ПРАКТИКУМ
по
ЭКОНОМИЧЕСКОЙ
ИНФОРМАТИКЕ

П.П.Мельников, И.В.Миронова, И.Ю.Шполянская

ПРАКТИКУМ ПО ЭКОНОМИЧЕСКОЙ ИНФОРМАТИКЕ

Под редакцией П.П.Мельникова

В трех частях

Часть III

Допущено Министерством образования
Российской Федерации
в качестве учебного пособия
для студентов высших учебных заведений,
обучающихся по специальностям подготовки
дипломированных специалистов "Финансы и кредит",
"Бухгалтерский учет, анализ и аудит",
"Мировая экономика"

Финансы и статистика

Москва
2002

ИП
Издательство "Перспектива"

УДК [004.4:33] (076.5)
ББК 65ф.я73
М48

РЕЦЕНЗЕНТЫ:

Кафедра автоматизированных информационных технологий Финансовой академии при Правительстве РФ;
С.А. Чернецов,
доктор экономических наук, профессор

Мельников П.П. и др.

М48 Практикум по экономической информатике: Учеб. пособие: В 3-х ч. — Ч. III / П.П. Мельников, И.В. Миронова, И.Ю. Шполянская; Под ред. П.П. Мельникова. — М.: Финансы и статистика; Перспектива, 2002. — 160 с.

ISBN 5-279-02474-0 (Финансы и статистика)
ISBN 5-88045-057-0 (Перспектива)

Часть III — последняя книга «Практикума по экономической информатике» — посвящена основам программирования. В качестве инструментальной среды рассматривается Visual Basic for Application, работающий совместно с приложением MS Excel. В каждый раздел пособия включены упражнения и практические задания для самостоятельной работы, примеры, контрольные вопросы. В упражнениях предлагаются типовые экономические задачи и приводится последовательность разработки программ для их решения.

Для преподавателей, аспирантов, студентов экономических вузов, слушателей институтов повышения квалификации.

М $\frac{2404000000 - 157}{010(01) - 2001}$ без объявл.

УДК [004.4:33](076.5)
ББК 65ф.я73

ISBN 5-279-02474-0
ISBN 5-88045-057-0

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Часть III учебного пособия «Практикум по экономической информатике» предназначена для изучения и практического освоения приемов использования инструментальной среды Visual Basic for Application (VBA) и MS Excel, создания несложных программ для решения экономических задач на персональном компьютере. Учебное пособие подготовлено с учетом того, что читатель в достаточной степени освоил приемы работы с MS Excel в рамках материала, изложенного в части I практикума.

Учебно-методическое пособие предназначено для студентов очной и заочной формы обучения всех специальностей экономических вузов, а также для желающих изучить основы программирования в среде VBA самостоятельно. Оно может быть также рекомендовано преподавателям для проведения практических занятий.

При подготовке пособия авторы не ставили цель полностью описать все возможности VBA, а исходили только из требований учебных программ различных вузов. Для более глубокого освоения программирования на VBA следует пользоваться дополнительными источниками.

В пособии представлен материал, в подготовке которого приняли участие преподаватели кафедр вычислительной техники нескольких экономических вузов: доц. П.П. Мельников (Финансовая академия при Правительстве РФ) — общее редактирование, введение, приложение, пп. 1, 2, 3, 5, 7, 8, 9, упражнения 1, 3—5, 8—11, 13—17; доц. И.В. Миронова (Российская экономическая академия им. Г.В. Плеханова) — пп. 4, 5.2, 5.4, 5.8.1, 5.8.4, 6, упражнения 2, 6, 7, 12; доц. И.Ю. Шполянская (Ростовский-на-Дону государственный экономический университет) — рекомендации по содержанию и структуре материала.

1. VBA как система объектно-ориентированного программирования

1.1. Основные понятия объектно-ориентированного программирования

В основе объектно-ориентированного программирования (ООП), управляемого событиями, лежат понятия *класс, инкапсуляция, объект, наследование, полиморфизм, событие и сообщение.*

В качестве объектов могут рассматриваться конкретные предметы, а также абстрактные или реальные сущности. Например, объектами могут быть покупатель, фирма, производящая товары, банк, заказ на поставку и др.

В частном случае, в VBA *объектом* являются элементы пользовательского интерфейса, которые создаются на Форме пользователя (UserForm) или на рабочем листе, а также рабочая книга и ее элементы.

Объект является комбинацией состояния и поведения. Состояние описывается переменными экземпляра, а его возможное поведение характеризуется присущими ему методами.

Каждый объект является представителем некоторого *класса* однотипных объектов, т.е. объект является экземпляром класса. Класс определяет общие для всех его объектов методы и свойства.

Методы — это программные процедуры, реализующие некоторый алгоритм, который определяет взаимодействие объектов класса с внешней средой.

Свойства представляют собой характеристики (атрибуты), присущие объектам (например, размер шрифта, название и др.).

Инкапсуляция — это скрытие информации. При объектно-ориентированном программировании возможен доступ к объекту только через его методы и свойства. Внутренняя структура объекта скрыта от пользователя, т.е. объекты — это самостоятельные сущности, отделенные от внешнего мира. Инкапсуляция позволяет изменять реализацию объектов любого класса без опасений, что это вызовет нежелательные побочные эффекты в программной системе. Это мощное средство обеспечивает многократное использование одного и того же программного кода, позволяя собирать программу из готовых модулей, как здание из отдельных кирпичиков, но различной архитектуры и функционального назначения.

Наследование — это возможность выделить свойства, методы и события одного объекта и приписать их другому объекту, иногда с их модификацией. С точки зрения программиста, новый класс должен содержать только коды и данные для новых или изменяющихся методов.

Полиморфизм — это способность объектов выбирать операцию на основе данных, принимаемых в сообщении. Каждый объект может реагировать по-своему на одно и тоже сообщение. Например, команда Print будет по-разному воспринята черно-белым или цветным принтером.

В заключение следует сказать, что ООП — это новая технология программирования, позволившая перейти к индустриальным методам разработки программного обеспечения.

1.2. Общие сведения о VBA

Существует целый ряд систем программирования, позволяющих в той или иной степени реализовать концепцию объектно-ориентированного подхода при разработке программных средств. Наиболее полные возможности имеют такие системы, как Си++, Java, Visual Basic. В отличие от Visual Basic, VBA не является языком объектно-ориентированного программирования в строгом смысле этого слова, тем не менее, в нем очень широко используются элементы ООП и связанные с ним понятия.

VBA положен в основу системы программирования Visual Basic. Корпорация Microsoft использует возможности VBA при разработке приложений Windows, осуществляя параллельное с разработкой тестирование приложений с помощью макроязыка.

Система программирования Visual Basic for Application (Visual Basic для приложений — VBA) занимает важное место в стратегии программных продуктов фирмы Microsoft.

Другим подмножеством VBA является язык описания сценариев VBScript. Его основным назначением является создание интерактивных Web-страниц.

Таким образом, изучив программирование в среде VBA, вы без больших усилий можете освоить программирование на Visual Basic или VBScript.

Так что же такое VBA?

Visual Basic for Application — это подмножество Visual Basic (VB), которое включает почти все его средства создания приложений, структуры данных и управляющие структуры, возможность

создания пользовательских типов данных. VBA, как и VB, является языком визуального и событийно управляемого программирования — в нем есть возможность создания форм со стандартным набором элементов управления и написания процедур, обрабатывающих события, которые возникают при тех или иных действиях системы и конечного пользователя. Кроме того, он позволяет применять элементы ActiveX. В общем, это полноценный язык программирования, хотя и не обладающий всеми возможностями последних версий Visual Basic. С другой стороны, VBA позволяет работать с огромным набором объектов — по существу, в нем определены все объекты приложений MS Office.

Отметим одну, может быть главную, особенность программирования на VBA. Программист, создавая проект на каком-либо языке программирования, работает в соответствующей среде, где язык — главное *средство*, а создание кода — главная *цель* действия программиста. А при работе на VBA целью является создание документа в широком смысле (документа Word, рабочей книги Excel, презентации и т. д.). Проект на VBA — результат побочной деятельности по созданию документа. Более того, проект на VBA, независимо от какого-либо документа, создать нельзя, даже если никакие свойства этого документа не используются.

Приступая к очередному сеансу работы, программист открывает одно из приложений MS Office, и в этот момент в языке VBA автоматически становится доступным объект Application, определяющий это приложение, а также все встроенные в него объекты. Можно определить и создать объект Application для любого приложения MS Office, получив тем самым доступ ко всем его объектам.

Итак, VBA отличается от Visual Basic и прочих языков программирования тем, что он предоставляет возможность непосредственной работы с объектами MS Office. Это позволяет эффективно использовать его для автоматизации деятельности, связанной с обработкой различных типов документов. Обычные средства VBA, унаследованные от Visual Basic, — важная, но не определяющая часть языка.

VBA позволяет существенно расширить вычислительные возможности Excel. С помощью VBA можно легко и быстро создавать различные приложения, даже не являясь специалистом в области программирования. VBA имеет графическую инструментальную среду, позволяющую создавать экранные формы и управляющие элементы. С его помощью можно создавать свои собственные функции для Excel, вызываемые мастером функций,

разрабатывать макросы, создавать собственные меню и многое другое. VBA придает документам элегантность и позволяет с легкостью решать многие задачи, о возможности выполнения которых средствами Excel можно только мечтать.

Как и Visual Basic, VBA реализует концепцию визуального программирования, управляемого событиями.

1.3. Объекты, методы, свойства, события

Объектам VBA присуща функциональность — они действуют определенным образом и могут откликаться на определенные ситуации. При этом, если свойства объекта определяют его внешний вид и состояние, методы объекта определяют те задачи, которые может выполнить данный объект. Методы, по сути дела, представляют собой сегмент программного кода, внедренный в объект.

Существует определенный формат программного кода, задающего установку свойства и использование метода:

```
Объект. Свойство = Значение
Объект. Метод [Параметр1 [.....]
```

Здесь **Объект** — имя настраиваемого объекта; **Свойство** — это характеристика, которую нужно изменить; **Метод** — это команда, которая используется для изменения объекта; **Значение** — это новая установка свойства; **Параметр** — это аргумент, используемый методом.

Следующий пример показывает, как можно изменить текст в строке заголовка окна приложения посредством установки нового значения свойству Caption объекта Application:

```
Application.Caption = "Пример установки свойства"
```

Различные типы объектов могут иметь различные методы.

Следующий пример показывает использование метода CountA (подсчет количества не пустых строк) объекта Application для определения номера последней не пустой строки активного рабочего листа:

```
НомерСтроки=Application.CountA(ActiveSheet.Columns(1))
```

Объекты могут реагировать на события — действия пользователя или другие внешние действия, например, щелчок по кнопке, изменение текста, нажатие клавиши и др. Событие представляет собой действие, распознаваемое объектом, для которого можно запрограммировать отклик.

Иногда свойства и методы объекта оказываются связанными в том смысле, что выполнение некоторого метода приводит к изменению свойств объекта. В свою очередь, изменение некоторых свойств может вызвать наступление событий.

Программа может обрабатывать два основных типа событий: инициируемые пользователем и генерируемые системой. События, инициируемые пользователем, возникают в результате его действий: нажатие клавиши, щелчки кнопками мыши. Но есть события, являющиеся следствием действий пользователя.

Таким образом, любое из действий пользователя может вызвать целый набор событий, и порядок их вызова может быть важным. Основными действиями пользователя, генерирующими вызов событий в программе, являются следующие:

- запуск программы;
- нажатие клавиши;
- щелчок кнопкой мыши;
- перемещение мыши;
- выход из программы.

Суть программирования на VBA заключается в двух понятиях: событие и отклик на него. Если пользователь производит какое-то воздействие на систему, допустим, щелкает на кнопке, тогда в качестве отклика выполняется код созданной пользователем процедуры. Этот специальный вид процедур, генерирующих отклик на события, называется процедурами обработки событий. Если такой отклик не создан (не написана соответствующая процедура), то система никак не будет реагировать на это событие. Таким образом, задачей пользователя является разработка программного кода процедур для обработки различных событий, которые являются важными с точки зрения функционирования программы.

Вопросы для самоконтроля

1. Назовите и поясните основные понятия объектно-ориентированного программирования.
2. В чем заключается преимущество технологии ООП?
3. Что такое программирование, управляемое событиями? В чем его особенности?
4. Каково назначение VBA?
5. Что такое событие?
6. Что такое метод объекта?
7. Какие конструкции применяются для установки свойств объектов и доступа к их методам?

2. Проект VBA и его элементы

2.1. Структура проекта VBA

Проект — эта та часть программы, которая видима на экране при ее создании. На рисунке 2.1 приведен пример проекта VBA для приложения Excel.

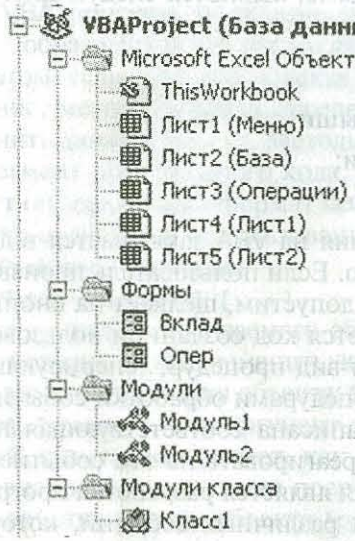

Рис. 2.1. Структура проекта VBA

Как видно из рисунка, проект VBA имеет иерархическую структуру и включает: объекты Excel, формы, стандартные модули и модули класса. Объектами Excel, входящими в проект, являются рабочие книги (WorkBooks), рабочие листы (WorkSheets), диаграммы (Charts). С каждым из этих объектов связан специальный модуль, в который может быть помещен программный код, выполняющий определенные действия.

В модулях класса размещается программный код, описывающий методы класса и его члены — данные для хранения значений свойств. В модулях могут быть размещены программные коды макросов, отдельно выполняемых процедур и функций. *В модулях форм* записываются коды процедур обработки событий формы и элементов управления, размещенных на ней. *В модулях рабочих листов* помещаются процедуры обработки событий рабочих листов и элементов управления, размещаемых на рабочих листах.

Таким образом, проект включает две части: интерфейсную, т.е. видимую при выполнении программы, и программную, которая сосредоточена в различных модулях и реализует выполнение заданных действий.

Весь проект представляет собой один файл — рабочую книгу, и сохраняется вместе с ней.

2.2. Структура программы VBA

Программа VBA представляет собой совокупность процедур и функций, размещенных, в зависимости от особенностей решаемой задачи, в одном или нескольких модулях. Каждый модуль имеет две области: общую область и область подпрограмм. В общей области помещаются операторы описания переменных, которые являются общими для всех процедур и функций этого модуля. В области подпрограмм помещается код программы.

В VBA программный код, реализующий какие-либо действия, оформляется в виде процедур и функций. Благодаря этому создаваемые программы имеют хорошую структурированность и наглядность. Разработанные отдельные функции или процедуры можно накапливать в библиотеках и в дальнейшем по мере необходимости использовать их.

2.2.1. Типы процедур и функций и их определение

2.2.1.1. Определение процедур

Различают следующие типы процедур:

- процедуры обработки событий;
- процедуры макросов;
- процедуры пользователя.

Процедуры обработки событий связаны с каким-либо объектом и имеют следующий синтаксис:

```
Private Sub ИмяОбъекта_Событие()
        << Код обработки события >>
End Sub
```

где `Private` — ключевое слово, определяющее область видимости подпрограммы; `Sub` — ключевое слово, определяющее тип подпрограммы; `ИмяОбъекта` — имя объекта, с которым связывается процеду-

ра; Событие — вид обрабатываемого события; End Sub — ключевые слова, указывающие на окончание блока подпрограммы.

Например, процедура обработки события Click для объекта Кнопка имеет вид:

```
Private Sub CommandButton1_Click()
    <<Код обработки события>>
End Sub
```

Процедуры такого типа вызываются тогда, когда происходит соответствующее событие.

Процедуры обработки макросов создаются при записи макросов. Они имеют синтаксис:

```
Sub ИмяМакроса()
' Имя Макроса Макрос
' Макрос создан Дата
  Программный код макроса
End Sub
```

Макрос может быть вызван комбинацией клавиш, щелчком на значке, связанным с макросом, либо из программы с помощью конструкции вида:

```
Call ИмяМакроса
```

или просто:

```
ИмяМакроса
```

Процедура пользователя создается им для выполнения каких-либо типовых действий в рамках разрабатываемой программы. Синтаксис процедуры пользователя имеет вид:

```
[Private | Public] Sub ИмяПроцедуры(СписокПараметров)
    <<Тело подпрограммы (строки кода)>>
End Sub
```

где [Private | Public] — необязательные ключевые слова, определяющие область видимости подпрограммы; Sub — ключевое слово, определяющее тип подпрограммы; ИмяПроцедуры — имя подпрограммы. СписокПараметров служит для передачи процедуре исходных данных для вычислений (может отсутствовать). Он состоит из элементов списка, разделенных запятыми.

Элемент списка параметров имеет синтаксис:

```
ИмяЭлемента As ТипДанных
```

где ИмяЭлемента — идентификатор; As — ключевое слово; ТипДанных — тип данных элемента списка.

Пример простой подпрограммы, вызывающей макрос:

```
Public Sub AscMe()
    Макрос1
    [Другие операторы]
End Sub
```


Процедура пользователя может быть вызвана из другой подпрограммы, так же как и процедура макроса оператором Call или указанием ее имени:

```
Private Sub Command1_Click()
  Call AscMe 'Вызывается подпрограмма AscMe
  [Другие операторы]
End Sub

Private Sub Command1_Click()
  AscMe 'Вызывается подпрограмма AscMe
  [Другие операторы]
End Sub
```

Используя формальные параметры, в процедуру пользователя можно передавать значения переменных:

```
Private Sub Макрос1()
  Dim Str As String
  Str = "Параметр, передаваемый подпрограмме"
  AscMe (Str)
End Sub

Public Sub AscMe(St As String)
  MsgBox St
End Sub
```

2.2.1.2. Определение функций

Функция — это подпрограмма, которая выполняет действия в пределах своего блока и возвращает единственное значение.

В VBA различают следующие виды функций:

- функции пользователя,
- функции модулей классов.

Функция пользователя имеет следующий синтаксис:

```
[Private | Public] Function ИмяФункции(СписокПараметров)_
    As ТипДанных
  <<Тело функции (строки кода)>>
  ИмяФункции= ВозвращаемоеЗначение
End Function
```

где [Private | Public] — необязательные ключевые слова, определяющие область видимости функции; Function — ключевое слово, указывающее на то, что это функция; ИмяФункции — имя функции;

СписокПараметров — список параметров (может отсутствовать); As — ключевое слово, предваряющее значение типа данных; ТипДанных — тип данных возвращаемого значения; ВозвращаемоеЗначение — значение, возвращаемое функцией; End Function — ключевые слова, указывающие на окончание блока функции.

Обращение к функции может производиться из процедуры, из другой функции. Если в функции предусмотрено рекурсивное обращение, то ее можно вызывать из нее самой. Если функция записана в модуле, то ее можно вызвать из Excel с помощью мастера функций.

При вызове из процедуры или из функции в программном операторе указывается имя функции и передаваемые ей параметры.

Функции модулей класса — это специальные функции, предназначенные для установки свойств объектов разработанного пользователем класса, для получения этих свойств и для определения методов класса. Эти функции могут быть помещены только в модуль класса. Синтаксис этих функций и их назначение будут рассмотрены позднее.

Вопросы для самоконтроля

1. Что такое проект и какие элементы он может содержать?
2. Какие типы модулей может включать проект?
3. Какие существуют типы процедур и функций?
4. Как определяются процедуры и функции?

3. Среда разработки

3.1. Активизация редактора VBA

Проекты VBA создаются с помощью среды разработки, создаваемой редактором Visual Basic.

Для активизации редактора можно использовать следующие варианты:

- выполнить команду меню Excel: Сервис ➤ Макрос ➤ Редактор Visual Basic (Tools ➤ Macro ➤ Visual Basic Editor);
- щелкнуть на кнопке  Редактор Visual Basic (Visual Basic Editor) панели инструментов Visual Basic в окне приложения Excel;
- нажать комбинацию клавиш <Alt>+<F11>.

После выполнения одной из этих команд активизируется редактор Visual Basic и на экране появится его главное окно.

Вернуться из редактора в приложение Excel можно одним из следующих двух действий:

- щелкнуть на кнопке Excel панели задач;
- щелкнуть на кнопке Вид Microsoft Excel (View Microsoft Excel) на панели инструментов редактора  ;
- нажать комбинацию клавиш <Alt>+<F11>.

Чтобы закрыть редактор VBA и вернуться в рабочую книгу, достаточно закрыть главное окно или выполнить команду меню Файл ➤ Закрыть и вернуться в Microsoft Excel.

3.2. Структура редактора VBA

Главное окно редактора обычно занимает весь экран (рис. 3.1). В окне имеются: строка заголовка, меню и панель инструментов. В строке заголовка выводится имя текущей рабочей книги. В главном окне размещаются все другие окна редактора.

3.2.1. Окно проекта

Окно проекта активизируется выполнением команды меню Вид ➤ Окно проекта (View ➤ Project Window) или щелчком на кнопке Окно проекта (Project Window) панели инструментов ре-

дактора . В окне проекта отображается список проектов всех открытых рабочих книг и иерархическая структура каждого проекта (рис. 3.1).

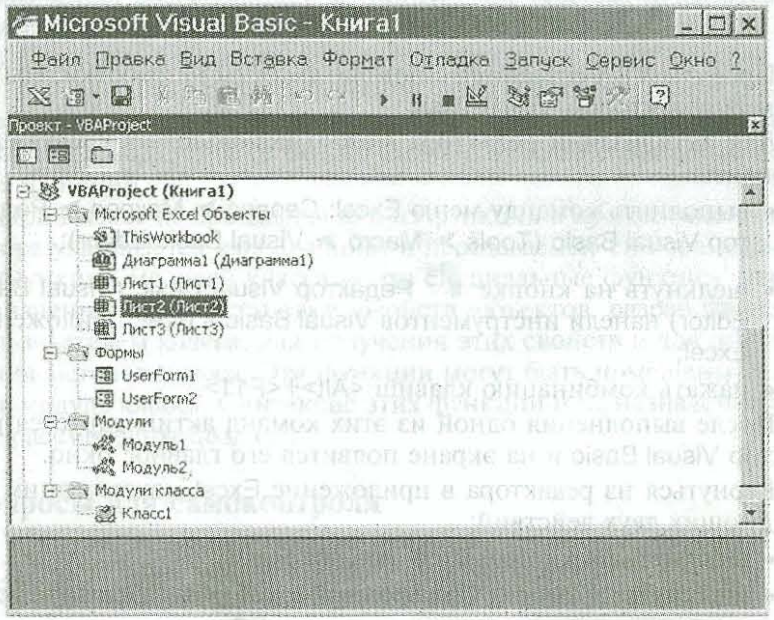

Рис. 3.1. Главное окно и окно проекта

В проекте автоматически создается модуль для каждого рабочего листа и для всей рабочей книги. Кроме того, модули создаются для каждой пользовательской формы, макросов и классов. По назначению модули бывают двух типов — модули объектов и стандартные. К стандартным относятся модули, содержащие макросы или функции. Стандартные модули добавляются в проект командой меню Вставка ➤ Модуль (Insert ➤ Module) или соответствующей командой контекстного меню. Стандартный модуль также может быть создан автоматически при записи макроса.

К модулям объектов относятся модули, связанные с рабочей книгой, рабочими листами, формами и модули класса.

Чтобы удалить какой-либо модуль, его нужно выделить в окне проекта и выполнить команду меню Файл ➤ Удалить (File ➤ Delete).

Благодаря тому, что в окне проекта выводятся проекты всех открытых рабочих книг, можно легко копировать программные коды из одного проекта в другой.

3.2.2. Окно редактирования кода

Окно редактирования кода выполняет функции текстового редактора для ввода и изменения кода процедур и функций проекта. Открыть окно редактирования кода (рис. 3.2) для соответствующего модуля можно, если выполнить одно из следующих действий:

- в окне проекта сделать двойной щелчок на выбранном модуле;
- в окне проекта выделить модуль и выполнить команду меню Вид ➤ Программа;
- в окне проекта выделить модуль и нажать клавишу <F7>.

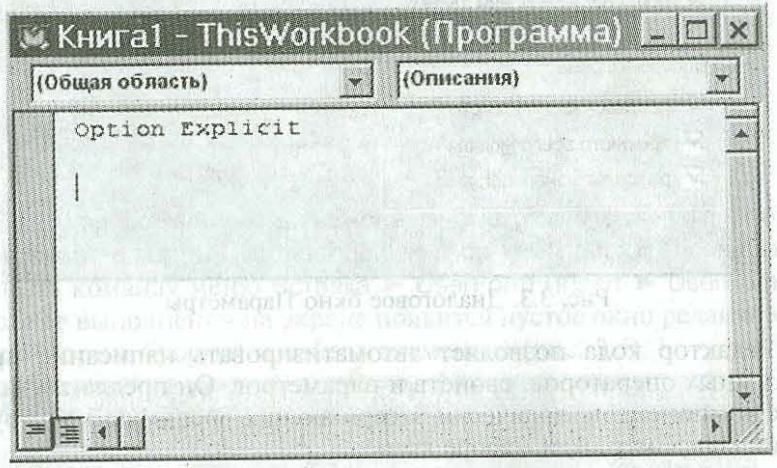

Рис. 3.2. Окно редактирования кода

В окне редактирования можно просмотреть код отдельной процедуры или код всего модуля. Выбор режима просмотра окна редактирования производится одним из двух способов:

- нажатием одной из двух кнопок, размещенных в левом нижнем углу окна редактирования кода;
- установкой или снятием флажка Просмотр всего модуля (Default to Full Module View) в диалоговом окне Параметры (Options), которое открывается при выполнении команды меню Сервис ➤ Параметры (Tool ➤ Options) (рис. 3.3).

В верхней части окна редактирования кода размещены два раскрывающихся списка. Левый список содержит имена объектов, а правый содержит перечень событий, допустимых для объекта, выбранного в левом списке.

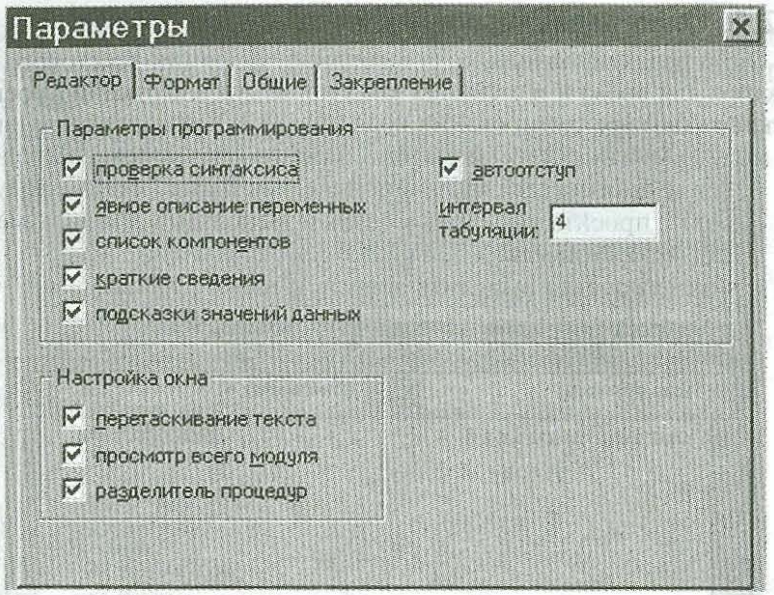

Рис. 3.3. Диалоговое окно Параметры

Редактор кода позволяет автоматизировать написание программных операторов, свойств и параметров. Он предлагает список компонентов, логически завершающих вводимую конструкцию. Например, при наборе кода:

```
Dim Переменная1 As
```

после ввода слова `As` на экране отобразится список компонентов, которые могут логически завершить эту конструкцию (рис. 3.4). Если выполнить двойной щелчок на выбранном элементе списка или нажать клавишу <Tab>, то выбранное имя будет вставлено в код программы.

Редактор кода, кроме перечисленных возможностей, позволяет получить информацию о ключевом слове, процедуре, функции, свойстве или методе. Для этого достаточно установить курсор на нужное ключевое слово и нажать клавишу <F1>.

3.2.3. Окно редактирования формы

Для создания диалоговых окон приложений VBA служат пользовательские формы. Пользовательская форма (UserForm) служит базой диалогового окна, на которой в зависимости от решаемой задачи размещают нужные элементы управления.

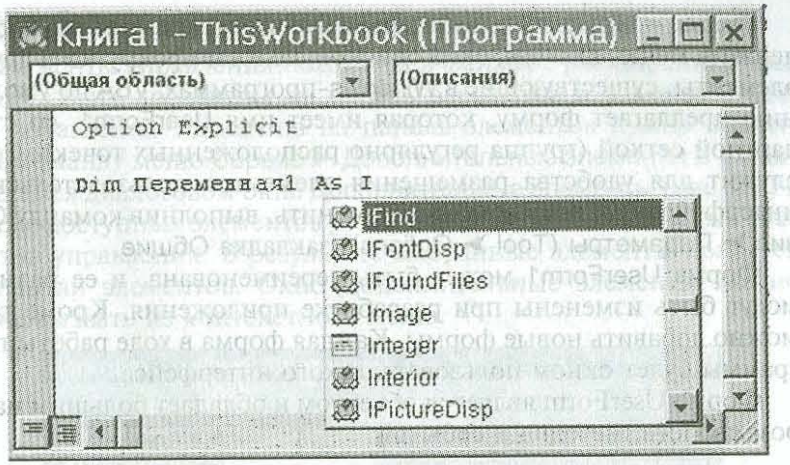

Рис. 3.4. Список компонентов

Редактор форм является основным инструментом визуального программирования. Чтобы добавить форму в проект, нужно выполнить команду меню Вставка ➤ UserForm (Insert ➤ UserForm). После ее выполнения на экране появится пустое окно редактирования формы (форма) и панель элементов (рис. 3.5).

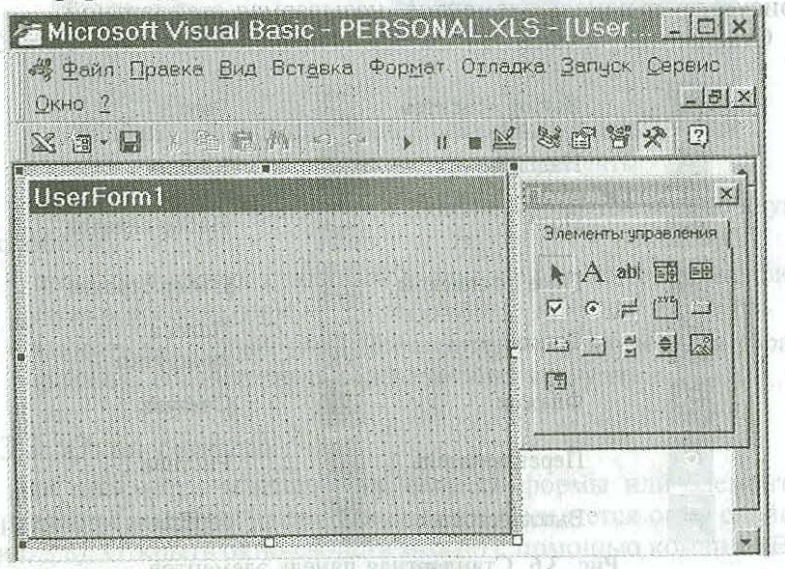

Рис. 3.5. Окно редактирования формы

Форма — это окно в интерфейс пользователя. Она может содержать меню, кнопки, окна списков, полосы прокрутки и другие элементы, существующие в Windows-программах. VBA по умолчанию предлагает форму, которая имеет имя UserForm1, со стандартной сеткой (группа регулярно расположенных точек). Сетка служит для удобства размещения элементов пользовательского интерфейса. Шаг сетки можно изменить, выполнив команду Сервис ➤ Параметры (Tool ➤ Options), закладка Общие.

Форма UserForm1 может быть переименована, и ее размеры могут быть изменены при разработке приложения. Кроме того, можно добавить новые формы. Каждая форма в ходе работы программы будет окном пользовательского интерфейса.

Форма UserForm является объектом и обладает большим набором свойств, методов и событий.

3.3. Панель элементов

Панель элементов содержит элементы управления, которые можно поместить в форме. Открыть панель элементов можно с помощью команды Вид ➤ Панель элементов или инструмента  .

В исходном состоянии на рабочей поверхности главного окна присутствует панель элементов, называемая стандартной.

Стандартные элементы управления представлены на рисунке 3.6.

Рис. 3.6. Стандартная панель элементов

3.3.1. Дополнительные элементы управления

Элементы управления находятся в файлах с расширением .ocx и могут быть добавлены на панель элементов и удалены с нее. Для добавления элементов на панель элементов нужно выполнить команду меню Сервис ➤ Дополнительные элементы и в появившемся диалоговом окне Дополнительные элементы (рис. 3.7) в списке доступных элементов установить флажки для нужных элементов управления. В результате выбранные элементы появятся на панели элементов. Окно Дополнительные элементы можно также вызвать из контекстного меню.

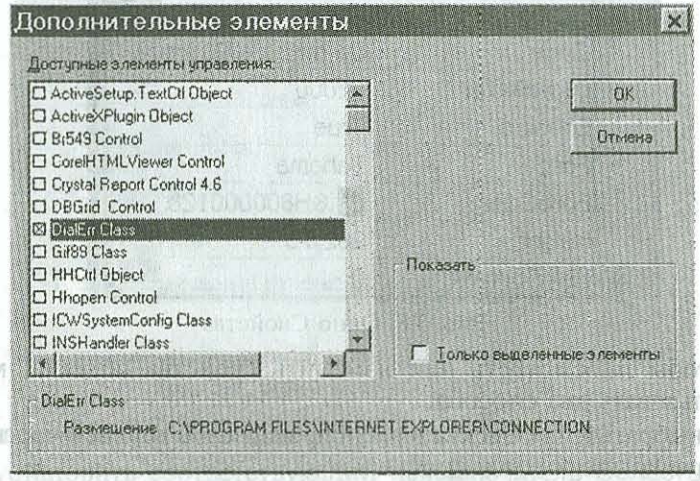

Рис. 3.7. Окно Дополнительные элементы

Удалить элемент управления с панели элементов можно двумя способами:

- щелкнуть *правой* клавишей мыши на элементе и в появившемся контекстном меню выполнить команду Удалить;
- вызвать диалоговое окно Дополнительные элементы и убрать флажок, соответствующий названию элемента.

3.3.2. Окно свойств

Для просмотра и изменения свойств формы или элементов управления во время проектирования используется окно свойств (рис. 3.8). Открыть окно свойств можно с помощью команды Вид ➤ Окно свойств или инструмента  .

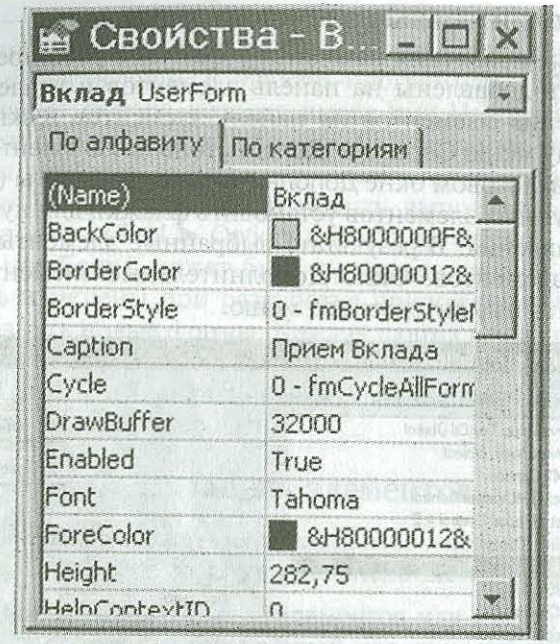

Рис. 3.8. Окно Свойства

Чтобы просмотреть или изменить свойства объекта, можно использовать два способа:

- выбрать имя объекта в расположенном в верхней части окна раскрывающемся списке;
- маркировать объект в форме или саму форму (щелкнуть один раз).

После выполнения одного из этих действий в окне Свойства появится список свойств объекта и их значения. Чтобы изменить значение свойства, нужно найти его в списке и изменить значение в правой колонке. Если в области значений появились три точки, то нужно щелкнуть на них, чтобы открылось диалоговое окно. Если появилась указывающая вниз стрелка, то щелчок на ней раскрывает список возможных значений.

3.4. Создание функций пользователя

Для создания функции пользователя нужно выполнить действия:

- если в проекте нет модуля, то создать его, выполнив команду меню редактора Вставка ➤ Модуль;
- выполнить команду меню редактора VB Вставка ➤ Процедура;

- в открывшемся диалоговом окне Вставка процедуры установить переключатель Функция (рис. 3.9);

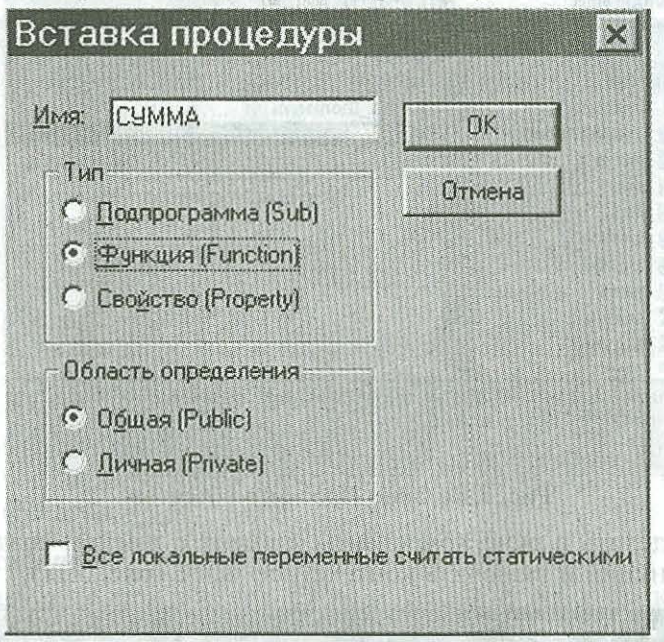

Рис. 3.9. Окно Вставка процедуры

- в окне Имя ввести имя функции;
- установить соответствующий переключатель Область определения;
- щелкнуть на ОК. После выполнения этих действий в окне модуля появится заготовка функции (заголовок и окончание), между которыми нужно поместить код тела функции;
- ввести список параметров функции, их типов данных, а также указать тип возвращаемого функцией значения;
- используя команду меню Вид ➤ Просмотр объектов или нажав клавишу <F2> вызвать окно Просмотр объектов;
- раскрыть список верхнего левого окна (Список проектов) и выбрать из него VBA Project. В окне Классы отобразятся элементы текущего проекта;
- выбрать в этом окне модуль, в котором создана функция — в окне Компоненты отобразятся элементы, которые содержатся в этом модуле (рис. 3.10);

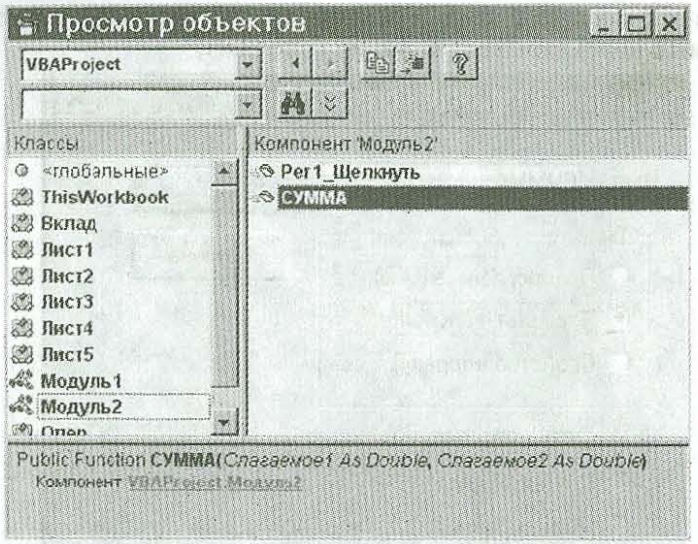

Рис. 3.10. Окно Просмотр объектов

- выделить в окне Компоненты элемент с именем созданной функции и включить контекстное меню (рис. 3.11);

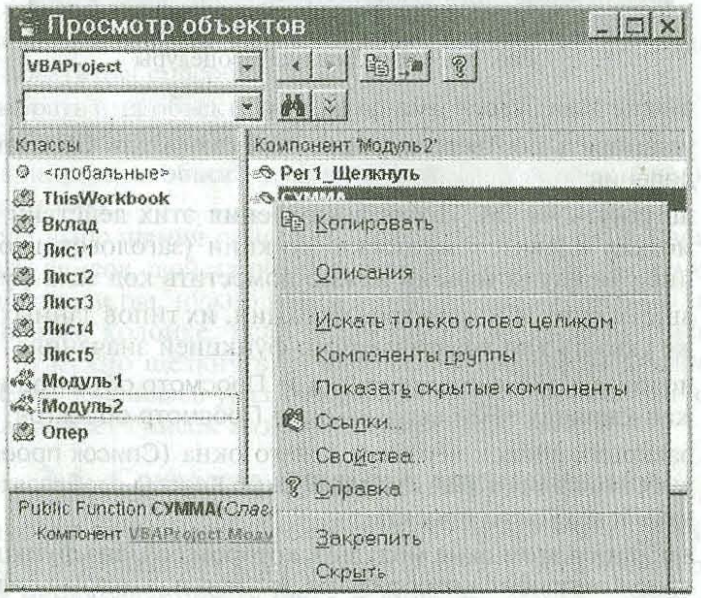

Рис. 3.11. Контекстное меню

- выполнить команду контекстного меню Свойства — откроется окно Параметры компонента (рис. 3.12).

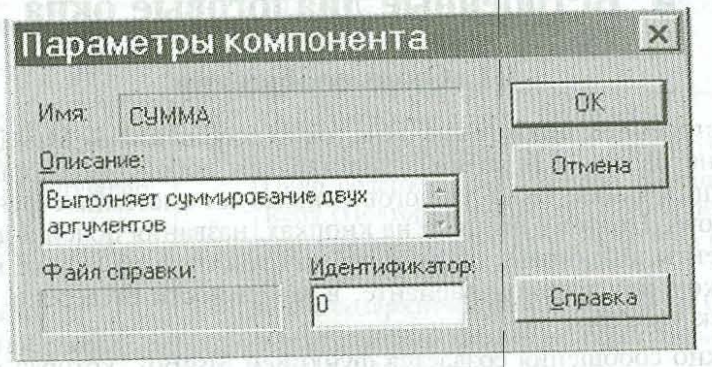

Рис. 3.12. Окно Параметры компонента

- в поле Описание этого окна ввести текст краткого описания функции, если необходимо, то указать файл справки и идентификатор.

Упражнение 1

1. Загрузите приложение Excel и создайте рабочую книгу с именем «Упражнения по программированию».
2. Одним из способов запустите редактор VBA.
3. В проект добавьте модуль.
4. Откройте окно редактирования кода для созданного модуля.
5. Закройте окно редактирования кода для модуля.
6. В проект добавьте форму пользователя.
7. Откройте окно редактирования кода для формы.
8. Удалите форму из проекта.
9. В созданный модуль, выполняя команду меню Вставка ➤ Подпрограмма, поместите функцию с именем СУММА;
10. Сохраните рабочую книгу.

4. Встроенные диалоговые окна

4.1. Окно сообщения

Встроенные диалоговые окна (окно сообщения — MessageBox и окно ввода — InputBox) представляют собой операторы или функции языка. Окна диалогов используют системные функции Windows. Поэтому надписи на кнопках, названия полей и другие элементы диалоговых окон могут отображаться на экране в английском или русском варианте, в зависимости от версии и настроек Windows.

Окно сообщения создается функцией MsgBox, которая имеет следующий синтаксис:

```
MsgBox(prompt[, buttons] [, title] [, helpfile, context])
```

Функция возвращает значение, соответствующее выбранной пользователем кнопке в окне сообщения. Ниже приведены значения и имена возвращаемых констант (табл. 4.1). Имена можно использовать вместо значений в любом месте программы.

Таблица 4.1.

Константы для определения параметра `prompt`

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)

Параметр `prompt` обязательный. Это строка, которая выдается в окне сообщения. Ее длина ограничена 1024 символами.

Параметр `buttons` необязательный (табл. 4.2). Значение параметра — целое число, равное сумме значений, определяющих набор кнопок, коды значков, кнопки по умолчанию в окне сообщения, а также модальность окна. Возможные значения описаны ниже. По умолчанию значение параметра равно 0.

Таблица 4.2

Константы для определении параметра `buttons`

Константа	Значение	Описание
Наборы кнопок окна сообщения		
VbOKOnly	0	Окно содержит только кнопку OK
VbOKCancel	1	Окно содержит кнопки OK и Cancel (Отмена)
vbAbortRetryIgnore	2	Окно содержит кнопки Abort (Прервать), Retry (Повторить) и Ignore (Пропустить)
vbYesNoCancel	3	Окно содержит кнопки Yes (Да), No (Нет) и Cancel (Отмена)
vbYesNo	4	Окно содержит кнопки Yes (Да) и No (Нет)
vbRetryCancel	5	Окно содержит кнопки Retry (Повторить) и Cancel (Отмена)
Пиктограммы окна сообщения		
vbCritical	16	Добавляет пиктограмму «Критическое сообщение». Часто после такого сообщения программа прекращает работу
vbQuestion	32	Добавляет пиктограмму «Запрос». Обычно используется, когда для продолжения работы программы требуется дополнительная информация.
vbExclamation	48	Добавляет пиктограмму «Предупреждение»
vbInformation	64	Добавляет пиктограмму «Информация». Чаще всего используется для сообщения о завершении выполнения некоторой задачи
Кнопка по умолчанию		
vbDefaultButton1	0	Первая
vbDefaultButton2	256	Вторая
vbDefaultButton3	512	Третья
vbDefaultButton4	768	Четвертая
Модальность окна		
vbApplicationModal	0	Модальность уровня приложения. Пока вы не нажмете одну из кнопок окна, вы не сможете вернуться в приложение, породившее это окно. При этом можно переключиться на другое приложение.
vbSystemModal	4096	Модальность системного уровня. Пока не будет закрыто это окно, никакие приложения не доступны.

Параметр title задает строку, которая является заголовком окна сообщения. Если параметр отсутствует, то в качестве заголовка используется имя приложения.

Окно сообщений может иметь еще два необязательных параметра (helpfile и context), касающихся справочной информации, относящейся к данному сообщению. О них можно прочитать в справочной системе.

Например, в результате выполнения оператора

```
Ans = MsgBox("Закончить?", vbYesNo + vbQuestion + _
    vbDefaultButton1, "Пример окна MsgBox")
```

появится окно сообщения как на рис. 4.1.

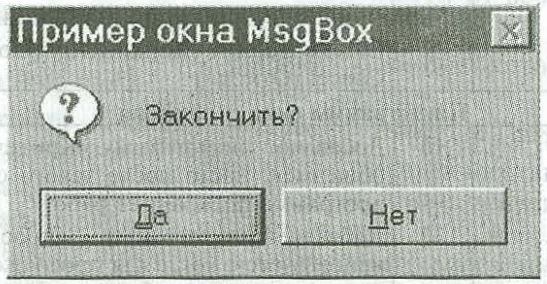

Рис. 4.1. Пример окна сообщения

Чтобы определить, какая кнопка была нажата, значение переменной Ans анализируется с помощью оператора If. Например:

```
If Ans = vbYes then UserForm1.Hide
```

4.2. Окно ввода

Окно ввода служит для ввода данных. Оно создается функцией InputBox(), имеющей следующий синтаксис:

```
InputBox(prompt[, title] [, default] [, xpos] [, ypos])
```

Окно содержит сообщение, указывающее, какие данные должен ввести пользователь, поле текста для ввода данных и две кнопки ОК и Отмена, которые используются для подтверждения или отмены ввода данных. Закончив ввод данных, пользователь должен щелкнуть на одной из кнопок. Если щелчок был сделан на кнопке ОК, то значением функции является текст, находящийся в поле ввода. Если щелчок был сделан на кнопке Отмена, то значением функции является пустая строка, независимо от того, что напечатал пользователь.

Параметры функции имеют следующий смысл: prompt — строка сообщения, которая будет напечатана в окне; title — строка, которая является заголовком окна. Если параметр не указан, то в качестве заголовка используется имя приложения; default — строка, помещаемая в текстовое поле (если параметр не указан, то поле текста будет пустым); xpos — расстояние в твипах от левой границы экрана до левой границы окна (если параметр не указан, окно центрируется по горизонтали); ypos — расстояние в твипах от верхней границы экрана до верхней границы окна.

Окно ввода может иметь еще два необязательных параметра (helpfile и context). О них можно прочитать в справочной системе.

Для преобразования введенной строки в другой тип данных используйте функции преобразования типов: CCur(), CDate(), CInt(), CLng(), CSng(), CVar() и другие.

В качестве примера на рисунке 4.2 приведено окно, созданное с помощью оператора:

```
A = InputBox("Введите значение:", "Пример окна InputBox")
```

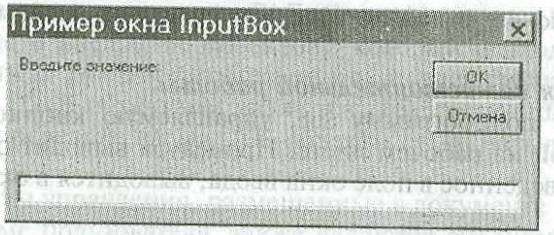

Рис. 4.2. Пример окна ввода

Упражнение 2

Создание простого макроса с использованием окон ввода и вывода

1. Поместите на рабочем листе кнопку панели Формы (рис. 4.3). Появится диалоговое окно Назначить макрос объекту.
2. В диалоговом окне Назначить макрос объекту шелкните на кнопке Создать. Запустится редактор Visual Basic и откроется окно редактирования кода макроса.
3. В процедуру обработки события Кнопка_Щелкнуть поместите следующий код:

```
Sub Кнопка1_Щелкнуть()
  Dim Строка As String
  Строка = InputBox("Введите значение:", "Пример окна InputBox")
  MsgBox Строка, vbExclamation, "Пример окна MsgBox"
End Sub
```

4. Щелкните на кнопке, размещенной на рабочем листе. Макрос начнет выполняться. Появится окно ввода (рис. 4.3).
5. Наберите в поле ввода окна какой-либо текст и щелкните на кнопке OK. Появится окно вывода, в котором будет выведен ранее введенный текст .
6. Щелкните на OK. Макрос завершит работу.

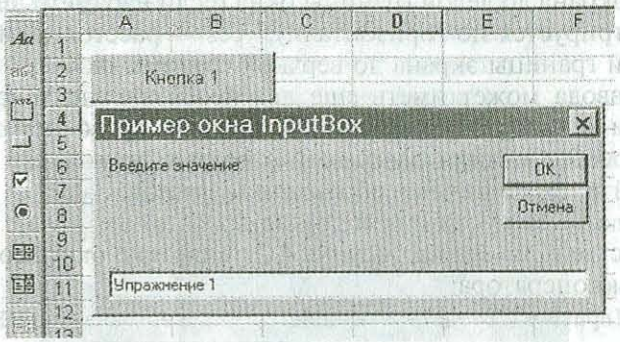

Рис. 4.3. Вывод окна ввода из процедуры макроса

Задание для самостоятельной работы

Создайте подпрограмму Sub, управляемую кнопкой формы, размещенной на рабочем листе. Процедура выводит окно ввода. Значение, введенное в поле окна ввода, выводится в окне вывода.

5. Объекты, их основные свойства, методы и события

5.1. Типы объектов VBA

Все объекты, с которыми работает VBA-программа, разделяются на три вида:

- объекты приложения;
- элементы управления;
- объекты ActiveX.

Все визуальные объекты приложения являются его объектами. Каждое из приложений имеет различные встроенные объекты. Например, для приложения Excel объектами являются *рабочий лист* (WorkSheet), *диапазон* (Range). Всего в VBA имеется более 100 встроенных объектов.

VBA использует механизм OLE (Object Linking and Embedding — связывание и внедрение объектов), который позволяет взаимодействовать с любыми программами, поддерживающими OLE. Например, OLE-объектами являются объекты WordArt, ClipArt и другие. В последнее время OLE-объекты стали называть объектами ActiveX.

Элементы управления, размещенные в документе приложения или в форме пользователя, также являются объектами. С их помощью осуществляется управление программой или вводятся в нее исходные данные.

В VBA существует возможность просмотра списка объектов, для этого есть специальное средство — окно Просмотр объектов (Object Browser). С его помощью можно просмотреть структуры классов объектов, их свойства, методы и события, а также получить контекстную справку.

Для вызова Object Browser нужно выполнить команду меню редактора Вид ➤ Просмотр Объектов (Object Browser) или нажать клавишу <F2>.

В левом верхнем углу окна просмотра объектов расположен раскрывающийся комбинированный список Проекты/Библиотеки (Project/Library), содержащий имена проектов и библиотек, доступных из приложения. В окне Классы (Classes) отображается структура класса или проекта — совокупность встроенных объектов. Если выбрать один из них, то в соседнем окне Компоненты

(Members) отобразятся элементы этого объекта (свойства, методы, события). Каждый элемент — библиотека, класс, проект, свойство, метод, событие, свойство типа перечисления обозначается соответствующим значком. Щелкнув по кнопке Справка, можно получить справку по выбранному элементу.

Для быстрого поиска нужного объекта в окне Просмотр объектов имеются инструменты поиска. Для этого ниже окна Проекты/Библиотеки расположен еще один комбинированный список. В нем можно задать имя элемента поиска, а затем щелкнуть по рядом расположенной кнопке поиска или нажать клавишу <Enter>. При необходимости поиск можно осуществлять во всех библиотеках (All libraries). Результаты поиска отображаются в специальном окне Результаты поиска (Search Results) (рис. 5.1).

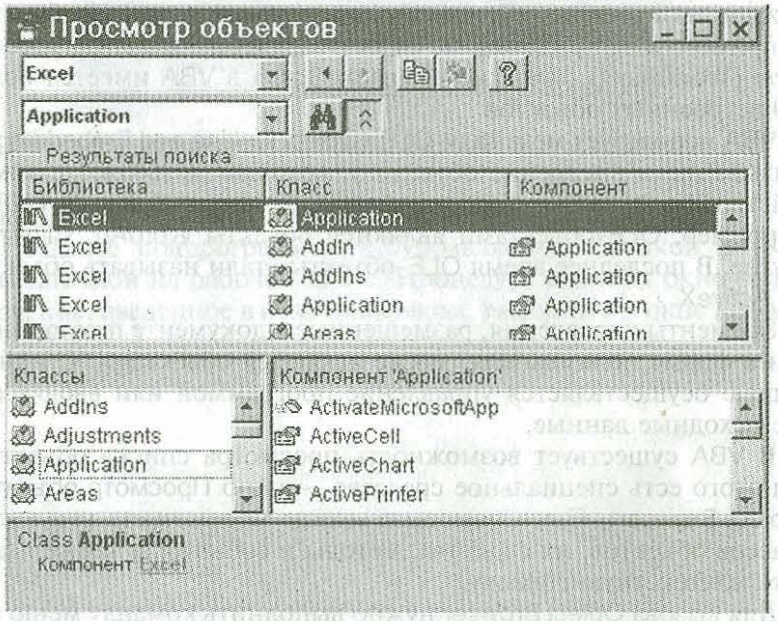

Рис. 5.1. Результаты поиска

В нижней части окна Просмотр объектов (Object Browser) расположено еще одно полезное средство — панель Детали (Details), отображающее сведения о выбранном элементе. Эта информация может содержать гиперссылки, щелкая по которым можно получить следующие порции информации. Для свойств в этом окне указывается их тип, для методов — полное описание всех аргументов, включая обязательные и возможные. Информацию из этого окна можно копировать в буфер обмена или непосредственно перемешать в свой текст.

5.2. Свойства, методы и события элементов управления

Элементы управления — это объекты, которые можно поместить в окне формы. Как все объекты, они имеют свойства и методы. Свойства элементов управления определяют их внешний вид (положение, размер, цвет) и поведение. Изменять свойства элементов управления можно как во время проектирования, так и во время выполнения программы. Метод — это процедура, которая воздействует на объект во время выполнения. Например, для перемещения элемента управления используется метод Move.

Свойство Name определяет имя, которое используется для ссылок на элемент управления в программе. Имена должны удовлетворять условиям, предъявляемым к именам в языке VBA. Можно использовать русские буквы. Рекомендуется сразу после того, как вы поместили элемент управления в форму, изменить имя, заданное по умолчанию, на другое, отражающее назначение объекта. Если вы где-нибудь в программе используете имя элемента управления, а потом поменяете значение свойства Name, то в тексте оно не изменится. В программе может быть много элементов управления, поэтому рекомендуется давать им имена, состоящие из двух частей: префикса, определяющего тип элемента, и собственно имени.

Существуют свойства, которые для всех или для многих элементов управления называются одинаково и имеют один и тот же смысл. Эти свойства приводятся в ниже. В дальнейшем они не будут указываться для элементов, а будут описываться только специфические свойства каждого элемента.

5.2.1. Общие свойства стандартных элементов управления

Name — Имя, которое используется для ссылок на элемент управления в программе. Нельзя изменить во время выполнения программы.

Left — Позиция элемента управления относительно левого края формы или рамки.

Top — Позиция элемента управления относительно верхнего края формы или рамки.

Height	Высота элемента управления.
Width	Ширина элемента управления.
Caption	Текст заголовка или надписи.
Enabled	Определяет, является ли элемент управления доступным. Возможные значения True/False. Если значение свойства равно False, элемент не доступен пользователю.
Visible	Определяет, будет ли элемент управления виден на экране во время выполнения программы (True/False). Если значение свойства равно False, элемент не виден на экране.
TabIndex	Определяет порядок перемещения от объекта к объекту с помощью клавиш <Tab> или <Shift>+<Tab>.

5.2.2. Общие методы стандартных элементов управления

SetFocus	Передает фокус объекту.
Drag	Служит для перетаскивания элемента управления по технологии Drag&Drop.
Move	С помощью этого метода элемент управления перемещают по форме, при этом можно изменять его размеры.

5.2.3. Элемент Кнопка

Элемент Кнопка (CommandButton) очень часто используется при разработке интерфейса. На поверхности кнопки можно разместить надпись или рисунок, или и то и другое. Ниже описываются основные свойства и события элемента.

Свойства элемента Кнопка

Caption	Задает текст надписи на кнопке.
Cancel	Только одна из кнопок формы может иметь значение этого свойства True. Это должна быть кнопка, выполняющая функции кнопки Cancel (Отмена). Если форма активна, и пользователь нажмет <Esc>, то будут выполнены действия, связанные с этой кнопкой.
Default	Только одна из кнопок формы может иметь значение этого свойства True. Если ни одна из кнопок формы не находится в фокусе, и пользователь нажал <Enter>, то будут выполнены действия, связанные с этой кнопкой.
Picture	Определяет рисунок на поверхности кнопки. Во время выполнения для изменения свойства нужно использовать функцию LoadPicture. Например,

```
Btn1.Picture = LoadPicture("c:\Pic\copy.bmp")
```


Picture-Position	Определяет расположение рисунка относительно надписи.

События элемента Кнопка

Click	Возникает при нажатии пользователем кнопки мышью или на клавиатуре.
DblClick	Возникает при двойном щелчке мыши на кнопке.

5.2.4. Элемент Поле

Элемент Поле (TextBox) обеспечивает возможность ввода текста пользователем. Текстовые окна поддерживают ввод и редактирование текста без всякого вмешательства с вашей стороны. *Вырезать*, *копировать* и *вставлять* текст можно с помощью стандартных для Windows клавиш: <Ctrl>+<X>, <Ctrl>+<C>, <Ctrl>+<V>. Ниже описываются основные свойства элемента.

Свойства элемента Поле

Text	Главное свойство, содержащее текст, введенный пользователем или присвоенный ему программой. Тип значения `string`. Если вы используете элемент для ввода чисел или дат, то это значение нужно преобразовать в значение соответствующего типа, используя функции `Cdate`, `CInt`, `CDbl` и другие.
SelText	Содержит выделенный в поле текст.
Locked	При `Locked = True` пользователь не может ввести текст в поле.
MultiLine	При `MultiLine = True` поле может содержать более одной строки.
ScrollBars	Определяет наличие полос прокрутки при многострочном режиме.
BorderStyle	Определяет стиль обрамления.
BackColor	Определяет цвет фона.
BorderColor	Определяет цвет рамки.
ForeColor	Определяет цвет шрифта
Font	Используется для установки параметров шрифта.
PasswordChar	Если текстовое поле используется для введения пароля, то в это свойство записывается замещающий символ. Если свойство содержит пустую строку, то режим обычный.

5.2.5. Элемент Надпись

Элемент Надпись (Label) обычно используется для вывода различных текстов в форме. Он может содержать и рисунок. Пользо-

ватель не может изменить надпись, но программа в период выполнения может изменять значение надписей.

Свойства элемента Надпись

Caption	Содержит текст надписи.
Picture	Определяет рисунок, помещаемый в элемент. Во время выполнения для изменения свойства нужно использовать функцию LoadPicture. Например, `Lbl1.Picture = LoadPicture("c:\Pic\flower.bmp")`
PicturePosition	Определяет расположение рисунка относительно надписи.
AutoSize	По умолчанию имеет значение False. Если изменить на True, то элемент будет автоматически менять свои размеры в соответствии с текстом.
BackStyle	Задает режим для фона.
BorderStyle	Определяет стиль обрамления.
BackColor	Определяет цвет фона.
BorderColor	Определяет цвет рамки.
ForeColor	Определяет цвет шрифта
Font	Используется для установки параметров шрифта.

5.2.6. Элемент Поле со списком

Элемент управления Поле со списком (ComboBox) дает возможность пользователю выбрать нужную информацию из списка возможных значений или ввести ее в поле списка. Список в элементе Поле со списком может содержать несколько столбцов. Строки и столбцы нумеруются с начиная с числа 0. Вид элемента приведен на рисунке 5.2.

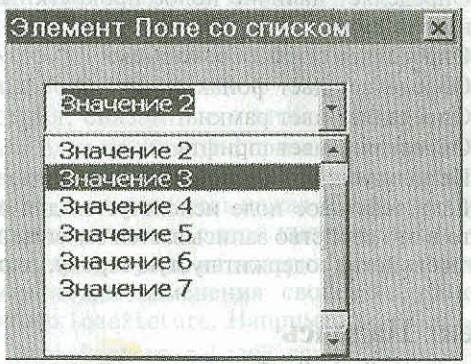

Рис. 5.2. Вид элемента Поле со списком

Основные свойства элемента Поле со списком

List	Используется для доступа к элементу списка. В качестве параметров указываются номера строк и столбцов. Также можно использовать для инициализации списка. Например, если в программе описан массив Dim MyArray(10), то присвоить списку значения этого массива можно, выполнив оператор:
	`MyList.List() = MyArray`
	Обратиться к элементу списка можно так:
	`MyList.List(i) = MyList.List(i) + 1`

ListIndex	Содержит номер текущей строки, который равен 1, если никакой элемент не выбран.
ListCount	Количество строк в списке.
RowSource	Определяет источник элементов списка. В качестве значения используется ссылка на диапазон рабочего листа Microsoft Excel. Например, `MyList.RowSource = "A1:A10"`
Text	Содержит выбранное или введенное значение, которое отображается в текстовом поле.
Style	Определяет, как пользователь может ввести значения в поле списка: 0 — поле с раскрывающимся списком (позволяет ввести данные, которых нет в списке); 2 — раскрывающийся список (не позволяет ввести новые данные).

Основные методы элемента Поле со списком

AddItem	`Object.AddItem` *строка*`[,` *индекс*`]` Добавляет элемент *строка* в список. Если задан *индекс*, то элемент помещается в указанную позицию. Если индекс не задан, то элемент добавляется в конец списка.
RemoveItem	`Object.RemoveItem (`*индекс*`)` Удаляет из списка элемент с заданным индексом
Clear	Удаляет все строки из списка.

Основные события элемента Поле со списком

Change	Введено или выбрано новое значение списка.
Enter	Возникает перед тем как элемент получит фокус от другого элемента на этой же форме.
Exit	Возникает непосредственно перед тем, как фокус будет передан от данного элемента другому на той же форме.

5.2.7. Элемент Список

Элемент Список (ListBox) применяется для хранения списка значений. Во время работы приложения пользователь может выбрать из списка одно или несколько значений. Вид элемента приведен на рисунке 5.3.

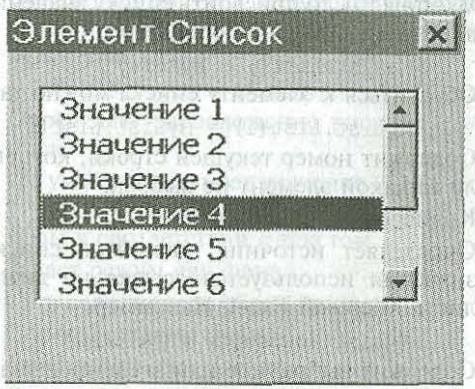

Рис. 5.3. Элемент Список

Основные свойства элемента Список

List, ListCount, RowSource	Имеют такой же смысл, как у элемента Поле со списком.
ListIndex	Содержит номер текущей строки. Равно 1, если никакой элемент не выбран. Если выделено несколько строк, то равно номеру строки, которая имеет фокус.
MultiSelect	Определяет, можно ли выделить несколько элементов списка.
Selected	Массив, состоящий из того же количества элементов, что и список. Для каждого элемента списка свойство равно `True`, если элемент выделен, и `False`, если нет.
Text	Возвращает выбранный в списке элемент.

Основные методы и события элемента Список такие же, как у элемента Поле со списком.

5.2.8. Элемент Рамка

Элемент Рамка (Frame) используется в приложениях для создания визуальных или функциональных групп элементов, чаще всего переключателей и флажков. Он относится к элементам-контейнерам.

Контейнером называется элемент управления (или объект вообще), содержащий другие элементы управления. *Дочерний элемент управления* — это то, что содержится в контейнере. Например, поместим кнопку в элемент управления Рамка. Тогда кнопка прикрепляется к краям рамки. Кнопка будет перемещаться, когда перемещается рамка, а свойства Top и Left кнопки будут определены относительно рамки. Удаляя контейнер, вы удаляете и все элементы, содержащиеся в нем.

Создавая группу элементов, сначала поместите в форму элемент Рамка, а потом — дочерние элементы. Вид элемента приведен на рисунке 5.4.

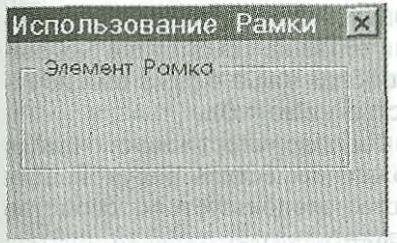

Рис. 5.4. Элемент Рамка

Основным свойством элемента Рамка является Caption. Оно содержит текст, который выводится в левом верхнем углу рамки и обычно является названием группы элементов.

Если вы хотите использовать в качестве фона внутри рамки рисунок, то укажите его в свойстве Picture. Свойства PictureAlignment, PictureSizeMode и PictureTiling определяют расположение и размер рисунка.

5.2.9. Элемент Флажок

Элемент управления Флажок (CheckBox) дает возможность пользователю осуществить выбор типа Да/Нет. Можно создать группу элементов, однако все флажки в этой группе будут независимы друг от друга. Вид элемента приведен на рисунке 5.5.

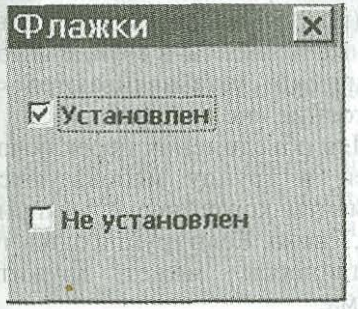

Рис. 5.5. Элемент Флажок

Основные свойства элемента Флажок

Caption	Надпись, которая выводится рядом с элементом.
Value	1 (True) — флажок установлен; 0 (False) — флажок не установлен

Основное событие для этого элемента — Click. Оно возникает при изменении свойства Value либо с помощью мыши, либо с помощью клавиатуры (когда элемент находится в фокусе и нажата клавиша Пробел), либо программным путем.

5.2.10. Элемент Выключатель

Элемент Выключатель (ToggleButton) предназначен для тех же целей, что и элемент Флажок. Они отличаются только внешним видом. Выключатель имеет вид кнопки, которая может находиться в двух состояниях: отпущена или утоплена. На рисунке 5.6. выключатель 1 включен (значение свойства Value = 1), а выключатель 2 — отключен (значение свойства Value = 0).

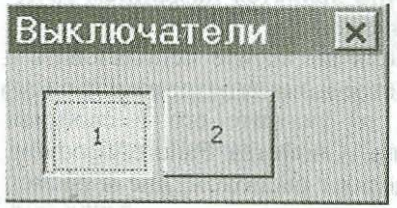

Рис 5.6. Элемент Выключатель

На поверхности выключателя можно поместить рисунок. Для этого используется свойство Picture.

5.2.11. Элемент Переключатель

Переключатели (элементы OptionButton) обычно объединяются в группу. Они предназначены для выбора одного из нескольких взаимоисключающих значений (рис. 5.7). При выборе пользователем нужного элемента остальные элементы в группе автоматически устанавливаются в состояние «не выбран».

Рис. 5.7. Элемент Переключатель

Для объединения переключателей в группу используются элементы-контейнеры, например, Рамка. Если группа одна, то контейнером может служить форма.

Основные свойства элемента Переключатель

Caption	Надпись, которая выводится рядом с элементом.
Value	True — позиция выбрана (помечена точкой); False — позиция не выбрана

Основное событие элемента Переключатель — Click. Оно возникает при изменении значения свойства Value. В группе переключателей событие возникает только для того элемента, чье значение становится True.

5.2.12. Элемент Набор вкладок

Элемент Набор вкладок (TabStrip) (рис. 5.8) представляет собой элемент-контейнер, состоящий из прямоугольной области, в которую вы можете поместить другие элементы, и строки ярлычков. При работе с элементом у пользователя создается впечатление, что он оперирует несколькими вкладками. На самом деле вкладка не содержит элементов, они находятся на форме. Используя соответствующие события элемента Набор вкладок (например, Click или Change), можно менять свойства элементов размещенных в контейнере в зависимости от выбранного ярлычка. Поэтому на экране вкладки выглядят по-разному, хотя все время отображаются одни и те же объекты.

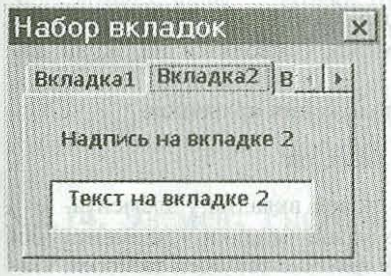

Рис. 5.8. Элемент Набор вкладок

Когда вы помещаете элемент Набор вкладок в форму, он содержит две вкладки. Чтобы *добавить*, *удалить*, *переименовать* вкладку или *изменить порядок* вкладок, нужно выполнить соответствующую команду контекстного меню. Для вызова меню используйте *правую* кнопку мыши. Предварительно выберите необходимую вкладку, щелкнув по ее ярлычку.

Во время выполнения программы каждой вкладке соответствует объект Tab. Все они объединены в семейство Tabs. Элементы в семействе пронумерованы, начиная с 0. Для доступа к конкретному элементу можно использовать его номер или имя.

Чтобы добавить или удалить вкладку во время выполнения программы, нужно применить методы семейства Tabs.

Объекты Tab и семейство Tabs не имеют собственных событий. Следует использовать события элемента TabStrip.

Основные свойства элемента Набор вкладок

SelectedItem	Возвращает выбранный в данный момент объект Tab. Используется для доступа к свойствам текущего объекта Tab. Например, `TabStrip1.SelectedItem.Caption`
Value	Номер активной вкладки.

Tabs	Используется для доступа к семейству Tabs или конкретному объекту семейства. Например, чтобы получить доступ к свойству Caption вкладки с номером 1 у элемента TabStrip1, можно использовать выражение
	`TabStrip1.Tabs(1).Caption` или
	`TabStrip1.Tabs.Item(1).Caption`
	Если переменная TabName содержит имя нужной вкладки, то выражение может иметь вид
	`TabStrip1.Tabs(TabName).Caption` или
	`TabStrip1.Tabs.Item(TabName).Caption`

MultiRow	Определяет, может ли элемент иметь больше одной строки ярлычков. Если значение равно True, ярлычки могут располагаться в несколько строк. Если значение свойства равно False, то все ярлычки располагаются в одну строку. При необходимости появляется полоса прокрутки, позволяющая просматривать все ярлычки.
Style	Определяет стиль элемента (выводятся на экран ярлычки или кнопки).
TabOrientation	Определяет место расположения ярлычков.

Основные события элемента Набор вкладок

Change	Возникает при выборе новой вкладки у элемента.
Click	Возникает при выборе новой вкладки у элемента или при щелчке мышью на ярлычке текущей вкладки. Параметром является номер выбранной вкладки.

Семейство Tabs имеет единственное свойство Count. Оно равно количеству объектов в семействе.

Методы семейства Tabs

Add	Создает новую вкладку.
Clear	Удаляет все объекты из семейства Tabs.
Item	Возвращает вкладку с указанным номером.
Remove	Удаляет вкладку.

5.2.13. Элемент Набор страниц

Элемент Набор страниц (MultiPage) позволяет создавать многостраничные диалоговые окна (рис. 5.9). Каждая страница — это форма, содержащая свои собственные элементы. Для перехода между страницами обычно используются ярлычки.

При создании набор состоит из двух страниц. Чтобы *добавить*, *удалить*, *переименовать* страницу или *изменить порядок* страниц, щелкните мышью на нужном ярлычке страницы и используйте соответствующую команду контекстного меню.

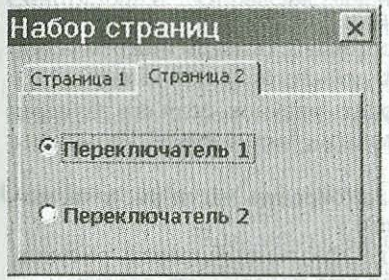

Рис 5.9. Элемент Набор страниц

Во время выполнения программы каждой странице соответствует объект Page. Все объекты Page объединены в семейство Pages.

Объект Page и семейство Pages обладают теми же свойствами и методами, что и объект Tab и семейство Tabs, описанные выше для элемента Набор вкладок.

Основные свойства элемента Набор страниц

SelectedItem, Value, MultiRow, Style, TabOrientation	Имеют тот же смысл, что и аналогичные свойства элемента Набор вкладок.
Pages	Используется для доступа к семейству Pages или конкретному объекту семейства. Аналогично свойству Tabs объекта Набор вкладок.

5.2.14. Элемент Рисунок

Элемент Рисунок (Image) предназначен для вывода содержимого графических файлов в форме. Элемент не позволяет редактировать рисунок. Поддерживаются следующие форматы файлов: *.bmp, *.cur, *.gif, *.ico, *.jpg, *.wmf.

Основные свойства элемента Рисунок (Image)

AutoSize	Если свойство имеет значение True, то при загрузке рисунка его размеры сохраняются, при этом элемент Image изменит свои размеры так, чтобы они точно соответствовали размерам рисунка.

Picture	Определяет файл, содержащий рисунок. Во время выполнения программы следует использовать функцию LoadPicture:
	`Image1.Picture = LoadPicture("ИмяФайла")`
	Чтобы удалить рисунок, выполните
	`Image1.Picture = ""`

PictureAlignment	Определяет расположение рисунка внутри объекта Image. Возможны значения: `FmPictureAlignmentTopLeft` (в левом верхнем углу); `FmPictureAlignmentTopRight` (в правом верхнем углу); `FmPictureAlignmentCenter` (в центре); `FmPictureAlignmentBottomLeft` (в левом нижнем углу); `FmPictureAlignmentBottomRight` (в правом нижнем углу).
PictureSizeMode	Определяет, как выводить рисунок, если его размеры не соответствуют размерам объекта Image. Возможны значения: `fmPictureSizeModeClip` (обрезать части рисунка, которые не помещаются); `fmPictureSizeModeStretch` (масштабировать рисунок так, чтобы он занимал всю поверхность объекта); `fmPictureSizeModeZoom` (изменить размеры рисунка, сохраняя пропорции, так чтобы он по высоте или по ширине точно соответствовал размерам объекта).
PictureTiling	Если значение свойства равно True, то рисунок покрывает поверхность объекта Image в виде мозаики. Если значение свойства равно False, то рисунок выводится в единственном экземпляре.

На рисунке 5.10 показано, как выглядит одна и та же картинка при различных значениях свойства PictureSizeMode.

Рис 5.10. Использование свойства PictureSizeMode в элементе Рисунок

5.2.15. Элемент Счетчик

Элемент Счетчик позволяет уменьшать и увеличивать числовые значения. При щелчке на соответствующей стрелке изменяется значение самого элемента, но внешне это никак не выражается. Чтобы с помощью элемента Счетчик изменять значение другого элемента, нужно написать программу.

Основные свойства элемента Счетчик

Max	Максимальное возможное значение (целое).
Min	Минимальное возможное значение (целое).
SmallChange	Шаг изменения значения (целое) при щелчке на одной из стрелок. По умолчанию равен 1.
Value	Текущее значение элемента.
ControlSource	Определяет ячейку рабочего листа, которая связывается со значением элемента. Если изменяется значение свойства Value, то автоматически изменяется и значение этой ячейки, и наоборот.
Orientation	Определяет ориентацию элемента (вертикальную или горизонтальную).

Основные события элемента Счетчик

Change	Возникает, когда пользователь нажимает на одну из кнопок элемента.
SpinDown	Возникает, когда пользователь нажимает на кнопку со стрелкой вниз или влево.
SpinUp	Возникает, когда пользователь нажимает на кнопку со стрелкой вверх или вправо.

5.2.16. Элемент Полоса прокрутки

Элемент Полоса прокрутки (ScrollBar) позволяет выбирать значение из заданного диапазона с помощью мыши (изменяя положение бегунка).

Основные свойства элемента Полоса прокрутки

Min	Устанавливает наименьшее значение для элемента, отвечающее крайнему левому или крайнему верхнему положению в соответствующей полосе прокрутки. Диапазон рекомендуемых значений от −32767 до 32767.
Max	Устанавливает наибольшее значение для элемента, отвечающее крайнему правому или крайнему нижнему положению в соответствующей полосе прокрутки. Диапазон рекомендуемых значений от −32767 до 32767.
Value	Значение из диапазона от Min до Max, которое соответствует положению бегунка полосы прокрутки.

LargeChange	Устанавливает величину изменения свойства Value после щелчка на полосе прокрутки между бегунком и кнопкой со стрелкой (целое от 1 до 32767)
SmallChange	Устанавливает величину изменения свойства Value после щелчка на кнопке со стрелкой (целое от 1 до 32767)
ControlSource	Определяет ячейку рабочего листа, которая связывается со значением элемента. Если изменяется значение свойства Value, то автоматически изменяется и значение этой ячейки, и наоборот.
Orientation	Определяет ориентацию элемента (вертикальную или горизонтальную).

События элемента Полоса прокрутки

Change	Возникает, когда бегунок занял новое положение, или при щелчке на кнопке со стрелкой, или при изменении значения свойства Value в программе.
Scroll	Возникает при движении бегунка (непрерывно).

5.2.17. Элемент RefEdit

Элемент RefEdit (редактирование ссылок) предназначен для ввода и редактирования текста, содержащего ссылки на ячейки или диапазоны ячеек рабочего листа, например, формул. Вы можете вставить ссылку в текущей позиции курсора с помощью мыши так же, как это можно делать, например, в мастере функций Excel или в строке формул.

Многие свойства элемента RefEdit имеют такой же смысл, как у элемента Поле. Основным свойством элемента RefEdit является свойство Text, содержащее редактируемую строку.

5.3. Размещение элементов управления в форме

Элемент управления может быть добавлен в форму следующими способами:

- щелкнуть на выбранном элементе панели элементов, переместить указатель мыши в форму и щелкнуть клавишей мыши;
- поместить курсор мыши на выбранный элемент, нажать клавишу мыши и, не отпуская ее, перетащить элемент в форму, отпустить клавишу.

После того, как элементы управления внесены в форму, они становятся объектами или программируемыми элементами пользовательского интерфейса.

Процесс выравнивания элементов управления в форме можно облегчить используя команды меню Формат. Для этого следует выполнить следующие действия:

- щелчком мыши маркировать первый элемент, по которому будут устанавливаться размеры других элементов управления;
- удерживая клавишу <Shift>, маркировать все другие выравниваемые элементы;
- выполнить команду меню Формат, и далее выбрать соответствующий выполняемому действию пункт меню.

Упражнение 3

Создание и редактирование формы пользователя

1. Запустите редактор Visual Basic.
2. Добавьте в проект форму, выполнив команду меню Вставка ➤ UserForm. Установите для нее следующие свойства: Name — МояФорма; Caption — Упражнение 3.
3. Поместите в поле формы две кнопки. Установите для одной из них значение свойства Caption «OK», для другой — «Закрыть» (рис. 5.11).

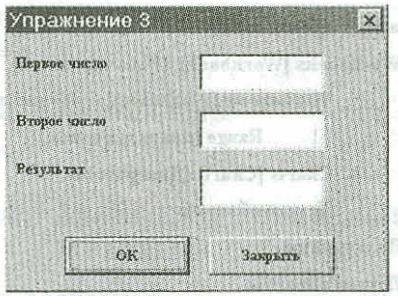

Рис. 5.11. Созданная форма пользователя

4. Поместите в форму три элемента Поле и два элемента Надпись. Для элементов Надпись свойствам Caption установите значения: Первое число; Второе число; Результат.
5. Удерживая клавишу <Shift> маркируйте все элементы управления. Для свойства Font всех элементов установите значения: Шрифт — Times New Roman; начертание — полужирный; размер — 11.
6. Маркируйте элементы Надпись. Выполните команду меню Формат ➤ Выровнять размер ➤ По ширине и высоте.
7. Поступая таким же образом, выровняйте размеры элементов Поле, а затем элементов Кнопка.

8. Маркируйте элементы Надпись. Выполните команду меню Формат ➤ Выровнять ➤ По правому краю.
9. Поступая таким же образом, выровняйте по правому краю элементы Поле, а затем элементы Кнопка.
10. Выполните команду меню Запуск ➤ Запуск подпрограммы/UserForm. На экране появится форма (рис. 5.11).

5.4. Основные объекты приложения MS Excel

Объектная модель Microsoft Excel насчитывает множество объектов. В рамках настоящего пособия не ставится цель изучить их все. Для разработки большинства программ достаточно знать основные из них, такие как Application, Workbook(s), Worksheet(s), Range, Chart.

Различают объекты и их семейства. Семейство (объект Collection) представляет собой объект, содержащий в себе несколько других объектов одного типа. Все объекты Excel и их семейства имеют родовые иерархические отношения. Иерархия основных объектов MS Excel приведена на рисунке 5.12.

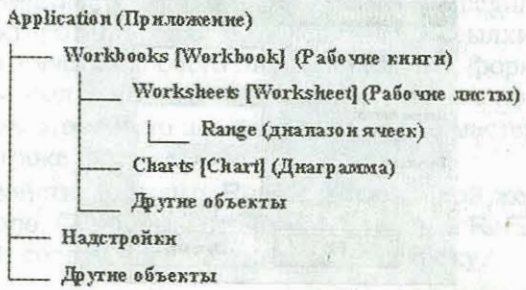

Рис. 5.12. Иерархия встроенных объектов MS Excel

Главным в иерархии объектов Excel является объект Application (Приложение), которое представляет само приложение Excel. Этот объект имеет более 120 свойств и 40 методов, которые предназначены для установки параметров приложения Excel. Кроме того, объект Application позволяет вызывать более 400 встроенных функций рабочего листа при помощи конструкции вида:

```
Application.ФункцияРабочегоЛиста(Аргументы)
```

Например:

```
Application.Pi() 'Вычисление числа Пи.
Application.Sum(Аргументы)
```


Подчиненными объектами в иерархии объектов являются: объекты семейств WorkBooks (Рабочие книги), Worksheets (Рабочие листы), Range (Диапазон).

Как было отмечено в п. 1.3, существует следующий формат программного кода, задающего установку свойства и использование метода объекта:

```
Объект. Свойство = ЗначениеСвойства
Объект. Метод [Параметр1 [.....]]
```

Если X является свойством — участником объекта Application, то обращение к этому свойству возвращает ссылку на объект X. Обращение `Application.X.Y.Z`, где X, Y и Z — свойства-участники, позволяет добраться до объекта Z, находящегося на третьем уровне вложенности. Обычно цепочка именования начинается спецификатором (объектом) `Application`, но иногда его можно опустить. Некоторые свойства и методы объекта Apiplication относятся к глобальным. Для них спецификатор `Application` разрешается опускать, непосредственно именуя глобальный элемент. Вот пример нескольких обращений к элементам объекта Application:

```
Application. ActiveDocument
```

Можно короче:

```
ActiveDocument
ActiveSheet
```

Можно указать и полный путь:

```
Application. ActiveSheet
```

5.5. Объект Application, основные свойства, методы и события

Свойства объекта Application

Свойства	Описание и допустимые значения
ActiveWorkbook, ActiveSheet, ActiveCell, ActiveChart	Возвращают активный объект: рабочую книгу, лист, ячейку, диаграмму. Свойство ActiveCell содержится в ActiveSheet, а свойства ActiveSheet и ActiveChart в ActiveWorkbook. Например: `ActiveCell.Value = "Новое значение"` присваивает значение активной ячейке.
ThisWorkbook	Возвращает рабочую книгу, содержащую выполняющийся в данный момент макрос.

Calculation	Устанавливает режим вычислений. Возможные значения: `xlCalculationAutomatic` (автоматический режим), `xlCalculationManual` (вручную).
Caption	Возвращает текст в строке имени активного окна.
DisplayFormulaBar	`True` (`False`) — строка формул выводится (не выводится) на экран.
DisplayScrolBars	`True` (`False`) — полосы прокрутки видны (не видны) в окне Excel.
ScreenUpdating	Если равно `True`, то изображение на экране обновляется во время выполнения программы, если `False` — то нет.
DisplayStatusBar	`True` (`False`) — строка состояний видна (не видна) в окне Excel.

Основные методы объекта Application

Методы	Действия
Calculate	Вызывает принудительные вычисления во всех открытых рабочих книгах.
Run	Запускает на выполнение подпрограмму или макрос: `Run (ИмяМакроса, Аргументы)`.
Volatile	Вызывает перевычисление функции пользователя при изменении значений параметров. Оператор `Application.Volatile` нужно поместить в теле функции.
Wait	Временно приостанавливает работу приложения: `Wait (Time)`.
OnTime	Назначает выполнение процедуры на определенное время: `OnTime(ВремяЗапуска, ИмяПроцедуры, …)`.
Quit	Закрывает приложение.

События объекта Application

Событие	Когда происходит
NewWorkBook	При создании новой рабочей книги.
WorkbookActivate	При активизации рабочей книги.
WorkbookBeforeClose	Перед закрытием рабочей книги.
WorkbookBeforeSave	Перед сохранением рабочей книги.
WorkbookDeactivate	Когда рабочая книга теряет фокус.
WorkbookNewSheet	При добавлении нового листа в рабочую книгу.
WorkbookOpen	При открытии рабочей книги.

5.6. Основные свойства, методы и события семейства WorkBooks

Объект Workbook — это файл рабочей книги. Получить объект Workbook можно используя свойства Workbooks, ActiveWorkbook или ThisWorkbook объекта Application.

Основные свойства объектов семейства Workbooks

Свойства	Описание и допустимые значения
ActiveSheet	Возвращает активный лист книги.
ActiveChart	Возвращает активную диаграмму.
Count	Возвращает количество объектов семейства.
WriteReserved	`True` (`False`) — документ закрыт (открыт) для записи.
Sheets, Worksheets, Charts	Возвращают семейства всех рабочих листов книги и всех диаграмм соответственно.

Основные методы объектов семейства Workbooks

Методы	Действия
Activate	Активизирует рабочую книгу (первый лист становится активным).
Add	Создает новую рабочую книгу.

Close, Open, OpenText	Закрытие (открытие) рабочей книги, открытие текстового файла с таблицей данных. Например, рабочая книга закрывается без сохранения:
	`Workbooks("Book1.xls").Close SaveChanges:=False`
	Открытие рабочей книги:
	`Workbooks.Open "Book1.xls"`

Save, SaveAs	Сохранение рабочей книги (сохранение в другом файле). Например, запросить у пользователя имя файла и сохранить активную рабочую книгу можно кодом программы:
	`fName= Application.GetSaveAsFilename`
	`ActiveWorkbook.SaveAs Filename:=fName`

События объектов семейства Workbooks

Событие	Когда происходит
BeforeClose	Перед закрытием рабочей книги.
BeforeSave	Перед сохранением рабочей книги.
Deactivate	Когда рабочая книга теряет фокус.

NewSheet	При добавлении нового листа в рабочую книгу.
Open	При открытии рабочей книги.
SheetActivate	При активизации рабочего листа.
SheetDeactivate	Когда рабочий лист теряет фокус.

Упражнение 4

Свойства и методы объектов Application, Workbooks

1. На рабочем листе с именем Лист1 поместите кнопку формы.
2. Назначьте для этой кнопки макрос с именем Кнопка1_Щелкнуть.
3. В окне редактирования кода редактора Visual Basic запишите следующий программный код:

```
Option Explicit
Sub Кнопка1_Щелкнуть()
    'Изменение надписи в строке заголовка приложения
    Application.Caption = "Упражнение №4 – Свойства, методы и_
      события"
    MsgBox "Обратите внимание! Изменилась надпись в строке _
      заголовка приложения", vbInformation
    'Убираем строку формул
    Application.DisplayFormulaBar = False
    MsgBox "Обратите внимание! Строка формул не отображается",_
      vbInformation
    'Убираем панель состояний
    Application.DisplayStatusBar = False
    MsgBox "Обратите внимание! Панель состояний не _
      отображается", vbInformation
    Application.DisplayStatusBar = True
    MsgBox "Обратите внимание! Панель состояний вновь _
      отображается", vbInformation
    'Изменение размера окна приложения
    Application.WindowState = xlNormal
    MsgBox "Обратите внимание! Размер окна приложения _
      стал нормальным", vbInformation
    Application.WindowState = xlMinimized
    MsgBox "Обратите внимание! Размер окна приложения _
      стал минимальным" , vbInformation
    Application.WindowState = xlMaximized
    MsgBox "Обратите внимание! Размер окна приложения _
      стал вновь максимальным", vbInformation
    'Создание новой рабочей книги
```



```
    Workbooks.Add
    MsgBox "Обратите внимание! Создана новая рабочая книга", _
      vbInformation
    'Активизация созданной рабочей книги
    Workbooks("Книга2").Activate
End Sub
```

4. Прочитайте все команды программы и попытайтесь понять их назначение и синтаксис записи. Обратите внимание на текст комментариев.
5. Запустите макрос на выполнение.
6. Проследите за тем, какие действия выполняет программа.
7. Сопоставьте команды программы и выполняемые ею действия.
8. Изучите правила записи операторов кода.

5.7. Основные свойства и методы объектов семейства WorkSheets

Объект Worksheet представляет собой рабочий лист. Объект Worksheet можно получить, используя свойства ActiveSheet или Worksheets объекта Workbook.

Свойства объектов семейства WorkSheets

Свойства	Описание и допустимые значения
Name	Возвращает имя рабочего листа: `Worcsheets(1).Name="Итоги"`
Visible	`True` (`False`) — рабочий лист видим (невидим) на экране.
Range	Возвращает ссылку на указанный диапазон ячеек. Например: `ActiveSheet.Range("B1")`
UsedRange	Возвращает диапазон ячеек рабочего листа.
ActiveCell	Возвращает активную ячейку рабочего листа.

Методы объектов семейства Worksheets

Методы	Выполняемые действия
Activate	Активизирует рабочий лист: `Worksheet(2).Activate`
Add	Создает новый рабочий лист. Параметры: `Before` — лист, перед которым будет размещен новый лист; `After` — лист после которого будет помещен новый лист; `Count` — число добавляемых листов; `Type` — тип добавляемого листа. Например, `ActiveWorkbook.Worksheets.Add`

Delete	Удаляет рабочий лист: `Worksheets(1).Delete`

Evaluate	Преобразует текстовую строку в объект Excel или значение. Используется, например, для ввода ссылок на ячейки:
	`MyCell = InputBox("Введите имя ячейки")`
	`Evaluate(myCell).Value = "Новое значение"`

Copy	Копирование активного рабочего листа в другое место рабочей книги:
	`Worksheets("Лист2").`
	`Copy After:= Worksheets("Лист3")`

Move	Перемещение активного рабочего листа в другое место рабочей книги:
	`Worksheets("Лист2").`
	`Move After:= Worksheets("Лист3")`

События объекта Worksheet

Событие	Когда происходит
BeforeClose	Перед закрытием рабочей книги.
BeforeSave	Перед сохранением рабочей книги.
Deactivate	Когда рабочая книга теряет фокус.
NewSheet	При добавлении нового листа в рабочую книгу.
Open	При открытии рабочей книги.
SheetActivate	При активизации рабочего листа.
SheetDeactivate	Когда рабочий лист теряет фокус.

5.8. Объект Range

5.8.1. Адресация ячеек в Excel

Для ссылок на ячейки в Excel используются два формата:

Формат A1. Ссылка состоит из имени столбца (обозначаются буквами от A до IV, 256 столбцов максимально) и номера строки (от 1 до 65536). Например, `A77`. Для ссылки на диапазон ячеек указываются адреса левой верхней и правой нижней ячейки диапазона, разделенных двоеточием. Например, `B10:B20`, `7:7` (все ячейки в 7-й строке), `5:10` (все ячейки между 5-й и 10-й строками включительно), `D:D` (все ячейки в столбце `D`), `H:J` (все ячейки между столбцами `H` и `J` включительно). Признаком абсолютной ссылки является знак доллара перед именем строки или столбца.

Формат R1C1. В формате R1C1, после буквы «R» указывается номер строки ячейки, после буквы «C» — номер столбца. Например, абсолютная ссылка R1C1 эквивалентна абсолютной ссылке $A$1 для формата A1. Для задания относительной ссылки указывается смещение по отношению к активной ячейке. Смещение указывается в квадратных скобках. Знак указывает направление смещения. Например, `R[-3]C` (относительная ссылка на ячейку, расположенную на три строки выше в том же столбце). `R[2]C[2]` (относительная ссылка на ячейку, расположенную на две строки ниже и на два столбца правее). `R2C2` (абсолютная ссылка на ячейку, расположенную во второй строке и во втором столбце). `R[-1]` (относительная ссылка на строку, расположенную выше текущей ячейки), `R` (абсолютная ссылка на текущую строку).

Полный адрес ячейки может содержать также имя рабочего листа и адрес книги. После имени листа ставится знак «!», а адрес книги заключается в квадратные скобки. Например:

```
[МояКнига.xls]Лист1!D2.
```

Объект Range используется для работы с ячейками, строками, столбцами, а также их группами. Для доступа к объекту чаще всего используются свойства Range и Cells, хотя есть и другие возможности.

Если используется свойство Range, то в качестве аргумента указывается любая допустимая в Excel ссылка в формате A1. Если имя листа не указывается, то используется активный лист. Например:

```
'Ячейке A5 листа Лист1 присвоить значение 5
Worksheets("Лист1").Range("A5").Value = 5
'Ячейке A5 текущего листа присвоить значение 5
Range("A5").Value = 5
```

Свойство Cells используется для доступа к отдельной ячейке. В качестве аргументов указываются номер строки и столбца. Например, так можно присвоить значение ячейке A5 первого рабочего листа:

```
Worksheets(1).Cells(5,1).Value = 5
```

Можно также использовать свойство Cells для альтернативного указания диапазона. Например:

```
Range("A2:C3") и Range(Cells(2,1), Cells(3,3))
```

определяют один и тот же диапазон.

5.8.2. Основные свойства объекта Range

Свойства	Описание и допустимые значения
Value	Возвращает значение из ячейки или диапазона: `X=Range("A2").Value`
Name	Возвращает имя диапазона: `Range("B1:B4").Name="Итого"`
CurrentRagion	Возвращает количество строк текущего диапазона.
WrapText	`True` (`False`) — разрешает (не разрешает) перенос текста при вводе в диапазон.
EntireColumn, EntireRow	Возвращает строку и столбец.
ColumnWidth, RowHeight	Возвращает ширину столбцов и высоту строк диапазона.

Font	Возвращает объект Font (шрифт). Например:
	`With Worksheets("Л1").Range("B5").Font`
	`.Size = 14`
	`.Bold = True`
	`.Italic = True`
	`End With`

Formula	Формула в формате A1. Например, так можно ввести формулу в ячейку B5: `Range("B5").Formula = "=$A$4+$A$10"` При считывании значения, возвращается текстовая строка (как в строке формул).
FormulaLocal	Формула в формате A1 с учетом языка пользователя (для неанглоязычных версий Excel). Например: `Range("B5").FormulaLocal = "=ПИ()"`
FormulaR1C1	Формула в формате R1C1. Например, `Range("B1").FormulaR1C1 = "=R1C1+1"`
FormulaR1C1Local	Формула в формате R1C1 с учетом языка пользователя (для неанглоязычных версий Excel).
HorizontalAlignment	Горизонтальное выравнивание. Возможные значения: `xlHAlignGeneral` (обычное), `xlHAlignCenter` (по центру), `xlHAlignCenterAcrossSelection` (по центу выделения), `xlHAlignJustify` (по ширине), `xlHAlignLeft` (по левому краю), `xlHAlignRight` (по правому краю) и другие.
VerticalAlignment	Вертикальное выравнивание. Возможные значения: `xlVAlignBottom` (по нижнему краю), `xlVAlignCenter` (по центру), `xlVAlignTop` (по верхнему краю) и другие.

Методы объекта Range можно разделить на две большие группы: методы, относящиеся к самому объекту, и методы, реализующие команды. Многие из них имеют параметры, которые здесь описываются лишь частично. Подробнее о параметрах этих методов можно прочитать, например, в справочной системе Excel. Для изучения методов, реализующих команды, рекомендуется записать макрос, выполняющий нужную команду, и проанализировать полученный код.

5.8.3. Основные методы объекта Range

Методы	Действия
Adress	Возвращает адрес ячейки.
AutoFit	Автоматически настраивает ширину столбца и высоту строки. Например: `Range("B1:B3").Columns.AutoFit` Использование свойства Columns или Rows в данном случае необходимо, так как значением диапазона должны быть строки или столбцы, иначе будет выдаваться ошибка.
Clear	Очищает диапазон. Например: `Range("A1:C5").Clear`

Copy	Копирует диапазон в другой диапазон или буфер обмена (если параметр Destination не задан). Например, так можно скопировать значения с одного листа (Л1) на другой (Л2):
	`Worksheets("Л1").Range("A1:B3").Copy _`
	`Destination:=Worksheets("Л2").Range("A5")`

Cut	Копирует диапазон с удалением в другой диапазон или буфер обмена (если параметр Destination не задан). Например, скопируем диапазон ячеек с удалением в буфер обмена: `Worksheets("Лист2").Range("A1:D4").Cut`
Delete	Удаляет диапазон. Параметр Shift определяет направление сдвига ячеек. Например: `Range("A6:D6").Delete Shift:=xlShiftToLeft`
Insert	Вставляет ячейку или диапазон ячеек. Например, так можно вставить строку перед шестой строкой на листе «Лист2»: `Worksheets("Лист2").Rows(6).Insert`
Select	Выделяет диапазон: `Range("A1:C7").Select`

5.8.4. Методы объекта Range, реализующие команды Excel

Кроме методов, реализующих команды объект Range имеет методы, которые используют команды Excel.

Метод	Действия
DataSeries	Создает прогрессию. `DataSeries(rowcol,date,step,stop,trend)` Вручную метод выполняется с помощью команды Правка ➤ Заполнить ➤ Прогрессия.
AutoFill	Автозаполнение. Автоматически заполняет ячейки диапазона элементами последовательности: Объект (Диапазон, Тип).
AutoFilter	Автофильтр. Реализует запрос на фильтрацию данных на рабочем листе: `Объект.AutoFilter(Поле, Условие1, Оператор, Условие2).` Соответствует команде Данные ➤ Фильтр ➤ Автофильтр.
AdvancedFilter	Расширенный фильтр. Соответствует команде Данные ➤ Фильтр ➤ Расширенный фильтр.
Consolidate	Объединение данных из нескольких диапазонов в одну итоговую таблицу. Соответствует команде Данные ➤ Консолидация.
Find	Поиск данных. Вручную вызывается командой Правка ➤ Найти.
tblGoalSeek	Подбор параметра. Вручную выполняется с помощью команды Сервис ➤ Подбор параметра.
Sort	Сортировка данных. Вручную выполняется с помощью команды Данные ➤ Сортировка.
Subtotal	Добавляет промежуточные итоги. Вручную вызывается командой Данные ➤ Промежуточные итоги.

Упражнение 5

Свойства и методы объекта Range и Selection

1. На рабочем листе с именем Лист1 поместите кнопку формы.
2. Назначьте для этой кнопки макрос с именем Кнопка1_Щелкнуть.
3. В окне редактирования кода редактора Visual Basic запишите следующий программный код.

```
Option Explicit
Sub Кнопка1_Щелкнуть()
    'В ячейку A1 записывается текст
    Range("A1").Value = "Упражнение 5"
    'Выделяется ячейка A1
    Range("A1").Select
    With Selection
        'Получаем адрес активной ячейки
        MsgBox "Адрес активной ячейки" & .Address()
        'Получаем значение в активной ячейке
        MsgBox "Значение активной ячейки" & .Value
    End With
    'Изменяем параметра шрифта для активной ячейки
    With Selection
        .Font.Size = 16
    End With
    'В ячеки вводим числовые значения и формулу
    Range("A2").Value = 2
    Range("B2").Value = 4
    Range("C2").Formula = "=A2^B2"
    'Получаем количество строк в диапазоне
    MsgBox "Количество строк в области A1:C2 =" _
    Range("A1:C2").Rows.Count
    MsgBox "Количество строк в текущем диапазоне" & Range("A1") _
    .CurrentRegion.Rows.Count
    'Очищаем диапазон
    Range("A1:C2").Clear
    'Используем объект Cells
    Cells(1, 1) = "Упражнение 5"
End Sub
```

4. Прочитайте все команды программы и попытайтесь понять их назначение и синтаксис записи. Обратите внимание на текст комментариев.
5. Запустите макрос на выполнение.
6. Проследите за тем, какие действия выполняет программа.
7. Сопоставьте команды программы и выполняемые ей действия.

Упражнение 6

Пусть на рабочем листе имеется таблица. В левой ячейке ее первой строки находится заголовок таблицы. В следующей строке — заголовки столбцов. В остальных строках — данные. Количество строк заранее не известно. Создайте диалоговое окно, которое позволит отформатировать таблицу: разместит заголовок таблицы по центру над столбцами, изменит шрифт (размер — 14,

курсив, цвет — красный), заголовки столбцов расположит в центре, изменит шрифт на полужирный. В окне также имеется поле для ввода диапазона ячеек и кнопка, позволяющая убрать форматирование для указанных ячеек. Вид диалогового окна и результат форматирования показан на рисунках 5.13, 5.14.

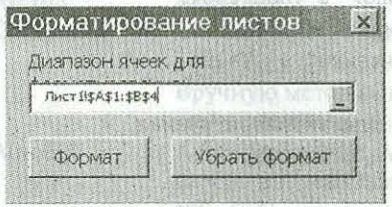

Рис. 5.13. Диалоговое окно

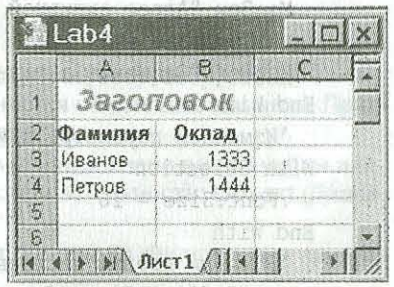

Рис. 5.14. Результат форматирования

Выполните следующие действия:

1. Создайте приведенную на рисунке 5.13 форму. На ней размещены элемент Надпись, две кнопки и элемент RefEdit. Установите необходимые свойства элементов.
2. В общей области окна программы формы декларируйте переменные:

```
Dim myR As Range
Dim Заголовок As Range
Dim Названия As Range
Dim c As Integer
Dim r As Integer
```

3. Напишите процедуры обработки события Click для кнопок. Они могут быть примерно такими:

```
Private Sub КнопкаФорматировать_Click()
    'присваиваем переменной myR значение (ссылка на
    'диапазон берется из элемента RefEdit)
    Set myR = Range(RefEdit1.Text)
    r = myR.Rows.Count 'число строк в диапазоне
    c = myR.Columns.Count 'число столбцов в диапазоне
    Set Заголовок = Range(myR.Cells(1,1), myR.Cells(1,c))
    Set Названия = Range(myR.Cells(2,1), myR.Cells(2,c))
    Заголовок.Select
    Selection.HorizontalAlignment = xlCenterAcrossSelection
    With Заголовок.Font
```



```
        .Name = "Arial Cyr"
        .FontStyle = "полужирный курсив"
        .Size = 14
        .ColorIndex = 3
    End With
    Названия.HorizontalAlignment = xlCenter
    With Названия.Font
        .Name = "Arial Cyr"
        .FontStyle = "полужирный"
        .Size = 10
    End With
End Sub

Private Sub КнопкаУбратьФормат_Click()
    Set myR = Range(RefEdit1.Text)
    myR.ClearFormats
End Sub
```

4. Проверьте работу программы. Для вызова окна используйте кнопку, размещенную на рабочем листе.

6. Объекты для программирования меню и панелей инструментов

6.1. Иерархия объектов

Для программирования строк меню и панелей инструментов (будем называть их панелями команд) используется семейство CommandBars, состоящее из объектов CommandBar. В семействе CommandBars хранятся все панели команд конкретного приложения. Это семейство содержится в объекте Application. Для доступа к конкретному объекту CommandBar нужно указать его имя или индекс. Например:

```
Application.CommandBars("Standard")
```

Каждый объект CommandBar содержит семейство CommandBarControls, состоящее из всех элементов управления данной панели команд. Для доступа к этому семейству используется свойство Controls объекта CommandBar. Для доступа к конкретному элементу дополнительно нужно указать номер или название элемента. Например:

```
CommandBars("Standard").Controls(1)
```

Элементы семейства CommandBarControls относятся к одному из трех типов:

CommandBarButton (кнопка или элемент меню, вызывающий выполнение команды или подпрограммы);

CommandBarComboBox (сложные элементы меню: поле ввода, раскрывающийся список или поле со списком);

CommandBarPopup (подменю или контекстное меню).

Иерархия объектов приведена на рисунке 6.1.

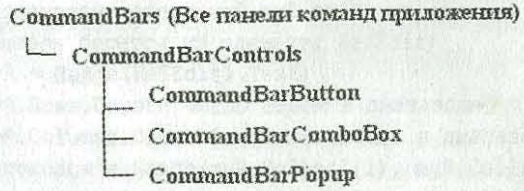

Рис. 6.1. Иерархия объектов для программирования меню

6.2. Объект CommandBar

Основные свойства объекта CommandBar

Enabled — Определяет возможность доступа (`True` — панель команд доступна, `False` — нет).

Visible — Определяет, видим ли объект (`True` — видим, `False` — нет). Например, так можно скрыть панель инструментов Форматирование:

```
Application.CommandBars("Formatting").Visible = False
```

Controls — Возвращает семейство CommandBarControls, состоящее из всех элементов управления конкретного меню или панели инструментов.

Position — Определяет расположение объекта. Возможные значения: `msoBarLeft`, `msoBarTop`, `msoBarRight`, `msoBarBottom`, `msoBarFloating` (для панели инструментов), `msoBarPopup` (для контекстных меню и подменю).

Основные методы объекта CommandBar

Add — Создает новую панель команд и добавляет ее в семейство CommandBars. Имеет параметры:
`Name` — имя создаваемой панели команд;
`Position` — местоположение панели команд (`msoBarLeft`, `msoBarTop`, `msoBarRight`, `msoBarBottom`, `msoBarFloating`, `msoBarPopup`);
`MenuBar` — определяет, нужно ли заменять активную панель команд (`True` — заменять, `False` — нет);
`Temporary` — определяет, нужно ли удалять панель команд при закрытии приложения (`True` — удалять, `False` — нет).

Delete — Удаляет панель команд.

Reset — Восстанавливает встроенную панель команд в исходное состояние (по умолчанию).

ShowPopUp — Выводит на экран контекстную панель команд. Если координаты заданы, то панель размещается в указанном месте, в противном случае — в месте расположения указателя.

6.3. Объект CommandBarControls

Основные свойства объекта CommandBarControl

Caption — Текстовая строка, отображаемая в заголовке.

Enabled — Определяет возможность доступа (`True` — элемент доступен, `False` — нет).

OnAction — Имя процедуры, выполняемой при активизации элемента.

ShortcutText Комбинация клавиш для выбора пункта меню.

Style Определяет внешний вид объекта. Для кнопки может иметь следующие значения:
`MsoButtonAutomatic` — не содержит текста и рисунка;
`MsoButtonIcon` — содержит рисунок;
`MsoButtonCaption` — содержит текст (нужно задать свойство `Caption`);
`MsoButtonIconandCaption` — содержит текст и рисунок.

Visible Определяет, видим ли объект (`True` — видим, `False` — нет).

Основные методы семейства CommandBarControls

Add Добавляет новый элемент в панель команд. Возвращает объект `CommandBarButton`, `CommandBarComboBox` или `CommandBarPopup`. Имеет параметры:
`Type` — определяет тип добавляемого элемента (`msoControlButton` — кнопка или элемент меню, `msoControlEdit` — поле ввода, `msoControlDropdown` — раскрывающийся список, `msoControlComboBox` — поле со списком, `msoControlPopup` — подменю и другие).
Описание типов можно найти в описании свойства Type объекта CommandBarControl;
`Id` — целое число, которое определяет встроенный элемент управления. Если значение больше 1, то добавляемый элемент наследует внешний вид и действия, выполняемые этим встроенным элементом;
`Parameter` — параметр, используемый встроенным элементом управления;
`Before` — индекс или имя элемента управления, перед которым добавляется новый элемент. Если аргумент опущен, то элемент вставляется в конец панели элементов;
`Temporary` — определяет, нужно ли удалять новый элемент при закрытии приложения! (`True` — удалять, `False` — нет).

Упражнение 7

Создайте меню как на рисунке 6.2. Каждый пункт меню связан с процедурой, которая выполняется при выборе этого пункта меню. Меню создается при открытии рабочей книги и удаляется при закрытии приложения.

Для создания меню выполните следующие действия:

1. В окне проекта редактора VBA выберите лист ThisWorkbook и введите на нем следующую процедуру:

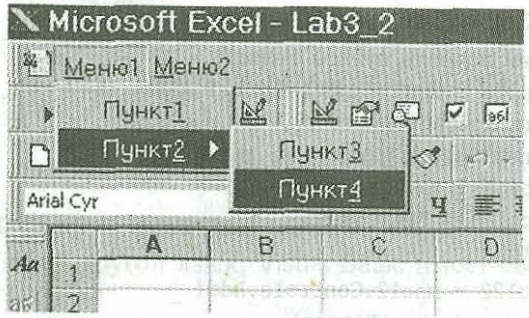

Рис. 6.2. Меню

```
Private Sub Workbook_WindowActivate( _
    ByVal Wn As Excel.Window)
  Dim myMnu As CommandBar
  Dim mnu1 As CommandBarPopup
  Dim mnu11 As CommandBarButton
  Dim mnu12 As CommandBarPopup
  Dim mnu121 As CommandBarButton
  Dim mnu122 As CommandBarButton
  Dim mnu2 As CommandBarButton
  'Создаем свою строку меню
  Set myMnu = Application.CommandBars.Add( _
    Name:="МоеМеню", MenuBar:=True, _
    Temporary:=True, Position:=msoBarTop)
  myMnu.Visible = True
  'Создаем меню Меню1
  Set mnu1 = myMnu.Controls.Add( _
    Type:=msoControlPopup)
  With mnu1
    .Caption = "&Меню1"
  End With
  'Создаем пункт Меню11
  Set mnu11 = mnu1.Controls.Add(Type:=msoControlButton)
  With mnu11
    .Caption = "Пункт&1"
    .OnAction = "Действие1"
  End With
  'Создаем пункт Меню12
  Set mnu12 = mnu1.Controls.Add(Type:=msoControlPopup)
```

```
    mnu12.Caption = "Пункт&2"
    'Создаем подменю пункта Меню12
    Set mnu121 = mnu12.Controls.Add( _
      Type:=msoControlButton)
    With mnu121
      .Caption = "Пункт&3"
      .OnAction = "Действие2"
    End With
    Set mnu122 = mnu12.Controls.Add( _
      Type:=msoControlButton)
    With mnu122
      .Caption = "Пункт&4"
      .OnAction = "Действие3"
    End With
    'Создаем меню Меню2
    Set mnu2 = myMnu.Controls.Add(Type:=msoControlButton)
    With mnu2
      .Style = msoButtonCaption
      .Caption = "&Меню2"
      .OnAction = "Действие4"
    End With
  End Sub
```

2. На листе модуля создайте следующие процедуры, которые будут выполняться при выборе пункта меню:

```
Public Sub Действие1()
  MsgBox "Выбран первый пункт Меню1"
End Sub
Public Sub Действие2()
  MsgBox "Выбран первый пункт подменю Меню1"
End Sub
Public Sub Действие3()
  MsgBox "Выбран второй пункт подменю Меню1"
End Sub
Public Sub Действие4()
  MsgBox "Выбрано Меню2"
End Sub
```

3. Сохраните рабочую книгу и закройте ее.
4. Вновь откройте рабочую книгу и проверьте результаты своей работы.

7. Язык программирования VBA

7.1. Данные и их описание

7.1.1. Алфавит и лексемы языка

Как и любой другой язык, Visual Basic имеет свой алфавит. В него входят:

- прописные и строчные буквы латинского алфавита (A — Z, a — z);
- прописные и строчные буквы кириллицы (А — Я, а — я);
- цифры от 0 до 9;
- символ подчеркивания «_».

Из этих символов конструируются *идентификаторы* — имена переменных, констант, процедур, функций, меток переходов, а также имена типов. Кроме этих символов в состав алфавита также входят:

- неизображаемые символы («обобщенные пробельные символы»), используемые для отделения лексем друг от друга (пробел, табуляция, переход на новую строку);
- специальные символы, участвующие в построении конструкций языка: + — * / \ ^ = > < [ ] ( ) . , : ; { } ‘ & @;
- составные символы, воспринимаемые как один символ: <= >= <>.

Разделители в составных символах недопустимы.

Программный код Visual Basic представляет собой последовательность лексических единиц (лексем), записанных в соответствии с принятыми синтаксическими правилами, которая реализует некоторую семантическую конструкцию. Для обеспечения читаемости и понятности в тексте программы помещаются комментарии. В Visual Basic определен однострочный комментарий. Комментарий представляет собой последовательность любых символов, размещаемых на одной строке исходного текста программы, которая начинается со знака « ‘ » (апостроф) или с ключевого слова Rem:

```
' Это комментарий
Rem Это тоже комментарий
```

Лексема — это единица текста программы, которая имеет определенный смысл для компилятора и которая не может быть

разбита в дальнейшем. В Visual Basic различают шесть классов лексем: свободно выбираемые и используемые идентификаторы; служебные (зарезервированные) слова; константы; строки (строковые константы); операции (знаки операций); разделители (знаки пунктуации).

Посредством идентификаторов обозначают имена переменных, констант, процедур и функций. Идентификатор представляет собой последовательность букв, цифр и символов подчеркивания. Выбирая идентификатор для имени, следует учитывать два обстоятельства. Во-первых, имя должно быть содержательным, т.е. отражать назначение переменной, что делает программу более читабельной. Во-вторых, Visual Basic накладывает на имена следующие ограничения:

- имя должно начинаться с буквы;
- имя не должно содержать *точки*, *пробела*, *разделительных символов*, *знаков операций*, а также *специальных символов*;
- имя должно быть уникальным, оно не должно совпадать с зарезервированными словами Visual Basic или с другими именами;
- длина имени может включать до 255 символов, но следует иметь в виду, что Visual Basic учитывает только первые 31 символов от начала имени.

Примеры правильных имен: `strMyName`, `i`, `intNumOne`, `StrInputValue`, `intNumber2`, `strФамилия`, `Номер`, `Адрес_Организации` и т.п.

Примеры неправильных имен: `2Week` — имя начинается с цифры; `_Номер` — имя начинается со знака подчеркивания; `Second.Week` — в имени есть точка; `Dim`, `As`, `Private` — эти слова являются зарезервированными; `Number One` — в имени есть пробел.

В Visual Basic, как и в других языках программирования, есть зарезервированные (ключевые) слова, которые нельзя выбирать в качестве идентификаторов имен. Перечень зарезервированных слов приведен в таблице 7.1.

Имена могут быть простыми или составными. Имена следует выбирать такими, чтобы они несли в себе больше информации о назначении переменной, процедуры или функции.

Примеры простых и составных имен переменных: `N`, `I`, `Number`, `Номер` — простые имена переменных; `ФамилияСтудента` — составное имя переменной, оно несет в себе информацию о фамилии студента.

Таблица 7.1.

Зарезервированные слова Visual Basic

All	Else	Is	ON	SELECT
As	Empty	JOIN	On	Set
ASC	Error	Len	Option	Static
Binary	False	Let	Optional	Step
BY	For	Lock	ParamArray	String
ByRef	Friend	Me	Print	TABLE
ByVal	Get	Mid	Private	Then
CREATE	IN	New	Property	Time
Date	INDEX	Next	Public	To
DESC	Input	Nothing	Resume	WITH
DROP	INTO	Null	Seek	WithEvents

Существуют соглашения по стилю имен, которых желательно придерживаться:

- идентификатор должен понятным образом отражать назначение переменной (это правило способствует пониманию программы);
- лучше использовать строчные имена, в случае составных названий нужно отделять друг от друга составляющие их слова подчеркиванием или начинать новое слово с прописной буквы;
- имена из прописных букв используются для определения констант.

7.1.2. Объявление переменных

Переменные — это объекты, предназначенные для хранения данных. В различные моменты времени переменные могут хранить различные значения. В переменных можно запоминать какие-либо значения и извлекать их из них. Переменную можно представить как простейший объект программы следующим образом:

Имя переменной связывает переменную с некоторой областью памяти. Имена переменных позволяют различать их в программе, осуществлять доступ к различным участкам памяти для записи данных и их извлечения.

Перед использованием переменных в программе их нужно объявлять (декларировать). При объявлении переменной необходимо указать, что объявляется переменная, задать имя переменной и указать ее тип. Тип определяет способ представления пере-

менной. В переменных можно хранить практически любые типы данных: число, строку текста, экземпляр объекта, элементы управления и т.д. В Visual Basic различают две группы типов данных: основные (иногда их называют базовыми или встроенными) и типы данных, определяемые пользователем.

Для эффективного использования памяти необходимо правильно выбирать тип переменной. В таблице 7.2 приведены базовые типы переменных Visual Basic, необходимая для их размещения память и диапазон возможных значений.

Таблица 7.2.

Базовые типы переменных Visual Basic

Тип	Хранимая информация	Занимаемая память	Интервалы значений
Целочисленные типы			
Byte	Целые числа	1 байт	от 0 до 255
Boolean	Логические значения	2 байта	True (Истина) или False (Ложь)
Integer	Целые числа	2 байта	от −32768 до 32767
Long Integer	Длинные целые числа	4 байта	+/−2.1E9
Типы с плавающей точкой			
Single	Вещественные числа одинарной точности с плавающей точкой	4 байта	от −3.402823E38 до −1.401298E−45 для отрицательных чисел и от 1.401298E−45 до 3.402823E38 для положительных
Double	Вещественные числа двойной точности с плавающей точкой	8 байт	от −1.7976313486232E308 до −4.94065645841247E−324 для отрицательных чисел и от 4.94065645841247E−324 до 1.7976313486232E308 для положительных
Строковые типы			
String (строка фиксированной длины)	Текстовая информация (строка)	1 байт на каждый символ	От 1 до 65400

Тип	Хранимая информация	Занимаемая память	Интервалы значений
String (строка переменной длины)	Текстовая информация (строка)	10 байт + 1 байт на каждый символ	От 0 до двух миллиардов символов
Объектные типы			
Object	Рисунок или ссылка на любой другой объект	4 байта	Ссылка на объект
Типы Variant			
Variant	Значения любого из перечисленных типов данных	16 байт для чисел, 22 байта + 1 байт на каждый символ для строк	Любое числовое или строковое значение
Прочие типы			
Currency	Числа, имеющие до 15 цифр до десятичной точки и 4 цифры после нее (денежные единицы)	8 байт	от −922337203685477.5808 до 922337203685477.5808
Date	Информация о дате и времени	8 байт	от 1 января 1000 г. до 31-го декабря 9999 г.
Decimal	Десятичное число	14 байт	Целое — 29 знаков Вещественное — 27 знаков после запятой

Декларация переменных может быть явной или неявной. Для явного определения переменных существует два способа. Первый (предпочтительный) способ, предполагает использование следующего синтаксиса:

```
[Static [Private | Public] Dim ИмяПеременной1[ As Тип1]
```

где `Dim` (`Размер`) — ключевое слово, которое сообщает Visual Basic, что декларируется переменная и резервируется область памяти для ее хранения;

ИмяПеременной — имя переменной (идентификатор, не входящий в перечень ключевых слов Visual Basic);

As (Как) — ключевое слово, которое сообщает Visual Basic, что определяется тип данных для переменной;

Тип — тип данных для объявляемой переменной;

Private (Частный), Public (Общий) — ключевые слова, определяющие область видимости переменной;

Static (Статический) — ключевое слово, которое определяет, сохраняет ли переменная свое значение при завершении блока программы (процедуры, функции) и выходе из него.

При подготовке кода программы среда программирования оказывает помощь пользователю: после набора ключевого слова As раскрывается список, в котором наряду с другими типами объектов, указаны и базовые типы переменных. Тип переменной можно установить, дважды щелкнув на имени типа в этом списке или нажав клавишу <Tab>.

Другим способом явного объявления переменных является указание типа с помощью суффикса. В этом случае тип данных переменной определяется с помощью добавления в конец ее имени специального символа описания типа — суффикса. Поэтому использование ключевого слова As не требуется.

Явное объявление переменных с помощью суффикса имеет следующий синтаксис:

```
[Static | Public | Private] Dim ИмяПеременнойСуффикс
```

Например:

Dim strInputMsg$ — объявляется переменная типа «строка» (String);

Static sngCalcAverage! — объявляется переменная типа Singl;

Private intNumVal% — объявляется переменная типа «целое» (Integer).

В таблице 7.3 приведены типы переменных и соответствующие им суффиксы, применяемые при декларации типов.

Таблица 7.3.

Суффиксы, определяющие тип переменной

Тип переменной	Суффикс
Integer	%
Long	&
Single	!
Double	#
Currency	@

Окончание табл. 7.3.

Тип переменной	Суффикс
String	$
Byte	Нет
Boolean	Нет
Date	Нет
Object	Нет
Variant	Нет

Неявное объявление переменных осуществляется также двумя способами. Первый способ заключается в использовании оператора DefТип_Данных. Этот оператор устанавливает тип данных для переменных, параметров процедур, и тип возвращаемого значения для процедур типа Function и Property Get, имена которых начинаются с определенных символов. Синтаксис оператора имеет вид:

```
DefТип_Данных ДиапазонБукв [,ДиапазонБукв]…
```

где Тип_Данных — сокращенное название типа данных; ДиапазонБукв — указывает границы диапазона имен, для которых задается тип данных по умолчанию.

Например: 1) DefInt A-K — переменные, имена которых начинаются с символов A — K, будут иметь тип Integer; 2) DefStr L-Z — переменные, имена которых начинаются с символов L — Z, будут иметь строковый тип.

При втором способе неявного объявления переменная декларируется просто указанием ее имени, например MyVal или MyName$, в тексте программы.

Следует отметить, что хороший стиль программирования предполагает использование явной декларации с помощью ключевых слов Dim, Private, Public, Static. Неявное объявление переменных без необходимости применять не следует, так как в последующем могут возникнуть непредвиденные ошибки. Для того, чтобы избежать неприятностей в случае ошибочной записи имени переменной, необходимо в общей области программного модуля помещать оператор Option Explicit. В этом случае Visual Basic будет расценивать любую неявно объявленную переменную как ошибочную, например:

```
Option Explicit
Dim intMyNumber As Integer 'Явное объявление переменной
DefInt i 'Неявное объявление переменной
intMyNumber=10 'Для явно декларированной переменной ошибки _
    не будет
```

```
intMyNum=10 'При ошибочном указании имени
  'Option Explicit включит предупреждение об ошибке
intNamber=6 'Для неявно декларированной
  'переменной будет выдано сообщение об ошибке
```

Примеры декларации переменных:

```
Dim x As Integer, strName$, lngOld&
Dim y As Integer
```

7.1.3. Строковые переменные

Различают строки переменной и фиксированной длины. Строки переменной длины могут содержать до двух миллиардов символов. Когда такой переменной присваивается значение, то размер переменной изменяется так, чтобы он соответствовал длине присвоенного строкового значения.

Строка фиксированной длины — это строка постоянного размера, указанного при объявлении переменной. Если такой строке присваивается значение более длинное, то лишние символы отбрасываются. Если значение, которое присваивается, короче, то остающееся место заполняется пробелами. Строковые переменные фиксированной длины должны декларироваться явно. Синтаксис декларации следующий:

```
Dim VarName As String * ДлинаСтроки
```

где ДлинаСтроки — целочисленная переменная или константа, содержащая число, которое указывает длину строковой переменной.

Например:

```
Dim strMyName As String* 20 'Объявляется строковая _
  переменная фиксированной длины в 20 символов
```

или

```
Option Explicit
Dim intLen As Integer 'Объявляется целочисленная переменная
intLen = 30 'Инициализация значения целочисленной переменной
Dim MyName As String * intLen 'Объявляется строковая _
  переменная длиной в 30 символов
MyName = "Петров"
```

7.1.4. Константы

Константы — это объекты, значения которых остаются постоянными и не могут быть изменены во время выполнения программы. Константы могут быть именованными и неименованными. Синтаксис языка определяет три типа констант: символьные, целые числа и вещественные числа.

Символьная константа служит для изображения отдельных знаков и представляет собой лексему, состоящую из символа (или любой последовательности символов), заключенного в кавычки. Например, «P», «Program», «3.14», «+» — неименованные символьные константы.

Синтаксисом языка предусмотрены десятичные целые константы, шестнадцатеричные целые константы и восьмеричные целые константы. Целая десятичная константа представляется десятичным целым числом:

44, 684, 0, 1024 — неименованные *десятичные целые константы.*

Целая шестнадцатеричная константа представляется как последовательность шестнадцатеричных цифр, перед которой записан префикс &H:

&H16 — неименованная шестнадцатеричная константа, соответствующая десятичному целому 22;

&HFF — неименованная шестнадцатеричная константа, соответствующая десятичному целому 255.

Восьмеричная целая константа представляется как последовательность цифр, не содержащая десятичных цифр старше 7, которой предшествует префикс &O:

&O16 — восьмеричное представление десятичного целого 14;

&O100 — восьмеричное представление десятичного числа 64.

Вещественные константы представляются в памяти ЭВМ в форме с плавающей точкой. Каждая вещественная константа состоит из следующих частей: целая часть (десятичная целая константа); десятичная точка; дробная часть (десятичная целая константа); признак показателя «e» или «E»; показатель (десятичная целая константа):

```
44. , 3.14159 , 44e0 , .314159E1
```

Различают встроенные константы Visual Basic (предопределенные константы) и константы создаваемые пользователем.

Встроенные константы используются, например, для определения цветовых наборов, задач доступа к данным, кодов клавиш, контуров и т.д. Встроенные в Visual Basic константы имеют префикс vb. Встроенные константы, которые могут использоваться для различных функций, определены в разделах справки для этих функций. Для того, чтобы узнать конкретное значение константы, можно воспользоваться окном Просмотр объектов (Object Browser) (рис. 7.1), которое вызывается клавишей <F2> или щелч-

ком на соответствующей кнопке панели инструментов. После выбора константы ее наименование и значение появится в текстовом поле в нижней части окна Object Browser.

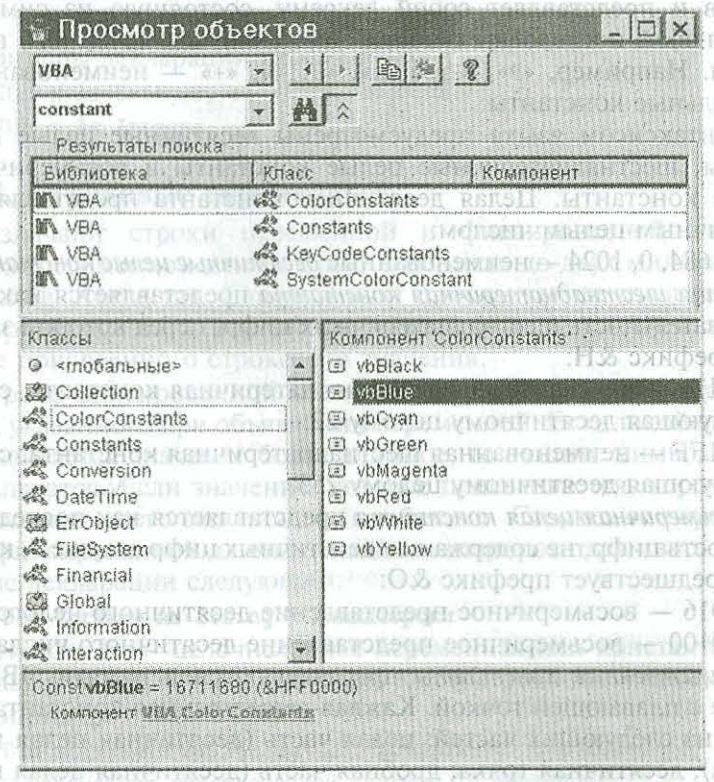

Рис. 7.1. Просмотр констант в окне Просмотр объектов

Примеры использования предопределенных констант:

```
MsgBox "Текстовое сообщение", vbInformation
```

В этом примере предопределенная константа `vbInformation` указывает, что в окне вывода (рис. 7.2) должен быть значок «Информация».

```
MsgBox "Нет данных для расчета", vbExclamation
```

В этом примере предопределенная константа указывает, что в окне вывода должен быть значок «Внимание».

Однако встроенного запаса констант при подготовке программы иногда бывает недостаточно. В этом случае можно создавать свои собственные именованные константы. Для определения

константа служит ключевое слово `Const`. Синтаксическая конструкция для декларирования констант напоминает оператор для декларации переменной и имеет вид:

```
[Public|Private] Const ИМЯ_КОНСТАНТЫ [As Тип] = значение
```

Например:

```
Const PI As Single = 3.1415 'Объявлена именованная _
    числовая константа для хранения значения числа Pi.
```

Рис. 7.2. Окно вывода vbInformation

7.1.5. Области видимости переменных и констант

Переменная может быть доступна для всей программы или только для одной или нескольких ее частей. Область программы, в которой может быть использована переменная, называется областью видимости переменной. Переменные могут быть видимы в одной процедуре, в любой процедуре какой-либо формы или во всей программе. Если переменная видима, то она доступна и, следовательно, существует. Переменная может существовать и быть доступной для некоторых частей программы и при этом быть недоступной («невидимой») для других частей программы. Пример:

```
Option Explicit
Dim strФамилия As String * 18 'Глобальная переменная
Dim strИмя As String * 10 'Глобальная переменная
Private Sub UserForm_Initialize()
Dim strАдрес As String * 30 'Локальная переменная
strФамилия = "Петров"
strИмя = "Леонид"
strАдрес = "Москва, Кронштадская, дом 37"
Вывод strАдрес
End Sub
Sub Вывод(strD As String)
Dim strФамилия As String * 18 'Локальная переменная
strФамилия = "Это уже не Петров!"
MsgBox strФамилия
```

```
    MsgBox strИмя
    MsgBox strD
End Sub
```

В результате выполнения этой программы в окнах вывода будет следующий текст:

Это уже не Петров!
Леонид
Москва, Кронштадская, дом 37.

Приведем пояснения: strФамилия и strИмя объявлены как глобальные переменные и существуют, пока программа выполняется. StrAдрес и strD являются локальными переменными и существуют лишь тогда, когда выполняется процедура Вывод, в которой они объявлены. В момент вызова этой процедуры глобальная переменная strФамилия существует, но является недоступной, так как ее область видимости перекрыта одноименной локальной переменной, поэтому в окно вывода будет выдано значение той переменной strФамилия, которая объявлена в процедуре Вывод. Переменная strИмя также является глобальной, но она доступна для процедуры Вывод. Переменная strAдрес объявлена в процедуре UserForm_Initialize, но она доносит значение до окна вывода, так как передается в процедуру Вывод в качестве параметра.

Область видимости переменной задается при ее декларации одним из ключевых слов:

Dim — объявляет локальные переменные, существующие только во время вызова процедур или функций, в которых они объявлены. Если переменная объявляется в разделе глобальных объявлений модуля или формы, то она доступна для всех процедур и функций этого модуля. Для других модулей она будет не видна.

Private — не может объявлять переменную внутри процедуры или функции, при объявлении в разделе глобальных объявлений модуля Dim и Private равнозначны.

Public — объявленная таким образом переменная является глобальной на уровне приложения и доступна из всех его модулей.

Различают динамические и статические переменные. Статические переменные объявляются внутри процедуры или функции и вне их недоступны. В отличие от обычных локальных переменных они не инициализируются при входе в процедуру или функцию, где они объявлены. Для создания статической переменной необходимо при ее объявлении вместо ключевого слова Dim указать слово Static:

```
Static intNumber As Integer
```


7.1.6. Декларация массивов

В Visual Basic различают два вида переменных — простые переменные и переменные структурного типа. Простые переменные служат для идентификации и резервирования памяти под одно данное. Переменные структурного вида предназначены для идентификации и резервирования памяти для нескольких данных. Частным случаем переменной структурного вида является массив. Массив представляет собой структуру, все элементы которой имеют одинаковый тип. Например, это могут быть данные, определяющие вектор или матрицу. Массивы могут быть одномерными и многомерными. Так, для отображения вектора может быть использован одномерный массив, а для отображения матрицы — многомерный.

Декларация массива имеет следующий вид:

```
[Public | Private] Dim Имя_Массива(индексы) As Тип_Данных
```

где Dim — ключевое слово, указывающее, что объявляется переменная; Имя_Массива — идентификатор, определяющий имя массива; Индексы — значение индекса (номера) последнего элемента в массиве, считая с нулевого; As — ключевое слово, предваряющее указание типа элементов массива; Тип_Данных — любой, действительный для Visual Basic тип данных — базовый, или созданный пользователем.

Например, декларация одномерного массива из восьми элементов выглядит следующим образом:

```
Dim MyArray(7) As Integer 'Одномерный массив из 8 элементов
```

При декларации многомерного массива в поле индекса указывается несколько индексов, в соответствии с размерностью массива. Например, двумерный массив из шести столбцов и пяти строк декларируется следующим образом:

```
Dim strMyArray(4,5) As String 'Двумерный массив _
 из 5 строк и 6 столбцов
```

По умолчанию значение нижней границы массива при таком объявлении равно нулю. В этом случае говорят, что 0 — базовый индекс массива. При необходимости базовый индекс можно изменить путем использования ключевого слова To при объявлении массива:

```
Dim MyArray (3 To 10) As String
```

В этом примере базовому индексу массива установлено значение 3. Подобным образом можно устанавливать как положительные, так и отрицательные базовые индексы:

```
Dim MyArray (-3 To 4) As String
```

Иногда в процессе выполнения программы размер массива требуется изменить. В этом случае первоначально массив декларируют как динамический. Для этого в декларации не указывается размерность, например:

```
Dim MyArray () As String
```

Количество элементов в динамическом массиве и его размерность в процессе выполнения программы можно переопределить с помощью ключевого слова ReDim. Синтаксическая конструкция переопределения массива имеет вид:

```
ReDim [Preserve] ИмяМассива(индексы) [As ТипДанных]
```

где ReDim — ключевое слово, указывающее, что переопределяются размеры массива; Preserve — необязательное ключевое слово, с помощью которого дается указание, чтобы все элементы переопределяемого массива сохранили свое значение; индексы — размерности массива (до 60).

Пример:

```
Dim MyArray () As String 'Декларация динамического массива
Dim ValArray As Integer 'Декларация переменной _
  для хранения размерности массива
ValArray=9 'Инициализация значения
ReDim strMyArray (intValArray) 'Одномерный массив _
  из 10 элементов (базовый индекс равен 0)
ReDim MyArray (3 To ValArray, 1 To ValArray) 'Двумерный _
  массив с базовыми индексами, отличными от нуля
```

7.1.7. Типы данных, определяемые пользователем

Выше упоминалось, что Visual Basic позволяет создавать собственные типы данных или, как их называют, типы данных, определяемые пользователем. Они являются типами структурного вида. Эти типы создаются на основе базовых типов Visual Basic. Возможность создавать свои типы данных полезна в тех случаях, когда программа работает с группой элементов различного базового типа, но связанных между собой по смыслу.

Создание нового типа осуществляется следующей конструкцией:

```
Type ИмяТипа
  Имя1 As Тип 'Структурный элемент создаваемого типа _
    – базовый тип
  Имя2 As Тип 'Структурный элемент создаваемого типа
    '– базовый тип
  . . .
  ИмяN As Тип 'Структурный элемент создаваемого типа
    '– базовый тип
End Type
```


где Type — ключевое слово, которое указывает, что создается новый пользовательский тип данных; ИмяТипа — имя создаваемого типа (идентификатор); NameN As Тип — описание структурного элемента создаваемого типа; End Type — ключевые слова, завершающие описание нового типа.

Например, пользовательский тип данных, предназначенный для хранения фамилии, даты рождения и даты поступления на работу сотрудника может быть объявлен следующим образом:

```
Type Сотрудник
  strФамилия As String 'Структурный элемент для хранения _
    фамилии
  ДатаРождения As Date 'Структурный элемент для хранения _
    даты рождения
  ДатаПоступления As Date 'Структурный элемент для хранения _
    даты поступления на работу
End Type
```

Созданный тип данных может быть использован в программе. Для этого в разделе деклараций программы надо объявить переменную такого типа. Объявление переменной выполняется также, как и в случае базовых типов:

```
Dim udtСлужащий As Сотрудник 'Объявлена переменная _
  пользовательского типа Сотрудник
```

Обращение к элементу пользовательского типа имеет синтаксис:

ИмяПеременной.ИмяСтруктурногоЭлемента

Пример: `udtСлужащий.strФамилия="Иванов"`

Пользовательский тип данных может содержать структурные элементы, тип которых также является пользовательским. Можно также декларировать массив, элементы которого принадлежат к определяемому пользовательскому типу.

7.2. Операторы, выражения и операции

Строка с кодом в исходном тексте программы Visual Basic называется программным оператором. *Программный оператор* — это неделимое предложение, выполняющее какое-либо действие.

Он может представлять собой любую комбинацию ключевых слов Visual Basic, свойств, функций, операций и символов, совокупность которых представляет собой корректную конструкцию, распознаваемую Visual Basic. Завершенный программный оператор может состоять из единственного ключевого слова, например Beep, или же комбинаций элементов, например, как следующий оператор, присваивающий значение системного времени свойству Caption (надпись) объекта Label1:

```
Label1.Caption = Time
```

Правила, применяемые при построении программных операторов, называются синтаксисом.

Программный оператор может включать выражения. Выражение (expression) — это комбинация знаков операций и операндов, а также скобки. Назначение любого выражения — получение некоторого значения. Синтаксическая конструкция выражения может быть представлена в виде:

```
Операнд1 [операция Операнд 2 [операция Выражение]]
```

Например, вычисление площади круга может быть записано выражением:

```
(3.14159 * D^2)/4
```

Так как результатом вычисления выражения является некоторое значение, то в программном операторе выражение может быть представлено непосредственно значением, например:

```
Pi = 3.14159
```

В зависимости от типа формируемых значений определяются типы выражений. Например, если значениями выражения являются целые и вещественные числа, то говорят об арифметических выражениях.

Для формирования и последующего вычисления выражений служат операции. Для записи операций Visual Basic имеет знаки операций, которые воспринимаются компилятором как отдельные лексемы. Каждая операция имеет свой приоритет (ранг). Операции ранга 1 имеют наивысший приоритет и в программном операторе выполняются первыми. Операции одного ранга в выражениях выполняются в соответствии с правилами ассоциативности (слева направо или наоборот).

Пример программного оператора для вычисления накопленной стоимости:

```
dblНС = dblПВ+ dblПВ * dblНП/100
```

Запись dblПВ+ dblПВ * dblНП/100 представляет собой арифметическое выражение, в котором операндами являются неименованная числовая константа 100, а так же переменные dblНС — для хранения величины накопленной стоимости, dblПВ — для хранения величины начального вклада и dblНП — для хранения нормы прибыли. Операнды связаны между собой знаками операций.

7.2.1. Операция присваивания

При декларации переменной происходит связывание имени переменной с областью памяти, в которой будет храниться ее значение. Однако это значение после декларации может оказаться произвольным. Для того чтобы присвоить переменной нужное значение используется операция присваивания. Присваивание имеет следующую синтаксическую конструкцию:

```
ИмяПеременной = Выражение
```

где ИмяПеременной — имя переменной (идентификатор); символ «=» — знак операции присваивания.

Выражение может быть значением (например, числом), либо комбинацией переменных, констант и функций, связанных знаками операций. Например:

```
intI = 6 'Переменной intI целого типа присваивается значение 6
'Структурной составляющей Фамилия пользовательского
типа Служащий присваивается фамилия Иванов:
Служащий. Фамилия = "Иванов"
TxtFirstName.Text = FirstName
```

В последнем операторе значение переменной FirstName присваивается элементу Text пользовательского типа данных с именем TxtFirstName.

7.2.2. Математические операции

Математические операции применяются для записи формул. Формула представляет собой программный оператор, содержащий числа, переменные, операторы и ключевые слова или же комбинацию этих элементов и вычисляющий новое значение. Список математических операций Visual Basic и их рангов приведен в таблице 7.4.

Операции сложения, вычитания, умножения и деления называют основными математическими операциями и дополнительных пояснений они не требуют.

Таблица 7.4.

Математические операции

Операция	Математическое действие	Ранг (приоритет)
[Операнд1] + [Операнд2]	Сложение	7
[Операнд1] − [Операнд2]	Вычитание	7
−[Операнд1]	Изменение знака числа	3
[Операнд1] *[Операнд2]	Умножение	4
[Операнд1] / [Операнд2]	Деление	4
[Операнд1] \ [Операнд2]	Целая часть от деления	5
[Опер.1]Mod [Опер.2]	Остаток от деления	6
[Операнд1] ^ [Операнд2]	Возведение в степень	2

Остальные математические операции называются дополнительными. Они применяются в специальных математических формулах и при обработке текстовой информации. Для пояснения, как они работают, рассмотрим примеры:

`Result = 10 \ 3` — результат 3 (целая часть от деления);

`Result = 10 Mod 3` — результат 1 (остаток от деления);

`Result = 3^2` — результат 9;

`Result = 9^ 0.5` — результат 3;

`Result = 2^ -2` — результат 0.25.

Общие правила применения математических операций определяются следующим синтаксисом:

```
Result = Операнд1 Операция Операнд2 [Операция Операнд3 [
…Операция ОперандN ]
```

где `Result` — переменная, содержащая результат выполнения оператора; `Операнд1`, `Операнд2`, ..., `ОперандN` — переменные, константы, числовые значения, функции.

Последовательность вычислений в программном операторе определяется приоритетом операций. Если в программном операторе некоторые выражения заключены в круглые скобки, то такие выражения вычисляются первыми (скобки являются операциями с приоритетом 1). Например:

```
Result = (25+45) * 10^2
```

В этом программном операторе в первую очередь будет вычислено выражение в скобках (25+45), затем число 10 будет возведено в степень 2 и только после этого будет выполнено умножение значения результата вычисления в скобках на значение результата возведения в степень. Результат вычисления — 7000.

Упражнение 8

Создание функции пользователя с линейным алгоритмом для вычисления накопленной стоимости

Накопленная стоимость по выданному кредиту (или по вкладу) вычисляется в соответствии с формулой:

$$S=P(1+iT),$$

где S — накопленное значение; P — величина кредита; i — номинальная процентная ставка; T — срок (в месяцах).

Требуется разработать функцию пользователя для расчета накопленной стоимости.

Прежде всего необходимо определить, какие данные являются исходными для решения задачи, и определите типы этих данных.

Для нашей задачи исходными данными являются величина кредита, номинальная процентная ставка и срок кредита. Удобно данные, их идентификаторы и типы представить в виде таблицы 7.5:

Таблица 7.5.

Идентификаторы и типы исходных данных

№ п/п	Описание переменной	Идентификатор	Тип
1	Величина кредита	ИсходнаяСумма	Sting
2	Процентная ставка	Ставка	Sting
3	Срок	Срок	Sting
4	Имя функции	НакСумма	Double

Приступая к созданию функции:

1. Выполните команду меню Вставка ➤ Модуль, а затем — Вставка ➤ Подпрограмма.
2. В поле имени появившегося окна укажите имя функции — `НакСумма`. Установите переключатель Функция. Щелкните на ОК. В окне редактирования кода появится заготовка функции вида:

```
Public Function НакСумма()
End Function
```

3. Внутри скобок заголовка функции введите описания ее параметров (исходных данных) в соответствии с приведенной таблицей, а за скобками укажите тип значения, возвращаемого функцией:

```
Public Function НакСумма(ИсходнаяСумма As String, _
    Ставка As String, Срок As String) As Double
```

Под заголовком функции поместите код функции:

```
'Вычисление накопленного значения исходной суммы по формуле
'S=P(1+iT),
'S – накопленное значение
'P – величина ссуды
'i – номинальная процентная ставка
'T – срок в месяцах
НакСумма = ИсходнаяСумма * (1 + Ставка * Срок / 12)
```

4. Сделайте комментарий к разработанной функции.
3. Проверьте работу функции на примере, вызывая ее с помощью мастера функций Excel.
5. При необходимости устраните ошибки.

Правильный результат — 1100 (для P=1000, i=10% и T=12 месяцев).

7.2.3. Операции отношения

В отличие от математических операций, результатом выполнения которых может быть любое значение, операция отношения может иметь только два результирующих значения — `True` (Истина) и `False` (Ложь), которые могут быть присвоены переменным типа `Boolean` или определенному свойству объекта. Перечень операций отношения Visual Basic приведен в таблице 7.5.

Таблица 7.6.

Операции отношения

Операция	Описание
[Операнд1]= [Операнд2]	Равно. Результат — `True`, если первый операнд равен второму.
[Операн1]<> [Операнд2]	Не равно. Результат — `True`, если первый операнд не равен второму.
[Операнд1]> [Операнд2]	Больше. Результат — `True`, если первый операнд больше второго.
[Операнд1]< [Операнд2]	Меньше. Результат — `True`, если первый операнд меньше второго.
[Операнд1]=> [Операнд2]	Больше или равно. Результат — `True`, если первый операнд больше или равен второму.

Окончание таблицы 7.6

Операция	Описание
[Операнд1]<= [Операнд2]	Меньше или равно. Результат — `True`, если первый операнд меньше или равен второму.
[Строка] Like [Маска]	Соответствие маске. Результат — `True`, если строка соответствует маске.
[Операнд1] Is [Операнд2]	Ссылка на объект. Результат — `True`, если обе переменные ссылаются на один и тот же объект.

Операции отношения применяются для записи выражений условия, результатом выполнения которых являются значения `True` (Истина) или `False` (Ложь). Примеры использования операций отношения приведены в таблице 7.7.

Таблица 7.7.

Использование операций отношения

Выражение условия	Результат
`25 <> 30`	`True` (Истина) (25 не равно 30)
`25 < 30`	`True` (Истина) (25 меньше 30)
`25 > 30`	`False` (Ложь) (25 не больше 30)
`Text1.Text = "Петров"`	`True` (Истина), если слово `Петров` является содержимым первого текстового поля, в противном случае — `False` (Ложь)
`Number >= 100`	`True` (Истина), если переменная `Number` содержит значение не меньше 100, в противном случае — `False` (Ложь)

7.2.4. Логические операции

Логические операции применяются в логических выражениях. Если существует несколько условий выбора в операциях отношения, то эти операции связываются между собой логическими операциями. Логические операции Visual Basic приведены в таблице 7.8.

Таблица 7.8.

Логические операции

Логическая операция	Действие
[Операнд1] And [Операнд2]	Если связываемые условия имеют значение True, то результирующее значение также будет True.
[Операнд1] Or [Операнд2]	Если одно из связываемых условий будет иметь значение True, то результирующим значением также будет True.
Not [Операнд]	Если условие имеет значение True, то результирующим значением будет False. Если условие имеет значение False, то результирующее значение будет True.
[Операнд1] Xor [Операнд2]	Если только одно из связываемых условий имеет значение True, то результирующее значение так же будет True. Если оба условия имеют одинаковые значения, то результирующее значение будет False.

Таблица 7.9.

Примеры применения логических операций

Логическое выражение	Результат
TextBox1.Text = "Иванов" And _ Number < 20	True (Истина), если оба выражения сравнения имеют значение True (Истина).
TextBox1.Text = "Иванов" Xor _ Number < 20	False (Ложь), если оба выражения сравнения имеют значение True (Истина).
TextBox1.Text = "Иванов" Or _ Number < 20	True (Истина), если хотя бы одно выражение сравнения имеет значение True (Истина). False (Ложь), если оба выражения имеют значение False (Ложь).
Not Number < 20	True (Истина), если выражение сравнения имеет значение False (Ложь).

7.2.5. Операции для работы со строками

В Visual Basic есть только одна операция для работы со строками — это операция конкатенации. *Конкатенация* позволяет объединить значения двух или нескольких строковых переменных

или строковых констант. Знаком операции конкатенации является символ амперсанд (&). При конкатенации строк значение второй строки добавляется в конец первой строки. Результатом операции является более длинная строка, составленная из исходных строк.

Выражение с применением операции конкатенации имеет следующий вид:

```
strВыражение_1 & strВыражение_2 [ . . . & strВыражение_N]
```

где strВыражение являются строковыми выражениями, которые могут быть любыми допустимыми строками (строковыми переменными, строковыми константами или функциями обработки строк). Знак амперсанда между строковыми выражениями указывает, что производится конкатенация этих выражений. Знак амперсанда отделяется от выражения пробельным символом. В одном операторе можно объединять любое количество строковых выражений.

Например:

`strResult = "Студент" & "Иванов"` — здесь объединяются две неименованные строковые константы. Результатом операции конкатенации будет значение «Студент Иванов».

Тот же результат будет получен при использовании следующего кода:

```
Dim strName As String
Dim strResult As String
strName = "Иванов"
strResult = "Студент" & strName 'Здесь в операции _
    конкатенации участвуют неименованная строковая _
    константа и строковая переменная.
```

Объединение строк может быть также выполнено с помощью операции сложения (знак операции «+»). В этом случае результат операции зависит от типа операндов. Например, если операнды являются числовыми типами, то будет выполняться арифметическая операция сложения, если операнды строкового типа, то выполняется их объединение.

7.3. Операторы управления Visual Basic

Операторы управления языка служат для управления работой программы. К ним относятся:

операторы передачи управления;
операторы выбора;
операторы циклов.

7.3.1. Операторы передачи управления

Операторы передачи управления применяются в программе для реализации безусловных алгоритмических конструкций. Они выполняют переход с одного участка программы на любой другой без какого-либо условия. Оператор перехода имеет следующий вид:

```
GoTo идентификатор
```

где GoTo — ключевое слово; идентификатор — одна из меток программы.

Метка — это идентификатор, помещаемый слева от программного оператора и отделенный от него двоеточием. Например:

```
m1: Text1.Text = "это метка"
```

Оператор перехода для перехода к оператору с меткой m1 будет иметь вид:

```
GoTo m1
```

Следует сказать о том, что для получения хорошего стиля программирования следует избегать применения оператора GoTo, так как в этом случае ухудшается читаемость и понимание программы.

7.3.2. Операторы выбора

Операторы выбора используются в программе для реализации условных алгоритмических конструкций, которые вызывают выполнение различных частей программы в соответствии с условиями, существующими на момент выполнения этих операторов. Одним из важных элементов программного оператора выбора является выражение условия, значением которого может быть Истина (True) или Ложь (False). Такие выражения условия записываются с помощью операторов отношения.

В Visual Basic есть два основных типа операторов выбора: операторы условия If … Then и переключатели, реализуемые оператором Select Case.

7.3.2.1. Операторы If ... Then

В Visual Basic есть два типа операторов If … Then — линейный и блочный.

Линейный оператор If … Then используется для того, чтобы выполнить какой-либо один оператор, если некоторое условие будет истинным. Условие является выражением или функцией, истинность которого оценивается. Синтаксическая конструкция линейного оператора имеет две формы — безальтернативную:

```
If условие Then Выражение
```


и альтернативную:

```
If условие Then Выражение_1 Else Выражение_2
```

В безальтернативной форме при значении условия True сначала выполняется выражение, следующее за ключевым словом Then, а затем следующий в последовательности оператор. Если условие принимает значение False, то выражение следующее за словом Then не выполняется, а выполняется следующий оператор.

Примеры линейного оператора If … Then безальтернативной формы:

```
If intNamber<3 Then intResult = 10*2 'Если переменная _
    intNamber содержит значение меньше трех, то переменной _
    intResult будет присвоено значение 20
If optAddition.Value = True Then z = x + y
```

Пример линейного оператора If … Then … Else:

```
If intNamber<3 Then intResult = 10*2 Else intResult =30
```

В результате выполнения этого оператора переменной intResult будет присвоено значение 20, если переменная intNamber содержит значение меньше трех, в противном случае переменной intResult будет присвоено значение 30.

Блочный оператор If … Then используется тогда, когда в случае истинности условия необходимо выполнить несколько программных операторов (блок операторов). Блочный оператор также может быть безальтернативным и альтернативным. Структура безальтернативного блочного оператора следующая:

```
If Условие Then
  Программный оператор 1
  Программный оператор 2
  . . .
  Программный оператор n
End If
```

Здесь End If указывает на окончание блока оператора If.

Альтернативный блочный оператор If применяется в тех случаях, когда при выполнении условия необходимо выполнить один набор программных операторов, а при невыполнении — другой. Это реализуется с помощью оператора If … Then … Else, синтаксис которого следующий:

```
If условие Then
  <<Блок программных операторов, выполняемых при значении
  условия True>>
Else
  <<Блок программных операторов, выполняемых при значении
```

```
    условия False>>
  End If
```

Операторы If могут быть вложенными друг в друга. Такое вложение операторов применяется, если нужно проверить какое-либо условие при другом условии, которое является истинным (например, если фамилия Иванов и он торговый агент). Формат вложенного оператора If следующий:

```
If условие_1 Then
  If условие_2 Then
    Блок операторов 1
  Else
    Блок операторов 2
  EndIf
EndIf
```

Пример применения вложенного оператора If:

```
If Name = "Иванов" Then
  If Rang = "Торговый агент" Then
    Text1.Text = Name & Rang
  Else
    Text1.Text="Агента с указанной фамилией в списке нет"
  EndIf
EndIf
```

В этом примере условие первого оператора If проверяет, соответствует ли значение переменной Name значению неименованной строковой константы "Иванов". Если результат этой операции True, то вложенный оператор If сделает проверку, является ли Иванов торговым агентом. В случае истинности результата в текстовое поле будет выведено значение «Иванов торговый агент». В противном случае — «Агента с указанной фамилией в списке нет».

При использовании вложенных операторов If важно не перепутать варианты сочетания условий. Нужно помнить правило: альтернатива Else считается принадлежащей ближайшему оператору If, не имеющему ветви Else.

В Visual Basic также предусмотрена конструкция для работы с несколькими операторами If. Несколько операторов If применяются в случаях, когда необходимо рассмотреть еще несколько условий в дополнение к исходному. Для этого служит конструкция If … Then … ElseIf. В отличие от вложенных операторов конструкция с несколькими операторами If позволяет проверить дополнительное условие, если исходное условие принимает значение

False. Форма записи нескольких операторов If имеет следующую синтаксическую конструкцию:

```
If выражение_условия_1 Then
  Программный оператор 1
ElseIf выражение_условия_2 Then
  Программный оператор 2
Else
  Программный оператор 3
EndIf
```

Например:

```
If CorrectAnsver.Text >=8 Then
  Ball.Text ="Отлично"
ElseIf CorrectAnsver.Text >=6 Then
  Ball.Text = "Хорошо"
ElseIf CorrectAnsver.Text >=4 Then
  Ball.Text = "Удовлетворительно"
Else Ball.Text = "Неудовлетворительно"
EndIf
```

Приведенный код программы определяет количество правильных ответов и выставляет оценку. Он работает следующим образом. Сначала проверяется значение условия в операторе If. Если оно принимает значение True, то выполняется оператор (или блок операторов), следующий непосредственно за ключевым словом Then, после чего программа перейдет к выполнению оператора, следующего за EndIf. Если первое условие принимает значение False, то программа перейдет к выполнению первого оператора ElseIf, чтобы проверить выполнение его условия. Если оно имеет значение True, то выполняется оператор Ball = 4 и программа перейдет к выполнению оператора, следующего за EndIf. В противном случае эта последовательность действий повторится для следующего оператора ElseIf и так до тех пор, пока не будут проверены все из них.

7.3.3. Переключатели

Переключатели в Visual Basic реализуются оператором Select Case, который позволяет сделать выбор из нескольких альтернативных вариантов в зависимости от значения условного выражения. Синтаксис его следующий:

```
Select Case Выражение
  Case Значение_1
```

```
        Оператор, выполняемый при совпадении Значения_1 и
        значения Выражения.
    Case Значение_2
        Оператор, выполняемый при совпадении Значения_2 и
        значения Выражения.
    . . .
    Case Значение_N
        Оператор, выполняемый при совпадении Значения_N и
        значения Выражения.
    End Select
```

В операторе `Select Case` можно использовать операции отношения. Для этого надо включить в выражение ключевое слово `Is` или ключевое слово `To`. Ключевое слово `Is` дает указание компилятору сравнить значение проверяемой переменной со значением выражения, следующего за ключевым словом `Is`. Ключевое слово `To` задает диапазон значений. Например, следующий код фрагмента программы, использующий оператор `Select Case` позволяет выставить оценку исходя из общего количества набранных баллов:

```
Select Case ПравильныйОтвет.Text
  Case 8 To 10 'Если сумма баллов в диапазоне 8-10
    Балл.Text = "Отлично"
  Case 6 To 7 'Если сумма баллов в диапазоне 6-7
    Балл.Text = "Хорошо"
  Case 4 To 5 'Если сумма баллов в диапазоне 4-5
    Балл.Text = "Удовлетворительно"
  Case Is < 4 'Если сумма баллов меньше 4
    Балл.Text = "Неудовлетворительно"
End Select
```

Упражнение 9

Разработка функции пользователя с проверкой корректности исходных данных

Требуется доработать функцию пользователя, разработанную в упражнении 8, таким образом, чтобы она проверяла корректность исходных данных. Данные корректны, если они являются числовыми. Для этой цели используем оператор `If … Then`.

1. Откройте окно редактирования кода для модуля, содержащего функцию `НакСумма`.
2. Дополните ее операторами проверки условий корректности. Поместите операторы для вывода сообщений оператору, если данные некорректны. После доработки функции ее код будет следующий:

```
'Вычисление накопленного значения исходной суммы по формуле
'S=P(1+iT),
'S – накопленное значение
'P – величина ссуды
'i – номинальная процентная ставка
'T – срок в месяцах
'Проверка корректности данных
If (IsNumeric(ИсходнаяСумма) And IsNumeric(Ставка) And _
    IsNumeric(Срок)) = False Then
  MsgBox "Ошибка в исходных данных", vbInformation, _
   "Вычисление накопленной суммы"
  НакСумма = 0
  Exit Function
Else
  НакСумма = ИсходнаяСумма * (1 + Ставка * Срок / 12)
End If
```

Здесь применена встроенная функция `IsNumeric`, которая возвращает значение `True`, если ее аргумент является числом, и `False`, если аргумент не число.

4. Проверьте работу функции для корректных и некорректных данных, вызывая ее с помощью мастера функций Excel.

7.4. Программирование циклов

Цикл — это оператор или группа операторов, которые программа многократно выполняет до тех пор, пока не получит команду начать выполнение чего-либо другого. В Visual Basic существуют два основных типа циклов — циклы со счетчиком (с известным числом повторений) и циклы с условием. Циклы со счетчиком используют в тех случаях, когда необходимо выполнить некоторые действия определенное число раз. Циклы с условием применяются тогда, когда некоторые действия в программе должны повторяться до тех пор, пока выполняется определенное условие или до тех пор, пока не будет выполнено определенное условие.

7.4.1. Циклы со счетчиком

Циклы со счетчиком (с известным числом повторений) в Visual Basic еще называют циклами `For`, или циклами `For … Next`. Так они

называются потому, что начало и конец этих циклов определяются операторами For и Next. Синтаксис цикла For … Next таков:

```
For СчетчикЦикла = НачальноеЗначение To
                   КонечноеЗначение [Step Шаг]
  операторы
  [Exit For]
  Next [СчетчикЦикла]
```

где For — ключевое слово Visual Basic, обозначающее начало цикла;

СчетчикЦикла — переменная, определенная в качестве счетчика цикла;

НачальноеЗначение — число, задающее начальное значение счетчика;

To — ключевое слово Visual Basic, разделяющее НачальноеЗначение и КонечноеЗначение — число, задающее значение счетчика, при котором цикл завершается;

Step — ключевое слово Visual Basic, используемое для задания шага цикла, необязательный аргумент;

Шаг — число, задающее шаг цикла, т.е. значение, на которое увеличивается (или уменьшается) значение счетчика на каждом шаге, это число может быть отрицательным;

Exit For — оператор досрочного выхода из цикла (необязательный);

Next — ключевое слово Visual Basic, обозначающее конец цикла.

В начале цикла For … Next определяется переменная-счетчик, а так же начальное и конечное значения этой переменной. В самом начале выполнения цикла переменная-счетчик устанавливается в начальное значение. Каждый раз, когда программа пройдя через цикл, возвращается к его началу, значение счетчика увеличивается. Если используется ключевое слово Step, то переменная-счетчик изменяется в соответствии с числом, указанным после ключевого слова Step. Например:

```
For I = 0 To 10 Step 2 'Значение I будет увеличиваться на 2
```

Если ключевое слово Step отсутствует, то значение шага равно единице.

Каждый раз, когда значение переменной-счетчика изменяется, оно сравнивается с заданным конечным значением счетчика. Если значение счетчика превышает конечное значение, программа сразу выходит из цикла и переходит к той строке кода, которая следует за циклом.

Например:


```
Option Explicit
Dim I As Integer
For I = 1 To 4
Sum =Sum + 2
Next I
```

Этот цикл эквивалентен написанию четырех операторов Sum=Sum + 2 в тексте программы.

Цикл For … Next может быть прерван досрочно, например, при достижении какого-либо условия. Для этого в нужном месте цикла нужно поместить оператор Exit For. Например:

```
Option Explicit
 Dim Sum As Integer
  Dim j As Integer
  Sum = 2
  For j = 1 To 10
    Sum = Sum + j
    If Sum > 6 Then
      Exit For 'Выход из цикла, если значение Sum больше 6
    End If
  Next j
  TextBox1.Text = Sum
```

В этом примере цикл прерывается досрочно, когда значение переменной Sum будет больше 6. В результате выполнения этого кода в текстовом поле будет выведено вычисленное значение переменной Sum равное 8.

7.4.2. Циклы с условием

Главной особенностью циклов с условием является условие, которое может быть любым выражением, принимающим значение True (Истина) или False (Ложь). В Visual Basic есть два основных цикла с условием — цикл Do While … Loop и цикл Do Until … Loop. Оба они могут быть с предусловием или с постусловием.

Циклы Do While | Until имеют следующий синтаксис:

Цикл с предусловием:

```
Do While | Until Выражение
  Операторы
  [Exit Do]
Loop
```

Цикл с постусловием:

```
Do
  Операторы
  [Exit Do]
Loop While | Until Выражение
```

где `Do` — ключевое слово; `While | Until` — ключевые слова, указывающие тип цикла; `Выражение` — выражение условия, принимающее значение `True` или `False`; `Loop` — ключевое слово, указывающее на окончание цикла.

Цикл `Do While` выполняется до тех пор, *пока выражение условия* имеет значение `True`.

Пример цикла `Do While`, реализующего алгоритм программы, аналогичный приведенной в примере для цикла `For` с досрочным прерыванием:

```
Option Explicit
Dim Sum As Integer
Dim j As Integer
Sum = 2
Do While Sum < 7
  Sum = Sum + j
  j = j + 1
Loop
TextBox1.Text = Sum
```

В результате выполнения этого кода в текстовом поле будет выведено вычисленное значение переменной `Sum` равное 8.

В отличие от цикла `Do While` цикл `Do Until` выполняется до тех пор, пока *выражение условия имеет значение* `False`.

Пример цикла `Do Until`, реализующего алгоритм программы аналогичный приведенному выше:

```
Option Explicit
Dim Sum As Integer
Dim j As Integer
Sum = 2
Do Until Sum >7
  Sum = Sum + j
  j = j + 1
Loop
TextBox1.Text = Sum
```

В результате выполнения этого примера в текстовом поле будет выведено вычисленное значение переменной `Sum` равное 8.

Иногда бывает необходимо прервать цикл `Do ... Loop`, если выполняется какое-либо дополнительное условие. Это может быть сделано с помощью оператора `Exit Do`. Например:

```
Option Explicit
Dim Sum As Integer
Dim j As Integer
Sum = 2
Do Until Sum > 7
  Sum = Sum + j
  j = j + 1
  If j > 3 Then
    Exit Do 'Досрочный выход из цикла Do ... Loop
  End If
Loop
TextBox1.Text = Sum
```

В этом примере цикл с условием досрочно прерывается, если выполняется дополнительное условие `j > 3`. В результате выполнения программного кода переменная `Sum` будет иметь значение, равное 8.

Используя цикл с условием, можно организовать бесконечный цикл. Иногда это бывает необходимо, а иногда это происходит из-за ошибки пользователя. Для создания бесконечного цикла условное выражение должно быть таким, которое никогда не выполняется или выполняется всегда. Например:

```
Do Until 1
  Операторы
Loop
```

Выйти из такого бесконечного цикла и прервать работу программы можно нажав комбинацию клавиш `<Ctrl+Break>`.

Упражнение 10

Расчет накопленной стоимости при неравномерных непериодических платежах рассчитывается по формуле:

$$I = \sum_{j=1}^{k-1} P_j i\, dt$$

где P_j — сумма j-го платежа; i — номинальная годовая ставка; dt — продолжительность периода, за который расчитываются проценты; k — количество платежей.

Требуется разработать функцию пользователя для расчета накопленной стоимости по методу простых процентов по приведенной выше формуле.

Задаваемыми исходными данными для решения задачи являются величины суммы платежей, даты их выплаты и номинальная процентная ставка. Вычисляемыми данными являются количество платежей и продолжительность периода.

Таблица 7.10.

Исходные и вычисляемые данные

Переменная	Идентификатор	Тип данных	Примечание
Величина платежа	Платеж	Variant	Параметр функции, передающий значения диапазона ячеек
Дата платежа	Дата	Variant	Параметр функции, передающий значения диапазона ячеек
Номинальная ставка	Ставка	Single	Параметр функции
K	K	Integer	Вычисляемое значение
	i	Integer	Счетчик цикла
Dt	Дельта	Integer	Вычисляемое значение как разность между двумя смежными датами
Текущая сумма	Summa	Double	Вспомогательная переменная
Функция	НАКСУММА	Double	Имя функции (возвращаемое значение)

Примечание. Если параметр функции передает значения массива ячеек, то он должен иметь тип `Variant`.

Для создания функции выполните действия:

1. Создайте модуль.
2. Вставьте в созданный модуль подпрограмму-функцию.
3. Введите в нее следующий код:

```
Public Function НАКСУММА(Ставка As Single, _
   Платеж As Variant, Дата As Variant) As Double
 Dim K As Integer
 Dim i As Integer
 Dim Summa As Double 'Текущая сумма вклада
 Dim Дельта As Integer 'Разность в днях между
 'двумя платежами
 Summa = Платеж(1)
 K = Платеж.Count 'общее количество платежей
 For i = 2 To K
   Дельта = Дата(i) - Дата(i - 1)
   Summa = Summa + Summa * Ставка * Дельта / 365 + Платеж(i)
 Next
 НАКСУММА = Summa
End Function
```


Для проверки работоспособности функции решите следующую задачу:

Клиент сделал вклад на текущий счет в банке в сумме 1000 р. под 60 процентов годовых. Через 3, 6 и 9 месяцев он вложил еще по 1000 р. В конце учетного года клиент закрыл счет. Какую сумму он получил при закрытии счета?

На рабочем листе Excel подготовьте исходные данные, как показано ниже, и выполните расчет, применив созданную функцию.

Платежи	Даты платежей		
1000 р.	01.01.01		
1000 р.	01.04.01		
1000 р.	01.07.01		
1000 р.	01.10.01	**Накопленная сумма**	**Правильный результат**
0	31.12.01		5740,95 р.

7.5. Встроенные функции

Visual Basic имеет набор встроенных функций. Перечень наиболее часто используемых функций приведен в Приложении 2. По назначению встроенные функции объединяются в следующие группы:

- финансово-математические функции;
- функции преобразования типа;
- математические функции;
- функции статуса;
- функции обработки строк;
- прочие функции;
- функции даты и времени;

- функции для работы с массивами;
- функции для работы с файлами;
- прочие функции.

7.5.1. Финансово-математические функции

Эта группа функций предназначена для выполнения некоторых наиболее распространенных типовых финансовых расчетов. Перечень и назначение финансово-математических функций приведены в таблице П.1 Приложения.

Рассмотрим примеры использования некоторых из них.

Упражнение 11

Использование встроенных финансово-математических функций

Пусть необходимо вычислять величину амортизации основных фондов в указанный период эксплуатации. Стоимость автомобиля в начале эксплуатации 150 тыс. р., а его стоимость в конце эксплуатации через 6 лет составляет 20 тыс. р. Требуется определить величину амортизации для третьего года эксплуатации. Расчеты сделать для случая равномерной амортизации и для случая двукратной амортизации. Для решения задачи можно воспользоваться финансово-математическими функциями SYD и DDB (табл. П.1).

Функции имеют следующий синтаксис:

```
SYD(Стоимость, Ликвидная_стоимость, Жизнь, Период)
DDB(Стоимость, Остаточная_стоимость, Время_эксплуатации,
    Период, Кратность)
```

Параметрами этих функций, посредством которых задаются исходные данные для расчетов, являются:

`Стоимость` — стоимость единицы основного фонда в начале эксплуатации;

`Остаточная стоимость` — стоимость в конце эксплуатации;

`Время эксплуатации` — продолжительность эксплуатации в периодах (месяц, год);

`Период` — номер периода, для которого производится расчет;

`Кратность` — целое число, определяющее кратность амортизации (метод расчета).

Для создания функции определим ее параметры, оформив их в виде таблицы 7.11.

Таблица 7.11.

Параметры функций

Параметр	Идентификатор	Тип
Стоимость в начале эксплуатации	Стоимость	Double
Стоимость в конце эксплуатации	Остаточная_стоимость	Double
Время эксплуатации	Время_эксплуатации	Double
Номер периода	Период	Integer
Кратность амортизации	Кратность	Byte
Тип амортизации	Тип	Byte
Имя функции	АМОРТИЗАЦИЯ	Double

Задание:

1. Создайте в одном из модулей проекта функцию для расчета амортизации с именем `АМОРТИЗАЦИЯ`.
2. Поместите в нее следующий код:

```
Public Function АМОРТИЗАЦИЯ(Стоимость As Double, _
    Остаточная_стоимость As Double, Время_эксплуатации As _
    Integer, Период As Integer, Тип As Byte, _
    Кратность As Byte) As Double
    'Проверка вида расчета. Если Тип 0, то кратный метод
    'иначе равномерная амортизация
    If Тип 0 Then
        АМОРТИЗАЦИЯ = DDB(Стоимость, Остаточная_стоимость, _
        Время_эксплуатации, Период, Кратность)
    Else
        АМОРТИЗАЦИЯ = SYD(Стоимость, Остаточная_стоимость, _
        Время_эксплуатации, Период)
    End If
End Function
```

3. На рабочем листе поместите исходные данные для проверки функции (рис. 7.3).
4. Примените созданную функцию для расчета амортизации по данным таблицы.
5. Отладьте код программы, если необходимо.
6. Поместите в свойство функции ее краткое описание.

	Кратная амортизация	Равномерная амортизация
Начальная стоимость	150 000,00р.	150 000,00р.
Конечная стоимость	20 000,00р.	20 000,00р.
Срок эксплуатации	6	6
Период расчета	3	3
Тип аморитизации	1	0
кратность	2	0
Результат:		
Сумма амортизации	22 222,22р.	24 761,90р.

Рис. 7.3. Результаты расчета амортизации

7.5.2. Функции преобразования типов

Visual Basic, в некоторых случаях, выполняет автоматическое преобразование одного типа данных в другой. Однако в процессе автоматического преобразования могут возникать ошибки, а в некоторых случаях автоматическое преобразование не выполняется. Поэтому в процессе разработки программы пользователь должен сам определять необходимость преобразования типов данных. Для этого нужно применять функции преобразования типов (табл. П.2).

Рассмотрим пример. Допустим, надо сложить два числа, значения которых вводятся с помощью управляющих элементов `TextBox`. Если вычисления будут выполняться программой, код которой имеет вид:

```
TextBox3.Text= TextBox1.Text + TextBox2.Text
```

то результат вычисления будет неверным, так как в этом случае выполнится объединение строк. Например, если исходными значениями будут 100 и 200, то получится результат 100200, что, конечно, не является результатом арифметического сложения.

Для того, чтобы получить правильный результат, предварительно нужно преобразовать исходные величины строкового типа в числовые целого типа (или другого числового типа). Это преобразование может быть выполнено с помощью функции `CInt`. Тогда фрагмент кода для выполнения сложения будет иметь вид:

```
TextBox3.Text = CInt(TextBox1.Text) + CInt(TextBox2.Text)
```

При выполнении этого кода будет получен правильный результат — 300. Также правильный результат будет получен, если фрагмент кода будет следующий:

```
TextBox3.Text = Val(TextBox1.Text) + Val(TextBox2.Text)
```


В последнем фрагменте для преобразования типов применена функция `Val`, которая преобразует строку числовых символов в число, поэтому выполняется не объединение строк, а сложение чисел. Различие этих двух функций заключается в следующем: для функции `CInt` аргументом должна быть строка только числовых символов, иначе будет ошибка; для функции `Val` аргумент может содержать не числовые символы. В этом случае для правильного преобразования числовые символы должны располагаться в начале строки. Если лидирующее положение занимают не числовые символы, то результатом преобразования будет значение 0. Так если `TextBox1.Text` содержит значение `100BB`, `TextBox2.Text` — `200Text`, то все равно результатом сложения будет значение 300. Если `TextBox1.Text` содержит значение `A100`, `TextBox2.Text` — `Text200`, то результатом сложения будет 0.

7.5.3. Математические функции

Математические функции (табл. П.3) предназначены для выполнения типовых математических расчетов (вычисление значений тригонометрических функций, логарифмов, получения псевдослучайных чисел и др.). Пусть, например, требуется вычислить значение квадратного корня числа 100. Фрагмент кода может иметь следующий вид:

```
Option Explicit
Dim dblАргумент As Double, dblРезультат As Double
  dblАргумент=100
  dblРезультат=Sqrt(dblАргумент)
```

В результате выполнения этого кода переменная `dblРезультат` получит значение 10.

7.5.4. Функции обработки строк

Функции обработки строк Visual Basic приведены в таблице П.5. Они служат для выполнения операций со строками. В качестве иллюстрации применения некоторых из них рассмотрим пример.

В финансовых расчетах часто приходится округлять полученные значения до двух знаков после запятой. Такое округление можно выполнить, если исходное значение предварительно поместить в строковую переменную и сделать следующие шаги:

- вычислить номер позиции разделителя используя функцию `Instr`;

- выделить целую часть до позиции разделителя с помощью функции Left;
- выделить дробную часть с помощи функции Mid;
- преобразовать дробную часть, вставив символ разделителя после второго знака;
- преобразовать полученное значение дробной части в тип с плавающей точкой используя функцию преобразования типа CDbl;
- округлить полученное число до целого с помощью функции Int;
- используя операцию объединения строк сформировать строковое представление округленного числа.

Ниже приведен фрагмент кода, выполняющий такое округление.

```
Option Explicit
Dim StrИсходнаяСтрока As String * 10
Dim strЦелаяЧасть As String * 4
Dim strДробнаяЧасть As String * 7
Dim intПозицияРазделителя As Integer
Dim dblДробнаяЧасть As Double
StrИсходнаяСтрока = TextBox1.Text
intПозицияРазделителя = InStr(1, StrИсходнаяСтрока, ",")
strЦелаяЧасть = Left(StrИсходнаяСтрока, _
    intПозицияРазделителя - 1)
strДробнаяЧасть = Mid(StrИсходнаяСтрока, _
    intПозицияРазделителя + 1, 2) & "," _
    & Mid(StrИсходнаяСтрока, intПозицияРазделителя + 3)
dblДробнаяЧасть = CDbl(strДробнаяЧасть)
strДробнаяЧасть = Fix(dblДробнаяЧасть)
TextBox2.Text = strЦелаяЧасть & "," & strДробнаяЧасть
```

Вопросы для самоконтроля

1. Какие символы допускается применять в именах переменных?
2. Можно ли имя переменной начинать с цифры или с символа подчеркивания?
3. Какие базовые типы данных поддерживает Visual Basic?
4. Напишите оператор, явно декларирующий строковую переменную для хранения почтового адреса.
5. Какой оператор должен присутствовать в разделе глобальных объявлений, чтобы исключить ошибки в случае появления неявно объявленных переменных?

6. Напишите оператор, декларирующий одномерный массив из 7 элементов целого типа и имеющий базовый индекс – 3.
7. Что такое динамические массивы и как они декларируются?
8. Каково различие между переменной и константой. Какие типы констант поддерживает Visual Basic?
9. Что представляет собой пользовательский тип данных, как он определяется и объявляется?
10. Каково различие между линейным и блочным оператором If … Then?
11. Какой оператор Visual Basic позволяет сделать выбор из нескольких альтернативных вариантов?
12. Как определяется цикл с известным числом повторений?
13. Какие циклы с условием поддерживает Visual Basic?
14. В чем различие циклов с предусловием и постусловием ?
15. Что такое функция и как она определяется?
16. Как производится обращение к функции?
17. Какие типы встроенных функций есть в Visual Basic?

Задания для самостоятельной работы

Задание 1

Разработайте функцию пользователя для расчета реальной доходности с учетом налога на прибыль, которая вычисляется по формуле:

$$r = \frac{i(1-g)-h}{1+h},$$

где i — годовая процентная ставка; g — налог на прибыль, выраженный в процентах; h — годовой темп инфляции, вычисляемый по формуле:

$$h = (1+ h_{1/12})^{12} - 1,$$

где $h_{1/12}$ — месячный темп инфляции.

Для проверки и отладки программы используйте следующие исходные данные: $i = 60\%$, $h_{1/12} = 3\%$, $g = 25\%$. Для них реальная доходность составит $r = 1{,}7\%$.

Задание 2

Разработайте функцию пользователя для вычисления годовой процентной ставки контракта по кредиту, взятого на определенный срок при известных сумме долга и сумме, подлежащей возврату. Используйте формулу:

$$i = \frac{S-P}{PT},$$

где P — сумма кредита; S — сумма, подлежащая возврату; T — срок кредита.

Для проверки и отладки программы используйте следующие исходные данные: $T = 4$ месяца, $P = 100$ тыс., $S = 110$ тыс. Для них годовая ставка составит $i = 30\%$

Задание 3

Разработайте функцию пользователя для вычисления накопленной по процентам суммы, при условии, что известна годовая процентная ставка, а вклады делаются неравномерными и непериодическими платежами (см. условие задачи в упражнении 10). Для расчетов используйте формулу:

$$I = \sum_{j=2}^{k} i(t_{j-1} - t_1) R_j,$$

где i — годовая процентная ставка; k — количество платежей; R_j — величина j-того платежа; t_j — дата j-того платежа.

Для проверки программы используйте условие задачи из упражнения 10. Для этих условий значение $I = 1500$.

8. Разработка программ с использованием форм пользователя

8.1. Работа с управляющими элементами

8.1.1. Элементы Кнопка, Поле, Надпись

Упражнение 12

Задание: Создайте собственное диалоговое окно для пересчета рублей в доллары по текущему курсу. Примерный вид диалогового окна приведен ниже. Для вывода окна на экран используйте кнопку, которую поместите на первом рабочем листе.

Рис. 8.1. Диалоговое окно программы перерасчета валюты

Для выполнения задания сделайте следующее:

1. Перейдите в редактор VBA, для этого выполните команду Сервис ➤ Макрос ➤ Редактор Visual Basic. Добавьте в проект новую форму (команда Вставка ➤ UserForm). Добавьте в форму нужные элементы (2 кнопки, 2 элемента Поле, 4 элемента Надпись). Для вывода значения в долларах будет использоваться элемент Надпись, так как это значение пользователь не вводит, оно вычисляется.
2. Выведите на экран окно свойств, если его нет (команда Вид ➤ Окно свойств). Измените имена элементов, заданные по умолчанию в соответствии с таблицей 8.1.
3. Измените значения свойства Caption у надписей и кнопок.

Таблица 8.1.

Элементы управления

Элемент	Имя	Устанавливаемые свойства	Обрабатываемые события
Форма	МояФорма	Caption — Пересчет рублей в доллары	нет
Кнопка	КнопкаПересчитать	Caption — Пересчитать	Проверяет корректность данных (числовые значения в полях). Если данные некорректны, то выводит сообщение оператору. Выполняет вычисления и выводит результат
Кнопка	КнопкаПоказать	Caption — Отмена	Click — Выход из программы
Поле	ПолеРубли	нет	нет
Поле	ПолеКурс	нет	нет
Надпись	НадписьЗначение	Caption — Доллары	нет
Надпись	По умолчанию	Caption — Рубли	нет
Надпись	По умолчанию	Caption — Курс	нет

4. Напишите процедуру обработки события Click для кнопки Пересчитать. Если выполнить двойной щелчок на кнопке, то вы попадете в окно кода, где уже вставлена заготовка для процедуры. Введите текст процедуры, приведенный ниже, но учтите, что имена ваших элементов могут быть другими:

```
Private Sub КнопкаПересчитать_Click()
Dim X As Double
 If IsNumeric(ПолеРубли.Text) = False Then
   MsgBox "Введите правильно значение в рублях"
   ПолеРубли.SetFocus
   Exit Sub
 End If
 If IsNumeric(ПолеКурс.Text) = False Then
   MsgBox "Введите правильно курс"
   ПолеКурс.SetFocus
   Exit Sub
 End If
```



```
 X = CDbl(ПолеРубли.Text) / CDbl(ПолеКурс.Text)
 НадписьЗначение.Caption = Format(X, "Standard")
End Sub
```

5. Напишите процедуру обработки события Click для кнопки Отмена. Она должна находиться в окне кода вашей формы:

```
Private Sub КнопкаОтмена_Click()
  МояФорма.Hide
End Sub
```

6. Вставьте на Лист1 кнопку для вывода вашего диалогового окна. Для этого перейдите на первый лист (не закрывайте редактор Visual Basic). Если панель Элементы управления не видна, то выведите ее на экран (команда Вид ➤ Панели инструментов ➤ Элементы управления). Измените надпись на кнопке (свойство Caption). Для вывода окна свойств используйте соответствующую кнопку панели Элементы управления или контекстное меню кнопки (должен быть включен режим конструктора).
7. Напишите процедуру обработки события Click для этой кнопки. Вернитесь в редактор Visual Basic, в окне проекта выберите Лист1 и выполните на нем двойной щелчок. Вы попадете в окно кода этого листа. В левом списке в верхней части окна выберите имя кнопки, а в правом списке событие Click, будет вставлена заготовка для соответствующей процедуры. Введите текст процедуры:

```
Private Sub КнопкаПоказать_Click()
  МояФорма.Show 'Показать форму
End Sub
```

8. Вернитесь в Excel и запустите программу, щелкнув на кнопке, расположенной на Листе1.
9. Проверьте корректность работы программы, при необходимости отладьте ее.

8.1.2. Элементы Переключатель и Рамка

Упражнение 13

Постановка задачи. Разработать программу для расчета величины амортизации основных фондов. В программе предусмотреть проверку корректности вводимых исходных данных и выдачу сообщений оператору в случае обнаружения какой-либо ошибки.

Под амортизацией понимается уменьшение стоимости оборудования или другого имущества в процессе эксплуатации. Величину этого уменьшения оценивают за единицу времени.

Метод. Для расчета величины амортизации на период эксплуатации служит встроенная функция VBA имеющая имя SYD. Ее синтаксис следующий:

```
SYD(ПервичнаяСтоимость, ОстаточнаяСтоимость, _
    ВремяАмортизации, ПериодРасчета, Кратность)
```

где ПервичнаяСтоимость — начальная стоимость основных фондов; ОстаточнаяСтоимость — стоимость в конце эксплуатации; ВремяАмортизации — полная продолжительность эксплуатации (в периодах); ПериодРасчета — номер периода, для которого производится расчет; Кратность — указывает метод, применяемый для расчета — стандартный (параметр опущен) или метод кратного расчета (указывается кратность, выраженная целым числом).

Разработка формы пользователя. Перед созданием формы пользователя необходимо определиться, какие элементы управления необходимы программе, а именно — для ввода исходных данных, указания на выполнение вычислений и завершения работы программы. Для расчета с использованием функции SYD необходимо задать значения пяти ее параметров. Следовательно, в форме пользователя должны быть размещены пять элементов Поле для ввода исходных данных. Еще один элемент Поле необходим для вывода результата. Для указания метода, по которому производится расчет, в форме должны быть помещены два переключателя, размещенные в одном контейнере Рамка. Для включения режима вычислений и для выхода из программы в форму пользователя следует поместить два элемента Кнопка. Кроме этих элементов, в форме должны быть размещены элементы Надпись, поясняющие назначение элементов управления. Теперь, когда ясно, какие элементы управления должны быть в форме, можно приступить к ее созданию. Выполните следующие шаги:

1. Запустите редактор Visual Basic.
2. Выполните команду меню Вставка ➤ UserForm. Появится окно редактирования формы.
3. Разместите в окне редактирования формы управляющие элементы, как показано на рисунке 8.2.
4. Установите свойства элементов управления в соответствии с таблицей 8.2.

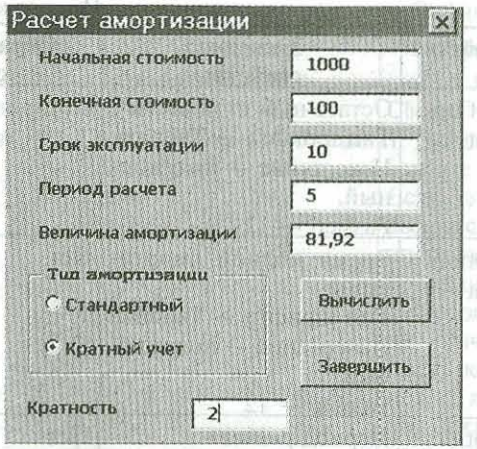

Рис. 8.2. Диалоговое окно для расчета амортизации

Таблица 8.2.

Элемент	Свойство	Значение свойства	Обрабатываемое событие
User-Form1	Caption	Расчет амортизации имущества.	Initialize: 1. Очищает поля ввода данных. 2. Устанавливает свойству Value переключателя OptionButton1 значение True.
TextBox1 TextBox2 TextBox3 TextBox4 TextBox5 TextBox6	Font	Тип шрифта — Tahoma. Начертание — обычный. Размер — 12.	Нет
Label1	Caption Font	Начальная стоимость. Тип шрифта — Tahoma. Начертание — полужирный. Размер — 12	Нет

Продолжение табл. 8.2.

Элемент	Свойство	Значение свойства	Обрабатываемое событие
Label2	Caption Font	Остаточная стоимость. Тип шрифта — Tahoma. Начертание — полужирный. Размер — 12.	Нет
Label3	Caption Font	Время полной амортизации. Тип шрифта — Tahoma. Начертание — полужирный Размер — 12	Нет
Label4	Caption Font	Период расчета Тип шрифта –Tahoma Начертание — полужирный. Размер — 12	Нет
Label5	Caption Font	Величина амортизации. Тип шрифта — Tahoma. Начертание — полужирный. Размер — 12.	Нет
Label6	Caption Font	Кратность. Тип шрифта — Tahoma. Начертание — полужирный. Размер — 12.	Нет
Frame1	Caption Font	Тип амортизации. Тип шрифта — Tahoma. Начертание — полужирный. Размер — 12.	Нет
Option-Button1	Caption Font	Стандартный. Тип шрифта — Tahoma. Начертание — обычный. Размер — 8.	Нет
Option-Button2	Caption Font	Кратный учет. Тип шрифта — Tahoma. Начертание — обычный. Размер — 8.	Нет

Окончание табл. 8.2.

Элемент	Свойство	Значение свойства	Обрабатываемое событие
Command-Button1	Caption Font	Вычислить. Тип шрифта — Tahoma. Начертание — полужирный. Размер — 12.	Click: 1. Проверяет, есть ли данные для расчета. Если отсутствуют, то выдается сообщение оператору и выход из процедуры. 2. Проверяет: больше ли начальная стоимость конечной. Если нет, то выдает сообщение оператору и выход из процедуры. 3. Проверяет: больше ли период амортизации номера текущего периода. Если нет, то выдает сообщение оператору и выход из процедуры. 4. Если установлен признак простого расчета, то выполняет расчет по стандартному методу, в противном случае — по методу кратного учета. Результат помещается в поле с именем TextBox5.
Command-Button2	Caption Font	Выход. Тип шрифта — Tahoma. Начертание — полужирный. Размер — 12.	Click: Завершает выполнение программы.

Разработка процедур обработки событий. Процедура обработки события формы Initialize. Выполните действия:

1. Щелчком мыши маркируйте форму.
2. Выполните команду меню Вид ➤ Программа. Откроется окно редактирования кода. В списке событий окна редактирования выберите событие Initialize. В окне редактирования кода появится заготовка процедуры обработки этого события формы.

3. Введите следующий код:

```
Private Sub UserForm_Initialize()
    'Очищаем поля
  TextBox1.Text = «»
  TextBox2.Text = «»
  TextBox3.Text = «»
  TextBox4.Text = «»
  TextBox5.Text = «»
  TextBox6.Text = «»
    'Устанавливаем переключатель
  OptionButton1.Value = True
End Sub
```

4. Раскройте список объектов окна редактирования формы и щелчком выберите в нем элемент CommandButton1.
5. В списке событий окна редактирования выберите событие Click. В окне редактирования кода появится заготовка процедуры обработки этого события для выбранного элемента.
6. Введите в процедуру следующий код:

```
Private Sub CommandButton1_Click()
    'Проверяем, есть ли данные для расчета
  If TextBox1.Text = "" Or TextBox2.Text = "" _
    Or TextBox3.Text = "" Or TextBox4.Text = "" Then
    MsgBox "Нет данных для расчета", _
    vbExclamation, "Амортизация"
    Exit Sub
  End If
    'Преобразуем данные к числовому формату
  dblПервичнаяСтоимость = CDbl(TextBox1.Text)
  dblОстаточнаяСтоимость = CDbl(TextBox2.Text)
  intВремяАмортизации = CInt(TextBox3.Text)
  intПериодРасчета = CInt(TextBox4.Text)
    'Проверка корректности данных
  If dblПервичнаяСтоимость < dblОстаточнаяСтоимость Then
    MsgBox "Ошибка! Остаток больше начальной стоимости",
      vbExclamation, "Амортизация"
    TextBox1.SetFocus
    Exit Sub
  End If
    'Проверка корректности данных
  If intВремяАмортизации < intПериодРасчета Then
    MsgBox "Ошибка в сроке амортизации", _
```



```
      vbExclamation, "Амортизация"
    TextBox3.SetFocus
    Exit Sub
  End If
    'По какому методу производить расчет?
  If OptionButton1.Value = True Then
    blnПризнак = True
  Else
    blnПризнак = False
  End If
  If blnПризнак = True Then
    dblВеличинаАмортизации = SYD(dblПервичнаяСтоимость, _
    dblОстаточнаяСтоимость, intВремяАмортизации, _
    intПериодРасчета)
  Else
    TextBox6.SetFocus
    intКратность = CInt(TextBox6.Text)
    dblВеличинаАмортизации = DDB(dblПервичнаяСтоимость, _
    dblОстаточнаяСтоимость, intВремяАмортизации, _
    intПериодРасчета, intКратность)
  End If
  TextBox5.Text = CStr(dblВеличинаАмортизации)
End Sub
```

7. Для процедуры обработки события Click второй кнопки запишите код

```
Private Sub CommandButton2_Click()
  End
End Sub
```

8. Раскройте список объектов окна редактирования кода и выберите в нем строку Общая область. Курсор переместится в общую область модуля формы.
9. Введите в общую область следующий код:

```
Option Explicit
    'Декларации переменных
  Dim dblПервичнаяСтоимость As Double
  Dim dblОстаточнаяСтоимость As Double
  Dim intВремяАмортизации As Integer
  Dim intПериодРасчета As Integer
  Dim intКратность As Integer
  Dim blnПризнак As Boolean
  Dim dblВеличинаАмортизации As Double
```

10. Запустите программу на компиляцию, устраните синтаксические ошибки.
11. Проверьте правильность выполняемых расчетов.
12. Сохраните проект.

8.1.3. Элементы Список, Счетчик, Выключатель

Упражнение 14

Работу с элементами управления Список, Счетчик и Выключатель рассмотрим на примере программы для расчетов ипотечных кредитов.

Постановка задачи. Известны: цена приобретаемого имущества; величина первоначального взноса, выраженная в процентах; годовая ставка (в процентах); срок погашения ссуды (в месяцах или в количестве лет).

Требуется разработать программу с диалоговым окном, вычисляющую: величину периодических выплат; общую сумму выплат; общую сумму комиссионных и величину начального взноса.

Метод. Для расчета величины постоянных периодических выплат в библиотеке Visual Basic есть специальная функция `Pmt` (аналог функции Excel `ПЛАТ`). Ее синтаксис следующий:

```
Pmt(Ставка, Кпер, Нз, [Бз [, Тип]])
```

где `Ставка` — процентная ставка за период; `Кпер` — количество периодов выплат для погашения кредита; `Нз` — сумма, которую нужно погасить; `Бз` — баланс наличности, который нужно достичь после последней выплаты; `Тип` — признак того, когда производится выплата (0 — в конце периода, 1 — в начале периода платежа).

Разработка формы пользователя. В диалоговом окне должны быть предусмотрены элементы для ввода исходных данных и для вывода результатов. Для ввода данных используем 3 элемента Поле (для ввода цены имущества, величины первого взноса, и количества периодов выплаты). Для удобства установки количества периодов выплаты поместим в форму элемент Счетчик. Для указания типа периодов (месяцы или годы) используем 2 элемента Переключатель. Для указания, когда производится платеж (в начале периода или в конце) используем элемент Выключатель. Для ввода годовой процентной ставки применим элемент Список. Для вывода рассчитанных величин используем элементы Надпись. Для управления расчетом поместим элемент Кнопка. Вторую кнопку используем для выхода из программы. После размещения в форме всех элементов управления и установки свойств Caption

для элементов управления и самой формы она будет выглядеть примерно так, как изображено на рисунке 8.3.

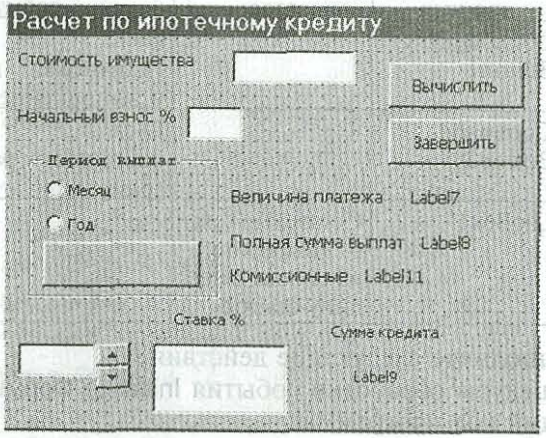

Рис. 8.3. Форма пользователя для расчета ипотеки

Для определения свойств элементов управления и обрабатываемых событий составим таблицу.

Таблица 8.3.

Элемент	Присваиваемое имя	Значение свойства	Обрабатываемое событие
UserForm	Ipoteca	Caption — Расчет по ипотечному кредиту	Initialize: 1. Установка границ элемента Счетчик. 2. Определение значений для элемента Список. 3. Удаление текста из надписей, в которые выводится результат.
TextBox1	TextBox1		Change: 1. Записывает свое значение в элемент Счетчик.
Toggle-Button1	ToggleButton1	ControlTipText — «Щелкните, чтобы переключить»	Click: 2. Устанавливает значение признака (1 — если платеж делается в начале периода и 0 — если в конце).
Spin-Button1	SpinButton1		Change: 3. Записывает свое значение в элемент TextBox1.

Окончание табл. 8.3.

Элемент	Присваиваемое имя	Значение свойства	Обрабатываемое событие
CommandButton1	CommandButton1		Click: Подготовка переменных и выполнение расчетов.
CommandButton2	CommandButton2		Click: Завершает работу программы.
ListBox1	ListBox1	ControlTipText — «Выберите из списка величину ставки».	

Далее выполните следующие действия:

1. Для процедуры обработки события Initialize Формы запишите следующий код программы:

```
Private Sub UserForm_Initialize()
    'Назначение значений элементу Список
    ListBox1.List = Array(3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)
    'Установка минимального и максимального значения счетчику
    SpinButton1.Min = 0
    SpinButton1.Max = 1000
    OptionButton1.Value = True
    Label1.Caption = "Месяцев"
    ToggleButton1.Caption = "В начале периода"
    Label7.Caption = ""
    Label8.Caption = ""
    Label9.Caption = ""
    Label11.Caption = ""
End Sub
```

2. Для процедуры обработки события Change элемента TextBox1 запишите следующий код:

```
Private Sub TextBox1_Change()
    If TextBox1.Text = "" Then Exit Sub
    SpinButton1.Value = CInt(TextBox1.Text)
End Sub
```

3. Для процедуры обработки события Change элемента Счетчик запишите код:

```
Private Sub SpinButton1_Change()
    TextBox1.Value = SpinButton1.Value
End Sub
```


4. Для процедуры обработки события Click элемента Кнопка запишите код:

```
Private Sub CommandButton1_Click()
    Dim Ставка As Double
    Dim Кпер As Integer
    Dim Тип As Byte
    Dim Нз As Double
    Нз = TextBox2.Value - TextBox2.Value * TextBox3.Value / 100
    Label9.Caption = Нз
    Ставка = ListBox1.Value
    Ставка = Ставка / 100
    If OptionButton1.Value = True Then
        Ставка = Ставка / 12
    End If
    Кпер = TextBox1.Value
    If ToggleButton1.Value = True Then Тип = 1 Else Тип = 0
    Label7.Caption = Int(Pmt(Ставка, Кпер, -Нз, , Тип))
    Label8.Caption = Label7.Caption * Кпер
    Label11.Caption = Label8.Caption - Label9.Caption
End Sub
```

5. Для процедуры обработки события Click другого элемента Кнопка запишите код:

```
Private Sub CommandButton2_Click()
End
End Sub
```

6. На рабочем листе Лист1 поместите кнопку формы и назначьте для нее макрос для обработки события Щелкнуть. Запишите код:

```
Sub Кнопка1_Щелкнуть()
    'Показать форму
    Ipoteca.Show
End Sub
```

7. Перейдите на рабочий лист Лист1 и щелкните на размещенной в нем кнопке. На экране появится созданная форма (рис. 8.4).
8. Введите данные для расчета и проверьте, правильно ли выполняются вычисления. В случае необходимости отладьте программу.

Рис. 8.4. Вид диалогового окна с результатами расчета

Задание для самостоятельной работы

Разработайте программу с диалоговым окном для расчета величины постоянных периодических выплат при погашении кредита. Исходными данными для расчета являются:

- величина кредита;
- количество выплат за год;
- срок кредита (в количестве годов);
- годовая процентная ставка.

В программе предусмотрите проверку корректности вводимых данных и выдачу сообщения пользователю в случае некорректных данных.

9. Разработка программ для рабочего листа

Упражнение 15

Разработка программы для создания базы данных Страхование

Постановка задачи. Разработать программу для создания базы данных по учету страхователей и вида страхования с диалоговым окном.

В базе данных должны отражаться следующие сведения:

- фамилия и имя страхователя;
- адрес (город, улица, дом, квартира);
- пол страхователя;
- срок страхования;
- вид страхования.

Разработка формы пользователя. Форма пользователя с элементами управления для создания базы данных может иметь такой же вид, как изображенная на рисунке 9.1. Для удобства ввода данных, кроме элементов Поле в форме предусмотрены: элемент Счетчик — для установки срока страхования; элемент Поле со списком — для ввода вида страхования.

Рис. 9.1. Форма диалогового окна для программы создания базы данных

Выполните следующие действия:

1. Создайте форму пользователя и разместите в ней элементы управления.
2. Элементам управления присвойте имена и свойства в соответствии с таблицей, приведенной ниже (табл. 9.1).

Таблица 9.1.

Элементы формы

Элемент	Имя	Свойства	События
Форма	Страхование	По умолчанию	Initialize: 1. Формирует названия колонок таблицы. 2. Закрывает строку формул. 3. В строку имени приложения заносит текст. 4. Назначает кнопке ОК всплывающую подсказку и клавишу <Enter>, а кнопке Отмена — клавишу <Esc>. 5. Присваивает значения элементу Список. 6. Устанавливает переключатель ПолМуж.
Поле	ПолеФамилия	По умолчанию	Нет
Поле	ПолеИмя	По умолчанию	Нет
Поле	ПолеГород	По умолчанию	Нет
Поле	ПолеУлица	По умолчанию	Нет
Поле	ПолеДом	По умолчанию	Нет
Поле	ПолеСрок	По умолчанию	Change: Свойству Value элемента Счетчик присваивает значение свойства Value элемента ПолеСрок.
Счетчик	Счетчик	По умолчанию	Change: Свойству Value элемента ПолеСрок присваивает значение свойства Value элемента Счетчик.
Список	Список	По умолчанию	Нет
Переключатель	ПолМуж	По умолчанию	Нет

Окончание табл. 9.1.

Элемент	Имя	Свойства	События
Переключатель	ПолЖен	По умолчанию	Нет
Кнопка	КнопкаОК	Caption = OK	Click: Вычисляет номер первой пустой строки. Заносит данные в таблицу БД.
Кнопка	КнопкаОтмена	Caption =Отмена	Click: 1. Отменяет надпись в строке заголовка приложения. 2. Останавливает выполнение программы.

3. Рабочему листу с именем «Лист2» присвойте имя «База данных».
4. На рабочем листе с именем «Лист1» поместите элемент управления Кнопка панели элементов Visual Basic и поместите на ней надпись «Страхование». С помощью этой кнопки будет вызываться разрабатываемое диалоговое окно.
5. Для процедуры обработки события Click этой кнопки запишите код:

```
Private Sub CommandButton1_Click()
    'Инициализация формы Страхование
  Страхование.Show
End Sub
```

6. Для формирования заголовков столбцов базы данных создайте процедуру Sub пользователя, которая будет вызываться из процедуры обработки события формы Initialize. Для этого выполните команду меню Visual Basic Вставка ➤ Процедура и присвойте ей имя ЗаголовокРабочегоЛиста. Процедура выполняет следующие действия:
   - проверяет, заполнена ли первая строка — строка заголовков столбцов (по значению ячейки A1), если заполнена, то не выполняет никаких действий и завершает работу и передает управление в точку вызова;
   - если первая строка не заполнена, то в ячейки первой строки рабочего листа записывает названия граф таблицы базы данных, комментарии к ним, закрепляет первую строку и завершает работу, передавая управление в точку своего вызова.

7. В окне редактирования кода введите текст программы этой процедуры:

```
Private Sub ЗаголовокРабочегоЛиста()
  If Range("A1").Value = "Фамилия" Then
    Range("A2").Select
    Exit Sub
  End If
  ActiveSheet.Cells.Clear
  Range("A1:I1").Value = Array("Фамилия", "Имя", "Пол", "Город",
    "Улица", "Дом", "Квартира", "Вид страхования", "Срок")
  Range("2:2").Select
  ActiveWindow.FreezePanes = True
  Range("A2").Select
  Range("A1").AddComment
  Range("A1").Comment.Visible = False
  Range("A1").Comment.Text Text:="Фамилия клиента"
  Range("B1").AddComment
  Range("B1").Comment.Visible = False
  Range("B1").Comment.Text Text:="Имя клиента"
  Range("C1").AddComment
  Range("C1").Comment.Visible = False
  Range("C1").Comment.Text Text:="Пол клиента"
  Range("D1").AddComment
  Range("D1").Comment.Visible = False
  Range("D1").Comment.Text Text:="Адрес клиента"
End Sub
```

8. Декларируйте вспомогательные переменные, поместив в Общей области их объявление:

```
Dim Пол As String * 3
Dim НомерСтроки As Integer
```

9. В процедуру обработки события Initialize формы введите код:

```
Private Sub UserForm_Initialize()
    'Активизируем рабочий лист с именем "База данных"
  Worksheets("База данных").Activate
    'Вызываем процедуру формирования названий граф БД
ЗаголовокРабочегоЛиста
    'Изменяем текст в строке заголовка приложения
  Application.Caption = "Регистрация. База данных страхования"
    'Закрываем строку формул
  Application.DisplayFormulaBar = False
    'Назначаем кнопке ОК клавишу Enter и всплывающую подсказку
```



```
  With КнопкаОК
    .Default = True
    .ControlTipText = "Ввод данных в базу данных"
  End With
    'Назначаем кнопке Отмена клавишу Esc и всплывающую подсказку
  With КнопкаОтмена
    .Cancel = True
    .ControlTipText = "Кнопка отмены"
  End With
    'Присваиваем значения элементам Списка
  Список.List = Array("Жизни", "Убытков", "Имущества")
    'Устанавливаем переключатель
  ПолМуж.Value = True
End Sub
```

10. В процедуру обработки события Click кнопки ОК запишите код:

```
Private Sub КнопкаОК_Click()
    'Вычисляем номер первой незаполненной строки таблицы
  НомерСтроки = _Application.CountA(ActiveSheet.Columns(1)) + 1
  With Страхование
    If .ПолМуж.Value = True Then
      Пол = "Муж"
    Else
      Пол = "Жен"
    End If
  End With
    'Записываем в ячейки текущей строки значения полей формы
  With ActiveSheet
    .Cells(НомерСтроки, 1).Value = ПолеФамилия
    .Cells(НомерСтроки, 2).Value = ПолеИмя
    .Cells(НомерСтроки, 3).Value = Пол
    .Cells(НомерСтроки, 4).Value = ПолеГород
    .Cells(НомерСтроки, 5).Value = ПолеУлица
    .Cells(НомерСтроки, 6).Value = ПолеДом
    .Cells(НомерСтроки, 7).Value = ПолеКвартира
    .Cells(НомерСтроки, 8).Value = Список.Value
    .Cells(НомерСтроки, 9).Value = ПолеСрок
  End With
End Sub
```

11. Для процедуры обработки события Click кнопки Отмена запишите код:

```
Private Sub КнопкаОтмена_Click()
  'Восстанавливаем прежний текст в строке заголовка приложения
Application.Caption = Empty
  'Завершаем выполнение программы
End
End Sub
```

12. Для процедур обработки события Change элементов Счетчик и ПолеСрок введите следующий код программы:

```
Private Sub Счетчик_Change()
    'Записываем значение счетчика в элемент ПолеСрок
  With Страхование
    .ПолеСрок.Text = CStr(.Счетчик.Value)
  End With
End Sub

Private Sub ПолеСрок_Change()
    'Записываем значение элемента ПолеСрок в счетчик
  With Страхование
    .Счетчик.Value = CInt(.ПолеСрок.Text)
  End With
End Sub
```

13. Запустите программу на выполнение, щелкнув на кнопке, размещенной на рабочем листе. Проверьте работоспособность и, в случае необходимости, устраните ошибки в программе.
14. Сохраните проект.

Упражнение 16

Разработка программы для создания базы данных регистрации вкладов

Постановка задачи. Требуется разработать программу для создания базы данных Excel, управляемую с помощью кнопок, размещенных на рабочем листе, и использующую форму пользователя для ввода данных. База данных служит для хранения следующих сведений: фамилии вкладчика, суммы вклада, вида вклада, отделение банка и примечания.

Разработка формы пользователя. Возможный вариант формы пользователя приведен на рисунке 9.2. В ней для ввода типа вклада помещен элемент Поле со списком, а для указания отделения банка в одном контейнере помещены 3 элемента Переключатель.

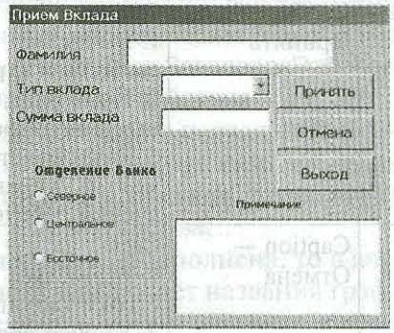

Рис. 9.2. Форма диалогового окна для программы регистрации вкладов

Выполните следующие действия:

1. Создайте форму пользователя и разместите в ней элементы управления, как показано на рисунке 9.2.
2. Элементам управления присвойте имена и свойства в соответствии с таблицей, приведенной ниже (табл. 9.2).

Таблица 9.2.

Элементы формы

Элемент	Имя	Свойства	События
Форма	UserForm		Initialize: 1. Устанавливает количество элементов в списка. 2. Присваивает значения элементам списка.
Форма	Вклад	Caption — Прием вклада	Initialize: 1. Изменяет текст в строке заголовка приложения. 2. Закрывает строку формул. 3. Устанавливает всплывающие подсказки. 4. Формирует заголовки столбцов таблицы, обращаясь к специально созданной процедуре.

Окончание табл. 9.2.

Элемент	Имя	Свойства	События
Кнопка	Принять	Caption — Принять	Click: 1. Вычисляет номер первой свободной строки. 2. Проверяет, введены ли данные в поля формы. 3. Записывает данные из элементов управления Формы в ячейки рабочего листа.
Кнопка	Отмена	Caption — Отмена	Click: 1. Вычисляет номер последней строки таблицы. 2. Удаляет содержимое из ячеек этой строки.
Кнопка	Выход	Caption — Выход	Click: Завершает выполнение программы.
Поле	Фамилия		
Поле	СуммаВклада		
Поле со списком	ТипВклада		
Переключатели	Северное Центральное Восточное		
Поле	Примечание		

3. Рабочему листу с именем «Лист2» присвойте имя «База», а рабочему листу«Лист1» — «Меню».
4. На рабочем листе с именем «Меню» поместите элемент управления Кнопка панели элементов Visual Basic и поместите на ней надпись «Прием вклада». С помощью этой кнопки будет вызываться разрабатываемое диалоговое окно.
5. Для процедуры обработки события Click этой кнопки запишите код:

```
Private Sub ПриемВклада_Click()
    'Вызываем процедуру формирования заголовков БД
    ЗаголовокРабочегоЛиста
End Sub
```


6. Для формирования заголовков столбцов базы данных создайте процедуру Sub пользователя, которая будет вызываться из процедуры обработки кнопки рабочего листа Прием вклада. Для этого выполните команду меню Visual Basic Вставка ➤ Процедура и присвойте ей имя ЗаголовокРабочегоЛиста. Процедура выполняет следующие действия:
   - проверяет, заполнена ли первая строка — строка заголовков столбцов (по значению ячейки A1), если заполнена, то не выполняет никаких действий и завершает работу, передавая управление в точку вызова;
   - если первая строка не заполнена, то в ячейки первой строки рабочего листа записывает названия граф таблицы базы данных, комментарии к ним, закрепляет первую строку и завершает работу, передавая управление в точку своего вызова.
7. В окне редактирования кода введите текст программы этой процедуры:

```
Private Sub ЗаголовокРабочегоЛиста()
    'Активизируем рабочий лист
  Application.Worksheets("База").Activate
    'Проверяем, есть ли названия столбцов БД
  With ActiveSheet
    If .Range("A1").Value = "Фамилия" Then
      .Range("A2").Select
    Else
      'Очищаем рабочий лист
      ActiveSheet.Cells.Clear
      'Записываем названия столбцов
      Range("A1:E1").Value = Array("Фамилия", "Тип", _
        "Сумма", "Отделение", "Примечание")
      'Фиксируем первую строку
      .Range("2:2").Select
      ActiveWindow.FreezePanes = True
      .Range("A2").Select
      'Вставляем комментарии
      .Range("A1").AddComment
      .Range("A1").Comment.Visible = False
      .Range("A1").Comment.Text Text:="Фамилия клиента"
      .Range("B1").AddComment
      .Range("B1").Comment.Visible = False
      .Range("B1").Comment.Text Text:="Тип вклада"
      .Range("C1").AddComment
```

```
        .Range("C1").Comment.Visible = False
        .Range("C1").Comment.Text Text:="Сумма вклада"
        .Range("D1").AddComment
        .Range("D1").Comment.Visible = False
        .Range("D1").Comment.Text Text:="Отделение банка"
      End If
    End With
    'Вызываем элемент Форма с именем Вклад
    Вклад.Show
  End Sub
```

8. В процедуру обработки события Initialize формы UserForm введите код:

```
Private Sub UserForm_Initialize()
  With Вклад
    .Северное.Value = True
      'Установим длину элемента Список
    .ТипВклада.ListRows = 3
      'Присвоим значения элементам списка
    .ТипВклада.List = Array("Срочный", "Депозит", "Текущий")
      'Установим фокус элементу Кнопка с именем Принять
    .Принять.SetFocus
  End With
End Sub
```

9. В процедуру обработки события Initialize формы с именем Вклад запишите код:

```
Private Sub Вклад_Initialize()
    'Изменим название в строке заголовка приложения
  Application.Caption = "Регистрация. База данных Банк"
    'Закрываем строку формул
  Application.DisplayFormulaBar = False
  With Принять
    .Default = True
    'Установка всплывающей подсказки
    .ControlTipText = "Ввод данных в базу данных"
  End With
  With Отмена
    .Cancel = True
    .ControlTipText = "Кнопка отмены"
  End With
  ЗаголовокРабочегоЛиста
End Sub
```


10. В процедуру обработки события Click элемента Кнопка с именем Принять введите код;

```
Private Sub Принять_Click()
  'Декларация переменных
  Dim Фамилия As String
  Dim ТипВклада As String
  Dim СуммаВклада As Double
  Dim Отделение As String
  Dim Примечание As String
  Dim НомерСтроки As Integer
    'Вычисление номера первой свободной строки
  НомерСтроки = _
    Application.CountA(ActiveSheet.Columns(1)) + 1
  With Вклад
    If .Фамилия.Text = "" Then
      MsgBox "Вы забыли указать фамилию", vbExclamation
      Exit Sub
    End If
    If .ТипВклада.Value = "" Then
      MsgBox "Вы забыли указать тип вклада", vbExclamation
      Exit Sub
    End If
    Фамилия = .Фамилия.Text
    ТипВклада = .ТипВклада.Value
    If .Северное.Value = True Then Отделение = "Северное"
    If .Центральное.Value = True Then Отделение = "Центральное"
    If .Восточное.Value = True Then Отделение = "Восточное"
    If IsNumeric(.СуммаВклада.Text) = False Then
      MsgBox "Введена неверная сумма", vbExclamation
      Exit Sub
    End If
    СуммаВклада = CDbl(.СуммаВклада.Text)
    Примечание = .Примечание.Text
  End With
    'Записываем данные в ячейки рабочего листа
  With ActiveSheet
    .Cells(НомерСтроки, 1).Value = Фамилия
    .Cells(НомерСтроки, 2).Value = ТипВклада
    .Cells(НомерСтроки, 3).Value = СуммаВклада
    .Cells(НомерСтроки, 4).Value = Отделение
    .Cells(НомерСтроки, 5).Value = Примечание
```

```
    End With
End Sub
```

11. В процедуру обработки события Click элемента Кнопка с именем Отмена поместите код:

```
Private Sub Отмена_Click()
  Dim НомерСтроки As Integer
    'Вычисляем номер последней строки
  НомерСтроки = Application.CountA(ActiveSheet.Columns(1))
    'Удаляем содержимое ячеек строки
  With ActiveSheet
    .Cells(НомерСтроки, 1).Value = ""
    .Cells(НомерСтроки, 2).Value = ""
    .Cells(НомерСтроки, 3).Value = ""
    .Cells(НомерСтроки, 4).Value = ""
    .Cells(НомерСтроки, 5).Value = ""
  End With
End Sub
```

12. В процедуру обработки события Click элемента Кнопка с именем Выход введите код:

```
Private Sub Выход_Click()
    'Активизируем рабочий лист с именем Меню
  Sheets("Меню").Activate
    'Завершаем выполнение программы
  End
End Sub
```

13. Перейдите в приложение Excel, активизируйте рабочий лист Меню и щелкните на кнопке Прием вклада. Программа должна активизироваться и на экране появится созданное диалоговое окно.
14. Отладьте и проверьте работу программы во всех режимах.
15. Сохраните проект.

Упражнение 17

Разработка программы для выполнения операций по вкладам

Постановка задачи. Разработать программу для выполнения операций по вкладам, сведения о которых содержатся в базе данных, создаваемой программой, разработанной в упражнении 16. В программе необходимо предусмотреть следующие функции:

- поиск нужной записи в базе данных на рабочем листе с именем «База»;

- выполнение операции по приему на счет и снятие со счета суммы денег;
- отмену операции по вкладу.

Разработка формы пользователя. Учитывая требования, изложенные в постановке задачи, возможный вариант формы пользователя может быть таким же, как изображенный на рисунке 9.3.

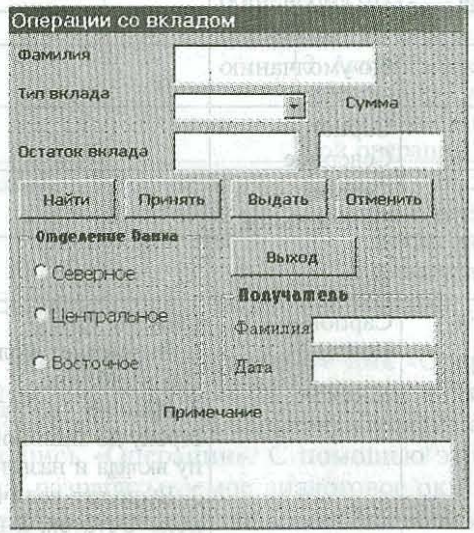

Рис. 9.3. Форма пользователя для программы Операции со вкладами

Выполните следующие действия:

1. Создайте форму пользователя и разместите в ней элементы управления как показано на рис. 9.3.
2. Элементам управления присвойте имена и свойства в соответствии с таблицей, приведенной ниже.

Таблица 9.3.

Элемент	Имя	Свойства	События
Форма	Опер	Caption — Операции со вкладом	Initialize: 1. Устанавливает один из переключателей. 2. Устанавливает количество элементов списка и присваивает ему значения. 3. Активизирует рабочий лист с именем «База».

Продолжение табл. 9.3.

Элемент	Имя	Свойства	События
Поле	Фам	По умолчанию	
Поле	Величина	По умолчанию	
Поле	Остаток	По умолчанию	
Поле	Получ	По умолчанию	
Поле	Дата	По умолчанию	
Поле со списком	Тип	По умолчанию	
Переключатель	Сев	Caption — Северное	
Переключатель	Центр	Caption — Центральное	
Переключатель	Вост	Caption — Восточное	
Кнопка	Найти	Caption — Найти	Click: 1. Вычисляет количество записей в таблице. 2. Находит в таблице нужную строку по заданной фамилии, типу вклада и названию отделения. 3. Выводит величину остатка в поле Остаток, а в поле Дата — текущую дату.
Кнопка	Принять	Caption — Принять	Click: 1. Проверяет корректность числовой информации в поле Величина. 2. Записывает в базу данных «База» новую вычисленную сумму. 3. Активизирует рабочий лист «Операции». 4. Записывает в базу данных «Операции» данные о проведенной операции.
Кнопка	Отменить	Caption — Отменить	Click: Восстанавливает в текущей строке таблицы «База» данные, которые были до операции.

Окончание табл. 9.3.

Элемент	Имя	Свойства	События
Кнопка	Выдать	Caption — Выдать	Click: 1. Проверяет корректность числовой информации в поле Величина. 2. Вычисляет остаток и записывает его в базу данных «База». 3. Активизирует рабочий лист «Операции» и записывает в таблицу сведения о проведенной операции.
Кнопка	Выход	Caption — Выход	Click: 1. Активизирует рабочий лист с именем «Меню». 2. Завершает выполнение программы.

3. Рабочему листу «Лист3» присвойте имя «Операции».
4. На рабочем листе с именем «Меню» поместите элемент управления Кнопка панели элементов Visual Basic и поместите на ней надпись «Операции». С помощью этой кнопки будет вызываться разрабатываемое диалоговое окно.
5. Для процедуры обработки события Click этой кнопки в модуле рабочего листа запишите код:

```
Private Sub Операции_Click()
    'активизируем форму с именем Опер
  Опер.Show
End Sub
```

6. На рабочем листе «Операции» в первой строке напишите заголовки колонок таблицы в следующей последовательности: «Фамилия владельца вклада», «Дата операции», «Фамилия получателя», «Тип вклада», «Выдано», «Принято», «Отделение», «Примечание».
7. В программном модуле формы в общей области запишите текст объявления переменных:

```
Option Explicit
Dim Фамилия As String
Dim ТипВклада As String
Dim СуммаВклада As Double
Dim Отделение As String
Dim Примечание As String
```

```
Dim Количество As Integer
Dim Номер As Integer
```

8. Для процедуры обработки события UserForm_Initialize введите код:

```
Private Sub UserForm_Initialize()
  With Опер
      'Устанавливаем переключатель
    .Сев.Value = True
      'Количество элементов в Поле со списком
    .Тип.ListRows = 3
      'Значения элементов списка
    .Тип.List = Array("Срочный", "Депозит", "Текущий")
      'Устанавливаем фокус кнопке Найти
    .Найти.SetFocus
      'Активизируем рабочий лист "База"
    Application.Worksheets("База").Activate
  End With
End Sub
```

9. В процедуру обработки события Click кнопки Найти введите код:

```
Private Sub Найти_Click()
  'Активизируем рабочий лист База
 Worksheets("База").Activate
   'Вычисляем количество заполненных строк
 Количество = Application.CountA(ActiveSheet.Columns(1))
 Фамилия = Фам.Text
 ТипВклада = Тип.Value
 If Сев.Value = True Then Отделение = "Северное"
 If Центр.Value = True Then Отделение = "Центральное"
 If Вост.Value = True Then Отделение = "Восточное"
 With ActiveSheet
   For Номер = 1 To Количество
   'Поиск нужной записи по заданным параметрам
   If .Cells(Номер, 1) = Фамилия And .Cells(Номер, 2) = _
   ТипВклада And .Cells(Номер, 4) = Отделение Then Exit For
   Next
     'Отображаем остаток вклада в поле Остаток
   Остаток.Text = .Cells(Номер, 3)
 End With
 If Номер = Количество + 1 Then
   MsgBox "Такого счета в базе нет", vbInformation
 End If
   'Отображаем текущую дату в поле Дата
```



```
   Дата.Text = Date
End Sub
```

10. В Процедуру обработки события Click кнопки Принять запишите код:

```
Private Sub Принять_Click()
  'проверяем корректность данных
  If IsNumeric(Величина.Text) = False Then
    MsgBox "Ошибка в поле Суммы", vbInformation, "БАНК"
    Exit Sub
  End If
  With ActiveSheet
    'записываем в ячейку новое значение остатка
   .Cells(Номер, 3) = CStr(CDbl(.Cells(Номер, 3)) + _
   CDbl(Величина.Text))
  End With
    'Активизируем рабочий лист Операция
  Worksheets("Операции").Activate
   'Вычисляем номер первой пустой строки
  Количество = Application.CountA(ActiveSheet.Columns(1)) + 1
  With ActiveSheet
     'Записываем в ячейки данные о проведенной операции
    .Cells(Количество, 1) = Получ.Text
    .Cells(Количество, 2) = Дата.Text
    .Cells(Количество, 3) = Фам.Text
    .Cells(Количество, 4) = Тип.Text
    .Cells(Количество, 6) = Величина.Text
    .Cells(Количество, 7) = Отделение
    .Cells(Количество, 8) = Прим.Text
  End With
End Sub
```

11. В процедуру обработки события Click кнопки Выдать запишите код:

```
Private Sub Выдать_Click()
  'Проверяем корректность введенного числа
 If IsNumeric(Величина.Text) = False Then
   MsgBox "Ошибка в поле Суммы", vbInformation, "БАНК"
   Exit Sub
 End If
 With ActiveSheet
   'Записываем в ячейку новый остаток
  .Cells(Номер, 3) = CStr(CDbl(.Cells(Номер, 3)) - _
```

```
        CDbl(Величина.Text))
    End With
    ' Активизируем рабочий лист "Операции"
    Worksheets("Операции").Activate
    ' Вычисляем номер первой свободной строки
    Количество = Application.CountA(ActiveSheet.Columns(1)) + 1
    With ActiveSheet
      ' Записываем данные о проведенной операции
      .Cells(Количество, 1) = Получ.Text
      .Cells(Количество, 2) = Дата.Text
      .Cells(Количество, 3) = Фам.Text
      .Cells(Количество, 4) = Тип.Text
      .Cells(Количество, 5) = Величина.Text
      .Cells(Количество, 7) = Отделение.Text
      .Cells(Количество, 8) = Прим.Text
    End With
  End Sub
```

12. В процедуру обработки события Click кнопки Отменить введите код:

```
  Private Sub Отменить_Click()
    With ActiveSheet
      ' Восстанавливаем прежний остаток в БД База
      .Cells(Номер, 3) = Остаток.Text
    End With
    Worksheets("Операции").Activate
    ' Удаляем данные о проведенной операции
    With ActiveSheet
      .Cells(Количество, 1) = ""
      .Cells(Количество, 2) = ""
      .Cells(Количество, 3) = ""
      .Cells(Количество, 4) = ""
      .Cells(Количество, 6) = ""
      .Cells(Количество, 7) = ""
      .Cells(Количество, 8) = ""
    End With
  End Sub
```

13. В процедуру обработки события Click кнопки Выход введите код:

```
  Private Sub Выход_Click()
    ' Активизируем рабочий лист "Меню"
    Worksheets("Меню").Activate
```



```
    End
  End Sub
```

14. Перейдите в приложение Excel, активизируйте рабочий лист «Меню» и щелкните на кнопке Операции. Программа должна активизироваться и на экране появится созданное диалоговое окно.
15. Отладьте и проверьте работу программы во всех режимах.
16. Сохраните проект.

Задания для самостоятельной работы

Задание 1. Разработайте программу с диалоговым окном для создания на рабочем листе базы данных по учету выданных кредитов. Вид диалогового окна приведен на рисунке 9.4. Таблица базы данных должна содержать: фамилию, имя, адрес заемщика, срок кредита, сумму кредита, пол заемщика, сведения о залоге и его сумме.

Рис. 9.4. Форма программы для создания базы данных по учету кредитов

Задание 2. Разработайте программу с диалоговым окном для расчета величины постоянных периодических выплат при погашении кредита. Исходными данными для расчета являются:

- величина кредита;
- количество выплат за год;
- срок кредита (в количестве годов);
- годовая процентная ставка.

В программе предусмотрите проверку корректности вводимых данных и выдачу сообщения пользователю в случае некорректных данных.

Приложение

Встроенные функции Visual Basic

Таблица П.1.

Финансово-математические функции

Имя функции, параметры	Возвращаемое значение
DDB(Стоимость, Остаточная_стоимость, Время_эксплуатации, Период, Кратность)	Вычисляет амортизацию фондов в течение заданного интервала времени.
FV(Ставка, Кпер, Плата [, Нз[, Тип]])	Вычисляет накопленную стоимость при известном размере регулярного взноса и постоянной процентной ставке.
IPmt(Ставка, Период, Кпер, Нз[, Бз[, Тип]])	Вычисляет сумму выплат при известном размере регулярного взноса и постоянной процентной ставке.
IRR(Величина()[, guess])	Вычисляет внутреннюю норму доходности при известной последовательности выплат и поступлений.
MIRR(Величина(),Ставка_финанс, Ставка_реинвест)	Вычисляет модифицированную внутреннюю норму доходности.
NPer(Ставка, Платеж, Нз, Бз, Тип)	Вычисляет количество периодов времени, необходимых для достижения заданной фактической стоимости при постоянном размере выплат и постоянной процентной ставке.
NPV(Ставка, Величина())	Вычисляет чистую приведенную стоимость инвестиционного проекта при известном размере выплат и поступлений и при постоянной дисконтной ставке.
PPMT(Ставка, Период, Кпер, Нз, Бз, Тип)	Вычисляет величину постоянного взноса для достижения определенной суммы при постоянной процентной ставке.
Pmt(Ставка, Кпер, Нз[, Бз[, Тип]])	Аналогична PPMT, но позволяет вычислить величину выплаты в зависимости от того, когда она производится (в начале или в конце периода).

Окончание табл. П.1.

Имя функции, параметры	Возвращаемое значение
PV(Ставка, Кпер, Плата[, Бз[, Тип]])	Вычисляет приведенную стоимость при известном и постоянном размере выплаты, периоде и постоянной процентной ставке.
Rate(Кпер, Плата, Нз[, Бз[, Тип]])	Вычисляет процентную ставку, необходимую для достижения заданной стоимости при известном периоде выплат.
SLN(Стоимость, Ликвидная_стоимость, Жизнь)	Вычисляет величину амортизации фондов линейным методом.
SYD(Стоимость, Ликвидная_стоимость, Жизнь, Период)	Вычисляет величину годовой амортизации фондов за определенный период.

Таблица П.2.

Функции преобразования типов

Имя функции, параметры	Возвращаемое значение
Asc(СтрокаСимволов)	Возвращает код первого символа строки.
CBool(Выражение)	Преобразует числовое выражение или строку в булево значение: True, если значение выражения отлично от нуля, или False если значение выражения равно нулю.
CByte(Выражение)	Преобразует числовое выражение или строку в байт. Аргумент может принимать значения в диапазоне 0—255
CCur (Выражение)	Преобразует числовое выражение или строку в число денежного формата.
CDate(Выражение)	Преобразует числовое выражение или строку в дату.
lparCDbl(Выражение)	Переводит числовое выражение или строку в число типа Double.
CDec(Выражение)	Преобразует числовое выражение или строку в число типа Decimal.
Cint(Выражение)	Преобразует числовое выражение или строку в число типа Integer.
CLng(Выражение)	Преобразует числовое выражение или строку в число типа Long.
CSng(Выражение)	Преобразует числовое выражение или строку в число типа Single.

Окончание табл. П.2.

Имя функции, параметры	Возвращаемое значение
CStr(Выражение)	Преобразует числовое выражение или строку в строку.
Cvar(Выражение)	Преобразует числовое выражение или строку в число типа Variant.
Hex(Число)	Преобразует числовое выражение или строку в шестнадцатеричное число.
Oct(Число)	Преобразует числовое выражение или строку в восьмеричное число.
Val(Строка)	Преобразует строку цифровых символов в число.

Таблица П.3.

Математические функции

Имя функции, параметры	Возвращаемое значение
Abs(ЧисловоеВыражение)	Абсолютное значение числа.
Atn(ЧисловоеВыражение)	Арктангенс от значения параметра, заданного в радианах.
Cos(ЧисловоеВыражение)	Косинус указанного в радианах угла.
Exp(Числовое_выражение)	Возвращает число *e*, возведенное в указанную степень, где *e* — основание натурального логарифма.
Fix(ЧисловоеВыражение)	Возвращает результат округления выражения с плавающей точкой до целой части. Для положительных значений аргумента возвращает ближайшее меньшее число, а для отрицательных — ближайшее большее.
Int(ЧисловоеВыражение)	Возвращает результат округления выражения с плавающей точкой до целой части. В случае отрицательного параметра возвращает ближайшее меньшее отрицательное число.
Log(ЧисловоеВыражение)	Возвращает натуральный логарифм от значения числового выражения.

Окончание табл. П.3.

Имя функции, параметры	Возвращаемое значение
Rnd[(Число]	Возвращает псевдослучайное число одинарной точности в интервале от 0 до 1. Необязательный параметр, устанавливает то, как генерируется следующее псевдослучайное число.
Sgn(ЧисловоеВыражение)	Возвращает +1, если значение параметра положительно, -1, если отрицательно, и 0, если 0.
Sin(ЧисловоеВыражение)	Возвращает синус угла от значения параметра, заданного в радианах.
Sqr(ЧисловоеВыражение)	Возвращает квадратный корень числового выражения.
Tan(ЧисловоеВыражение)	Возвращает тангенс угла от значения параметра, заданного в радианах.

Таблица П.4.

Функции статуса

Имя функции, параметры	Возвращаемое значение
IsArray(ИмяПеременной)	Возвращает булево значение, указывающее статус параметра: True, если переменная типа Variant является массивом, и False в противном случае.
IsDate(Выражение)	Возвращает булево значение, указывающее статус параметра: True, если значение параметра является датой, и False в противном случае.
IsEmpty(Выражение)	Возвращает булево значение, указывающее статус параметра типа Variant: True, если переменная инициализирована.
IsError(Выражение)	Возвращает булево значение, указывающее статус параметра: True, если значение параметра является ошибочным.
IsMissing(ИмяПараметра)	Возвращает булево значение, указывающее статус параметра. Если необязательный параметр передан в процедуру, то True, False — в противном случае.
IsNull(Выражение)	Возвращает булево значение, указывающее статус параметра. Если выражение содержит действительные данные, то True, False — в противном случае.

Окончание табл. П.4.

Имя функции, параметры	Возвращаемое значение
IsNumeric(Выражение)	Возвращает булево значение True, если параметр является числом, False — в противном случае.
IsObject(Ссылка)	Возвращает булево значение True, если параметр относится к типу Object, False — в противном случае.
TypeName(ИмяПеременной)	Возвращаемая строка содержит информацию о переменной-параметре.
VarType(ИмяПеременной)	Возвращает значение типа Integer, указывающее на разновидность значения, содержащегося в Variant-переменной, переданной в качестве параметра.

Таблица П.5.

Функции обработки строк

Имя функции, параметры	Возвращаемое значение
Asc(СтроковоеВыражение)	Возвращает ASCII код первого символа в строковом выражении.
Chr(asciiКодСимвола)	Возвращает символ, соответствующий указанному коду ASCII.
Instr([НачальнаяПозиция, СтроковоеВыражение1, СтроковоеВыражение2[0/1])	Возвращает номер позиции первого обнаружения Строки2 в Строке1. НачальнаяПозиция — устанавливает позицию строки, с которой начинается поиск (если параметр опущен, то поиск начинается с позиции первого символа).
Lcase(СтроковоеВыражение)	Возвращает строку, подобную строке параметра, но состоящую из строчных букв.
Left(СтроковоеВыражение, КоличествоСимволов)	Возвращает строку, извлеченную из строки-параметра, начиная от начала строки, и содержащую заданное количество символов.
Len(СтроковоеВыражение/ ИмяПеременной)	Возвращает количество символов в строке или количество байт, необходимых для хранения переменной.
LTrim(СтроковоеВыражение)	Удаляет начальные пробелы из строки-параметра.

Окончание табл. П.5.

Имя функции, параметры	Возвращаемое значение
Mid(СтроковоеВыражение, НачальноеЗначение [,Длина])	Возвращает часть строки (подстроку) начиная от символа с позиции НачальноеЗначение и содержащую количество символов, равное значению параметра Длина.
Right(СтроковоеВыражение, Длина)	Возвращает строку с указанным в параметре Длина количеством символов, начиная от конца строки, заданной параметром СтроковоеВыражение.
Rtrim(СтроковоеВыражение)	Возвращает строку идентичную строке-параметру, но с удаленными конечными пробелами.
Space(Длина)	Возвращает строку пробелов, указанной длины.
Str(ЧисловоеВыражение)	Возвращает строковое представление числа.
StrComp(Строка1, Строка2, РежимСравнения)	Возвращает результат сравнения строк в виде числа.
StrConv(Строка, Преобразование)	Возвращает преобразованную строку. Вид преобразования зависит от значения параметра Преобразование.
String(Длина, asciiКод СтроковоеВыражение)	Возвращает строку указанной длины, состоящую из повторяющегося символа, заданного переменной asciiКод или первым символом из параметра СтроковоеВыражение.
Trim(СтроковоеВыражение)	Возвращает строку с удаленными начальными и конечными пробелами в строке-параметре.
Val(СтроковоеВыражение)	Переводит строковое представление числа в число.

Таблица П.6.

Функции даты и времени

Имя функции, параметры	Возвращаемое значение
Date	Возвращает или устанавливает текущую системную дату в формате ДД-ММ-ГГ.
DateAdd(Интервал, Количество, Дата)	Возвращает дату, полученную увеличением заданной на количество интервалов.

Окончание табл. П.6.

Имя функции, параметры	Возвращаемое значение
DateDiff(Интервал, НачальнаяДата, КонечнаяДата [,ПервыйДеньНедели [,ПерваяНеделяГода]])	Возвращает количество интервалов времени между двумя датами типа Long. Интервал — единица измерения интервала: Yyyy — год; Q — квартал; M — месяц;
	Y — день года; W — день недели; WW — неделя; H — час; N — минута; S — секунда.
DatePart(Интервал, Дата [,ПервыйДеньНедели [,ПерваяНеделяГода]])	Возвращает заданную часть даты.
DateSerial(Год, Месяц, День)	Возвращает дату дня, соответствующую заданным параметрам.
DateValue(Дата)	Возвращает дату, содержащуюся в строке-параметре.
Day(Дата)	Возвращает число от 1 до 31, соответствующее числу месяца в параметре Дата.
Hour(Время)	Возвращает число в диапазоне 0–23, соответствующее часу.
Minute(Время)	Возвращает число в диапазоне 0–59, соответствующее минуте.
Month(Дата)	Возвращает число в диапазоне 1–12, соответствующее месяцу.
Now	Возвращает системную дату и время в формате ДД.ММ.ГГ ЧЧ:ММ:СС.
Second(Время)	Возвращает число в диапазоне 1–59, соответствующее секунде.
Time	Возвращает текущее системное время в формате ЧЧ:ММ:СС.
Timer	Возвращает количество секунд, прошедших с полуночи до текущего момента.
Weekday(Дата[,ПервыйДеньНедели])	Возвращает номер дня недели, соответствующий переданной в параметре дате.
Year(Дата)	Возвращает число, соответствующее году переданной в параметре даты.

Таблица П.7.

Функции для работы с массивами

Имя функции, параметры	Возвращаемое значение
Array(СписокАргументов)	Создает массив типа Variant. Список аргументов — список значений, присваиваемых элементам массива.
IsArray(ИмяПеременной)	Возвращает True, если переменная является массивом, False – в противном случае.
LBound(ИмяМассива [,РазмерностьМассива])	Возвращает минимальное допустимое значение указанной размерности.
UBound(ИмяМассива [,РазмерностьМассива])	Возвращает максимальное допустимое значение указанной размерности.
Erase Список массивов	Повторно инициализирует элементы массивов фиксированной длины и освобождает память, отведенную для динамического массива.

Таблица П.8.

Функции для работы с файлами

Имя функции, параметры	Возвращаемое значение
EOF(НомерФайла)	Проверка окончания открытого файла с указанным номером.
INPUT(n, НомерФайла)	Возвращает строку символов, считанных из указанного файла: n — количество символов (байтов) для чтения, НомерФайла — номер открытого файла.
FILEATTR(НомерФайла, Атрибут)	Возвращает информацию об открытом файле. НомерФайла — номер открытого файла, Атрибут указывает тип возвращаемой информации. Если Атрибут равен 1, то возвращается значение, указывающее режим доступа к файлу. Если Атрибут равен 2 — возвращается указатель файла.
FREEFILE	Возвращает номер следующего доступного неиспользуемого файла.

Окончание табл. П.8.

Имя функции, параметры	Возвращаемое значение
LOC(НомерФайла)	Возвращает текущую позицию в файле. НомерФайла — номер открытого файла. Для бинарных файлов возвращается позиция последнего считанного или записанного байта. Для файлов прямого доступа возвращается номер последней записанной или считанной записи. Для последовательных файлов возвращается позиция текущего байта в файле.
LQF(НомерФайла)	Возвращает длину файла в байтах.
SEEK(НомерФайла)	Возвращает текущую позицию файла.

Таблица П.9.

Прочие функции

Имя функции, параметры	Возвращаемое значение
MsgBox(СтрокаСообщения, Опция, Заголовок, Путь_к_справочному файлу, Константа контекстной ссылки)	Отображает текстовую строку в окне сообщений.
SPC(Число)	Оставляет заданное количество пробелов в операторах PRINT.
TAB(Столбец)	Перемещает курсор в указанную позицию печати.
VARPTR(ИмяПеременной)	Возвращает адрес смещения переменной.

Литература

1. Гарнаев А. Использование MS Excel и VBA в экономике и финансах. — СПб.: БХВ – Санкт-Петербург, 2000.
2. Кутуков Б.В. Основы финансовой и страховой математики. — М.: Дело, 1998.
3. Линке Маркус. Visual Basic 5. Справочник. — М.: ЗАО «Издательство БИНОМ», 1998.
4. Назаров С.В., Мельников П.П. Программирование на MS Visual Basic. — М.: Финансы и статистика, 2001.
5. Бадд Тимоти. Объектно-ориентированное программирование. — СПб.: Питер, 1997.

Учебное издание

Мельников Петр Петрович
Миронова Ирина Васильевна,
Шполянская Ирина Юрьевна

Практикум по экономической информатике

В трех частях
Часть III

Ответственный за выпуск *Л. А. Табакова*
Редактор *Г. В. Афанасьева*
Верстка *Л. С. Моджорян*

ИБ № 4371
Лицензия ЛР № 010156 от 29.01.97
Лицензия ЛР № 060865 от 24.03.97

Сдано в набор 2.07.01. Подписано в печать 2.08.01
Бумага офсетная № 1. Формат 60×90 $^{1}/_{16}$. Гарнитура «Таймс»
Усл. печ. л. 9,7. Уч.-изд. л. 10,7
Тираж 10 000 экз. Заказ № 1624

Издательство «Финансы и статистика»
101000, г. Москва, ул. Покровка, 7
Телефон (095) 925-35-02, факс (095) 925-09-57
E-mail: mail@finstat.ru *http:* //www.finstat.ru

Издательство «Перспектива»
107061, г. Москва, ул. 9-я Рота, 15
Тел.: (095) 963-25-36, 965-56-58
E-mail: books-id@mtu-net.ru

Отпечатано с готовых диапозитивов в ФГУП ордена «Знак Почета»
Смоленской областной типографии им. В. И. Смирнова.
214000, г. Смоленск, проспект им. Ю. Гагарина, 2.
Тел.: 3-01-60; 3-46-20; 3-46-05

ISBN 5-279-02474-0
9 785279 024742