Результат подбора параметра**, а в ячейке **B1** вместо начального приближения будет находиться найденное значение корня. В данном случае 1,618037.

Итак, использование средства **Подбор параметра** требует довольно большого объема работы. Создадим приложение, которое будет облегчать работу пользователя при решении уравнения.

Перейдем непосредственно к конструированию приложения. Создайте форму с четырьмя надписями, четырьмя полями ввода, счетчиком и кнопкой (рис. 9.37). Установите значения свойства `Name` элементов управления, как показано в табл. 9.14. В модуле формы и в модуле рабочего листа **Лист1** наберите приводимый код (см. файл *10-Усовершенствование средства Подбор параметра.xlsm* на компакт-диске).

Таблица 9.14. *Значения свойства* `Name`*, установленные в окне* ***Properties***

Элемент управления	**Значение свойства `Name`**	**Описание**
Поле ввода	`txtBegin`	Пользователь вводит начальное приближение к корню
Надпись	`lblBegin`	Пояснительная надпись для поля ввода `txtBegin`
Поле ввода	`txtEquation`	Пользователь вводит левую часть уравнения. Уравнение должно быть приведено к виду, когда его правая часть равна 0. Вводимая левая часть уравнения должна начинаться со знака равенства, записываться по правилам языка программирования и вместо неизвестного должен быть использован символ "x". Например, `=x^2-x-1` В программе введенное выражение будет преобразовано в формулу рабочего листа
Надпись	`lblEquation`	Пояснительная надпись для поля ввода `txtEquation`
Поле ввода	`txtRoot`	Выводится найденное значение корня. Поле недоступно для пользователя
Надпись	`lblRoot`	Пояснительная надпись для поля ввода `txtRoot`
Поле ввода	`txtTol`	Вводится относительная точность нахождения корня при помощи счетчика
Надпись	`lblTol`	Пояснительная надпись для поля ввода `txtTol`
Счетчик	`spnTol`	Задает относительную точность нахождения корня, которая выводится в поле `txtTol`

Таблица 9.14 *(окончание)*

Элемент управления	Значение свойства `Name`	Описание
Кнопка	`cmdOK`	Считывает начальное приближение, левую часть уравнения и точность нахождения корня. Присваивает ячейке **B1** активного рабочего листа имя "x". Проверяет, преобразуется ли введенная левая часть уравнения в формулу рабочего листа. Если нет, то на экране отображается сообщение, и выполнение программы прерывается. Если данные введены корректно, то начальное приближение вводится в ячейку **B1** (ее имя теперь "x") рабочего листа. Формула левой части уравнения вводится в ячейку **B2**. При помощи метода **Подбор параметра** (метод `GoalSeek` диапазона) ищется корень. Если корень не найден, отображается сообщение. Если корень найден, то он предварительно форматируется с учетом введенной точности нахождения, а затем выводится в поле `txtRoot`

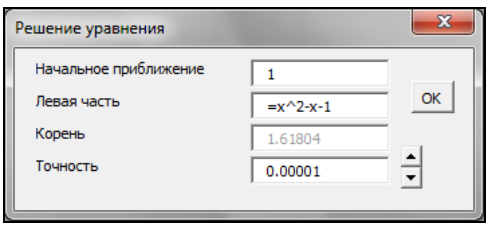


Рис. 9.37. Окно **Решение уравнения**

Наши итоги

Несомненно, разобранные примеры и использование средств MS Excel **Подбор параметра** и **Поиск решений** позволят вам автоматизировать многие задачи, связанные с построением оптимизационных моделей, осуществить их быстрое решение для заданных входных параметров, а также применять данные инструменты и других различных ситуациях. Материал данной главы научил вас:

- формализации задач линейного программирования в общем случае;
- организовывать рабочий лист Excel для работы с различными оптимизационными задачами;
- использовать надстройку **Поиск решения** для получения результатов некоторых задач оптимизации;
- применять функции, программирующие **Поиск решения**;
- работать со средством **Подбор параметра**;
- использовать метод `GoalSeek` объекта `Range`;
- реализовывать приложение, упрощающее работу со средством **Подбор параметра**.

Глава 10

Интеграция Microsoft Office Excel и XML

Как указывалось ранее, начиная с пакета Microsoft Office 2007, введен новый формат файлов — формат Microsoft Office Open XML. Наиболее отличительной его особенностью является использование новой концепции интерфейса — ленты (ribbon), которая заменила традиционные меню и панели инструментов и направлена на наиболее эффективное и удобное достижение намеченной цели. Кроме того, изменен стандартный формат хранения рабочей книги из двоичного формата, принятого в прежних версиях, на формат, основанный на XML.

Расширенный язык разметки (Extensible Markup Language, XML) был утвержден в 1996 году Консорциумом W3C (World Wide Web Consortium) как *метаязык*, предназначенный для описания языков разметки. XML является улучшенным вариантом языка HTML. Как правило, язык HTML используют для платформонезависимого представления данных, например, для отображения данных в браузерах, а язык XML — для платформо-независимого хранения и передачи данных в Интернете, однако использование браузера для этого не является обязательным.

Как правило, в языках разметки имеются *теги* для структурирования данных. Однако если в HTML имеется фиксированный набор тегов для описания элементов данных, то в XML теги вообще отсутствуют. Вместо этого XML позволяет разработчику самому создавать такой язык разметки, который в точности соответствует требованиям конкретного приложения.

Более подробная информация, связанная с XML, представлена на сайте консорциума W3C (World Wide Web Consortium) по адресу **http://www.w3.org/**.

> ***Примечание***
>
> Файлы рассмотренных в данной главе примеров можно найти в папке Glava_10 на компакт-диске.

Что необходимо знать о формате XML?

XML (Extensible Markup Language) — это расширяемый язык разметки, разработанный на базе SGML (Standard Generalized Markup Language — стандартный обобщенный язык разметки был утвержден международной организацией по стандартизации International Standards Organization (ISO) в качестве стандарта ISO 8879:1986 в 1986 г.) как формат универсального представления данных. Этот формат позволяет совершенно различным приложениям обмениваться данными через

Интернет. Основу XML-документов по аналогии с SGML- или, например, с HTML-документами составляют теги (или маркеры) — пометки в документе, которые можно определить по угловым скобкам, например, <p>, <h1>. Теги представляют собой коды, которые используются для определения структуры документа, его визуального оформления и смысла данных. Так, в HTML-документах теги служат для определения оформления данных, в документах XML — для определения структуры и смысла данных.

Следует отметить, что XML предоставляет более расширенные возможности по сравнению, например, с языком HTML, т. к. позволяет создавать дополнительные элементы, с помощью которых можно описывать и определять новые данные, объекты и их свойства, отделяя данные от представления их в виде HTML, благодаря чему удается преодолеть ограничения HTML в возможности описания нестандартных объектов. XML официально принят консорциумом W3C (World Wide Web Consortium), который занимается стандартами, относящимися к World Wide Web.

Язык XML является платформо-независимым: любая программа, которая предполагает использование XML, может считывать и обрабатывать XML-данные независимо от операционной системы или аппаратных средств. Благодаря универсальному формату представления, с XML-данными можно работать во многих приложениях Microsoft Office 2007.

При работе с документами в XML-формате можно указать следующие файлы:

- веб-страница для просмотра полученного документа (например, в формате HTML);
- файл XSL, содержащий описание структуры внешнего представления документа;
- файл XML, включающий в себя данные, являющиеся источником для заполнения документа;
- файл XSD, содержащий описание структуры данных, обычно его называют *схемой данных*. Схема данных может не выделяться в отдельный файл, а добавляться прямо в файл XML, который содержит сами данные.

Такая структура обусловлена логическим разделением документа XML на отдельные части: данные (XML), структуру данных (XSD) и представление данных (XSL, например, преобразование в формат HTML). Концептуально все эти файлы вместе образуют единый веб-документ, который можно просмотреть с помощью обозревателя Internet Explorer 5 и выше. Однако файл XML (возможно, вместе с файлом XSD) может быть использован приложениями, распознающими этот формат данных, независимо от других частей документа. Так, например, файл в формате HTML содержит лишь сценарий, который активизируется при загрузке страницы и загружает данные на страницу из источника в формате XML.

Отделение самих данных от их представления (в формате HTML) и помещение их в отдельный файл в формате XML открывает возможность другим приложениям, воспринимающим этот специально разработанный универсальный формат, получать и обрабатывать данные из такого документа независимо от их представления. Отделение данных от их представления позволяет приложениям также применять различные способы для отображения одних и тех же данных XML с помощью нескольких различных схем представления данных.

Файл XSD называется *схемой XML*. Его содержание удовлетворяет стандарту XML Schema Definition (XSD), официально принятому консорциумом W3C. Файл схемы XML описывает структуру данных в универсальном виде, включая информацию о названиях элементов, типах данных, комбинациях элементов, а также об атрибутах элементов. Схема XML определяет модель представления данных в формате XML: задает правила для тегов и текста. Применение схемы XML гарантирует правильное восприятие данных в формате XML другими приложениями и корректное преобразование этих данных в другие форматы данных.

Схема XML содержит описание данных, но не содержит описание того, как они должны быть отображены в программе просмотра. Ранее для реализации отображения данных в HTML использовались файлы CSS, включавшие соответствующую информацию о представлении данных на языке описания стилей Cascading Style Sheet. Однако это не слишком удобно, т. к. разработчику пришлось бы изучать еще и язык CSS в дополнение к XML, к тому же CSS предоставляет недостаточно средств для контроля над выводом данных. В современных приложениях чаще применяется более гибкое средство для описания внешнего представления данных XSL — язык XSL (Extensible Stylesheet Language). Он позволяет точно выбрать данные, которые требуется отобразить, задать порядок расположения элементов данных, модифицировать и добавить дополнительную информацию. Кроме того, этот язык похож на XML: в нем для создания шаблона стиля вывода данных используются теги, подобные тегам в XML, и конструкции HTML. Заметим, что для отображения данных XML в Internet Explorer 5 или выше не обязательно присоединять файлы CSS или XSL, т. к. эта программа просмотра имеет собственное описание стиля, применяющееся по умолчанию. Используйте собственные файлы описания стилей, чтобы обеспечивать единообразный внешний вид ваших веб-страниц, основанных на данных XML. Непосредственно в XSL можно выделить следующие части спецификации: язык преобразования стилей XSL for Transformation (XSLT) и язык для верстки XML XSL-FO (XSL Formatting Objects), т. е. унифицированный язык представления, который сохраняет все данные документа внутри себя.

Изучаем синтаксис XML

Основные компоненты документа XML

По аналогии с документом HTML, в документе XML также имеются теги. Основными компонентами документа XML являются *элементы* (elements), *атрибуты* (attributes) и *комментарии* (comments).

Элементы используются для разметки частей (секций) документа XML и имеют следующий синтаксис:

```
<Element> Content </Element>
```

Здесь `<Element>` — открывающий тег (start tag), `</Element>` — закрывающий тег (end tag), а `Content` — содержание (значение) элемента. Например:

```
<name> Walkenbach </name>
```

Содержание относится к символьным данным, а элементы — к разметке документа. В свою очередь, символьные данные подразделяются на проверяемые

символьные данные (Parsed Character Data, PCDATA) и не проверяемые (Unparsed Character Data).

Элементы могут не иметь содержания. Например:

```
<cellphone></cellphone>
```

В этом случае их можно объединять в один тег `<cellphone/>`.

Также элементы могут быть вложены в другие элементы:

```
<employee>
  <name> Walkenbach </name>
  <salary> 10000 </salary>
</employee>
```

У элементов могут быть атрибуты, которые служат для задания дополнительной информации для элемента и укорачивают код. Атрибут — это пара *имя=значение*, расположенная в открывающем теге. Например, `currency` является атрибутом тега `<salary>`:

```
<salary currency="USD"> 10000 </salary>
```

Или же, элемент `<person>` можно было бы с помощью атрибутов записать следующим образом (листинг 10.1).

Листинг 10.1. Использование атрибутов в XML-коде

```
<employee>
   <person lastname="Garnaev" firstname="Andrej"
           email="garnaev@yandex.ru"/>
   <person lastname="Rudikova" firstname="Lada"
           email="rudikowa@gmail.com"/>
</employee>
```

Кроме того, атрибуты позволяют разбивать элементы на категории. Например, в следующем коде (листинг 10.2, см. также файл *1-Example.xml* на компакт-диске) в зависимости от значения атрибута `type` в элементе `<person>` содержится либо конфиденциальная, либо общедоступная информация.

Листинг 10.2. Разбивка элементов по категориям при помощи атрибутов

```
<?xml version="1.0" standalone="yes" ?>
<employee>
   <person type="work">
      <lastname>Bond</lastname>
      <firstname>James</firstname>
      <email>bond007@yandex.com</email>
   </person>
   <person type="work">
      <lastname>Cooper</lastname>
      <firstname>Gary</firstname>
      <email>gcooper@yandex.com</email>
```

```
   </person>
   <person type="personal">
      <lastname>Cooper</lastname>
      <firstname>Gary</firstname>
      <marriedstatus>new married</marriedstatus>
      <homephone>354-56-56</homephone>
   </person>
</employee>
```

Комментарий в языке XML задается следующим образом:

```
<!--  Пример комментария  -->
```

Структура документа XML

Документ XML состоит из *пролога* (prolog) и *корневого элемента* (root element), включающего все остальные элементы. Пролог содержит информацию о номере используемой в документе версии XML и, как правило, информацию о кодировке символов. Часто пролог также содержит информацию о декларации одиночного документа и наличии или отсутствии ссылок на внешний файл разметки, который может оказать влияние непосредственно на редактируемый XML-файл. Так, значение `"yes"` в декларации одиночного документа говорит об отсутствии внешних деклараций разметки, которые оказывали бы влияние на информацию, которую XML-процессор передает приложению. Например, пролог с декларированием одиночного документа выглядит так:

```
<?xml version="1.0" encoding="Windows-1251" standalone="yes"?>
```

Таким образом, простейший документ XML может выглядеть так, как представлено в листинге 10.3 (см. также файл *2-Example.xml* на компакт-диске).

Листинг 10.3. Пример простейшего XML-документа

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--  Пример документа XML  -->
<employee>
   <person lastname="Walkenbach" firstname="John"
           email="johnw@yandex.com"/>
   <person lastname="Wiley" firstname="Gary"
           email="gwiley@yandex.com"/>
</employee>
```

Как правило, XML-код можно набирать в тестовом редакторе, сохраняя его с расширением xml. Так, например, листинг 10.3 можно набрать в Блокноте и затем открыть его в браузере Internet Explorer (рис. 10.1).

Следует отметить, что XML-документ, открытый в браузере, может быть просмотрен поэлементно. Так, если щелкнуть на знаке "минус" слева от элемента `<employee>`, вложенные элементы будут скрыты (рис. 10.2).

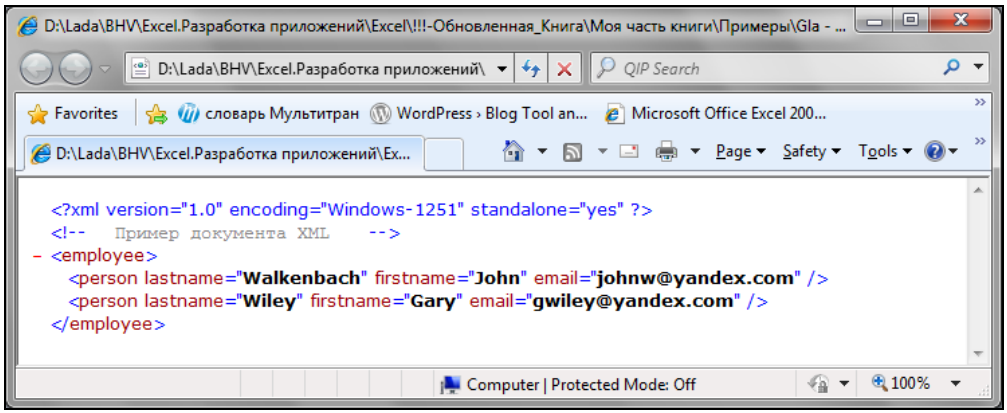


Рис. 10.1. Простейший XML-документ, открытый в браузере Internet Explorer

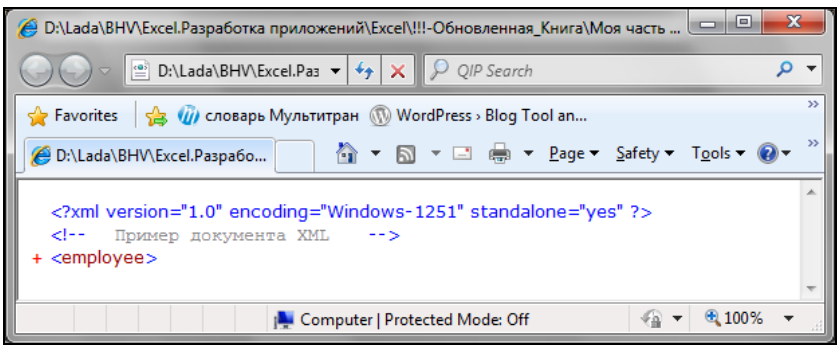


Рис. 10.2. Скрытие элементов в документе XML

Русификация XML

Для русификации XML при объявлении документа следует указать атрибут `encoding`, задающий соответствующую кодировку (листинг 10.4, см. также файл *3-Example.xml* на компакт-диске).

Листинг 10.4. Русификация XML

```
<?xml version="1.0" encoding="Windows-1251"?>
<employee>
   <person firstname="Иван" email="petrov@gmail.com"> Петров </person>
   <salary currency="р."> 100000 </salary>
</employee>
```

Зачем нужны схемы XML?

Как указывалось ранее, при работе с XML-документами необходимо иметь *схему данных*, описывающих их структуру. На практике обычно пролог содержит *схему XML* (XML schema), описывающую, какие элементы может содержать

использующий эту схему документ, какие атрибуты соответствуют каким элементам и т. п. Если проводить аналогии с базой данных, то схема XML напоминает описание атрибутов и типов данных для таблиц в базе. Для описания схем существует специальный язык — XSD (XML Schema Definition Language, язык описания схем XML).

Процесс сопоставления содержимого XML-документа на соответствие некой XML-схеме называется *проверкой* или *валидацией* (validation).

Непосредственно схема может размещаться в документе, но чаще она находится в отдельном файле с расширением xsd, а в сам XML-документ помещается ссылка на этот файл.

> ***Примечание***
>
> Схемы в документе может не быть вообще: ни внутри, ни в виде ссылки. В таком случае проверка документа на соответствие некоторой схеме проводится либо "вручную", либо программным путем.

Пространства имен

В одном XML-документе может использоваться несколько схем. В таком случае возникает проблема совпадения имен: в разных схемах могут быть заданы одинаковые имена, и если некоторый документ ссылается на две схемы, в каждой из которых одно и то же имя элемента определяется по-своему, то возникает вопрос: какой из двух вариантов задействован для конкретного имени в документе?

Чтобы решить указанную проблему, вводят понятие *пространства имен* (namespace). При указании имени всегда можно определить соответствующее ему пространство имен. Само пространство имен также обязано иметь *уникальное имя* (префикс). Для указания префикса можно использовать URL (Uniform Resource Locator, универсальный указатель ресурса).

Пространство имен задается внутри открывающего тега элемента:

```
<namespacePrefix:elementName xmlns:namespacePrefix = "URL">
```

Используемый URL не обязательно должен указывать на реальный файл, т. к. главная задача URL — обеспечение уникальности.

Документ может использовать несколько пространств имен, одно из которых может не иметь имени (листинг 10.5, см. также файл *4-Example.xml* на компакт-диске). В таком случае оно называется *пространством имен по умолчанию* (default namespace).

Листинг 10.5. Использование пространства имен по умолчанию

```
<?xml version="1.0" encoding="Windows-1251" ?>
<!--  Использование пространства имен по умолчанию -->
<employee xmlns = "http://www.myorg.ru/staff">
  <name> Петров </name>
  <salary currency="р."> 100000 </salary>
</employee>
```

Отметим, что схема также является документом XML и должна удовлетворять следующим требованиям:

- все схемы должны иметь элемент верхнего уровня с именем `schema`;
- все схемы должны использовать одно и то же базовое *пространство имен* (namespace), URL которого имеет вид: **http://www.w3.org/2001/XMLSchema**. Кроме базового пространства имен в схеме могут использоваться также и дополнительные пространства имен.

Например, схему XML с базовым пространством имен `bn` можно определить так, как представлено в листинге 10.6 (см. также файл *5-Schema.xsd* на компакт-диске).

Листинг 10.6. Пример схемы XML

```
<?xml version="1.0" encoding="Windows-1251" ?>
<bn:schema xmlns:bn="http://www.w3.org/2001/XMLSchema">
   <bn:element name="employee">
      <bn:complexType>
         <bn:sequence>
            <bn:element name="name"   type="bn:string"/>
            <bn:element name="salary" type="bn:integer"/>
         </bn:sequence>
      </bn:complexType>
   </bn:element>
</bn:schema>
```

Схема XML, расположенная в документе

Приведенную в листинге 10.6 схему XML можно вставить непосредственным образом в XML-документ (листинг 10.7).

Листинг 10.7. Пример использования схемы в XML-документе

```
<?xml version="1.0" encoding="Windows-1251" ?>
<employees>
  <!— Начало схемы -->
<bn:schema xmlns:bn="http://www.w3.org/2001/XMLSchema">
   <bn:element name="employee">
      <bn:complexType>
         <bn:sequence>
            <bn:element name="name"   type="bn:string"/>
            <bn:element name="salary" type="bn:integer"/>
         </bn:sequence>
      </bn:complexType>
   </bn:element>
</bn:schema>
  <!—Конец схемы -->
   <employee>
```

```
      <name> Петров </name>
      <salary>10000</salary>
   </employee>
   <employee>
      <name> Сидоров </name>
      <salary>15000</salary>
   </employee>
</employees>
```

Внешняя схема XML

Итак, в предыдущем разделе мы рассмотрели использование схемы внутри XML-документа. Однако наиболее оптимальным считается использование *внешней схемы*, хранящейся в отдельном файле.

Наберите код из листинга 10.6 в текстовом редакторе, например в Блокноте, и сохраните его под именем *5-Schema.xsd*.

Чтобы указать в документе, что для его проверки будет использована схема, находящаяся в файле *5-Schema.xsd*, требуется указать имя этого файла в специальном атрибуте (из пространства имен **http://www.w3.org/2001/XMLSchema-instance**).

В случае, когда документ ссылается на какие-либо пространства имен (кроме указанного выше), используется атрибут `schemaLocation`, в противном случае — `noNamespaceSchemaLocation` (листинг 10.8, см. также файл *5-Example.xml* на компакт-диске).

Листинг 10.8. XML-документ со ссылкой на схему 5-Schema.xsd

```
<?xml version="1.0" encoding="Windows-1251" ?>
<!--  Использование внешней схемы XML  -->
<employee  xmlns:bni="http://www.w3.org/2001/XMLSchema-instance"
    bni:schemaLocation="employee 5-Schema.xsd">
      <name> Петров </name>
      <salary>10000</salary>
</employee>
```

Рассмотрим еще один пример XML-схемы, который сохраним под именем *6-Schema.xsd*, и позволяющий использовать список из нескольких записей (листинг 10.9, см. также файл *6-Schema.xsd* на компакт-диске). Соответствующий этой схеме документ XML приведен в листинге 10.10 (см. файл *6-Example.xml* на компакт-диске).

Листинг 10.9. Пример схемы XML для верификации списка из нескольких записей

```
<?xml version="1.0" encoding="Windows-1251" ?>
<bn:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <bn:element name="employees">
  <bn:complexType>
   <bn:sequence>
      <bn:element ref="employee" maxOccurs="unbounded"/>
```

```
   </bn:sequence>
  </bn:complexType>
 </bn:element>
    <bn:element name="employee">
     <bn:complexType>
      <bn:sequence>
        <bn:element name="firstname"   type="bn:string"/>
        <bn:element name="lastname"   type="bn:string"/>
        <bn:element name="salary" type="bn:integer"/>
      </bn:sequence>
     </bn:complexType>
    </bn:element>
</bn:schema>
```

Листинг 10.10. XML-документ со списком записей по нескольким сотрудникам

```
<?xml version="1.0" encoding="Windows-1251" ?>
<!--  Пример использования схемы XML  -->
<employees   xmlns:bni="http://www.w3.org/2001/XMLSchema-instance">
             bni:schemaLocation="employee 6-Schema.xsd"
   <employee>
      <firstname> Иван </firstname>
      <lastname> Петров </lastname>
      <salary> 10000 </salary>
   </employee>
   <employee>
      <firstname> Дмитрий </firstname>
      <lastname> Федоров </lastname>
      <salary> 9000 </salary>
   </employee>
   <employee>
      <firstname> Анна </firstname>
      <lastname> Котова </lastname>
      <salary> 15000 </salary>
   </employee>
</employees>
```

Экспортируем и импортируем данные XML в рабочую книгу Excel

Как выполнить импорт данных XML в Excel?

Если в XML-документе имеется схема XML, то при импорте данных из такого документа Excel может использовать информацию из связанной с XML-документом схемы и хранить ее в *картах XML* соответствующей рабочей книги,

куда импортированы данные из исходного документа. Когда в исходном документе схема отсутствует, Excel пытается создать карту XML самостоятельно — на основе анализа данных, которые содержатся в исходном документе.

Импорт данных из XML-файла в случае отсутствия схемы XML

Для импорта данных из XML-файла:

1. Перейдите на вкладку **Файл** ленты и выберите команду **Открыть**.
2. В открывшемся окне **Открытие документа** нажмите кнопку со списком **Все файлы Excel** и выберите тип **Файлы XML (*.xml)**. После этого в области списка файлов будут отображены только файлы этого типа.
3. Выберите в области списка файлов необходимый файл и нажмите кнопку **Открыть**.
4. В появившемся диалоговом окне **Открытие XML** (рис. 10.3) выберите переключатель **XML-таблица** и нажмите кнопку **OK**.

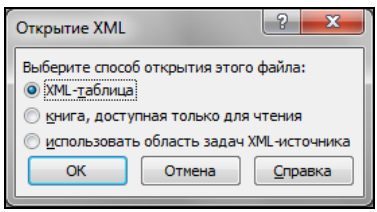


Рис. 10.3. Диалоговое окно **Открытие XML**

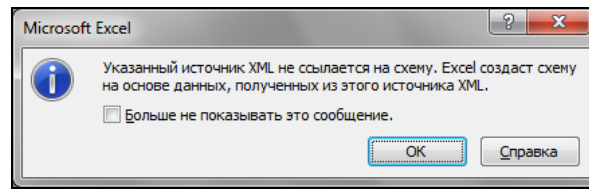


Рис. 10.4. Предупреждение Excel о создании схемы данных на базе XML-файла

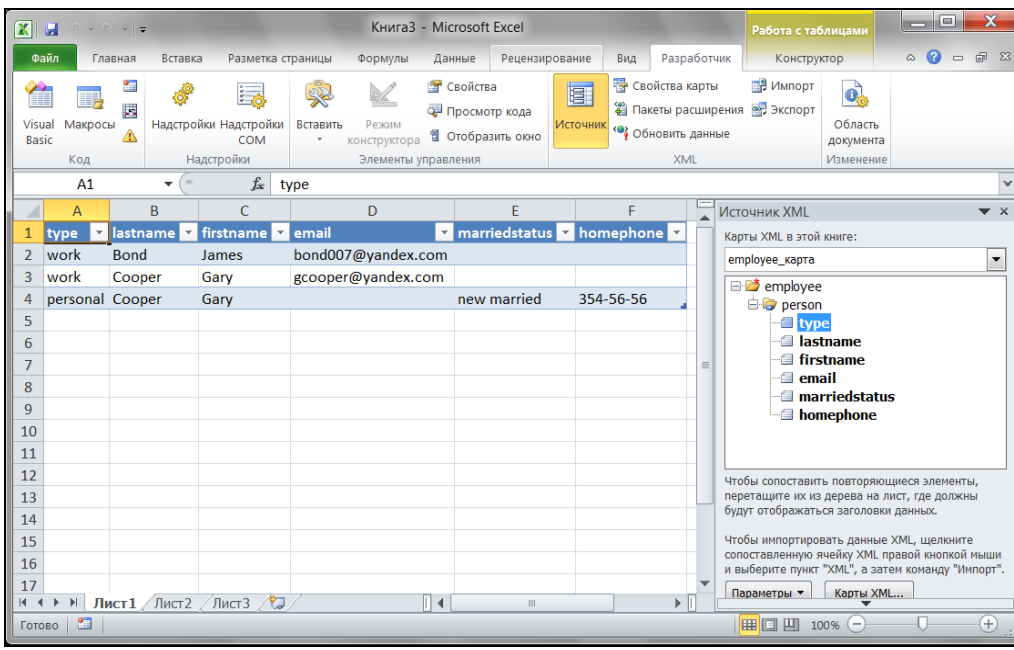


Рис. 10.5. Результаты импорта документа XML в рабочую книгу Excel

5. Если открываемый файл не содержит схемы данных, появится предупреждающее сообщение о том, что Excel создаст схему (рис. 10.4).
6. Нажмите кнопку **OK** — данные из XML-файла будут импортированы на рабочий лист Excel.
7. Чтобы просмотреть карту XML, которая создана Excel, перейдите на вкладку **Разработчик** ленты и в группе команд **XML** щелкните по кнопке **Источник**: в области задач, расположенной в правой части, откроется панель **Источник XML** с созданной картой XML (рис. 10.5).

Создание карты XML и импорт данных из файла XML

Итак, пусть у нас имеется некоторая схема XML (см., например, листинг 10.9). Для создания карты XML в MS Excel выполните следующие действия.

1. Откройте файл Excel (или создайте новый), в который необходимо импортировать данные из документа XML.
2. Перейдите на вкладку **Разработчик** ленты и в группе команд **XML** щелкните по кнопке **Источник**: в области задач, расположенной в правой части, откроется панель **Источник XML** (рис. 10.6).
3. Нажмите кнопку **Карты XML** в правом нижнем углу панели **Источник XML** — откроется диалоговое окно **Карты XML**.
4. В окне **Карты XML** нажмите кнопку **Добавить** для открытия диалогового окна **Выберите источник XML** (рис. 10.7, выберите также файл *6-Example.xsd* на компакт-диске).

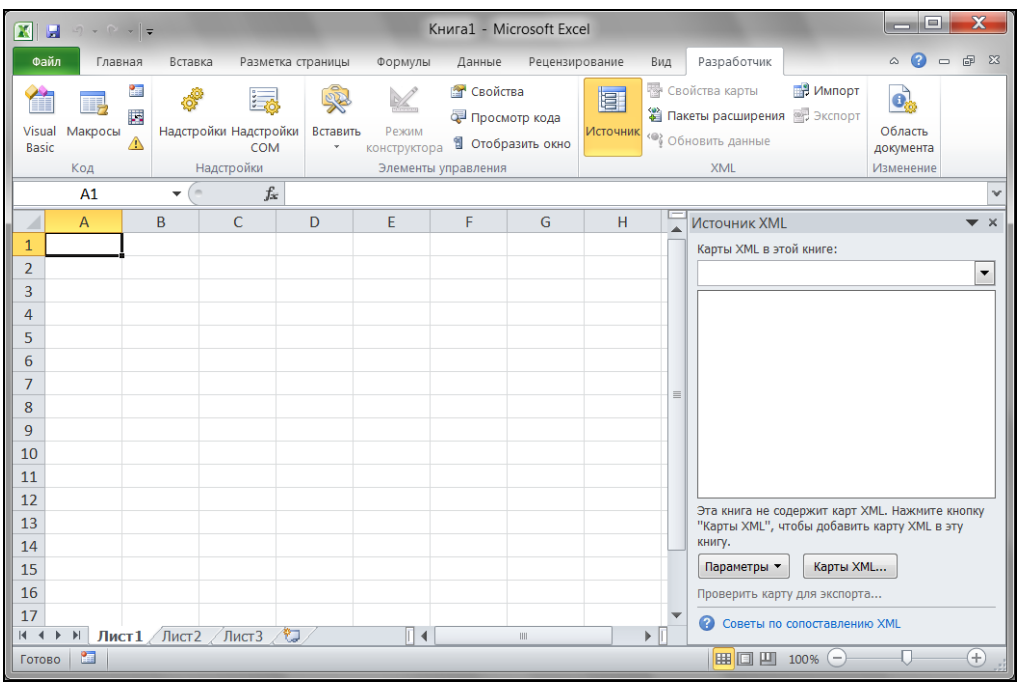


Рис. 10.6. Вкладка **Разработчик** и панель **Источник XML**

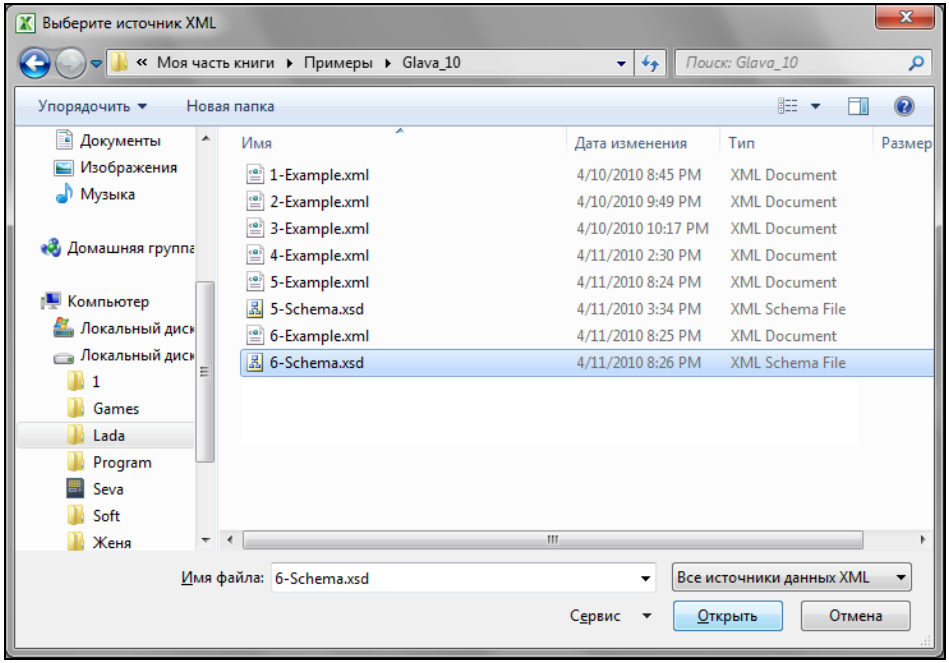


Рис. 10.7. Диалоговое окно выбора источника для XML-карты

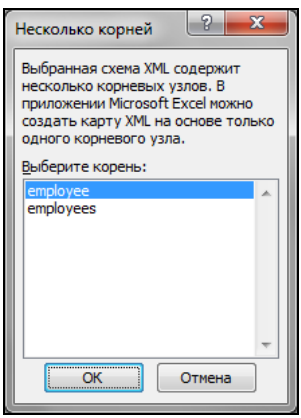


Рис. 10.8. Диалоговое окно выбора корневого узла для XML-карты

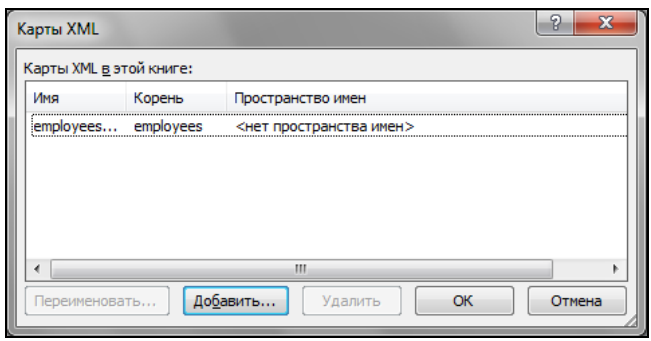


Рис. 10.9. Диалоговое окно **Карты XML**

5. Выберите файл для создания карты и нажмите кнопку **Открыть**.
6. Если в выбранном источнике содержится несколько корневых узлов, появится диалоговое окно выбора корневого узла для XML-карты (рис. 10.8).
7. Выберите подходящий вариант из списка и нажмите кнопку **OK**. Строка с параметрами добавляемой карты появится в окне **Карты XML** (рис. 10.9).

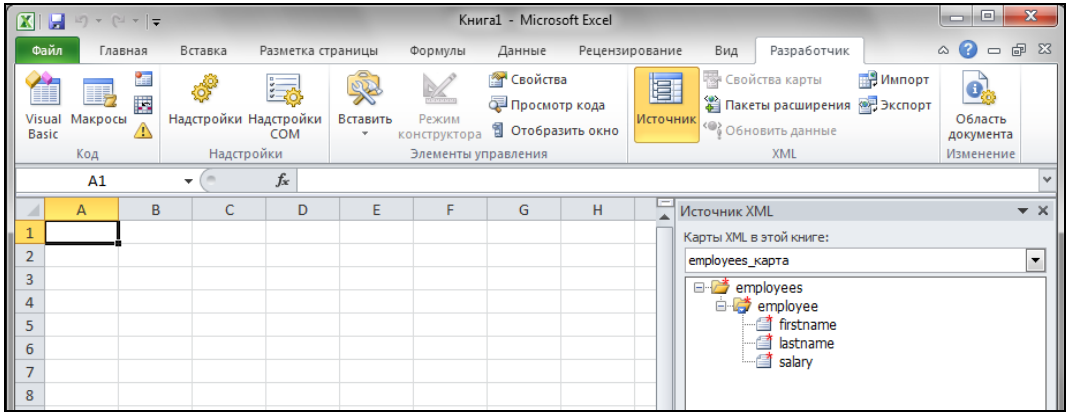


Рис. 10.10. Результат добавления карты XML в рабочую книгу Excel

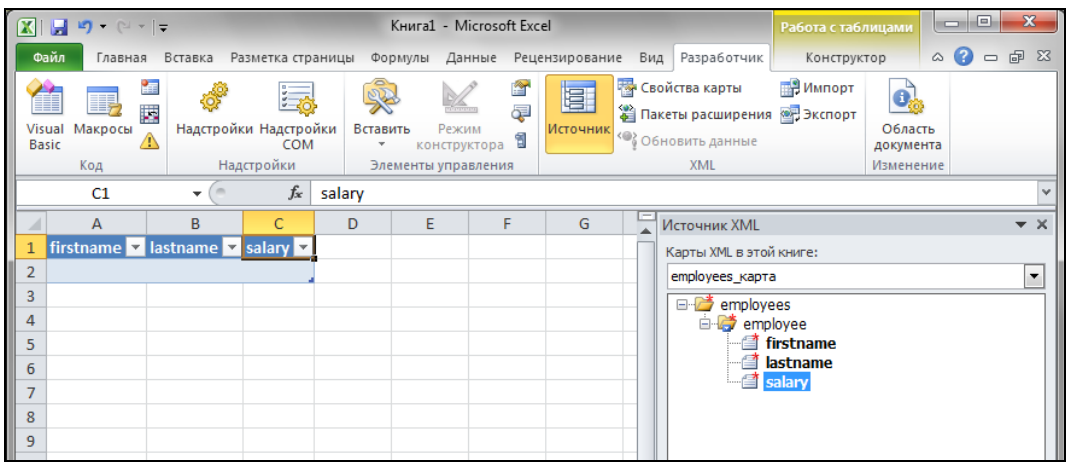


Рис. 10.11. Рабочий лист Excel с шаблоном для импорта данных из документа XML

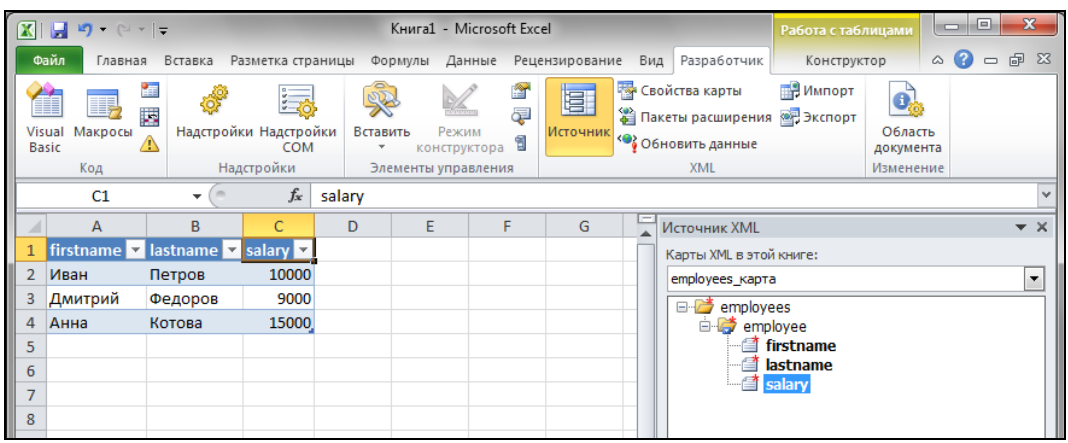


Рис. 10.12. Рабочий лист Excel со всеми данными

8. Выделите имя добавленной схемы в списке диалогового окна **Карты XML** и нажмите кнопку **OK**. Карта добавится в рабочую книгу и будет отображена на панели **Источник XML** в рабочей области Excel (рис. 10.10).
   Для того чтобы воспользоваться добавленной картой XML:
9. Перетащите с помощью мыши нужные элементы с панели **Источник XML** на рабочий лист для указания того, какие поля отображать на рабочем листе (рис. 10.11).
10. Перейдите на вкладку **Разработчик** ленты и в группе команд **XML** щелкните по кнопке **Импорт**. Далее, после указания необходимого файла, данные будут импортированы на рабочий лист Excel (рис. 10.12, см. также файл *7-Импорт XML-данных в рабочую книгу.xlsx* на компакт-диске).

Как выполнить экспорт данных из Excel в документ XML?

Экспорт данных с рабочего листа Excel в документ XML аналогичен импорту, однако действия выполняются в обратном порядке. Следует также помнить, что выполнить экспорт данных в XML-документ без соответствующей схемы невозможно. Итак, выполните следующие действия.

1. Откройте файл Microsoft Excel, который содержит также карту XML (см. файл *8-Файл для экспорта.xlsx* на компакт-диске).

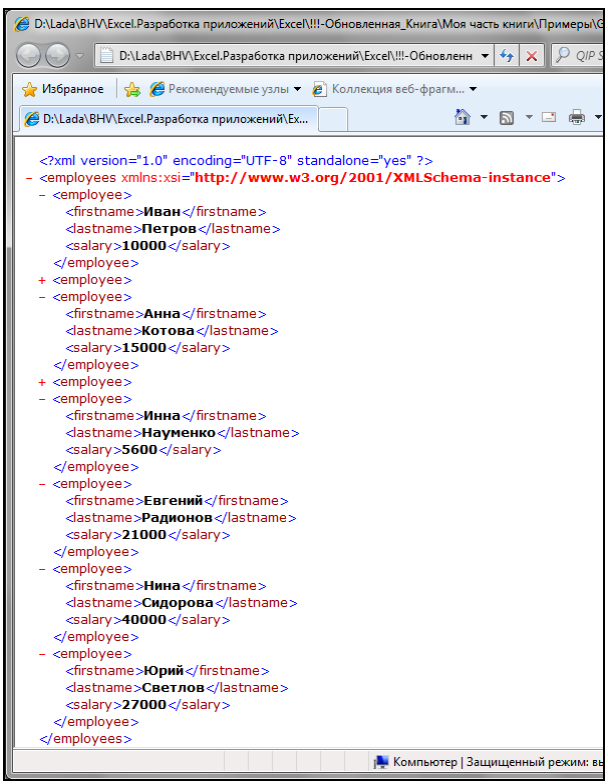


Рис. 10.13. Документ XML, полученный в результате экспорта данных из Excel

2. Перейдите на вкладку **Разработчик** ленты и выберите в группе **XML** команду **Экспорт**.
3. В открывшемся окне **Экспорт XML** выберите расположение экспортируемых данных и в поле **Имя файла** введите имя для сохраняемого файла XML. Нажмите кнопку **Экспорт**.
4. Убедитесь, что созданный вами XML-файл имеет кодировку русских букв (UTF-8) и его можно просмотреть в браузере (рис. 10.13, см. также файл *9-Экспорт данных из книги Excel.xml* на компакт-диске).

Как выполнить импорт и экспорт с помощью VBA?

Если вы хотите выполнить импорт или экспорт с использованием VBA, необходимо воспользоваться соответствующим методом.

Для импорта XML-данных в рабочую книгу Excel (листинг 10.11) используется метод `XmlImport`:

```
expression.XmlImport(Url, ImportMap, Overwrite, Destination)
```

- *expression* — ссылка на объект — рабочую книгу.
- *Url* — URL-ссылка или полный путь к файлу с XML-данными.
- *ImportMap* — карта, используемая при импорте файла; если данные были ранее импортированы, то содержит ссылку на объект, хранящий карту XML-данных.
- *Overwrite* — определяет, следует ли перезаписывать данные, которые были сопоставлены карте, заданной в параметре *ImportMap*; для перезаписи данных параметру *Overwrite* необходимо присвоить значение `True`; для добавления новых данных в существующие — значение `False`; по умолчанию данному параметру присваивается значение `True`.
- *Destination* — указывает диапазон расположения импортируемых данных; параметр содержит ссылку на левую верхнюю ячейку диапазона.

Листинг 10.11. Импорт XML-файла

```
Sub Imp()
    ActiveWorkbook.XmlImport URL:="D:\Data.xml", _
        ImportMap:=Nothing, Overwrite:=True, Destination:=Range("$A$1")
End Sub
```

Для экспорта данных, находящихся в книге MS Excel и содержащих карту (листинг 10.12), используется метод `ExportXml`:

```
expression.ExportXml(Data)
```

- *expression* — ссылка на объект — рабочую книгу, содержащую карту XML.
- *Data* — указание полного пути к экспортируемому файлу.

Листинг 10.12. Экспорт данных в XML-файл

```
Sub Exp()
    ActiveWorkbook.XmlMaps("employees_map").Export URL:="D:\Export.xml"
End Sub
```

Наши итоги

Прочитав эту главу, вы познакомились с основами языка XML и взаимодействием Microsoft Office Excel и XML. Теперь более понятным для вас стал и Ribbon-код, с помощью которого вы в *главе 5* производили настройку пользовательского интерфейса Excel. Итак, вы:

- ❒ узнали, что представляет собой формат XML и для чего он необходим;
- ❒ изучили основные компоненты документа XML, структуру документа XML;
- ❒ познакомились со схемами XML и узнали, где они могут находиться;
- ❒ научились импортировать данные XML в рабочую книгу Excel;
- ❒ научились производить экспорт данных с рабочих листов Excel в формат XML.

Глава 11

MS Excel и Интернет — рядом!

В настоящее время всемирная сеть Интернет охватывает, практически, все аспекты промышленной, научной, культурной, социальной и иной деятельности людей. Использование новых технологий в представлении, передаче и обработке данных, интеграция и анализ больших массивов данных, информационная поддержка программных средств — вот основные направления, которые сегодня характеризуют формирование международного информационного пространства Интернета. Естественно, что взаимодействие офисных приложений пакета Microsoft со Всемирной сетью получает с каждой своей новой версией дальнейшее развитие и расширение тех или иных возможностей. Корпорация Microsoft предоставляет пользователям различные стороны взаимодействия с мировым информационным пространством: публикация данных в Интернете, настройка соответствующего интерфейса приложения, тесная интеграция со всеми приложениями Microsoft Office, поддержка формата XML, использование узла SharePoint и многое другое.

Примечание

Файлы рассмотренных в данной главе примеров можно найти в папке Glava_11 на компакт-диске.

Что нужно знать об Интернете?

Удобный интерфейс Всемирной паутины (World Wide Web, WWW) связывает большое количество ресурсов, имеющихся в Интернете. Используя WWW, можно перемещаться между тысячами компьютерных узлов, системными приложениями, файлами и документами. Простота перемещения между документами и возможность читать их с помощью любой компьютерной системы позволили внедрить веб-технологию в организациях. Многие корпорации и предприятия разрабатывают собственные сети на основе технологий Интернета (интрасети), чтобы размещать в них внутреннюю информацию, предназначенную только для сотрудников.

Итак, обращаясь к ресурсам, находящимся на других компьютерах, мы имеем дело с ресурсами компьютерной сети. Компьютерная сеть представляет собой два или более компьютера, объединенных общим каналом передачи данных. По территориально-организационным признакам компьютерные сети подразделяются на:

- *локальные сети* (Local Area Network, LAN) — охватывают организацию, группу организаций либо район и используют единый высокоскоростной канал передачи данных;

- *глобальные сети* (Wide Area Network, WAN) — распространяют свое действие по всему миру и используют все возможные каналы связи (включая, например, спутниковые).

Обычно локальную сеть называют *интранетом* (или *интрасетью*), в то время как Всемирная глобальная сеть получила название *Интернет*. Как правило, архитектурным принципом построения сетей является принцип "клиент — сервер".

В настоящее время Интернет представляет собой виртуальное пространство, состоящее из программного обеспечения, сетей различного уровня, компьютеров и терминалов (для ввода и отображения данных), которое постоянно растет и обновляется, отвечая новым потребностям современного общества. Множество локальных вычислительных сетей, входящих в Интернете, связаны между собой высокоскоростными каналами связи на континентальном уровне.

Учитывая огромное количество сетей, образующих Интернет, для того чтобы попасть в нужное место, необходимо знать употребляемые форматы адресов. Номера, которые используются для идентификации компьютера в Интернете, называются *IP-адресами*. У каждого компьютера в Интернете есть свой уникальный IP-адрес, состоящий из комбинации четырех групп цифр, каждая из которых не превышает 255 в десятичной записи. Поскольку IP-адрес не очень удобен для пользователей, каждый компьютер в Интернете имеет еще и DNS-адрес (Domain Name Service, доменная служба имен), например, **www.domainname.org**. Такое имя называется *доменным*.

Для доступа к определенному виду ресурсов, имеющихся в Интернете, например, для просмотра информации, которая размещается на странице, необходимо ввести адрес этой страницы в Интернете. Этот адрес называется *унифицированным указателем ресурсов* или *URL* (Uniform Resource Locator). В зависимости от того, каким образом необходимо получить доступ к документу (через локальный диск, локальную сеть, веб-узел или файловый архив), URL может выглядеть по-разному (даже для одного и того же документа). URL состоит из двух частей: спецификатора протокола для доступа к данному ресурсу и спецификатора расположения самого ресурса. Например:

- **file://c:\sales\sales.htm** — файл на локальном компьютере;
- **file://brig\sales\sales.htm** — файл на компьютере в локальной сети;
- **http://brig/sales/sales.htm** — файл на веб-сервере в интрасети;
- **http://brig.boreas.ru/sales/sales.htm** — файл на удаленном веб-сервере в Интернете;
- **ftp://brig.boreas.ru/sales/sales.htm** — файл на удаленном FTP-сервере в Интернете.

Если конкретный файл в адресе URL не указан, то открывается веб-страница, установленная по умолчанию для данного веб-сервера.

Термин *"веб-сервер"* (веб-узел) может трактоваться несколькими способами. С одной стороны, это набор документов, связанных гиперссылками (при этом у веб-сервера есть основная страница, через которую за один или несколько шагов доступны все другие страницы). С другой стороны, термин "веб-сервер" может означать компьютер, на котором размещен набор документов, доступный через локальную или глобальную сеть. Наконец, последнее значение этого термина — программное обеспечение, предназначенное для доступа к набору документов через локальную или глобальную сеть. Везде в данной главе, где это особо не оговорено, мы будем иметь в виду первое значение термина "веб-сервер".

Веб-страница (или страница Интернета, или документ в формате HTML) — это текстовый файл, содержащий специальные команды разметки документа. При открытии веб-страницы в простом текстовом редакторе (например, в Блокноте) вы увидите именно эти команды. Однако будучи открытой с помощью программы просмотра Интернета, такой как Internet Explorer, Mosaic или Netscape, веб-страница может отображать текст, графику, гиперссылки на другие документы, а также элементы управления. Секрет в том, что программа просмотра веб-страниц содержит интерпретатор команд языка HTML, содержащихся в файле веб-страницы.

Язык HTML (Hypertext Markup Language, язык гипертекстовой разметки) является системой разметки документов для их дальнейшей публикации в сети World Wide Web. Документы, подготовленные в формате HTML, включают в себя рисунки и ссылки, а также команды форматирования. Для просмотра этих документов используется средство просмотра веб-страниц (например, программа Internet Explorer).

Гиперссылка — это текст, выделенный синим цветом или подчеркиванием (или иным определенным пользователем образом), либо графическое изображение. При щелчке по гиперссылке осуществляется переход к файлу, определенному месту в файле, странице HTML в World Wide Web или странице HTML в интрасети. Гиперссылки могут также указывать, например, на протокол эмуляции терминала Telnet, группы новостей (newsgroup) или узлы FTP. При переходе между страницами с помощью гиперссылок создается и сохраняется хронология просмотра всех страниц. Средства просмотра веб-страниц, подобные Internet Explorer, имеют на панелях инструментов кнопки перемещения, которые позволяют двигаться вперед или назад от одной просмотренной страницы к другой.

Публикация — это процесс вывода таблиц, форм и отчетов в статическом или динамическом формате HTML с последующей установкой всех связанных файлов в виде приложений World Wide Web на один из веб-серверов, например, на Microsoft Internet Information Server или Microsoft Personal Web Server.

Для просмотра информации, размещаемой в Интернете, используются специальные программы — браузеры (или *веб-обозреватели*). *Браузеры* представляют собой программы, которые обеспечивают доступ пользователям к информации, удобные средства для ее просмотра и создания собственных веб-страниц.

Один из известных браузеров, который используют как удобное и надежное средство навигации по ресурсам Интернета, — Microsoft Internet Explorer (рис. 11.1).

При помощи Internet Explorer можно просматривать не только страницы Интернета, но и работать с документами Microsoft Office Word, рабочими листами Excel, презентациями PowerPoint вне зависимости от того, был ли сохранен документ в виде веб-страницы или в стандартном для приложения формате. При открытии документов, сохраненных в стандартном для создавшего их приложения формате, в окне Internet Explorer появляются меню и панели инструментов соответствующего приложения, позволяющие редактировать документ прямо в Internet Explorer. Это стало возможным благодаря технологии ActiveX.

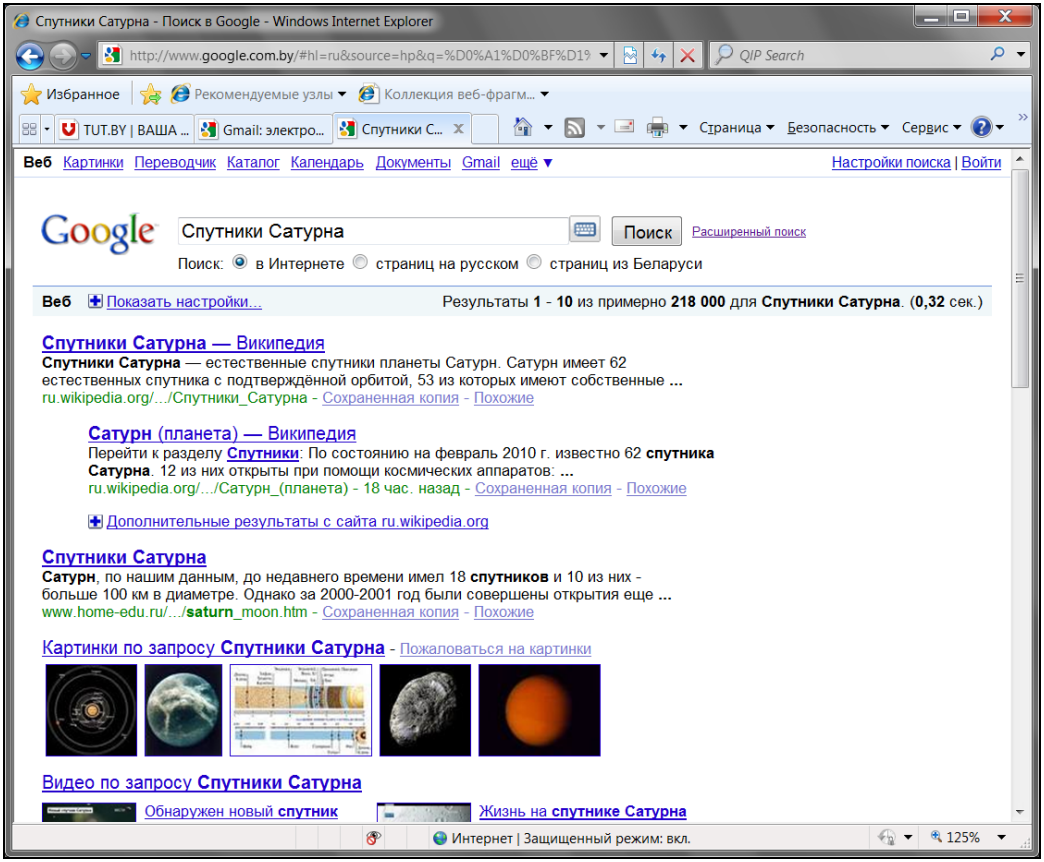


Рис. 11.1. Окно браузера Microsoft Internet Explorer

Отметим также основные службы Интернета:

- *WWW* (World Wide Web, Всемирная паутина) — средство для работы с гипертекстами, позволяющее извлекать и хранить разнотипную информацию (текстовую, графическую, видео-, аудио- и др.); гипертекстовые документы размещаются на веб-серверах, входящих в Интернет;
- *FTP* (File Transfer Protocol, протокол передачи файлов) — способ пересылки файлов между компьютерами сети независимо от их типов, особенностей операционных систем, файловых систем и форматов файлов;
- *E-mail* (Electronic Mail, электронная почта) — средство передачи и получения электронных сообщений между пользователями сети;
- *Usenet* (служба телеконференций) — средство рассылки электронных сообщений для пользователей сети (одно сообщение отправляется большой группе пользователей для публичного обсуждения);
- *IRC* (Internet Relay Chat, беседа через Интернет) — служба прямого общения в Интернете многих пользователей в реальном масштабе времени;
- *ICQ* (I seek you — "Я ищу тебя") — служба интерактивного общения для пользователей Всемирной сети, для которых не обязательно иметь постоянный IP-адрес.

Пользователь данной службы регистрируется на центральном сервере **www.icq.com** и получает персональный идентификационный номер UIN (Universal Internet Number, универсальный интернет-номер), по которому всегда можно установить связь с другими пользователями сети, использующими данную службу интерактивного общения.

В Microsoft Office объединены две мощные информационные технологии, определяющие модель работы с компьютером. Первая основана на возможности размещения информации в любом месте — на локальном жестком диске, в локальной или корпоративной сети или в Интернете. Другая — на том, что пользователи реально работают не с приложениями, а непосредственно с документами и содержащейся в них информацией. В результате можно выбрать один из двух возможных подходов к работе:

- работа преимущественно с приложениями Microsoft Office с эпизодическими обращениями к интрасети или Интернету за необходимой веб-страницей, документом, надстройкой для приложения или дополнительной информацией о программе;
- работа преимущественно внутри браузера Internet Explorer, использование его в качестве единственной среды, в которой можно просматривать и редактировать любой документ, расположенный на вашем жестком диске, в сети компании или в Интернете.

Работаем с гиперссылками в Microsoft Office Excel

Как добавить гиперссылки на документы MS Office?

Часто, работая с книгой Microsoft Excel, приходится использовать документы, которые подготовлены в различных приложениях MS Office. Например, добавляя гиперссылки на рабочие листы, можно перемещаться между этими документами одним щелчком мыши.

Для создания гиперссылки на другой документ MS Office (в том числе и на текущую рабочую книгу, но на другой диапазон ячеек или другой рабочий лист) на рабочем листе MS Excel:

1. Выберите диапазон или фигуру, с которой гиперссылка будет связана.
2. Перейдите на вкладку ленты **Вставка** и в группе команд **Ссылки** щелкните по кнопке **Гиперссылка**.
3. В открывшемся окне **Вставка гиперссылки** (рис. 11.2) установите необходимые опции. Так, переключатели **файлом, веб-страницей**, **местом в документе**, **новым документом**, **электронной почтой** группы **Связать с** указывают документ, на который будет дана ссылка в гиперссылке. Поле **Текст** задает текст гиперссылки. В поле **Папка** выбирается папка, в которой лежит искомый документ. В списках **текущая папка**, **просмотренные страницы**, **последние файлы** можно выбрать искомый файл. В раскрывающемся списке **Адрес** можно указать URL документа, кнопка **Подсказка** позволяет добавить текст к всплы-

вающей подсказке гиперссылки, а кнопка **Закладка** позволит выбрать конкретное место, в которое будет производиться переход по гиперссылке.

4. После установки необходимых параметров нажмите кнопку **OK**.

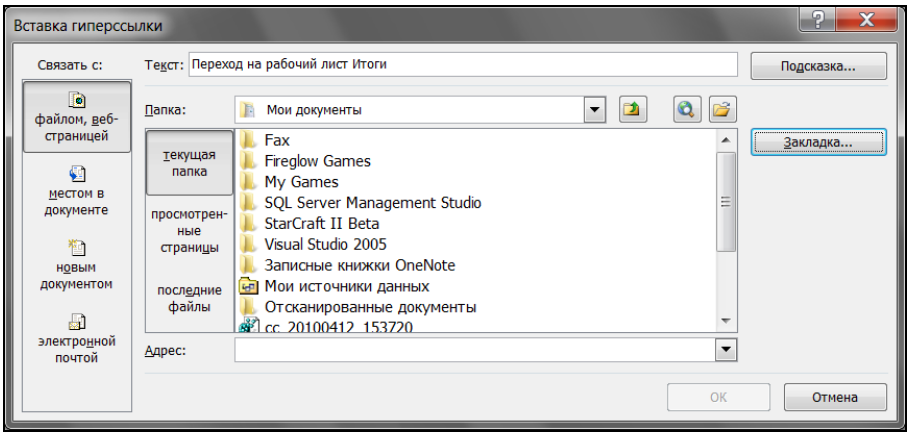


Рис. 11.2. Окно **Вставка гиперссылки**

Отредактировать гиперссылку можно, щелкнув на ней правой кнопкой мыши. На экране отобразится контекстное меню, которое предлагает три команды по работе с гиперссылками:

- **Изменить гиперссылку** — открывает окно **Изменение гиперссылки**. В нем можно изменить путь к связанному файлу и его имя, а также добавить пояснительное примечание к гиперссылке;
- **Открыть гиперссылку** — открывает связанный документ, как это делается при щелчке на гиперссылке;
- **Удалить гиперссылку** — удаляет гиперссылку.

Как задать гиперссылку формулой рабочего листа?

Гиперссылку можно создавать также функцией рабочего листа `ГИПЕРССЫЛКА()`: `ГИПЕРССЫЛКА(`*`адрес`*`; имя)`

- *`адрес`* — это путь и имя файла для открываемого документа. Адрес может ссылаться на место в документе, например, на определенную ячейку или именованный интервал на листе книги MS Excel или на закладку в документе MS Word. Путь может представлять собой путь к файлу, записанному на жестком диске, может также быть адресом в формате UNC сервера (в MS Excel для Windows) или адресом URL в Интернете или интрасети;
- *`имя`* — текст перехода или численное значение, отображаемое в ячейке. Имя отображается синим цветом и с линией подчеркивания. Если аргумент опущен, ячейка в качестве текста перехода отображает аргумент *`адрес`*.

> ***Примечание***
>
> В качестве значений параметров функции `ГИПЕРССЫЛКА()` могут быть либо текстовые выражения, либо ссылка на ячейку.

Так, если в книге Department.xlsx из ячейки, например **B3**, необходимо перейти в ячейку **A20** рабочего листа **Отчет** (кстати, ячейка **B3** может сама располагаться на рабочем листе **Отчет**, т. е. допустимо движение как внутри рабочего листа, так и между листами), то в ячейку **B3** надо ввести гиперссылку

```
=ГИПЕРССЫЛКА("[Department.xlsx]Отчет!A20";"Перейти в отчет")
```

либо формулу

```
=ГИПЕРССЫЛКА(A1;B1)
```

где в ячейку **A1** введено `[Department.xlsx]Отчет!A20`, а в ячейку **B1** — `Перейти в отчет`.

Первая из следующих двух гиперссылок открывает рабочую книгу Sales.xlsx, находящуюся в корневом каталоге диска D:, а вторая — открывает эту книгу и активизирует рабочий лист **Май**:

```
=ГИПЕРССЫЛКА("D:\Sales.xlsx";"Продажи")
=ГИПЕРССЫЛКА("[D:\Sales.xlsx]Май!A1";"Продажи")
```

Гиперссылку на русскоязычную веб-страницу корпорации Microsoft можно организовать, например, следующим образом:

```
=ГИПЕРССЫЛКА("http://www.microsoft.com/ru/ru/","Microsoft")
```

Что такое условная гиперссылка?

Чтобы гиперссылка включалась или выключалась в зависимости от какого-либо условия, используйте ее совместно с функцией `ЕСЛИ()`. Например:

```
=ЕСЛИ(C1="Май";ГИПЕРССЫЛКА("[D:\Sales.xlsx]Май!A1";"Продажи");"")
```

Автоматическое изменение гиперссылки в зависимости от значения в какой-либо ячейке легко реализуется при помощи функции `ЕСЛИ()` или `ВЫБОР()`. Выполните, например, следующее:

1. Создайте и сохраните на диске D: следующие файлы рабочих книг Microsoft Office Excel: May.xlsx, June.xlsx, July.xlsx (см. папку *1-Пример с функцикй ВЫБОР* на компакт-диске).
2. Откройте новую рабочую книгу и добавьте, например, в ячейку **A1** следующую формулу:

   ```
   =ВЫБОР(1; ГИПЕРССЫЛКА("D:\May.xlsx";"Май");
   ГИПЕРССЫЛКА("D:\June.xlsx";"Июнь"); ГИПЕРССЫЛКА("D:\July.xlsx";"Июль"))
   ```

3. Убедитесь, что в ячейке **A1** отображается гиперссылка **Май** (рис. 11.3), при щелчке по которой открывается соответствующий файл, расположенный на диске D:.
4. Изменив номер первого аргумента (индекса) в функции `ВЫБОР()` на 2 или 3, проверьте переход по гиперссылкам соответственно к файлу June.xlsx или July.xlsx.

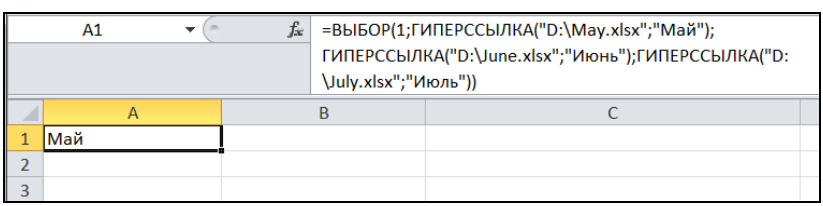


Рис. 11.3. Использование функции `ВЫБОР()` в ячейке рабочего листа

Объект *Hyperlink* и семейство *Hyperlinks*

Гиперссылки в VBA представлены объектом `Hyperlink`, который является элементом семейства `Hyperlinks`, состоящего из всех гиперссылок рабочего листа или диапазона. Это семейство имеет два метода: `Add` (добавить новую гиперссылку в семейство) и `Delete` (удалить все гиперссылки из семейства).

`Add(`*`Anchor`*`, `*`Address`*`, `*`SubAddress`*`, `*`ScreenTip`*`, `*`TextToDisplay`*`)`

- *`Anchor`* — задает местоположение гиперссылки. Может быть либо объектом `Range`, либо объектом `Shape`.
- *`Address`* — адрес гиперссылки.
- *`SubAddress`* — область в документе (например, диапазон ячеек или закладка), на которую происходит переход.
- *`ScreenTip`* — текст всплывающей подсказки.
- *`TextToDisplay`* — текст, отображаемый на гиперссылке.

В табл. 11.1 перечислены свойства, а табл. 11.2 — методы объекта `Hyperlink`.

Таблица 11.1. *Свойства объекта* `Hyperlink`

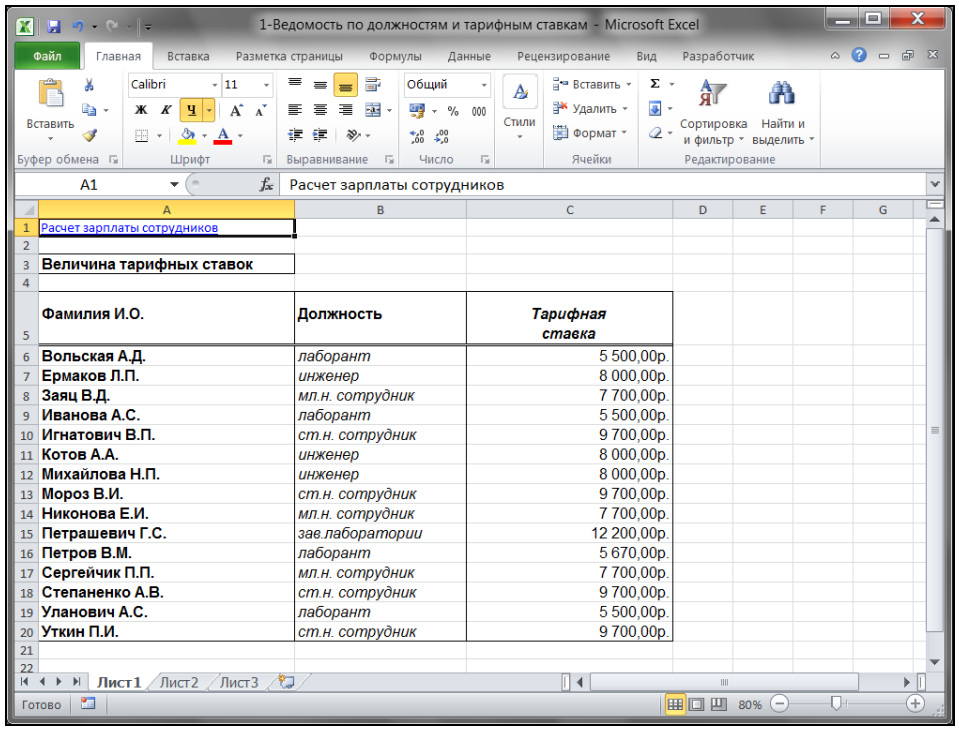
Свойство	Описание
`Address`	Адрес гиперссылки
`EmailSubject`	Передаваемая строка текста
`Range`	Возвращает объект `Range`, которому назначена гиперссылка
`ScreenTip`	Текст всплывающей подсказки
`Shape`	Возвращает объект `Shape`, которому назначена гиперссылка
`SubAddress`	Область в документе, на которую происходит переход
`TextToDisplay`	Текст, отображаемый на гиперссылке
`Type`	Возвращает объект, к которому подсоединена гиперссылка. Допустимые значения: `msoHyperlinkInlineShape`, `msoHyperlinkRange`, `msoHyperlinkShape`

Таблица 11.2. *Методы объекта* `Hyperlink`

Метод	Описание
`AddToFavorites`	Добавить в список избранных ссылок
`CreateNewDocument`	Создать новый документ, связанный с указанной гиперссылкой
`Delete`	Удалить гиперссылку
`Follow`	Перейти по специфицированной гиперссылке

Рассмотрим пример программного создания гиперссылки.

1. Подготовьте вначале два файла (сохраните их в формате с поддержкой макросов): *1-Ведомость по должностям и тарифным ставкам.xlsm* и 2-*Ведомость по зарплате.xlsm (*см. папку *2-Пример использования гиперссылки* на компакт-диске*)*.

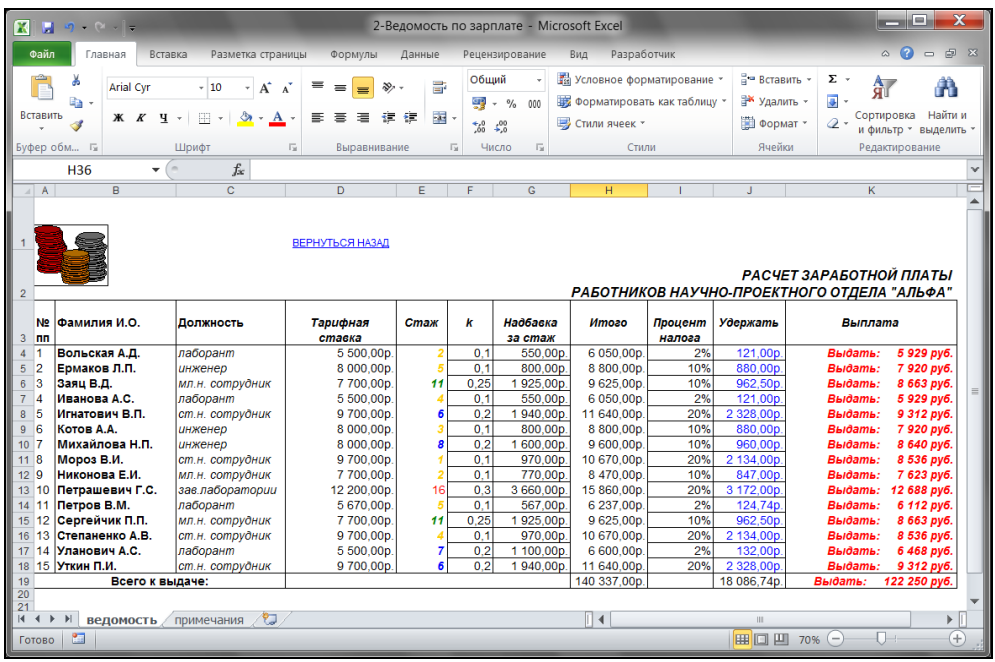


1-Ведомость по должностям и тарифным ставкам - Microsoft Excel

A1: Расчет зарплаты сотрудников

	A	B	C
1	Расчет зарплаты сотрудников		
2			
3	**Величина тарифных ставок**		
4			
5	**Фамилия И.О.**	**Должность**	***Тарифная ставка***
6	**Вольская А.Д.**	*лаборант*	5 500,00р.
7	**Ермаков Л.П.**	*инженер*	8 000,00р.
8	**Заяц В.Д.**	*мл.н. сотрудник*	7 700,00р.
9	**Иванова А.С.**	*лаборант*	5 500,00р.
10	**Игнатович В.П.**	*ст.н. сотрудник*	9 700,00р.
11	**Котов А.А.**	*инженер*	8 000,00р.
12	**Михайлова Н.П.**	*инженер*	8 000,00р.
13	**Мороз В.И.**	*ст.н. сотрудник*	9 700,00р.
14	**Никонова Е.И.**	*мл.н. сотрудник*	7 700,00р.
15	**Петрашевич Г.С.**	*зав.лаборатории*	12 200,00р.
16	**Петров В.М.**	*лаборант*	5 670,00р.
17	**Сергейчик П.П.**	*мл.н. сотрудник*	7 700,00р.
18	**Степаненко А.В.**	*ст.н. сотрудник*	9 700,00р.
19	**Уланович А.С.**	*лаборант*	5 500,00р.
20	**Уткин П.И.**	*ст.н. сотрудник*	9 700,00р.

Рис. 11.4. Рабочий лист файла *1-Ведомость по должностям и тарифным ставкам.xlsm*



2-Ведомость по зарплате - Microsoft Excel

ВЕРНУТЬСЯ НАЗАД

РАСЧЕТ ЗАРАБОТНОЙ ПЛАТЫ РАБОТНИКОВ НАУЧНО-ПРОЕКТНОГО ОТДЕЛА "АЛЬФА"

№ пп	Фамилия И.О.	Должность	Тарифная ставка	Стаж	k	Надбавка за стаж	Итого	Процент налога	Удержать	Выплата
1	**Вольская А.Д.**	*лаборант*	5 500,00р.	2	0,1	550,00р.	6 050,00р.	2%	121,00р.	Выдать: 5 929 руб.
2	**Ермаков Л.П.**	*инженер*	8 000,00р.	5	0,1	800,00р.	8 800,00р.	10%	880,00р.	Выдать: 7 920 руб.
3	**Заяц В.Д.**	*мл.н. сотрудник*	7 700,00р.	11	0,25	1 925,00р.	9 625,00р.	10%	962,50р.	Выдать: 8 663 руб.
4	**Иванова А.С.**	*лаборант*	5 500,00р.	4	0,1	550,00р.	6 050,00р.	2%	121,00р.	Выдать: 5 929 руб.
5	**Игнатович В.П.**	*ст.н. сотрудник*	9 700,00р.	6	0,2	1 940,00р.	11 640,00р.	20%	2 328,00р.	Выдать: 9 312 руб.
6	**Котов А.А.**	*инженер*	8 000,00р.	3	0,1	800,00р.	8 800,00р.	10%	880,00р.	Выдать: 7 920 руб.
7	**Михайлова Н.П.**	*инженер*	8 000,00р.	8	0,2	1 600,00р.	9 600,00р.	10%	960,00р.	Выдать: 8 640 руб.
8	**Мороз В.И.**	*ст.н. сотрудник*	9 700,00р.	1	0,1	970,00р.	10 670,00р.	20%	2 134,00р.	Выдать: 8 536 руб.
9	**Никонова Е.И.**	*мл.н. сотрудник*	7 700,00р.	2	0,1	770,00р.	8 470,00р.	10%	847,00р.	Выдать: 7 623 руб.
10	**Петрашевич Г.С.**	*зав.лаборатории*	12 200,00р.	16	0,3	3 660,00р.	15 860,00р.	20%	3 172,00р.	Выдать: 12 688 руб.
11	**Петров В.М.**	*лаборант*	5 670,00р.	5	0,1	567,00р.	6 237,00р.	2%	124,74р.	Выдать: 6 112 руб.
12	**Сергейчик П.П.**	*мл.н. сотрудник*	7 700,00р.	11	0,25	1 925,00р.	9 625,00р.	10%	962,50р.	Выдать: 8 663 руб.
13	**Степаненко А.В.**	*ст.н. сотрудник*	9 700,00р.	4	0,1	970,00р.	10 670,00р.	20%	2 134,00р.	Выдать: 8 536 руб.
14	**Уланович А.С.**	*лаборант*	5 500,00р.	7	0,2	1 100,00р.	6 600,00р.	2%	132,00р.	Выдать: 6 468 руб.
15	**Уткин П.И.**	*ст.н. сотрудник*	9 700,00р.	6	0,2	1 940,00р.	11 640,00р.	20%	2 328,00р.	Выдать: 9 312 руб.
	Всего к выдаче:						140 337,00р.		18 086,74р.	Выдать: 122 250 руб.

Рис. 11.5. Рабочий лист файла *2-Ведомость по зарплате.xlsm*

2. Введите на листе модуля ЭтаКнига файла *1-Ведомость по должностям и тарифным ставкам.xlsm* следующий код (листинг 11.1).

Листинг 11.1. Создание новой гиперссылки. Модуль ЭтаКнига файла *1-Ведомость по должностям и тарифным ставкам.xlsm*

```
Private Sub Workbook_Open()
   With Worksheets(1)
    .Hyperlinks.Add Anchor:=.Range("A1"), _
     Address:="D:\2-Пример использования гиперссылки\" & _
                "2-Ведомость по зарплате.xlsm", _
     ScreenTip:="НПО Альфа", TextToDisplay:="Расчет зарплаты сотрудников"
   End With
End Sub
```

3. На листе модуля ЭтаКнига файла *2-Ведомость по зарплате.xlsm* введите следующий код (листинг 11.2).

Листинг 11.2. Создание новой гиперссылки. Модуль ЭтаКнига файла *2-Ведомость по зарплате.xlsm*

```
Private Sub Workbook_Open()
   With Worksheets(1)
    .Hyperlinks.Add Anchor:=.Range("D1"), _
      Address:="D:\2-Пример использования гиперссылки\" & _
                 "1-Ведомость по должностям и тарифным ставкам.xlsm", _
      ScreenTip:="Должностные оклады НПО Альфа", _
      TextToDisplay:="ВЕРНУТЬСЯ НАЗАД"
   End With
End Sub
```

4. Убедитесь, что при открытии файла *1-Ведомость по должностям и тарифным ставкам.xlsm* в ячейке **A1** появляется гиперссылка (рис. 11.4), по которой осуществляется переход к файлу *2-Ведомость по зарплате.xlsm*, и наоборот (рис. 11.5).

Переход по гиперссылке из списка

Гиперссылки можно также применять и при организации данных на рабочем листе. Так, например, вы можете использовать список, который непосредственно связан с гиперссылками в ячейках рабочего листа. В качестве примера рассмотрим подготовку ведомости "Примеры лабораторных работ" (рис. 11.6), в которой используется элемент управления **Список**, позволяющий переходить к необходимому файлу примера (см. также папку *3-Пример использования списка и гиперссылок* на компакт-диске).

Итак, для подготовки примера выполните следующие действия.

1. Подготовьте необходимые файлы примеров с расширением xlsm, которые будут использованы при переходе по гиперссылкам, и расположите их в папке *3-Пример использования списка и гиперссылок*.

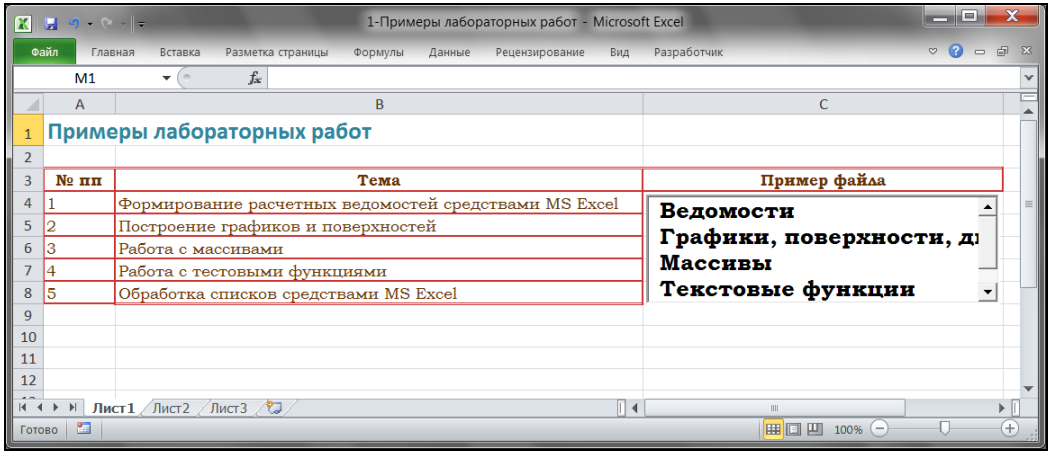


Рис. 11.6. Ведомость "Примеры лабораторных работ"

2. Откройте новую рабочую книгу и подготовьте на рабочем листе ведомость в соответствии с рис. 11.6, причем в ячейки **C4:C8** добавьте гиперссылки на соответствующий подготовленный файл примера. Сохраните подготовленный файл в указанной выше папке под именем *1-Примеры лабораторных работ.xlsm*.
3. В каждом из подготовленных файлов пункта 1 организуйте обратный переход по гиперссылке в файл *1-Примеры лабораторных работ.xlsm*, используя метод `Add` для добавления гиперссылки (см., например, листинг 11.3).

Листинг 11.3. Переход по гиперссылке в исходный файл. Модуль `ЭтаКнига`

```
Private Sub Workbook_Open()
Dim i As Integer
  For i = 1 To 3
   With Worksheets(i)
    .Hyperlinks.Add Anchor:=.Range("A1"), _
      Address:="D:\3-Пример использования списка и гиперссылок\" & _
                  "1-Примеры лабораторных работ.xlsm", _
      ScreenTip:="Вернуться в начальный файл", TextToDisplay:="НАЗАД"
    End With
  Next
End Sub
```

4. Разместите поверх диапазона **C4:C8** элемент управления **Список** (ListBox), воспользовавшись кнопкой со списком **Вставить**, расположенной на вкладке **Разработчик** ленты в группе **Элементы управления**.
5. Выполните щелчок мышью по элементу управления **Список** (ListBox) и введите на листе модуля `Лист1` код из листинга 11.4, который использует метод `Follow` для перехода по указанной гиперссылке.

Листинг 11.4. Переход по гиперссылке из списка. Модуль Лист1

```
Private Sub ListBox1_Click()
  Hyperlinks(ListBox1.ListIndex + 1).Follow
End Sub
```

6. На листе модуля ЭтаКнига введите код из листинга 11.5, который заполняет список и создает соответствующие объекты Hyperlink.

Листинг 11.5. Переход по гиперссылке из списка. Модуль ЭтаКнига

```
Private Sub Workbook_Open()
   Worksheets(1).ListBox1.ColumnCount = 2
   Worksheets(1).ListBox1.ColumnWidths = "100;0"
   Worksheets(1).ListBox1.Clear
   Dim lst(4, 1) As String
   lst(0, 0) = "Ведомости" :  lst(0, 1) = " 2-Ведомости.xlsm"
   lst(1, 0) = "Графики, поверхности, диаграммы"
   lst(1, 1) = "3-Графики_поверхности_диаграммы.xlsm"
   lst(2, 0) = "Массивы"   :  lst(2, 1) = "4-Массивы.xlsm"
   lst(3, 0) = "5-Текстовые функции"
   lst(3, 1) = "5-Текстовые функции.xlsm"
   lst(4, 0) = "Списки"    :  lst(4, 1) = "6-Списки.xlsm"

   Worksheets(1).ListBox1.List = lst
   Dim i As Integer
   Dim r As Hyperlink
   For Each r In Worksheets(1).Hyperlinks
      r.Delete
   Next
   For i = 0 To 4
     Worksheets(1).Hyperlinks.Add Anchor:=Worksheets(1).Cells(i + 4, 3), _
                          Address:=lst(i, 1), TextToDisplay:=lst(i, 0)
   Next
End Sub
```

7. Расположите подготовленную папку с файлами на диске D: вашего компьютера и проверьте правильность работы ваших гиперссылок.

Работаем с веб-страницами

Веб-запрос и получение данных с веб-страницы

Веб-страницы часто содержат информацию, которая необходима для анализа в Microsoft Office Excel. Так, например, в MS Excel можно анализировать котировки акций, используя данные, поступающие непосредственно с веб-страницы, или таблицу с данными о продажах с личной веб-страницы некоторой организации.

При необходимости можно извлечь обновляемые данные (в этом случае их можно обновлять непосредственно в MS Excel в соответствии с последними изменениями веб-страницы) или получить данные с веб-страницы и хранить их на листе статически.

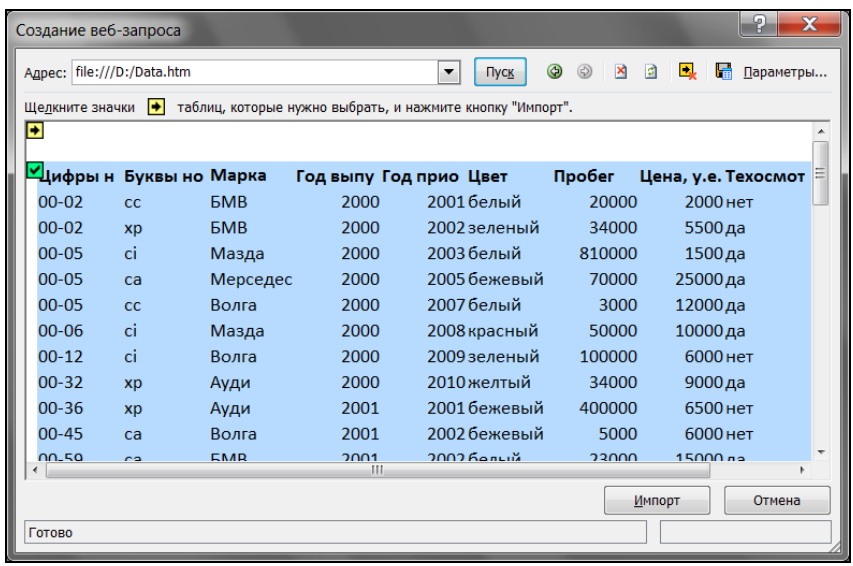
Использование веб-запроса позволяет получить с веб-страницы данные, например, отдельную таблицу, несколько таблиц или весь текст, и провести их анализ, используя средства MS Excel.

Для создания веб-запроса выполните, например, следующие действия:

1. Поместите файл Data.htm на диск D: вашего компьютера.
2. Откройте новую рабочую книгу, перейдите на вкладку ленты **Данные** и в группе **Получение внешних данных** выберите команду **Из Интернета**.
3. В открывшемся окне **Создание веб-запроса** в поле адреса введите ссылку на расположенный вами файл **file:///D:/Data.htm**. (рис. 11.7, см. также соответствующий файл в папке *4-Получение веб-запроса* на компакт-диске).

> ***Примечание***
>
> Поле **Адрес** предназначено для ввода URL необходимой веб-страницы. Например, если на вашем компьютере установлен сервер IIS (об этом будет рассказано далее), то его главным каталогом является C:\Inetpub\wwwroot. Вы можете расположить в этом каталоге, например, файл с расширением asp (веб-страница, также содержащая данные). Тогда в поле **Адрес** вам следует ввести следующую ссылку: **http://localhost/имя_файла.asp**. Ссылку в виде C:\Inetpub\wwwroot*имя_файла*.asp указывать в этом окне нельзя.

4. Используя кнопки со стрелками, выделите на веб-странице искомую таблицу.
5. Нажмите кнопку **Импорт**.



Цифры н	Буквы но	Марка	Год выпу	Год прио	Цвет	Пробег	Цена, у.е.	Техосмот
00-02	cc	БМВ	2000	2001	белый	20000	2000	нет
00-02	xp	БМВ	2000	2002	зеленый	34000	5500	да
00-05	ci	Мазда	2000	2003	белый	810000	1500	да
00-05	ca	Мерседес	2000	2005	бежевый	70000	25000	да
00-05	cc	Волга	2000	2007	белый	3000	12000	да
00-06	ci	Мазда	2000	2008	красный	50000	10000	да
00-12	ci	Волга	2000	2009	зеленый	100000	6000	нет
00-32	xp	Ауди	2000	2010	желтый	34000	9000	да
00-36	xp	Ауди	2001	2001	бежевый	400000	6500	нет
00-45	ca	Волга	2001	2002	бежевый	5000	6000	нет
00-59	ca	БМВ	2001	2002	белый	23000	15000	да

Рис. 11.7. Окно **Создание веб-запроса**

6. В открывшемся окне **Импорт данных** (рис. 11.8) установите переключатель **Куда следует поместить данные?** в положение **Имеющийся лист** и в соответствующее поле введите ссылку на ячейку (например, **A1**), в которой будет располагаться левый верхний угол импортируемой таблицы.

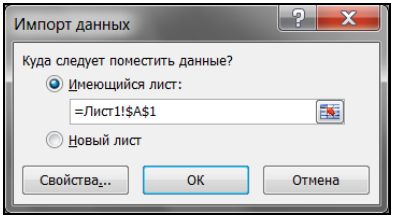


Рис. 11.8. Окно **Импорт данных**

7. Нажмите кнопку **Свойства** для задания параметров запроса.
8. В окне **Свойства внешнего диапазона** (рис. 11.9) поле **Имя** задает имя запроса. Группа **Определение запроса** служит для сохранения пароля и сохранения определения запроса. Группа **Обновление экрана** задает режим обновления импортированных данных. Группа **Формат и разметка данных** устанавливает параметры форматирования. Группа **Если количество строк в диапазоне изменится** задает алгоритм действия при изменении размеров диапазона. Флажок **заполнить формулами соседние столбцы** определяет, надо ли вводить вместе с данными и пояснительные формулы. Оставьте все предлагаемые по умолчанию параметры окна **Свойства внешнего диапазона** и нажмите кнопку **OK**.

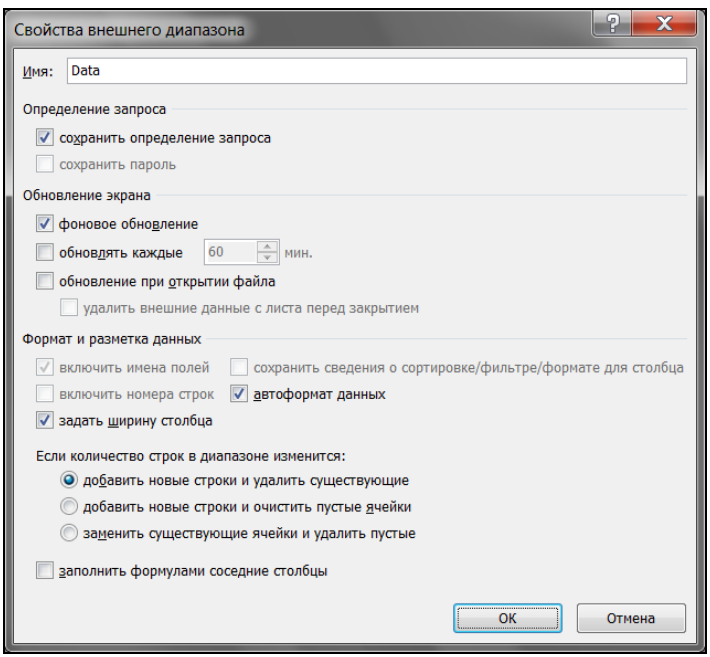


Рис. 11.9. Окно **Свойства внешнего диапазона**

9. Окно **Свойства внешнего диапазона** закроется. Нажмите кнопку **OK** окна **Импорт данных**.
10. Данные из выбранной таблицы веб-страницы будут импортированы на рабочий лист (рис. 11.10).

A1 | Цифры номера

	A	B	C	D	E	F	G	H	I	J	K
1	Цифры но	Буквы ном	Марка ма	Год выпус	Год приоб	Цвет	Пробег	Цена, у.е.	Техосмотр	Владелец	
2	00-02	сс	БМВ	2000	2001	белый	20000	2000	нет	Мышко	
3	00-02	хр	БМВ	2000	2002	зеленый	34000	5500	да	Рагойша	
4	00-05	сі	Мазда	2000	2003	белый	810000	1500	да	Ильющенко	
5	00-05	са	Мерседес	2000	2005	бежевый	70000	25000	да	Блотак	
6	00-05	сс	Волга	2000	2007	белый	3000	12000	да	Константинова	
7	00-06	сі	Мазда	2000	2008	красный	50000	10000	да	Кузьмицкая	
8	00-12	сі	Волга	2000	2009	зеленый	100000	6000	нет	Макар	
9	00-32	хр	Ауди	2000	2010	желтый	34000	9000	да	Иванова	
10	00-36	хр	Ауди	2001	2001	бежевый	400000	6500	нет	Григорьева	
11	00-45	са	Волга	2001	2002	бежевый	5000	6000	нет	Васильев	
12	00-59	са	БМВ	2001	2002	белый	23000	15000	да	Козлов	
13	00-65	сс	Мерседес	2001	2002	красный	7900	6000	да	Оскирко	
14	00-80	сс	БМВ	2001	2002	красный	70000	6500	да	Костечко	
15	00-82	хр	БМВ	2001	2003	белый	40000	3000	нет	Беломызова	
16	00-88	сі	Ауди	2001	2003	красный	79000	3000	да	Васильев	
17	00-97	хр	Мерседес	2001	2005	белый	40000	9000	да	Гинз	
18	00-98	сс	Опель	2001	2005	белый	650000	1100	да	Заяц	
19	20-59	сс	Опель	2001	2005	желтый	2000	15000	да	Михолап	
20	23-57	сі	Мазда	2001	2005	зеленый	7900	26000	да	Радюкевич	
21	23-76	са	Запороже	2001	2005	зеленый	65000	16000	да	Гончарук	
22	23-98	хр	Мерседес	2001	2005	красный	150000	4000	да	Рыбак	
23	30.Чэр	са	Мерседес	2001	2005	красный	650000	3500	да	Степанов	
24	32-09	сі	Мерседес	2001	2007	бежевый	30000	15000	нет	Котов	
25	36-07	са	Мерседес	2001	2010	бежевый	23000	14000	да	Кузьма	
26	36-22	са	Мерседес	2001	2010	зеленый	650000	1100	да	Иванченко	
27	36-55	сі	Ауди	2002	2002	зеленый	7900	2000	да	М илашевская	
28	40-51	са	БМВ	2002	2002	зеленый	100000	6000	да	Иванов	

Рис. 11.10. Таблица, импортированная на рабочий лист в результате веб-запроса

Создаем скрипты

А теперь разберем несколько несложных примеров создания скриптов, которые продемонстрируют возможности языка программирования VBScript. Язык VBScript является усеченной версией языка VBA и позволяет писать сценарии как для Интернета, так и для Windows.

Обычно под скриптом понимается программа или программный файл-сценарий. Если же дать более точное определение, то скриптом называется, практически, любая исполняемая процедура. В интернет-технологиях понятие "скрипт" характеризует исполняемую процедуру, которая запускается на выполнение со стороны сервера по требованию (запросу), поступившему от конкретной веб-страницы.

Скрипты находят применение в различных областях. Например, с их помощью пользователь может обращаться к базам данных, наблюдать статистику посещений на веб-сайте (счетчики посещаемости), делать записи в гостевых книгах, оставлять комментарии к статьям и т. п.

Непосредственно скрипт располагается в сети по-разному. С одной стороны, его можно разместить, например, на том же сервере, где расположена и вызывающая его страница. С другой стороны, скрипт размещается на другом (удаленном) сервере в локальной или глобальной интернет-сети. Следует также помнить, что запуск

скрипта влечет выполнение какого-либо действия, не всегда полезного, например, для владельца сервера. В силу этого, обычно не на всех серверах разрешается выполнение скриптов — провайдеры дополнительно оговаривают условия предоставления такой возможности.

Как создать скрипты для веба на стороне клиента?

Используя VBScript, можно писать код непосредственно в HTML-файлах, и этот код будет выполняться при загрузке такого файла в браузер, т. е. на стороне клиента. Такой код должен быть помещен внутрь парного тега `<script>`, атрибут `language` которого специфицирует язык, на котором написан сценарий. В качестве примера приведем код из листинга 11.6, который обеспечивает отображение окна с приветствием при щелчке на надписи "Hello, World!" (рис. 11.11, см. также файл *5-Helloworld.htm* на компакт-диске).

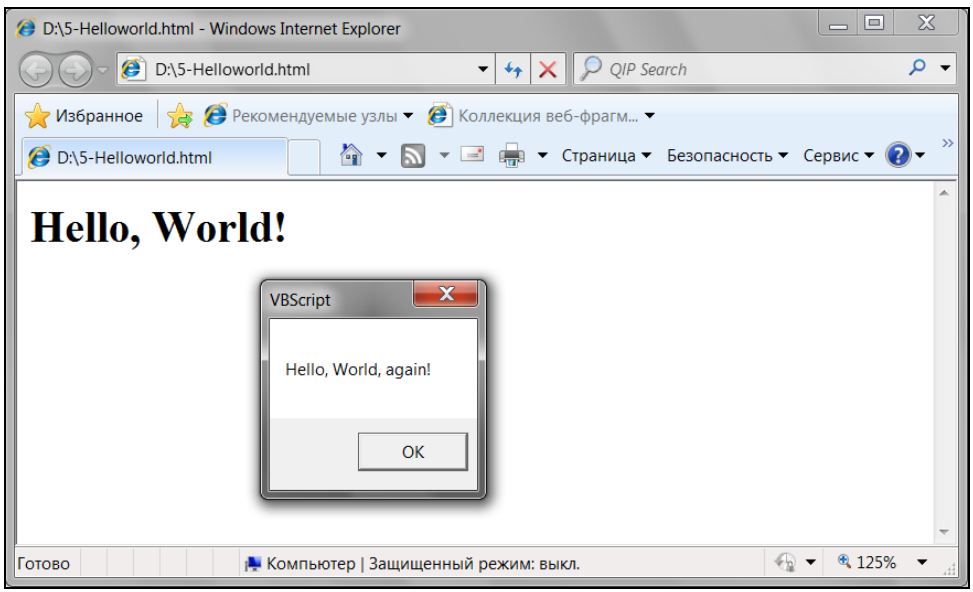


Рис. 11.11. Скрипт для Windows с отображением приветствия при щелчке на надписи

Листинг 11.6. Приветствие. Файл Helloworld.html

```
<html>
<head>
   <script language="VBScript">
      Sub hello()
         Msgbox "Hello, World, again!"
      End Sub
```

```
   </script>
</head>
<body>
  <h1 onclick="hello">Hello, World!</h1>
</body>
</html>
```

Для создания и выполнения приведенного здесь примера выполните следующее.

1. Откройте стандартное приложение Windows — Блокнот (Notepad).
2. В открывшемся окне наберите предлагаемый код (листинг 11.6).
3. Сохраните созданный код в файле с расширением html, например, Hello-world.html в корневом каталоге диска D:.
4. Откройте созданный вами файл, т. е. либо дважды щелкните по нему левой кнопкой мыши, либо нажмите клавишу <Enter>.
5. В окне браузера отобразится написанное вами приветствие: "Hello, World!" Если вашим браузером блокируется выполнение сценариев или элементов ActiveX, произведите соответствующий щелчок мышью в появившемся сообщении окна браузера и разрешите выполнения заблокированного содержимого.
6. Щелкните в окне браузера по надписи "Hello, World!" и убедитесь в выполнении созданного скрипта (см. рис. 11.11).

Как создать скрипты для веба на стороне сервера?

Скрипты Microsoft Active Service Pages (ASP, активные серверные страницы), написанные на VBScript, позволяют создавать сценарии, которые выполняются на стороне сервера. Из всей выходной информации ASP-сценарии передают на компьютер клиента только текст и HTML-теги, где они воспроизводятся в окне браузера. При помощи объектной модели ADO (ActiveX Data Object) ASP доступны базы данных, хранимые на сервере, а при помощи FSO — текстовые файлы, хранимые там же. Это позволяет легко создавать различные веб-приложения типа виртуальных каталогов. Упрощенно ADO можно описать как модель доступа к данным, оптимизированную для использования в веб-проектах.

Для применения ASP необходимо установить веб-сервер Internet Information Server (IIS) корпорации Microsoft для работы с операционной системой, например, Windows XP, Windows Vista или Windows 7.

Windows предоставляет возможность проигрывать серверные приложения на компьютере клиента. Для этого файл серверного приложения (например, Hello.asp, приведенный в листинге 11.7) необходимо разместить в каталоге C:\Inetpub\wwwroot компьютера клиента. Это корневой каталог сервера, создаваемый по умолчанию при его инсталляции. Клиенты, подключенные к вашему компьютеру с помощью HTTP-протокола, автоматически попадают в этот каталог. В каталоге имеется файл iisstart.htm (для более ранних версий Windows — файл Default.htm), который открывается в браузере клиента при его соединении с сервером без указания имени документа. В поле адреса браузера для открытия asp-файла с вашего компьютера следует вводить:

```
http://127.0.0.1/имя_файла.asp
```

или

`http://localhost/`*`имя_файла`*`.asp`

где `http://127.0.0.1` или `http://localhost` — зарезервированный IP-адрес для подключения к серверу, запущенному на компьютере, с которого поступил запрос на подключение.

В качестве примера создадим следующий код (листинг 11.7), который отображает в браузере различные приветствия в зависимости от времени загрузки документа (рис. 11.13). Для того чтобы компьютер мог различать, какая часть кода из ASP-файла должна выполняться на серверной стороне, необходимо воспользоваться тегом `<% statements %>`, где *`statements`* — операторы, выполняемые на сервере. Для определения времени загрузки документа используется функция `Time()`, которая возвращает искомую величину.

Итак, выполните следующие действия.

1. Проверьте, чтобы на вашем компьютере был установлен веб-сервер IIS. В противном случае выполните его установку. Так, например, если вы используете Windows 7, то для установки веб-сервера IIS перейдите к Панели управления, далее выберите — **Программы и компоненты** (находится в категории **Программы** Панели управления), а затем щелкните по задаче **Включение или отключение компонентов Windows**. В открывшемся окне **Компоненты Windows** выберите **Службы IIS** (рис. 11.12).

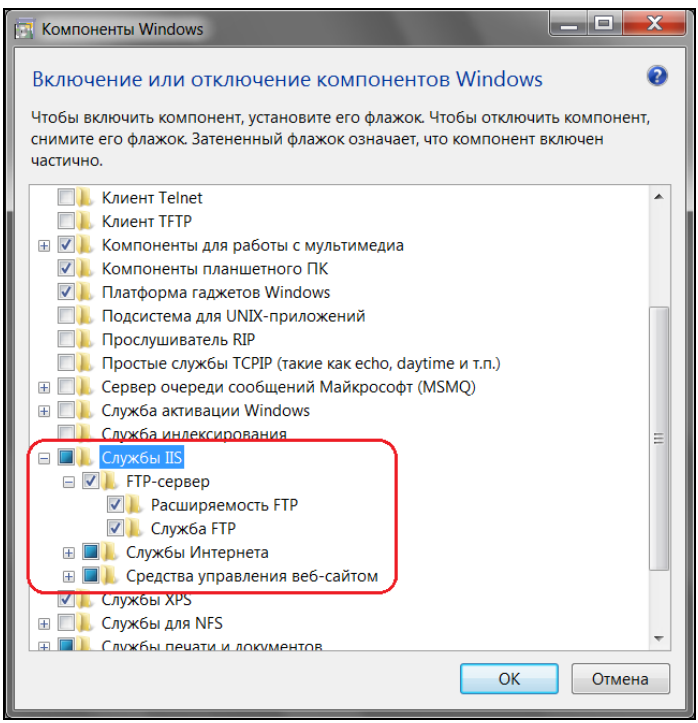


Рис. 11.12. Окно **Компоненты Windows**

2. В Блокноте (Notepad) наберите предлагаемый код (см. листинг 11.7) и сохраните файл с расширением asp, например, 6-Hello.asp.
3. Поместите созданный файл 6-Hello.asp в каталог C:\Inetpub\wwwroot на вашем компьютере.
4. Запустите файл iisstart.htm, который расположен в каталоге C:\Inetpub\wwwroot, и введите в адресной строке открывшегося окна браузера, например, следующий адрес: **http://127.0.0.1/6-hello.asp**.
5. В окне браузера отобразится приветствие в зависимости от времени загрузки документа (рис. 11.13).

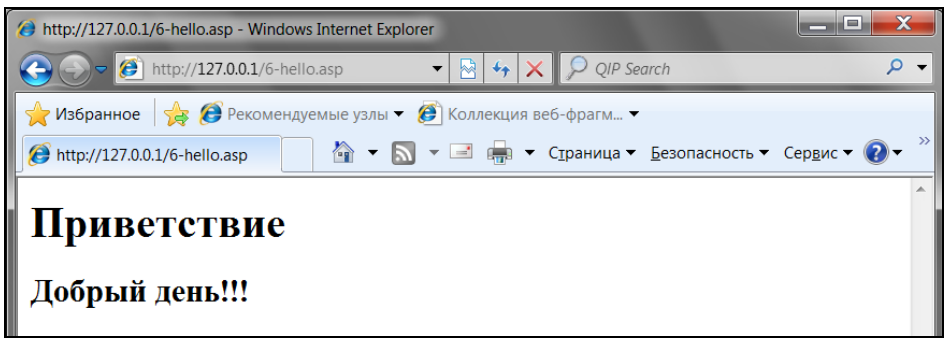


Рис. 11.13. Скрипт серверного сценария с приветствием, зависящим от времени загрузки документа в окно браузера

6. Щелкните в открытом окне браузера правой кнопкой мыши и из появившегося контекстного меню выберите команду **Просмотр HTML-кода**. Вы увидите код, который получает клиент от сервера при проигрывании данного проекта. Например, загрузив файл *6-Hello.asp* в браузер в 17:12, можно получить следующий переданный код (листинг 11.8, см. также файл *6-Hello.asp* на компакт-диске).

Листинг 11.7. Приветствие. Файл 6-Hello.asp

```
<html>
<body>
<h1>Приветствие</h1>
   <% If Time>= #04:00:00# And Time < #12:00:00# Then %>
      <h2>Доброе утро!!!</h2>
   <% ElseIf Time>= #12:00:00# And Time < #18:00:00# Then %>
      <h2>Добрый день!!! </h2>
   <% ElseIf Time>= #18:00:00# And Time < #23:00:00# Then %>
      <h2>Добрый вечер!!! </h2>
   <%Else%>
      <h2>Доброй ночи!!! </h2>
   <%End If %>
</body>
</html>
```

Листинг 11.8. Приветствие. Код, передаваемый клиенту

```
<html>
<body>
  <h1>Приветствие</h1>
  <h2>Добрый день!!! </h2>
</body>
</html>
```

Следующий код (листинг 11.9) отображает в браузере различные приветствия в зависимости от дня недели, в который был загружен документ (рис. 11.14). Для того чтобы компьютер мог различать, какая часть кода из ASP-файла должна выполняться на серверной стороне, надо воспользоваться тегом `<% statements %>`, где *`statements`* — операторы, выполняемые на сервере. Знак равенства внутри тега `<%=Variable%>` представляет собой сокращенное обозначение метода `write` объекта `Response`, который используется для пересылки информации от сервера клиенту (см. файлы в папке *7-Скрипты на стороне сервера* на компакт-диске).

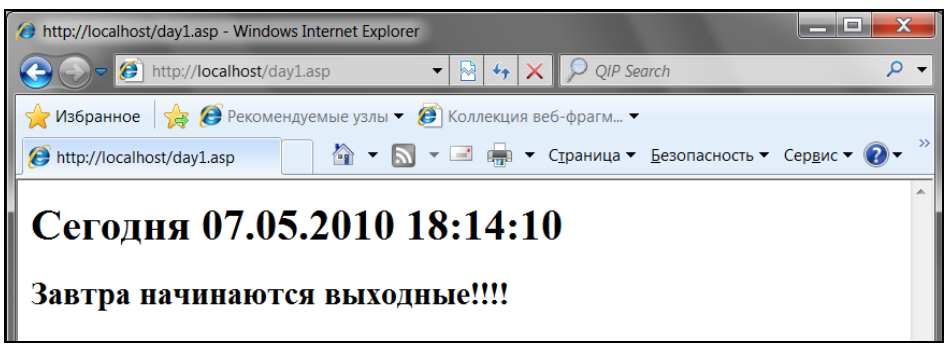


Рис. 11.14. Приветствие, передаваемое от сервера в зависимости от дня недели

Листинг 11.9. Ваш первый скрипт на стороне сервера. Файл day1.asp

```
<%@ Language=VBScript %>
<html>
<h1>Сегодня <%= Now%></h1>
<% If Weekday(Now) = vbFriday Then %>
<h2>Завтра начинаются выходные!!!!</h2>
<% ElseIf Weekday(Now) = vbSaturday Or  Weekday(Now) = vbSunday Then %>
<h2> Выходные!!!!</h2>
<% Else %>
<h2>Надо еще поработать</h2>
<% End If %>
</html>
```

Приведем тот же самый код, но написанный с использованием объекта `Response` и его метода `write`, который осуществляет пересылку информации от сервера клиенту (листинг 11.10).

Листинг 11.10. Ваш первый скрипт на стороне сервера. Файл day2.asp

```
<%@ Language=VBScript %>
<html>
<h1>Сегодня <%response.write Now%></h1>
<%
  If Weekday(Now) = vbFriday Then
     response.write "<h2>Завтра начинаются выходные!!!!</h2>"
  ElseIf Weekday(Now) = vbSaturday Or  Weekday(Now) = vbSunday Then
    response.write "<h2>Завтра начинаются выходные!!!!</h2>"
  Else
    response.write "<h2>Надо еще поработать</h2>"
  End If
%>
</html>
```

Как передать данные от клиента к серверу?

Для передачи данных от клиента к серверу часто используются формы, которые группируют в себе несколько элементов управления. В примере (листинг 11.11 и листинг 11.12), который приводится далее, на форме имеются два поля — **Имя** и **E-Mail**, и две кнопки — **Submit** и **Reset** (рис. 11.15). Нажатие кнопки **Submit** инициирует выполнение ASP-файла, указанного в качестве значения атрибута `action` тега `<form>` (в данном случае Answer.asp). Свойство `form` серверного объекта `Request` позволяет передать этому файлу значения из тегов управления, расположенных в форме. Идентификация тегов производится по значениям их атрибутов `name`. После заполнения формы пользователь нажимает кнопку **Submit**, что вызывает передачу данных на сервер и обработку их программой Answer.asp, которая подтверждает получение сервером данных (рис. 11.16). Нажатие же кнопки **Reset** приведет к сбросу значений элементов управления, входящих в форму, которые используются по умолчанию (см. также файлы в папке *8-Передача данных от клиента серверу* на компакт-диске).

Для выполнения данного примера:

1. В Блокноте (Notepad) наберите предлагаемый код (листинг 11.11) и сохраните файл под именем ask.htm.

Листинг 11.11. Передача данных от клиента серверу. Файл ask.htm

```
<html>
<body>
<h1>Введите Ваши данные</h1>
<form id="frmName" method="post" action="http:\\localhost\answer.asp">
  Имя: <input type="text" name="txtName">
```

```
  <br>
  E-Mail <input type="text" name="txtEMail">
 <br>
  <input type="submit" value="Submit"><input type="reset" value="Reset">
</form>
</body>
</html>
```

2. Код из листинга 11.12 также наберите в Блокноте и сохраните файл под именем answer.asp.

Листинг 11.12. Передача данных от клиента серверу. Файл answer.asp

```
<%@ Language=VBScript %>
<html>
<body>
<%
   Response.Write "<H2> Полученные данные </H2>"
   Response.Write "Имя:  " & Request.form("txtName")
   Response.Write "<BR>"
   Response.Write "E-Mail: " & Request.form("txtEMail")
%>
</body>
</html>
```

3. Поместите созданные файлы в каталог C:\Inetpub\wwwroot на вашем компьютере.
4. Запустите файл iisstart.htm, который расположен в каталоге C:\Inetpub\wwwroot, и введите в адресной строке открывшегося окна браузера следующий адрес: **http://localhost/ask.htm**.
5. В окне браузера отобразится форма с запросом данных (рис. 11.15).

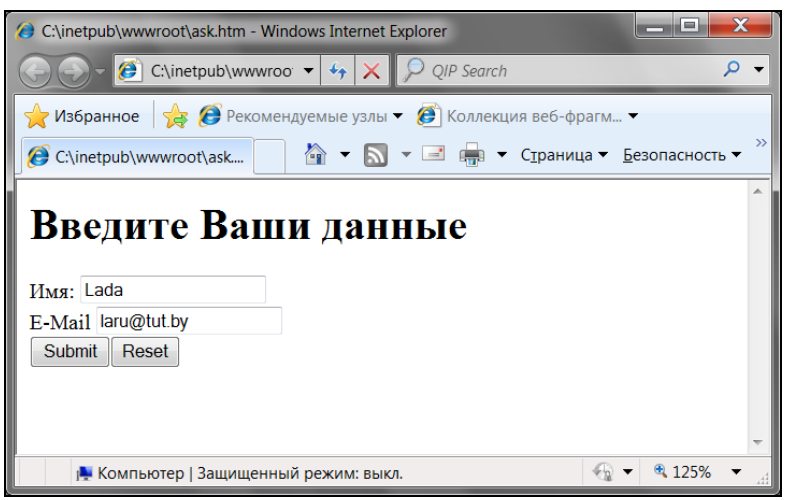


Рис. 11.15. Форма для передачи данных

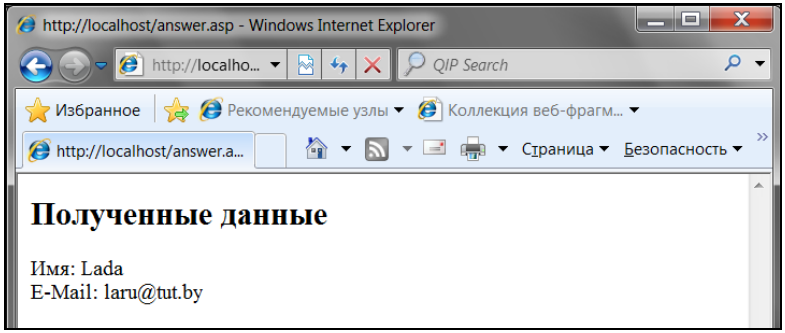


Рис. 11.16. Ответ, переданный сервером клиенту

6. Щелкните в открытом окне браузера по кнопке **Submit** и проверьте выполнение ASP-запроса (рис. 11.16). Проверьте также функционирование кнопки **Reset**.

Наши итоги

Эта глава помогла вам познакомиться с Интернетом и с основными службами, которые вы там можете найти. Конечно, материал данной главы не может охватить все аспекты, связанные с Интернетом и технологиями, которые используются во Всемирной сети. Однако, внимательно прочитав предложенный здесь материал, вы имеете представление:

- ❒ об общих ресурсах, которые предоставляет Интернет;
- ❒ о возможности использования гиперссылок в рабочих книгах Excel;
- ❒ о включении гиперссылок в программы на VBA;
- ❒ об использовании веб-запроса и получении данных с веб-страницы в рабочую книгу Excel;
- ❒ а также — о скриптах, их назначении и использовании в Интернете.

Глава 12

Об интеграции приложений

В этой главе мы рассмотрим технологию Automation, которой обладает VBA, что существенно упрощает процесс и сокращает время разработки проектов. Технология Automation предоставляет разработчику доступ к объектам и методам других приложений, максимально приспособленных для решения специализированных задач. На различных примерах мы продемонстрируем, как можно интегрировать работу Microsoft Office Excel с Microsoft Office Word, Microsoft Office Access, Microsoft Office Outlook и Microsoft Office PowerPoint.

Примечание

Файлы рассмотренных в данной главе примеров можно найти в папке Glava_12 на компакт-диске.

Что такое технология ActiveX?

Как вам известно, каждое из приложений пакета MS Office является очень гибким продуктом, предоставляющим пользователю большие возможности. Используя эти возможности, можно создать более эффективные и технологичные приложения. Технология ActiveX предоставляет возможность взаимодействия различных приложений MS Office друг с другом, а также с любыми программами, поддерживающими эту технологию, например, с Visual Basic, при помощи модели составных объектов (Component Object Model, COM). Данная технология называлась раньше OLE (Object Linking and Embedding, связывание и внедрение объектов). В настоящее время OLE является частью технологии ActiveX.

Технология ActiveX предназначена для простого и эффективного взаимодействия различных продуктов. Например, MS Word или MS Excel могут предоставить свои объекты друг другу. Это позволяет создавать документы, состоящие из элементов, разработанных в различных приложениях. ActiveX предоставляет разработчику два мощных средства для создания приложений:

- *OLE* — связывание и внедрение объектов разных приложений, что позволяет непосредственно в документе редактировать объекты, созданные другими приложениями;
- *Automation* — технология, которая предоставляет возможность программного управления как внедренными объектами, так и объектами других приложений. Таким образом, значительно сокращается время разработки, т. к. нет нужды

заново реализовывать те возможности, которые не включены в качестве встроенных средств, но которые имеются у других специализированных приложений.

Примечание

Технология ActiveX подразумевает, что интегрируемые приложения, объекты которых используются в документе, установлены и зарегистрированы в системном реестре. Регистрация приложений, как правило, происходит автоматически при их установке.

Связываем и внедряем объекты

Технология ActiveX позволяет внедрять объект, созданный в одном приложении, в документ, созданный другим приложением. Например, можно внедрить рабочий лист или диаграмму MS Excel в документ MS Word. Конечно, можно сделать и, наоборот, в рабочий лист MS Excel внедрить документ MS Word.

Объектом называется произвольный элемент, созданный в каком-либо одном приложении (*приложении-источнике* или *сервере*), который можно поместить в документ другого приложения (*приложения-приемника* или *клиента*), причем сделать это так, что вместе со вставленными данными будет храниться информация о приложении, создавшем этот объект. Впоследствии это дает возможность редактировать объект средствами создавшего его приложения. При *внедрении* все данные копируются в документ-контейнер. При *связывании* в контейнер записывается только информация о приложении-источнике, из которого взят вставляемый объект.

В документ можно внедрять как еще не существующий (новый) объект, так и уже существующий.

Связь данных

Если же вам необходимо в документе использовать технологию связывания данных, то в приложениях Microsoft Office она может осуществляться двумя способами:

- *с помощью формулы удаленной ссылки*, которая вводится с клавиатуры или вставляется в документ с помощью команды **Специальная вставка**, которая вбирается из списка при нажатии кнопки со стрелкой **Вставить**, расположенной в группе **Буфер обмена** на вкладке **Главная** ленты;
- *с использованием макросов*, управляющих динамическим обменом данными (DDE).

Многие приложения Microsoft Office могут получать данные из других приложений пакета, причем при изменении данных в приложении-сервере данные в приложении-клиенте обновляются автоматически.

Способ обновления связанных данных вы можете задать по выбору: автоматическое или ручное. Если обновления производятся вручную, то связанные приложения работают быстрее.

Для связывания, например, документа-клиента Microsoft Office Word 2010 и другого приложения вам нужно произвести следующие действия:

1. Откройте документ-клиент Word и приложение-сервер.
2. Активизируйте приложение-сервер, т. е. перейдите к нему.
3. Выделите данные, которые подлежат связыванию.

4. Перейдите на вкладку **Главная** ленты и в группе **Буфер обмена** нажмите кнопку **Копировать**.
5. Активизируйте документ-клиент Word и установите курсор в место вставки связываемых данных.
6. Перейдите на вкладку **Главная** ленты и в группе **Буфер обмена**, нажав кнопку со стрелкой **Вставить**, выберите команду **Специальная вставка**.
7. В открывшемся окне **Специальная вставка** установите переключатель **Связать**, выберите способ связывания из списка **Как** и нажмите кнопку **ОК**.

> ***Совет***
>
> При работе с документами в Microsoft Office удобно использовать гиперссылки.

Кроме того, связать документ-клиент Word 2010 и другое приложение можно также с использованием окна **Вставка объекта**, для вызова которого перейдите на вкладку **Вставка** ленты и в группе **Текст**, нажав кнопку со стрелкой **Объект**, выберите команду **Объект**. В открывшемся окне **Вставка объекта** перейдите на вкладку **Создание из файла** (рис. 12.1), в поле **Имя файла** укажите местоположение приложения-сервера, используя при необходимости кнопку **Обзор**, а также установите флажки возле опции **Связь с файлом** и **В виде значка**. Вы можете также подобрать соответствующий значок, который будет отображаться в документе, для чего используйте кнопку **Сменить значок**.

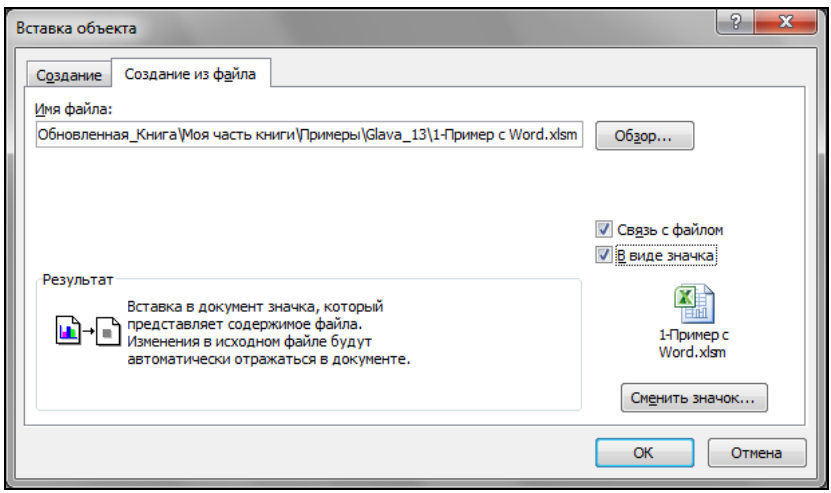


Рис. 12.1. Окно **Вставка объекта**

В результате такой связи у вас в документе будет содержаться значок, щелчок мышью по которому приводит к открытию соответствующего приложения-сервера.

> ***Примечание***
>
> Если в окне **Вставка объекта** на вкладке **Создание из файла** вы установите лишь флажок **Связь с файлом**, то в документе-клиенте у вас будет отображаться содержимое документа-сервера (рис. 12.2), щелчок по которому приведет к открытию документа-сервера.

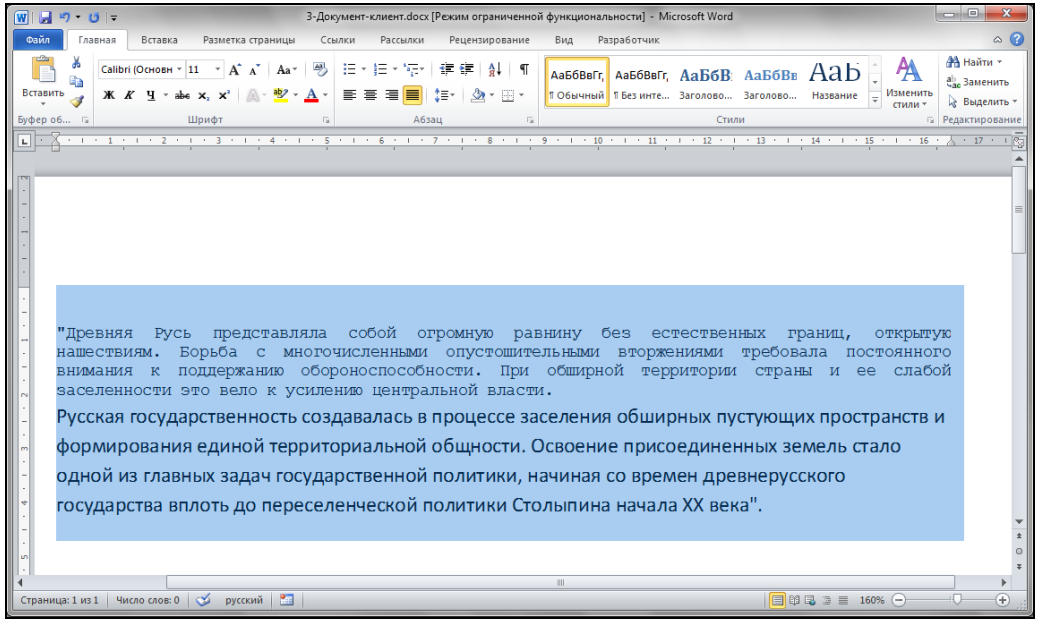


Рис. 12.2. Отображение в документе-клиенте данных, связанных с документом-сервером

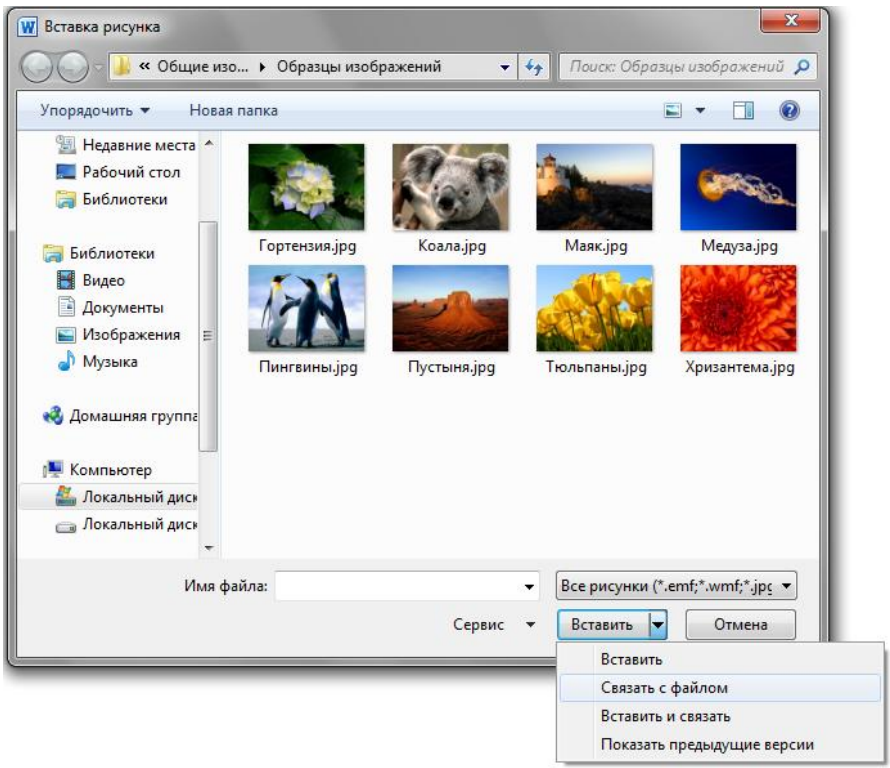


Рис. 12.3. Окно **Вставка рисунка**

Для приложений Microsoft Office вы можете задать режим обновления данных: перейдите на вкладку ленты **Файл** и нажмите кнопку **Параметры**. В открывшемся окне выберите слева категорию **Дополнительно**, а справа в группе **Общие** установите флажок **Автоматически обновлять связи при открытии.**

Примечание

При вставке рисунков в документ с помощью окна **Вставка рисунка** (перейдите на вкладку **Вставка** ленты и в группе **Иллюстрации** нажмите кнопку **Рисунок**) у вас также имеется возможность настроить связь Word-документа с графическим файлом: нажмите в окне **Вставка рисунка** кнопку со стрелкой **Вставить** и выберите необходимую команду (рис. 12.3).

Внедрение данных из других приложений

В приложениях Microsoft Office 2010 имеется также возможность внедрения данных в документ из любого приложения-сервера, поддерживающего OLE-технологию.

После внедрения данные становятся частью документа конкретного приложения Microsoft Office 2010. При редактировании таких данных приложение-сервер запускается из приложения-клиента. Вставленный объект сохраняется вместе с файлом документа, и его редактирование не приводит к изменению исходного файла.

После загрузки приложения-сервера из приложения-клиента можно просматривать и обрабатывать внедренный объект и одновременно видеть документ, в который внедрен этот объект. Такая возможность называется *местной активацией*.

Внедрение объекта в документ приложения Microsoft Office Word 2010 производится двумя способами:

- с использованием окна **Вставка объекта** (см. рис. 12.1), которое позволяет создать внедряемый объект сразу в приложении-клиенте:
  - на вкладке **Создание** можно выбрать внедряемый объект по типу приложения-сервера;
  - на вкладке **Создание из файла** можно выбрать внедряемый объект в виде файла;

Примечание

Не забудьте, что в данном случае в окне **Вставка объекта** (см. рис. 12.1) опция **Связь с файлом** не устанавливается.

- путем копирования из того документа, в котором он находится.

Вставка внедряемых объектов в документ Microsoft Office 2010 производится с помощью двух типов приложений:

- любых приложений-серверов, поддерживающих OLE;
- надстроек, которые прилагаются к Microsoft Office 2010. Надстройки не являются самостоятельными приложениями и используются только из какого-либо приложения-клиента.

Приложения Windows, которые частично поддерживают OLE, могут не появиться в диалоговом окне **Вставка объекта**. Однако существует возможность внедрения таких объектов одним из следующих способов:

- ❒ выполните копирование (в документе приложения-сервера);
- ❒ перейдите на вкладку **Главная** ленты и в группе **Буфер обмена**, нажав кнопку **Вставить**, выберите команду из списка **Специальная вставка** (в приложении-клиенте).

Немного примеров

Пусть для начала нам необходимо подготовить документ в Word, содержащий список и примеры лабораторных работ по курсу "Прикладные и интегрированные пакеты", оформить его в виде таблицы, содержащей внедренные объекты (рабочие книги MS Excel) в виде значков (рис. 12.4, см. также файл *1-Список и примеры лабораторных работ (внедрение).docx* на компакт-диске).

Для подготовки такого документа используйте рекомендации, которые приведены далее.

1. Создайте новый документ Word и добавьте название для таблицы.
2. Подготовьте макет таблицы и добавьте информацию в заголовок таблицы и в столбцы: **№ ПП**, **Тема**.

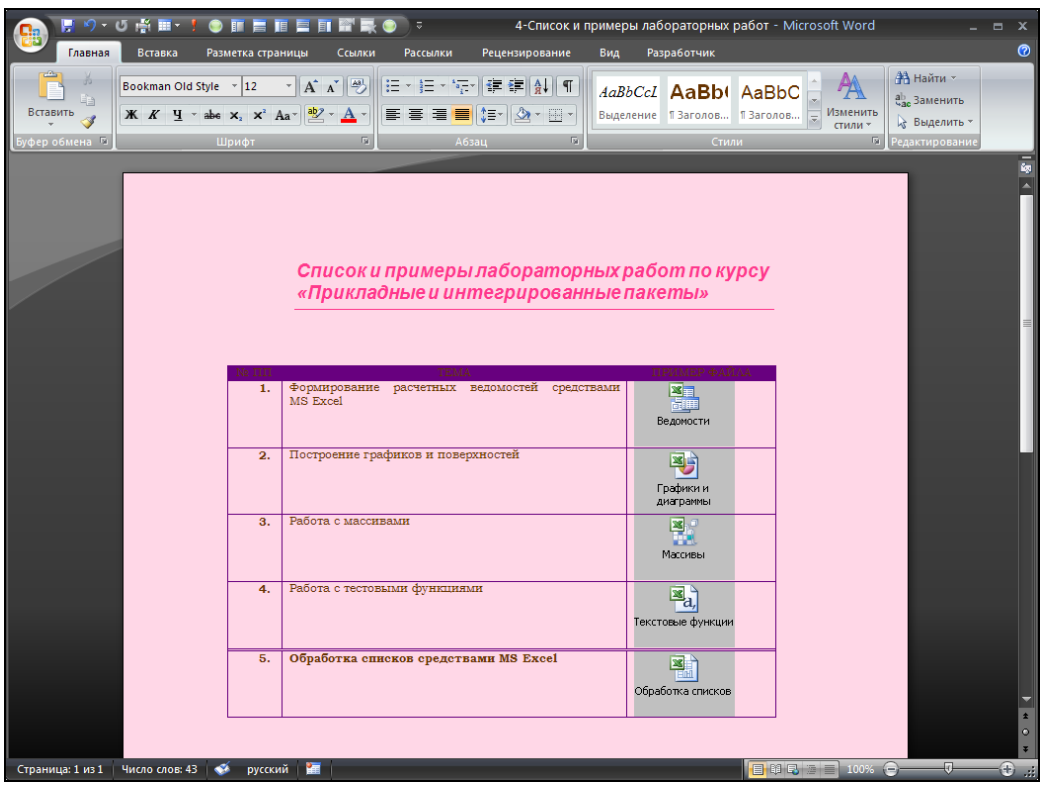


Рис. 12.4. Документ Word с внедренными объектами

3. Для внедрения объектов в столбец **Пример файла**, которые будут выводиться в виде значков, перейдите на вкладку **Вставка** ленты и в группе **Текст**, нажав кнопку со стрелкой **Объект**, выберите команду **Объект**.
4. В диалоговом окне **Вставка объекта** на вкладке **Создание из файла** (рис. 12.5) укажите в поле **Имя файла** местоположение файла, поставьте флажок возле **В виде значка**, с помощью кнопки **Сменить значок** выберите подходящий значок и добавьте подпись (рис. 12.6).
5. Отформатируйте полученную таблицу и сохраните ваш подготовленный документ.

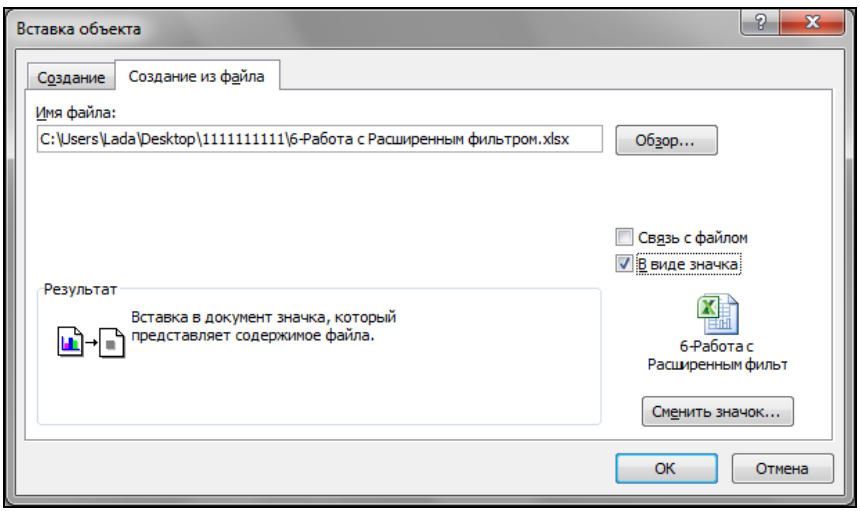


Рис. 12.5. Диалоговое окно **Вставка объекта**

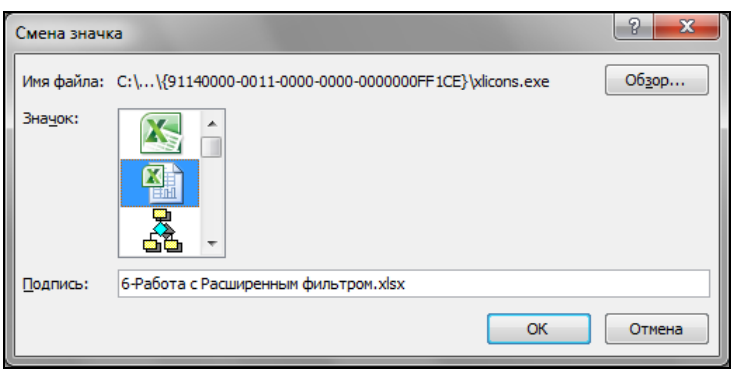


Рис. 12.6. Диалоговое окно **Смена значка**

А сейчас мы рассмотрим подготовку Word-документа "Сводная табличная ведомость", который содержит данные таблицы из книги Excel (рис. 12.7, см. также файл *2-Список и примеры лабораторных работ (связь).docx* на компакт-диске).

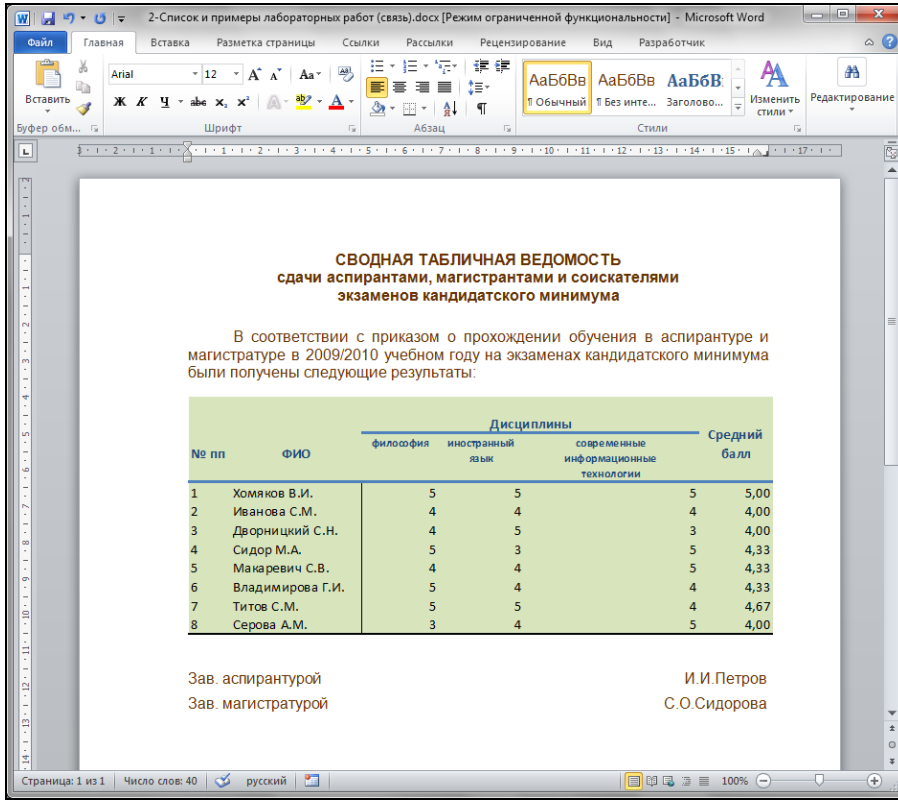


Рис. 12.7. Word-документ (клиент) с документом Excel (сервером)

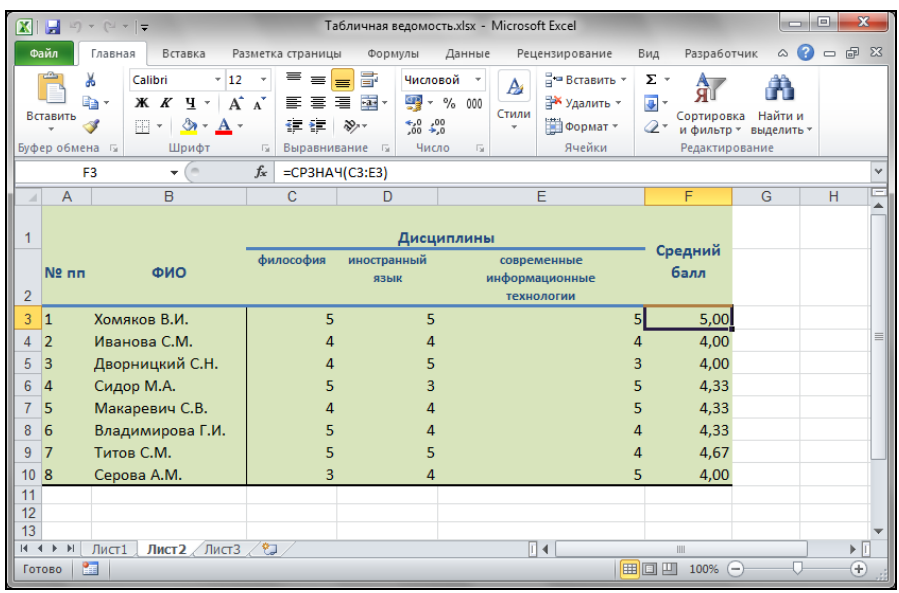


Рис. 12.8. Подготовка Excel-документа

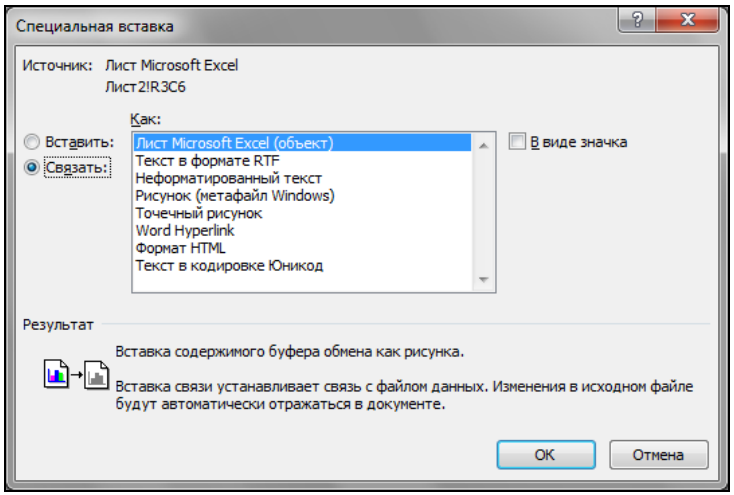


Рис. 12.9. Диалоговое окно **Специальная вставка** — осуществление связи с файлом

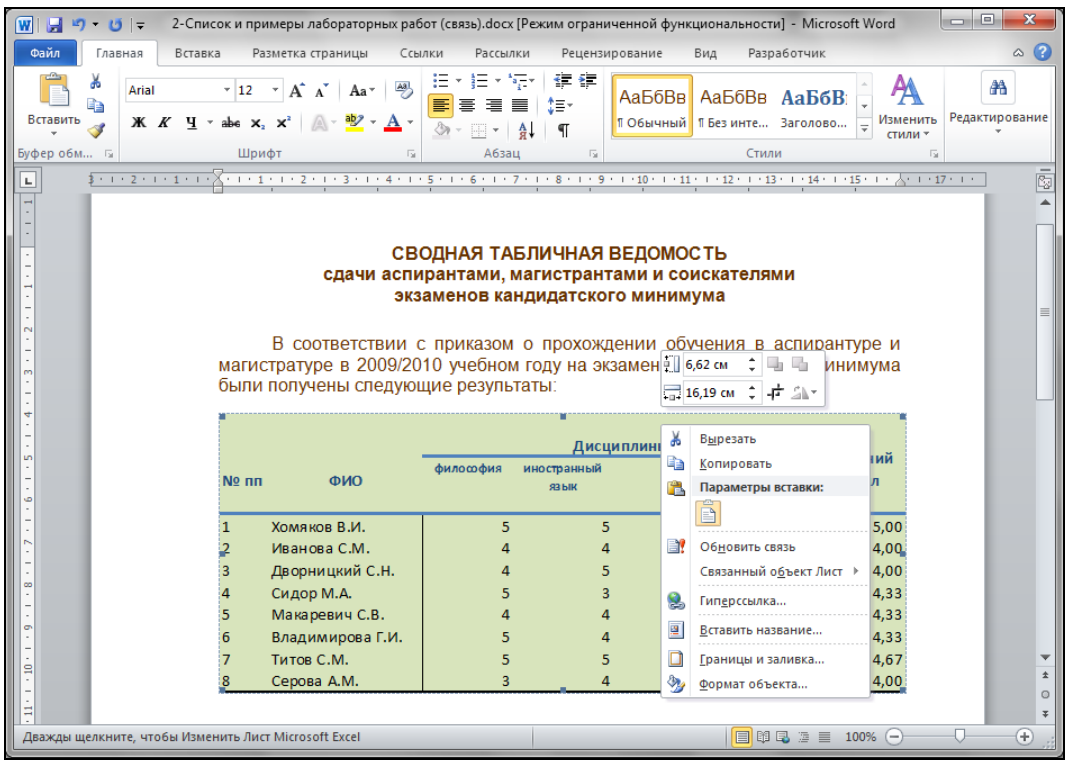


Рис. 12.10. Контекстное меню связанного объекта в документе-клиенте

Приведем последовательность действий при подготовке этого документа.

1. Подготовьте Word-документ (клиент): введите необходимую текстовую информацию и оставьте место для таблицы Excel.
2. Подготовьте Excel-документ (сервер) в соответствии с рис. 12.8 (см. также файл *Табличная ведомость.xlsx* на компакт-диске), используя формулу для вычисления среднего значения.
3. Выделите подготовленную таблицу в документе Excel, перейдите на вкладку **Главная** ленты и в группе **Буфер обмена** щелкните по кнопке **Копировать**.
4. Активизируйте приложение Word и установите курсор в точку вставки связываемых данных.
5. Перейдите на вкладку **Главная** ленты и в группе **Буфер обмена**, нажав кнопку со стрелкой **Вставить**, выберите команду **Специальная вставка**.
6. В появившемся диалоговом окне установите необходимые опции (рис. 12.9) и нажмите кнопку **OK**.
7. Сохраните полученный Word-документ.

> ***Совет***
>
> Для работы со связанным объектом в документе-клиенте удобно использовать его контекстное меню (рис. 12.10).

Управляем объектами с помощью технологии Automation

Automation — это ключевая технология, используемая в большинстве средств разработки корпорации Microsoft. Она дает возможность программно управлять объектами из других приложений, т. е. позволяет разработчикам использовать объекты (и все, что с ними связано) других приложений в качестве компонентов собственных приложений. Одним из ключевых свойств технологии Automation является ее независимость от языка программирования.

Приложения поддерживают технологию Automation одним из двух способов:

- в качестве приложения-источника (или *объекта*, или *сервера Automation*, или *объекта ActiveX*) — объекты такого приложения используются другим приложением или средствами программирования через интерфейс программирования;
- в качестве приложения-приемника (или *контроллера*, или *клиента Automation*) — это приложение управляет объектами приложения-источника.

> ***Примечание***
>
> Некоторые приложения могут быть только приложениями-источниками либо только приложениями-приемниками, но есть и такие, которые могут выступать и в той и в другой роли (например, MS Excel, MS Access).

Для программного управления объектом Automation необходимо:

1. Создать переменную, представляющую собой объект.
2. Использовать эту переменную для доступа к объектам, находящимся в приложениях-источниках. Объекты приложения источника образуют библиотеку *объектов-серверов*.
3. По завершении работы с объектом присвоить переменной значение `Nothing`, освобождая переменную.

Программные идентификаторы приложений-серверов Automation

В табл. 12.1 приведены названия приложений, типов и классов объектов Automation, а также их программные идентификаторы, которые иногда называются ProgID, используемые при программировании объектов Automation. Обратите внимание, что если применяется программный идентификатор без указания версии, то объект создается на основе наиболее современной установленной версии программы. В общем случае идентификатор ProgID имеет следующий синтаксис:

Appname.*ObjectType*

- *Appname* — имя приложения сервера.
- *ObjectType* — тип или класс объектов.

Таблица 12.1. *Имена серверов Automation*

Приложение	Тип объекта	Идентификатор
Excel	Application	Excel.Application
	Workbook	Excel.Sheet
	Workbook	Excel.Chart
Access	Application	Access.Application
	CurrentData	Access.CodeData, Access.CurrentData
	CurrentProject	Access.CodeProject, Access.CurrentProject
Word	Application	Word.Application
	Document	Word.Document, Word.Template
PowerPoint	Application	PowerPoint.Application
Outlook	Application	Outlook.Application

Функции доступа к объектам Automation

Для доступа к объектам Automation приложения-сервера используются две функции: `CreateObject()` и `GetObject()`.

Функция `CreateObject()` возвращает и создает ссылку на объект ActiveX.

```
CreateObject(Class[, Servername])
```

- *Class* — имя объекта Automation.
- *Servername* — параметр используется только при создании объекта Automation в сети и устанавливает имя сервера, где будет создан объект Automation.

Функция `GetObject()` возвращает и создает ссылку на объект ActiveX, сохраненный в файле.

```
GetObject([Pathname] [, Class])
```

- *Pathname* — полное имя файла; если параметр опущен, то необходимо указать значение параметра *Class*.
- *Class* — имя объекта Automation.

Функция `GetObject()` подобна функции `CreateObject()`. Но есть и некоторое различие между ними. Функцию `GetObject()` можно использовать для доступа к существующим документам, хранящимся в файлах, а также и для доступа к объекту `Application` любого уже запущенного приложения MS Office. Для этого надо вызвать функцию `GetObject()` без первого параметра. Этот способ доступа к объекту `Application` любого уже запущенного приложения MS Office применяется, когда нет необходимости в запуске еще одного экземпляра приложения, что происходит при работе функции `CreateObject()`.

Позднее и раннее связывание

Позднее связывание (late binding) происходит, когда тип для переменной, которая будет представлять собой объект Automation, указывается как `Object`.

Таким образом, при позднем связывании переменная, задающая объект Automation, имеет тип `Object`. Тип `Object` позволяет создавать объекты любой природы. В этом смысле он подобен типу `Variant`. Такая чрезмерная общность определения переменной понижает производительность приложения. Для достижения наилучшей производительности приложения необходимо определить конкретный тип для переменной, которая будет представлять собой объект Automation. Например, если используется Excel, то надо установить тип переменной `Excel.Application`.

Второй подход называется *ранним связыванием* (early binding) и происходит на этапе компиляции. При раннем связывании, чтобы определить переменную определенного класса, перед написанием кода, необходимо сослаться на библиотеку объектов серверов Automation. Для этого в редакторе Visual Basic выберите команду **Tools | References**. В появившемся диалоговом окне **References** в списке **Available References** установите флажок подсоединяемой библиотеки объектов, например **Microsoft Word 14.0 Object Library**.

Выполнив предварительные действия, можно перейти к созданию нового экземпляра объекта Automation при помощи ключевого слова `New`. Например:

```
Dim objWorkbook As New Word.Application
```

Совет

Если вы хотите, чтобы при написании кода после набора точки на экране отображался список со свойствами и методами объекта Automation, то используйте раннее связывание. Кроме того, раннее связывание позволяет использовать константы подсоединяемой объектной модели без их объявления.

Предупреждение

К недостаткам раннего связывания можно отнести привязанность создаваемого вами приложения к конкретной версии библиотеки объектов. Выход в свет новой версии продукта, объектную модель которого вы используете посредством раннего связывания, приведет к необходимости повторного распространения и вашего обновленного продукта. Проекты, созданные с помощью позднего связывания, не зависят от новых версий используемых объектных моделей.

Организуем совместную работу Microsoft Excel и Microsoft Word

MS Word является одним из наиболее распространенных средств по подготовке документов и отчетов, когда-либо созданных для Windows, и любой программист, конечно, захочет использовать его возможности в своих проектах. В данном разделе на примерах показано, как, используя объектную модель MS Word, можно создавать отчетную документацию из MS Excel. Надо обратить внимание на то, что перед выполнением программ следует установить ссылку на библиотеку объектов Microsoft Word 14.0 Object Library в окне **References**, отображаемом на экране выбором в редакторе Visual Basic команды **Tools | References**.

Создание нового документа Microsoft Word функцией *CreateObject()*

Покажем (см. файл *1-Пример с Word.xlsm* на компакт-диске), как открывается документ MS Word (в нашем случае файл *Пример.docx* с компакт-диска, который следует расположить в рабочем каталоге проекта). Первоначально программа проверяет наличие файла Пример.docx в рабочем каталоге приложения. Если он отсутствует, то программа информирует об этом пользователя и прерывает свое выполнение. Если же файл есть, то программа с помощью функции `CreateObject()` запускает еще один экземпляр MS Word, даже если MS Word уже был запущен, и в нем открывается файл. При открытии файла на экране отображается диалоговое окно с вопросом "Закрыть документ?". Нажатие кнопки **OK** в этом окне приведет как к закрытию окна, так и созданного экземпляра Word с загруженным файлом. Нажатие кнопки **Нет** оставит документ открытым. Для реализации данного демонстрационного приложения расположите на рабочем листе кнопку, установите значение ее свойства `Name` равным `cmdCreateNew`, а в модуле рабочего листа наберите следующий код (листинг 12.1).

Листинг 12.1. Открытие документа Word при помощи функции `CreateObject()`. Модуль рабочего листа

```
Private Sub cmdCreateNew_Click()
   Dim objWord As Object
   Dim File As String
   File = ThisWorkbook.Path & "\Пример.docx"
   If Dir(File) = Empty Then
      MsgBox "Документ не найден", vbInformation, "Word"
      Exit Sub
   End If
   Set objWord = CreateObject("Word.Application")
   With objWord
      .Visible = True
      .Documents.Open Filename:=File
```

```
   End With
   Select Case MsgBox("Закрыть документ?", vbQuestion + vbYesNo, "Word")
      Case vbYes
         objWord.Quit
         Set objWord = Nothing
      Case vbNo
         Set objWord = Nothing
    End Select
End Sub
```

ПРИМЕЧАНИЕ

При работе с MS Word с помощью технологии Automation необходимо не забыть закрыть в коде приложение, т. к. в противном случае по умолчанию оно остается открытым. Закрытие MS Word производится методом `Quit`. После того как MS Word закрыт, можно уничтожить созданный объект `objWord`.

Открытие документа Microsoft Word функцией *GetObject()*

В данном примере мы не только покажем, как открывается существующий документ MS Word, но и объясним, как проверить, запущен ли уже MS Word. Для этого применим функцию `GetObject()`. Если Word запущен, то для открытия файла можно воспользоваться функцией `GetObject()`. Если же нет, то использование этой функции приведет к генерированию ошибки 429. В таком случае надо применить функцию `CreateObject()`. Кроме того, в нашей программе предусмотрим печать открытого документа, при этом программа первоначально спрашивает пользователя, надо ли печатать документ. Документ печатается, если только пользователь подтвердит необходимость печати. По завершении печати документ закрывается, а также закрывается MS Word, если он был открыт программой для печати документа.

Итак, для выполнения примера: расположите на рабочем листе кнопку, установите значение ее свойства `Name` равным `Печать документа Word`, а в модуле рабочего листа наберите соответствующий код (см. файл *2-Пример с Word.xlsm* на компакт-диске). И, кроме того, не забудьте установить ссылку на библиотеку объектов Microsoft Word 14.0 Object Library.

ПРИМЕЧАНИЕ

Обратите внимание, что данный пример демонстрирует не только то, как выводится на печать документ, но и то, как проверяется, открыт ли уже экземпляр приложения или нет.

Отправка отчета из MS Excel в MS Word

Продемонстрируем, как можно построить отчетный документ MS Word на основе данных, полученных при расчетах в MS Excel. В качестве примера создадим приложение, моделирующее бросание кости (рис. 12.11). Игрок в поле вводит число попыток и нажимает кнопку **Игра**. В список выводится результат игры. При нажатии же кнопки **Отчет** создается новый документ MS Word, в котором на основе данных из списка конструируется таблица результатов игры (рис. 12.12).

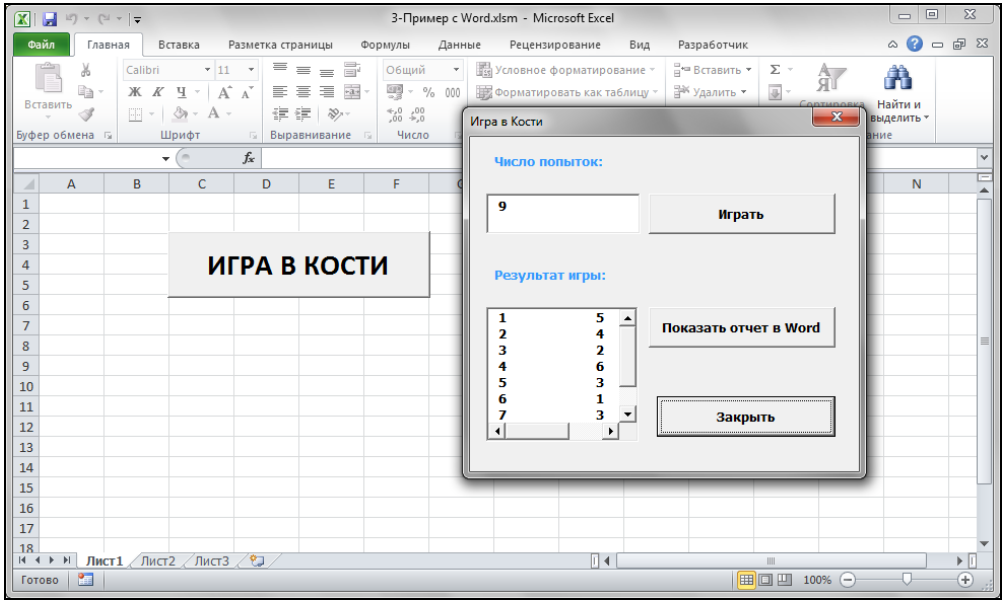
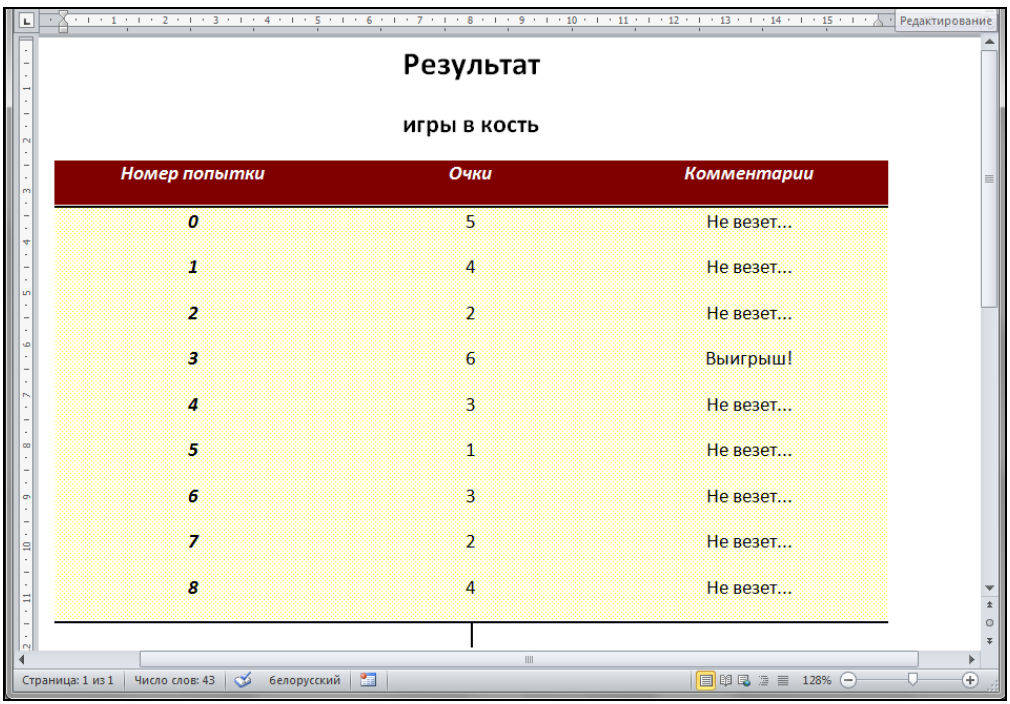


Рис. 12.11. Созданное приложение в Microsoft Office Excel 2010



Результат

игры в кость

Номер попытки	*Очки*	*Комментарии*
0	5	Не везет...
1	4	Не везет...
2	2	Не везет...
3	6	Выигрыш!
4	3	Не везет...
5	1	Не везет...
6	3	Не везет...
7	2	Не везет...
8	4	Не везет...

Рис. 12.12. Созданный отчет из MS Excel в MS Word

Для реализации примера расположите на рабочем листе кнопку, установив ее свойство `Captions` равным `ИГРА В КОСТИ`. Далее создайте форму. Расположите на ней две текстовые надписи, три кнопки, текстовое поле и список. При помощи окна **Properties** установите элементам управления значения свойств, как показано в табл. 12.2.

Таблица 12.2. *Значения свойств, установленные в окне* ***Properties***

Элемент управления	Свойство	Значение
Форма	`Caption`	`Игра`
Кнопка	`Name`	`cmdGame`
	`Caption`	`Игра`
Кнопка	`Name`	`cmdReport`
	`Caption`	`Отчет`
Кнопка	`Name`	`CloseForm`
	`Caption`	`Закрыть`
Список	`Name`	`lstRes`
Поле	`Name`	`txtNum`

Выполните двойной щелчок по созданной форме и в открывшемся окне введите необходимый код (см. файл *3-Пример с Word.xlsm* на компакт-диске). Для корректного функционирования программы установите ссылку на библиотеку объектов Microsoft Word 14.0 Object Library в окне **References**, отображаемом на экране выбором в редакторе Visual Basic команды **Tools | References**.

Добавьте на листе модуля для рабочего листа **Лист1** процедуру, которая будет визуализировать форму (см. также файл *3-Пример с Word.xlsm* на компакт-диске).

Используем Access в качестве сервера автоматизации

MS Access является мощным средством по созданию баз данных и работе с информацией, в них хранящейся. Так, в частности, этот программный продукт весьма эффективен при создании удобного визуального интерфейса по работе с базой данных и составлению отчетов.

В файле *4-Импорт из Access.xlsm*, расположенном на компакт-диске, приведен пример (см. код в стандартном модуле Module1), когда из приложения Excel открывается база данных Access (см. файл *Gustomer.accdb* на компакт-диске), и данные из таблицы экспортируются на активный лист рабочей книги Excel.

Перед тем как создавать эту функцию, необходимо добавить две ссылки в редакторе VBA Excel: на библиотеку Microsoft Access 14.0 Object Library и на Microsoft DAO 3.6 Object Library. Создав объект `Access.Application`, можно получить доступ ко всем остальным объектам Access: таблицам, формам, отчетам. В данной программе сначала создается объект `Recordset` из нужного запроса, а затем в цикле формируется строка заголовков полей, и переносятся все данные. Чтобы правильно

перенеслись даты, выполняется проверка: если тип данных в поле — `Дата/Время`, то соответствующая ячейка на листе Excel форматируется. На рис. 12.13 представлена таблица **Страны** и результат ее экспорта в Excel.

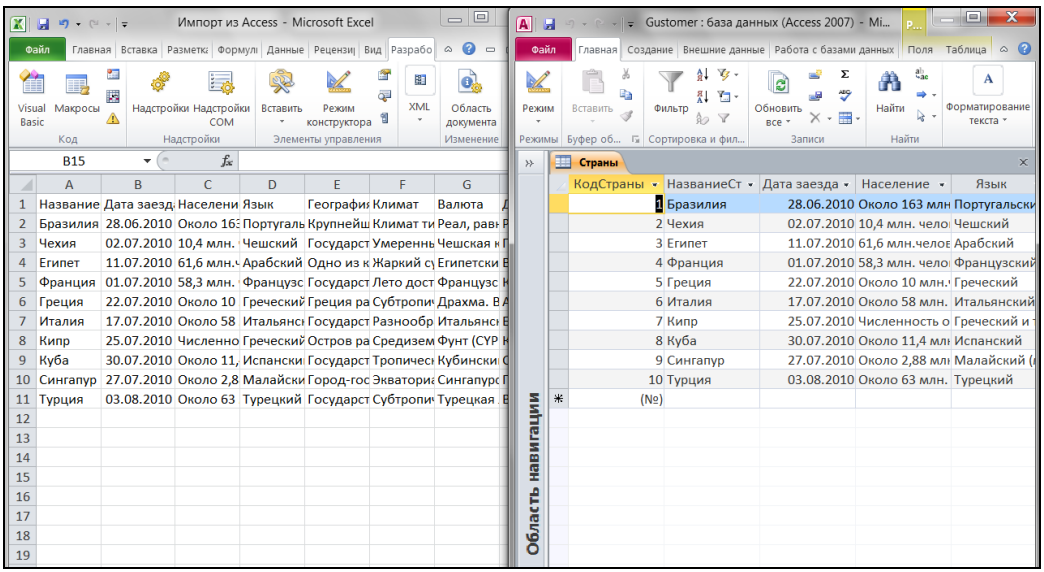


Рис. 12.13. Таблица **Страны** и результат ее экспорта в Excel

В качестве сервера автоматизации Microsoft Office Access применяется обычно для создания мастеров, генерирующих приложения баз данных в Access, при этом можно использовать функции `CreateForm()`, `CreateControl()` и `CreateReportControl()`.

Отправляем сообщения по электронной почте

А теперь рассмотрим пример взаимодействия Microsoft Excel и Outlook. Так, пусть на рабочем листе расположены данные о клиентах, количествах их покупках и соответствующих скидках (рис. 12.14, см. также файл *5-Пример с Outlook.xlsm* на компакт-диске) и кнопка, позволяющая автоматизировать уведомления клиентам.

Нам необходимо сделать соответствующую почтовую рассылку с указанием для каждого клиента их персональных данных (рис. 12.15).

Итак, прежде всего, вам необходимо проверить, чтобы были сделаны соответствующие настройки для вашего аккаунта электронной почты в Microsoft Outlook.

Далее переходите непосредственно к реализации примера. Расположите на рабочем листе рядом с данными кнопку, для которой установите свойство `Caption` равным `Отправить сообщение клиенту`. Для данной кнопки введите соответствующий код (см. модуль рабочего листа **Лист1из** файла *5-Пример с Outlook.xlsm* на компакт-диске).

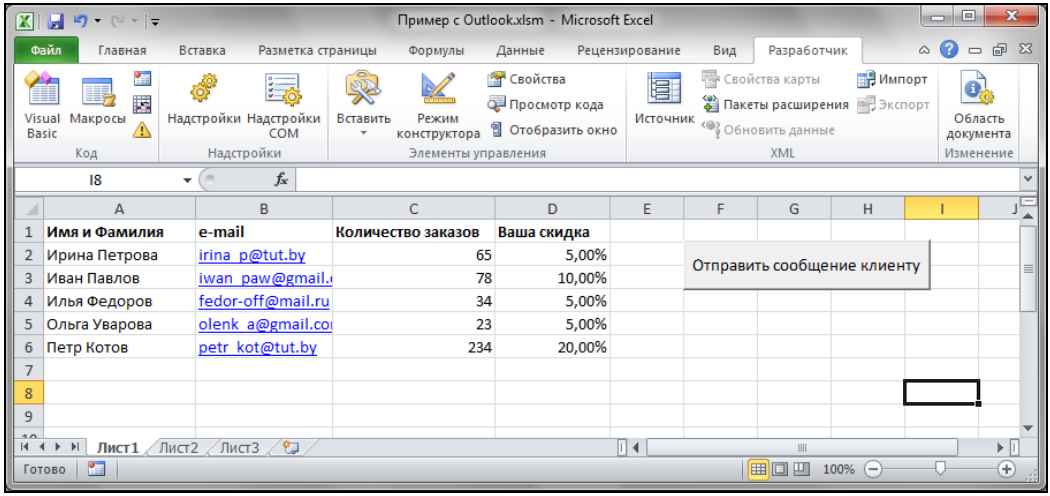


Рис. 12.14. Данные о клиентах на рабочем листе Excel

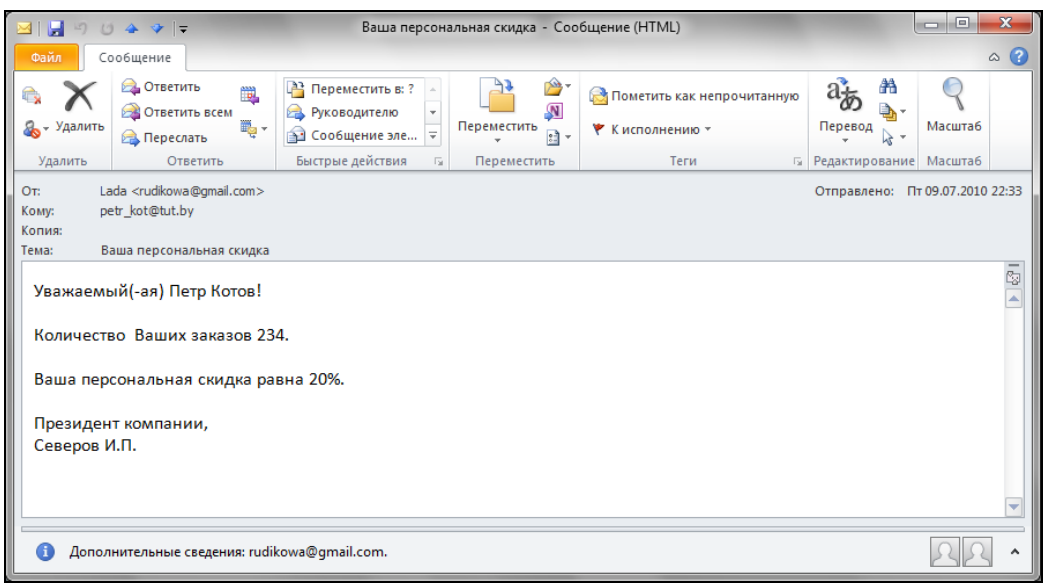


Рис. 12.15. Пример подготовленного письма

А для автоматической отправки сообщений по электронной почте с помощью Outlook введите в стандартном модуле **Module1** необходимый код (см. также файл *5-Пример с Outlook.xlsm* на компакт-диске). Укажите в качестве подключаемой библиотеки Microsoft Outlook 14.0 Object Library.

Примечание

Обратите внимание, что если вы изменили настройки Центра управления безопасностью, то, скорее всего, при запуске вашей программы у вас появится соответствую-

щее окно предупреждения (рис. 12.16). Если вы не хотите, чтобы в дальнейшем выводилось данное окно, сделайте соответствующие настройки в окне **Центр управления безопасностью**.

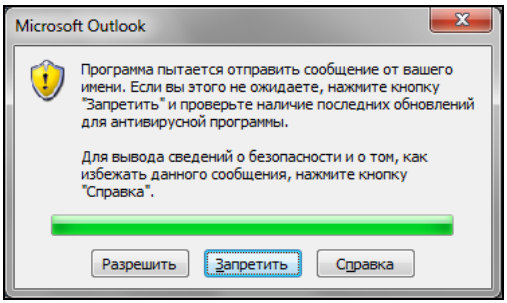


Рис. 12.16. Окно предупреждения Microsoft Outlook

Создаем презентацию в MS PowerPoint

MS PowerPoint является удобным средством по созданию презентаций. Использование же технологии ActiveX позволяет автоматизировать процесс конструирования слайдов для таких презентаций.

Рассмотрим пример, связанный с подготовкой презентации по данным, расположенным на рабочем листе Microsoft Office Excel (рис. 12.17).

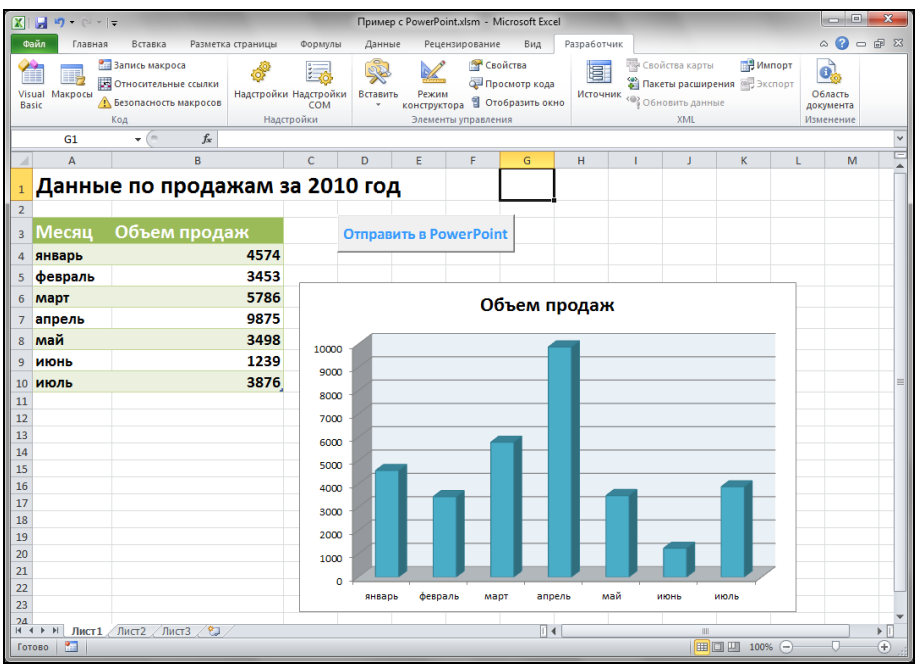


Рис. 12.17. Подготовленные данные в Microsoft Excel для создания презентации

Итак, подготовьте данные на рабочем листе и расположите рядом с ними кнопку, которая будет генерировать презентацию (см. файл *Презентация из Excel.pptx* на компакт-диске). Соответственно в модуле рабочего листа **Лист1** и в стандартном модуле **Module1** введите необходимый код (см. файл *6-Пример с PowerPoint.xlsm* на компакт-диске). Отметим, что процедура из стандартного модуля **Module1** создает презентацию, состоящую из трех слайдов, а затем начинает показ созданной презентации (рис. 12.18).

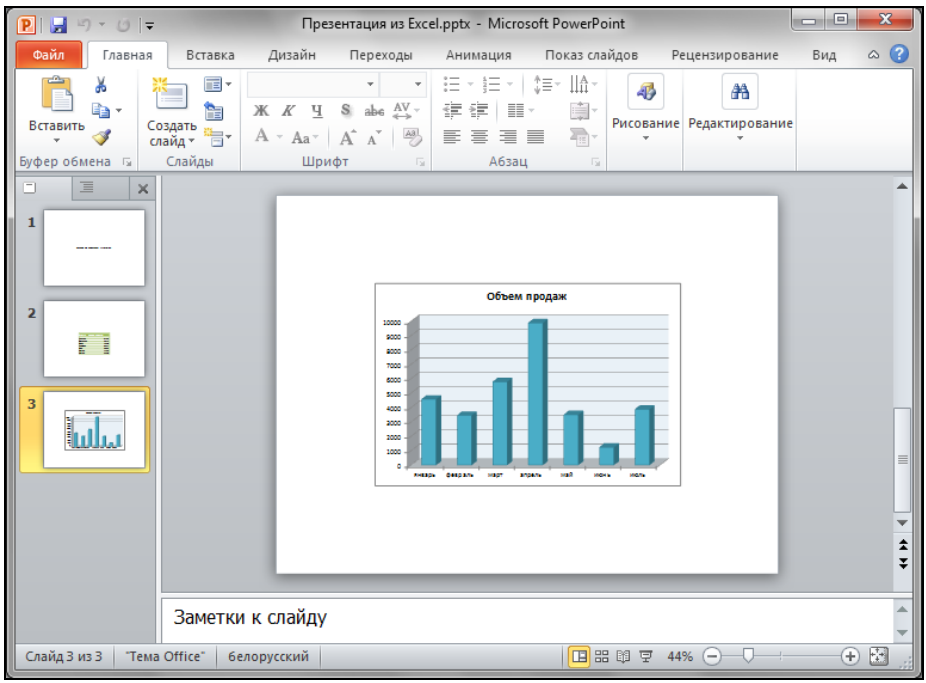


Рис. 12.18. Создание презентации в MS PowerPoint из MS Excel

Наши итоги

Итак, вы познакомились с основными технологиями, которые поддерживают взаимодействие приложений Microsoft Office. Знание этих возможностей, несомненно, поможет вам при подготовке документов различного плана, а также расширит использование документов-серверов в документах-клиентах. Итак, теперь вы можете:

- дать определение основным технологиям, которые используются для обмена информацией между документами приложений Microsoft Office;
- отличить внедренный объект от связанного объекта;
- связывать данные с документом-клиентом;
- внедрять данные других приложений в документ Word;
- использовать технологию ActiveX для программной интеграции различных документов пакета Microsoft Office.

П Р И Л О Ж Е Н И Я

Приложение 1

Краткая справка по Visual Basic for Applications

Язык VBA является объектно-ориентированнм языком программирования. Знание объектных моделей VBA и владение технологией объектно-ориентированного программирования позволяет создавать в Microsoft Office Excel 2010 соответствующие приложения, которые выполняют требуемую обработку и визуализацию данных.

В этом приложении мы рассмотрим базовые элементы объектной модели Microsoft Office Excel 2010: `Application`, `Workbook`, `Worksheet`, `Range`, `Window`, инкапсулирующие в себе данные о самом приложении, рабочей книге, диапазоне и окне. Заметим, что более подробную информацию об объектной модели Microsoft Office Excel 2010 можно получить из имеющейся справки по VBA 7.0 в Microsoft Office Excel 2010: "Excel 2010 Developer Reference: Help and How-to" или на сайте компании Microsoft: **http://msdn.microsoft.com/en-us/library/ee814737**.

> ***Примечание***
>
> Подробное изложение работы с базовыми объектами Microsoft Excel вы можете найти в книге: Гарнаев А. Ю. Micosoft Excel 2002: разработка приложений. — СПб.: БХВ-Петербург, 2003. Отметим, что многие примеры *главы 3* данной вниги легко переносятся в новые версии Microsoft Excel.

Основные понятия объектной модели

Дадим основные понятия, которые необходимы для работы с объектами на языке VBA.

Под *объектом* понимается некоторая абстракция, для которой характерны некоторые собственные признаки и поведение, отличающие рассматриваемый объект от других аналогичных объектов. Объектами Microsoft Office Excel могут являться, например, рабочая книга, рабочий лист, активная ячейка, диаграмма и др.

Абстрактная совокупность однотипных объектов, которые имеют общий набор свойств и обладают одинаковым поведением, определяет *класс*. Как правило, каждый объект представляет собой экземпляр определенного класса.

Свойством называют отдельную характеристику объекта или класса. Вы, например, уже сталкивались со свойствами кнопки, которые являются свойствами объекта `CommandButton`. Свойство объекта может принимать определенное значение. Например, свойство **Тень** (Shadow) может принимать значение `True` или `False`, в зависимости от чего у кнопки будет присутствовать или отсутствовать тень (снизу и справа). Или, например, для диапазона ячеек рабочего листа (объект `Range`)

можно изменить такие свойства, как используемый шрифт (`Font`), содержимое ячеки или диапазона ячеек (`Formula`), значение ячейки (`Value`).

Метод представляет собой процедуру (или функцию) объекта или класса. Совокупность методов объекта определяет его "поведение". Например, объект `Workbook` (рабочая книга) имеет различные методы, например, метод `Save` сохраняет рабочую книгу, а метод `Close` закрывает ее.

Объект может реагировать на определенные *события*, происходящие в процессе работы приложения и влияющие на объект. Совокупность событий, на которые объект способен реагировать, определяется создателем класса, экземпляром которого является данный объект. Например, для кнопки (объект `CommandButton`) событием является ее нажатие (`Click`). Реакцией же объекта на произошедшее событие может быть выполнение данным объектом некоторой специальной процедуры, которая называется *процедурой обработки события*. Любому событию объекта может быть назначена некоторая процедура его обработки.

Для VBA актуальным является также и понятие семейства объектов. *Семейством* называется упорядоченный набор однотипных объектов — экземпляров одного класса. Семейство также является объектом. Одним из методов этого объекта является процедура, возвращающая ссылку на конкретный объект в семействе. Одним из свойств семейства является число объектов, хранящихся в нем. Например, в Microsoft Office Excel семейство `Workbooks` является совокупностью всех открытых в данный момент рабочих книг, семейство `Sheets` — всех листов рабочей книги (включая листы диаграмм), семейство `Worksheets` — всех рабочих листов в данной рабочей книге и т. п.

Объекты и семейства сгруппированы в виде иерархических структур, которые называются *объектными моделями*. В VBA определены специальные объектные модели для каждого компонента семейства Microsoft Office и объектные модели, общие для всех компонентов Microsoft Office. Объектные модели VBA можно изучать, используя справочную систему и окно просмотра объектов **Object Browser** (см., например, рис. П2.9).

Объектная модель Visual Basic для приложений

Объектная модель Visual Basic для приложений представляет собой совокупность независимых объектов, объединенных в одну библиотеку с названием VBA, которые используются всеми приложениями семейства Microsoft Office. Семейства и объекты этой библиотеки представлены в табл. П1.1.

Таблица П1.1. *Описание объектов Visual Basic для приложений*

Объект	Тип	Описание
`Collection`	Объект из библиотеки VBA	Упорядоченная совокупность объектов, с которой можно обращаться как с единым объектом
`Debug`	Объект	Позволяет выводить текущую информацию в окно отладки непосредственно во время выполнения кода VBA

Таблица П1.1 *(окончание)*

Объект	Тип	Описание
`Dictionary`	Объект из библиотеки Scripting	Объект-пара — ключ и элемент. Представляет собой аналог элемента ассоциативной памяти
`Drives`	Семейство из библиотеки Scripting	Содержит объекты `Drive`, предоставляющие информацию (только для чтения) обо всех доступных дисках. Является свойством объекта `FileSystemObject`. Каждый объект предоставляет доступ к свойствам конкретного локального или сетевого диска
`Err`	Объект из библиотеки VBA	Предназначен для обработки ошибок сервера Automation и ошибок модулей VBA во время выполнения кода VBA
`Files`	Семейство из библиотеки Scripting	Содержит объекты `File` и представляет собой совокупность всех файлов в данной папке. Является свойством объекта `FileSystemObject`. Объект `File` предоставляет доступ ко всем свойствам файла на диске
`FileSystemObject`	Объект из библиотеки Scripting	Предоставляет доступ к файловой системе компьютера
`Folders`	Семейство из библиотеки Scripting	Содержит объекты `Folder` и представляет собой совокупность всех папок внутри данной папки. Является свойством объекта `Folder` (свойство называется `SubFolders`). Каждый объект `Folder` предоставляет доступ ко всем свойствам папки на диске
`TextStream`	Объект из библиотеки Scripting	Обеспечивает последовательный доступ к текстовому файлу
`UserForms`	Семейство из библиотеки VBA	Содержит объекты `Object`, соответствующие объектам `UserForm`, и представляет собой совокупность пользовательских форм, загруженных в данный момент в приложение. Это семейство является свойством объекта `Global` из библиотеки VBA

Объектная модель Microsoft Office 2010

Управление приложениями семейства Microsoft Office 2010 осуществляется интерактивно (с помощью интерфейса пользователя) или программно (с помощью объектных моделей). Каждый из компонентов Microsoft Office предоставляет свои объектные модели в виде одноименной библиотеки объектов (файл с расширением olb), которая может быть использована в других приложениях. Microsoft Office Excel 2010, как компонент Microsoft Office, имеет свою библиотеку — Microsoft Excel 14.0 Object Library. По умолчанию, в Microsoft Office Excel 2010 доступны следующие объектные модели, реализованные в нескольких библиотеках:

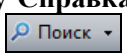
- библиотека объектов Microsoft Excel 14.0 Object Library — основная библиотека документов Excel, в которой хранится класс, задающий корневой объект `Application`, а также все вложенные классы объектов;

- библиотека объектов Microsoft Office 14.0 Object Library — библиотека объектов, общих для всех приложений Microsoft Office 2010;
- библиотека объектов OLE Automation (Stdole) — библиотека классов, позволяющая работать с OLE-объектами и реализовать связь и внедрение объектов;
- библиотека Visual Basic for Applications — библиотека классов языка VBA, включающая все стандартные функции и константы, встроенные в язык, классы `Collection` и `Object Browser`;
- проект VBAProject — проект, связанный с документом, в котором доступны созданные классы, методы, свойства, а также объекты классов стандартных библиотек.

Кроме этого, в Microsoft Office Excel 2010 используются при необходимости также и другие библиотеки.

Объектная модель Microsoft Office Excel 2010

Объектная модель Microsoft Office Excel 2010 представляет собой иерархию объектов, подчиненных одному объекту `Application`, который соответствует самому приложению MS Excel. Многие из этих объектов собраны в библиотеке объектов Microsoft Excel 14.0 Object Library, однако некоторые из них, например объект `Language Settings`, входят в библиотеку объектов Microsoft Office 14.0 Object Library, которая является общей для всех офисных приложений. Каждый объект из библиотеки Excel имеет в качестве свойства объект `Application` (в том числе и сам объект `Application` имеет свойство `Application`), который ссылается на активное приложение Microsoft Office Excel.

На рис. П1.1—П1.5 собраны основные компоненты объектной модели MS Excel. На этих рисунках имена единичных объектов находятся в прямоугольниках с более темным фоном. Имена объектов из семейств помещены в скобки после имени семейства. Объекты, справа от которых находятся треугольные стрелки, дополнительно раскрыты на других схемах. Объекты, которые скрыты в версии Microsoft Office Excel 2010, обведены рамкой. Следует отметить, что в данной версии появились и новые объекты и коллекции, Для того чтобы увидеть их полный список, перейдите к окну **Справка Excel**, щелкните по выпадающему списку возле кнопки **Область поиска** 

и выберите команду **Справка для разработчиков**. Далее отобразите в окне **Справка Excel** оглавление, используя кнопку **Показать оглавление**  на панели инструментов. В открывшемся слева оглавлении выберите раздел **What's New**, а в нем категорию — **New Objects, Collections, and Enumerations**. Справа отобразится весь список новых объектов, появившихся в версии Microsoft Office Excel 2010.

Дадим представление об общей схеме объектной модели.

Иерархия объектов Microsoft Office Excel 2007 образована так:

- каждый объект может содержать набор свойств, часть из которых также может являться ссылками на другие объекты;
- в каждый новый уровень иерархии входят объекты, ссылки на которые хранятся в объектах, расположенных на предыдущем уровне иерархии.

Если свойство объекта представляет собой ссылку на объект, определенный в другой библиотеке (не в библиотеке Excel), то для него приводится название этой библиотеки.

Исходя из схем объектной модели, можно определить, какие объекты описывают приложение, как они связаны между собой и как составить ссылку для доступа к конкретному объекту.

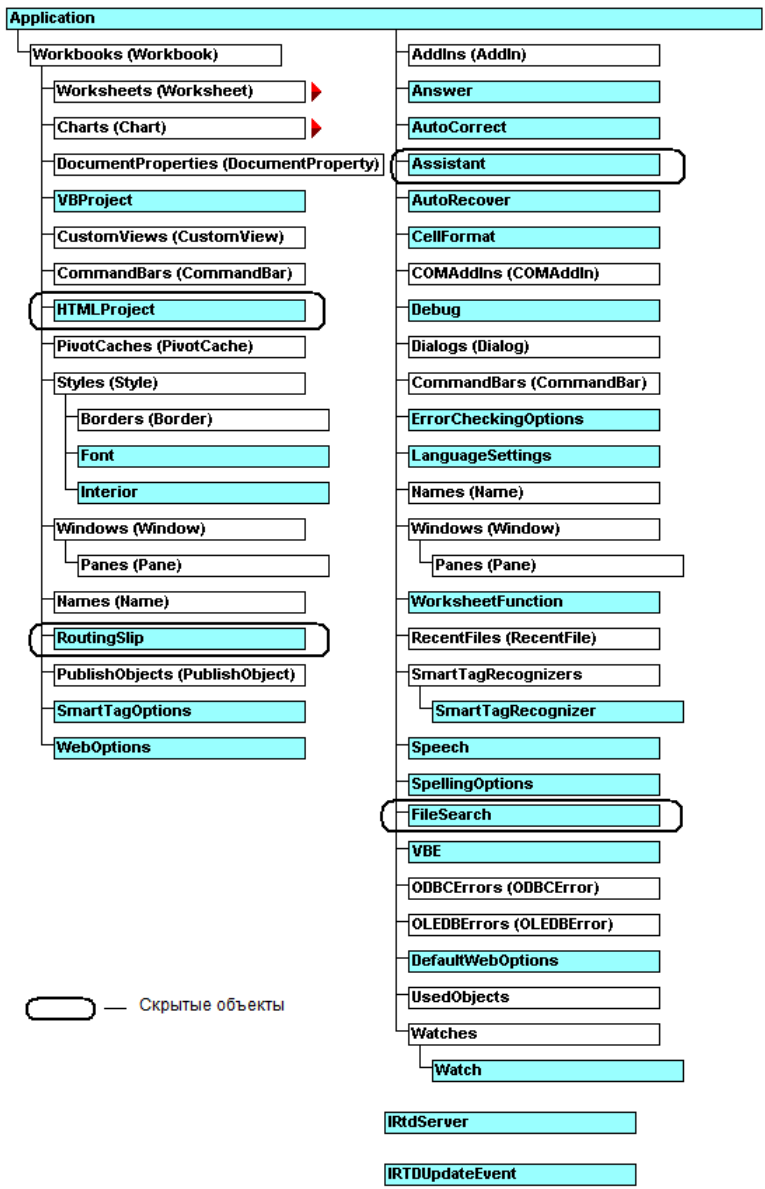


Рис. П1.1. Объект `Application` и непосредственно подчиняющиеся ему объекты

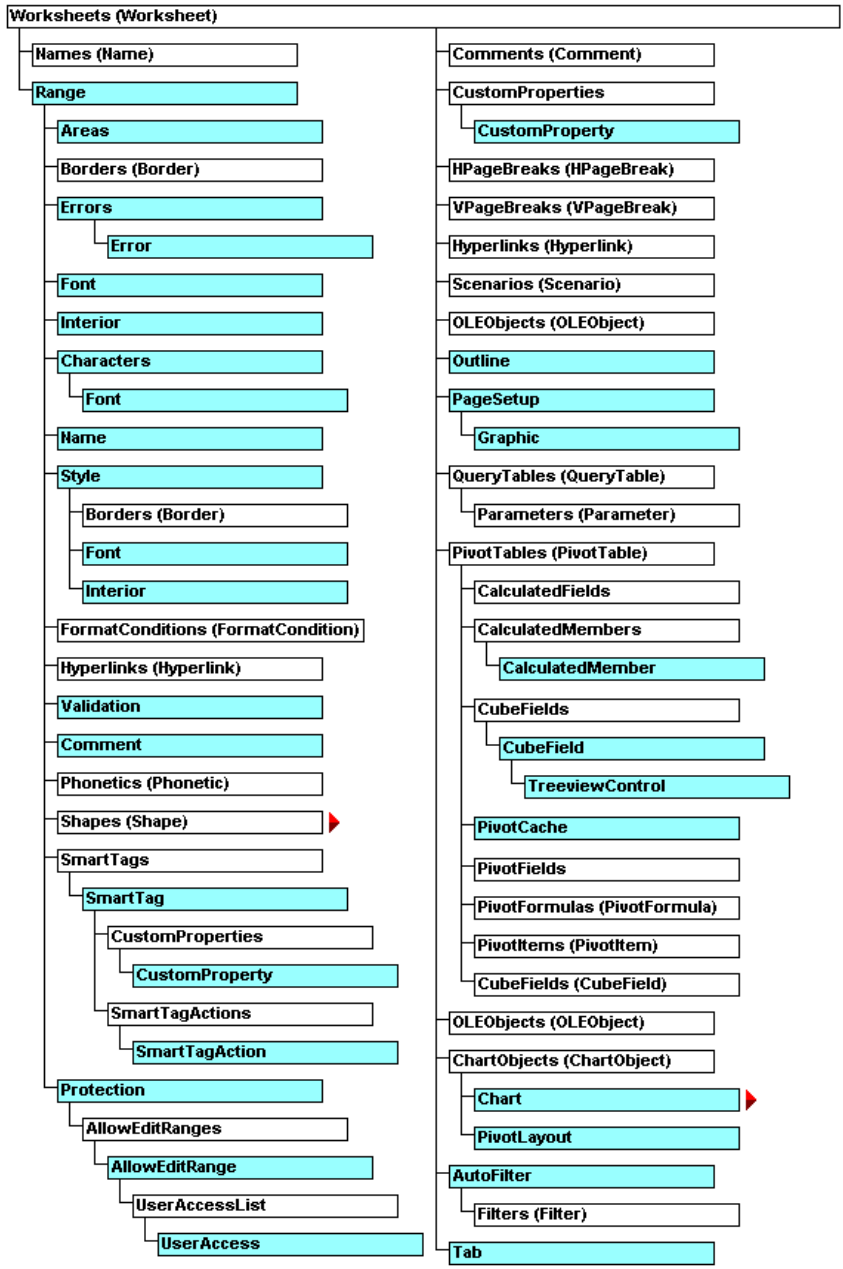


Рис. П1.2. Объект `Worksheet` и непосредственно подчиняющиеся ему объекты

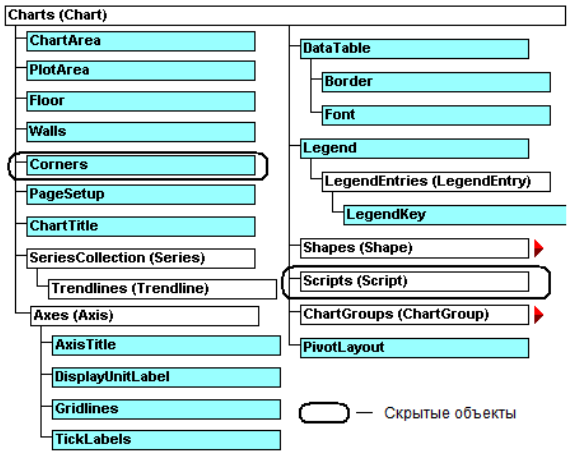


Рис. П1.3. Объект `Chart` и непосредственно подчиняющиеся ему объекты

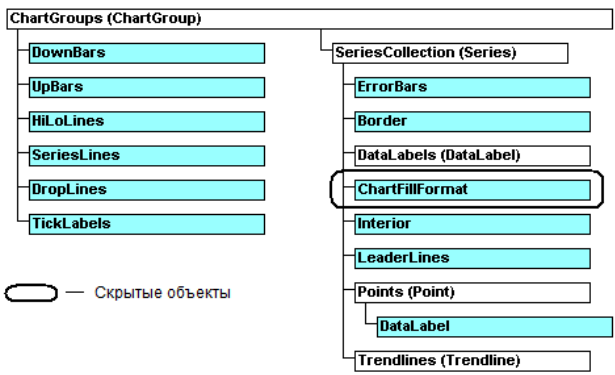


Рис. П1.4. Объект `ChartGroup` и непосредственно подчиняющиеся ему объекты

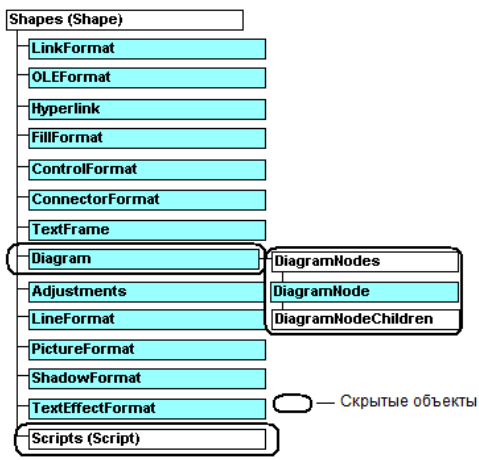


Рис. П1.5. Объект `Shape` и непосредственно подчиняющиеся ему объекты

Полная и неявная ссылка на объект

Полная ссылка на объект состоит из ряда имен вложенных последовательно друг в друга объектов. Разделителями имен объектов в этом ряду являются точки, ряд начинается с объекта `Application` и заканчивается именем самого объекта. Например, полная ссылка на ячейку **A1** рабочего листа **Продажи** рабочей книги с именем **Архив** имеет вид:

```
Application.Workbooks("Архив").Worksheets("Продажи").Range("A1")
```

Приводить каждый раз полную ссылку на объект совсем не обязательно. Обычно достаточно ограничиться только неявной ссылкой на объект. В неявной ссылке, в отличие от полной, объекты, которые активны в данный момент, как правило, можно опускать. В рассмотренном случае, если ссылка на ячейку **A1** дана в программе, выполняемой в среде MS Excel, то ссылка на объект `Application` может быть опущена, т. е. достаточно привести относительную ссылку:

```
Workbooks("Архив").Worksheets("Продажи").Range("A1")
```

Если в данном примере рабочая книга **Архив** является активной, то ссылку можно сократить еще:

```
Worksheets("Продажи").Range("A1")
```

В случае, когда и рабочий лист **Продажи** активен, то в относительной ссылке вполне достаточно ограничиться упоминанием только диапазона **A1**:

```
Range("A1")
```

Объект *Application* и его некоторые свойства

Объект `Application` — это главный (корневой) объект в иерархии объектов MS Excel, представляющий само приложение MS Excel. Он имеет огромное число свойств и методов, которые позволяют установить общие параметры приложения MS Excel. Рассмотрим некоторые примеры.

Ссылка на активную рабочую книгу, лист, ячейку, диаграмму и принтер

Свойства `ActiveWorkbook`, `ActiveSheet`, `ActiveCell`, `ActiveChart`, `ActivePrinter` объекта `Application` возвращают активную рабочую книгу, лист, ячейку, диаграмму и принтер. Объект `ActiveCell` содержится в `ActiveSheet`, а объекты `ActiveSheet` и `ActiveChart` в `ActiveWorkbook`. В следующем примере (листинг П1.1) в активной ячейке устанавливается красный цвет фона с зеленым полужирным шрифтом и в нее вводится строка текста `Май`. Активный рабочий лист переименовывается в **Отчет за май**, а цвет его ярлыка назначается желтым.

Для проверки приведенного здесь листинга:

1. Перейдите к окну редактора Visual Basic.
2. В окне редактора VBA добавьте лист стандартного модуля, выполнив команду **Insert | Module**.

3. Наберите в окне модуля код примера (листинг П1.1).
4. Выполните команду меню **Run | Run Sub/UserForm** либо нажмите соответствующую кнопку **Run | Run Sub/UserForm**  на панели инструментов **Standard**, либо нажмите клавишу <F5>.
5. Вернитесь в открытую книгу Microsoft Excel, нажав, например, кнопку **View Vicrosoft Excel**  на панели инструментов **Standard**, и проверьте результат выполнения программы.

Листинг П1.1. Ссылка на активный рабочий лист и ячейку

```
Sub DemoActives()
   With ActiveSheet
      .Tab.ColorIndex = 27 ' Желтый цвет
      .Name = "Отчет за май"
   End With
   With ActiveCell
      .Font.Bold = True
      .Font.Color = vbGreen
      ' vbGreen — встроенная константа, задающая зеленый цвет
      .Value = "Май"
      .Interior.Color = vbRed
      ' vbRed — встроенная константа, задающая красный цвет
   End With
End Sub
```

Инсталлированные надстройки

Свойство `AddIns` объекта `Application` возвращает семейство инсталлированных надстроек. Например, код из листинга П1.2 проверяет, установлены ли следующие надстройки: **Пакет анализа — VBA** и **Поиск решения**.

Напомним, что необходимые надстройки можно инсталлировать в окне **Параметры Excel**: перейдите на вкладку **Файл** ленты, щелкните по кнопке **Параметры**. В открывшемся окне **Параметры Excel** (рис. П1.6) выберите слева категорию **Надстройки**, справа в группе **Надстройки** приведен полный список активных в данный момент и неактивных надстроек. В случае, если вам необходимо добавить неактивную надстройку, выделите ее мышью в списке и нажмите кнопку **Перейти**. В результате откроется окно **Надстройки** (рис. П1.7), в котором следует поставить флажок перед требуемой настройкой и нажать кнопку **OK** для запуска процесса инсталляции надстройки.

Установленные надстройки будут находиться на вкладке ленты **Данные** в группе команд **Анализ** (см., например, рис. П1.8).

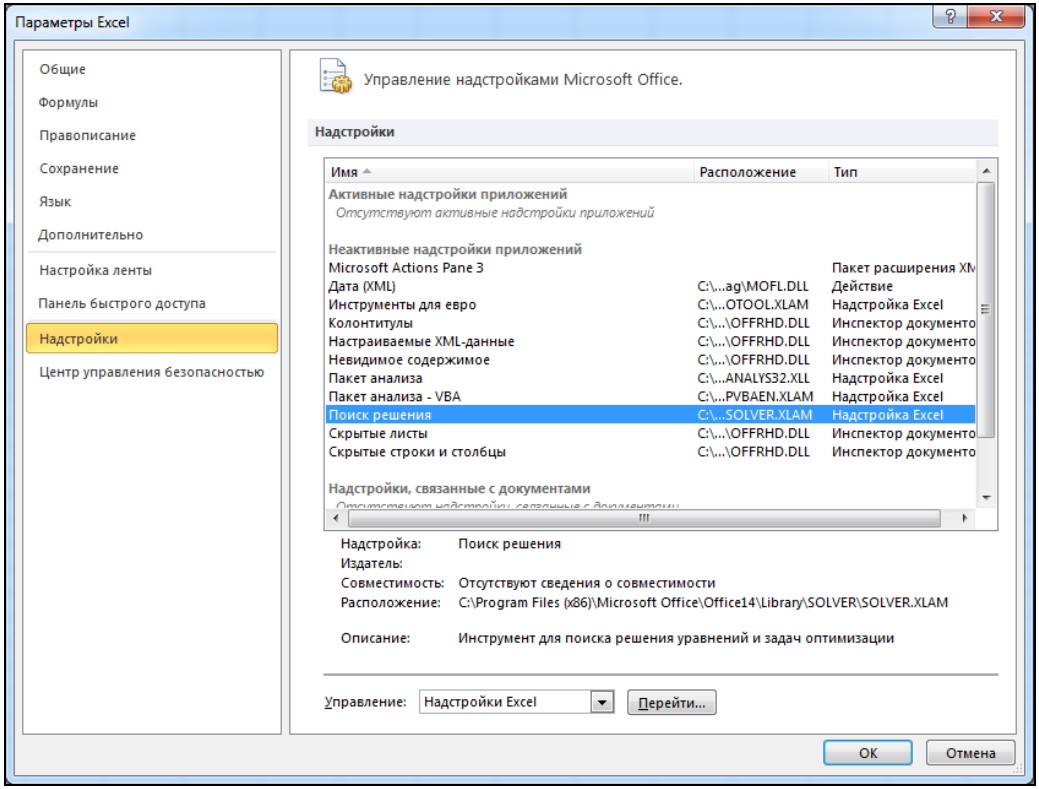


Рис. П1.6. Окно **Параметры Excel**, категория **Надстройки**

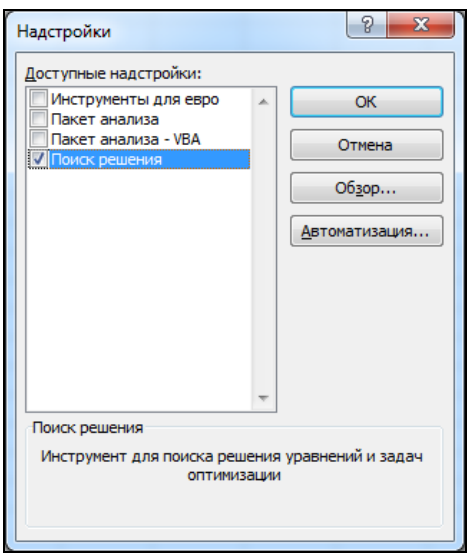


Рис. П1.7. Окно **Надстройки**

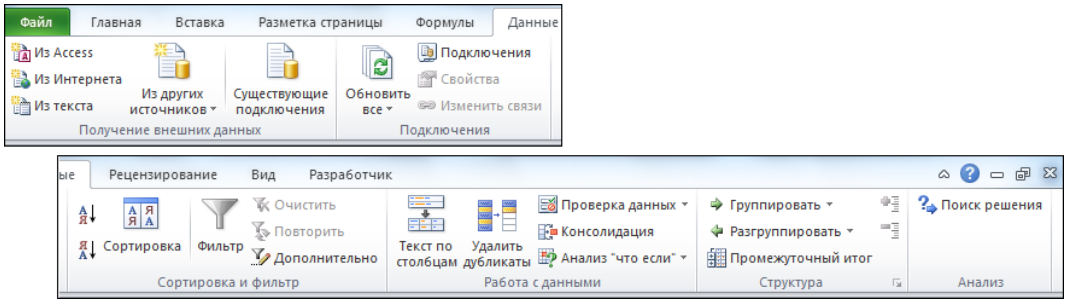


Рис. П1.8. Вкладка **Данные** на ленте

Листинг П1.2. Проверка установленных надстроек

```
Private Sub CB_1_Click()
If Application.AddIns("Пакет анализа - VBA").Installed = True Then
   MsgBox "Надстройка Пакет анализа - VBA инсталлирована"
Else
   MsgBox "Надстройка Пакет анализа - VBA не инсталлирована"
End If
If Application.AddIns("Поиск решения").Installed = True Then
   MsgBox "Надстройка Поиск решения инсталлирована"
Else
   MsgBox "Надстройка Поиск решения не инсталлирована"
End If
End Sub
```

Итак, для проверки, установлены ли у вас надстройки **Поиск решения** и **Пакет анализа — VBA**, выполните следующие действия.

1. В окне Microsoft Excel перейдите на вкладку **Разработчик** ленты, в группе команд **Элементы управления** щелкните по кнопке со стрелкой **Вставить** и из раскрывшегося списка выберите элемент управления ActiveX — **Кнопка**.
2. Нарисуйте на рабочем листе кнопку необходимых размеров и, воспользовавшись кнопкой **Свойства**, расположенной в группе команд **Элементы управления** на вкладке **Разработчик** ленты, установите следующие свойства для элемента управления **Кнопка** (CommantButton): `Name` — `CB_1`, `Caption` — `Проверка инсталяции надстроек`.
3. Выполните двойной щелчок мыщью по элементу управления **Кнопка**, расположенному на рабочем листе, и перейдите к окну редактора Visual Basic.
4. Наберите в окне модуля листа код примера (листинг П1.2).
5. Вернитесь в открытую книгу Microsoft Excel, нажав, например, кнопку **View Vicrosoft Excel**  на панели инструментов **Standard**.
6. Отключите режим конструктора, нажав соответствующую кнопку **Режим конструктора** в группе **Элементы управления** на вкладке **Разработчик** ленты.

7. Проверьте результат выполнения программы, нажав кнопку **Проверка инсталляции надстроек**, находящуюся на рабочем листе Microsoft Excel.

Свойство `AlertBeforeOverwriting` объекта `Application` позволяет управлять отображением диалогового окна с предупреждением при завершении операции drag-and-drop в непустой ячейке. Это свойство программирует установку или снятие флажка **Предупреждать перед перезаписью ячеек** в окне **Параметры Excel** категории **Дополнительно** в группе **Параметры правки** (рис. П1.9).

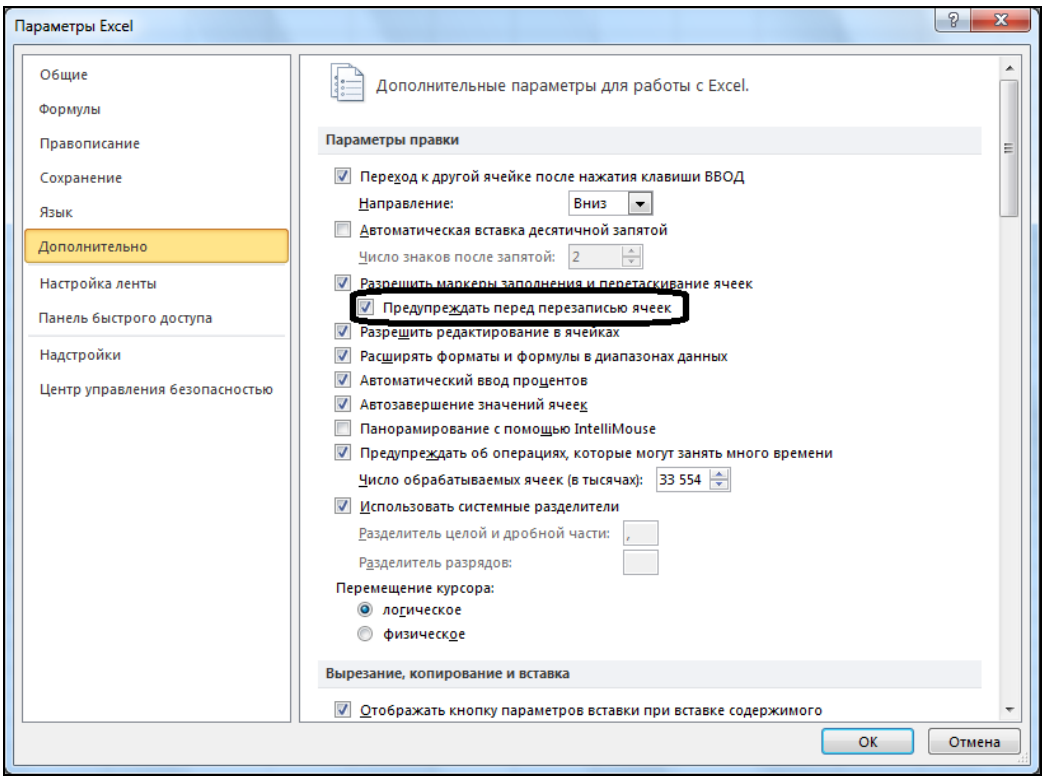


Рис. П1.9. Окно **Параметры Excel**, категория **Дополнительно**

Диапазон ячеек

Свойство `Cells` объекта `Application` возвращает объект типа `Range`, т. е. диапазон ячеек. У этого свойства допустимы два варианта синтаксиса — с параметрами и без них. Без параметров свойство возвращает все ячейки диапазона, а с параметрами — ячейку, стоящую в указанных строке и столбце. В следующем демонстрационном примере (листинг П1.3) у всех ячеек активного рабочего листа устанавливаются атрибуты шрифта — название (свойство `Name`), цвет (свойство `ColorIndex`) и размер (свойство `Size`), кроме того, в ячейку **A1** вводится значение `Алиса`, а в ячейку **A2** — `в стране чудес`.

Листинг П1.3. Работа с диапазоном ячеек

```
Sub DemoCells()
   With Application.Cells.Font
      .Name = "Arial"
      .ColorIndex = 3
      .Size = 10
   End With
   Application.Cells(1, 1).Value = "Алиса"
   Application.Cells(1, 2).Value = "в стране чудес"
End Sub
```

Столбцы и строки рабочего листа

Свойство `Columns` объекта `Application` возвращает объект типа `Range`, который в данном контексте представляет собой набор столбцов. У этого свойства допустимы два варианта синтаксиса — с параметром и без него. Без параметра свойство возвращает все столбцы диапазона, а с параметром — столбец со специфицированным номером.

Свойство `Rows` аналогично свойству `Columns`, но возвращает не столбцы, а строки.

Следующий код показывает, как, используя свойства `Columns` и `Rows`, можно во всех ячейках указанных столбца и строки задать стиль шрифта (в данном случае — полужирный), а также ввести одно и то же значение (1) за одну операцию.

```
Sub DemoColumns1()
   Application.Columns(1).Font.Bold = True
   Application.Columns(1).Value = 1
   Application.Rows(1).Font.Bold = True
   Application.Rows(1).Value = 1
End Sub
```

Установка заголовка окна MS Excel

Свойство `Caption` объекта `Application` возвращает или устанавливает текст из заголовка главного окна MS Excel. Установка значения свойства равным `Empty` (пустая строка) возвращает заголовок, используемый по умолчанию. В следующем примере (листинг П1.4) первая процедура обрабатывает событие `Open` рабочей книги и устанавливает в качестве заголовка окна приложения текст `Отчет за 2011 год`, а вторая — обрабатывает событие `BeforeClose` рабочей книги и возвращает окну имя, используемое по умолчанию, т. е. `Приложение_3` (рис. П1.10).

Листинг П1.4. Установка заголовка окна MS Excel. Модуль `ЭтаКнига`

```
Private Sub Workbook_Open()
   Application.Caption = "Отчет за 2011 год"
End Sub
Private Sub Workbook_BeforeClose(Cancel As Boolean)
   Application.Caption = Empty
End Sub
```

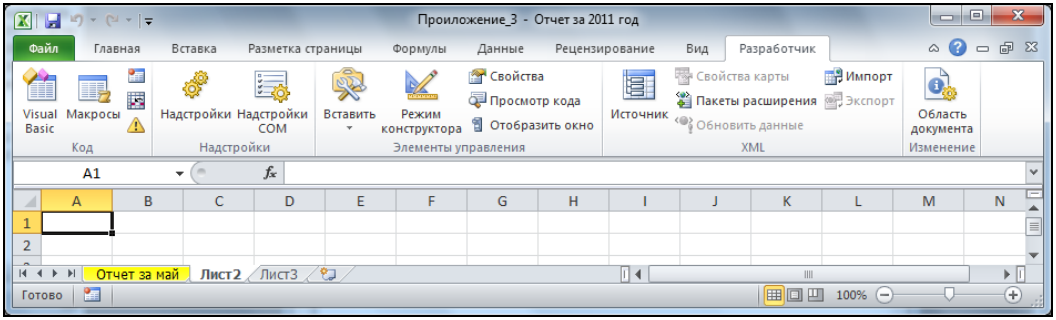


Рис. П1.10. Окно Microsoft Office Excel с пользовательским заголовком

Семейство встроенных диалоговых окон

Свойство `Dialogs` возвращает семейство `Dialogs` всех встроенных диалоговых окон. Параметр этого семейства идентифицирует окно. Метод `Show` отображает его на экране, а параметры этого метода задают опции, специфицируемые в отображаемом окне. Метод `Show` возвращает значение `True`, если задача, поставленная в отображаемом окне, была выполнена успешно. Например, приводимый далее код (листинг П1.5) отображает окно **Открытие документа** для открытия книги D:\test.xlsx с последующим сообщением о выбранных вами действиях.

Листинг П1.5. Открытие книги при помощи встроенного окна

```
Sub DemoDialogs()
   Dim idx As Long
   idx = Application.Dialogs(xlDialogOpen).Show("d:\test.xlsx")
   If idx Then
      MsgBox "Файл открыт"
   Else
      MsgBox "Файл не открыт"
   End If
End Sub
```

В свою очередь, следующий пример демонстрирует закрытие рабочей книги без вывода соответствующего предупреждения и без сохранения сделанных изменений (листинг П1.6).

Листинг П1.6. Закрытие рабочей книги без сохранения внесенных изменений

```
Private Sub Workbook_Close()
  Application.DisplayAlerts = False
  Workbooks("test2.XLSX").Close
  Application.DisplayAlerts = True
End Sub
```

Отображение строки формул, полосы прокрутки и строки состояния

Свойства `DisplayFormulaBar`, `DisplayScrollBars` и `DisplayStatusBar` объекта `Application` устанавливают, надо ли отображать строку формул, полосу прокрутки и строку состояния. Например, приведенный в листинге П1.7 код демонстрирует скрытие строки состояния текущей рабочей книги Excel. При изменении в листинге значения соответсвующего свойства на `True` строка состояния вновь появится внизу рабочей книги.

Листинг П1.7. Скрытие строки состояния рабочей книги Microsoft Office Excel

```
Sub StatusBar()
  AsaveStatusBar = Application.DisplayStatusBar
  Application.DisplayStatusBar = Fulse
End Sub
```

Полноэкранное отображение рабочего листа

Свойство `DisplayFullScreen` объекта `Application` управляет отображением рабочего листа в полноэкранном виде. Например, следующий код (листинг П1.8) обеспечивает полноэкранное отображение окна при щелчке правой кнопкой мыши на ячейке рабочего листа.

Листинг П1.8. Полноэкранное отображение рабочего листа. Модуль `ЭтаКнига`

```
Private Sub Workbook_SheetBeforeRightClick(ByVal Sh As Object, _
                                           ByVal Target As Range, _
                                           Cancel As Boolean)
   Cancel = True
   Application.DisplayFullScreen = True
End Sub
```

Установка высоты и ширины окна приложения

Свойства `Height` и `Width` объекта `Application` устанавливают высоту и ширину окна приложения. Эти свойства не могут быть заданы, если значение свойства `WindowState` равно `xlMaximized`.

Семейство всех имен активной рабочей книги

Свойство `Names` объекта `Application` возвращает семейство `Names` всех имен активной рабочей книги. Как и у любого семейства, свойство `Count` семейства `Names` указывает число его элементов. Свойство `Name` возвращает имя конкретного элемента семейства, а свойство `RefersToRange` — объект `Range`, для которого установлено имя. В примере, представленном далее, у первого листа рабочей книги определяется семейство всех заданных ему имен, после чего они и соответствующие им

ссылки на диапазоны выводятся в ячейки первого и второго столбцов рабочей книги (листинг П1.9).

Листинг П1.9. Семейство всех имен активной рабочей книги

```
Sub DemoNames()
   Dim nms As Names
   Dim wks As Worksheet
   Dim i As Integer
   Set nms = ActiveWorkbook.Names
   Set wks = Worksheets(1)
   If nms.Count = 0 Then
      MsgBox "Нет имен на рабочем листе" & wks.Name
   Else
      For i = 1 To nms.Count
         wks.Cells(i, 1).Value = nms(i).Name
         wks.Cells(i, 2).Value = nms(i).RefersToRange.Address
      Next
   End If
End Sub
```

Ссылка на выбранный объект

Вернуть ссылку на выбранный объект, например рабочий лист, диапазон, диаграмму, позволяет свойство `Selection`. Таким образом, тип элемента, возвращаемого свойством `Selection`, зависит от того, что было выбрано. Например, далее показано, что при помощи метода `Select` сначала выбирается диапазон **A1:A10**, затем свойство `Selection` возвращает ссылку на этот диапазон и, в заключение, методом `Clear` этот диапазон очищается.

```
Sub Clr()
  Worksheets(1).Range("A1:C10").Select
  Selection.Clear
End Sub
```

Методы объекта *Application*

У объекта `Application` имеется большая коллекция методов, позволяющих производить различные действия — от конвертации метрических единиц измерения до создания таймера. Основными методами объекта `Application` являются следующие:

- `ActivateMicrosoftApp`;
- `Calculate`;
- `CentimetersToPoints`;
- `CheckSpelling`;
- `ConvertFormula`;
- `Evaluate`;
- `Help`;
- `InchesToPoints`;
- `InputBox`;
- `Intersect`;
- `OnKey`;
- `OnTime`;
- `Quit`;
- `Run`;
- `Save`;
- `Union`;
- `Volatile`;
- `Wait`.

События объекта *Application*

Прежде чем применить событие объекта `Application`, необходимо создать модуль класса, там объявить объект типа `Application` и при этом объявлении использовать ключевое слово `WithEvents`. Например, так сделано в следующем коде (листинг П1.10, *а*). Кроме того, переименуйте модуль класса из `Class1`, например, в `MyAppWithEvent`.

Листинг П1.10, *а*. Объявление объекта типа `Application`. Модуль класса

```
Public WithEvents App As Application
```

После этого имя объекта `App` будет добавлено в список объектов, приводимый в списке **General** модуля класса. В списке **Declarations** приводятся ассоциированные с ним события. Например, следующий код (листинг П1.10, *б*) реализует обработку двух событий: `WorkbookActivate` и `WorkbookDeactivate`, генерируемых при активизации и деактивизации рабочей книги.

Листинг П1.10, *б*. Процедуры, обрабатывающие события объекта типа `Application`. Модуль класса

```
Private Sub App_WorkbookActivate(ByVal Wb As Workbook)
   MsgBox "Книга " & Wb.Name & " активизирована"
End Sub
Private Sub App_WorkbookDeactivate(ByVal Wb As Workbook)
   MsgBox "Книга " & Wb.Name & " деактивизирована"
End Sub
```

Теперь, для того чтобы процедура, обрабатывающая событие, исполнялась, остается объявить объект типа `MyAppWithEvent`, например, как это делается в следующим коде (листинг П1.10, *в*).

Листинг П1.10, *в*. Объявление объекта типа `MyAppWithEvent`. Модуль `ЭтаКнига`

```
Dim a As New MyAppWithEvent
Private Sub Workbook_Open()
   Set a.App = Application
End Sub
```

В заключение в табл. П1.2 перечислим основные события, связанные с объектом `Application`.

Таблица П1.2. *События объекта* `Application`

Событие	Описание
`NewWorkbook`	Создание новой рабочей книги
`SheetActivate`	Активизация листа. Может быть как рабочий лист, так и лист с диаграммой
`SheetBeforeDoubleClick`	Двойной щелчок на рабочем листе
`SheetBeforeRightClick`	Щелчок правой кнопкой мыши на рабочем листе. Событие происходит до того, как генерируется определенное по умолчанию аналогичное событие. Если значение параметра `Cancel` процедуры обработки этого события установить равным `True`, то определенное по умолчанию действие не выполняется
`SheetCalculate`	Пересчет на рабочем листе или изменение данных в диаграмме
`SheetChange`	Изменение данных в ячейках рабочего листа
`SheetDeactivate`	Деактивизация листа. Может быть как рабочий лист, так и лист с диаграммой
`SheetFollowHyperlink`	Щелчок на гиперссылке
`SheetPivotTableUpdate`	Обновление сводной таблицы
`SheetSelectionChange`	Смена выделения на рабочем листе
`WindowActivate`	Активизация окна
`WindowDeactivate`	Деактивизация окна
`WindowResize`	Изменение размеров окна
`WorkbookActivate`	Активизация рабочей книги
`WorkbookAddinInstall`	Инсталляция надстройки
`WorkbookAddinUninstall`	Удаление надстройки
`WorkbookBeforeClose`	Перед закрытием рабочей книги. Если значение параметра `Cancel` процедуры обработки этого события установить равным `True`, то по завершении ее работы книга не закроется
`WorkbookBeforePrint`	Перед печатью рабочей книги
`WorkbookBeforeSave`	Перед сохранением рабочей книги
`WorkbookDeactivate`	Деактивизация рабочей книги
`WorkbookNewSheet`	Добавление нового листа в рабочую книгу

***Таблица П1.2** (окончание)*

Событие	Описание
`WorkbookOpen`	Открытие рабочей книги
`WorkbookPivotTableCloseConnection`	Закрытие связей со сводной таблицей
`WorkbookPivotTableOpenConnection`	Установка связей со сводной таблицей

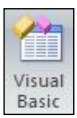
Наши итоги

В данном приложении мы сделали беглый обзор по Visual Basic for Application, познакомились с основными понятиями объектной модели и разобрали конкретные модели Microsoft Office 2010 и Microsoft Office Excel 2010. Кроме этого мы разобрали некоторые свойства объекта `Application`, а также его методы и события. Не забывайте, что полную иноформацию по всем интересующим вас моментам можно получить из имеющейся справки по VBA 7.0 в Microsoft Office Excel 2010: "Excel 2010 Developer Reference: Help and How-to" или на сайте компании Microsoft.

Приложение 2

Интегрированная среда разработки Microsoft Visual Basic

Где набирается код VBA?



Итак, код VBA набирается в редакторе Visual Basic, для перехода к которому щелкните на вкладке **Разработчик** и в группе **Код** и нажмите кнопку **Visual Basic**. Для быстрого перехода к редактору Visual Basic вы также можете использовать и комбинацию клавиш <Alt>+<F11>.

В результате вы попадаете в интегрированную среду разработки приложений (IDE) редактора Visual Basic (рис. П2.1). Эта среда разработки имеет стандартный интерфейс, характерный для типового окна Windows: строка заголовка, строка меню, панели инструментов (по умолчанию отображается панель инструментов **Standard**) и рабочая область, в которой открываются два окна — **Project - VBAProject** и **Properties**.

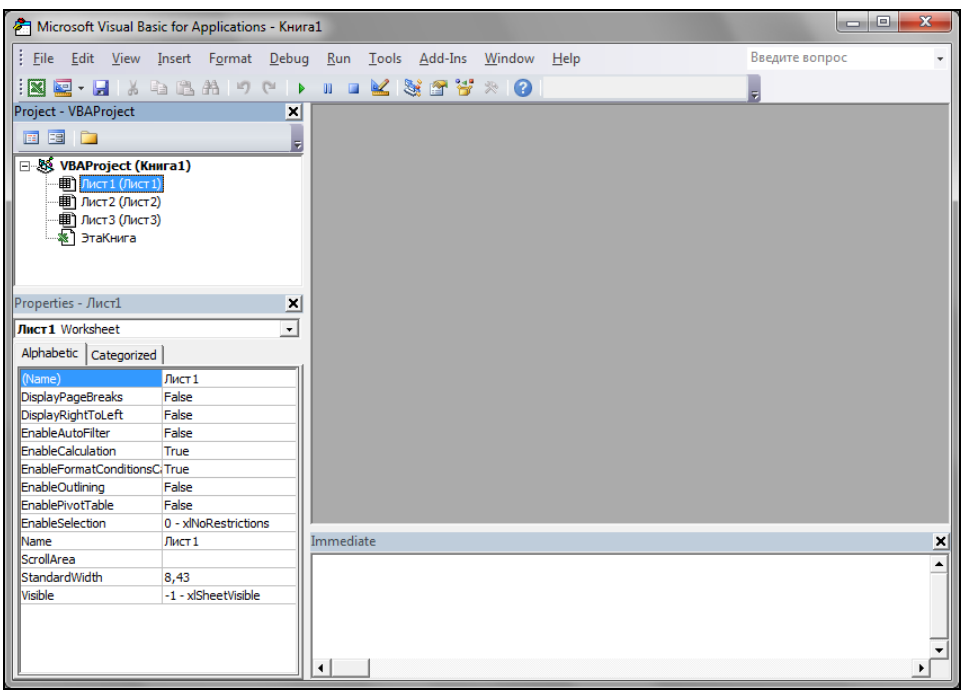


Рис. П2.1. Редактор Visual Basic

Примечание

Если в Microsoft Office Excel 2010 на ленте не отображается вкладка **Разработчик**, то добавьте ее, используя возможности настройки ленты.

Для возврата в рабочую книгу из редактора Visual Basic достаточно нажать кнопку **View Microsoft Excel** на панели инструментов **Standard** или же повторно воспользоваться сочетанием клавиш <Alt>+<F11>.

Структура редактора VBA

Рассмотрим теперь подробнее интерфейс редактора Visual Basic. Как правило, при написании программ вы столкнетесь со следующими основными компонентами редактора VBA:

- окна **Project - VBAProject (*Проект*)**;
- окна редактирования кода;
- окна форм;
- окна свойств;
- панели инструментов.

Окно *Project - VBAProject*

Окно **Project - VBAProject** открывается по умолчанию при запуске редактора Visual Basic. В случае, когда оно отсутствует, отобразить окно **Project - VBAProject** можно с помощью команды **View | Project Explorer** или кнопки **Project Explorer** на панели инструментов **Standard**. Для того чтобы скрыть отображение окна **Project - VBAProject** в редакторе Visual Basic, достаточно воспользоваться стандартной кнопкой закрытия окна. В окне **Project - VBAProject** представлена иерархическая структура файлов форм и модулей, входящих в текущий проект (рис. П2.2).

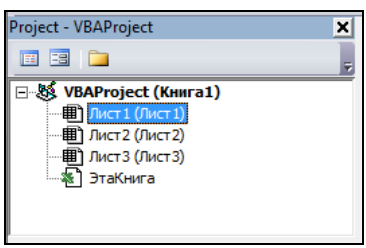


Рис. П2.2. Окно **Project - VBAProject**

Модуль представляет собой текстовый файл, в котором набирается необходимый код. Двойной щелчок на значке модуля в окне **Project - VBAProject** открывает соответствующее окно. Значок активного модуля в окне **Project - VBAProject** всегда выделяется ярким цветом.

В проекте модули создаются автоматически для каждого рабочего листа и для всей книги. Кроме того, модули создаются для каждой пользовательской формы, макросов и классов. По своему предназначению модули делятся на два типа: модули объектов и стандартные. К *стандартным модулям* относятся те, на которых записываются макросы. Такие модули добавляются в проект выбором команды **Insert | Module**. К *модулям объектов* относятся модули, связанные с рабочей книгой, рабочими листами, формами, а также модули класса.

Формы создаются выбором команды **Insert | UserForm**, а модули класса — **Insert | Class Module**. По мере создания, добавления и удаления файлов из проекта эти изменения отображаются в окне проекта. Отметим также, что удаление файла из окна проекта производится выбором значка файла с последующим выбором команды **File | Remove**.

Копирование модулей и форм из одного проекта в другой

В окне проекта выводятся проекты всех открытых рабочих книг. Это позволяет легко копировать формы, модули из одного проекта в другой при помощи простой буксировки значка файла в окне **Project - VBAProject**.

Окно редактирования кода

Двойной щелчок на значке файла в окне проекта открывает окно редактора кода для соответствующего модуля (рис. П2.3). Открыть модуль в редакторе кода для соответствующего объекта (например, рабочего листа) можно также выбором значка этого объекта в окне проекта с последующим использованием команды **View | Code**. Обратный переход от модуля к соответствующему объекту осуществляется командой **View | Object**.

Окно редактирования кода служит в качестве редактора для ввода кода процедур приложения. Код внутри модуля организован в отдельные разделы для каждого объекта, программируемого в модуле. В окне редактирования доступны два режима представления кода: просмотр отдельной процедуры и просмотр всего модуля. Переключение режимов работы окна редактирования кода осуществляется либо выбором одной из двух кнопок в левом нижнем углу окна редактирования кода (левая — отдельная процедура, правая — все процедуры модуля), либо установкой или снятием флажка **Default to Full Module View** вкладки **Editor** диалогового окна **Options** (рис. П2.4), отображаемого на экране командой **Tools | Options**. Если установлен режим просмотра всех процедур модуля, то процедуры можно отображать с разделителями (горизонтальной чертой, разделяющей две соседние процедуры) или без них. Отображением или не отображением разделителей управляет флажок **Procedure Separator**.

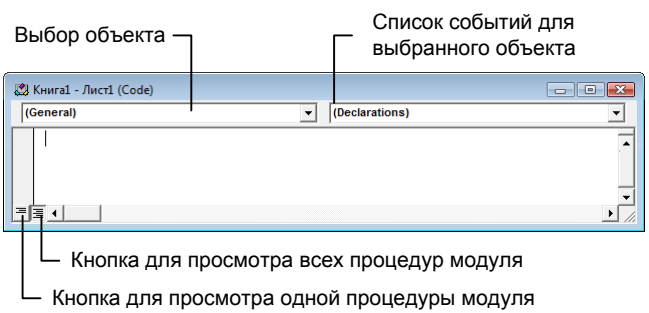


Рис. П2.3. Окно редактирования кода

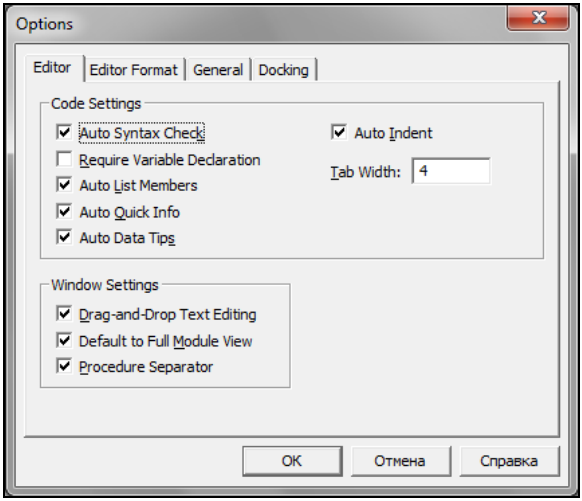


Рис. П2.4. Вкладка **Editor** диалогового окна **Options**

Два раскрывающихся списка в верхней части окна редактора кода облегчают ориентацию в процедурах (см. рис. П2.3). Левый раскрывающийся список позволяет выбрать объект, а правый — содержит список событий, допустимых для выбранного в левом списке объекта. Кроме того, выбор объекта и события приводит к созданию в редакторе кода первой и последней инструкции процедуры обработки события, связанного с выбранным объектом, если такой еще не существовало.

> ***Совет***
>
> Всегда используйте раскрывающиеся списки окна редактора кода для создания первой и последней инструкции процедуры обработки события, связанного с выбранным объектом. Использование автоматических средств, с одной стороны, ускоряет процесс создания кода (не надо его набирать вручную), а с другой стороны, позволяет избежать опечаток. О других средствах редактора кода, облегчающих набор кода, мы поговорим в следующем разделе.

Интеллектуальные возможности редактора кода

Написание программ на VBA существенно облегчается за счет способности редактора кода автоматически завершать написание операторов, свойств и параметров. При написании кода редактор сам предлагает пользователю список компонентов, логически завершающих вводимую пользователем инструкцию. Например, при наборе кода:

```
Range("A1").
```

после ввода точки на экране отобразится список компонентов (рис. П2.5), которые логически завершают данную инструкцию. Двойной щелчок на выбранном элементе из этого списка или нажатие клавиши <Tab> вставляет выбранное имя в код программы. При этом использование клавиши <Tab> вместо мыши является предпочтительным, т. к. эта клавиша находится прямо под рукой, и нажатие ее

производится только одним движением пальца левой руки, что не требует ни времени, ни усилий. В то же время работа с мышью при написании программ является слишком утомительным делом.

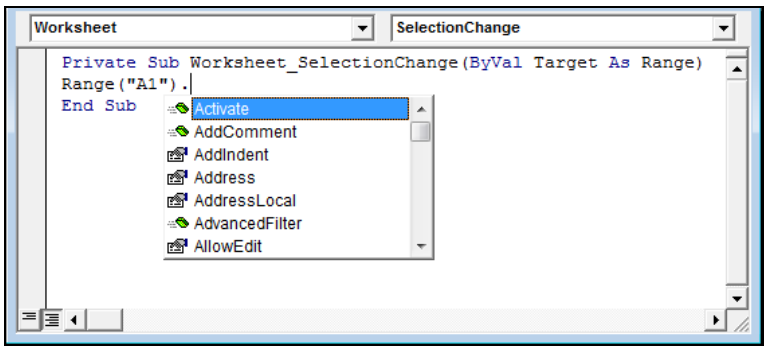


Рис. П2.5. Список компонентов

Отметим, что автоматическое отображение списка компонентов происходит только при установленном флажке **Auto List Members** вкладки **Editor** диалогового окна **Options** (см. рис. П2.4), отображаемого на экране, как указывалось ранее, выбором команды **Tools | Options**.

Список компонентов можно выводить на экран также и нажатием комбинации клавиш <Ctrl>+<J>, при этом список отображается как при установленном, так и снятом флажке **Auto List Members** вкладки **Editor** диалогового окна **Options**.

> ***Примечание***
>
> Список компонентов отображается только для существующих в форме или на рабочем листе элементов управлений. Поэтому, если в ваш проект должны входить элементы управления, сначала создайте их, а потом набирайте код.

Отображение списка компонентов, логически завершающих вводимую инструкцию, является одним из интеллектуальных качеств редактора кода. Другим его интеллектуальным качеством является автоматическое отображение на экране сведений (так называемая *всплывающая подсказка*) о процедурах, функциях, свойствах и методах после набора их имени (рис. П2.6).

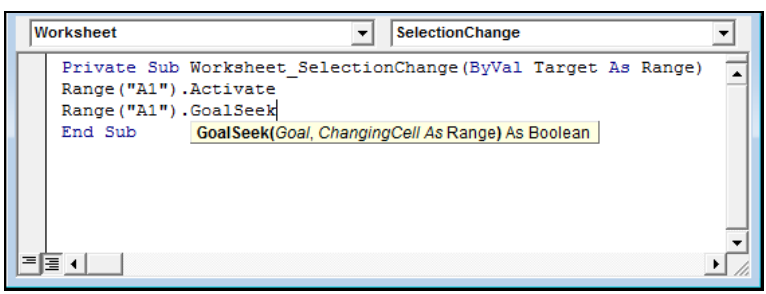


Рис. П2.6. Отображаемые сведения о вводимой процедуре

Автоматическое отображение на экране сведений о процедурах, функциях, свойствах и методах после ввода их имени происходит только при установленном флажке **Auto Data Tips** вкладки **Editor** диалогового окна **Options** (см. рис. П2.4).

Описанную выше всплывающую подсказку можно также выводить на экран нажатием комбинации клавиш <Ctrl>+<I>. При этом всплывающая подсказка отображается как при установленном, так и снятом флажке **Auto Data Tips** вкладки **Editor** диалогового окна **Options.**

Редактор кода также производит автоматическую проверку синтаксиса набранной строки кода сразу после нажатия клавиши <Enter>. Если после набора строки и нажатия клавиши <Enter> строка выделяется красным цветом, то это свидетельствует о наличии синтаксической ошибки в набранной строке. Понятно, что ошибку необходимо найти и исправить. Кроме того, если установлен флажок **Auto Syntax Check** вкладки **Editor** диалогового окна **Options** (см. рис. П2.4), помимо выделения красным цветом фрагмента кода с синтаксической ошибкой, на экране отобразится диалоговое окно, поясняющее, какая, возможно, произошла ошибка.

Для редактора кода характерна еще одна полезная возможность, повышающая эффективность работы пользователя. Так, если расположить курсор на ключевом слове языка VBA или имени процедуры, функции, свойства или метода и нажать клавишу <F1>, то на экране появится окно со справкой об этой функции. Обычно к этой справке прилагается и пример использования кода, что, несомненно, поможет при написании вашей процедуры.

Совет

Используйте справочный материал в любом случае, когда у вас возникают соответствующие трудности. Как правило, приведенных примеров бывает достаточно, чтобы разобраться в ситуации, с которой столкнулись вы при написании кода.

Окно *UserForm* (Редактирование форм)

В VBA формы используются для создания диалоговых окон разрабатываемых приложений. Редактор форм является одним из основных инструментов визуального программирования. Форма в проект добавляется выбором команды **Insert | UserForm** меню. В результате на экран выводится незаполненная форма с панелью инструментов **Toolbox** (рис. П2.7).

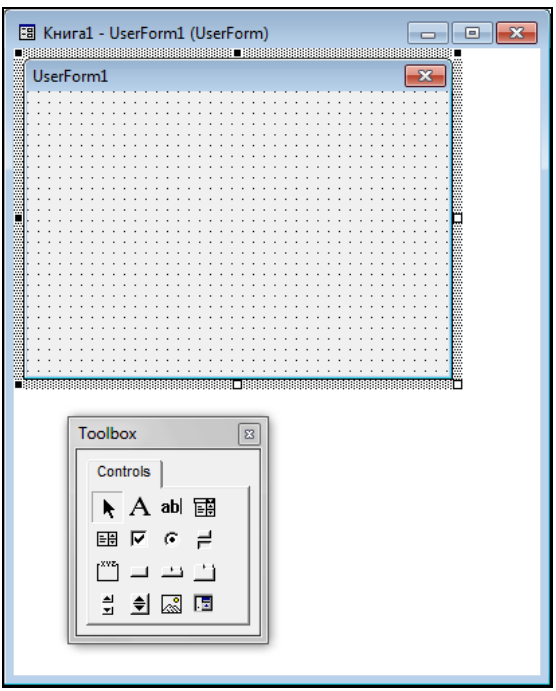


Рис. П2.7. Окно редактирования форм и панель инструментов **Toolbox**

Используя панель инструментов **Toolbox**, из незаполненной формы

можно сконструировать любое требуемое для приложения диалоговое окно. Размещение нового управляющего элемента на форме осуществляется такой последовательностью действий:

1. Щелкните на значке того элемента, который собираетесь разместить на форме.
2. Поместите указатель мыши на то место, где будет располагаться управляющий элемент.
3. Нажмите левую кнопку мыши и, не отпуская ее, растяните появившийся прямоугольник до требуемых размеров.
4. Отпустите кнопку мыши. Элемент управления создан на нужном месте.

Размеры формы и расположенных на ней элементов управления можно изменять. Технология изменения их размеров — стандартная для Windows: выделить изменяемый элемент, разместить указатель мыши на одном из размерных маркеров и протащить его при нажатой левой кнопке мыши так, чтобы объект принял требуемые размеры. Окно редактирования форм поддерживает операции буфера обмена. Таким образом, можно копировать, вырезать и вставлять элементы управления, расположенные на поверхности формы.

Для облегчения размещения и выравнивания элементов управления используется сетка. Параметры сетки устанавливаются в группе **Form Grid Settings** вкладки **General** диалогового окна **Options**, вызываемого командой **Tools | Options**:

- ❒ флажок **Show Grid** управляет отображением сетки на форме;
- ❒ поля **Width** и **Height** устанавливают расстояние по горизонтали и вертикали между соседними узлами сетки;
- ❒ флажок **Align Controls to Grid** управляет привязыванием элементов управления к сетке.

Иногда требуется переместить группу элементов управления в одном и том же направлении, например, несколько кнопок находятся на одной линии, и все их нужно переместить на несколько шагов сетки. При работе с группой элементов, как с одним объектом, следует первоначально сформировать группу элементов управления. Элементы управления можно объединить в группу, выделяя каждый из них щелчком мыши при нажатой клавише <Ctrl>. Для отмены выделения группы достаточно щелкнуть в любом месте формы, не занятой элементами управления.

После формирования группы элементов управления их легко совместно перемещать и изменять размеры при помощи команд линейки меню (выпадающего меню) категории **Format**, которые перечислены в табл. П2.1.

Таблица П2.1. *Команды выпадающего меню* ***Format***

Команда	Описание
Align | Lefts (**Centers**, **Rights**)	Выровнять группу выделенных элементов по левому краю (центру, правому краю)
Align | Tops (**Middles**, **Bottoms**)	Выровнять группу выделенных элементов по верхнему краю (середине, нижнему краю)
Align | To Grids	Выровнять по сетке
Make Same Size | Width (**Height**, **Both**)	Сделать элементы управления одной и той же ширины (высоты; и высоты, и ширины)

***Таблица П2.1** (окончание)*

Команда	Описание
Size to Fit	Установить размеры выделенных элементов управления так, чтобы они совпадали с размерами отображаемых в них надписей, рисунков
Size to Grid	Установить размеры выделенных элементов управления по сетке
Horizontal Spacing | Make Equal (**Increase**, **Decrease**, **Remove**)	Установить равное по горизонтали расстояние между выделенными элементами управления (увеличить его, уменьшить, удалить)
Vertical Spacing | Make Equal (**Increase**, **Decrease**, **Remove**)	Установить равное по вертикали расстояние между выделенными элементами управления (увеличить его, уменьшить, удалить)
Center in Form | Horizontally (**Vertically**)	Расположить выделенные элементы управления по центру формы (горизонтально, вертикально)
Arrange Buttons | Bottom (**Right**)	Упорядочить выделенные кнопки по нижнему краю формы (по правому краю формы)
Group	Группировка (объединение) выделенных элементов управления на форме (позволяет их обрабатывать как один объект)
Ungroup	Разгруппировка выделенных элементов управления на форме (позволяет разбить сгруппированные объекты на отдельные)
Order | Bring to Front (**Send to Back**, **Bring Forward**, **Send Backward**)	Расположить выделенные элементы управления на переднем плане (на заднем плане, на позицию (слой) вперед, на позицию (слой) назад)

Иногда на форме могут быть созданы лишние элементы управления. Для удаления лишних элементов управления достаточно их выделить и нажать клавишу <Delete> (или <Del>).

Окно *Properties* (Свойства)

В окне свойств перечисляются основные параметры свойств выбранной формы или элемента управления. Используя окно **Properties** (рис. П2.8), можно просматривать свойства и изменять их значения. Для просмотра свойств выбранного объекта следует либо щелкнуть кнопку **Properties Window**  панели инструментов **Standard**, либо выбрать команду меню **View | Properties Window**, либо нажать клавишу <F4>.

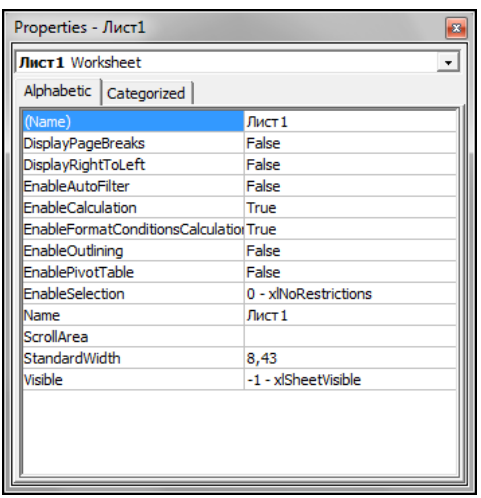


Рис. П2.8. Окно **Properties**

Окно свойств состоит из двух составных частей: верхней и рабочей. В верхней части окна свойств располагается раскрывающийся список, из которого можно выбрать любой элемент управления текущей формы или саму форму. Рабочая часть окна свойств состоит из двух вкладок: **Alphabetic** и **Categorized**, отображающих набор свойств в алфавитном порядке или по категориям. На обеих вкладках свойство `Name` (имя элемента управления) идет первым.

Изменяются значения свойств одним из следующих способов:

- вводом с клавиатуры значения свойства в соответствующее поле;
- выбором из раскрывающегося списка. Так можно выбрать значения большинства свойств. Раскрывающийся список активизируется щелчком на соответствующем поле окна свойств.

Совет

Свойство `Name` часто используется при создании процедур обработки событий для элемента управления. Оно определяет его имя. Присвоение объектам значимых имен облегчает чтение кода, делает его прозрачным. Например, предположим, что на форме создана кнопка `CommandButton1`, при нажатии которой закрывается форма. Тогда, следующий код:

```
Private Sub cmdExit_Click()

End Sub
```

значительно информативнее, чем код:

```
Private Sub CommandButton1_Click()

End Sub
```

Окно *Object Browser* (Просмотр объектов)

Окно **Object Browser** (рис. П2.9) отображается на экране выбором команды меню **View | Object Browser**, или нажатием кнопки **Object Browser**  , или нажатием клавиши <F2>. В данном окне приведен список всех объектов, которые имеются в системе и которые можно использовать при создании проекта.

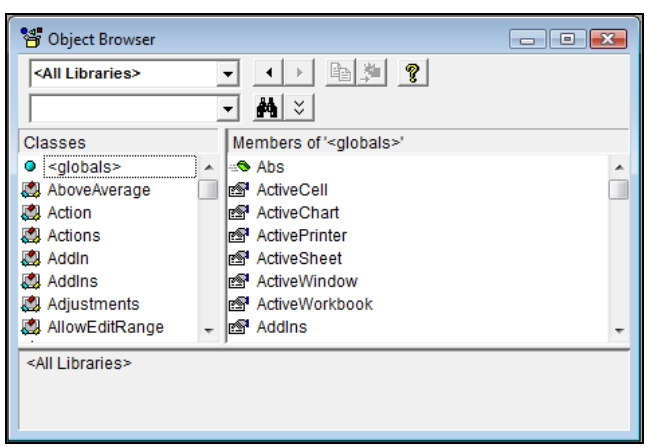


Рис. П2.9. Окно **Object Browser**

Окно **Object Browser** состоит из трех основных частей:

- раскрывающегося списка **Project/Library** в левом верхнем углу окна. В этом раскрывающемся списке можно выбрать различные проекты и библиотеки объектов. В частности, библиотеки объектов Excel, VBA, Office и VBAProject (объекты пользовательского проекта). Выбор в списке элемента **<All Libraries>** отображает список объектов всех библиотек;
- списка **Classes**. После выбора из раскрывающегося списка **Project/Library** просматриваемой библиотеки, например VBA, все классы объектов выбранной библиотеки выводятся в списке **Classes**;
- списка **Members**. После выбора класса из списка **Classes** просматриваемой библиотеки, например `FileSystem`, все компоненты выбранного класса выводятся в списке **Members**. При выделении строки в списке **Members** в нижней части окна **Object Browser** приводится дополнительная информация о выбранном компоненте.

Кроме того, если нажать кнопку **Help**, расположенную на панели инструментов в правой верхней части окна **Object Browser**, то на экране отобразится окно **Справочник Visual Basic** с подробной справкой по выделенному компоненту. Назначение же других кнопок панели инструментов окна **Object Browser** легко понять из контекста: **Search Text** — поле для ввода ключевых слов поиска; **Search** — кнопка для запуска процесса поиска; **Show Search Results** — отображение окна результатов поиска и др.

Наши итоги

В этом приложении вы подробно изучили интегрированную среду разработки Microsoft Visual Basic, которая имеет стандартный интерфейс типового окна Windows. Вы познакомились с основными окнами, с которыми предстоит работать в дальнейшем, узнали об интеллектуальных возможностях редактора кода и возможности получения быстрой справки при возникновении трудностей в ходе написания кода программ.

Приложение 3

Отладка приложений

При написании программ пользователь, независимо от его опыта, допускает те или иные ошибки. Условно ошибки, которые допускает разработчик при подготовке программ на языке VBA, можно поделить на три типа: ошибки компиляции, ошибки выполнения и логические ошибки. В данном приложении мы остановимся на их краткой характеристике.

Ошибки компиляции

Ошибки компиляции возникают, если Visual Basic не может интерпретировать введенный код. Например, при некорректном вводе числа скобок, неправильном имени, неполном вводе инструкции и т. д. Некоторые из этих ошибок обнаруживаются Visual Basic при завершении набора строки с инструкцией в редакторе кода нажатием клавиши <Enter>. Строка, в которой содержится ошибка, выделяется красным цветом, и на экране отображается диалоговое окно с сообщением о возможной причине, вызвавшей ошибку (рис. П3.1).

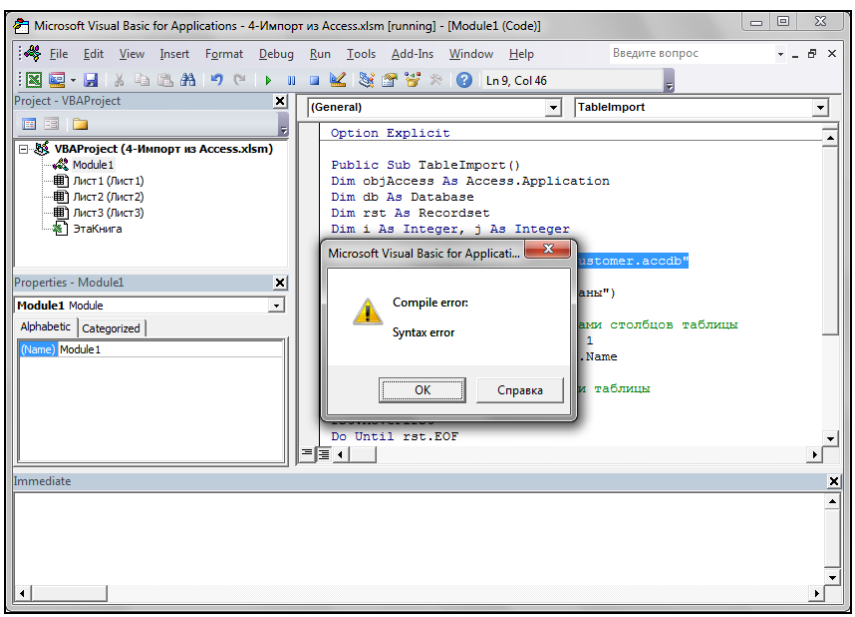


Рис. П3.1. Ошибка компиляции, обнаруженная при вводе инструкции

Другие ошибки компиляции обнаруживаются перед выполнением программы. Отметим, что VBA каждый раз автоматически компилирует программу при ее запуске на выполнение. В этом случае предполагаемое местоположение ошибки выделяется синим цветом, и на экране отображается диалоговое окно Microsoft VBA с сообщением о возможной причине, вызвавшей ошибку. Так, например, на рис. П3.2 допущена следующая ошибка — опущена инструкция `End Select` в операторе `Select`.

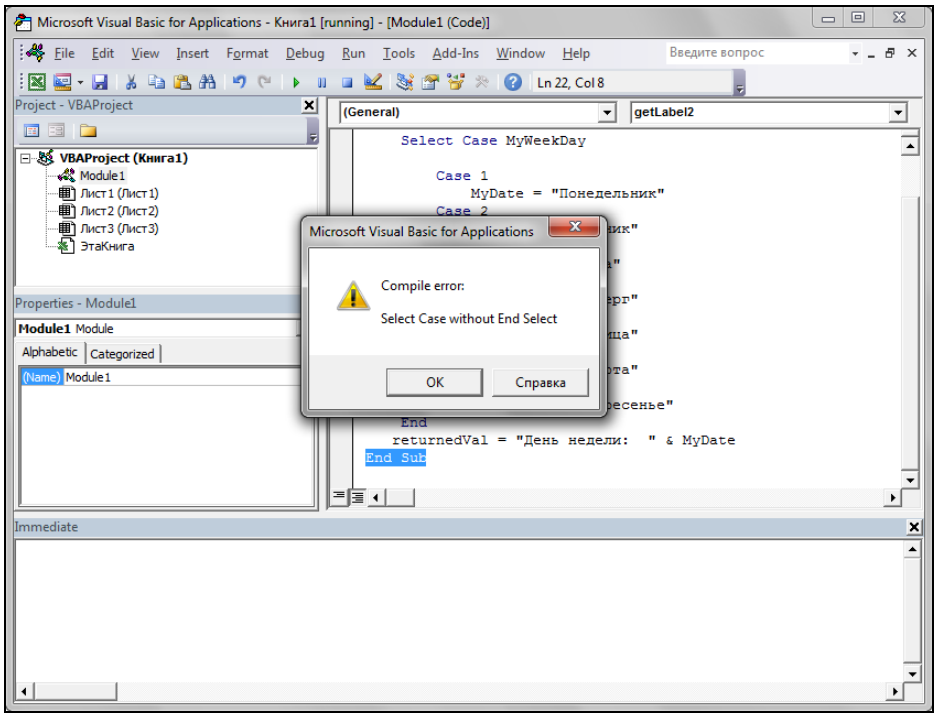


Рис. П3.2. Сообщение об ошибке компиляции в диалоговом окне Microsoft VBA

Ошибки выполнения

Ошибки выполнения возникают после успешной компиляции программы при ее выполнении. Причинами таких ошибок могут быть, например, следующие:

- некорректная информация при считывании файла с диска;
- некорректные данные, введенные пользователем; так, например, требуется число, а пользователь вводит строковую информацию;
- некорректность вычислений, например деление на ноль и т. д.

В этом случае на экране отображается диалоговое окно **Microsoft Visual Basic** с сообщением о номере ошибки и возможной причине, ее вызвавшей (рис. П3.3).

Если в диалоговом окне **Microsoft Visual Basic** нажать кнопку **Debug**, то в окне модуля желтым цветом будет выделена строка, вызвавшая ошибку, на которой

выполнение программы было прервано. Кроме того, эта строка будет помечена стрелочкой. VBA перейдет в режим прерывания (рис. П3.4, см. также файл *Приложение_1.xlsm* на компакт-диске).

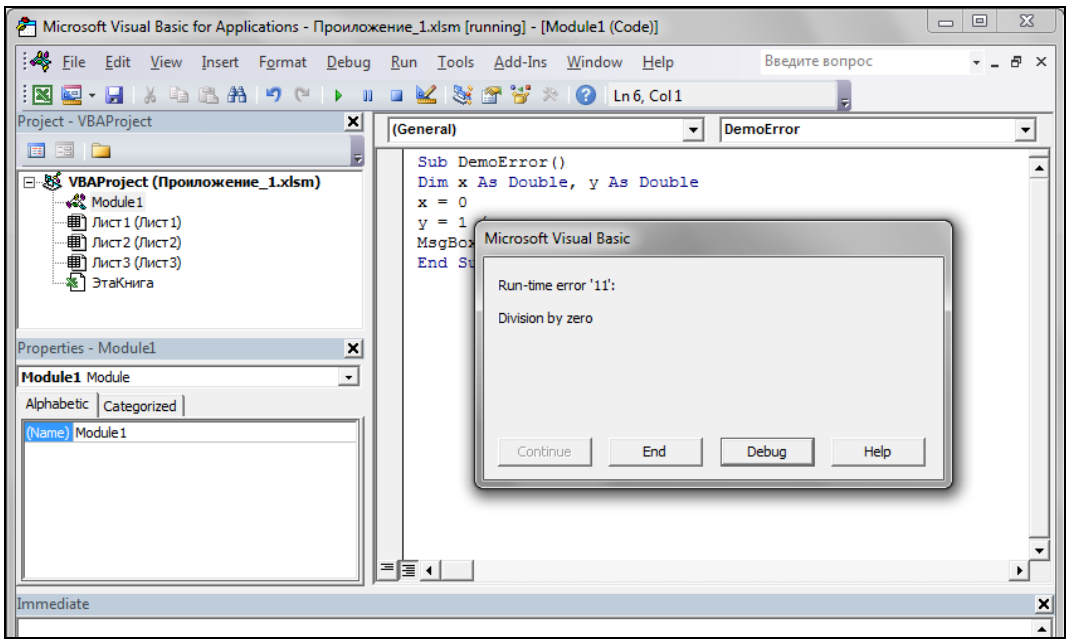


Рис. П3.3. Сообщение об ошибке выполнения в диалоговом окне **Microsoft Visual Basic**

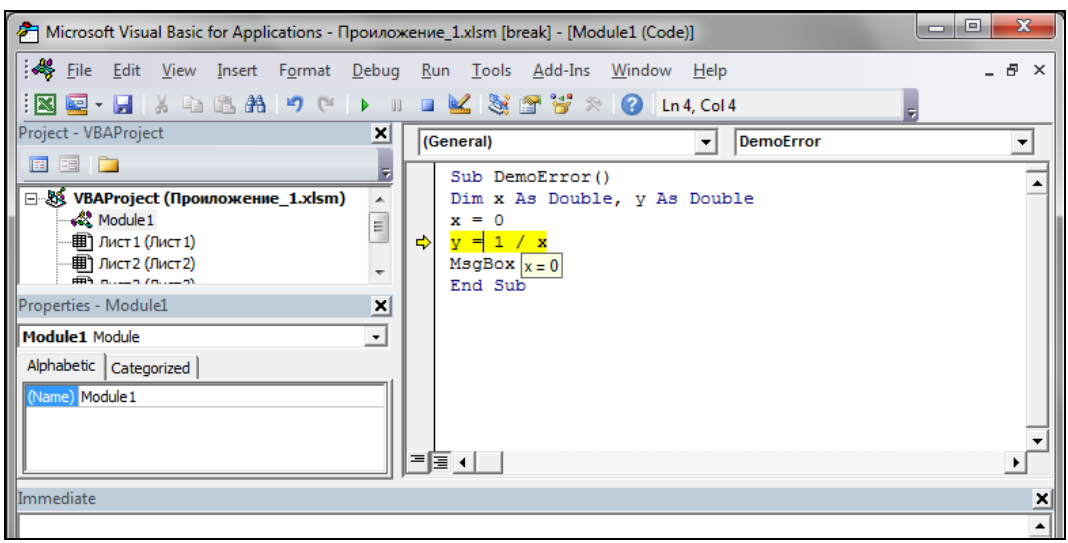


Рис. П3.4. Редактор кода в режиме прерывания

Одним из удобств режима прерывания является возможность узнать текущее значение переменных и свойств, для чего достаточно расположить указатель мыши на имени свойства или переменной. Это вызовет появление всплывающей подсказки с текущим значением переменной или свойства. В данном случае (см. рис. П3.4) видно, что значение переменной `y` равно 0, что и вызвало ошибку. Для задания режима вывода всплывающей подсказки с текущими значениями данных должен быть установлен флажок **Auto Data Tips** вкладки **Editor** диалогового окна **Options**, вызываемого командой **Tools | Options**.

Кроме режима прерывания, приложение может находиться в режиме разработки и режиме выполнения. В режиме разработки, собственно, и создается приложение — конструируются формы, набирается код. В режиме выполнения приложение получает управление, и мы взаимодействуем с ним так же, как это будет делать и пользователь нашего приложения. Переключение между режимами работы удобно производить при помощи трех кнопок панели инструментов **Standard** (кстати, они также включены в панель инструментов **Debug**), перечисленных в табл. П3.1.

Таблица П3.1. *Кнопки панели инструментов* ***Standard****, управляющие режимом работы программы*

Кнопка	Название	Описание
	**Run Macro**	Доступна в режиме конструирования. Переключает в режим выполнения. В режиме прерывания она также доступна, но играет роль кнопки **Run Sub/UserForm**
	**Break**	Доступна в режиме выполнения. Переключает в режим прерывания
	**Reset**	Доступна в режиме выполнения. Переключает в режим разработки

Логические ошибки

Логические ошибки труднее всего обнаружить и устранить. Эти ошибки не приводят к прерыванию выполнения программы, т. е. визуально все идет гладко и выглядит так, как будто программа работает безупречно. Но это только кажущаяся идиллия, поскольку программа выдает неверные результаты. Локализация логических ошибок связана с тщательным анализом алгоритма программы с привлечением средств отладки VBA.

Инструкция *Option Explicit*

Простейшим средством предотвращения случайных ошибок является использование инструкции `Option Explicit`. Эта инструкция предписывает объявлять все переменные, встречающиеся в программе. Использование этой инструкции позволяет избежать следующей трудно отслеживаемой ошибки. Предположим, что в программе используется переменная с именем `Ссуда`, а при наборе имени этой переменной где-то в программе вместо русской буквы "с" по ошибке набрана

латинская буква "c". Визуально эти имена ничем не отличаются друг от друга, но воспринимаются программой как имена различных переменных. Если бы была использована инструкция `Option Explicit`, и переменная `Ссуда` была объявлена, то компилятор указал бы на переменную `Ссуда` с латинской буквой "`c`" как на необъявленную, и эта трудно отслеживаемая ошибка была бы быстро найдена.

Пошаговое выполнение программ

Редактор Visual Basic позволяет осуществлять пошаговое выполнение программы. Такой режим можно задать либо при помощи панели инструментов **Debug**, либо из меню **Debug**, которое включает команды и соответствующие комбинации клавиш (рис. П3.5). Если панель инструментов **Debug** отсутствует на экране, то ее можно отобразить командой **View | ToolBars | Debug**.

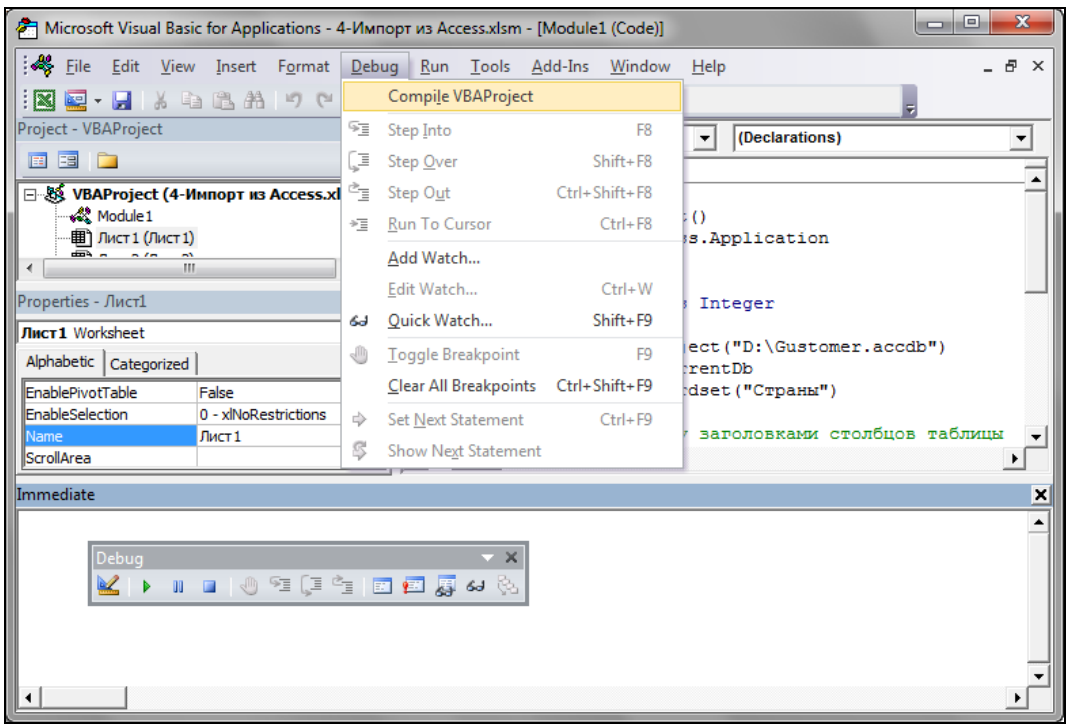


Рис. П3.5. Панель инструментов **Debug** и раскрывающееся меню **Debug**

Для выполнения программы в пошаговом режиме используются четыре команды:

- команда **Debug | Step Into** либо кнопка панели инструментов **Debug** осуществляют последовательную, шаг за шагом, отладку всей программы, включая процедуры, вызываемые в программе;
- команда **Debug | Step Over** либо кнопка панели инструментов **Debug** осуществляют последовательную, шаг за шагом, отладку всей программы, но не заходя в процедуры, вызываемые в программе. Если встречается процедура,

то она выполняется целиком, а не шаг за шагом, как это делается в команде **Debug | Step Into**;

- команда **Debug | Step Out** либо кнопка  панели инструментов **Debug** завершают выполнение текущей процедуры и останавливаются на следующей инструкции программы, откуда процедура была вызвана;
- команда **Debug | Run to Cursor** выполняет программу до инструкции, помеченной курсором.

Точка прерывания

VBA приостанавливает выполнение программы перед строкой кода, содержащей точку прерывания, и переключается в режим прерывания. Точка прерывания устанавливается или снимается командой **Debug | Toggle Breakpoint** либо кнопкой  панели инструментов **Debug**. В модуле точки прерывания выделяются полосой кирпичного цвета и кругом того же цвета (рис. П3.6).

В одном проекте может быть несколько точек прерывания. Все инструкции, расположенные выше, ниже точек прерывания и между ними, выполняются в обычном режиме.

Одновременно снять все точки прерывания можно командой **Debug | Clear All Breakpoints**.

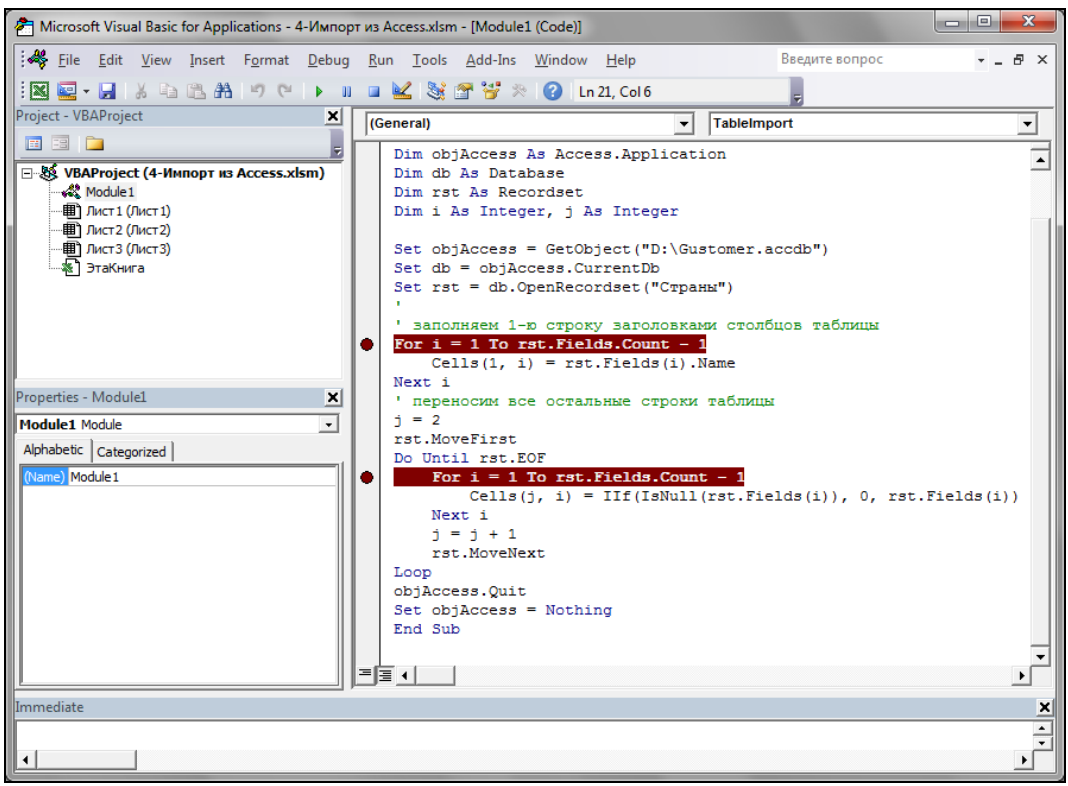


Рис. П3.6. Точки прерывания

Вывод значений свойств и переменных

Одним из удобств режима отладки является возможность узнать текущее значение переменных и свойств, для чего, как и в режиме прерывания, достаточно расположить указатель мыши на имени свойства или переменной. Это вызовет появление всплывающей подсказки с текущим значением переменной или свойства (см. рис. П3.4).

Окно *Watches*

Другим способом отслеживания текущих значений данных является использование диалогового окна **Watches**, которое отображается на экране либо командой **View | Watch Window**, либо командой **Debug | Quick Watch** (рис. П3.7, см. также файл *Приложение_2.xlsm* на компакт-диске). Диалоговое окно **Watches** позволяет одновременно отображать текущие значения нескольких переменных или свойств. Команда **Debug | Add Watch** добавляет новые контрольные значения в диалоговое окно **Watches**.

Удаление контрольного значения из диалогового окна производится его выделением и нажатием клавиши <Delete>.

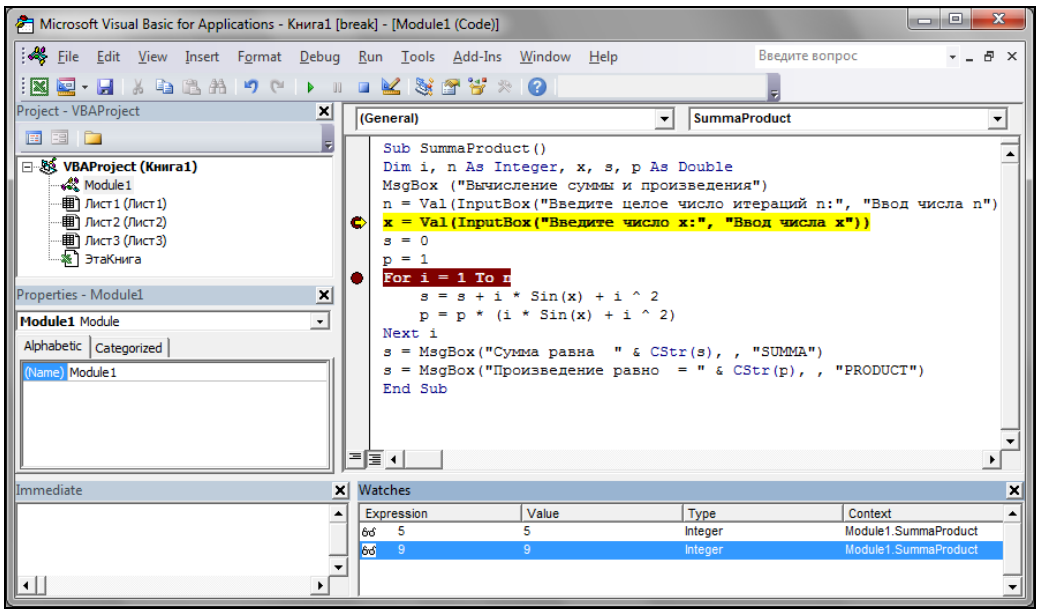


Рис. П3.7. Окно **Watches**

Окно *Locals*

Окно **Locals**, отображаемое на экране командой **View | Locals Window**, выводит значения всех переменных текущей процедуры, а не только специально выбранных, как это происходит в окне **Watches**. Внешний вид и структура обоих окон, **Watches** и **Locals**, одинаковы.

Окно *Immediate*

Окно **Immediate**, отображаемое на экране командой **View | Immediate Window**, предоставляет пользователю следующие возможности:

- набирать и вычислять отдельные инструкции VBA. Для этого достаточно в окне **Immediate** ввести соответствующую инструкцию и нажать клавишу <Enter>. Единственным ограничением на инструкцию является то, что она должна быть набрана в одну строку. Например:

```
s = 0: For i=1 to 5: s = s + i ^ 2: Next i: MsgBox s
```

- определять текущие значения переменных и свойств. Для этого в окне **Immediate** надо набрать вопросительный знак, имя переменной или свойства и нажать клавишу <Enter>. Например:

```
?x
```

- устанавливать новые текущие значения переменных. Для этого в окне **Immediate** надо набрать имя переменной, знак "равно" и новое значение переменной:

```
x = 15
```

- определять значения встроенных в VBA констант. Для этого в окне **Immediate** надо набрать вопросительный знак, имя константы и нажать клавишу <Enter>. Например:

```
? vbKeyReturn
```

Программный способ вывода значений в окно *Immediate*

Существует также программный способ вывода значения свойств и переменных в диалоговое окно **Immediate** при помощи метода `Print` объекта `Debug`. Далее приведен пример программного способа вывода значений переменных в окно **Immediate** (листинг П3.1).

Листинг П3.1. Программный способ вывода значений в окно *Immediate*

```
Dim n As Integer, Res As Integer
Randomize
For n = 1 To 10
   Res = Int(6 * Rnd()) + 1
   Debug.Print n, Res
Next
```

Наши итоги

Итак, в этом приложении нами рассмотрены возможные ошибки, которые могут быть допущены при разработке программ на VBA. Дана краткая характеристика ошибок и средства VBA, которые позволяют их обнаружить, а также проанализировать выполнение программ.

Приложение 4

Описание компакт-диска

К книге прикладывается компакт-диск, на котором размещены файлы рассматриваемых в книге примеров. Файлы сгруппированы по главам, т. е. примеры для *главы 1* вы найдете в папке Glava_1 и т. д.

Рекомендуемая литература

1. Берк К., Кэйри П. Анализ данных с помощью Microsoft Excel / Пер. с англ. — М.: Издательский дом "Вильямс", 2005. — 560 с.
2. Бухвалов А., Бухвалова В., Идельсон А. Финансовые вычисления для профессионалов. — Дюссельдорф, Киев, Москва, Санкт-Петербург: BHV, 2001. — 320 с.
3. Винстон У. Л. Microsoft Office Excel 2007. Анализ данных и бизнес-моделирование (+ CD-ROM). — СПб.: БХВ-Петербург, 2008. — 608 с.: ил.
4. Гарнаев А. Ю. Excel, VBA, Internet в экономике и финансах. — СПб.: БХВ-Петербург, 2001. — 816 с.
5. Гарнаев А. Ю. Microsoft Excel 2002: Разработка приложений. — СПб.: БХВ-Петербург, 2002. — 763 с.
6. Гарнаев А. Ю. VBA: в подлиннике. — СПб.: БХВ-Петербург, 2005. — 848 с.
7. Гарнаев А. Ю. Самое главное об Excel. — СПб.: Питер, 2004. — 109 с.
8. Гарнаев А. Ю. Самоучитель VBA. — 2-е издание. — СПб.: БХВ-Петербург, 2004. — 540 с.
9. Гарнаев А. Ю., Гарнаев С. Ю. Web-программирование на Java и JavaScript. — СПб.: БХВ-Петербург, 2001. — 1040 с.
10. Долженков В., Стученков А. Microsoft Office Excel 2010 (+ CD-ROM). — СПб.: БХВ-Петербург, 2011. — 816 с.: ил. + Видеокурс (на CD-ROM) — (В подлиннике).
11. Карлберг К. Управление данными с помощью Excel / Пер. с англ. — М.: Издательский дом "Вильямс", 2005. — 448 с.
12. Курицкий Б. Поиск оптимальных решений средствами Excel 7.0. — СПб.: BHV, 1997.
13. Рудикова Л. В. Microsoft Excel для студента. — СПб.: БХВ-Петербург, 2005. — 368 с.: ил.
14. Рудикова Л. В. Microsoft® Office для студента. — СПб.: БХВ-Петербург, 2005. — 592 с.: ил.
15. Рудикова Л. В. Базы данных для студента. — СПб.: БХВ-Петербург, 2006. — 496 с.: ил.
16. Рудикова Л. В. Проектирование баз данных / Учебное пособие для студентов высш. учеб. заведений по специальностям "Программное обеспечение информационных технологий", "Экономическая кибернетика", "Прикладная матема-

тика (научно-педагогическая деятельность)", "Информационные системы и технологии (в экономике)". — Минск: ИВЦ Минфина, 2009. — 352 с.
17. Уокенбах Дж. Microsoft Excel 2010. Библия пользователя (+ CD-ROM). — М.: "Диалектика", 2011 г. — 912 с.
18. Уокенбах Дж. Microsoft Office Excel 2007. Библия пользователя. — М.: Издательский дом "Вильямс", 2008. — 816 с.
19. Уокенбах Дж. Microsoft Office Excel 2007: профессиональное программирование на VBA (+ CD-ROM). — М.: Диалектика, 2008. — 928 с.
20. Уокенбах Дж. Microsoft Office Excel 2010: профессиональное программирование на VBA (+ CD-ROM). — М.: Диалектика, 2011. — 944 с.
21. Уокенбах Дж. Формулы в Microsoft Excel 2010 (+ CD-ROM). — М.: Диалектика, 2011. — 704 с.
22. Харитонова И. А., Рудикова Л. В. Microsoft® Office Access 2007. — СПб.: БХВ-Петербург, 2008. — 1280 с.: ил. + Видеокурс (на CD-ROM) — (В подлиннике).

Предметный указатель

П

Р

С

Т