АНДРЕЙ ГАРНАЕВ, ЛАДА РУДИКОВА

bhv

Microsoft Office Excel 2010 разработка приложений

- БОЛЕЕ 300 ПРИМЕРОВ РАЗРАБОТКИ ПРИЛОЖЕНИЙ НА VBA
- КОНСТРУИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА И ФОРМ
- АВТОМАТИЗАЦИЯ ПОВСЕДНЕВНОЙ РАБОТЫ
- ИСПОЛЬЗОВАНИЕ ФОРМУЛ И ФУНКЦИЙ РАБОЧЕГО ЛИСТА
- ОБРАБОТКА И АНАЛИЗ ДАННЫХ
- РЕШЕНИЕ РАСЧЕТНЫХ ЗАДАЧ
- РАБОТА С БАЗАМИ ДАННЫХ
- ИНТЕГРАЦИЯ ПРИЛОЖЕНИЙ

PRO

ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

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

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).

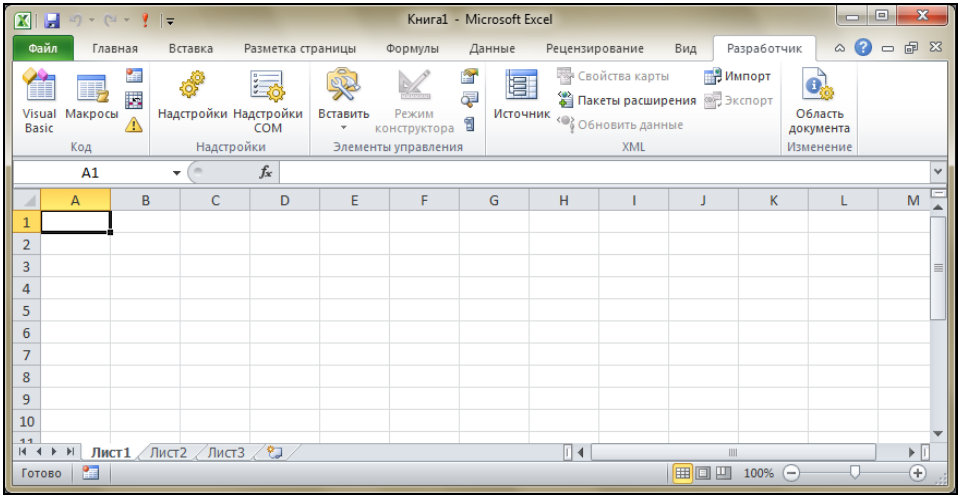

Рис. 1.1. Вкладка **Разработчик** на ленте

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.

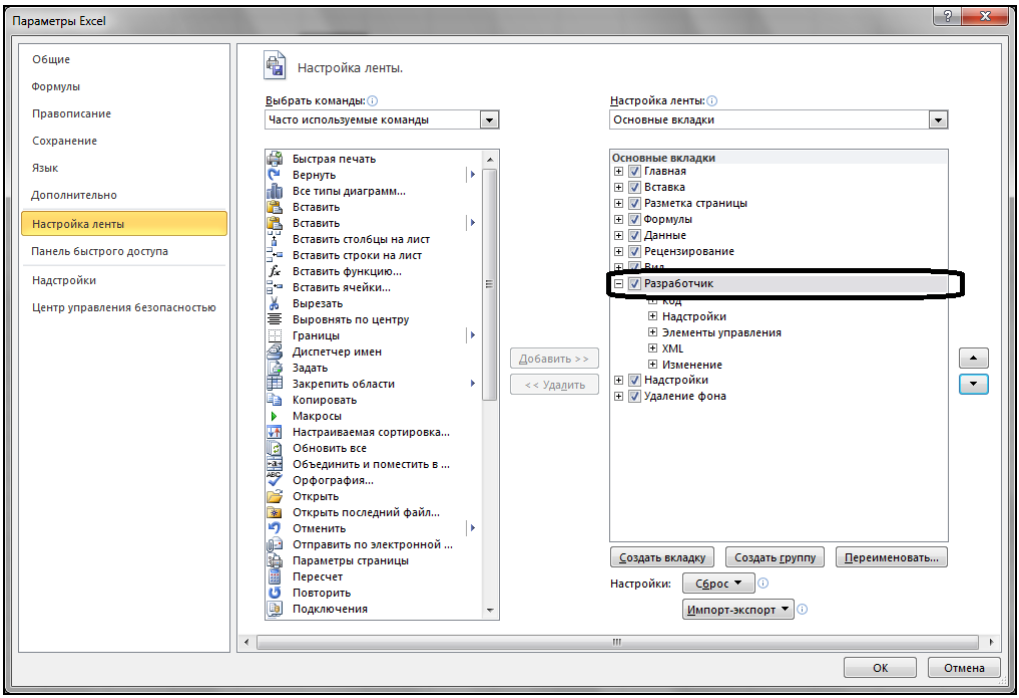

Рис. 1.2. Окно **Параметры Excel**

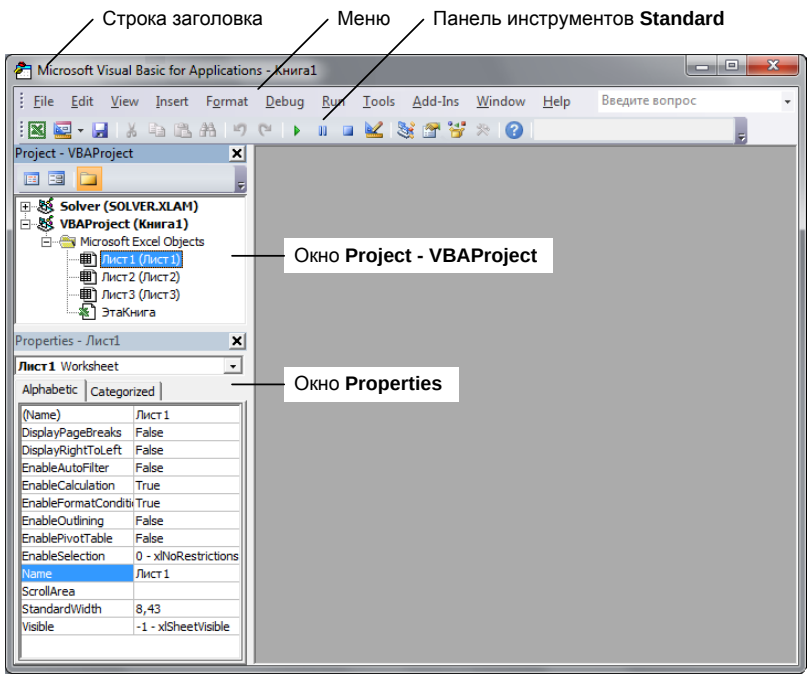

Рис. 1.3. Редактор Visual Basic

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).

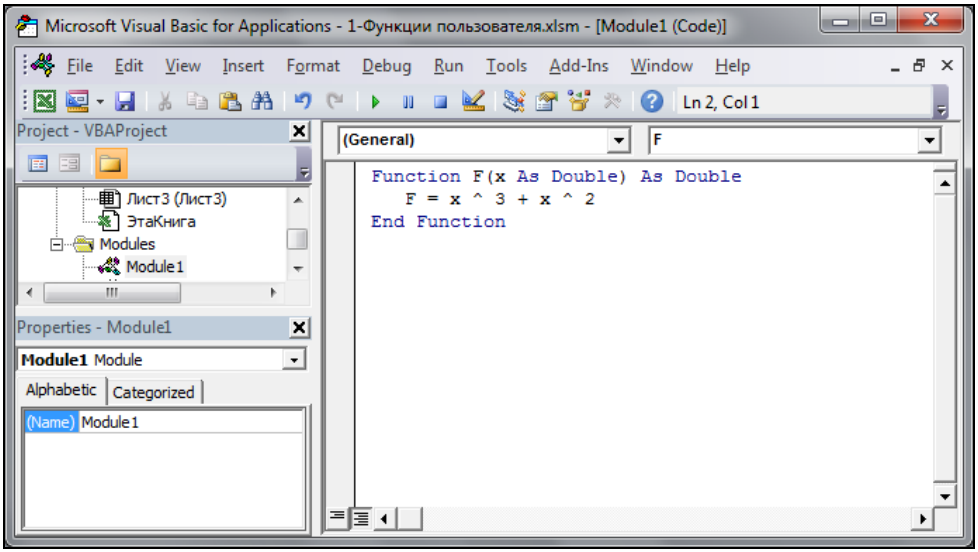

Рис. 1.4. Код пользовательской функции в модуле

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.

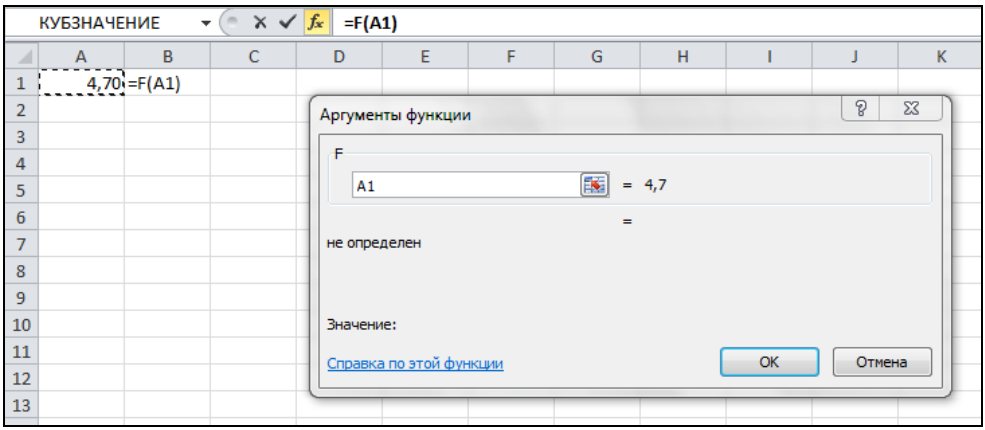

Рис. 1.5. Вычисление значения пользовательской функции при помощи мастера функций

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).

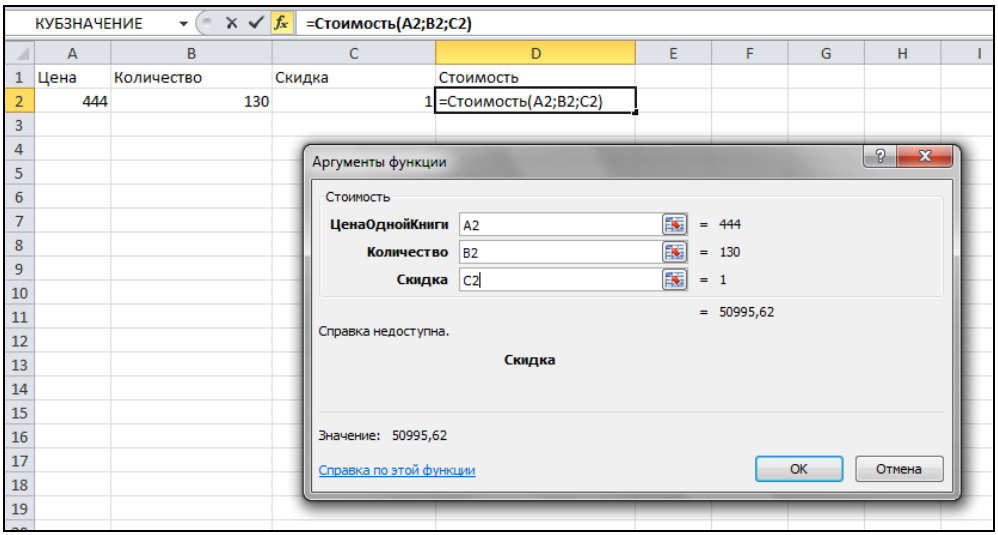

Рис. 1.6. Расчет стоимости партии книг

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.