

УДК 681.3.068+800.92 Visual Basic
ББК 32.973.26-018.1
К90

Культин Н. Б.

К90 Visual Basic. Освой на примерах. — СПб.: БХВ-Петербург,
2004. - 288 с: ил.

ISBN 5-94157-521-1

Рассмотрены примеры на языке Visual Basic — от простейших до приложений работы с графикой, мультимедиа и базами данных — которые демонстрируют назначение компонентов и раскрывают тонкости процесса программирования.

Справочник содержит описания базовых компонентов и наиболее часто используемых функций. На прилагаемом компакт-диске находятся исходные тексты программ.

Для начинающих программистов

УДК 681.3.068+800.92 Visual Basic

ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор *Екатерина Кондукова*

Зам. главного редактора *Игорь Шишигин*

Зав. редакцией *Григорий Добин*

Редактор *Владимир Красильников*

Компьютерная верстка *Натальи Караваевой*

Корректор *Виктория Пиотровская*

Дизайн обложки *Игоря Цырульникова*

Зав. производством *Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать **25.06.04.**

Формат 60x90^{1/16}. Печать офсетная. Усл. печ. л. 18.

Тираж 5000 экз. Заказ № 343

"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар № 77.99.02.953.Д.001537.03.02 от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов

в ГУП "Типография "Наука"

199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-521-1 Культин Н.Б.

Оформление издательство "БХВ-Петербург", 2004

Содержание

Предисловие	1
ЧАСТЬ I. ПРИМЕРЫ.....	3
Глава 1. Базовые компоненты.....	5
Общие замечания	5
Глава 2. Графика.....	55
Общие замечания	55
Глава 3. Мультимедиа.....	99
Общие замечания	99
Глава 4. Файлы.....	129
Общие замечания	129
Глава 5. Игры и полезные программы.....	141
Глава 6. Базы данных.....	207
Общие замечания	207
ЧАСТЬ II. КРАТКИЙ СПРАВОЧНИК.....	227
Глава 7. Компоненты.....	229
Форма.....	229
Label.....	231
TextBox	233
CommandButton.....	234

<i>CheckBox</i>	236
<i>OptionButton</i>	237
<i>ListBox</i>	238
<i>ComboBox</i>	239
<i>Timer</i>	241
<i>DriveListBox</i>	242
<i>DirListBox</i>	243
<i>FileListBox</i>	244
<i>PictureBox</i>	246
<i>Image</i>	248
<i>Shape</i>	249
<i>Line</i>	251
<i>UpDown</i>	252
<i>Common Dialog</i>	254
<i>MMControl</i>	255
Глава 8. Графика	257
<i>Print</i>	257
<i>Line</i>	258
<i>Circle</i>	259
<i>RGB</i>	260
Глава 9. Функции	264
Ввод и вывод.....	264
Математические функции.....	266
Преобразование данных.....	267
Работа со строками.....	268
Работа с файлами.....	273
Глава 10. События	278
Приложение. Содержание компакт-диска, прилагаемого к книге Культина Н. Б. "Visual Basic. Освой на примерах"	281
Предметный указатель	283

Предисловие

В последнее время резко возрос интерес к программированию. Это связано с развитием и внедрением в повседневную жизнь общества информационных технологий. Если человек имеет дело с компьютером, то рано или поздно у него возникает желание, а иногда и необходимость программировать.

Бурное развитие вычислительной техники, потребность в эффективных средствах разработки программного обеспечения, привели к появлению систем программирования, ориентированных на так называемую "быструю разработку". В основе идеологии систем быстрой разработки или *RAD-систем* (Rapid Application Development — среда быстрой разработки приложений) лежат технологии визуального проектирования и событийного объектно-ориентированного программирования. Суть этих технологий заключается в том, что среда разработки берет на себя большую часть рутинной работы по формированию программного кода, оставляя программисту решение задач по конструированию диалоговых окон и созданию функций обработки событий. В связи с этим, производительность процесса программирования, при использовании RAD-систем, фантастическая!

Среди RAD-систем особо выделяется среда Microsoft Visual Basic, которая позволяет создавать различные программы от простейших однооконных приложений, до программ управления базами

данных. В качестве языка программирования в среде Microsoft Visual Basic используется Visual Basic.

Чтобы научиться программировать, надо программировать — писать программы, решать конкретные задачи. Для этого необходимо изучить как язык программирования, так и среду разработки.

Освоить язык программирования Visual Basic не очень сложно. Труднее изучить среду разработки и научиться использовать ее компоненты. Хорошим подспорьем здесь могут быть програм-

мы, которые демонстрируют как назначение компонентов, так и особенности их использования.

В книге, которую вы держите в руках, собраны разнообразные примеры, которые не только демонстрируют возможности среды разработки Microsoft Visual Basic, но и знакомят с принципами организации обработки графической, звуковой информации, а также баз данных. Следует обратить внимание, что большинство примеров не являются учебными, в прямом смысле этого слова, и представляют собой вполне работоспособные программы.

Состоит книга из двух частей и приложения.

Первая часть содержит примеры. Примеры представлены в виде краткого описания, диалоговых окон и хорошо документированных текстов программ. Для простых задач приведены только функции обработки событий. Текст остальных программ приведен полностью.

Вторая часть книги — краткий справочник, в котором можно найти описание компонентов и функций Visual Basic.

Прилагаемый к книге компакт-диск содержит проекты, которые рассмотрены в ней в качестве примеров. Каждый проект находится в отдельном каталоге. Помимо файлов проекта, в каталоге находится исполняемый файл, что позволяет, без загрузки его в среду Microsoft Visual Basic, увидеть, как работает программа. Как уже упоминалось ранее, научиться программировать можно, только решая конкретные задачи. При этом достигнутые в этой области успехи в значительной степени зависят от опыта разработчика. Поэтому, чтобы получить от книги максимальную пользу, вы должны работать с ней активно. Изучайте листинги, старайтесь понять, как работают программы. Не бойтесь экспериментировать — вносите в программы изменения. Если что-то не понятно, обратитесь к справочнику в конце книги или к справочной системе Visual Basic.

Часть I

ПРИМЕРЫ

Глава 1

Базовые компоненты

Общие замечания

Процесс создания программы в Visual Basic состоит из двух шагов. Сначала нужно создать форму программы (диалоговое окно), затем написать процедуры обработки событий. Форма при-

ложения (так принято называть прикладные программы, работающие в Windows) создается путем добавления в форму компонентов и последующей их настройки.

В форме практически любого приложения есть компоненты, которые обеспечивают интерфейс (взаимодействие) между программой и пользователем. Такие компоненты называют базовыми. К базовым компонентам можно отнести следующие:

- Label — поле вывода текста;
- TextBox — поле ввода/редактирования текста;
- CommandButton — командная кнопка;
- CheckBox — независимая кнопка выбора;
- OptionButton — зависимая кнопка выбора;
- ListBox — список выбора;
- ComboBox — комбинированный список выбора.

Вид компонента, его размер и поведение определяют значения *свойств* (характеристик) компонента.

Основную работу в программе выполняют процедуры обработки *событий*.

Часть I. Примеры 6

Описание свойств базовых компонентов и основных событий можно найти в справочнике, который приводится во второй части книги.

Исходную информацию программа может получить из полей ввода/редактирования (компонент — TextBox), списка выбора (компонент — ListBox) или комбинированного списка (компонент — ComboBox). Для ввода значений логического типа можно ИСПОЛЬЗОВАТЬ компоненты CheckBox И OptionButton.

Конечные или промежуточные результаты программа может вывести в поле вывода текста (компонент — Label) или в окно сообщения (функция — MsgBox).

Для преобразования текста, находящегося, например, в поле ввода/редактирования, в число нужно использовать функцию Val.

Для преобразования числа, например значения переменной, в строку можно использовать функцию Format или CStr.

1. Программа пересчитывает скорость ветра из "метров в секунду" в "километры в час". Форма программы приведена на рис. 1.1.

Рис. 1.1. Форма программы **Скорость ветра**

```
' щелчок на кнопке Пересчет
Private Sub Command1_Click()
Dim ms As Integer ' скорость м/сек
Dim kmh As Single ' скорость км/ч
ms = Val(Text1.Text) ' ввод исходных данных
Глава I. Базовые компоненты 7
```

```
kmh = ms * 3.6 ' пересчет
' вывод результата
Label3.Caption = Format$(ms) + " м/сек - это " + _
Format$(kmh) + " км/ч"
End Sub
```

2. Программа, как и в примере 1, пересчитывает скорость ветра из "метров в секунду" в "километры в час". Форма программы приведена на рис. 1.1. Программа составлена таким образом, что пользователь смог ввести в поле **Скорость [м/сек]** только целое положительное число.

```

' щелчок на кнопке Пересчет
Private Sub Command1_Click()
Dim ms As Integer ' скорость м/сек
Dim kmh As Single ' скорость км/ч
' проверим, ввел ли пользователь число в поле Скорость
If Len(Text1.Text) = 0 Then
' если число не введено, то выводится сообщение
' "Нужно ввести скорость"
Label3.Caption = " Нужно ввести скорость "
Else
ms = Val(Text1.Text) ' ввод исходных данных
kmh = ms * 3.6 ' пересчет
' выводу результата
Label3.Caption = Format$(ms) + " м/сек - это " + _
Format$(kmh) + " км/ч"
End If
End Sub

```

```

' нажатие клавиши в поле Скорость
Private Sub Text1_KeyPress(KeyAscii As Integer)
' В поле Скорость можно вводить только цифры. Данная
' процедура проверяет, является ли введенный символ
' цифрой. Если нет, то введенный символ заменяется
' нулевым и в поле редактирования не отображается.

```

Часть I.Примеры 8

```

' KeyAscii - это код нажатой клавиши. 48 - 57 - коды
' цифр от 0 до 9, 8 - код клавиши <Backspace>.
TheInf Not (KeyAscii >= 48 And KeyAscii <=57 Or KeyAscii=8)
KeyAscii =0
End If
End Sub

```

3. Программа пересчитывает скорость ветра из "метров в секунду" в "километры в час". Форма программы приведена на рис. 1.1. Программа составлена таким образом, что пользователь может ввести в поле **Скорость** только целое положительное число. Вычисление выполняется как в результате щелчка мышью на кнопке **Пересчет**, так и в результате нажатия клавиши <Enter>, после ввода последней цифры данных в поле **Скорость**.

```

' процедура пересчитывает скорость из м/сек в км/ч
Private Sub WindSpeed()
Dim ms As Integer ' скорость м/сек
Dim kmh As Single ' скорость км/ч
' проверим, ввел ли пользователь число в поле Скорость
If Len(Text1.Text) = 0 Then • ;.,
' если число не введено, то выводится сообщение
' "Нужно ввести скорость"
Label3.Caption = " Нужно ввести скорость "
Else
ms = Val(Text1.Text) ' ввод исходных данных
kmh = ms * 3.6 ' пересчет
' вывод результата
Label3.Caption = Format$(ms) + " м/сек - это " + _

```

```

Format$(kmh) + " км/ч"
End If
End Sub
' щелчок на кнопке Пересчет

```

Глава I. Базовые компоненты 9

```

Private Sub Command1_Click()
Call WindSpeed
End Sub
' нажатие клавиши в поле Edit1
Private Sub Text1_KeyPress(KeyAscii As Integer)
' KeyAscii - код нажатой клавиши
' 48 - 57 - коды цифр от 0 до 9
' 8 - код клавиши <Backspace>
' 13 - код клавиши <Enter>
Select Case KeyAscii
Case 48 To 57, 8 ' цифры и <Backspace>
Case 13 ' <Enter>
Call WindSpeed
Case Else ' остальные символы
KeyAscii = 0
End Select
End Sub

```

4. Программа пересчитывает вес из фунтов в килограммы (1 фунт — 409,5 г). Форма программы приведена на рис. 1.2. Программа составлена таким образом, что кнопка **Пересчет** доступна только в том случае, если пользователь ввел исходные данные.

Рис. 1.2. Форма программы **Фунты-Килограммы**
 Часть I. Примеры 10

```

' щелчок на кнопке Пересчет
Private Sub Command1_Click()
Dim funt As Single ' вес в фунтах
Dim kg As Single ' вес в килограммах
' Кнопка Пересчет доступна только в том случае, если
' в поле Фунты есть данные. Поэтому наличие
информации
' в поле можно не проверять.
funt = Val(Text1.Text)
kg = funt * 0.4995
Label2.Caption = Format$(funt) + " ф - это " + _
Format$(kg) + " кг"
End Sub
' инициализация формы
Private Sub Form_Initialize()
' поле Фунты пусто (пользователь еще не ввел исходные
' данные), сделаем кнопку Пересчет недоступной
Comamand1.Enabled = False
End Sub
' содержимое поля Фунты изменилось
Private Sub Text1_Change()
' проверим, есть ли в поле Фунты исходные данные
If Len(Text1.Text) = 0 Then
Command1.Enabled = False ' кнопка Пересчет

```

```

недоступна
Else
Command1.Enabled = True ' кнопка Пересчет доступна
End If
End Sub
' нажатие клавиши в поле Фунты
Private Sub Text1_KeyPress(KeyAscii As Integer)
Select Case KeyAscii
Глава I. Базовые компоненты 11

```

```

Case 48 To 57, 8 ' цифры 0 - 9 и <Backspace>
' обработка десятичного разделителя
Case 44, 46 ' 44 - код запятой, 46 - код точки
' если в поле Фунты введена запятая, то
' заменим ее на десятичный разделитель
KeyAscii =46
' проверим, введен ли уже в поле Фунты
' десятичный разделитель
If InStr(Text1.Text, ".") > 0 Then
KeyAscii = 0
End If
Case Else
KeyAscii = 0 ' остальные символы запрещены
End Select
End Sub

```

5. Программа вычисляет скорость (км/час), с которой бегун пробежал дистанцию. Форма программы приведена на рис. 1.3. Количество минут задается целым числом, количество секунд - дробным.

Рис. 1.3. Форма программы Скорость бега
Часть I. Примеры 12

```

' щелчок на кнопке Вычислить
Private Sub Command1_Click()
Dim dist As Integer ' дистанция, метров
Dim min As Integer ' время, минуты
Dim sek As Single ' время, секунды
Dim v As Single ' скорость
' получение исходных данных из полей ввода
dist = Val(Text1.Text)
min = Val(Text2.Text)
sek = Val(Text3.Text)
' дистанция и время не должны быть равны нулю
If (dist = 0) Or ((min = 0) And (sek = 0)) Then
Label5.Caption = "Нужно задать дистанцию и время."
Exit Sub
End If
' вычисление
v = (dist / 1000) / ((min * 60 + sek) / 3600)
' вывод результата
Label5.Caption = "Дистанция: " + Format$(dist) + _
"М" + Chr(13) + _
"Время: " + Format$(min) + " МИН " + _
Format$(sek) + " сек " + Chr(13) + _

```

```
"Скорость: " + Format$(v, "0.00") + _  
" км/час"  
' функция Chr() возвращает символ по значению  
' числового кода в Ascii
```

```
End Sub
```

```
' щелчок на кнопке Завершить  
Private Sub Command2_Click()
```

```
' закрытие главной формы  
Глава I. Базовые компоненты 13
```

```
Unload Form1
```

```
End Sub
```

```
' нажатие клавиши в поле Дистанция
```

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
```

```
' KeyAscii - код символа, соответствующего нажатой  
клавише.
```

```
' Если символ недопустимый, то процедура заменяет его  
' на символ с кодом 0. В результате этого символ в поле  
' редактирования не появляется.
```

```
Select Case KeyAscii
```

```
Case 48 To 57, 8 ' цифры 0 - 9 и <Backspace>
```

```
Case 13
```

```
' при нажатии клавиши <Enter> курсор переводится  
' в поле Время: Минут
```

```
Text2.SetFocus
```

```
Case Else
```

```
KeyAscii = 0 ' остальные символы не отображаются
```

```
End Select
```

```
End Sub
```

```
' нажатие клавиши в поле Время: Минут
```

```
Private Sub Text2_KeyPress(KeyAscii As Integer)
```

```
Select Case KeyAscii
```

```
Case 48 To 57, 8 ' цифры 0 - 9 и <Backspace>
```

```
Case 13
```

```
' при нажатии клавиши <Enter> курсор переводится  
' в поле Время: Секунд
```

```
Text3.SetFocus
```

```
Case Else
```

```
KeyAscii = 0 ' остальные символы не отображаются
```

```
End Select
```

```
End Sub
```

```
' нажатие клавиши в поле Время: Секунд
```

```
Private Sub Text3_KeyPress(KeyAscii As Integer)
```

```
Часть I. Примеры 14
```

```
Select Case KeyAscii
```

```
Case 48 To 57, 8 ' цифры 0 - 9 и <Backspace>
```

```
Case 44, 46 ' десятичный разделитель
```

```
KeyAscii = 46
```

```
If InStr(Text3.Text, ".") <> 0 Then
```

```
KeyAscii = 0
```

```
End If
```

```
Case 13 ' клавиша <Enter>
```

```
' при нажатии клавиши <Enter> устанавливается
```

```
' фокус на кнопку Вычислить
```



```

Cammand1.SetFocus
Case Else
KeyAscii = 0 ' остальные символы не отображаются
End Select
End Sub

```

6. Программа вычисляет доход по вкладу. Она обеспечивает расчет простых и сложных процентов. Простые проценты начисляются в конце срока вклада, сложные — ежемесячно и прибавляются к первоначальной сумме вклада. В следующем месяце проценты начисляются на новую сумму. Форма программы приведена на рис. 1.4.

```

' щелчок на кнопке Вычислить
Private Sub Conmand1_Click()
Dim sum As Single ' сумма вклада
Dim pr As Single ' процентная ставка
Dim srok As Integer ' срок вклада
Dim dohod As Single ' доход по вкладу
Dim buf As Single
Dim i As Integer
' получение исходных данных
sum = Val(Text1.Text)
srok = Val(Text2.Text)
pr = Val(Text3.Text)

```

Глава I. Базовые компоненты 15

Рис. 1.4. Форма программы Доход по вкладу

```

If Option1.Value = True Then
' выбран переключатель Простые проценты
dohod » sum * (pr / 100) * (srok / 360)
Else
' выбран переключатель Сложные проценты
buf = sum
For i = 1 To srok
buf = buf + buf * (pr / 100)
' сумма в конце срока вклада записывается в buf
dohod = buf - sum
Next i
End If
sum = sum + dohod
Label4.Caption = "Доход:" + Format$(dohod, "0.00") + _
Chr(13) + _

```

```

"Сумма в конце срока вклада: " + _
Format$(sum, "0.00")

```

```

End Sub
' выбор переключателя Простые проценты
Private Sub Option1_Click()
Label2.Caption = "Срок (дней):"
Label4.Caption = ""
End Sub
' выбор переключателя Сложные проценты
Private Sub Option2_Click()
Label2.Caption = "Срок (мес.):"
Label4.Caption = ""
End Sub

```

7. Программа вычисляет сопротивление электрической цепи, состоящей из двух сопротивлений. Сопротивления могут

быть соединены последовательно или параллельно. Форма программы приведена на рис. 1.5. Если величина сопротивления цепи превышает 1000 Ом, результат выводится в кОм.

```
' щелчок на кнопке Вычислить
Private Sub Convnancl_Click ()
• Dim r1, r2 At Single ' значения сопротивлений
R1 U R2
Dim r Кв Single ' сопротивление цепи
' получение исходных' данных
r1 = Val(Text1.Text)
r2 = Val(Text2.Text)
If (r1 = 0) And (r2 = 0) Then
Label4.Caption = "Нужно задать величину хотя бы " +
—
"одного сопротивления. "
```

```
Exit Sub
```

```
End If
```

Глава I. Базовые компоненты 17

Рис. 1.5. Форма программы **Сопротивление электрической цепи**

```
' Переключатели "Последовательное соединение" и
' "Параллельное соединение" зависимы, поэтому о типе
' соединения можно судить по состоянию одного из них.
```

```
If Option1.Value = True Then
```

```
' выбран переключатель Последовательное соединение
```

```
r = r1 + r2
```

```
Else
```

```
' выбран переключатель Параллельное соединение
```

```
r = (r1 * r2) / (r1 + r2)
```

```
End If
```

```
Label4.Caption = "Сопротивление цепи: "
```

```
If r < 1000 Then
```

```
Label4.Caption = Label4.Caption + _
```

```
Format$(r/ "0.00") + " Ом"
```

Часть I. Примеры 18

```
Else
```

```
r = r / 1000
```

```
Label4.Caption = Label4. Caption + _
```

```
Format$(r, "0.00") + " кОм"
```

```
End If
```

```
End Sub
```

```
' щелчок на переключателе "Последовательное соединение"
```

```
Private Sub Option1_Click()
```

```
' пользователь изменил тип соединения,
```

```
' очистим поле вывода от предыдущего вычисления
```

```
Label4.Caption = ""
```

```
End Sub
```

```
' щелчок на переключателе "Параллельное соединение"
```

```
Private Sub Option2_Click()
```

```
Label4.Caption = ""
```

```
End Sub
```

8. Программа вычисляет силу тока, напряжение или сопротивление электрической цепи, используя закон Ома. Форма

программы приведена на рис. 1.6.

Рис. 1.6. Форма программы Закон Ома

Глава I. Базовые компоненты 19

```
' вычисление тока, напряжения или сопротивления
Sub Calculate()
Dim I As Single ' ток
Dim U As Single ' напряжение
Dim R As Single ' сопротивление
If Option1.Value = True Then
' ток
U = Val(Text1.Text)
R = Val(Text2.Text)
If R <> 0 Then
I = U / R
Label3.Caption = "Ток: " + Format$(I, "0.00")
+ "А"
Else
Label3.Caption = "Сопротивление не должно быть "
+ "равно нулю."
End If
Exit Sub
End If
If Option2.Value = True Then
' напряжение
I = Val(Text1.Text)
R = Val(Text2.Text)
U = I * R
Label3.Caption = "Напряжение: " + _
Format$(U, "0.00") + " В"
Exit Sub
End If
If Option3.Value = True Then
' сопротивление
U = Val(Text1.Text)
I = Val(Text2.Text)
Часть I.Примеры 20

If I <> 0 Then
R = U / I
Label3.Caption = "Сопротивление: " + _
Format$(R, "0.00") + " Ом"
Else
Label3.Caption = "Ток не должен быть равен нулю."
End If
End If
End Sub
' щелчок на кнопке Вычислить
Private Sub Comrnand1_Click()
If Text1.Text <> "" And Text2.Text <> "" Then
Calculate
Else
Label3.Caption = "Нужно ввести исходные данные " +
"в оба поля."
End If
End Sub
```

```

' выбор переключателя Ток
Private Sub Option1_Click()
Label1.Caption = "Напряжение (Вольт):"
Label2.Caption = "Сопротивление (Ом):"
Label3.Caption = ""
End Sub

```

```

' выбор переключателя Напряжение
Private Sub Option2_Click()
Label1.Caption = "Ток (Ампер):"
Label2.Caption = "Сопротивление (Ом):"
Label3.Caption = ""
End Sub

```

Глава I. Базовые компоненты 21

```

' выбор переключателя Сопротивление
Private Sub Option3_Click()
Label1.Caption = "Напряжение (Вольт):"
Label2.Caption = "Ток (Ампер):"
Label3.Caption = ""
End Sub
' нажатие клавиши в поле Напряжение/Ток
Private Sub Text1JKeyPress(KeyAscii As Integer)
Select Case KeyAscii
Case 48 To 57, 8 ' цифры и <Backspace>
Case 13 ' клавиша <Enter>
Text2.SetFocus
Case 44, 46 ' точка и запятая
KeyAscii =46
' не позволяет вводить знак запятой повторно
If InStr(Text1.Text, ".") < 0 Then
KeyAscii =0
End If
Case Else
KeyAscii = 0 ' остальные символы не отображаются
End Select
End Sub

```

```

' нажатие клавиши в поле Сопротивление/Ток
Private Sub Text2_KeyPress(KeyAscii As Integer)
Select Case KeyAscii
Case 48 To 57, 8 ' цифры и <Backspace>
Case 13 ' клавиша <Enter>
Calculate
Case 44, 46 ' точка и запятая
KeyAscii = 46
' не позволяет вводить знак запятой повторно
If InStr(Text2.Text, ".") <> 0 Then

```

Часть I.Примеры 22

```

KeyAscii =0
End If
Case Else
KeyAscii = 0 ' остальные символы не отображаются
End Select
End Sub

```

9. Программа вычисляет стоимость поездки на автомобиле, на-

пример, на дачу. Форма программы приведена на рис. 1.7.

Рис. 1.7. Форма программы Поездка на дачу

```
' При инициализации формы свойству Tag компонентов  
' Text1, Text2 и Text3 присваиваются соответствующее  
' значения 1, 2 и 3. Свойство Tag используется в  
' процедуре KeyPress.
```

```
Private Sub Fomr\_Initialize()
```

```
Text1.Tag = 1 ' значение свойства Tag поля Text1
```

```
Text2.Tag = 2 ' Tag поля Text2
```

```
Text3.Tag = 3 ' Tag поля Text3
```

```
End Sub
```

```
' щелчок на кнопке Вычислить
```

Глава I. Базовые компоненты 23

```
Private Sub Cammand1_Click()
```

```
Dim rast As Single ' расстояние
```

```
Dim cena As Single ' цена
```

```
Dim potr As Single ' потребление на 100 км
```

```
Dim surnm As Single ' сумма
```

```
Dim mes As String
```

```
' получение исходных данных
```

```
rast = Val(Text1.Text)
```

```
cena = Val(Text2.Text) ,
```

```
potr = Val(Text3.Text)
```

```
' При чтении данных из полей ввода
```

```
' возможен случай, когда пользователь оставит
```

```
' одно из полей ввода незаполненным.
```

```
If rast = 0 Or cena = 0 Or potr = 0 Then
```

```
Label4.Caption = "Данные нужно ввести во все поля."
```

```
If Len(Text1.Text) = 0 Then
```

```
Text1.SetFocus
```

```
Else
```

```
If Len(Text2.Text) = 0 Then
```

```
Text2.SetFocus
```

```
Else
```

```
Text3.SetFocus
```

```
End If
```

```
End If
```

```
Exit Sub
```

```
End If
```

```
sumn = (rast / 100) * potr * cena
```

```
mes = "Поездка на дачу "
```

```
If Check1.Value = Checked Then
```

```
summ = sumn * 2
```

Часть I. Примеры 24

```
mes = mes + "и обратно "
```

```
End If
```

```
mes = mes + Chr(13) + "обойдется в " + _
```

```
Format$(sumn, "0.00") + " руб."
```

```
Label4.Caption = mes
```

```
End Sub
```

```
' Процедура KeyPress обрабатывает нажатие клавиш в полях
```

```
' Расстояние, Цена и Потребление.
```

```
' В качестве входных параметров этой процедуры используются
```

```
' код нажатой клавиши KeyAscii и имя поля, для которого эта
```

```

' процедура должна выполняться.
Sub KeyPress(KeyAscii As Integer, Text As TextBox)
Select Case KeyAscii
Case 48 To 57, 8 ' цифры и <Backspace>
Case 44, 46 ' точка и запятая
KeyAscii =46
' не позволяет вводить знак запятой повторно
If InStr(Text.Text, ".") <> 0 Then
KeyAscii =0
End If
Case 13 ' клавиша <Enter>
Select Case Text.Tag
Case 1 ' клавиша нажата в поле Text1
Text2.SetFocus
Case 2 ' клавиша нажата в поле Text2
Text3.SetFocus
Case 3 ' клавиша нажата в поле Text3
Command1.SetFocus
End Select
Case Else
KeyAscii = 0 ' остальные символы не отображаются
End Select

```

Глава I. Базовые компоненты 25

```

End Sub
' нажатие клавиши в поле Расстояние
Private Sub Text1_KeyPress(KeyAscii As Integer)
Call KeyPress(KeyAscii, Text1)
End Sub
' нажатие клавиши в поле Цена
Private Sub Text2_KeyPress(KeyAscii As Integer)
Call KeyPress(KeyAscii, Text2)
End Sub
' нажатие клавиши в поле Потребление
Private Sub Text3_KeyPress (KeyAscii As Integer)
Call KeyPress(KeyAscii, Text3)
End Sub

```

10. Программа "Калькулятор" выполняет сложение и вычитание.

Форма программы приведена на рис. 1.8. Ниже представлены два варианта программы. В первом варианте для каждой цифровой кнопки создана отдельная процедура обработки события click. Во втором варианте событие click всех цифровых кнопок обрабатывает одна процедура, что позволило сократить текст программы.

Рис 1.8. Форма программы Калькулятор

Часть I.Примеры 26

```

' Вариант 1. Событие Click на каждой цифровой кнопке
' обрабатывает отдельная процедура
Dim accum As Single ' аккумулятор
Dim oper As Integer ' операция: 1 - "+", 2 - "-",
' 0 - "выполнить" (кнопка "=")
Dim f As Integer
' f = 0 ожидание первой цифры нового числа, например,
после

```

```

' выполнения операции , когда на индикаторе результат .
' f = 1 ожидание ввода остальных цифр .
' кнопка " 0 "
Private Sub Command0_Click()
If f = 0 Then ' первая цифра числа
Text1.Text = "0"
f = 1 ' ожидание остальных цифр
Else
' Следующее условие нужно для того , чтобы на индикаторе
' не появлялось нескольких нулей в начале числа .
If Text1.Text <> "0" Then
Text1.Text = Text1.Text + "0"
End If
End If
End Sub
' кнопка " 1 "
Private Sub Command1_Click()
If f = 0 Then ' первая цифра числа
Text1.Text = "1"
f = 1 ' ожидание остальных цифр
Else
' В случае , когда после выбора операции пользователь
' нажал " 0 " , переменной f присваивается значение 1 ,
Глава I. Базовые компоненты 27

' ожидается ввод остальных цифр числа . Для того чтобы
' в поле ввода не появлялось чисел , начинающаяся с
нуля
' и не являющихся дробными ( например " 01 " ) ,
необходимо
' выполнение следующего условия .
If Text1.Text <> "0" Then
Text1.Text = Text1.Text + "1"
Else: Text1.Text = "1"
End If
End If
End Sub
' кнопка " 2 "
Private Sub Command2_Click()
If f = 0 Then
Text1.Text = "2"
f = 1
Else
If Text1.Text <> "0" Then
Text1.Text = Text1.Text + "2"
Else: Text1.Text = "2"
End If
End If
End Sub
' кнопка " 3 "
Private Sub Command3_Click ()
If f = 0 Then
Text1.Text = "3"
f = 1
Else
If Text1.Text <> "0" Then

```

```
Text1.Text = Text1.Text + "3"  
Else: Text1.Text = "3"  
End If  
Часть I.Примеры 28
```

```
End if  
End Sub  
' КНОПКА "4"  
Private Sub Command4_Click()  
If f = 0 Then  
Text1.Text = "4"  
f = 1  
Else  
If Text1.Text <> "0" Then  
Text1.Text = Text1.Text + "4"  
Else: Text1.Text = "4"  
End If  
End If  
End Sub  
' КНОПКА "5"  
Private Sub Command5_Click()  
If f = 0 Then  
Text1.Text = "5"  
f = 1  
Else  
If Text1.Text <> "0" Then  
Text1.Text = Text1.Text + "5"  
Else: Text1.Text = "5"  
End If  
End If  
End Sub  
' КНОПКА "6"  
Private Sub Command6_Click()  
If f = 0 Then  
Text1.Text = "6"  
f = 1  
Глава I. Базовые компоненты 29
```

```
Else  
If Text1.Text <> "0" Then  
Text1.Text = Text1.Text + "6"  
Else: Text1.Text = "6"  
End If  
End If  
End Sub  
' КНОПКА "7"  
Private Sub Command7_Click()  
If f = 0 Then  
Text1.Text = "7"  
f = 1  
Else  
If Text1.Text <> "0" Then  
Text1.Text = Text1.Text + "7"  
Else: Text1.Text = "7"  
End If  
End If  
End Sub
```



```

' КНОПКА "8"
Private Sub Command8_Click()
If f = 0 Then
Text1.Text = "8"
f = 1
Else
If Text1.Text <> "0" Then
Text1.Text = Text1.Text + "8"
Else: Text1.Text = "8"
End If
End If
End Sub

```

Часть I.Примеры 30

```

' КНОПКА "9"
Private Sub Command9_Click()
If f = 0 Then
Text1.Text = "9"
f = 1
Else
If Text1.Text <> "0" Then
Text1.Text = Text1.Text + "9"
Else: Text1.Text = "9"
End If
End If
End Sub

```

```

' КНОПКА "C" - очистка
Private Sub CommandC_Click ()
Text1.Text = "0"
accum =0
oper =0
f = 0 ' ожидание первой цифры числа
End Sub

```

```

' КНОПКА "+"
Private Sub CommandPlus_Click()
' При нажатии на кнопку "+" нужно: выполнить предыдущую
' операцию, вывести результат на индикатор, запомнить
' текущую операцию и установить режим ожидания первой
' цифры нового числа .
If f = 0 Then
' пользователь щелкнул мышью по кнопке операции, но
' поле
' ввода находится в ожидании ввода первой цифры числа
oper = 1 ' запоминание операции
Else
' на индикаторе есть число, пользователь щелкнул
' мышью по кнопке операции

```

```

' на индикаторе есть число, пользователь щелкнул
' мышью по кнопке операции
Глава I. Базовые компоненты 31

```

```

DoOper
oper = 1
f = 0
End If
End Sub
КНОПКА "-"
Private Sub CommandMinus_Click()

```

```

' см. комментарий к процедуре обработки события Click
' на кнопке "+"
If f = 0 Then
ppер = 2 ' запоминание операции
Else
DoOper ' выполнение предыдущей операции
oper = 2 ' запоминание текущей операции
f = 0 ' ожидание первой цифры нового числа
End If
End Sub
' на кнопке 'Ч
If J = 0 Then
oper = 0
Else
DoOper
oper = 0
f = 0 "
End If
End Sub
' десятичная точка ", "
Private Sub CoitimandZ_Click()
кнопка =
Private Sub CornnandEnter_Click()
' см. комментарий к проц. обработки
события Click
Часть I.Примеры 32

```

```

If Text1.Text = "0" Then
Text1.Text = "0."
f = 1
End If
If InStr(Text1.Text, ".") = 0 Then
Text1.Text = Text1.Text + "."
End If
End Sub
' процедура выполнения операции
Sub DoOper()
Dim numb As Single ' число на индикаторе
' accum содержит результат предыдущей операции,
' oper - код операции, которую нужно выполнить,
' операнд находится на индикаторе.
numb = Val(Text1.Text)
Select Case oper
Case 0: accum = numb
Case 1: accum = accum + numb
Case 2: accum = accum - numb
End Select
Text1.Text = Format$(accum)
End Sub
' инициализация формы
Private Sub Form_Initialize()
oper =0
End Sub
' нажатие клавиши в поле ввода цифр
Глава I. Базовые компоненты 33

```

```

Private Sub Text1_KeyPress(KeyAscii As Integer)
' запрет ввода данных с клавиатуры
KeyAscii = 0
End Sub
' Вариант 2. Событие Click на всех цифровых кнопках
' обрабатывает одна процедура
' 1. Во время инициализации формы свойству Tag каждой
' цифровой кнопке надо присвоить значение, равное
цифре,
' которая должна появиться на индикаторе калькулятора.
' 2. Процедура DigitClick обрабатывает щелчок на кнопках
' иди _ иди % качестве входного параметра этой
' процедуры используется имя кнопки, для которой эта
' процедура должна выполняться.
'3. Процедура OperClick обрабатывает щелчок на кнопках "+", "-" и "="
Dim accum As Single ' аккумулятор
Dim oper As Integer ' операция: 1 - "+", 2 - "-",
' 0 - "выполнить" (кнопка "=")
Dim f As Integer
' f = 0 ожидание первой цифры нового числа, например,
после
' выполнения операции, когда на индикаторе результат.
' f = 1 ожидание ввода остальных цифр.
' процедура обработки щелчка на кнопках "0" - "9"
Sub DigitClick(Command As CommandButton)
Select Case Command.Tag
Case 1 To 9
If f = 0 Then ' первая цифра числа
Text1.Text = Format$(Command.Tag)
f = 1 ' ждем остальные цифры
Else
' В случае, когда после выбора операции
' пользователь нажал "0", переменной f
Часть I.Примеры 34

' присваивается значение 1, ожидается ввод
' остальных цифр числа. Для того чтобы в поле
' ввода не появлялось чисел, начинающихся с нуля,
' не являющихся дробными (например "01"),
' необходимо выполнение следующего условия.
If Text1.Text <> "0" Then
Text1.Text = Text1.Text + Format$(Command.Tag)
Else: Text1.Text = Format$(Command.Tag)
End If
End If
Case 0
If f = 0 Then ' первая цифра числа
Text1.Text = "0"
f = 1 . ' ожидание остальных цифр
Else
' Следующее условие нужно для того, чтобы на
' индикаторе не появлялось нескольких нулей
' в начале числа.
If Text1.Text <> "0" Then
Text1.Text = Text1.Text + "0"

```

```

End If
End If
End Select
End Sub
' процедура обработки щелчка на кнопках "+", "-" и "="
Sub OperClick(det As Integer)
' При нажатии кнопки "+", "-" или "=" нужно: выполнить
' предыдущую операцию, вывести результат на индикатор,
' запомнить текущую операцию и установить режим ожидания
1 первой цифры нового числа.
1 det определяет кнопку: 1 - "+", 2 - "-", 0 - "="
If f = 0 Then
' пользователь щелкнул мышью по кнопке операции, но поле ввода
Глава I. Базовые компоненты 35

' находится в ожидании ввода первой цифры числа
Select Case det ' запоминание операции
Case 0: oper = 0
Case 1: oper = 1
Case 2: oper = 2
End Select
Else
' на индикаторе есть число, пользователь щелкнул
' на кнопке операции
DoOper ' выполнение предыдущей операции
Select Case det ' запоминание текущей операции
Case 0: oper = 0
Case 1: oper = 1
Case 2: oper = 2
End Select
f = 0 ' ожидание первой цифры нового числа
End If
End Sub
' кнопка "0"
Private Sub Command0_Click()
Call DigitClick(Command0)
End Sub
' кнопка "1"
Private Sub Command1_Click()
Call DigitClick(Command1)
End Sub
' кнопка "2"
Private Sub Command2_Click()
Call DigitClick(Command2)
End Sub
Часть I. Примеры 36

' кнопка "3"
Private Sub Command3_Click()
Call DigitClick (Command3)
End Sub
' кнопка "4"
Private Sub Command4_Click()
Call DigitClick(Command4)
End Sub

```

```

' КНОПКА "5"
Private Sub Command5_Click()
Call DigitClick(Command5)
End Sub
' КНОПКА "6"
Private Sub Command6_Click()
Call DigitClick(Command6)
End Sub
' КНОПКА "7"
Private Sub Conunand7_Click()
Call DigitClick(Command7)
End Sub
' КНОПКА "8"
Private Sub Coinmand8_Click()
Call DigitClick(Command8)
End Sub
' КНОПКА "9"
Private Sub Cornmand9_Click()
Call DigitClick(Command9)
End Sub

```

Глава I. Базовые компоненты 37

```

' КНОПКА "C" - очистка Private Sub
CommandC_Click()
Text1.Text = "0"
accum = 0
oper = 0
f = 0 ' ожидание первой цифры числа
End Sub
' КНОПКА "+" Private Sub CommandPlus_Click()
Call OperClick(1) End Sub
' КНОПКА "-" Private Sub CamroandMinus_Click()
Call OperClick(2) End Sub
' КНОПКА "=" Private Sub CommandEnter_Click()
Call OperClick(0) End Sub
' десятичная точка "," Private Sub CommandZ_Click()
If Text1.Text = "0" Then Text1.Text = "0." f = 1
End If If InStr(Text1.Text, ".") = 0 Then
Text1.Text = Text1.Text + "."
End If
End Sub

```

Часть I. Примеры 38

```

' процедура выполнения операции
Sub DoOper()
Dim numb As Single ' число на индикаторе
' accum содержит результат предыдущей операции,
' oper - код операции, которую нужно выполнить,
' операнд отображается на индикаторе.
numb = Val(Text1.Text)
Select Case oper
Case 0: accum = numb
Case 1: accum = accum + numb
Case 2: accum = accum - numb
End Select
Text1.Text = Formats(accum)

```

End Sub

' инициализация формы

Private Sub Form_Initialize()

' задание значений свойства Tag для кнопок "0" - "9"

Command0.Tag =0

Command1.Tag =1

Command2.Tag =2

Command3.Tag =3

Command4.Tag =4

Command5.Tag =5

Command6.Tag =6

Command7.Tag =7

Command8.Tag =8

Command9.Tag =9

Глава I. Базовые компоненты 39

oper = 0

End Sub

' нажатие клавиши в поле ввода цифр

Private Sub Text1_KeyPress(KeyAscii As Integer)

' запрет ввода данных с клавиатуры

KeyAscii =0

End Sub

11. Программа "Электронные часы", отображающая текущее время. Форма и окно программы приведены на рис. 1.9.

Рис. 1.9. Форма и окно программы Время

' отображение текущего времени

Sub ShowTime()

Label1.Caption = CStr(Time)

End Sub

' инициализация формы

Private Sub Form_Initialize()

' настройка и запуск таймера

Timer1.Interval =1000 *' период сигналов таймера -*

1 с

Timer1.Enabled = **True** *' запуск таймера*

Label1.Font.Size =20 *' установка размера шрифта для*

' поля вывода времени

End Sub

' обработка события Paint

Private Sub Form_Paint()

Часть I.Примеры 40

Call ShowTime

End Sub

' обработка сигнала таймера

Private Sub Timer1_Timer()

Call ShowTime *' отображение времени*

End Sub

12. Программа "Электронные часы" отображает текущее время

и

дату. Форма и окно программы приведены на рис. 1.10.

Рис. 1.10. Форма и окно программы Время

' отображение текущего времени

Sub ShowTime()

```

Label1.Caption = CStr(Time)
End Sub
' инициализация формы
Private Sub Form_Initialize()
' настройка и запуск таймера
Timer1.Interval = 1000 ' период сигналов таймера -1с
Timer1.Enabled = True ' запуск таймера
' настройка полей вывода
Label1.Font.Size = 20 ' установка размера шрифта для
' поля вывода времени
Label2.Font.Size = 10 ' установка размера шрифта для
' поля вывода даты
Label1 .Alignment = 2 ' выравнивание по центру для полей

```

```

Label2.Alignment=2
' отображение даты
Label2.Caption = CStr(Date)
End Sub
' обработка события Paint
Private Sub Form_Paint()
Call ShowTime ' отображение времени
End Sub
' обработка сигнала таймера
Private Sub Timer1_Timer()
Call ShowTime ' отображение времени .
End Sub

```

13. Программа "Электронные часы" отображает текущее время, дату и день недели. Форма и окно программы приведены на рис. 1.11.

Рис. 1.11. Форма и окно программы Время

```

' отображение текущего времени
Sub ShowTime()
Label1.Caption = CStr(Time)
End Sub
' инициализация формы
Private Sub Form_Initialize()
Dim stDay, stMonth

```

```

stDay = Array("воскресенье", "понедельник", "вторник", _
"среда", "четверг", "пятница", "суббота")
stMonth = Array("января", "февраля", "марта", "апреля", _
"мая", "июня", "июля", "августа", _
"сентября", "октября", "ноября", "декабря")
Label2.Caption = "Сегодня " + Format$(Day(Date)) + _
" " + stMonth(Month(Date) - 1) + " "
+ _
Format$(Year(Date)) + " года, " + _
stDay(Weekday(Date) - 1) + "."
' отнимается 1, т.к. нумерация элементов
' массива начинается с 0
1 настройка и запуск таймера
Timer1.Interval = 1000 ' период сигналов таймера -1с
Timer1.Enabled = True ' запуск таймера
' настройка полей вывода
Label1.Font.Size = 20 ' установка размера шрифта для
' поля вывода времени

```

```

Label2.Font.Size = 10 ' установка размера шрифта для
' поля вывода даты
Label1.Alignment = 2 ' выравнивание по центру для полей
Label2.Alignment = 2 ' времени и даты
End Sub
' обработка события Paint
Private Sub Form_Paint()
Call ShowTime ' отображение времени
End Sub
' обработка сигнала таймера
Private Sub Timer1_Timer ()
Call ShowTime ' отображение времени
End Sub

```

Глава I. Базовые компоненты 43

14. Программа "Таймер" по истечении заданного интервала времени выводит сообщение. Имеется возможность приостановить работу таймера и затем пустить таймер снова. Форма и окно программы приведены на рис. 1.12.

Рис. 1.12. Форма и окно программы Таймер

```

Dim min As Integer ' количество минут
Dim sec As Integer ' количество секунд
' щелчок на кнопке "Пуск/Стоп"
Private Sub Command1_Click()
If Timer1.Enabled Then
' таймер работает, нужно остановить
Timer1.Enabled = False ' остановка таймера
Command1.Caption = "Пуск"
Label3.Visible = False ' скрывание индикатора
' поля ввода интервала и подписи полей нужно
' сделать видимыми
Label1.Visible = True
Text1.Visible = True
Label2.Visible = True
Text2.Visible = True
' вывод количества оставшихся минут и секунд
Text1.Text = Format$(min)
Text2.Text = Format$(sec)
Часть I.Примеры 44

```

```

Else
' таймер стоит, нуокно запустить
min = Val(Text1.Text)
sec = Val(Text2.Text)
If (sec = 0) And (min = 0) Then
Call MsgBox( "Нужно задать интервал." , _
vbOKOnly, "Таймер" )
Exit Sub
End If
Timer1.Enabled = True ' запуск таймера
' скрывание полей ввода интервала и подписей
Label1.Visible = False
Text1.Visible = False
Label2.Visible = False
Text2.Visible = False
Label3.Visible = True
Command1.Caption = "Стоп"
Call ShowTime

```



```

End If
End Sub
' инициализация формы
Private Sub Form_Initialize()
Label3.Visible = False ' индикатор _____ не отображается
' на форме
Label3.Alignment = 2 ' выравнивание по центру
' для индикатора
Label3.Font.Size = 20 ' размер шрифта для индикатора
Timer1.Interval =1000 ' период сигналов таймера - 1 с
Timer1.Enabled = False ' таймер остановлен
End Sub

```

Глава I. Базовые компоненты 45

```

' процедура выводит остаток времени
Sub ShowTime()
' вывод минут и секунд осуществляется двумя цифрами
Label3.Caption = Format$(min, "00") + ":" + _
Format$(sec, "00")
End Sub
' обработка сигнала таймера
Private Sub Timer1_Timer ()
If sec <> 0 Then
sec = sec - 1
Else
sec =59
min = min - 1
End If
Call ShowTime ' вывод оставшегося времени
If (min = 0) And (sec = 0) Then
' заданный интервал истек
Timer1.Enabled = False ' остановка таймера
Call MsgBox("Заданный интервал истек.", _
vbOKOnly, "Таймер")
' "Заданный интервал истек." - текст сообщения,
' vbOKOnly - окно сообщения будет содержать только
' кнопку "ОК",
' "Таймер" - заголовок сообщения.
Command1.Caption = "Пуск"
Label3.Visible = False ' скрывание индикатора
' поля ввода интервала и подписи полей нужно
' сделать видимыми

```

Часть I. Примеры 46

```

Label1.Visible = True
Text1.Visible = True
Label2.Visible = True
Text2.Visible = True
Text1.Text = "0"
Text2.Text = "0"
End If
End Sub

```

15. Программа "Таймер". По истечении заданного интервала времени, воспроизводится звуковой сигнал. Форма и окно программы приведены на рис. 1.12. Воспроизведение звука

осуществляется при помощи API-функции *PlaySound* из библиотеки *winmm.dll* (библиотека *Windows Multimedia*).

```
' функция воспроизведения звукового файла
Private Declare Function PlaySound Lib "winmm.dll" _
Alias "PlaySoundA" (ByVal lpszSoundName As String, _
ByVal hModule As Long, ByVal uFlags As Long) As Long
' lpszSoundName - имя файла или другой идентификатор,
' hModule - номер модуля прикладной программы, содержащей
звук
' (если данный параметр не требуется, то ему устанавливается
значение 0),
' uFlags - флаги спецификации воспроизводимого файла,
' например:
' SND_ALIAS = &H10000 - воспроизведение системного звука,
' SND_ASYNC = &H1 - асинхронное воспроизведение, т. е.
' приложение не ждет завершения воспроизведения звука, а
' параллельно продолжает работу,
' SND_FILENAME = &H20000 - указание полного пути к файлу,
' SND_LOOP = &H8 - воспроизведение файла по кругу до тех пор,
' пока не будет вызвана команда остановки
' воспроизведения звука,
' SND_NODEFAULT = &H2 - в случае, если указанный файл
_ не найден, не проигрывается стандартный звук Windows,
' SND_PURGE = &H40 - остановка воспроизведения всех звуков,
```

Глава I. Базовые компоненты 47

```
' при этом поле lpszSoundName должно быть пусто (""),
' SND_SYNC = &H0 - синхронное воспроизведение, т. е.
' приложение ожидает завершения воспроизведения звука,
' прежде чем продолжить работу, и др.
```

```
Const SND_ALIAS = &H10000
Const SND_ASYNC = &H1
Const SND_FILENAME = &H20000
Const SND_LOOP = &H8
Const SND_NODEFAULT = &H2
Const SND_PURGE = &H40
Const SND_SYNC = &H0
```

```
Dim min As Integer ' количество минут
Dim sec As Integer ' количество секунд
```

```
' щелчок на кнопке "Пуск/Стоп"
```

```
Private Sub Command1_Click()
```

```
If Timer1.Enabled Then
```

```
' таймер работает, нужно остановить
```

```
Timer1.Enabled = False ' остановка таймера
```

```
Command1.Caption = "Пуск"
```

```
Label3.Visible = False ' скрывание индикатора
```

```
' поля ввода интервала и подписи полей нужно
```

```
' сделать видимыми
```

```
Label1.Visible = True
```

```
Text1.Visible = True
```

```
Label2.Visible = True
```

```
Text2.Visible = True
```

```
' вывод количества оставшихся минут и секунд
```

```
Text1.Text = Format$(min)
```

```
Text2.Text = Format$(sec)
```

Часть I. Примеры 48

```
Else
' таймер стоит, нужно запустить
min = Val(Text1.Text)
sec = Val(Text2.Text)
If (sec = 0) And (min = 0) Then
Call MsgBox("Нужно задать интервал.", _ -
vbOKOnly, "Таймер")
Exit Sub
End If
Timer1.Enabled = True ' запуск таймера
' скрываем поля ввода интервала и подписей
Label1.Visible = False
Text1.Visible = False
Label2.Visible = False
Text2.Visible = False
Label3.Visible = True
Command1.Caption = "Стоп"
Call ShowTime
End If
End Sub
' инициализация формы
Private Sub Form_Initialize()
Label3.Visible = False ' индикатор не отображается
' на форме
Label3.Alignment = 2 ' выравнивание по центру
' для индикатора
Label3.Font.Size = 20 ' размер шрифта для индикатора
Timer1.Interval = 1000 ' период сигналов таймера - 1 с
Timer1.Enabled = False ' таймер остановлен
End Sub
```

Глава I. Базовые компоненты 49

```
' выводит, сколько времени осталось
Sub ShowTime()
' минуты и секунды выводим двумя цифрами
Label3.Caption = Format$(min, "00") + ":" + _
Format$(sec, "00")
End Sub
' сигнал от таймера
Private Sub Timer1_Timer()
If sec <> 0 Then
sec = sec - 1
Else
sec = 59
min = min - 1
End If
Call ShowTime ' вывод оставшегося времени
If (min = 0) And (sec = 0) Then
' заданный интервал истек
Timer1.Enabled = False ' остановка таймера
' воспроизведение звука:
Call PlaySound(CStr(CurDir) + "\ringer.wav", 0,
```

—

```

SND_FILENAME Or SND_ASYNC)
' Call PlaySound("SystemStart", 0, _
SND_ALIAS Or SND_ASYNC) -
' пример воспроизведения системного звука Запуск Windows
' (звук, соответствующий этому событию в Звуковой схеме
' Windows)
Command1.Caption = "Пуск"
Label3.Visible = False ' скрывание индикатора

```

Часть I. Примеры 50

```

' поля ввода интервала и подписи полей нужно
' сделать видимыми
Label1.Visible = True
Text1.Visible = True
Label2.Visible = True
Text2.Visible = True
Text1.Text = "0"
Text2.Text = "0"
End If
End Sub

```

16. Программа "Таймер". Для ввода интервала времени используется компонент *UpDown*. Форма и окно программы приведены на рис. 1.13.

Рис. 1.13. Форма и окно программы Таймер

Если проект создавался как *Standard*, то возможно, что на панели *ToolBox* компонента *UpDown* не будет. Для того чтобы он появился, необходимо в меню *Project* выбрать команду *Components* и подключить библиотеку

Microsoft Windows Common Controls-2 6.0

Чтобы обеспечить синхронизацию компонентов *UpDown* и *Text*, нужно свойству *Buddy Control* компонента *UpDown* присвоить имя соответствующего компонента *Text* (для *UpDown1* - *Text1*, для *UpDown2* - *Text2*), а в *Buddy Property* выбрать закладку

Глава I. Базовые компоненты 51

```

' Text. Свойства Max и Min компонента UpDown определяют
' максимальное и минимальное возможные значения.
Dim min As Integer ' количество минут
Dim sec As Integer ' количество секунд
' щелчок на кнопке "Пуск/Стоп"
Private Sub Command1_Click()
If Timer1.Enabled Then
' таймер работает, нужно остановить
Timer1.Enabled = False ' остановка таймера
Command1.Caption = "Пуск"
Label3.Visible = False ' скрывание индикатора
' поля ввода интервала и подписи полей нужно
' сделать видимыми
Label1.Visible = True
Text1.Visible = True
Label2.Visible = True
Text2.Visible = True
UpDown1.Visible = True
UpDown2.Visible = True
' вывод количества оставшихся минут и секунд
Text1.Text = Format$(min)
Text2.Text = Format$(sec)

```

```

Else
' таймер стоит, нужно запустить
min = Val(Text1.Text)
sec = Val(Text2.Text)
If (sec = 0) And (min = 0) Then
Call MsgBox("Нужно задать интервал.", _
Часть I.Примеры 52

vbOKOnly, "Таймер")
Exit Sub
End If
Timer1.Enabled = True ' запуск таймера
' скрывтие полей ввода интервала и подписей
Label1.Visible = False
Text1.Visible = False
Label2.Visible = False
Text2.Visible = False
Label3.Visible = True
UpDown1.Visible = False
UpDown2.Visible = False
Command1.Caption = "Стоп"
Call ShowTime
End If
End Sub
' инициализация формы
Private Sub Form_Initialize()
Label3.Visible = False ' индикатор не отображается
' на форме
Label3.Alignment = 2 ' выравнивание по центру
Ч
' для индикатора
Label3.Font.Size = 20 ' размер шрифта для индикатора
Timer1.Interval = 1000 ' период сигналов таймера -1с
Timer1.Enabled = False ' таймер остановлен
UpDown1.Max = 60 ' максимальное значение
' для поля "Минуты"
UpDown2.Max = 59 ' максимальное значение
' для поля "Секунды"
UpDown2.Wrap = True ' если в поле "Секунды" находится
Глава I. Базовые компоненты 53

' значение 59, то после нажатия
' кнопки Up (вверх) в поле
' появится 0, при нажатии
' Down (вниз) - 59.
UpDown1.Wrap = True ' тоже самое для поля "Минуты"
End Sub
' процедура выводит остаток времени
Sub ShowTime()
' вывод минут и секунд осуществляется двумя цифрами
Label3.Caption = Format(min, "00") + ":" + _
Format$(sec, "00")
End Sub
' обработка сигнала таймера
Private Sub Timer1_Timer ()

```

```

If sec <> 0 Then
sec = sec - 1
Else
sec =59
min = min - 1
End If
Call ShowTime ' вывод оставшегося времени
If (min = 0) And (sec = 0) Then
' заданный интервал истек
Timer1.Enabled = False ' остановка таймера
Call MsgBox( " Заданный интервал истек. " , _
vbOKOnly, " Таймер" )
' "Заданный интервал истек." - текст сообщения,
' vbOKOnly - окно сообщения будет содержать только
' кнопку "ОК",
Часть I.Примеры 54

' "Таймер" - заголовок сообщения.
Command1.Caption = "Пуск"
Label13.Visible = False ' скрывание индикатора
' поля ввода интервала и подписи полей нужно
' сделать видимыми
Label1.Visible = True
Text1.Visible = True
Label2.Visible = True
Text2.Visible = True
Text1.Text = "0"
Text2.Text = "0"
UpDown1.Visible = True
UpDown2.Visible = True
End If
End Sub
Private Sub Text1_KeyPress(KeyAscii As Integer)
' запрет ввода данных с клавиатуры
KeyAscii =0
End Sub
Private Sub Text2_KeyPress(KeyAsrii As Integer)
' запрет ввода данных с клавиатуры
KeyAscii =0
End Sub

```

Глава 2

Графика

Общие замечания

Программа может выводить графику на поверхность формы или компонента *PictureBox*.

Вычерчивание графических элементов (линий, окружностей, ПРЯМОУГОЛЬНИКОВ И Т. Д.) ВЫПОЛНЯЮТ МЕТОДЫ *Line* И *Circle*.

Цвет, стиль и толщину линий, вычерчиваемых методами *Line* и *Circle* определяют СВОЙСТВА *ForeColor*, *DrawStyle* И *DrawWidth* той графической поверхности, на которой рисует метод.

Цвет и способ (стиль) закраски внутренних областей геометрических фигур, вычерчиваемых методами *Line* и *circle*, определяют свойства *Filicolor* и *Filistyle* той графической поверх-

ности, на которой рисует метод.

Характеристики шрифта текста, выводимого методом *Print*, определяются свойством *Font* той графической поверхности, на которую метод выводит текст.

Основную работу по выводу графики на поверхность формы должна выполнять функция обработки события *Paint*.

1. Программа вычерчивает контур пятиконечной звезды в том месте, где произведен щелчок кнопкой мыши. При щелчке правой кнопкой мыши — контур звезды имеет черный цвет, при щелчке левой кнопкой — красный. Окно программы приведено на рис. 2.1.

Часть 1. Примеры 56

Рис. 2.1. Окно программы Звезды

```
' определение нового типа Point
Private Type Point
X As Integer
Y As Integer
End Type
Const PI = 3.14159265 ' число "пи"
Const R = 10 ' радиус звезды
Dim Color
' процедура рисует звезду
Sub StarDraw(x0 As Single, y0 As Single, R As Integer)
' x0, y0 - координаты центра звезды, R - радиус звезды.
Dim p(1 To 11) As Point ' массив координат лучей и
впадин
Dim a As Integer ' угол между осью OX и прямой,
' соединяющей центр звезды и конец
' луча или впадину
Dim i As Integer
a = 18 ' построение звезды начинается от
' правого горизонтального луча
For i = 1 To 10
Глава 2. Графика 57

If (i Mod 2=0) Then ' впадина
p(i).X = x0 + Round(R / 2.75 * Cos(a * 2 * PI / 360))
p(i).Y = y0 - Round(R / 2.75 * Sin(a * 2 * PI / 360))
Else ' луч
p(i).X = x0 + Round(R * Cos(a * 2 * PI / 360))
p(i).Y = y0 - Round(R * Sin(a * 2 * PI / 360))
End If
a = a + 36
Next i
p(11).X = p(1).X ' замыкание контура звезды
p(11).Y = p(1).Y
' вычерчивание контура звезды
For i = 1 To 10
Line (p(i).X, p(i).Y)-(p(i + 1).X, p(i + 1).Y), Color
Next i
End Sub
Private Sub Form_Load()
Form1.ScaleMode = vbPixels ' координаты задаются
' в пикселах
End Sub
```

```

' нажатие кнопки мыши
Private Sub Form_MouseDown(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
' Параметр Button определяет кнопку мыши:
' левая кнопка - Button = 1, правая - Button = 2
If Button = vbLeftButton Then ' нажата левая
кнопка
Color = RGB(255, 0, 0)
Else: Color = RGB(0, 0, 0) ' нажата правая кнопка
End If
Call StarDraw(X, Y, R)
End Sub
Часть I.Примеры 58

```

2. Программа "Тур". По поверхности диалогового окна случайным образом перемещается изображение "рожицы", на котором пользователь может сделать щелчок кнопкой мыши. Программа завершает работу после того, как пользователь сделает 10 щелчков. Вид диалогового окна, в начале работы программы, показан на рис. 2.2.

Рис. 2.2. Окно программы Тур

```

Dim fx, fy As Integer
Dim n As Integer
Dim p As Integer
' процедура стирает рожицу
Sub EraseFace(X, Y As Integer)
' задание цвета границы и цвета заливки,
' совпадающего с цветом формы
FillStyle = 0 ' способ заливки - сплошной
FillColor = Form1.BackColor
Circle (X, Y), 15, Form1.BackColor
End Sub
' процедура рисует рожицу
Глава 2. Графика 59

```

```

Sub PaintFace(X, Y As Integer)
FillStyle = 0 ' способ заливки - сплошной
FillColor = RGB(255, 255, 0) ' цвет заливки
Circle (X, Y), 15, RGB(0, 0, 0) ' лицо
FillColor = RGB(0, 0, 0)
Circle(X+5, Y - 5), 1, RGB(0, 0, 0) ' правый глаз
Circle(X-5, Y - 5), 1, RGB(0, 0, 0) ' левый глаз
FillStyle = 1 ' внутри окружность не закрашивается
' улыбка рисуется при помощи дуги
Circle (X, Y), 10, RGB(0, 0, 0), 3.14, 3.14 * 2
End Sub
' щелчок на кнопке "Ok"
Private Sub Command1_Click()
Command1.Visible = False
Label1.Visible = False
n = 0
P = 0
Call PaintFace(fx, fy)
Timer1.Interval = 800
Timer1.Enabled = True
End Sub
' инициализация формы
Private Sub Form_Initialize()

```



```

Label1.Caption=" Сейчас по экрану будет перемещаться" +
—
" картинка. Ваша задача – щелкнуть по картинке левой" +
—
" кнопкой мыши. Цель – максимальное число попаданий."
Form1.ScaleMode = 3 ' координаты задаются в пикселах
fx = 100 ' исходное положение рожицы
Часть I.Примеры 60

```

```

fy = 100
Randomize ' инициализация генератора случайных
чисел
End Sub
' нажатие клавиши мыши
Private Sub Form_MouseDown(Button As Integer, Shift As
Integer, X As Single, Y As Single)
If Button = 1 Then ' нажата левая кнопка
n = n + 1 ' увеличение счетчика щелчков
If (X > fx - 15) And (X < fx + 15) And _
(Y > fy - 15) And (Y < fy + 15) Then
p = p + 1 ' попадание
End If
If n = 10 Then
' игра закончена
Timer1.Enabled = False ' остановка таймера
' вывод результата игры
MsgBox "Выстрелов: 10. Попаданий: " + _
Format$(p) + ". ", vbOKOnly, "Игра
закончена."
' стирание рожицы, кнопка "Ok" и сообщение
' снова видны
Call EraseFace(fx, fy)
Label1.Visible = True
Command1.Visible = True
End If
End If
End Sub

```

Глава 2. Графика 61

```

' обработка сигнала таймера
Private Sub Timer1_Timer ()
' стирание рожицы
Call EraseFace(fx, fy)
' новое положение рожицы
fx = Int(((Form1.ScaleWidth - 30) * Rnd) + 1) + 15
' ((Form1.ScaleWidth - 30) * Rnd - это случайное
число,
' между 1 и (Form1.Width - 30) , 30 - диаметр рожицы
fy = Int(((Form1.ScaleHeight - 30) * Rnd) + 1) + 15
Call PaintFace(fx, fy)
End Sub

```

3. Программа рисует флаг РФ. Окно программы приведено на рис. 2.3.

Рис. 2.3. Окно программы **Флаг РФ**

```

Const L = 125 ' длина полосы флага

```

```

Const H = 25 ' высота полосы флага
Const x = 25 ' левый верхний угол флага
Const y = 25 инициализация формы
Private Sub Form_Initialize()
' установить размер формы, чтобы флаг был в ее центре
Form1.Height = (Form1.Height - Form1.ScaleHeight) +
Часть I.Примеры 62

(H * 3 + y * 2) * Screen.TwipsPerPixelY
Form1.Width = (Form1.Width - Form1.ScaleWidth) + _
(L + x * 2) * Screen.TwipsPerPixelX
' Form1.Height-высота формы, включая заголовок и границы
' Form1.ScaleHeight-высота формы без заголовка и границ,
' аналогично для Form1.Width и Form1.ScaleWidth.
' (H * 3 + y * 2) умножается на Screen.TwipsPerPixelY,
' т.к. Form1.Height задается в твипах, а не в пикселах,
' аналогично для (L + x * 2) и Screen.TwipsPerPixelX.
ScaleMode = vbPixels ' параметры прямоугольника задаются
' в пикселах
End Sub
' обработка события Paint
Private Sub Forro_Paint ()
Line (x, y)-Step(L, H) , RGB(255, 255, 255), BF
' (X/Y) - координаты левого верхнего угла
прямоугольника,
' (L,H) - длина и ширина прямоугольника,
' RGB (0, 0, 255) - цвет границы,
' BF - прямоугольник закрашивается внутри.
' Если BF заменить на B, то прямоугольник
' не будет закрашиваться внутри.
Line (x, y + H)-Step(L, H), RGB(0, 0, 255), BF
Line (x, y + 2 * H)-Step(L, H), RGB(255, 0, 0), BF
' граница флага
Line (x, y)-Step(L, 3 * H), RGB(0, 0, 0), B
End Sub

```

4. Программа вычерчивает на поверхности формы кривую Гильберта (рис. 2.4). Пример кривой Гильберта пятого порядка показан на рис. 2.5.

Глава 2. Графика 63

Кривые первого порядка
получаются путем
соединения кривых
нулевого порядка (точек)

Рис. 2.4. Кривая Гильберта

Рис. 2.5. Окно программыКривая Гильберта

' Кривая Гильберта состоит из четырех, соединенных прямыми,
' элементов: a, b, c и d. Каждый элемент строит
' соответствующая процедура.

```

Dim p As Integer
Dim u As Integer
' порядок кривой
' длина штриха

```

Часть I.Примеры 64

```

Dim size As Integer ' размер кривой
' процедура рисует элемент a
Sub a(i As Integer)

```

```

If i > 0 Then
Call d(i - 1)
Line -(CurrentX + u, CurrentY)
Call a(i - 1)
Line -(CurrentX, CurrentY + u)
Call a(i - 1)
Line -(CurrentX - u, CurrentY)
Call c(i - 1)
End If
End Sub
' процедура рисует элемент b
Sub b(i As Integer)
If i > 0 Then
Call c(i - 1)
Line -(CurrentX - u, CurrentY)
Call b(i - 1)
Line -(CurrentX, CurrentY - u)
Call b(i - 1)
Line -(CurrentX + u, CurrentY)
Call d(i - 1)
End If
End Sub
' процедура рисует элемент c
Sub c(i As Integer)
If i > 0 Then
Call b(i - 1)
Line -(CurrentX, CurrentY - u)
Call c(i - 1)
Глава 2. Графика 65

Line -(CurrentX - u, CurrentY)
Call c(i - 1)
Line -(CurrentX, CurrentY + u)
Call a(i - 1)
End If
End Sub
' процедура рисует элемент d
Sub d(i As Integer)
If i > 0 Then
Call a(i - 1)
Line -(CurrentX, CurrentY + u)
Call d(i - 1)
Line -(CurrentX + u, CurrentY)
Call d(i - 1)
Line -(CurrentX, CurrentY - u)
Call b(i - 1)
End If
End Sub
' инициализация формы
Private Sub Form_Initialize()
p = 5
u = 7
' размер кривой (количество элементов по длине или
ширине)
' определяется как 2 в степени p минус 1 ( $2^p - 1$ )
size = 2
For i = 1 To (p - 1)
size = size * 2

```

Next i

`size = size - 1`

`Form1.Height = (Form1.Height - Form1.ScaleHeight) + _`

Часть I.Примеры 66

`(size + 2) * u * Screen.TwipsPerPixelY`

`Form1.Width = (Form1.Width - Form1.ScaleWidth) + _`

`(size + 2) * u * Screen.TwipsPerPixelX`

`Form1.ScaleMode = 3`

End Sub

' обработка события Paint

Private Sub Form_Paint()

`Form1.CurrentX = u`

`Form1.CurrentY = u`

' вычерчивание кривой Гильберта

Call a(p)

End Sub

5. Программа выводит на поверхность формы изображение оцифрованной координатной сетки. Окно программы показано на рис. 2.6.

Рис. 2.6. Окно программы Координатная сетка

`Dim x0, y0 As Integer`

`Dim dx, dy As Integer`

`Dim h, w As Integer`

' координаты начала координатных осей

' шаг координатной сетки по X и Y

' высота и ширина области вывода

Глава 2. Графика 67

`Dim x, y As Integer`

`Dim lx, ly As Single`

`Dim dlx, dly As single`

`Dim cross A* Integer`

`Dim dcross As integer`

' координатной сетки

' метки (оцифровка) линий сетки

' по осям X и Y

' шаг меток (оцифровки) линий сетки

' по осям X и Y

' счетчик не оцифрованных линий сетки

' количество не оцифрованных линий

' между оцифрованными по оси X

' инициализация формы

Private Sub Form_Initialize()

`Form1.BackColor = RGB(255, 255, 255)`

`h`

`w`

`x0 = 30 ' оси начинаются в точке (30, 15 + h)`

`y0 = 15 + h`

`dx = 30`

`dy = 30`

`dcross = 1 ' линии сетки X помечаются:`

' dcross = 1 - каждая,

' dcross = 2 - через одну,

' dcross = 3 - через две и т. д.

`dlx = 0.5 ' шаг меток оси X`

`dly = 1 ' шаг меток оси Y, метками будут:`

' 1, 2, 3, 4 и т. д.

`cross = dcross`

Часть I.Примеры 68

```

Form1.Height = (Form1.Height - Form1.ScaleHeight) + _
(h + 30 + 15) * Screen.TwipsPerPixelY
Form1.Width = (Form1.Width - Form1.ScaleWidth) + _
(w + 30 + 15) * Screen.TwipsPerPixelX
Form1.ScaleMode = 3
End Sub
' обработка события Paint
Private Sub Form_Paint()
Line (x0, y0)-(x0, y0 - h) ' ось X
Line (x0, y0)-(x0 + w, y0) ' ось Y
' засечки, сетка и оцифровка по оси X
x = x0 + dx
ix = dix ;
While (x < x0 + w)
Line (x, y0 + 3)-(x, y0 - 3) ' засечка
cross = cross - 1
If cross = 0 Then ' оцифровка
Form1.CurrentX = x - 8
Form1.CurrentY = y0 + 5
Print Formats(lx, "0.0")
cross = dcross
End If
Form1.DrawStyle = 2 ' задание типа линии :
' пунктирный
Line (x, y0 - 5)-(x, y0 - h) ' линия сетки
Form1.DrawStyle = 0 ' задание типа линии :
' сплошной
lx = lx + dlx
x = x + dx
Wend

```

Глава 2. Графика 69

```

' засечки, сетка и оцифровка по оси Y
y = y0 - dy
ly * dly
while (y > y0 - h)
Line (x0 - 3, y)-(x0 + 3, y) ' засечка
Form1.CurrentX = x0 - 22 ' оцифровка
Form1.CurrentY = y - 5
Print Format$(ly, "0.0")
Form1.DrawStyle = 2
Line (x0 + 5, y)-(x0 + w, y) ' линия сетки
Form1.DrawStyle = 0
y = y - dy
ly = ly + dly
Wend
End Sub

```

6. Программа вычерчивает график функции (в данном примере функция имеет вид $2\sin(x)e^{x/5}$) на поверхности формы. Окно программы приведено на рис. 2.7.

Рис. 2.7. Окно программы График функции

```

' инициализация формы
Private Sub Form_Initialize()
Часть I.Примеры 70

```

```

Form1.BackColor = RGB(255, 255, 255)
End Sub

```

```

' функция, график которой будет построен
Function f(x As Single) As Single
f = 2 * Sin(x) * Exp(x / 5)
End Function
' процедура строит график функции
Sub DrawGraph()
' границы изменения аргумента функции:
Dim x1 As Single, x2 As Single
' границы изменения значения функции:
Dim y1 As Single, y2 As Single
Dim x As Single ' аргумент функции
Dim y As Single ' значение функции в точке x
Dim dx As Single ' приращение аргумента
Dim l, b As Integer ' левый нижний угол области
' вывода графика
Dim w, h As Integer ' ширина и высота области
' вывода графика
' масштаб по осям X и Y:
Dim mx As Single, my As Single
' точка начала координат:
Dim x0 As Integer, y0 As Integer
' область вывода графика
Form1.ScaleMode = 3
l = 10 ' X-координата левого
' верхнего угла
b = Form1.ScaleHeight - 20 ' Y-координата левого
' верхнего угла
h = Form1.ScaleHeight - 40 ' высота области вывода
w = Form1.ScaleWidth - 20 ' ширина области вывода
Глава 2. Графика 71

x1 = 0 ' нижняя граница диапазона аргумента
x2 = 25 ' верхняя граница диапазона аргумента
dx = 0.01 ' шаг аргумента
' нахождение максимального и минимального значений
' функции на отрезке [x1, x2]
y1 = f(x1) ' пусть это минимум
y2 = f(x2) ' пусть это максимум
X = x1
While (x <= x2)
Y = f(x)
If y < y1 Then y1 = y
' если нашлось меньшее значение, то считаем
' это значение минимальным и т. д.
If y > y2 Then y2 = y
' аналогично для максимума: если нашлось большее
' значение, то считаем это значение максимальным
' и т. д.
X = X + dx
Wend
' вычисление масштаба
my = h / Abs(y2 - y1) ' масштаб по оси Y
mx = w / Abs(x2 - x1) ' масштаб по оси X
' прорисовка координатных осей и подпись максимального
' и минимального значений

```

```

x0 = l
y0 = b - Abs(Round(y1 * my))
Line (l, b)-(l, b - h)
Line (x0, y0)-(x0 + w, y0)
Form1.CurrentX = l + 5
Часть I.Примеры 72

```

```

Form1.CurrentY = b - h
Print Format$(y2, "##0.000")
Form1.CurrentX = l + 5
Form1.CurrentY = b - 12
Print Format$(y1, "##0.000")
' построение графика
X = X1
While (x <= x2)
Y = f (x)
PSet (x0 + Round(x * mx), y0 - Round(y * my)), _
RGB(200, 0, 0)
x = x + dx
Wend
End Sub
' обработка события Paint
Private Sub Form_Paint()
Call DrawGraph
End Sub
' изменение размера окна программы
Private Sub Form_Resize()
Form1.Cls ' очистка формы
Call DrawGraph ' построение графика
End Sub

```

7. Программа выводит в диалоговое окно гистограмму (рис. 2.8) результат статистической обработки информации.

```

Dim n(1 To 5) As Integer ' количество пятерок,
четверок
' троек, двоек и
' общее количество оценок
Dim ncolor(2 To 5) ' цвета, соответствующие полям
' ввода данных
Dim ntext(2 To 5) ' подписи легенды
Глава 2. Графика 73

```

Гистограмма

Рис. 2.8. Окно программы Гистограмма

```

' Процедура строит гистограмму на поверхности
Sub gistr(Area As Object)
' высота и ширина области вывода диаграммы
Dim h As Integer, w As Integer
' наибольшее количество оценок
Dim max As Integer
' считывание количества оценок из полей ввода
' и определение их общего количества
n(5) = Val(Text5.Text)
n(4) = Val(Text4.Text)
n(3) = Val(Text3.Text)
n(2) = Val(Text2.Text)
n(1) = n(5) + n(4) + n(3) + n(2)
' определение наибольшего количества оценок
max = n(2)
For i = 3 To 5 Step 1
If max < n(i) Then max = n(i)

```

Next i

Часть I. Примеры 74

```
' проверка введенных данных
If (n(1) = 0) Or (max = 0) Then Exit Sub
' координаты задаются в пикселах
Area.ScaleMode = vbPixels
' определение рабочей области объекта Area
h = Area.ScaleHeight
w = Area.ScaleWidth
' очистка области от графики
Area.Cls
' установка размера шрифта
Area.Font.Size = 8
' вывод гистограммы на область Area
For i = 5 To 2 Step -1
' столбец
Area.Line (5 * (6 - i) + 35 * (5 - i), h)-
Step(35, -h *
(n(i) / max) * 0.8), ncolor(i), BF
Area.Line (5 * (6 - i) + 35 * (5 - i), h)-
Step(35, -h *
(n(i) / max) * 0.8), RGB(0, 0, 0), B
' подпись процента
Area.CurrentX = 5 * (6 - i) + 35 * (5 - i)
Area.CurrentY = h * (1 - (n(i) / max) * 0.8) - 15
Area.Print Format(n(i) / n(1), "0.0%")
Next i
' вывод легенды
ntext(5) = Left(Label5.Caption, Len(Label5.Caption)- 1)
ntext(4) = Left(Label4.Caption, Len(Label4.Caption)- 1)
ntext(3) = Left(Label3.Caption, Len(Label3.Caption)- 1)
ntext(2) = Left(Label2.Caption, Len(Label2.Caption)- 1)
```

Глава 2. Графика 75

```
For i = 5 To 2 Step -1
' обозначение (цвет)
Area.Line(5 * 6 + 35 * .4, h * 0.2 + 20 * (5 - i))
St*p(35, 15), ncolor(i), BF
Area.Line(5 * 6 + 35 * 4, h * 0.2 + 20 * (5 -
i))
St<p(35, 15), RGB(0, 0, 0), B
' подпись
Area.CurrentX = 5*7+35*5
Area.CurrentY = h * 0.2 + 20 * (5 - i)
Area.Print Format(ntext(i))
Next i
End Sub
' загрузка формы
Private Sub Form_Load()
' установка цвета фона компонента Picture1
Picture1.BackColor = RGB(255, 255, 255)
' очистка полей ввода информации
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
' установка цветов столбцов гистограммы
ncolor(5) = RGB(240, 0, 50)
ncolor(4) = RGB(0, 220, 0)
ncolor(3) = RGB(0, 50, 200)
ncolor(2) = RGB(245, 220, 0)
```


End Sub

' нажатие клавиши в поле Пятерок
Часть I.Примеры 76

```
Private Sub Text5_KeyPress(KeyAscii As Integer)
Select Case KeyAscii
Case 48 To 57, 8 ' цифры и <Backspace>
Case 13
Text4.SetFocus
Case Else
KeyAscii = 0
End Select
End Sub
```

' нажатие клавиши в поле Четверок

```
Private Sub Text4_KeyPress(KeyAscii As Integer)
Select Case KeyAscii
Case 48 To 57, 8 ' цифры и <Backspace>
Case 13 ' клавиша <Enter>
Text3.SetFocus
Case Else
KeyAscii = 0
End Select
End Sub
```

' нажатие клавиши в поле Троек

```
Private Sub Text3_KeyPress(KeyAscii As Integer)
Select Case KeyAscii
Case 48 To 57, 8 ' цифры и <Backspace>
Case 13 ' клавиша <Enter>
Text2.SetFocus
Case Else
KeyAscii = 0 ' остальные символы не
отображаются
End Select
End Sub
```

' нажатие клавиши в поле Двоек

```
Private Sub Text2_KeyPress(KeyAscii As Integer)
Select Case KeyAscii
остальные символы не отображаются
Глава 2. Графика 77
```

```
Case 48 To 57, 8 ' цифры и <Backspace>
Case 13 ' клавиша <Enter>
Command1.SetFocus
Case Else
KeyAscii = 0 ' остальные символы не отображаются
End Select
End Sub
```

```
' щелчок на кнопке Построить
Private Sub Command1_Click()
Call gistr(Picture1)
End Sub
```

8. Программа формирует на поверхности формы изображение идущих часов с часовой, минутной и секундной стрелками.

Окно программы показано на рис. 2.9.

Рис. 2.9. Окно программы Часы

```
Const R = 70 ' радиус циферблата часов
Const GRAD = 0.0174532 ' коэффициент пересчета угла
' из градусов в радианы
Const PI = 3.14159265
Dim x0 As Integer, y0 As Integer
```

Часть I. Примеры 78

```
Dim ahr As Integer ' положение стрелок
(угол)
Dim amin As Integer
Dim asec As Integer
' инициализация формы
Private Sub Form_Initialize()
Form1.Height = (Form1.Height - Form1.ScaleHeight) + _
(R + 10) * 2 * Screen.TwipsPerPixelY
Form1.Width < (Form1.Width - Form1.ScaleWidth) + _
(R + 10) * 2 * Screen.TwipsPerPixelX
' Form1.Height-высота формы, включая заголовок и границы
' Form1.ScaleHeight-высота формы без заголовка и границ,
' аналогично для Form1.Width и Form1.ScaleWidth.
' (R + 10) * 2 умножается на Screen.TwipsPerPixelY,
' т.к. Form1.Height задается в твипах, а не в пикселах,
' аналогично для (R + 10)*2 и Screen. TwipsPerPixelX.
x0 = R + 10
y0 = R + 10
' положение стрелок
ahr = 90 - Hour(Time) * 30 - (Minute(Time) / 12) * 6
amin = 90 - Minute(Time) * 6
asec = 90 - Second(Time) * 6
Timer1.Interval = 1000 ' период сигналов таймера - 1 с.
Timer1.Enabled = True ' запуск таймера
Form1.ScaleMode = 3
End Sub
' процедура вычерчивает вектор заданной длины из точки (x0, y0)
Sub Vector(x0 As Integer, y0 As Integer, _
a As Integer, l As Integer)
' x0, y0 - начало вектора
Глава 2. Графика 79

' a - угол между осью X и вектором
' l - длина вектора
Dim x, y As Integer ' координаты конца вектора
x = .Round(x0 + l * Cos(a * GRAD))
y = Round(y0 - l * Sin(a * GRAD))
Line (x0, y0)-(x, y)
End Sub
' процедура рисует стрелки
Sub DrawClock0
' шаг секундной и минутной стрелок - 6 градусов,
' часовой - 30.
' стирание изображений стрелок
Form1.DrawWidth = 3 ' задание толщины линии
Form1.ForeColor = Form1.BackColor
' часовая стрелка
Call Vector(x0, y0, ahr, R - 20)
' минутная стрелка
Call Vector(x0, y0, amin, R - 15)
' секундная стрелка
Call Vector(x0, y0, asec, R - 7)
```

```

' определение нового положения стрелок
ahr = SO - Hour(Time) * 30 - (Minute(Time)/12) * 6
amin = 90 - Minute(Time) * 6
asec = 90 - Second(Time) * 6
' прорисовка стрелок на новом положении
' часовая стрелка
Form1.DrawWidth = 3
Form1.ForeColor = RGB(0, 0, 0)
Call Vector(x0, y0, ahr, R - 20)

```

Часть I.Примеры 80

```

' минутная стрелка
Form1.DrawWidth = 2
Call Vector(x0, y0, amin, R - 15)
' секундная стрелка
Form1.DrawWidth = 1
Form1.ForeColor = RGB(200, 0, 0)
Call Vector(x0, y0, asec, R - 7)
End Sub
' прорисовка циферблата и начальных стрелок
Private Sub Form_Paint()
Dim x As Integer ' координаты маркера
Dim y As Integer ' на циферблате
Dim a As Integer ' угол между осью X и
' прямой (x0, y0) - (x, y)
Form1.DrawWidth = 1
Form1.ForeColor = RGB(0, 0, 0)
a = 0 ' метки ставим от 3-х часов, против
' часовой стрелки
' циферблат
While (a < 360)
x = x0 + Round(R * Cos(a * 2 * PI / 360))
y = x0 - Round(R * Sin(a * 2 * PI / 360))
If (a Mod 30) = 0 Then
Circle (x, y), 2
Else: Circle (x, y), 1
End If
a = a + 6 ' 1 минута - 6 градусов
Wend

```

Глава 2. Графика 81

```
Call DrawClock
```

```
End Sub
```

```
' прорисовка текущих положений стрелок часов
```

```
Private Sub Timer1_Timer ()
```

```
Call DrawClock
```

```
End Sub
```

9. Программа формирует на поверхности формы изображение идущих часов с часовой, минутной и секундной стрелками.

Окно программы приведено на рис. 2.10.

Рис. 2.10. Окно программы Часы

```
Const R = 60 ' радиус циферблата часов
```

```
Const GRAD = 0.0174532 ' коэффициент пересчета угла
```

```
' из градусов в радианы
```

```
Const PI = 3.14159265
```

```
Dim x0 As Integer, y0 As Integer
```

```
Dim ahr As Integer
```

Dim amin As Integer

Dim asec As Integer

Часть I.Примеры 82

' инициализация формы

Private Sub Form_Initialize()

Form1.Height = (Form1.Height - Form1.ScaleHeight) + _

(R + 30) * 2 * Screen.TwipsPerPixelY

Form1.Width = (Form1.Width - Form1.ScaleWidth) + _

(R + 30) * 2 * Screen.TwipsPerPixelX

x0 = R + 30

y0 = R + 30

' положение стрелок

ahr = 90 - Hour(Time) * 30 - (Minute(Time) / 12) * 6

amin = 90 - Minute(Time) * 6

asec = 90 - Second(Time) * 6

Timer1.Interval = 1000 *' период сигналов таймера - 1 с.*

Timer1.Enabled = True *' запуск таймера*

Form1.ScaleMode = 3

End Sub

' процедура вычерчивает вектор заданной длины из точки (x0, y0)

Sub Vector(x0 As Integer, y0 As Integer, _

a As Integer, l As Integer)

' x0, y0 - начало вектора

' a - угол между осью X и вектором

' l - длина вектора

Dim x, y As Integer *' координаты конца вектора*

x = Round(x0 + l * Cos(a * GRAD))

y = Round(y0 - l * Sin(a * GRAD))

Line (x0, y0)-(x, y)

End Sub

' процедура рисует стрелки

Sub DrawClock()

' шаг секундной и минутной стрелок - 6 градусов,

Глава 2. Графика 83

' часовой - 30.

' стирание изображений стрелок

Form1.DrawWidth = 3 *' задание толщины линии*

Form1.ForeColor = Form1.BackColor

' часовая стрелка

Call Vector(x0, y0, ahr, R - 20)

' минутная стрелка

Call Vector(x0, y0, amin, R - 15)

' секундная стрелка

Call Vector(x0, y0, asec, R - 7)

' определение нового положения стрелок

ahr = 90 - Hour(Time) * 30 - (Minute(Time) / 12) * 6

amin = 90 - Minute(Time) * 6

asec = 90 - Second(Time) * 6

' прорисовка стрелок на новом положении

' часовая стрелка

Form1.DrawWidth = 3

Form1.ForeColor = RGB(0, 0, 0)

Call Vector(x0, y0, ahr, R - 20)

' минутная стрелка

Form1.DrawWidth ~ 2

Call Vector(x0, y0, amin, R - 15)

' секундная стрелка

Form1.DrawWidth = 1

Form1.ForeColor = RGB(200, 0, 0)

```

Call Vector(x0, y0, asec, R - 7)
End Sub
' прорисовка циферблата и начальных стрелок
Private Sub Form_Paint()
Dim x As Integer ' координаты маркера
Dim y As Integer ' на циферблате

```

```

Часть I.Примеры 84

Dim a As Integer ' угол между осью X и
' прямой (x0, y0) - (x, y)
Dim h As Integer ' метка часовой риски
Form1.DrawWidth = 1
Form1.ForeColor = RGB(0, 0, 0)
a = 0 ' метки ставим от 3-х часов, против
' часовой стрелки
h = 3 ' угол 0 градусов - это 3 часа
' циферблат
While (a < 360)
x = x0 + Round(R * Cos(a * 2 * PI / 360))
y = x0 - Round(R * Sin(a * 2 * PI / 360))
If (a Mod 30) = 0 Then
Circle (x, y), 2
' вывод цифр по большему радиусу
CurrentX = x0 + Round((R + 15) * _
Cos(a * 2 * PI / 360)) - 7
CurrentY = x0 - Round((R + 15) * _
Sin(a * 2 * PI / 360)) - 7
Print h
h = h - 1
If h = 0 Then h = 12
Else: Circle (x, y), 1
End If
a = a + 6 ' 1 минута - 6 градусов
Wend
Call DrawClock

```

```

Глава 2. Графика 85

End Sub
' прорисовка положения стрелок часов
Private Sub Timer1_Timer()
Call DrawClock
End Sub

```

10. Программа демонстрирует принципы анимации и показывает, как можно заставить двигаться одно изображение на фоне другого. Окно программы приведено на рис. 2.11. Изображение объекта и фоновый рисунок (см. рис. 2.12) загружаются из файла.

Рис. 2.11. Окно программы Анимация

Рис. 2.12. Объект и фоновый рисунок

Часть I.Примеры 86

```

' инициализация формы
Private Sub Form_Initialize()
' загрузка картинки для самолета и фона
Picture1.BorderStyle = 0
Picture1.Picture = LoadPicture(CurDir + "\back.bmp")

```

```

Picture3.Borderstyle = 0
Picture3.Picture = LoadPicture(CurDir + "\plane. bmp")
Picture3.Visible = False
' Picture2 - это PictureBox, который обеспечивает
' движение самолета. В него загружается изображение
' самолета из Picture3 и часть фона из Picture1.
Picture2.Borderstyle = 0
Picture2.Height = Picture3.Height
Picture2.Width = Picture3.Width
Picture2.Left = Picture1.Left - Picture2.Width
Randomize ' инициализация генератора случайных
чисел
Picture2.Top = Int((500 * Rnd) + 1) +
Picture2.Height
Timer1.Interval = 10
Timer1.Enabled = True
End Sub
' обработка сигнала таймера
Private Sub Timer1_Timer ( )
Dim dx As Integer, dy As Integer
Dim npoint ' цвет точки
Picture2.Visible = False
If Picture2.Left > Picture1.Left + Picture1.Width Then
Picture2.Left = Picture1.Left - Picture2.Width
Picture2.Top = Int((500*Rnd)+1) + Picture2.Height
Глава 2. Графика 87

Else
Picture2.Left = Picture2.Left+25 ' двигаем
картинку
End If
' копирование определенной части фона (Picture1)
' на Picture2
Picture2.PaintPicture Picture1.Picture, _
-Picture2.Left, -Picture2.Top
' копирование самолета из Picture3 на Picture2
For dx = 1 To Picture3.Width Step Screen.TwipsPerPixelX
For dy = 1 To Picture3.Height Step
Screen.TwipsPerPixelY
' определение цвета точки
npoint = Picture3.Point(dx, dy)
' если цвет точки не белый (RGB (255, 255, 255)),
' то копируем точку из Picture3 на Picture2,
' т. о. по точкам копируем самолет.
If npoint <> RGB(255, 255, 255) Then
Picture2.PSet (dx, dy), npoint
End If
Next dy
Next dx
Picture2.Visible = True
End Sub
11. Программа выводит на поверхность формы анимационный
ролик, который загружается из графического файла. Форма
и окно программы приведены на рис. 2.13. Пример кадров
ролика показан на рис. 2.14.
Const f_name = "mult.trap" ' имя файла, содержащего

```

```

' кадры фильма
Const Nframe = 12 ' количество кадров в фильме
' (для данного файла)
Dim fw As Integer ' ширина и высота кадра
Dim fh As Integer

```

Часть I.Примеры 88

Рис. 2.14. Пример ролика (содержимое графического файла)

```

Dim currentFrame As Integer ' номер текущего кадра
' инициализация формы
Private Sub Form_Initialize()
Picture2.Visible = False
Picture2.AutoSize = True
' фильм загружается в Picture2 полностью, а затем
' по кадрам выводится в Picture1
Picture2.Picture = LoadPicture(CurDir + "\" + f_name)
fw = Round(Picture2.Width / Nframe)
fh = Picture2.Height
Picture1.Height = fh
Picture1.Width = fw
currentFrame = 1
Timer1.Interval = 100
Timer1.Enabled = True

```

Рис. 2.13. Форма и окно программы Мультипликация

Глава 2. Графика 89

```

End Sub
' обработка сигнала таймера
Private Sub Timer1_Timer ()
' если выводились все кадры - переход к первому
If currentFrame > Nframe Then currentFrame = 1
' вывод кадра
Picture1.PaintPicture Picture2.Picture, _
0, 0, , fw * (currentFrame - 1), 0, fw, fh
' PaintPicture выводит часть Picture2.Picture в
' Picture1.Picture:
' 0,0- координаты левого верхнего угла области вывода,
' fw * (currentFrame - 1), 0 - координаты начала
области,
' из которой будет копироваться изображение,
' fw, fh - размеры копируемой области
currentFrame = currentFrame + 1
End Sub

```

12. Программа, в окне которой прокручивается текст в стиле титров кинофильма (вместо текста может быть загружено любое графическое изображение). Форма и окно программы приведены на рис. 2.15.

Рис. 2.15. Форма и окно программы Прокрутка

```

Dim fw As Integer, fh As Integer
Dim x As Integer, y As Integer
' ширина и высота картинки

```

Часть I.Примеры 90

```

' щелчок на кнопке Ok
Private Sub Command1_Click()
Unload Form1
End Sub
' инициализация формы
Private Sub Form_Initialize()
Picture2.Visible = False
Picture2.AutoSize = True
Picture2.Picture = LoadPicture(CurDir + "\banner.bmp")
fw = Picture2.Width

```

```

fH = Picture2.Height
Form1.Width = Form1.Width + Picture2.Width -
Picture1.Width
Picture1.Width = Picture2.Width
x = 0
y = 0
Timer1.Interval = 50
Timer1.Enabled = True
End Sub
' обработка сигнала таймера
Private Sub Timer1_Timer ()
y = y + 10
' вывод части Picture1 в Picture2
Picture1.PaintPicture Picture2.Picture, _
-x, -y
' если выведенная часть картинке не полностью
' покрывает область вывода (Picture2), дорисовываем
' верхнюю часть картинку снизу
Глава 2. Графика 91

If y+Picture1.ScaleHeight >= Picture2.ScaleHeight Then
Picture1.PaintPicture Picture2.Picture, _
-0, -y + Picture2.ScaleHeight
End If
If y >= Picture2.ScaleHeight Then y = 0
End Sub
13. Программа отображает "бегущую" строку (строка — содер-
жимое графического файла). Окно программы приведено
на рис. 2.16.
Рис. 2.16. Окно программы Бегущая строка
Dim fW As Integer, fH As Integer ' ширина и высота
картинки
Dim x As Integer, y As Integer
' щелчок на кнопке Ok
Private Sub Command1_Click()
Unload Form1
End Sub
' инициализация формы
Private Sub Form1_Initialize()
Picture2.Visible = False
Picture2.AutoSize = True
Picture2.Picture = LoadPicture(CurDir + "\banner.bmp")
fW = Picture2.Width
fH = Picture2.Height
Form1.Height = Form1.Height + Picture2.Height - _
Часть I. Примеры 92

Picture1.Height
Command1.Top = Command1.Top + Picture2.Height - _
Picture1.Height
Picture1.Height = Picture2.Height
x = 0
y = 0
Timer1.Interval = 50
Timer1.Enabled = True
End Sub

```



```

' обработка сигнала таймера
Private Sub Timer1_Timer ( )
x = x + 40
' вывод части Picture1 в Picture2
Picture1.PaintPicture Picture2.Picture, _
-x, -y
' если выведенная часть картинки не полностью
' покрывает область вывода (Picture2), дорисовываем
' часть картинки рядом справа
If x + Picture1.ScaleWidth >= Picture2.ScaleWidth Then
Picture1.PaintPicture Picture2.Picture, _
-x + Picture2.ScaleWidth, -y
End If
if x >= Picture2.Width Then x=0
End Sub

```

14. Программа позволяет просмотреть иллюстрации, находящиеся в одном из каталогов компьютера. Окно программы приведено на рис. 2.17.

Глава 2. Графика 93

Рис. 2.17. Окно программы Просмотр иллюстраций

```

Dim dh As Integer
Dim dw As Integer
Dim direct As String
Dim fname As String
Dim ftype As String
' процедура выводит картинку в Image1
Sub show_pic(direct As String, first As Integer)
' first показывает, производился ли уже поиск файлов
' в папке direct: 1 -да, 0 - нет.
Dim resize As Single ' коэффициент пропорциональности
'при масштабировании
Image1.Visible = False

```

Часть I.Примеры 94

```

Image1.Picture = LoadPicture() ' выгрузка предыдущей
' картинки
Image1.Stretch = False
If first = 0 Then
' проверим, существует ли каталог, имя которого
' указано в поле ввода каталога
If Dir(direct, vbDirectory) = "" Then
Call MsgBox("Указанного каталога не существует.", _
vbOKOnly, "Просмотр иллюстраций")
Command1.Enabled = False
Exit Sub
Else
' имя 1-го найденного файла
fname = Dir(direct + ftype)
End If
End If
If fname <> "" Then
' файл найден
Command1.Enabled = True
Image1.Picture = LoadPicture(direct + fname)
' масштабирование
If Image1.Height > dh Or _

```

```
Image1.Width > dw Then  
Image1.Stretch = True  
resize = Image1.Width / Image1.Height  
If resize >= 1 Then ' длина рисунка больше ширины  
Глава 2. Графика 95
```

```
Image1.Width = dw ' или равна ей  
Image1.Height = dw / resize  
Else ' ширина рисунка больше длины  
Image1.Width = dh * resize  
Image1.Height = dh  
End If  
End If  
fname = Dir ' поиск следующего файла,  
' его имя записывается в fname  
If fname = "" Then Command1.Enabled = False  
' если следующий файл не найден, то кнопка Дальше  
' не доступна  
Else  
If first = 0 Then  
Call MsgBox("Файлов с данным расширением в " + _  
"указанном каталоге не обнаружено.", _  
vbOKOnly, "Просмотр иллюстраций")  
End If  
End If  
Image1.Visible = True  
End Sub  
' щелчок на кнопке Дальше  
Private Sub Command1_Click()  
' поиск следующего файла  
Call show_pic(direct, 1)  
End Sub  
' инициализация формы  
Private Sub Form_Initialize()  
Часть I.Примеры 96
```

```
ftype = "*.bmp"  
dw = Image1.Width  
dh = Image1.Height  
Command1.Enabled = False  
' путь к папке pictures в каталоге запущенной программы  
Text1.Text = CurDir + "\pictures\  
direct = Text1.Text  
' можно указать путь и к другой папке, например к той,  
' в которой установлен Windows:  
' Text1.Text = Environ("windir")  
' direct = Text1.Text + "\"  
Call show_pic(direct, 0)  
End Sub  
' выбор формата файлов - bmp  
Private Sub Option1_Click()  
ftype = "*.bmp"  
Call show_pic(direct, 0)  
End Sub  
' выбор формата файлов - jpg  
Private Sub Option2_Click()  
ftype = "*.jpg"
```

```

Call show_pic(direct, 0)
End Sub
' выбор формата файлов - gif
Private Sub Option3_Click()
ftype = "*.gif"
Call showjpic(direct, 0)
End Sub

```

Глава 2. Графика 97

```

' нажатие клавиши в поле ввода каталога
Private Sub Text1_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
direct = Text1.Text + "\"
Call show_pic(direct, 0)
End If
End Sub

```

15. Программа показывает, как можно получить фоновый рисунок путем многократного дублирования небольшой картинки, например, одного из стандартных рисунков Windows. Окно программы и битовый образ приведены на рис. 2.18.

Рис. 2.18. Окно программы **Фоновый рисунок** и его битовый образ
' загрузка формы

```

Private Sub Form_Load()
' фоновый рисунок - это первый найденный в папке
' Windows файл с расширением bmp;
' Environ("windir") - папка, в которую установлен Windows,
' Dir(Environ("windir") + "\*.bmp") - имя первого
' найденного в папке Windows файла с расширением bmp
Image1.Picture = LoadPicture(Environ("windir") + "\" + _
Dir(Environ("windir") + "\*.bmp"))
' фоновый рисунок можно загрузить, например, из папки,
' в которой находится программа:

```

Часть I. Примеры 98

```

' Image1.Picture=LoadPicture(CurDir+ "\background.bmp")
Image1.Visible = False
End Sub
' обработка события Paint
Private Sub Form_Paint()
Dim x As Integer ' координаты левого верхнего
угла
Dim y As Integer ' области, в которую выводится
Image1
For x = 0 To Form1.Width Step Image1.Width
For y = 0 To Form1.Height Step Image1.Height
Call Form1.PaintPicture(Image1, x, y)
Next y
Next x
End Sub

```

Глава 3. Мультимедиа 99

Глава 3

Мультимедиа

Общие замечания

Работу с анимацией и звуком обеспечивает компонент *Microsoft Multimedia Control* (MMControl).

При воспроизведении анимации и видео в качестве экрана

удобно ИСПОЛЬЗОВАТЬ компонент *PictureBox*.

Для управления громкостью воспроизведения звука необходимо использовать API-функции.

1. Программа "Звуки Windows" позволяет прослушать звуковые файлы, которые находятся в каталоге *Windows\Media*. Окно программы приведено на рис.3.1.

Рис. 3.1. Окно программы Звуки Windows

Часть I.Примеры 100

```
' для воспроизведения звуковых файлов в программе используется
' компонент Microsoft Multimedia Control (MMControl)
' щелчок в поле выбора файлов (File1)
Private Sub File1_Click()
Label1.Caption = "Выбранный файл:" + Chr(13) + _
File1.FileName
If File1.FileName <> "" Then
MMControl1.FileName = Environ("windir") + "\Media\" + _
File1.FileName
MMControl1.Command = "OPEN"
End If
End Sub
' инициализация формы
Private Sub Form_Initialize()
Label1.Caption = "Выберите звуковой файл и щелкните " + _
" на кнопке Play."
' папка, в которой находятся стандартные звуки Windows:
File1.Path = Environ("windir") + "\Media\"
' Environ ("windir") - путь к папке, в которую
' установлен Windows.
' Media - это подкаталог Windows, в котором содержатся
' стандартные звуки.
File1.Pattern = "*.wav"
' в поле выбора файлов (File1) будут отображаться файлы
' только с расширением wav
' в MMControl1 видны только кнопки Play и Stop
MMControl1.PauseVisible = False
MMControl1.PrevVisible = False
MMControl1.NextVisible = False
MMControl1.RecordVisible = False
```

Глава 3. Мультимедия 101

```
MMControl1.EjectVisible = False
MMControl1.BackVisible = False
MMControl1.StepVisible = False
End Sub
' действие MMControl1 завершено
Private Sub MMControl1_Done(NotifyCode As Integer)
' перемотка и остановка
MMControl1.Command = "PREV"
MMControl1.Command = "STOP"
End Sub
' выбор другого файла
Private Sub MMControl1_LostFocus()
' закрытие открытого файла
MMControl1.Command = "CLOSE"
End Sub
```

2. Программа, которая позволяет прослушать компакт-диск.

Во время воспроизведения в окне программы отображается номер воспроизводимого трека и время воспроизведения.

Окно программы приведено на рис. 3.2.

Рис. 3.2. Окно программы CD-player

Dim cd_id As Integer ' идентификация диска:

' 1 - сведения о диске выведены

Часть I.Примеры 102

```

' 0 - сведения не выведены
' процедура выводит сведения о диске
Sub cd_info()
Timer1.Enabled = False
MMControl1.Command = "CLOSE"
MMControl1.Command = "OPEN"
MMControl1.TimeFormat = 0
Label1.Caption = "00"
Label2.Caption = "00:00"
Label6.Caption = "00:00"
Command1.Caption = "Play"
If MMControl1.Tracks = 0 Then
Command1.Enabled = False ' кнопки управления
дискот
Command2.Enabled = False ' не доступны
Command3.Enabled = False
Label7.Caption = "В дисковоме нет диска."
Else
If MMControl1.Tracks <> 1 Then
Command1.Enabled = True ' кнопка Play доступна
Command2.Enabled = True ' кнопка Next доступна
Command3.Enabled = False ' кнопка Previous
' не доступна
' вывод сведений о диске
Label7.Caption = "всего треков на диске: " + _
Format(MMControl1.Tracks, "0#") + _
Chr(13) + _
"общее время звучания: " + _
toHMS(MMControl1.Length)
Глава 3. Мультимедия 103

Label1.Caption = Format(MMControl1.Track, "0#")
Label6.Caption = toHMS(MMControl1.TrackLength)
Else
Command1.Enabled = False ' кнопки управления
дискот
Command2.Enabled = False ' не доступны
Command3.Enabled = False
Label7.Caption = "Данный диск не является " + _
"музыкальным."
End If
End If
End Sub
' функция переводит длину трека, диска или текущую позицию
' воспроизводимого трека в формат "часы: минуты: секунды"
Private Function toHMS(time As Long) As String
Dim H As Integer ' часы
Dim M As Integer ' минуты
Dim S As Integer ' секунды
H = Int((time / 1000) / 3600)
M = Int((time / 1000 Mod 3600) / 60)
S = (time / 1000 Mod 3600) Mod 60
If H > 0 Then toHMS = Str(H) + ":"
toHMS = toHMS + Format(M, "0#") + ":" + Format(S,

```

```

"0#")
End Function
' щелчок на кнопке Play/Stop (воспроизведение/остановка)
Private Sub Command1_Click()
If Command1.Caption = "Play" Then
Timer1.Enabled = True
MMControl1.Command = "PLAY"
Command1.Caption = "Stop"
Часть I.Примеры 104

Else
Timer1.Enabled = False
Command1.Caption = "Play"
MMControl1.Command = "STOP"
MMControl1.Track = 1
Label1.Caption = Format(MMControl1.Track, "0#")
Label6.Caption = toHMS(MMControl1.TrackLength)
Label2.Caption = "00:00"
Command3.Enabled = False
Command2.Enabled = True
End If
End Sub
' щелчок на кнопке Next (следующая композиция)
Private Sub Command2_Click()
MMControl1.Command = "NEXT"
If MMControl1.Track = 2 Then Command3.Enabled = True
If MMControl1.Track = MMControl1.Tracks Then _
Command2.Enabled = False
Label6.Caption = toHMS(MMControl1.TrackLength)
Label1.Caption = Format(MMControl1.Track, "0#")
End Sub
' щелчок на кнопке Previous (предыдущая композиция)
Private Sub Command3_Click()
MMControl1.Command = "PREV"
If MMControl1.Track = 1 Then Command3.Enabled = False
If MMControl1.Track = MMControl1.Tracks - 1 Then _
Command2.Enabled = True
Label6.Caption = toHMS(MMControl1.TrackLength)
Label1.Caption = Format(MMControl1.Track, "0#")
Глава 3. Мультимедия 105

End Sub
' инициализация формы
Private Sub Form_Initialize ()
Label7.Alignment = 2 ' выравнивание по центру для поля
' вывода информации о диске
Timer1.Interval = 100
' тип воспроизводимых файлов - CDAudio
MMControl1.DeviceType = "CDAudio"
MMControl1.Visible = False
Call cd_info ' вывод сведений о диске
cd_id = 1
End Sub
' выгрузка формы
Private Sub Form_Unload(Cancel As Integer)
MMControl1.Command = "STOP"
MMControl1.Command = "CLOSE"

```

```

End Sub
' событие обновления для компонента MMControll
Private Sub MMControll_StatusUpdate()
' в дисковом находится диск, но сведения о нем еще
' не выведены
If MMControll1.Mode =525 And cd_id = 0 Then
Call cd_info ' вывод сведений о диске
cd_id = 1
End If
' в дисковом нет диска или дисковод открыт
If MMControll1.Mode =530 Then
Часть I.Примеры 106

Call cd_info
cd_id = 0
End If
End Sub
' обработка сигнала таймера
Private Sub Timer1_Timer()
' воспроизведение закончилось, остановка диска
If MMControll1.Position >= MMControll1.Length Then
MMControll1.Command = "STOP"
MMControll1.Track =1
Label1.Caption = Format(MMControll1.Track, "0#")
Label6.Caption = toHMS(MMControll1.TrackLength)
Label2.Caption = "00:00"
Command3.Enabled = False
Command2.Enabled = True
Timer1.Enabled = False
Exit Sub
End If
' диск воспроизводится
If MMControll1.Position - MMControll1.TrackPosition < _
MMControll1.TrackLength Then
Label2.Caption = toHMS(MMControll1.Position - _
MMControll1.TrackPosition)
' MMControll1.Position - текущая считываемая позиция,
' MMControll1.TrackPosition - позиция начала трека;
' оба свойства считаются от общего времени диска;
Else
' если трек закончился, переходим к следующему
MMControll1.Track = MMControll1.Track +1
Label1.Caption = Format(MMControll1.Track, "0#")

```

Глава 3. Мультимедия 107

```

Label6.Caption = toHMS(MMControll1.TrackLength)
Label2.Caption = toHMS(MMControll1.Position - _
MMControll1.TrackPosition)
If MMControll1.Track = 2 Then Command3.Enabled =
True
If MMControll1.Track = MMControll1.Tracks Then
Command2.Enabled = False
End If
End Sub

```

3. Программа "Аудио-плеер" позволяет прослушивать звуковые файлы. Имеется возможность регулировать громкость звука и

баланс непосредственно в диалоговом окне программы. Для изменения громкости и регулировки баланса используются компоненты *slider*. Выбор каталога, в котором находятся звуковые файлы, осуществляется в диалоговом окне **Выбор папки**. Для отображения списка файлов и дисков компьютера используются КОМПОНЕНТЫ *FileListBox* И *DirListBox*. ОКНО ПРОГРАММЫ ПОД именованием **Audio-player**, во время воспроизведения, и окно выбора папки приведены на рис. 3.3.

Рис. 3.3. Окна программы **Audio-player**
Часть I.Примеры 108

```
' Form1 - форма плеера, Form2 - форма выбора папки.
' Для того чтобы на панели ToolBox появились компоненты
' MMControl и Slider, необходимо в подпункте Components пункта
' меню Project выбрать компонент Microsoft Windows Common
' Controls 6.0 и Microsoft Multimedia Control 6.0
' обработка событий, процедуры и функции для
' формы плеера Form1
' функция возвращает значение громкости и баланса звука
Private Declare Function waveOutSetVolume Lib "winmm.dll" _
(ByVal uDeviceID As Long, ByVal dwVolume As Long) As Long
' функция устанавливает значение громкости и баланса звука
Private Declare Function waveOutGetVolume Lib "winmm.dll" _
(ByVal uDeviceID As Long, ByVal dwVolume As Long) As Long
' uDeviceID - ID(идентификатор) устройства, с которого
' считываем или для которого устанавливаем уровень звука.
' dwVolume: первые четыре разряда - уровень звука на правом
' канале, вторые четыре - на левом:
' 0000 - это 0, FFFF - это максимум.
' например: &HFFFFFFF - звук 100% на обоих каналах,
' &H0 - звук 0% на обоих каналах,
' &HFFFF0000 - звук 100% на правом канале, на левом - 0%.
' &HFFFF - 100% на левом канале, 0% - на правом.
Dim lVolume As Integer ' баланс звука (в процентах)
Dim rVolume As Integer
Dim rLevel As String ' уровень звука
Dim lLevel As String
' функция переводит считываемую позицию воспроизводимого
' файла
' в формат "минуты: секунды: сотые секунды"
```

Глава 3. Мультимедия 109

```
Private Function toHMS(time As Long) As String
Dim M As Integer ' минуты
Dim S As Integer ' секунды
Dim SS As Integer ' сотые секунды
M = Int((time / 1000 Mod 3600) / 60)
S = (time / 1000 Mod 3600) Mod 60
SS = (time / 10) Mod 100
toHMS = toHMS + Format(M, "0#") + ":" + _
Format(S, "0#") + ":" + Format(SS, "00")
End Function
' инициализация формы
Private Sub Form_Initialize()
File1.Pattern = "*.mp3"
```



```

' для того чтобы помимо mp3-файлов в списке отображались,
' например, файлы с расширением *.wav, нужно записать:
' File1.Pattern = "*.mp3; *.wav"
File1.Path = CurDir + "\mp3\"
MMControl1.Visible = False
MMControl1.TimeFormat = 0
Label1.Caption = "00:00:00"
Timer1.Interval = 100
Timer1.Enabled = False
If File1.ListCount = 0 Then
Command2.Enabled = False
Command3.Enabled = False
Command4.Enabled = False
End If
' уровень звука - 50% на обоих каналах
Часть I.Примеры 110

Slider1.Min = 0
Slider1.Max = 100
Slider1.Value = 50
' баланс звука - 100% на обоих каналах
Slider2.Min = -100
Slider2.Max = 100
rVolume = 100
lVolume = 100
' установка уровня звука и баланса
rLevel = Hex(65535 * (rVolume / 100) * _
(Slider1.Value / 100))
lLevel = Hex(65535 * (lVolume / 100) * _
(Slider1.Value / 100))
If Len(lLevel) < 4 Then
Do
lLevel = "0" + lLevel
Loop Until Len(lLevel) = 4
End If
Call waveOutSetVolume(0, "&H" & rLevel & lLevel)
End Sub
' щелчок на кнопке Open (выбор папки)
Private Sub Command1_Click()
' выводим форму выбора папки Form2, форму плеера Form1
' делаем недоступной
Form2.Show
Form1.Enabled = False
End Sub
' щелчок на кнопке Play/Stop (воспроизведение/остановка)
Private Sub Command2_Click()
Глава 3. Мультимедия 111

If File1.ListCount = 0 Then Exit Sub
If Command2.Caption = "Play" Then
' ListIndex - номер выбранного файла;
' если из списка выбора звуковых файлов File1 не выбран
' ни один из файлов, то ListIndex = -1
If File1.ListIndex = -1 Then File1.ListIndex = 0
' в MMControl1.FileName записывается полный путь
' к файлу, т. к. если записать только имя файла,

```

```

' m. e. MMControll.FileName = File1.FileName, то
' поиск файла с именем MMControll.FileName будет
' производиться в текущем каталоге программы,
' а подкаталоги не будут восприниматься
MMControll.FileName = File1.Path+"\ "+File1.FileName
MMControll.Command = "OPEN"
MMControll.Command = "PLAY"
Timer1.Enabled = True
Command2.Caption = "Stop"
Else
MMControll.Command = "STOP"
MMControll.Command = "CLOSE"
Timer1.Enabled = False
Command2.Caption = "Play"
Label1.Caption = "00:00:00"
End If
End Sub
' щелчок на кнопке Next (следующий файл)
Private Sub Command3_Click()
If File1.ListIndex < File1.ListCount - 1 Then

```

Часть I. Примеры 112

```

' в данный момент воспроизводится звуковой файл
If MMControll.Command = "PLAY" Then
MMControll.Command = "CLOSE"
File1.ListIndex = File1.ListIndex + 1
MMControll.FileName = File1.Path + "\" + _
File1.FileName
MMControll.Command = "OPEN"
MMControll.Command = "PLAY"
' звуковой файл не воспроизводится
Else
File1.ListIndex = File1.ListIndex + 1
MMControll.FileName = File1.Path + "\" + _
File1.FileName
End If
End If
End Sub

```

```

' щелчок на кнопке Prev (предыдущий файл)
Private Sub Command4_Click()
If File1.ListIndex > 0 Then
' в данный момент воспроизводится звуковой файл
If MMControll.Command = "PLAY" Then
MMControll.Command = "CLOSE"
File1.ListIndex = File1.ListIndex - 1
MMControll.FileName = File1.Path + "\" + _
File1.FileName
MMControll.Command = "OPEN"
MMControll.Command = "PLAY"
' звуковой файл не воспроизводится

```

Глава 3. Мультимедия 113

```

Else
File1.ListIndex = File1.ListIndex - 1
MMControll.FileName = File1.Path + "\" + _
File1.FileName

```

```

End If
End If
End Sub
' щелчок на списке звуковых файлов
Private Sub File1_Click()
' в данный момент воспроизводится файл
If Command2.Caption = "Stop" Then
MMControl1.Command = "CLOSE"
MMControl1.FileName = File1.Path + "\" + File1.FileName
MMControl1.Command = "OPEN"
MMControl1.Command = "PLAY"
End If
End Sub
' выгрузка формы
Private Sub Form_Unload(Cancel As Integer)
MMControl1.Command * "STOP"
MMControl1.Command *= "CLOSE"
End Sub
' прокрутка уровня звука
Private Sub Slider1_Scroll()
' установка уровня звука и баланса
rLevel = Hex(65535 * (rVolume / 100) * _
(Slider1.Value / 100))
lLevel = Hex(65535 * (lVolume / 100) * _
(Slider1.Value / 100))
If Len(lLevel) < 4 Then
Часть I.Примеры 114

Do
lLevel = "0" + lLevel
Loop Until Len(lLevel) =4
End If
Call waveOutSetVolume(0, "&H" & rLevel & lLevel)
End Sub
' изменение баланса звука
Private Sub Slider2_Scroll()
' установка баланса
If Slider2.Value = 0 Then ' на обоих каналах
поровну
lVolume = 100
rVolume =100
End If
If SIider2.Value > 0 Then ' на правом канале больше
lVolume = 100 - Slider2.Value
End If
If Slider2.Value < 0 Then ' на левом канале больше
rVolume = 100 - Abs(Slider2.Value)
End If
' установка уровня звука и баланса
rLevel = Hex(65535 * (rVolume / 100)*
(Slider1.Value / 100))
lLevel = Hex(65535 * (lVolume / 100)*
(Slider1.Value / 100))
If Len(lLevel) < 4 Then
Do
lLevel = "0" + lLevel
Loop Until Len(lLevel) = 4

```

```
End If
Call waveOutSetVolume(0, "&H" & rLevel & lLevel)
End Sub
' обработка сигнала таймера
Private Sub Timer1_Timer ()
If MMControl1.Position < MMControl1.Length Then
Label1.Caption = toHMS(MMControl1.Position)
Else
If File1.ListIndex < File1.ListCount - 1 Then
MMControl1.Command = "CLOSE"
File1.ListIndex = File1.ListIndex + 1
MMControl1.FileName = File1.Path + "\" + _
File1.FileName
MMControl1.Command = "OPEN"
MMControl1.Command = "PLAY"
Else
MMControl1.Command = "CLOSE"
If Check1.Value = 1 Then
File1.ListIndex = 0
MMControl1.FileName = File1.Path + "\" +
File1.FileName
MMControl1.Command = "OPEN"
MMControl1.Command = "PLAY"
Else
Timer1.Enabled = False
Command2.Caption = "Play"

```

Часть I. Примеры 116

```
Label1.Caption = "00:00:00"
File1.ListIndex = 0
End If
End If
End If
End Sub
' обработка событий, процедуры и функции для
' формы выбора папки Form2
' щелчок на кнопке Ok
Private Sub Command1_Click()
' передача пути к папке для списка звуковых файлов File1
' на форме плеера Form1
Form1.File1.Path = Dir1.Path
' в данный момент воспроизводится звуковой файл
If Form1.MMControl1.Command = "PLAY" Then
Form1.MMControl1.Command = "STOP"
Form1.MMControl1.Command = "CLOSE"
Form1.Timer1.Enabled = False
Form1.Command2.Caption = "Play"
Form1.Label1.Caption = "00:00:00"
If Form1.File1.ListCount <> 0 Then
Form1.File1.ListIndex = 0
Form1.MMControl1.FileName = Form1.File1.Path + _
"\ " +
Form1.File1.FileName
Form1.MMControl1.Command = "OPEN"

```

```

Form1.MMControl1.Command = "PLAY"
Form1.Timer1.Enabled = True
Form1.Command2.Caption = "Stop"
End If
End If
Unload Form2
End Sub
' щелчок на кнопке Cancel
Private Sub Command2_Click()
Unload Form2
End Sub
' смена диска
Private Sub Drive1_Change()
' В случае смены диска может возникнуть ошибка (выбор
' дисковода в том случае, если в нем нет диска). Для
' обработки этой ошибки используется переход
' к метке drive_error (смотри ниже).
On Error GoTo drive_error
Dir1.Path = Drive1.Drive + "\"
' обработка ошибок
drive_error:
' если возникла ошибка, то выбирается диск,
' выбранный раньше
Drive1.Drive = Dir1.Path
End Sub
' загрузка формы

```

Часть I. Примеры 118

```

Private Sub Form_Load()
Command1.Caption = "Ok"
Command2.Caption = "Cancel"
End Sub

```

выгрузка формы

```

Private Sub Form_Unload(Cancel As Integer)
Form1.Enabled = True
End Sub

```

4. Как правило, MP3-файл содержит подробную информацию (ID3v1 Tag) о музыкальном произведении: название, исполнитель, год выпуска альбома и др. Эта информация обычно отображается в окне программы воспроизведения. Программа "ID3v1 Tag Editor" позволяет редактировать ID3v1-теги MP3-файлов. Форма и окно программы приведены на рис. 3.4 и рис. 3.5.

Рис. 3.4. Форма программы редактора ID3v1 Tag Editor
Глава 3. Мультимедия 119

Рис. 3.5. Окно программы редактора ID3v1 Tag Editor

```

' определение нового типа id3tag
Private Type id3tag
Tag_Id As String * 3
Title As String * 30
Artist As String * 30
Album As String * 30
Year As String * 4
Comment As String * 30
Track As Integer
Genre As Byte

```

End Type

идентификация тега
название композиции
название исполнителя
название альбома
год издания
комментарий
номер трека
стиль исполнителя

Dim fPath As String

Dim gPath As String

' процедура считывает ID3v1 Tag из mp3-файла

Sub Id3v1tag(fPath As String)

Часть I.Примеры 120

Dim fNumber As Integer

Dim fTag As id3tag

fNumber = FreeFile

' FreeFile возвращает значение типа Integer в диапазоне
' от 1 до 511, представляющее собой следующий номер файла,
' доступный для использования инструкцией Open
' (не задействованный номер для открытия файла).
' открытие файла

Open fPath For Binary As fNumber

' Binary: 1 символ - это один байт

Seek #fNumber, LOF(fNumber) - 127

' Seek определяет текущую позицию чтения/записи
' в открытом файле. В данной записи устанавливается
' позиция чтения: LOF(fNumber) - 127;
' LOF(fNum) возвращает размер файла в байтах

Get #fNumber, , fTag.Tag_Id

' Get #fNumber, , fTag.Tag_Id осуществляет чтение
' информации (3-х символов, т.к. Tag_Id имеет длину
' 3 символа) из файла fNumber.
' Если в файле на позиции LOF (fNumber) - 121

содержится

' слово "TAG", данный mp3-файл содержит ID3v1 Tag,
' иначе - ID3v1 Tag не прописан.

' ID3v1 Tag имеет фиксированный размер в 128 байт:

' идентификация тега (TAG) - 3 символа,

' название песни - 30 символов,

' исполнитель - 30 символов,

' альбом - 30 символов,

' год - 4 символа,

' комментарий - 30 символов,

' жанр - 1 символ (байт).

Глава 3. Мультимедия 121

' Как определяется номер трека, будет написано ниже.

' Если слова в полях тега имеют меньшую длину, чем

' максимальная, то оставшиеся байты заполняются нулями

' в двоичном формате. Значение байта жанра - это номер,

' которому соответствует определенный жанр из списка

' жанров.

' ID3v2 Tag (вторая версия ID3 Tag) в файл записывается

' иначе:

' Идентификация ID3v2 Tag: если первым словом

' (3 символа) в открытом тр3 файле является слово ID3,
 ' то файл содержит ID3v2 Tag. В ID3v2 Tag добавлены
 ' следующие поля по сравнению с ID3v1 Tag: Composer,
 ' Orig.Artists, Copyright, URL, Encoded by.
 1 Кроме того, поля для ввода информации имеют не такой
 ' маленький размер по сравнению с ID3v1 Tag: 16 Мб -
 ' это максимальный размер для каждого поля ID3v2 Tag
 ' (фрейма), а максимальный размер всего Tag'a - 256Мб.
 ' Но ID3v2 Tag пытается использовать байты для
 ' информации как можно эффективнее, т. е. место для
 ' каждого фрейма не имеет фиксированной длины как

1 в ID3v1 Tag.

If fTag.Tag_Id = "TAG" Then

1 Title имеет длину 30 символов (указано при
 ' объявлении типа), поэтому Get ifNumber, , Title
 ' осуществляет чтение 30 символов начиная с
 ' позиции, которая установилась после чтения
 ' Tag_Id (Tag_Id имеет длину в 3 символа)

Get ttfNumber, , fTag.Title

' чтение остальной информации тега

Get ttfNumber, , fTag.Artist

Get #fNumber, , fTag.Album

Get ttfNumber, , fTag.Year

Часть I.Примеры 122

Get #fNumber, , fTag.Comment

Get #fNumber, , fTag.Genre

' Номер трека записывается в файл следующим образом:
 ' если поле для номера трека пусто, то в тег можно
 ' записать комментарий длиной 30 символов, если нет,
 ' то 30-м символом в комментарий дописывается номер
 ' трека + символ с кодом 0 перед ним. Это можно
 ' заметить, если в файл записать комментарий длиной
 ' ровно 30 символов и также записать номер трека.
 ' При последующем открытии в строчке комментария
 ' не будет хватать 2-х последних символов.
 ' В ID3v1-теге номер трека может быть от 1 до 255.
 ' Если, например, в программе WinAmp в поле номера
 ' трека ввести число 0 и сохранись, то номер трека
 ' при последующем открытии не будет отображаться.
 ' Если же ввести число, превышающее 255, то число
 ' будет преобразовано по модулю 256.

' определение номера трека:

If InStr(fTag.Comment, Chr(0)) Then _

fTag.Track = Asc(Right(fTag.Comment, 1))

1 InStrIfTag.Comment, Chr(0)) - поиск в строке

' fTag.Comment символа, код которого равен 0.

' Right(Right(fTag.Comment, 1) - усечение строки

' fTag.Comment до длины - 1 справа, т. е. чтение

' последнего символа строки.

' Функция Asc(...) возвращает код символа,

' возможные значения - от 0 до 255.

' при помощи тега кнопки Command.2. Tag помечаем,

' что в файле есть ID3v1 Tag

Command2.Tag =1

Else

Глава 3. Мультимедия 123

Call MsgBox(_

```

"ID3v1 Tag не найден, но его можно записать.",
-
vbOKOnly + vbInformation, "ID3v1 Tag не найден")
' При помощи тега кнопки Command2 помечаем,
' что в файле ID3v1 Tag не содержалось
Command2.Tag = 0
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Exit Sub
End If
' закрытие файла
Close #fNumber
' вывод информации:
Text1.Text = fTag.Title
Text2.Text = fTag.Artist
Text3.Text = fTag.Album
Text4.Text = fTag.Year
Text5.Text = fTag.Comment
If fTag.Track <> 0 Then
Text6.Text = fTag.Track
Else: Text6.Text = ""
End If
If fTag.Genre <= 147 Then
Часть I.Примеры 124

Combo1.ListIndex = fTag.Genre
Else: Combo1.ListIndex = 0
End If
End Sub
' процедура сохраняет ID3 Tag mp3-файла
Sub Save_Id3v1tag(fPath As String)
Dim fNumber As Integer
Dim fTag As id3tag
fNumber = FreeFile
' открытие файла
Open fPath For Binary As #fNumber
fTag.Tag_Id = "TAG"
fTag.Title = Text1.Text & _
String(30 - Len(Text1.Text), Chr(0))
fTag.Artist = Text2.Text & _
String(30 - Len(Text2.Text), Chr(0))
fTag.Album = Text3.Text & _
String(30 - Len(Text3.Text), Chr(0))
fTag.Year = Text4.Text & _
String(4 - Len(Text4.Text), Chr(0))
fTag.Track = Val(Text6.Text) Mod 256
If fTag.Track <> 0 Then
Text5.Text = Left(Text5.Text, 28)
fTag.Comment = Text5.Text & _
String(29 - Len(Text5.Text), Chr(0))
& Chr(fTag.Track)
Else
fTag.Comment = Text5.Text & _
String(30 - Len(Text5.Text), Chr(0))
Глава 3. Мультимедия 125

```



```

End If
If Combo1.ListIndex = -1 Then
fTag,Genre = 0
Else
fTag.Genre = Combo1.ListIndex
End If
' если в файле уже содержался ID3v1 Tag, то он
' записывается с позиции FileLen (fPath) -127,
' если нет, то с последней позиции (с конца
файла)
If Command2.Tag <> 0 Then
Put #fNumber, FileLen(fPath) - 127, fTag.Tag_Id
Else: Put #fNumber, FileLen(fPath), fTag.Tag_Id
End If
Put #fNumber, , fTag.Title
Put #fNumber, , fTag.Artist
Put #fNumber, , fTag.Album
Put #fNumber, , fTag.Year
Put #fNumber, , fTag.Comment
Put #fNumber, , fTag.Genre
' закрытие файла
Close #fNumber
End Sub
' щелчок на кнопке Open
Private Sub Command1_Click()
CommonDialog1.DialogTitle = "Выберите mp3-файл"
CommonDialog1.Filter = "mp3-файлы (*.mp3)|*.mp3"
CommonDialog1.FilterIndex = 1
CommonDialog1.Flags = cdlOFNNoChangeDir
' вывод стандартного окна Windows открытия файлов
Часть I.Примеры 126

CommonDialog1.ShowOpen
If CommonDialog1.FileName <> "" Then
Command2.Enabled = True
Command2.Tag = 0
fPath = CommonDialog1.FileName
Call Id3v1tag(fPath)
End If
End Sub
' щелчок на кнопке Save
Private Sub Command2_Click()
Call Save_Id3v1tag(fPath)
Call Id3v1tag(fPath)
End Sub
' щелчок на кнопке Exit
Private Sub Command3_Click()
Unload Me
End Sub
' инициализация формы
Private Sub Form_Initialize()
Dim fNumber As Integer
Dim curString As String
fNumber = FreeFile
Form1.Icon = LoadPicture(CurDir + "Veditor.ico")

```

```
Text1.Text=""
Text2.Text=""
Text3.Text=""
Text4.Text=""
Text5.Text=""
```

Глава 3. Мультимедия 127

```
Text6.Text = ""
Text1.MaxLength=30
Text2.MaxLength=30
Text3.MaxLength=30
Text4.MaxLength=4
Text5.MaxLength=30
Text6.MaxLength=3
Command2.Enabled = False
Command2.Tag =0
gPath = CurDir + "\id3v1__genre_list.txt"
' запись в Combo1 списка, возможных жанров
Open gPath For Input As #fNumber ' открытие файла
Do While Not EOF (fNumber)
Line Input #fNumber, curString ' считывание строки
' добавление элемента в Combo1
Combo1.AddItern Right(curString, Len(curString)-4), _
Val(Left(curString, 3))
Loop
' закрытие файла
Close #fNumber
' для того чтобы в Combo1 нельзя было вводить символы
' при выборе элемента, нужно перед запуском программы
' присвоить значение 2 - Dropdown list свойству Style.
End Sub
' нажатие клавиши в поле Year
Private Sub Text4_KeyPress(KeyAscii As Integer)
Часть I.Примеры 128

Select Case KeyAscii
Case 48 To 57, 8 ' код клавиш 0 - 9 и
Backspace
Case Else
KeyAscii = 0 ' остальные символы
запрещены
End Select
End Sub
' нажатие клавиши в поле Track
Private Sub Text6_Change()
Select Case KeyAscii
Case 48 To 57, 8 ' код клавиш 0 - 9 и
Backspace
Case Else
KeyAscii = 0 ' остальные символы запрещены
End Select
End Sub
```

Глава 4

Файлы

Общие замечания

При выполнении файловых операций возможны ошибки; для обработки ошибок выполнения файловых операций нужно использовать инструкцию *OnError*.

1. Программа "Чтение файла" выводит содержимое текстового файла в поле *Text*. Форма программы приведена на рис. 4.1.

Рис. 4.1. Форма программы Чтение файла

' Для того чтобы в файле, который открывается для просмотра, корректно, распознавались знаки перекома к следующей строке

Часть I. Примеры 130

```
' и текст не выводился одной строкой, необходимо свойству  
' MultiLine поля Text2 присвоить значение True. Это необходимо  
' сделать, до того, как программа будет запущена, т. к.  
' свойство MultiLine во время работы программы доступно только  
' для чтения. Также свойству ScrollBars необходимо присвоить  
' значение 2 - Vertical, тогда у Text2 появится  
' полоса вертикальной прокрутки.
```

```
Dim FileName As String ' путь к открываемому файлу
```

```
' щелчок на флаге "Только чтение"
```

```
Private Sub Check1_Click()
```

```
If Check1.Value = 1 Then
```

```
' флаг "Только чтение" установлен
```

```
Text2.Locked = True
```

```
Command2.Enabled = False
```

```
Else
```

```
' флаг "Только чтение" снят
```

```
Text2.Locked = False
```

```
Command2.Enabled = True
```

```
End If
```

```
End Sub
```

```
' щелчок на кнопке "Открыть"
```

```
Private Sub Command1_Click()
```

```
Dim current_string ' буфер чтения
```

```
FileName = Text1.Text
```

```
Text2.Text = "" ' удаление содержимого
```

```
' предыдущего файла
```

```
If Dir(FileName) = "" Or FileName = "" Or _
```

```
Len(FileName) <= 3 Then
```

```
' файл не найден
```

Глава 4. Файлы 131

```
FileName = ""
```

```
Call MsgBox("Ошибка доступа к файлу.", _
```

```
vbOKOnly, "Чтение файла")
```

```
Exit Sub
```

```
End If
```

```
' функция Input применяется для построчного чтения сим-ов
```

```
' из файла и вывода их в поле Text2.Text:
```

```
' Line Input #FileNumber, current_string -
```

```
' считывание строки,
```

```
' FileNumber - любой номер файла,
```

```
' current_string - переменная, в которую записывается
```

```

' считываемая строка ;
' функцию Input можно применять для чтения определенного
' количества символов :
' Input(Number, #FileNumber)
' Number - число возвращаемых символов или байтов ,
' FileNumber - любой номер файла ;
' http://
Open FileName For Input As #1 ' открытие файла
Do While Not EOF(1) ' EOF(1) - конец файла 1 (FileNumber),
' т.е. будет происходить чтение
' всего файла
Line Input #1, current_string ' считывание строки
Text2.Text = Text2.Text + current_string + vbCrLf
' vbCrLf - константа, которая осуществляет переход
' к новой строке
Loop
Close #1 ' закрытие файла
' если выбран флаг Только чтение
If Check1.Value = 1 Then
Text2.Locked = True ' запрет редактирования

```

Часть I. Примеры 132

```

' содержимого текстового файла
Command2.Enabled = False
Else
' флаг не выбран
Text2.Locked = False
Command2.Enabled = True
End If
End Sub
' щелчок на кнопке "Сохранить"
Private Sub Command2_Click()
If FileName = "" Then Exit Sub
If Dir(FileName) <> "" And Check1.Value = 0 Then
Open FileName For Output As #1 ' открытие файла для
' сохранения изменений
Print #1, Text2.Text
Close #1
End If
End Sub
' инициализация формы
Private Sub Form_Initialize()
Command2.Enabled = False
Text2.Locked = True
End Sub
' нажатие клавиши в поле редактирования
Private Sub Text1_KeyPress(KeyAscii As Integer)
' клавиша <Enter>
If KeyAscii = 13 Then Command1.SetFocus
End Sub

```

Глава 4. Файлы 133

2. Программа позволяет просматривать текстовые файлы. Окно программы приведено на рис. 4.2.

Рис. 4.2. Окно программы **Просмотр текстовых файлов**

```

' Для корректного отображения содержимого текстового файла
1 необходимо свойству MultiLine поля Text2 присвоить значение
' True. Свойству ScrollBars необходимо присвоить значение
1 2 - Vertical, тогда у Text2 появится полоса вертикальной
' прокрутки. Для того чтобы нельзя было редактировать
' содержимое файла, свойству Locked нужно присвоить

```

```
' значение True.  
Dim FileName As String ' путь к открываемому файлу  
' выбор каталога в поле Dir1  
Часть I.Примеры 134
```

```
Private Sub Dir1_Change( )  
File1.Path = Dir1.Path ' вывод списка файлов,  
содержащихся  
' в папке  
Text1.Text = "" ' очистка поля вывода  
' содержимого файла  
End Sub  
' смена диска  
Private Sub Drive1_Change( )  
' В случае смены диска может возникнуть ошибка ( выбор  
' дисковода в том случае, если в нем нет диска). Для  
' обработки этой ошибки используется переход  
' к метке drive_error ( смотри ниже).  
On Error GoTo drive_error  
Dir1.Path = Drive1.Drive + "\"  
File1.Path = Drive1.Drive  
Text1.Text = ""  
' обработка ошибок  
drive_error:  
' если возникла ошибка, то выбирается диск,  
' выбранный раньше  
Drive1.Drive = Dir1.Path  
End Sub  
' щелчок в поле списка файлов  
Private Sub File1_Click( )  
' Событие Click для File1 происходит тогда, когда  
' произошел щелчок на каком-либо файле в поле File1.  
' Если файлов в поле нет, то события Click не происходит,  
' т. е. проверить то, что имя файла File1.FileName не равно  
' "" при открытии файла не нужно.  
If Len(File1.Path) <> 3 Then  
Глава 4. Файлы 135
```

```
' если количество символов в File1.Path = 3,  
' то File1.Path - это корень диска (например "с:\").  
filepath = File1.Path + "\" + File1.FileName  
Else: filepath = File1.Path + File1.FileName  
End If  
Text1.Text = "" ' удаление из поля вывода содержимого  
' предыдущего файла  
Open filepath For Input As #1 ' открытие файла  
Text1.Text = Input(LOF(1), 1)  
' В этой программе используется способ чтения текстового  
' файла, при котором считывается сразу весь файл. Он  
' выполняется намного быстрее, чем построчный, который  
' использовался в программе "Чтение файла".  
' LOF(1)- длина файла, находящегося по адресу filepath.  
Close #1 ' закрытие файла  
End Sub  
' инициализация формы  
Private Sub Form_Initialize()  
Drive1.Drive = "C"  
File1.Pattern = "*.txt"  
Text1.Text = ""
```

End Sub

3. Программа "Поиск файла" производит поиск файла в указанном пользователем каталоге и его подкаталогах. Окно программы приведено на рис.4.3.

```
Dim cDir As String ' путь к папке, в которой будет  
' производиться поиск
```

```
Dim fileMask As String ' имя или маска для поиска
```

```
Dim n As Integer ' количество файлов, удовлетворяющих  
' параметрам поиска
```

Часть I.Примеры 136

Рис. 4.3. Окно программы Поиск файла

```
' поиск файла в указанном каталоге и его подкаталогах  
' осуществляется при помощи рекурсивной функции find  
Function find(dir_path As String)
```

```
Dim back_path As String ' обратный путь, если есть  
' подкаталоги
```

```
Dirl.Path = dir_path
```

```
File1.Path = dirjpath
```

```
' поиск в текущем каталоге
```

```
If File1.ListCount <> 0 Then
```

```
' File1.ListCount - количество файлов по адресу
```

```
' File1.Path
```

```
If Mid(fileMask, 1, 1) = "*" Then
```

```
' установка фильтра на список File1, если нужны  
все
```

```
' файлы с определенным расширением или любые файлы,
```

Глава 4. Файлы 137

```
' и вывод всех найденных файлов в список List1
```

```
File1.Pattern = fileMask
```

```
For i = 0 To (File1.ListCount - 1) Step 1
```

```
n = n + 1
```

```
' добавление найденного файла в список List1
```

```
' (в список добавляется полный путь к файлу)
```

```
If Len(File1.Path) <> 3 Then
```

```
List1.AddItem File1.Path + "\" + File1.List(i)
```

```
Else: List1.AddItem File1.Path + File1.List(i)
```

```
End If
```

```
' List1.AddItem File1.Path + File1.List(i) -
```

```
' добавление элемента File1.Path + File1.List (i)
```

```
' в список List1
```

```
Next i
```

```
Else
```

```
' если нужны файлы с конкретным именем и расширением
```

```
For i = 0 To (File1.ListCount - 1) Step 1
```

```
If File1.List(i) = fileMask Then
```

```
n = n + 1
```

```
' добавление найденного файла в список List1
```

```
If Len(File1.Path) <> 3 Then
```

```
List1.AddItem File1.Path + "\" + _
```

```
File1.List(i)
```

```
Else: List1.AddItem File1.Path + File1.List(i)
```

```
End If
```

```
End If
```

```
Next i
```

```
End If
```

Часть I.Примеры 138

```

End If
' заход в подкаталог каталога Dir1.Path осуществляется
' следующим образом:
i Dir1.List (i), где i - номер подкаталога;
' всего подкаталогов - Dir1.ListCount,
' их нумерация идет от 0 до Dir1.ListCount-1
' если есть подкаталоги
If Dir1.ListCount > 0 Then
back_path = Dir1.Path ' обратный путь
For j = 0 To (Dir1.ListCount - 1) Step 1
Dir1.Path = back_path
Call find(Dir1.List(j))
Next j
End If
' вывод количества найденных файлов
Label2.Caption = "Найдено файлов: " + Format(n)
End Function
' щелчок на кнопке "Найти"
Private Sub Coramand1_Click()
If Text1.Text <> "" Then
Dir1.Visible = False
Command1.Enabled = False
' очистка результата предыдущего поиска
n = 0
Label2.Caption = "Найдено файлов: " + Format(n)
List1.Clear
' считывание маски
Глава 4. Файлы 139

fileMask = Text1.Text
' определение пути к папке для поиска
If Len(Dir1.Path) > 3 Then ' если выбран не
корень
cDir = Dir1.Path + "\"
Else: cDir = Dir1.Path
End If
Call find(cDir)
Dir1.Path = cDir
Dir1.Visible = True
Command1.Enabled = True
If n = 0 Then
Call MsgBox("Файлов, удовлетворяющих параметру " +
"поиска не найдено.", , "Поиск файла")
End If
Else
Call MsgBox("Нужно ввести параметр поиска.", , _
"Поиск файла")
End If
End Sub
' смена диска
Private Sub Drive1_Change()
' В случае смены диска может возникнуть ошибка (выбор
' дисковода в том случае, если в нем нет диска). Для
' обработки этой ошибки используется переход
' к метке drive_error (смотри ниже).

```

```

On Error GoTo drive_error
Dir1.Path = Drive1.Drive + "\"
' обработка ошибок
Часть I.Примеры 140

```

```

drive_error:
' если возникла ошибка, то выбирается диск,
' выбранный раньше
Drive1.Drive = Dir1.Path
End Sub
' инициализация формы
Private Sub Form_Initialize()
Drive1.Drive = "c"
File1.Visible = False
End Sub
' нажатие клавиши в поле ввода имени файла или маски
' для поиска
Private Sub Text1_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then Coramand1.SetFocus
End Sub

```

Глава 5

Игры и полезные программы

В этой главе собраны программы-игры, а также другие полезные программы, которые могут пригодиться для практики программирования.

1. Всем известна логическая игра "15". Вот ее правила. В прямоугольной коробочке находятся 15 фишек, на которых написаны числа от 1 до 15. Размер коробочки 4x4, таким образом, в коробочке есть одна пустая ячейка. В начале игры фишки перемешаны (рис. 5.1). Задача игрока состоит в том, чтобы, не вынимая фишки из коробочки, выстроить фишки в правильном порядке (рис. 5.2). Программа "Игра 15" реализует описанную игру. Форма и окно программы приведены на рис. 5.3. Фишка, на изображении которой игрок делает щелчок левой кнопкой мыши, перемещается в пустую клетку.

Рис. 5.1. В начале игры фишки перемешаны Рис. 5.2. Правильный порядок фишек

Часть I.Примеры 142

Рис. 5.3. Форма и окно программы Игра 15

```

Const H = 4, W = 4
Const CH = 50, CW = 50
Dim stp(1 To W, 1 To H) As Integer ' правильное расположение
' фишек
Dim pole(1 To W, 1 To H) As Integer ' игровое поле
Dim ex As Integer, ey As Integer ' координаты пустой клетки
' новая игра
Sub newGame()
For i = 1 To W Step 1
For j = 1 To H Step 1
' правильное расположение фишек
stp(i, j) = (i - 1) * W + j
pole(i, j) = stp(i, j)
Next j
Next i
stp(W, H) = 0
' переметивание фишек и вывод поля

```



```
Call mixer
Call showPole
End Sub
' вывод игрового поля
Sub showPole()
Dim i As Integer, j As Integer
Dim X As Integer, Y As Integer ' x,y - координаты
вывода
' текста в клетке
' очистка поля от предыдущего хода
Form1.Cls
' сетка: вертикальные линии
For i = 1 To (W - 1)
Line (i * CW, 0)-(i * CW, CH * H)
Next i
' сетка: горизонтальные линии
For i = 1 To (H - 1)
Line (0, i * CH)-(CW * W, i * CH)
Next i
' содержимое клеток (цифры фишек)
For i = 1 To H
Y = (i - 1) * CH + 18
For j = 1 To W
X = (j - 1) * CW + 18
Select Case pole(i, j)
Case 0:
Case 1 To 9:
CurrentX = X
CurrentY = Y
Print " " + Format(pole(i, j))
Case 10 To 15:
CurrentX = X
CurrentY = Y

```

Часть I. Примеры 144

```
Print Format (pole(i, j))
End Select
Next j
Next i
End Sub
' функция проверяет, расположены ли фишки в нужном
порядке
Function Finish() As Boolean
Dim row As Integer, col As Integer
' координаты фишки
row=1
col=1
Finish = True ' пусть фишки расположены в нужном порядке
For i = 1 To (W * H - 1)
If pole(row, col) <> i Then
Finish = False
Exit Function
End If
' к следующей клетке
```

```

If col < W Then
col = col + 1
Else
col = 1
row = row + 1
End If
Next i
End Function
' процедура перемешивает фишки
Глава 5. Игры и полезные программы 145

```

```

Sub mixer ()
Dim x1 As Integer, y1 As Integer ' координаты пустой фишки
Dim x2 As Integer, y2 As Integer ' координаты фишки,
' перемещаемой на место
' пустой
Dim d As Integer ' направление перемещения,
' относительно пустой
' фишки
x1 = W
y1 = H
For i = 1 To W * H * 10 Step 1
Do
x2 = x1
y2 = y1
d = Int((Rnd * W) + 1)
' предполагается, что поле
' квадратное
Select Case d
Case 1: x2 = x2 + 1
Case 2: x2 = x2 - 1
Case 3: y2 = y2 + 1
Case 4: y2 = y2 - 1
End Select
Loop Until (x2 >= 1) And (x2 <= W) And
(y2 >= 1) And (y2 <= H)
pole(y1, x1) = pole(y2, x2)
pole(y2, x2) = 0
x1 = x2
y1 = y2
Next i
Часть I. Примеры 146

```

```

' запоминание координаты пустой клетки
ex = x1
ey = y1
End Sub
' процедура "перемещает" фишку в соседнюю пустую клетку,
если
' она, конечно, есть.
Sub fMove(cx As Integer, cy As Integer)
' ex, cy - клетка, в которой игрок сделал щелчок
' проверка возможности обмена
If Not ((Abs(cx - ex) = 1) And (cy - ey = 0) Or
-
(Abs(cy - ey) = 1) And (cx - ex = 0)) Then Exit
Sub
' обмен: перемещение фишки из (x, y) в (ex, ey)
pole(ey, ex) = pole(cy, cx)
pole(cy, cx) = 0
ex = cx

```

```

ey = cy
' вывод поля
Call showPole
If Finish = True Then
y_n = MsgBox("Цель достигнута!" + Chr(13) + _
"Еще раз?", vbYesNo, "Игра 15")
If y_n = vbYes Then Call newGame ' новая игра
If y_n = vbNo Then Unload Form1 ' завершение
' работы программы
End If
End Sub

```

Глава 5. Игры и полезные программы 147

```

' инициализация формы
Private Sub Form_Initialize ()
Form1.Width = (Form1.Width - Form1.ScaleWidth) + _
(CW * W) * Screen.TwipsPerPixelX
Form1.Height = (Form1.Height - Form1.ScaleHeight) + _
(CH * H) * Screen.TwipsPerPixelY
Form1.Font.Size = 10
Form1.ScaleMode = 3
Randomize
Call newGame
End Sub
' щелчок мыши на форме
Private Sub Form_MouseDown(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
' X, Y - координаты щелчка
Dim ex As Integer, ey As Integer ' координаты клетки
' преобразуем щелчок в координаты клетки
cx = Int(X / CW) + 1
cy = Int(Y / CH) + 1
Call fMove(cx, cy)
End Sub
' обработка события Paint
Private Sub Form_Paint()
Call showPole
End Sub

```

2. Игра *Puzzle* — аналог игры "15". Однако здесь игрок перемещает не фишки с цифрами, а фрагменты картинки. Цель игры — выстроить фишки в правильном порядке (собрать картинку). Окно программы приведено на рис. 5.4. Часть I. Примеры 148

Рис. 5.4. Окно программы *Puzzle* (Собери картинку)

```

Const H = 4, W = 4 ' размер поля - 4x4
Dim CH As Integer, CW As Integer ' размер клетки
Dim stp(1 To W, 1 To H) As Integer ' правильное расположение
' фишек
Dim pole(1 To W, 1 To H) As Integer ' игровое поле
Dim ex As Integer, ey As Integer ' координаты пустой клетки
' новая игра
Sub newGame()
For i = 1 To W Step 1
For j = 1 To H Step 1
' правильное расположение фишек
stp(i, j) = (i-1)* W + j

```

```

pole(i, j) = stp(i, j)
Next j
Next i

```

Глава 5. Игры и полезные программы 149

```

stp(W, H) = 0
' перемешивание фишек и вывод поля
Call mixer
Call showPole
End Sub
' вывод игрового поля
Sub showPole()
Dim i As Integer, j As Integer
Dim X As Integer, Y As Integer ' x, y - координаты вывода
' текста в клетке
' вывод фишки (фрагмент картинки)
For i = 1 To H
Y = (i - 1) * CH
For j = 1 To W
X = (j - 1) * CW
Select Case pole(i, j)
Case 0:
' вывод пустой клетки
Line (X, Y)-Step(CW, CH), _
Form1.BackColor, BF
Line (X, Y)-Step(CW, CH), RGB(0, 0, 0), B
Case Else:
Dim t ' контролирует переход к следующей
' строке фишек
t = pole(i, j) / W
' текущая фишка не последняя в строке
If t <> Int (pole(i, j) / W) Then
Form1.PaintPicture Picture1.Picture, _
Часть I.Примеры 150

X, Y, , , _
CW * (pole(i, j) - _
W * Int(pole(i, j) / W) - 1), _
CH * (Int(pole(i, j) / H)), _
CW, CH
Else
' последняя фишка в строке
Form1.PaintPicture Picture1.Picture, _
X, Y, , , _
CW * (pole(i, j) - _
W * Int (pole(i, j) / W) + 3), _
CH * (Int (pole(i, j) / H) - 1), _
CW, CH
End If
' контур фишки
Line (X, Y)-Step(CW, CH), RGB(0, 0, 0), B
End Select
Next j
Next i
End Sub
' процедура перемешивает фишки
Sub mixer ()
Dim x1 As Integer, y1 As Integer ' координаты пустой клетки
Dim x2 As Integer, y2 As Integer ' координаты фишки,
' перемещаемой на место
' пустой

```

```

Dim d As Integer ' направление перемещения,
' относительно пустой
' клетки
x1 = W
y1 = H

```

Глава 5. Игры и полезные программы 151

```

For i = 1 To W * H * 10 Step 1
Do
x2=x1
y2=y1
d = Int((Rnd * W) + 1) ' предполагается/ что поле
' квадратное
Select Case d
Case 1: x2 = x2 + 1
Case 2: x2 = x2 - 1
Case 3: y2 = y2 + 1
Case 4: y2 = y2 - 1
End Select
Loop Until (x2 >= 1) And (x2 <= W) And _
(y2 >= 1) And (y2 <= H)
pole(y1, x1) = pole(y2, x2)
pole(y2, x2) = 0
x1 = x2
y1 = y2
Next i
' запоминание координаты пустой клетки
ex = x1
ey = y1
End Sub
' процедура "перемещает" фишку в пустую соседнюю клетку, если
' она есть, конечно
Sub fMove(cx As Integer, cy As Integer)
' ex, cy - клетка, в которой игрок сделал щелчок

```

Часть I. Примеры 152

```

' проверка возможности обмена
I If Not ((Abs(cx - ex) = 1) And (cy - ey = 0) Or _
(Abs(cy - ey) = 1) And (ex - ex = 0)) Then
Exit Sub
' обмен - перемещение фишки из (x, y) в (ex, ey)
pole(ey, ex) = pole(cy, cx)
pole(cy, cx) = 0
ex = cx
ey = cy
' вывод поля
Call showPole
If Finish = True Then
y_n = MsgBox("Цель достигнута!" + Chr(13) + _
"Еще раз?", vbYesNo, "Игра 15")
If y_n = vbYes Then Call newGame ' новая игра
If y_n = vbNo Then Unload Form1 ' завершение
' работы программы
End If
End Sub

```

```

' функция проверяет, расположены ли фишки в нужном порядке
Function Finish() As Boolean
Dim row As Integer, col As Integer
' координаты фишки
row=1
col=1
Finish = True ' пусть фишки расположены в нужном порядке
For i = 1 To (W * H - 1)
If pole(row, col) <> i Then
Глава 5.Игры и полезные программы 153

Finish = False
Exit Function
End If
' к следующей клетке
If col < W Then
col = col + 1
Else
col = 1
row = row + 1
End If
Next i
End Function
' инициализация формы
Private Sub Form_Initialize()
' загрузка картинки
Picture1.Visible = False
Picture1.AutoSize = True
Picture1.BorderStyle = 0
Picture1.Picture = LoadPicture(CurDir +
"\picture.jpg")
' определение размера фишки
CH = Picture1.Height / H
CW = Picture1.Width / w
' установка размеров игрового поля
Form1.Width = (Form1.Width - Form1.ScaleWidth) + _
Picture1.Width
Form1.Height = (Form1.Height - Form1.ScaleHeight) + _
Picture1.Height
Randomize
Часть I.Примеры 154

Call newGame
End Sub
' щелчок мыши на форме
Private Sub Form_MouseDown(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
' X, Y - координаты щелчка
Dim cx As Integer, cy As Integer ' координаты
клетки
' преобразуем щелчок в координаты клетки
cx = Int(X / CW) + 1
cy = Int(Y / CH) + 1
Call fMove(cx, cy)
End Sub
' обработка события Paint

```

```

Private Sub Form_Paint( )
Call showPole
End Sub

```

3. Игра "Парные картинки" развивает внимание. Вот ее правила. Игровое поле разделено на клетки, за каждой из которых скрыта картинка. Картинки парные, то есть на игровом поле есть две клетки, в которых находятся одинаковые картинки. В начале игры все клетки "закрыты". Щелчок левой кнопкой мыши "открывает" клетку, в клетке появляется картинка. Теперь надо найти клетку, в которой находится такая же картинка. Щелчок в другой клетке открывает вторую картинку. Если картинки в открытых клетках одинаковые, то эти клетки "исчезают". Если разные — клетки остаются открытыми. Очередной щелчок закрывает открытые клетки и открывает следующую. Следует обратить внимание, что две открытые клетки закрываются даже в том случае, если в открытой клетке такая же картинка, как и в одной из двух уже открытых. Игра заканчивается, когда игрок откроет (найдет) все парные картинки.

Глава 5. Игры и полезные программы 155

Рис. 5.5. Файл с картинками

Рис. 5.6. Форма программы Парные картинки

Рис. 5.7. Окно программы Парные картинки

Часть I. Примеры 156

В приведенной игре все картинки квадратные и находятся в одном файле (рис. 5.5). Размер игрового поля (количество клеток по горизонтали и вертикали) определяется количеством картинок в файле. Зная высоту и ширину общей картинки в файле, программа вычисляет количество картинок и устанавливает соответствующий размер игрового поля. Форма программы приведена на рис. 5.6, а ее окно на рис. 5.7.

```

' Для того чтобы добавить в окно программы меню, нужно
' щелкнуть правой кнопкой в окне редактирования формы
' и в открывшемся меню выбрать пункт Menu Editor.
' В появившемся окне нужно создать два пункта:
' "Новая игра" (значение свойств: для Caption - Новая игра,
' для Name - new_game) ;
' "?" (значение свойств: для Caption - ?, для Name
- about).

```

```

Const MAX_SIZE = 32 ' максимальное кол. парных картинок

```

```

Const MAX_H = 8 ' максимальный размер поля - 8x8

```

```

Const MAX_W = 8

```

```

' объявление нового типа col_row

```

```

Private Type col_row

```

```

col As Integer

```

```

row As Integer

```

```

End Type

```

```

Dim Pole(1 To MAX_H, 1 To MAX_W) As Integer

```

```

' Pole(i,j) - код состояния клетки поля,

```

```

' Pole(i,j) < 100 - клетка закрыта,

```

```

' Pole(i,j) > 100 и < 200 - клетка открыта,

```

```

' игрок видит картинку,

```

```

' Pole(i,j) > 200 - игрок нашел пару для этой картинки

```

```

Dim n As Integer ' количество открытых пар картинок

```

```

Dim c As Integer ' количество открытых в данный момент

```

```
Dim open1 As col_row
Dim open2 As col_row
Dim W As Integer ' количество клеток поля
' по горизонтали
Dim H As Integer ' кол-во клеток поля по вертикали
' произведение N на H должно быть
' кратно 2
Dim WK As Integer ' ширина клетки
Dim HK As Integer ' высота клетки
' инициализация формы
Private Sub Form_Initialize()
' размеры задаются в пикселах
Form1.ScaleMode = vbPixels
Picture1.ScaleMode = vbPixels
' картинка загружается целиком в Picture1,
' а затем определенные части выводятся
' на форму методом PaintPicture
Picture1.BorderStyle = 0
Picture1.Visible = False
Picture1.AutoSize = True
Picture1.Picture = LoadPicture(CurDir +
"\pictures4x4.bmp")
' размер игрового поля
W = 4
H = 4
' размер клетки (фрагмента картинки)
Часть I. Примеры 158

HK = Picture1.Height
WK = Picture1.Width / Int(H * W / 2)
' установка размеров игрового поля
Form1.Width = (W * (WK + 5) - 5 + 1 - Form1.ScaleWidth)
*
Screen.TwipsPerPixelX + Form1.Width
Form1.Height = (H * (HK + 5) - 5 + 1 - Form1.ScaleHeight)
* Screen.TwipsPerPixelY + Form1.Height
' инициализация генератора случайных чисел
Randomize
' создание новой игры
Call NewGame
Timer1.Enabled = False
Timer1.Interval = 300
End Sub
' щелчок мыши на форме
Private Sub Form_MouseDown(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
Dim col_ As Integer ' номер клетки по горизонтали
Dim row_ As Integer ' номер клетки по вертикали
col_ = Int(X / WK) + 1
row_ = Int(Y / HK) + 1
If Pole(col_, row_) > 200 Then Exit Sub
' щелчок на месте одной из двух уже найденных
' парных картинок
```


' открытых клеток нет
Глава 5. Игры и полезные программы 159

```
If c = 0 Then  
c = 1  
open1.col = col_  
open1.row = row_  
' клетка помечается как открытая  
Pole(open1.col, open1.row) = _  
Pole(open1.col, open1.row) + 100  
' прорисовка клетки  
Call Kletka(open1.col, open1.row)  
Exit Sub  
End If  
' открыта одна клетка, открытие второй  
If c = 1 Then  
open2.col = col_  
open2 . row = row_  
' если открыта одна клетка и щелчок сделан  
' в этой же клетке, то ничего не происходит  
If (open1.col = open2.col) And _  
(open1.row = open2.row) Then  
Exit Sub  
Else  
c = 2 ' теперь открыты две клетки  
Pole(open2.col, open2.row) = _  
Pole(open2.col, open2.row) +100  
' прорисовка клетки  
Call Kletka(open2.col, open2.row)  
Часть I.Примеры 160  
  
' проверка, одинаковые ли открытые картинки  
If Pole(open1.col, open1.row) = _  
Pole(open2.col, open2.row) Then  
' открыты две одинаковые картинки  
n = n + 1  
Timer1.Enabled = True ' запуск таймера  
' процедура обработки события Timer уберет  
' две одинаковые картинки  
End If  
End If  
Exit Sub  
End If  
' открыты 2 клетки с разными картинками: они  
закрываются,  
' и открывается новая клетка, в которой сделан щелчок  
If c = 2 Then  
' закрытие открытых клеток  
Pole(open1.col, open1.row) = _  
Pole(open1.col, open1.row) - 100  
Pole(open2.col, open2.row) = • _  
Pole(open2.col, open2.row) - 100  
Call Kletka(open1.col, open1.row)  
Call Kletka(open2.col, open2.row)  
' запись в open1 номера текущей клетки  
open1.col = col_  
open1.row = row_  
Глава 5.Игры и полезные программы 161  
  
' увеличение счетчика открытых клеток
```

```

C = 1
' открытие текущей клетки
Pole(open1.col, open1.row) = _
Pole(open1.col, open1.row) + 100
Call Kletka(open1.col, open1.row)
End If
End Sub
' процедура рисует клетку поля
Sub Kletka (col As Integer, row As Integer)
Dim X As Integer, Y As Integer ' левый верхний угол
' клетки (координаты)
' преобразование координат клетки в координаты
' на поверхности формы
X = (col - 1) * (WK + 5)
Y = (row - 1) * (HK + 5)
If Pole(col, row) > 200 Then
' для этой клетки найдена пара, клетку нужно
' убрать с поля
Line (X, Y)-Step(WK, HK), Form1.BackColor, BF
End If
If (Pole(col, row) > 100) And (Pole(col, row) < 200) Then
' клетка открыта, нужно вывести картинку;
' Pole(col, row) - номер картинки + 100,
' 100 - признак того, что клетка открыта;
Часть I.Примеры 162

' вывод соответствующей части картинки в клетку
Form1.PaintPicture Picture1.Picture, _
X, Y, , , WK*(Pole(col, row) - 100 - 1), 0, WK, HK
' граница клетки
Line (X, Y)-Step(WK, HK), RGB(0, 0, 0), B
End If
If (Pole(col, row) > 0) And (Pole(col, row) <
100) Then
' клетка закрыта, рисуется только контур
Line (X, Y)-Step(WK, HK), Form1.BackColor, BF
Line (X, Y)-Step(WK, HK), RGB(0, 0, 0), B
End If
End Sub
' процедура рисует поле
Sub ShowPole()
Dim row As Integer, col As Integer
For row = 1 To H
For col = 1 To W
Call Kletka(col, row)
Next col
Next row
End Sub
' новая игра
Sub NewGame()
Dim k As Integer ' кол. парных _____ картинок
Dim r As Integer ' случайное число
Dim buf(1 To MAX_SIZE) As Integer
' в buf(i) записывается, сколько чисел i записано
Глава 5.Игры и полезные программы 163

```

```

' в массив Pole
k = Int(H * W / 2)
' обнуление массива buf(i)
For i = 1 To k
buf(i) = 0
Next i
n = 0
c = 0
' в массив Pole записываются случайные числа от 1 до k,
' каждое число должно быть записано два раза
For i = 1 To H
For j = 1 To W
DO
r = Int((k * Rnd) + 1) ' случайные числа
' от 1 до k
Loop Until buf(r) < 2
Pole(i, j) = r ' код картинки
buf(r) = buf(r) + 1
Next j
Next i
' теперь поле сгенерировано
Call ShowPole ' вывод поля
End Sub
' щелчок на пункте меню "Новая игра"
Private Sub new_game_Click()
Call NewGame
End Sub

```

Часть I. Примеры 164

```

' щелчок на пункте меню "?" (сведения о игре)
Private Sub about_Click()
Call MsgBox("Парные картинки 2004", vbOKOnly, _
"Парные картинки")
End Sub
' обработка события Paint
Private Sub Form_Paint()
Call ShowPole
End Sub
' обработка сигнала таймера
Private Sub Timer1_Timer ()
' сигнал таймера происходит тогда, когда найдены
' две одинаковые клетки;
' в массиве Pole эти клетки помечаются как совпавшие
Pole(open1.col, open1.row) = _
Pole(open1.col, open1.row) + 100
Pole(open2.col, open2.row) = _
Pole(open2.col, open2.row) + 100
c = 0
' прорисовка клетки
Call Kletka(open2.col, open2.row)
Call Kletka(open1.col, open1.row)
' остановка таймера
Timer1.Enabled = False
' если открыты все пары клеток (картинок)
If n = Int(W * H / 2) Then _
Call MsgBox("Game Over!", vbOKOnly, "Парные картинки")
End Sub

```

Глава 5. Игры и полезные программы 165

4. Игра "Сапер" развивает логическое мышление. Вот ее правила. Игровое поле состоит из клеток, в каждой из которых может быть мина. Задача игрока — найти все мины и пометить их флажками. Используя кнопки мыши, игрок может открыть клетку или поставить в нее флажок, указав тем самым, что в клетке находится мина. Клетка открывается щелчком левой кнопки, флажок ставится по щелчку правой кнопки мыши. Если в клетке, которую открыл игрок, есть мина, то происходит взрыв (сапер ошибся) и игра заканчивается (рис. 5.8). Если в клетке мины нет, то в ее поле появляется число, соответствующее количеству мин, находящихся в соседних клетках. Анализируя информацию о количестве мин в клетках, соседствующих с уже открытыми, игрок может обнаружить и пометить флажками все мины. Ограничений на количество клеток, помеченных флажками, нет, однако для завершения игры (в выигрышном варианте) флажки должны быть установлены только в тех клетках, в которых есть мины (ошибочно установленный флажок можно убрать, щелкнув правой кнопкой мыши в поле клетки, где _____ он был установлен).

В рассматриваемой программе изображения мины; флажка; мины, помеченной флажком; и клетки, в которой произошел взрыв, загружаются из файла (рис. 5.9) в компонент *picture1* и затем, во время игры, выводятся в соответствующие клетки игрового поля. Форма программы приведена на рис. 5.10.

```
Const MR = 10 ' количество клеток по вертикали
Const MC = 10 ' количество клеток по горизонтали
Const NM = 10 ' количество мин
Const W = 40 ' ширина клетки поля
Const H = 40 ' высота клетки поля
' минное поле
```

```
Dim Pole(0 To MR + 1, 0 To MC + 1) As Integer
' значение элемента массива:
' 0...8 - количество мин в соседних клетках,
' 9 - в клетке мина,
' 100..109 - клетка открыта,
' 200..209 - в клетку поставлен флаг
Часть I.Примеры 166
```

Рис. 5.8. Окно программы Сапер в конце игры

Рис. 5.9. Файл с изображениями состояний клетки

Рис. 5.10. Форма программы Сапер

Глава 5. Игры и полезные программы 167

```
Dim nMin As Integer ' количество найденных мин
Dim nFlag As Integer ' количество поставленных флагов
Dim status As Integer ' статус игры: 0 - начало игры,
' 1 - идет игра, 2 - результат игры
' загрузка формы
Private Sub Form_Load()
Dim row As Integer, col As Integer
' В неотображаемые элементы массива (клетки по границе
' игрового поля) записывается число -3. Это значение
' используется процедурой p_орел для завершения
' рекурсивного процесса открытия соседних пустых клеток
For row = 0 To MR + 1
For col = 0 To MC + 1
Pole(row, col) = -3
Next col
```

```

Next row
Call newGame ' новая игра
' установка размеров формы
Form1.Width = (Form1.Width - Form1.ScaleWidth) + _
(MC * W) * Screen.TwipsPerPixelX
Form1.Height = (Form1.Height - Form1.ScaleHeight) + _
(MR * H) * Screen.TwipsPerPixelY
' размеры объектов на форме задаются в пикселах
Form1.ScaleMode = vbPixels
' загрузка иконки для окна программы
Form1.Icon = LoadPicture(CurDir + "\saper.ico")
Picture1.BorderStyle = 0
Часть I.Примеры 168

```

```

Picture1.Visible = False
Picture1.AutoSize = True
Picture1.ScaleMode = vbPixels
' загрузка в Picture1 файла изображений для игры
Picture1.Picture = LoadPicture(CurDir + "\images.bmp")
' в файле находится 4 изображения, соответствующие
' состояниям клетки: изображение флага, мины, флага и мины,
' мины, на которой подорвались
End Sub
' процедура выводит на форму содержимое клетки
Sub Kletka(row As Integer, col As Integer, status As Integer)
Dim X As Integer, Y As Integer ' координаты верхнего
' левого угла области
' вывода клетки
X = (col - 1) * W
Y = (row - 1) * H
' начало игры
If status = 0 Then
' неоткрытая серая клетка
Line (X, Y)-Step(W, H), Form1.BackColor, BF
Line (X, Y)-Step(W, H), RGB(0, 0, 0), B
Exit Sub
End If
' идет игра
If Pole(row, col) < 100 Then
' неоткрытая серая клетка
Line (X, Y)-Step(W, H), Form1.BackColor, BF
Line (X, Y)-Step(W, H), RGB(0, 0, 0), B
' если игра завершена (status = 2) и клетка с миной
' была закрыта, открываем ее
Глава 5.Игры и полезные программы 169

```

```

If (status = 2) And (Pole(row, col) = 9) Then _
Call Mina(X, Y)
Exit Sub
End If
' открытие клетки
Line (X, Y)-Step(W, H), RGB(255, 255, 255), BF
Line (X, Y)-Step(W, H), RGB(0, 0, 0), B
' клетка открыта, в соседних клетках нет мин
If (Pole(row, col) = 100) Then Exit Sub
' в клетку поставлен флаг
If (Pole(row, col) >= 200) Then Call Flag(X, Y)

```

```

' клетка открыта, в соседних клетках есть мины
If (Pole(row, col)>=101) And (Pole(row, col)<=108) Then
Form1.Font.Size = 14
Form1.ForeColor = RGB(0, 0, 200)
Form1.CurrentX = X + 3
Form1.CurrentY = Y + 3
' вывод количества мин в соседних клетках
Print Str(Int(Pole(row, col) - 100))
Exit Sub
End If
' на этой мине подорвались
If (Pole(row, col) = 109) Then
Call MinaBoom(X, Y)
End If
' правильно поставленный флаг
If (Pole(row, col) = 209) And (status = 2) Then
Часть I.Примеры 170

Call Flag_Mina(X, Y)
End If
End Sub
' процедура выводит поле
Sub ShowPole(status As Integer)
Dim row As Integer, col As Integer
For row = 1 To MR
For col = 1 To MC
Call Kletka(row, col, status)
Next col
Next row
End Sub
' рекурсивная процедура открывает текущую и все соседние
' клетки, в которых нет мин
Sub n_open(row As Integer, col As Integer)
If Pole(row, col) = 0 Then
Pole(row, col) = 100
Call Kletka(row, col, 1)
' примыкающие клетки по вертикали и горизонтали
Call n_open(row, col - 1)
Call n_open(row - 1, col)
Call n_open(row, col + 1)
Call n_open(row + 1, col)
' примыкающие диагонально
Call n_open(row - 1, col - 1)
Call n_open(row - 1, col + 1)
Call n_open(row + 1, col - 1)
Call n_open(row + 1, col + 1)
Else
If (Pole(row, col) < 100) And (Pole(row, col)
<> -3) Then
Pole(row, col) = Pole(row, col) + 100
Глава 5.Игры и полезные программы 171

Call Kletka(row, col, 1)
End If
End If
End Sub

```

```

' процедура генерирует новое игровое поле
Sub newGame ()
Dim row As Integer ' координаты клетки
Dim col As Integer
Dim n As Integer ' количество поставленных мин
Dim k As Integer ' количество мин в соседних
клетках
' очистка игрового поля
For row = 1 To MR
For col = 1 To MC
Pole(row, col) = 0
Next col
Next row
' расстановка мин
Randomize ' инициализация ГСЧ
n = 0 ' количество мин
DO
row = Int((MR * Rnd) + 1)
col = Int((MC * Rnd) + 1)
If (Pole(row, col) <> 9) Then
Pole(row, col) = 9
n = n + 1
End If
Loop Until (n = NM)
' вычисление количества мин в соседних клетках
Часть I.Примеры 172

' для каждой клетки
For row = 1 To MR
For col = 1 To MC
If (Pole(row, col) <> 9) Then
k = 0
If Pole(row - 1, col - 1) = 9 Then k = k + 1
If Pole(row - 1, col) = 9 Then k = k + 1
If Pole(row - 1, col + 1) = 9 Then k = k + 1
If Pole(row, col - 1) = 9 Then k = k + 1
If Pole(row, col + 1) = 9 Then k = k + 1
If Pole(row + 1, col - 1) = 9 Then k = k + 1
If Pole(row + 1, col) = 9 Then k = k + 1
If Pole(row + 1, col + 1) = 9 Then k = k + 1
Pole(row, col) = k
End If
Next col
Next row
status = 0 ' начало игры
nMin = 0 ' нет обнаруженных мин
nFlag = 0 ' нет поставленных флагов
End Sub
' процедура выводит флаг
Sub Flag(X As Integer, Y As Integer)
' копирование изображения флага из Picture1 на форму
' методом PaintPicture
Form1.PaintPicture Picture1.Picture, _
X, Y, , , 0, 0, W, H
' граница клетки

```

```
Line (X, Y)- Step (W, H), RGB(0, 0, 0), B  
End Sub
```

Глава 5. Игры и полезные программы 173

```
' процедура выводит мину  
Sub Mina(X As Integer, Y As Integer)  
' копирование изображения мины из Picture1 на форму  
1 методом PaintPicture  
Form1.PaintPicture Picture1.Picture, _  
X, Y, , , W, 0, W, H  
' граница, клетки  
Line (X, Y)-Step (W, H), RGB(0, 0, 0), B  
End Sub  
' процедура выводит флаг и мину  
Sub Flag_Mina(X As Integer, Y As Integer)  
' копирование изображения флага и мины из Picture1 на  
форму  
' методом PaintPicture  
Form1.PaintPicture Picture1.Picture, __  
X, Y, , , 2 * W, 0, W, H  
' граница клетки  
Line (X, Y)-Step (W, H), RGB(0, 0, 0), B  
End Sub  
' процедура выводит мину, на которой подорвались  
Sub MinaBoom(X As Integer, Y As Integer)  
' копирование изображения мины, на которой подорвались  
' из Picture1 на форму методом PaintPicture  
Form1.PaintPicture Picture1.Picture, _  
X, Y, , , 3 * W, 0, W, H  
' граница клетки  
Line (X, Y)-Step (W, H), RGB(0, 0, 0), B  
End Sub  
' нажатие кнопки мыши на игровом поле  
Часть I. Примеры 174
```

```
Private Sub Form_MouseDown(Button As Integer, _  
Shift As Integer, X As Single, Y As Single)  
Dim row As Integer, col As Integer  
If status = 2 Then Exit Sub ' игра завершена  
If status = 0 Then status = 1 ' первый щелчок  
' преобразование координат мыши в индексы клетки поля  
row = Int(Y / H) + 1  
col = Int(X / W) + 1  
' нажатие левой кнопки мыши  
If Button = vbLeftButton Then  
If Pole(row, col) = 9 Then  
' открыта клетка, в которой есть мина  
Pole(row, col) = Pole(row, col) + 100  
status = 2 ' игра закончена  
Call ShowPole(status) ' вывод поля  
Else  
' открытие клетки  
If Pole(row, col) < 9 Then Call n_open(row, col)  
End If  
End If
```



```

' нажатие правой кнопки мыши
If Button = vbRightButton Then
' в клетке стоит флаг, пользователь хочет убрать
его
If Pole(row, col) >= 200 Then
nFlag = nFlag - 1
' уберем флаг из клетки
Pole(row, col) = Pole(row, col) - 200
' закрытие клетки
Глава 5. Игры и полезные программы 175

Call Kletka(row, col, status)
' в клетке нет флага, пользователь хочет его поставить
Else
' если клетка открыта, то флаг нельзя поставить,
' если клетка закрыта, то можно
If Pole(row, col) >= 100 Then Exit Sub
nFlag = nFlag + 1
Pole(row, col) = Pole(row, col) + 200 ' установка
' флага
Call Kletka(row, col, status) ' вывод флага
If Pole(row, col) = 209 Then
riMin = nMin + 1
' если все флаги расставлены на правильных местах
If (nMin = NM) And (nFlag = NM) Then
status = 2 ' игра закончена
Call ShowPole(status) ' вывод поля
End If
End If
End If
End If
End Sub
' обработка события Paint
Private Sub FormPaint()
' вывод игрового поля
Call ShowPole(status)
End Sub
Часть I. Примеры 176

```

```

' выбор пункта меню "О программе"
Private Sub about_Click()
Dim mes As String
mes = "ИГРА САПЕР" + vbCrLf + vbCrLf + _
"Программа демонстрирует работу с графикой," + _
vbCrLf + _
"массивами и использование рекурсии." + vbCrLf
Call MsgBox(mes, vbOKOnly + vbInformation, " О
программе" )
End Sub
' выбор пункта меню "Справка"
Private Sub help_Click()
' для вывода help-файла используется компонент CowmonDialog
CoramonDialog1.HelpCoramand = cdlHelpForceFile
CoirmonDialog1.HelpFile = CurDir + "\saper.hlp"
CoramonDialog1.ShowHelp
End Sub

```

```

' выбор пункта меню "Новая игра"
Private Sub new_game_Click()
Call newGame ' новая игра
Call ShowPole(status) ' вывод игрового поля
End Sub

```

5. Программа "Будильник". После того как пользователь установит время сигнала, задаст текст напоминания и щелкнет на кнопке *Ok*, окно программы исчезает с экрана. В установленное время на экране появляется окно с напоминанием. Появление окна сопровождается звуковым сигналом. Форма программы приведена на рис. 5.11.

```

' функция воспроизведения звукового файла
Private Declare Function PlaySound Lib "winram.dll" _
Alias "PlaySoundA" (ByVal lpszSoundName As String, _
ByVal hModule As Long, ByVal uFlags As Long) As Long
' lpszSoundName - имя файла или другой идентификатор,
Глава 5. Игры и полезные программы 177

```

Рис. 5.11. Форма программы Будильник

```

' hModule - номер модуля прикладной программы, содержащей звук
' (если данный параметр не требуется, то ему устанавливается
' значение 0),
' uFlags - флаги спецификации воспроизводимого файла,
' например:
' SND_ALIAS = &H10000 - воспроизведение системного звука,
' SND_ASYNC = &H1 - асинхронное воспроизведение, т. е.
' приложение не ждет завершения воспроизведения звука, а
' параллельно продолжает работу,
' SND_FILENAME = &H20000 - указание полного пути к файлу,
' SND_LOOP = &H8 - воспроизведение файла по кругу до тех пор,
' пока не будет вызвана команда остановки
' воспроизведения _____ звука,
' SND_NODEFAULT = &H2 - в случае, если указанный файл
' не найден, не проигрывается стандартный звук Windows,
' SND_PURGE = &H40 - остановка воспроизведения всех звуков,
' при этом поле lpszSoundName должно быть пусто (""),
' SND_SYNC = &H0 - синхронное воспроизведение, т. е.
' приложение ожидает завершения воспроизведения звука,
' прежде чем продолжить работу, и др.
Const SND_ALIAS = &H10000
Const SND_ASYNC = &H1
Const SND_FILENAME = &H20000
Часть I. Примеры 178

```

```

Const SND_LOOP = &H8
Const SND_NODEFAULT = &H2
Const SND_PURGE = &H40
Const SND_SYNC = &H0
Dim message As String ' сообщение, которое нужно вывести
Dim alarmTime ' время, на которое установлен
' будильник
' щелчок на кнопке Ok
Private Sub Command1_Click()
alarmTime = Label3.Caption
message = Text1.Text
' форма становится невидимой
Form1.Hide
End Sub
' инициализация формы

```

```

Private Sub Form_Initialize()
Label2.Font.Size = 14
Label3.Font.Size = 14
UpDown1.Min = 0
UpDown1.Max = 23
UpDown2.Min = 0
UpDown2.Max = 59
UpDown1.Wrap = True ' переход от максимального
' значения элемента UpDown1
' к минимальному и наоборот
UpDown2.Wrap = True ' аналогично для UpDown2
Timer1.Enabled = True
Timer1.Interval = 1000
Глава 5.Игры и полезные программы 179

Label2.Caption = Format(Time, "hh:mm")
End Sub
' нажатие клавиши в поле ввода сообщения
Private Sub Text1_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then Command1.SetFocus
End Sub
' обработка сигнала таймера
Private Sub Timer1_Timer()
If Form1.Visible = True Then
Label2.Caption = Format(Time, "hh:mm")
Else
' время срабатывания будильника
If Format(Time, "hh:mm") = alarmTime Then
' воспроизведение звука
Call PlaySound(CStr(CurDir) + "\ringer.wav", 0,
_
SND_FILENAME Or SND_ASYNC)
' вывод сообщения
If message <> "" Then
Call MsgBox(CStr(message), vbOKOnly,
"Будильник")
Else: Call MsgBox("Будильник!", vbOKOnly, _
"Будильник")
End If
' форма становится видимой
Form1.Show
End If
End If
End Sub
' изменение параметра Сигнал: часы
Private Sub UpDown1_Change()
Часть I.Примеры 180

Label3.Caption = Format(UpDown1.Value, "0#") + ":"
+ _Format(UpDown2.Value, "0#")
End Sub
' изменение параметра Сигнал: минуты
Private Sub UpDown2_Change()
Label3.Caption = Format(UpDown1.Value, "0#") + ":"
+ _Format(UpDown2.Value, "0#")
End Sub

```

6. Программа "Будильник". После того как пользователь устанавливает время сигнала и щелкнет на кнопке *Ok*, окно программы исчезает с экрана, а значок программы появится на системной панели (рис. 5.12). В установленное время на экране появляется окно программы. Появление окна сопровождается звуковым сигналом. Форма программы приведена на рис. 5.13.

Рис. 5.12. Значок программы **Будильник** на системной панели во время работы программы

® Будильник [x]

Рис. 5.13. Форма программы **Будильник**

Глава 5. Игры и полезные программы 181

```
' В этой программе при нажатии правой кнопки мыши на иконке
' в системной панели появляется меню, в котором содержится
' два пункта: Восстановить и Выход. Для того чтобы это было
' возможным, нужно щелкнуть правой кнопкой в окне
' редактирования формы и в открывшемся меню выбрать пункт
' Menu Editor. В появившемся окне нужно создать элементы:
' PopupSysTray, PopupExit и PopupRestore, причем PopupExit и
' PopupRestore должны быть подпунктами PopupSysTray. Свойству
' Visible элемента PopupSysTray нужно задать значение False.
' Свойству Caption элементов нужно присвоить соответствующие
' значения: Будильник, Восстановить и Выход.
' функция, позволяющая добавлять, удалять или изменять иконку
' в системном меню
```

```
Private Declare Function Shell_NotifyIcon Lib "shell32" _
Alias "Shell_NotifyIconA" _
(ByVal dwMessage As Long, pnid As NOTIFYICONDATA) As Boolean
' dwMessage - параметр, определяющий действие, которое будет
' совершаться (добавление иконки, удаление иконки, изменение
' ее параметров),
' lpData - параметр, в котором содержатся данные о том, какая
' иконка должна появиться в системном меню, какая надпись
' должна появиться при наведении мыши на иконку и др.
' объявление типа, который требуется функции Shell_NotifyIcon
Private Type NOTIFYICONDATA
cbSize As Long
hwnd As Long
uID As Long
uFlags As Long
uCallbackMessage As Long
hIcon As Long
szTip As String * 64
End Type
```

Часть I. Примеры 182

```
' константы, используемые функцией Shell_NotifyIcon
Const NIM_ADD = 0 ' добавление иконки в системное меню
Const NIM_MODIFY = 1 ' изменение параметров иконки
Const NIM_DELETE = 2 ' удаление иконки
' константы для действий кнопками мыши
Const WM_LBUTTONDOWN = &H201 ' левая кнопка: MouseDown
Const WM_LBUTTONUP = &H202 ' левая кнопка: MouseUp
Const WM_LBUTTONDOWNBLCLK = &H203 ' левая кнопка: DblClick
Const WM_RBUTTONDOWN = &H204 ' правая кнопка: MouseDown
Const WM_RBUTTONUP = &H205 ' правая кнопка: MouseUp
Const WM_RBUTTONDOWNBLCLK = &H206 ' правая кнопка: DblClick
' константы для типа NOTIFYICONDATA
Const NIF_MESSAGE = 1
Const NIF_ICON = 2
Const NIF_TIP = 4
Const WM_MOUSEMOVE = &H200
' функция воспроизведения звукового файла
```

```

Private Declare Function PlaySound Lib "winmm.dll" _
Alias "PlaySoundA" (ByVal lpszSoundName As String, _
ByVal hModule As Long, ByVal uFlags As Long) As Long
' lpszSoundName - имя файла или другой идентификатор,
' hModule - номер модуля прикладной программы, содержащей
звук
' (если данный параметр не требуется, то ему устанавливается
' значение 0),
' uFlags - флаги спецификации воспроизводимого файла,
Const SND_ALIAS = &H10000
Const SND_ASYNC = &H1
Const SND_FILENAME = &H20000
Const SND_LOOP = &H8
Глава 5.Игры и полезные программы 183

```

```

Const SND_NODEFAULT = &H2
Const SND_PURGE = &H40
Const SND_SYNC = &H0
Dim alarmTime ' время, на которое установлен
' будильник
Dim aSysIcon As NOTIFYICONDATA ' параметры иконки
' щелчок на кнопке Ok
Private Sub Command1_Click()
' считывание времени срабатывания будильника
alarmTime = Format(UpDown1.Value, "0#") + ":" + _
Format(UpDown2.Value, "0#")
' запись комментария в aSysIcon
aSysIcon.szTip = "Будильник: " & alarmTime &
vbNullChar
Shell_NotifyIcon NIM_MODIFY, aSysIcon
' скрывание формы
Form1.Hide
' добавление иконки в системную панель
Shell_NotifyIcon NIM_ADD, aSysIcon
' пункт меню "Восстановить" доступен
Form1.PopupRestore.Enabled = True
End Sub
' инициализация формы
Private Sub Form_Initialize()
Label1.Font.Size = 14
Label2.Font.Size = 14
Label3.Font.Size = 14
Label1.Alignment = 2 ' выравнивание по центру для поля
Часть I.Примеры 184

```

```

' отображения текущего времени
UpDown1.Min = 0
UpDown1.Max = 23
UpDown2.Min = 0
UpDown2.Max = 59
UpDown1.Wrap = True ' переход от максимального
' значения элемента UpDown1
' к минимальному и наоборот
UpDown2.Wrap = True ' аналогично для UpDown2
Timer1.Enabled = True ' и наоборот
Timer1.Interval = 1000
' вывод текущего времени
Label1.Caption = Format(Time, "hh:mm")
Form1.ScaleMode = vbPixels
' загрузка иконки для формы
Form1.Icon = LoadPicture(CurDir + "\alarm.ico")
' задание параметров иконки
With aSysIcon
.cbSize = Len(aSysIcon)
.hwnd = Form1.hwnd
.uID = 0&
.uFlags = NIF_ICON Or NIF_TIP Or NIF_MESSAGE

```

```
.uCallbackMessage = WM_MOUSEMOVE
.hlcon = Form1.Icon
.szTip = "Будильник: " & alarmTime & vbNullChar
End With
End Sub
Глава 5.Игры и полезные программы 185
```

```
' обработка сигнала таймера
Private Sub Timer1_Timer ()
If Form1.Visible = True Then
Label1.Caption = Format(Time, "hh:mm")
Else
' время срабатывания будильника
If Format(Time, "hh:mm") = alarmTime Then
' воспроизведение звука
Call PlaySound(CStr(CurDir) + "\ringer.wav", 0,
—
SND_FILENAME Or SNDJVSYNCH)
' форма становится видимой
Form1.Show
' пункт меню "Восстановить" недоступен
Form1.PopupRestore.Enabled = False
End If
End If
End Sub
' изменение параметра Сигнал: часы
Private Sub UpDown1_Change()
Label2.Caption = Format(UpDown1.Value, "0#")
End Sub
' изменение параметра Сигнал: минуты
Private Sub UpDown2_Change()
Label3.Caption = Format(UpDown2.Value, "0#")
End Sub
' изменение размеров окна формы
Private Sub Form_Resize()
' если нажата кнопка минимизировать/ то окно программы
Часть I.Примеры 186
```

```
' убирается с панели задач и добавляется иконка программы
' в системное меню
' WindowState возвращает значение состояния формы:
' vbNormal - обычное состояние формы (развернутое),
' vbMinimized - свернутое состояние (окно отображается
' на панели задач),
' vbMaximized-окно развернуто до максимально возможного
' размера
If Form1.WindowState = vbMinimized Then
alarmTime = Format(UpDown1.Value, "0#") + ":" + _
Format(UpDown2.Value, "0#")
' запись комментария в aSysIcon
aSysIcon.szTip="Будильник: "& alarmTime & vbNullChar
Shell_NotifyIcon NIM_MODIFY, aSysIcon
' скрывание формы
Form1.Hide
' добавление иконки программы в системное меню
Shell_NotifyIcon NIM_ADD, aSysIcon
' пункт меню "Восстановить" доступен
Form1.PopupRestore.Enabled = True
End If
End Sub
' контроль движения курсора и нажатия кнопок мыши
Private Sub Form_MouseMove(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
' щелчок мыши на иконке на панели инструментов
Select Case X
Глава 5.Игры и полезные программы 187
```

```

' щелчок левой кнопкой: значение X = 513
Case WM_LBUTTONDOWN
' восстановление формы
Form1.Windowstate = vbNormal
Form1.Show
' пункт меню "Восстановить" не доступен
Form1.PopupRestore.Enabled = False
' щелчок правой кнопкой: значение X = -516
Case WM_RBUTTONDOWN
' вывод системного меню
Form1.PopupMenu Form1.PopupSysTray
End Select
End Sub
' закрытие формы
Private Sub FormJUnload(Cancel As Integer)
' удаление иконки из системного меню
ShellJNotifyIcon NIM_DELETE, aSysIcon
End Sub
' щелчок на пункте меню "Выход"
Private Sub PopupExit_Click()
Shell_NotifyIcon NIM_DELETE, aSysIcon
Unload Me
End Sub
' щелчок на пункте меню "Восстановить"
Private Sub PopupRestore_Click()
Form1.Windowstate = vbNormal
Form1.Show

```

Часть I.Примеры 188

```

' пункт меню "Восстановить" не доступен
Form1.PopupRestore.Enabled = False
End Sub

```

7. Программа "Экзаменатор" предназначена для контроля знаний. Испытуемому предлагается тест — ряд вопросов, на которые он должен ответить путем выбора правильного ответа из нескольких вариантов (рис. 5.14).

Рис. 5.14. Окно программы Экзаменатор

Тест представляет собой текстовый файл определенной структуры. Имя файла теста задается в командной строке запуска программы (в окне **Запуск программы** после имени выполняемого файла или в поле **Объект** (тоже после имени выполняемого файла), если программа запускается щелчком на ярлычке, обозначающем программу). Ниже приведен пример файла теста.

```

Экономика
Оценка - ОТЛИЧНО!
6
Оценка - ХОРОШО.
5
Оценка - УДОВЛЕТВОРИТЕЛЬНО.
4
Оценка - ПЛОХО!
3
Карл Маркс написал книгу:
"Материализм и эмпириокритицизм"
О
Глава 5.Игры и полезные программы 189

```

```

"Как нам бороться с инфляцией"
О
"Капитал"
1
Что означает выражение "Делать бизнес"?
обманывать и хитрить
О
учиться в школе бизнесменов
О

```

заниматься конкретным делом, приносящим доход

1

Когда впервые появились бартерные сделки?

при первобытнообщинном строе

1

в период общественного разделения труда

0

в наше время

0

Слово "бухгалтер" переводится с немецкого как:

человек, держащий книгу

1

человек, считающий на счетах

0

человек, работающий с большой кипой бумаг

0

Как переводится с английского "ноу-хау", и что оно обозначает?

секрет

0

новое предприятие

0

новая идея (знаю как)

1

Конкуренция в переводе с латинского:

столкновение

1

соревнование

0

конкурс

0

Часть I.Примеры 190

Структура приведенного ранее теста следующая:

□ первая строка — заголовок теста;

□ после заголовка следует описание четырех уровней оценок;

Следует обратить внимание, что каждое сообщение, вопрос и ответ в файле теста должны представлять собой одну строку.

Dim q As Integer ' общее количество вопросов

Dim r As Integer ' количество правильных

' ответов

Dim rate(1 To 4) As Integer ' критерии оценок

Dim comment (1 To 4) As String ' комментарии

Dim f As String ' буфер чтения

' процедура считывает вопрос из файла и выводит на форму

Sub NextQuestion()

If Not EOF(1) **Then** ' файл не закончился

' считывание и вывод вопроса

Line Input #1, f

Label1.Caption = f

' считывание вариантов ответа

Line Input #1, f

Для каждого уровня задается сообщение и количество правильных ответов, необходимых для достижения этого уровня.

после уровней оценок следуют вопросы и варианты ответов;

После каждого альтернативного ответа стоит 1 или 0. Единица показывает, что данный вариант ответа — правильный, 0 — нет.

Глава 5.Игры и полезные программы 191

Option1.Caption = f

Line Input #1, f

Option1.Tag = f

' верный ли ответ: 1 - верный, 0 - нет;

' 0 или 1 записывается в свойство Tag соответствующего

' компонента Option

Line Input #1, f

Option2.Caption = f


```

Line Input #1, f
Option2.Tag = f
Line Input #1, f
Option3.Caption = f
Line Input #1, f
Option3.Tag = f
' ни один из вариантов ответа не выбран
Option1.Value = False
Option2.Value = False
Option3.Value = False
Command1.Enabled = False
' увеличение счетчика вопросов
q = q + 1
' если этот вопрос - последний
If EOF(1) Then Command1.Caption = "Завершить"
End If
End Sub
' щелчок на кнопке Дальше/Завершить/Снова
Private Sub Command1_Click()
Часть I.Примеры 192

If Option1.Value = True Then r = r + Option1.Tag
If Option2.Value = True Then r = r + Option2.Tag
If Option3.Value = True Then r = r + Option3.Tag
If Command1.Caption = "Снова" Then
Command1.Caption = "Дальше"
Label1.Height = Label1.Height / 2
Option1.Visible = True
Option2.Visible = True
Option3.Visible = True
' переход к началу файла
Seek #1, 1
Line Input #1, f
Form1.Caption = f
' комментарии и критерии оценок
For i = 1 To 4 Step 1
Line Input #1, f
comment(i) = f
Line Input #1, f
rate(i) = f
Next
' обнуление счетчиков
q = 0
r = 0
' вывод первого вопроса
NextQuestion
Exit Sub
End If
If Command1.Caption = "Завершить" Then
Глава 5.Игры и полезные программы 193

Option1.Visible = False
Option2.Visible = False
Option3.Visible = False
Label1.Height = Label1.Height * 2
Label1.Caption="Тестирование завершено." + Chr(13) +
"Правильных ответов: " + Format$(r)
+ " из " + _
Format$(q) + "."
i = 1
While (r < rate(i)) And (i < 4)
i = i + 1
Wend
' вывод комментария
Label1.Caption = Label1.Caption + Chr(13)+ comment(i)
Command1.Caption = "Снова"
Else
NextQuestion

```

```

End If
End Sub
' щелчок на кнопке Завершить
Private Sub Command2_Click()
Close #1 ' закрытие файла
Unload Form1
End Sub
' инициализация формы
Private Sub Form_Initialize()
Label1.Font.Size = 10
FileName = CurDir + "\test.txt"
Часть I.Примеры 194

Open FileName For Input As #1 ' открытие файла
' для чтения
Line Input #1, f ' чтение названия теста
Form1.Caption = f
' комментарии и критерии оценок
For i = 1 To 4 Step 1
Line Input #1, f
comment(i) = f
Line Input #1, f
rate(i) = f
Next
' обнуление счетчиков
q = 0
r = 0
' вывод первого вопроса
NextQuestion
End Sub
' выбор первого варианта ответа
Private Sub Option1_Click()
Command1.Enabled = True
End Sub
' выбор второго варианта ответа
Private Sub Option2_Click()
Command1.Enabled = True
End Sub
' выбор третьего варианта ответа
Private Sub Option3_Click()
Command1.Enabled = True
End Sub
Глава 5.Игры и полезные программы 195

```

8. Программа "Экзаменатор" предназначена для контроля знаний. Испытуемому предлагается тест — ряд вопросов, на которые он должен ответить путем выбора правильного ответа из нескольких вариантов ответа. Вопрос может сопровождаться иллюстрацией. Количество альтернативных ответов к вопросу может быть от 2 до 4. Пример окна программы приведен на рис. 5.15, форма программы показана на рис. 5.16. Вопросы теста, так же как и для предыдущей программы, находятся в текстовом файле. Файл теста состоит из разделов:

- название;
- заголовок;
- оценки;
- вопросы.

Название — строка текста. Название теста выводится в заголовке стартового окна программы тестирования.

Заголовок содержит общую информацию о тесте, например, о его назначении. Заголовок может состоять из нескольких строк.

Признаком конца заголовка является точка, стоящая в начале строки.

Пример заголовка файла теста приведен ниже.

Сейчас Вам будут предложены вопросы о знаменитых памятниках и архитектурных сооружениях Санкт-Петербурга, Вы должны из предложенных нескольких вариантов ответа выбрать правильный.

За заголовком следует раздел оценок, в котором приводятся названия оценочных уровней и количество баллов, необходимое для достижения этих уровней.

Название каждого уровня должно располагаться в одной строке.

Пример раздела оценок приведен далее.

Часть I.Примеры 196

Рис. 5.15. Пример окна программы **Экзаменатор**

Рис. 5.16. Форма программы **Экзаменатор**

Глава 5.Игры и полезные программы 197

Отлично

100

Хорошо

85

Удовлетворительно

60

Плохо

50

За разделом оценок следует раздел вопросов. Каждый вопрос начинается текстом вопроса, за которым в отдельной строке указывается имя файла иллюстрации (должно начинаться символом \) или точка (если к вопросу иллюстрации нет).

После вопроса следуют альтернативные ответы. Текст альтернативного ответа может занимать несколько строк. В строке, следующей за текстом варианта ответа, располагается оценка (количество баллов) за выбор этого ответа. Если альтернативный ответ не является последним для текущего вопроса, то перед оценкой ставится запятая, если последний — то точка. Пример вопроса

Архитектор Исаакиевского собора:

\isaak.bmp

Доменико Трезини

,0

Огюст Монферран

,1

Карл Росси

.0

В приведенном примере к вопросу есть иллюстрация, второй ответ — правильный.

Const N_LEV = 4 ' количество уровней оценки

Const N_ANS = 4 ' максимальное количество

' вариантов ответа

Dim fileName As String ' имя файла теста

Dim summa As Integer ' количество набранных очков

' (правильных ответов)

Часть I.Примеры 198

Dim vopros As Integer ' номер текущего вопроса

Dim otv As Integer ' номер выбранного ответа

Dim f As String ' буфер чтения

' сумма, соответствующая уровню оценки

Dim level(1 To N_LEV) As Integer

' сообщение, соответствующее уровню

Dim mes(1 To N_LEV) As String

' оценка за выбор ответа

Dim score(1 To N_ANS) As Integer

' щелчок на кнопке Ok (Дальше)

Private Sub Command1_Click ()

```

Select Case Command1.Tag
' вывод первого вопроса
Case 0:
Command1.Enabled = False
Call resetForm
Call voprosToScr
Command1.Tag = 1
Command1.Caption = "Дальше"
' вывод остальных вопросов
Case 1:
summa = summa + score (otv)
Command1.Enabled = False
Call resetForm
If Not EOF(1) Then
Call voprosToScr
Else
Глава 5.Игры и полезные программы 199

Close #1
Command1.Caption = "Ok"
Form1.Caption = "Результат"
Command1.Tag =2
Command1.Enabled = True
Call itog ' вывести результат
End If
' завершение работы
Case 2:
Unload Me
End Select
End Sub
' нажатие кнопки Выход
Private Sub Command2_Click()
Unload Me
End Sub
' инициализация _____ формы
Private Sub Form_Initialize()
fileName = CurDir + "\test.txt"
' открытие файла для чтения
Open fileName For Input As #1
Call resetForm
Call info ' вывод информации о тесте
Call getLevel ' чтение информации об оценках
Form1.ScaleMode = vbPixels
Command1.Tag = 0
Label1.Wordwrap = True
Часть I.Примеры 200

Label1.AutoSize = True
End Sub
' вывод информации о тесте
Sub info()
Dim buf As String
Line Input #1, f ' чтение названия теста
Form1.Caption = f ' вывод названия теста
buf = ""
Do ' чтение информации о тесте
Line Input #1, f
If Mid(f, 1, 1) = "0" Then _
buf = buf + f + vbCrLf
Loop Until Mid(f, 1, 1) = "."
' Mid(f, 1, 1) считывает из строки f с первой
позиции

```

```

' 1 символ
' вывод информации о тесте
Label1.Caption = buf
Command1.Caption = "Ok"
End Sub
' чтение информации об оценках за тест
Sub getLevel()
Dim i As Integer
i = 1
Do
Line Input #1, f
If Mid(f, 1, 1) <> "." Then
mes(i) = f ' сообщение
Глава 5.Игры и полезные программы 201

Line Input #1, f ' оценка
level(i) = f
i = i + 1
End If
Loop Until Mid(f, 1, 1) = "."
End Sub
' установки для начала теста
Sub resetForm()
' кнопки вариантов ответа и Picture1 невидимы
Option1.Visible = False
Option2.Visible = False
Option3.Visible = False
Option4.Visible = False
Option1.Caption = ""
Option2.Caption = ""
Option3.Caption = ""
Option4.Caption = ""
Option1.Value = False
Option2.Value = False
Option3.Value = False
Option4.Value = False
Image1.Visible = False
End Sub
' масштабирование иллюстрации
Sub showPicture()
Dim w As Integer, h As Integer ' максимально возможные
' размеры иллюстрации
' коэффициент пропорциональности при масштабировании
Dim resize As Single
Часть I.Примеры 202

Image1.Stretch = False
Image1.Top = Label1.Top + Label1.Height + 7
' вычисление допустимых размеров картинки
w = Form1.ScaleWidth - Label1.Left * 2
h = Command1.Top - Label1.Top - Label1.Height - 7*2
' размер области вывода иллюстрации зависит от количества
' вариантов альтернативных ответов - чем меньше
количество
' вариантов ответа, тем больше область
If Option1.Caption <> "" Then h = h - Option1.Height - 7

```

```

If Option2.Caption <> "" Then h = h - Option2.Height - 7
If Option3.Caption <> "" Then h = h - Option3.Height - 7
If Option4.Caption <> "" Then h = h - Option4.Height
- 7

```

```

' если размер картинке меньше w на h, то она
' не масштабируется
' масштабирование по длине

```

```

If (Image1.Height > h) Then
resize = Image1.Width / Image1.Height
Image1.Stretch = True
Image1.Width = h * resize
Image1.Height = h

```

```

End If

```

```

' масштабирование по ширине

```

```

If (Image1.Width > w) Then
Image1.Stretch = True
Image1.Width = w
Image1.Height = w / resize

```

```

End If

```

```

Image1.Visible = True

```

Глава 5. Игры и полезные программы 203

```

End Sub

```

```

' вывод вопроса

```

```

Sub voprosToScr( )

```

```

Dim i As Integer

```

```

Dim s As String, buf As String

```

```

Dim ifn As String ' файл иллюстрации

```

```

vopros = vopros + 1

```

```

Form1.Caption = "Вопрос " + Str(vopros)

```

```

buf = ""

```

```

' чтение вопроса

```

```

Do

```

```

Line Input #1, f

```

```

If Mid(f, 1, 1) <> "." And Mid(f, 1, 1) <> "\"

```

```

_Then

```

```

buf = buf + f + " "

```

```

End If

```

```

Loop Until Mid(f, 1, 1) = "." Or Mid(f, 1, 1) =
"\ "

```

```

' вывод вопроса

```

```

Label1.Caption = buf

```

```

' иллюстрация загружается, но выводится только после того,

```

```

' как будут прочитаны альтернативные ответы и определен

```

```

' максимально возможный размер области формы, который можно

```

```

' использовать для ее вывода

```

```

If Mid(f, 1, 1) <> "\" Then

```

```

Image1.Tag = 0 ' к вопросу нет иллюстрации

```

```

Else ' к вопросу есть иллюстрация

```

```

Image1.Tag = 1

```

```

ifn = Mid(f, 2)

```

```

' Mid(f, 2) - считывание всех символов из строки f

```

```

' начиная со 2-й позиции

```

Часть I. Примеры 204

```

' при чтении иллюстрации произошла ошибка

```

```

' (файл иллюстрации не найден)
On Error Resume Next
Image1.Picture = LoadPicture(CurDir + "\n + ifn)
If Err Then Image1.Tag = 0
End If
i = 1
' считывание вариантов ответа
DO
buf = ""
Do ' считывание текста варианта ответа
Line Input #1, f
If Mid(f, 1, 1) = "." And Mid(f, 1, 1) <> ","
-
Then
buf = buf + f + " "
End If
Loop Until Mid(f, 1, 1) = "." Or Mid(f, 1, 1) = ","
' прочитан альтернативный ответ
score(i) = Int(Mid(f, 2, 1))
Select Case i
Case 1: Option1.Caption = buf
Case 2: Option2.Caption = buf
Case 3: Option3.Caption = buf
Case 4: Option4.Caption = buf
End Select
i = i + 1
Loop Until Mid(f, 1, 1) = "."
' теперь прочитана иллюстрация и альтернативные ответы
Глава 5. Игры и полезные программы 205

' текст вопроса уже выведен, иллюстрация - нет
If Image1.Tag = 1 Then ' есть иллюстрация к
вопросу
Call showPicture
End If
' вывод альтернативных ответов
If Option1.Caption <> "" Then
If Image1.Tag = 1 Then
Option1.Top = Image1.Top + Image1.Height + 7
Else: Option1.Top = Label1.Top + Label1.Height + 7
End If
Option1.Visible = True
End If
If Option2.Caption <> "" Then
Option2.Top = Option1.Top + Option1.Height + 7
Option2.Visible = True
End If
If Option3.Caption <> "" Then
Option3.Top = Option2.Top + Option2.Height + 7
Option3.Visible = True
End If
If Option4.Caption <> "" Then
Option4.Top = Option3.Top + Option3.Height + 7
Option4.Visible = True
End If
End Sub
' определение достигнутого уровня

```

```

Sub itog()
Dim i As Integer
Dim buf As String
Часть I.Примеры 206

buf = "Результаты тестирования:" + vbCrLf +
"Всего баллов: " + Str(summa)
i = 1
While (summa < level(i)) And (i < N_LEV)
i = i + 1
Wend
buf = buf + vbCrLf + mes(i)
Label1.Caption = buf
End Sub
' выбор 1-го варианта ответа
Private Sub Option1_Click()
otv = 1
Command1.Enabled = True
End Sub
' выбор 2-го. варианта ответа
Private Sub Option2_Click()
otv = 2
Command1.Enabled = True
End Sub
' выбор 3-го варианта ответа
Private Sub Option3_Click()
otv = 3
Command1.Enabled = True
End Sub
' выбор 4-го варианта ответа
Private Sub Option4_Click()
otv = 4
Command1.Enabled = True
End Sub

```

Глава 6

Базы данных

Среда *Visual Basic* имеет развитые средства, обеспечивающие работу с базами данных.

Общие замечания

Создать базу данных (таблицу данных) и наполнить ее информацией можно при помощи утилиты *visData* — *Visual Data Manager*, которая входит в состав *Visual Basic*.

Для того чтобы программа могла работать с базой данных, на компьютере должен быть установлен драйвер соответствующей базы данных и компоненты, обеспечивающие работу с данными.

База данных должна быть зарегистрирована в системе. Зарегистрировать базу данных как источник данных *ODBC* можно при помощи системной утилиты — Администратор источников данных *ODBC*, доступ к которой можно получить через **Панель управления** (команда **Пуск ► Настройка ► Панель управления**).

1. Программа работы с локальной базой данных — "Склад" (см. рис. 6.1). Форма программы приведена на рис. 6.2. Для

управления процессом просмотра используется компонент *Adodc*, на поверхность которого помещены командные кнопки *Command1* (Добавить запись), *Command2* (Удалить запись) и *Command3* (Сохранить изменения).

Часть I.Примеры 208

Рис. 6.1. Окно программы *Магазин*, обеспечивающей работу с базой данных "Склад"

Рис. 6.2. Форма программы *Магазин*, обеспечивающей работу с базой данных "Склад"

База данных "Склад" содержит информацию о товарах. Создать базу данных можно в *Microsoft Access* или при помощи утилиты

Глава 6.Базы данных 209

Visual Data Manager. База данных, которая представляет собой таблицу (файл — *stock.mdb*), должна быть зарегистрирована в системе под именем *stock* как источник данных *ODBC* (см. табл. 6.1). Для доступа к базе данных используется компонент *Adodc*. Форма программы приведена на рис. 6.2.

Таблица 6.1. Поля таблицы *stock* базы данных "Склад"

Поле Тип Размер Информация

Title Text 50 Наименование

Cost Currency Цена

Comment Memo Описание

Photo Text 30 Файл иллюстрации (фото)

' размер области вывода иллюстраций

Const imH = 3600

Const imW = 3600

Dim gPath As String ' путь к файлам иллюстраций

' загрузка формы

Private Sub Form_Load ()

Dim p As Integer

' ограничения на количество символов

' для текстовых полей

Text1.MaxLength = 50

Text2.MaxLength = 8

Text4.MaxLength = 30

' ограничение на поле *Text3* не устанавливается, т. к.

' тип поля - Memo

Adodc1.ConnectionString = "DSN=stock"

' получить информацию о размещении БД

DataEnvironment1.Connection1.ConnectionString =

Часть I.Примеры 210

Adodc1.ConnectionString

DataEnvironment1.Connection1.Open

gPath = DataEnvironment1.Connection1.DefaultDatabase

DataEnvironment1.Connection1.Close

' DefaultDatabase - имя файла БД (без расширения)

' иллюстрации в том же каталоге, что и файл БД

p = InStrRev(gPath, "\") ' позиция символа (просмотр

' от конца строки)

gPath = Mid(gPath, 1, p)

' открыть базу данных

Adodc1.CommandType = adCmdText

Adodc1.RecordSource = "Select * From stock"

Adodc1.Refresh

' если в базе данных нет записей

If Adodc1.Recordset.RecordCount = 0 Then

```

' Command2.Enabled = False ' кнопка
"Сохранить"
' не доступна
Command3.Enabled = False ' кнопка "Удалить"
' не доступна
1 если в базе данных нет записей и пользователь
' попытался сохранить данные, предварительно не нажав
' кнопку "Добавить запись", то в программе возникнет
1 ошибка доступа по записи. Чтобы этого не произошло,
' предварительно добавляется запись. При этом если
' пользователь все-таки нажмет кнопку добавить, то двух
' новых записей не появится, т. к. если ни одно из полей
' не заполнено, то запись не добавляется.

```

```
Adodc1.Recordset.AddNew
```

```
End If
```

Глава 6. Базы данных 211

```
End Sub
```

```
' щелчок на кнопке "Добавить запись"
```

```
Private Sub Command1_Click()
```

```
Adodc1.Recordset.AddNew
```

```
' очистить область отображения
```

```
' иллюстрации
```

```
Image1.Picture = LoadPicture()
```

```
Label5.Visible = False
```

```
Text1.SetFocus
```

```
End Sub
```

```
' щелчок на кнопке "Сохранить"
```

```
Private Sub Command2_Click()
```

```
' если первая запись
```

```
If Adodc1.Recordset.RecordCount = 0 Then
```

```
Command3.Enabled = True ' теперь кнопка "Удалить запись"
```

```
' доступна
```

```
End If
```

```
Adodc1.Recordset.Update
```

```
' без этой строки после сохранения БД в полях автоматически
```

```
' появились бы данные первой записи.
```

```
' чтобы этого не происходило, нужно перейти в поле, которое
```

```
' сохраняется, т. е. последнее измененное.
```

```
' Adodc1.Recordset.Bookmark=Adodc1.Recordset.LastModified
```

```
End Sub
```

```
' щелчок на кнопке "Удалить запись"
```

```
Private Sub Command3_Click()
```

Часть I. Примеры 212

```
Dim r
```

```
r = MsgBox("Удалить запись?", vbQuestion + vbOKCancel,
"База данных")
```

```
If r = vbOK Then
```

```
If Adodc1.Recordset.RecordCount <> 0 Then
```

```
Adodc1.Recordset.Delete
```

```
If Adodc1.Recordset.EOF Then
```

```
Adodc1.Recordset.MoveNext
```

```
Else
```

```
Adodc1.Recordset.MoveLast
```

```

End If
End If
If Adodc1.Recordset.RecordCount = 0 Then
Adodc1.Recordset.AddNew
Command2.Enabled = False
Command3.Enabled = False
End If
End If
End Sub
' содержимое поля Label 6 изменяется
' при переходе к следующей или предыдущей записи
Private Sub Label6_Change()
Call ShowPicture_
End Sub
' нажатие клавиши в поле Наименование
Private Sub Text1_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then Text2.SetFocus
End Sub
' Нажатие клавиши в поле Цена
Глава 6. Базы данных 213

Private Sub Text2_KeyPress(KeyAscii As Integer)
Dim p As Integer ' позиция точки
' в поле Цена можно ввести только число
Select Case KeyAscii
Case Asc(0) To Asc(9) ' цифры
Case 9, 8 ' <Tab>, <Backspace>
Case 13
Text3.SetFocus ' <Enter>
Case Asc("."), Asc(",")
KeyAscii = Asc(".н")
n = InStrd(Text2.Text, ".")
If n <> 0 Then KeyAscii = 0
' остальные символы запрещены
Case Else
KeyAscii = 0
End Select
End Sub
' нажатие клавиши в поле Фото
Private Sub Text4_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Call ShowPicture_ ' показать иллюстрацию
Command2.SetFocus ' фокус на кнопку "Сохранить"
End If
End Sub
' показать иллюстрацию в поле Image1
Sub ShowPicture_()
Dim kx, ky, k ' коэффициенты масштабирования
' по x, y и общий
Часть I. Примеры 214

Image1.Visible = False
Image1.Stretch = False
Label5.Visible = False
If Label6.Caption = "" Then
Label5.Caption = "Иллюстрация не задана."

```

```

Label5.Visible = True
Exit Sub
End If
On Error GoTo er
Image1.Picture = LoadPicture(gPath + Label6.Caption)
Image1.Stretch = False
' вычислить коэффициенты масштабирования:
' по высоте
If Image1.Height > imH Then
ky = im / Image1.Height
Else
ky = 1
End if
' по ширине
If Image1.Width > imW Then
kx = imW / Image1.Width
Else
kx = 1
End If
If ky < kx Then
K = ky
Else

```

Глава 6. Базы данных 215

```

K = KX
End If
Image1.Width = imW * K
Image1.Height = imH * K
Image1.Stretch = True
Image1.Visible = True
er: ' ошибка загрузки картинки
Label5.Caption = "Ошибка доступа к файлу" + vbCrLf +
gPath + Label6.Caption
Label5.Visible = True
End Sub

```

2. Программа работы с базой данных "Записная книжка" демонстрирует отображение информации в табличной форме и использование SQL-запросов для выбора информации из базы данных. Создать базу данных (в формате *dBase*) можно при помощи утилиты *Visual Data Manager*. База данных, которая представляет собой таблицу *adrbk.dbf* (см. табл. 6.2), должна быть зарегистрирована в системе под именем *adrbk* как источник данных *ODBC*. Для доступа к базе данных используется компонент *Adodc*, для отображения информации — *DataGrid*. Форма программы приведена на рис. 6.3. В табл. 6.3 и 6.4 приведены значения свойств компонентов *Adodc1* и *DataGrid1*.

Таблица 6.2. Поля таблицы *adrbk* базы данных "Записная книжка"

Поле Тип Размер Информация

```

Name Text 15 Фамилия, имя
Tel Text 15 Телефон
Cell Text 15 Сотовый телефон
Email Text 30 Адрес электронной почты

```

Часть I. Примеры 216

Таблица 6.4. Значение свойств компонента

DataGrid1

Свойство Значение Комментарий

DataSource1 Adodc1 Источник данных, которые отображаются в поле компонента

Рис. 6.3. Форма программы "Записная книжка"
Глава 6. Базы данных 217

```
' Щелчок на кнопке Поиск
Private Sub Command1_Click()
Dim Keyword As String
Keyword = InputBox("Введите имя или фамилию" + vbCrLf
+ _
"( можно не полностью) ", "Поиск", " ")
If Keyword <> "" Then
' Примеры запросов :
' Select * From Adrbk Where Name Like 'Ку%'
' Select * From Adrbk Where Name Like '%Платон%'
Adodc1.RecordSource =
"SELECT * FROM adrbk where Name like " + _
Chr(39) + "%" + Keyword + "%" + Chr(39)
Adodc1.Refresh ' выполнить запрос
End If
End Sub
' Щелчок на кнопке Все записи
Private Sub Command2_Click()
Adodc1.RecordSource = "SELECT * From adrbk ORDER BY Name"
' MsgBox Adodc1.RecordSource
Adodc1.Refresh ' выполнить запрос
End Sub
```

3. Программа работы с базой данных "Ежедневник" демонстрирует отображение информации в табличной форме и использование SQL-запросов _____ для выбора информации из баз данных. Окно программы приведено на рис. 6.4. База данных "Ежедневник" (таблица *dp.dbf*) должна быть зарегистрирована в системе как источник данных ODBC (под именем *dp*). Создать базу данных (в формате *dBase*) можно при помощи утилиты *Visual Data Manager*. Форма программы приведена
Часть I. Примеры 218

Рис. 6.4. Окно программы "Ежедневник"
(результат выполнения запроса)

Рис. 6.5. Форма программы "Ежедневник"
Глава 6. Базы данных 219

на рис. 6.5. Для доступа к базе данных используется компонент *Adodc*, для отображения информации в табличной форме — компонент *DataGrid*. Значения свойств компонентов *Adodc* и *DataGrid* приведены в табл. 6.6 и 6.7, структура таблицы *dp* (базы данных) — в табл. 6.5.

Таблица 6.5. Поля таблицы базы данных "Ежедневник"

Поле Тип Размер Информация

Fwhen

Fwhat

Date/Time

Text 50

Дата проведения мероприятия

Мероприятие

Таблица 6.6. Значение свойств компонента *Adodc1*

Свойство Значение Комментарий

ConnectionString *DSN=dp* Команда подключения к

базе данных (задает источник данных)
CoinmandType adCmdText Команда — SQL-запрос
RecordSource SELECT fwwhat, SQL-команда: выбрать со-
fwwhen FROM держимое полей fwwhen и
dp ORDER BY fwwhat таблицы dp и упоря-
fwwhen дочитать записи по содер-
жимому поля fwwhen

Таблица 6.7. Значение свойств компонента DataGrid1

Свойство Значение Комментарий

DataSource1 Adodc1 Источник данных, которые
отображаются в поле ком-
понента

Option Explicit

Dim myMonthName (12)

Private Sub Form_Initialize()

Dim Today As Date

myMonthName(1) = "января" : myMonthName(2) = "февраля"

Часть I. Примеры 220

myMonthName(3) = "марта" : myMonthName(4) = "апреля"

myMonthName(5) = "мая" : myMonthName(6) = "июня"

myMonthName(7) = "июля" : myMonthName(8) = "августа"

myMonthName(9) = "сентября" : myMonthName(10) = "октября"

myMonthName(11) = "ноября" : myMonthName(12) = "декабря"

Today = Date

Label1.Caption = "Сегодня" + Str(Day(Today)) + _

myMonthName(Month(Today)) + ", " + _

WeekdayName(Weekday(Today, vbMonday), False, 0)

' Установить ширину столбцов таблицы.

' Размер компонентов измеряется в твипах.

' 300 и 70 - это ширина в пикселах (так привычнее)

DataGrid1.Columns(0).Width=70* Screen.TwipsPerPixelX

DataGrid1.Columns(1).Width=300* Screen.TwipsPerPixelX

' Клавиша <Tab> используется для перемещения

' курсора в строке таблицы

DataGrid1.TabAction = dbgGridNavigation

End Sub

' Дела, запланированные на сегодня

Private Sub Command1_Click()

Dim p As String ' параметр критерия запроса

Dim n ' количество записей, удовлетворяющих

' критерию запроса

Label2.Caption = "На сегодня"

p = Format(Date, "mm/dd/yy")

Глава 6. Базы данных 221

p = Replace(p, ".", "/")

p = "#" + p + "#"

Adodc1.RecordSource="SELECT DP.* From DP " + _

"WHERE (DP.FWHEN = " + p + ")"

Adodc1.Refresh ' выполнить запрос

If Adodc1.Recordset.RecordCount = 0 Then

MsgBox "На сегодня никаких дел не запланировано",

vbOKOnly, "Ежедневник"

End If

End Sub

' завтра и ближайшие дни

```

Private Sub Command2_Click()
Dim sDay As Date
Dim fDay As Date
Dim dow As Integer ' день недели (номер)
' параметры критерия SQL-запроса
Dim p1 As String
Dim p2 As String
sDay = Date
sDay = sDay + 1
fDay = sDay
dow = Weekday(sDay, vbMonday)
If dow >= 5 Then
Label2.Caption = "На завтра"
Else
Label2.Caption = "Завтра и ближайшие дни"
Часть I.Примеры 222

End If
Select Case dow
Case 6 ' завтра суббота
fDay = fDay + 2
Case 7 ' завтра воскресенье
fDay = fDay + 1
End Select
' в параметре запроса дата должна быть
' указана в виде: #mm/dd/yy#
p1 = Format (sDay, "imi/dd/yy")
p1 = "#" + Replace (p1, ".", "/" ) + "#"
p2 = Format(fDay, "mm/dd/yy")
p2 = "#" + Replace(p2, ".", "/" ) + "#"
Adodc1.RecordSource = "SELECT DP.* From DP " + _
"WHERE ( DP.FWHEN >= " + p1 + _
") and ( DP.FWHEN <= " + p2 + " ) ORDER
BY fwhen"
Adodc1.Refresh ' выполнить запрос
If Adodc1.Recordset.RecordCount = 0 Then
MsgBox "На завтра и ближайшие дни никаких дел" + _
vbCrLf + _ не запланировано", _
vbOKOnly, "Ежедневник"
End If
End Sub
' на этой неделе
Private Sub Command3_Click()
Глава 6.Базы данных 223

Dim sDay As Date
Dim fDay As Date
Dim dow As Integer ' день недели (номер)
' параметры критерия SQL-запроса
Dim p1 As String
Dim p2 As String
Label2.Caption = "На этой неделе"
sDay = Date ' получить текущую дату
dow = Weekday(sDay, vbMonday)
fDay = sDay + 7 - dow
' в параметре запроса дата должна быть

```

```

' указана в виде: #mm/dd/yy#
p1 = Format(sDay, "mm/dd/yy")
p1 = "#" + Replace(p1, ".", "/") + "#"
p2 = Format(fDay, "mm/dd/yy")
p2 = "#" + Replace(p2, ".", "/") + "#"
' Сформировать SQL-запрос
Adodc1.RecordSource = "SELECT DP.* From DP " + _
"WHERE ( DP.FWHEN >= " + p1 + _
") and ( DP.FWHEN <= " + p2 + ") ORDER BY
fwhen"
Adodc1.Refresh ' выполнить запрос
If Adodc1.Recordset.RecordCount = 0 Then
MsgBox "На эту неделю никаких дел не запланировано" , _
vbOKOnly, "Ежедневник"

```

End If

End Sub

```

' Все, что намечено сделать (все записи)
Private Sub Command4_Click()
Label2.Caption = "Все, что намечено сделать"
Adodc1.RecordSource =
' "SELECT fwhen,fwhat FROM (dp) ORDER BY fwhen"
Adodc1.Refresh

```

End Sub

' след. неделя

```

Private Sub Command5_Click()
Dim sDay As Date ' понедельник следующей недели
Dim fDay As Date ' воскресенье следующей недели
Dim dow As Integer ' день недели (номер)
' параметры критерия SQL-запроса
Dim p1 As String
Dim p2 As String
Label2.Caption = "На следующей неделе"
sDay = Date
dow = Weekday(sDay, vbMonday)
sDay = sDay + 8 - dow
fDay = sDay + 6
' в параметре запроса дата должна быть
' указана в виде: #mm/dd/yy#

```

Глава 6. Базы данных 225

```

p1 = Format(sDay, "rran/dd/yy")
p1 = "#" + Replace(p1, ".", "/") + "#"
p2 = Format(fDay, "mm/dd/yy")
p2 = "#" + Replace(p2, ".", "/") + "#"
' Сформировать SQL-запрос
Adodc1.RecordSource = "SELECT DP.* From DP " + _
"WHERE ( DP.FWHEN >= " + p1 + _
") and ( DP.FWHEN <= " + p2 + ") ORDER BY fwhen"
Adodc1.Refresh ' выполнить запрос
If Adodc1.Recordset.RecordCount = 0 Then
MsgBox "На следующую неделю никаких дел" + _
"не запланировано" , _
vbOKOnly, "Ежедневник"

```

End If

Часть II

КРАТКИЙ

СПРАВОЧНИК

Часть II. Краткий справочник 228

Данная часть представляет собой краткий справочник по компонентам и функциям, которые использовались для решения задач, рассмотренных в первой части книги.

Глава 7

Компоненты

В этой главе приведено краткое описание базовых компонентов Microsoft Visual Basic 6.0. Описание других компонентов можно найти в справочной системе.

По умолчанию, в Visual Basic размеры (*width*, *Height*) и координаты (*Left*, *top*) компонентов измеряются в специальных единицах — твипах (*twip*). Пятьсот шестьдесят семь твипов образуют один логический сантиметр — единицу измерения расстояния на экране монитора. Отрезок длиной в один логический сантиметр на экране будет иметь длину точно один сантиметр при распечатке данных на принтере.

Форма

Форма (объект *Form*) является основой программы. Свойства формы (табл. 7.1) определяют вид окна программы.

Таблица 7.1. Свойства объекта *Form* (формы)

Свойство

Name Имя формы. Используется для управления формой и доступа к ее компонентам или свойствам

Caption Текст заголовка

top Расстояние от верхней границы формы до верхней границы экрана

Left Расстояние от левой границы формы до левой границы экрана

Часть II. Краткий справочник 230

Таблица 7.1 (продолжение)

Свойство

Width

Height

ScaleWidth

ScaleHeight

ScaleMode

BorderStyle

Icon

BackColor

ForeColor

Описание

Ширина формы. Задается в твипах

Высота формы. Задается в твипах

Ширина рабочей области формы, то есть без учета ширины левой и правой границ. Может задаваться

как в твипах, так и в других единицах
Высота рабочей (клиентской) области формы, то есть без учета высоты заголовка и ширины нижней и верхней границ формы. Может задаваться как в твипах, так и в других единицах
Определяет единицы измерения размеров формы и объектов на ней. Значение этого свойства не влияет на единицы измерения свойств *Width* и *Height*, независимо от него их значения измеряются в твипах
Стиль (вид) границы формы (окна программы). Граница может быть обычной (*sizable*), тонкой (*FixedSingle*) или вообще отсутствовать (*None*). Если значение свойства равно *FixedSingle*, то изменить размер окна программы путем перемещения границы нельзя, но окно можно развернуть на весь экран (кнопка *Maximize*) или свернуть (кнопка *Minimize*). Если значение свойства равно *None*, то граница окна отсутствует. Изменить размер и положение такого окна нельзя. Если значение свойства равно *FixedDialog*, то окно — "модальный диалог" (нельзя изменить размер окна, нельзя свернуть окно, доступ к другим окнам программы блокируется)
Значок (иконка) в заголовке окна
Цвет фона формы. Цвет фона можно задать, выбрав его из палитры цветов или указав привязку к текущей цветовой схеме операционной системы. Во втором случае цвет определяется текущей цветовой схемой и выбранным компонентом привязки. В этом случае он меняется при изменении цветовой схемы операционной системы
Цвет, используемый при выводе текста и для контуров графических объектов
Глава 7. Компоненты 231

Таблица 7.1 (окончание)

Свойство

FillColor Цвет заполнения внутренней части графических объектов

Fillstyle Стиль (способ) заливки графических объектов:

Transparent (1) — прозрачный цвет (заливки нет);

Solid (0) — сплошная заливка; *HorizontalLine*

(2) — горизонтальная штриховка; *VerticalLine*

(3) — вертикальная штриховка. Цвет линий штриховки определяет значение свойства *FillColor*

DrawMode Способ вывода графических объектов. Например, если значение свойства равно *Blackness* (1), то

цвет всех контуров объектов и цвет заливки будет

черным (значение этого свойства не влияет на цвет

текста, выводимого при помощи метода *Print*)

DrawWidth Толщина линии для графических объектов

Drawstyle Стиль контура графических объектов, тип линии:

Solid (0) — сплошная линия; *Dash* (1) — пунктир-

ная линия; *Dot* (2) — линия из точек; *Dash-Dot*

(3) — линия "точка-тире"; *Dash-Dot-Dot* (4) — линия

"тире-точка-точка"; *Transparent* (5) — "прозрачная"

линия

Font Шрифт. Шрифт, заданный в этом свойстве, исполь-

зуется при выводе текста непосредственно на по-

верхность формы (например, при помощи команды

Print)

MaxButton Признак наличия (*True*) или отсутствия (*False*) в за-

головке формы кнопки "Развернуть окно на весь

экран"

MinButton Признак наличия (*True*) или отсутствия (*False*) в за-

головке формы кнопки "Свернуть окно — присутст-

вие или отсутствие кнопки сворачивания окна

Label

Компонент *Label* (рис. 7.1) предназначен для вывода текста на

поверхность формы. Свойства компонента (табл. 7.2) определяют вид и расположение текста на поверхности формы.
Часть II. Краткий справочник 232

Рис. 7.1. Компонент Label

Таблица 7.2. Свойства компонента Label

Свойство

Name Имя компонента. Используется в программе для доступа к компоненту и его свойствам

Caption Отображаемый текст

Left Расстояние от левой границы поля вывода до левой границы формы

Top Расстояние от верхней границы поля вывода до верхней границы формы

Height Высота поля вывода

width Ширина поля вывода

AutoSize Признак того, что размер поля определяется его содержимым

Wordwrap Признак того, что слова, которые не помещаются в текущей строке, автоматически переносятся на следующую строку

Alignment Задаёт способ выравнивания текста внутри поля.

Текст может быть выровнен по левому краю—

LeftJustify (0), по центру—*Center* (2) или по

правому краю — *RightJustify* (1)

Font Шрифт, используемый для отображения текста.

Уточняющие свойства *Name*, *Size* и *FColor* задают способ начертания, размер и цвет символов

BackColor Цвет фона области вывода текста. Цвет можно задать, выбрав его из палитры цветов или указав привязку к текущей цветовой схеме операционной системы

Глава 7. Компоненты 233

Таблица 7.2 (окончание)

Свойство

Backstyle Управляет отображением фона области вывода текста. Область вывода текста может быть закрашена (*opaque*) или быть "прозрачной" (*Transparent*).

Если значение свойства равно *Opaque*, то цвет за-
краски области определяет значение свойства

BackColor

visible Позволяет скрыть компонент (значение— *False*)

или сделать его видимым (значение — *True*)

TextBox

Компонент *TextBox* (рис. 7.2) представляет собой поле ввода-редактирования текста. Свойства компонента приведены в табл. 7.3.

Рис. 7.2. Компонент *TextBox*

Таблица 7.3. Свойства компонента *TextBox*

Свойство

Name

Text

Left

Top

Height

Имя компонента. Используется в программе для доступа к компоненту и его свойствам, в частности — для доступа к тексту, введенному в поле редактирования

Текст, находящийся в поле редактирования

Расстояние от левой границы компонента до левой границы формы

Расстояние от верхней границы компонента до верхней границы формы

Высота поля

Таблица 7.3 (окончание)

Свойство Описание

Width

Font

Locked

MultiLine

ScrollBars

Visible

Ширина поля

Шрифт, используемый для отображения содержимого поля

Используется для ограничения возможности изменить текст в поле редактирования. Если значение свойства равно *False*, то текст в поле редактирования изменить нельзя

Делает возможным многострочное отображение текста

Управляет отображением полос прокрутки:

Vertical — только полоса вертикальной прокрутки;

Horizontal — только полоса горизонтальной про-

крутки; *Both* — вертикальная и горизонтальная

полосы прокрутки; *None* — без полос прокрутки

Позволяет скрыть (*False*) компонент или сделать его видимым (*True*)

CommandButton

Компонент *CommandButton* (рис. 7.3) представляет собой командную кнопку. Свойства компонента приведены в табл. 7.4.

Таблица 7.4. Свойства компонента *CommandButton*

Свойство

Name Имя компонента. Используется в программе для доступа к компоненту и его свойствам

Caption Текст на кнопке

Глава 7. Компоненты 235

Таблица 7.4 (окончание)

Свойство

Left

Top

Height

Width

Enabled

Visible

Style

Picture

DisabledPicture

DownPicture

ToolTipText

Описание

Расстояние от левой границы кнопки до левой границы формы

Расстояние от верхней границы кнопки до верхней границы формы

Высота кнопки

Ширина кнопки

Признак доступности кнопки. Если значение свойства равно *True*, то кнопка доступна. Если значение свойства равно *False*, то кнопка не доступна, т. е. при щелчке на кнопке никаких событий не возникает

Позволяет скрыть кнопку (значение — *False*) или сделать ее видимой (значение — *True*)

Вид кнопки: "обычная" (*standard*) или "графическая" (*Graphical*). Графическая кнопка —

это кнопка, на поверхности которой есть картинка

Свойство задает картинку для "графической" кнопки. Картинка отображается на поверхности формы, если значение свойства *Style* равно *Graphical*

Задаёт картинку для недоступной "графической" кнопки. Картинка отображается, если значение свойства *Style* равно *Graphical*, а свойства *Enabled* — *False*

Задаёт картинку для нажатой "графической" кнопки. Картинка отображается в момент нажатия кнопки, если значение свойства *style* равно *Graphical*

Задаёт текст подсказки, которая появляется при позиционировании указателя мыши на кнопке

Часть II. Краткий справочник 236

CheckBox

Компонент *CheckBox* (рис. 7.4) представляет собой независимый переключатель. Свойства компонента приведены в табл. 7.5.

Рис. 7.4. Компонент *CheckBox*

Таблица 7.5. Свойства компонента *CheckBox*

Свойство

Name Имя компонента. Используется в программе для доступа к компоненту и его свойствам

Caption Текст, который находится справа от флажка

Состояние переключателя: *Checked*—флажок установлен (в квадратике есть "галочка"); *Unchecked* — флажок сброшен (нет "галочки")

Left Расстояние от левой границы флажка до левой границы формы

Top Расстояние от верхней границы флажка до верхней границы формы

Height Высота компонента

width Ширина компонента (флажок и область для пояснительного текста)

Font Шрифт, используемый для отображения пояснительного текста

Visible Позволяет скрыть компонент (значение свойства — *False*) или сделать его видимым (значение свойства — *True*)

Глава 7. Компоненты 237

OptionButton

Компонент *OptionButton* (рис. 7.5) представляет зависимую кнопку, состояние которой определяется состоянием других кнопок группы. Свойства компонента приведены в табл. 7.6.

Рис. 7.5. Компонент *OptionButton*

Таблица 7.6. Свойства компонента *OptionButton*

Свойство

Name Имя компонента. Используется для доступа к компоненту и его свойствам

Caption Текст, который находится справа от кнопки

Состояние кнопки. Если кнопка выбрана, то значение свойства — *True*, если кнопка не выбрана, значение свойства — *False*

Tag *Visual Basic* не использует это свойство. Пользователь может использовать его по своему назначению

Left Расстояние от левой границы флажка до левой границы формы

top Расстояние от верхней границы флажка до верхней границы формы

Height Высота компонента

width Ширина компонента (флажок и область для пояснительного текста)

Font Шрифт, используемый для отображения поясняющего текста

visible Позволяет скрыть компонент (значение свойства — *False*) или сделать его видимым (значение свойства — *True*)

Часть II. Краткий справочник 238

ListBox

Компонент *ListBox* (рис. 7.6) представляет собой список, в котором можно выбрать нужный элемент. Свойства компонента приведены в табл. 7.7.

Рис. 7.6. Компонент *ListBox*

Таблица 7.7. Свойства компонента *ListBox*

Свойство

Name Имя компонента. В программе используется для доступа к компоненту и его свойствам

BackColor Цвет фона области вывода списка. Цвет можно задать, выбрав его из палитры цветов или указав привязку к текущей цветовой схеме операционной системы

ForeColor Цвет, используемый для отображения элементов списка

List Элементы списка — массив строк

ListCount Количество элементов списка

Sorted Признак необходимости автоматической сортировки (значение свойства— *True*) списка после добавления очередного элемента

Listindex Номер выбранного элемента списка (нумерация элементов списка начинается с нуля)

MultiSelect Позволяет сделать возможным множественный выбор из списка. Если значение этого свойства— *None* (0), то выбрать несколько элементов из списка нельзя. При значении, равном *simple* (1), каждый элемент, на котором произвелся щелчок, становится выбранным. Отмена выбора производится при помощи повторного щелчка мыши или при помощи клавиши— пробел. Если значение свойства равно *Extended* (2), то множественный выбор осуществляется при помощи мыши и клавиши *<Shift>* или *<Ctrl>* (щелчок кнопкой мыши на элементе списка при нажатой клавише *<Shift>* помечает элемент как выбранный, повторный щелчок отменяет выбор)

Глава 7. Компоненты 239

Таблица 7.7 (окончание)

Свойство

style Стиль (вид) списка. Если значение свойства равно *CheckBox* (1), то перед каждым элементом списка отображается компонент *Checkbox*. При этом для выбора элемента из списка нужно установить соответствующий флажок. При значении свойства, равном *Standard* (0), список имеет стандартный вид

Left Расстояние от левой границы списка до левой границы формы

top Расстояние от верхней границы списка до верхней границы формы

Height Высота поля списка

width Ширина поля списка

Font Шрифт, используемый для отображения элементов списка

visible Позволяет скрыть компонент (значение свойства — *False*) или сделать его видимым (значение свойства — *True*)

ComboBox

Компонент *ComboBox* (рис. 7.7) дает возможность ввести данные в поле редактирования путем набора на клавиатуре или путем выбора из списка. Свойства компонента приведены в табл. 7.8.

Рис. 7.7. Компонент *ComboBox*

Часть II. Краткий справочник 240

Таблица 7.8. Свойства компонента *ComboBox*

Свойство

Name

Font

BackColor

ForeColor

List

ListCount

ListIndex

Style

Text

Sorted

Locked

Описание

Имя компонента. Используется для доступа к компоненту и его свойствам

Шрифт, используемый для отображения элементов списка

Цвет фона области вывода списка. Цвет можно задать, выбрав его из палитры цветов или указав привязку к текущей цветовой схеме операционной системы

Цвет, используемый для отображения элементов списка

Элементы списка — массив строк

Количество элементов списка

Номер элемента, выбранного в списке. Если ни один из элементов списка не был выбран, то значение свойства равно -1. Нумерация элементов начинается с нуля

*Стиль (вид) списка. Если значение свойства равно *DropDownCombo* (0), то данные в поле редактирования можно ввести с клавиатуры или выбрать из списка (чтобы получить доступ к списку, его надо раскрыть). Если значение свойства равно *SimpleCombo* (1), то данные можно ввести в поле редактирования с клавиатуры или выбрать из списка, причем _____ список доступен всегда. Если значение свойства равно *DropDownList* (2), то данные в поле редактирования можно ввести только путем выбора из списка*

Содержимое поля редактирования (данные, введенные пользователем с клавиатуры или выбранные из списка)

*Признак необходимости автоматической сортировки списка после добавления очередного элемента (значение свойства — *True*)*

Блокировка списка. Определяет запрет выбора элемента из списка (строка ввода также блокируется)

Глава 7. Компоненты 241

Таблица 7.8 (окончание)

Свойство

Left Расстояние от левой границы компонента до левой границы формы

top Расстояние от верхней границы компонента до верхней границы формы

Height Высота компонента

width Ширина компонента

Visible Позволяет скрыть компонент (*False*) или сделать его видимым (*True*)

Timer

*Компонент *Timer* (рис. 7.8) обеспечивает генерацию последовательности событий *Timer*. Свойства компонента приведены в табл. 7.9.*

*Рис. 7.8. Компонент *Timer**

Таблица 7.9. Свойства компонента *Timer*

Свойство Описание

Name Имя компонента. Используется для доступа к компо

ненту и его свойствам

Interval Период генерации события *Timer*. Задается в миллисекундах

Enabled Разрешение работы (запуск/остановка). Разрешает (значение свойства—*True*) или запрещает (значение свойства—*False*) генерацию события *Timer*

Часть II. Краткий справочник 242

DriveListBox

Компонент *DriveListBox* (рис. 7.9) является выпадающим списком, отображающим диски компьютера. Свойства компонента приведены в табл. 7.10.

Рис. 7.9. Компонент *DriveListBox*

Таблица 7.10. Свойства компонента *DriveListBox*

Свойство

Name Имя компонента. Используется для доступа к компоненту и его свойствам

Font Шрифт, используемый для отображения названий дисков

BackColor Цвет фона области вывода списка дисков. Цвет можно задать, выбрав его из палитры цветов или указав привязку к текущей цветовой схеме операционной системы

ForeColor Цвет шрифта, используемого для отображения имен дисков

List Элементы списка— массив строк. В нем находятся названия дисков (например, с:)

ListCount Количество дисков компьютера

ListIndex Номер элемента, выбранного в списке дисков. Если ни один из элементов списка не был выбран, то значение свойства равно -1. Нумерация дисков начинается с нуля

Left Расстояние от левой границы компонента до левой границы формы

Top Расстояние от верхней границы компонента до верхней границы формы

Глава 7. Компоненты 243

Таблица 7.10 (окончание)

Свойство

Height Высота компонента

width Ширина компонента

visible Позволяет скрыть компонент (значение свойства - *False*) или сделать его видимым (значение свойства — *True*)

DirListBox

Компонент *DirListBox* (рис. 7.10) отображает структуру указанного каталога. Свойства компонента приведены в табл. 7.11.

Рис. 7.10. Компонент *DirListBox*

Таблица 7.11. Свойства компонента *DirListBox*

Свойство

Name

Font

BackColor

ForeColor

List

Описание

Имя компонента. Используется для доступа к компоненту и его свойствам

Шрифт, используемый для отображения названий каталогов и подкаталогов

Цвет фона области вывода списка каталогов. Цвет можно задать, выбрав его из палитры цветов или указав привязку к текущей цветовой схеме операционной системы

Цвет шрифта, используемого для отображения названий каталогов и подкаталогов

Элементы указанного каталога (подкаталога) — массив строк. В элемент массива записывается полный путь доступа к подкаталогу
Часть II. Краткий справочник 244

Таблица 7.11 (окончание)

Свойство Описание

ListCount Количество подкаталогов указанного каталога
Listindex Номер элемента, выбранного в списке каталогов и подкаталогов. Нумерация подкаталогов начинается с нуля. Если в списке выбран каталог, структура которого отображается в компоненте *Dir1*, то значение свойства равно -1. Если в структуре выбрать каталог на один уровень выше (при этом список должен отображаться), то значение будет равно -2. Значение свойства продолжает уменьшаться на 1 при каждом переходе вверх по дереву каталогов на один уровень
Path Путь к каталогу
Left Расстояние от левой границы компонента до левой границы формы
top Расстояние от верхней границы компонента до верхней границы формы
Height Высота компонента
Width Ширина компонента
visible Позволяет скрыть компонент (значение свойства - *False*) или сделать его видимым (значение свойства — *True*)

FileListBox

Компонент *FileListBox* (рис. 7.11) отображает список файлов указанного каталога. Свойства компонента приведены в табл. 7.12.

Рис. 7.11. Компонент *FileListBox*

Глава 7. Компоненты 245

Таблица 7.12. Свойства компонента *FileListBox*

Свойство

Name Имя компонента. Используется для доступа к компоненту и его свойствам
Font Шрифт, используемый для отображения названий каталогов и подкаталогов
BackColor Цвет фона для области вывода списка каталогов. Цвет можно задать, выбрав его из палитры цветов или указав привязку к текущей цветовой схеме операционной системы
ForeColor Цвет шрифта, используемого для отображения списка файлов указанного каталога
List Элементы указанного каталога (список файлов)-массив строк. В элемент массива записывается название файла (имя.расширение)
Listcount Количество файлов в указанном каталоге
List Index Номер элемента, выбранного в списке файлов. Нумерация файлов начинается с нуля
Path Путь к каталогу
Archive Свойство определяет, отображаются ли в списке файлы с установленным атрибутом — "архивный"
Hidden Свойство определяет, отображаются ли в списке файлы с установленным атрибутом — "скрытый"
ReadOnly Свойство определяет, отображаются ли в списке файлы с установленным атрибутом— "только чтение"
System Свойство определяет, отображаются ли в списке файлы с установленным атрибутом — "системный"
Normal Свойство определяет, отображаются ли в списке файлы с установленным атрибутом— "архивный", "только чтение", без атрибутов или с всевозможными комбинациями этих атрибутов
Left Расстояние от левой границы компонента до левой границы формы

Таблица 7.12 (окончание)

Свойство Описание

Top Расстояние от верхней границы компонента до верхней границы формы

Height Высота компонента

width Ширина компонента

Visible Позволяет скрыть компонент (значение свойства — *False*) или сделать его видимым (значение свойства — *True*)

PictureBox

Компонент *PictureBox* (рис. 7.12) обеспечивает отображение графики. Изображение можно загрузить из файла или сформировать из графических примитивов (нарисовать) во время работы программы. Свойства компонента приведены в табл. 7.13.

Рис. 7.12. Компонент *PictureBox*

Таблица 7.13. Свойства _____ компонента *PictureBox*

Свойство Описание

Name Имя компонента. Используется для доступа к компоненту и его свойствам

Picture Картинка (объект *BitMap*), отображаемая в поле компонента. Задать картинку можно во время разработки формы или загрузить из файла во время работы программы (функция *LoadPicture*)

AutoSize Свойство разрешает (*True*) или запрещает (*False*) автоматическое изменение размера компонента (области вывода иллюстрации) в соответствии с размером картинки, загруженной в компонент

Глава 7. Компоненты 247

Таблица 7.13 (продолжение)

Свойство

BorderStyle

BackColor

Font

ForeColor

FillColor

FillStyle

DrawStyle

DrawWidth

ScaleWidth

ScaleHeight

Стиль границы компонента. Если значение свойства— *FixedSingle* (1), то граница стандартная (тонкая линия), если— *None* (0), то граница не отображается

Цвет фона компонента. Цвет можно задать, выбрав его из палитры цветов или указав привязку к текущей цветовой схеме операционной системы

Шрифт, которым метод *Print* выводит текст

Для метода *Print* задает цвет символов, для методов вычерчивания графических примитивов (объектов) - цвет линий

Задает цвет закрашки внутренних областей графических примитивов (объектов), вычерчиваемых в поле (на поверхности) компонента

Стиль закрашки графических объектов, вычерчиваемых в поле компонента соответствующими методами:

Solid (0)—сплошная заливка; *HorizontalLine* (2) —

горизонтальная штриховка; *VerticalLine* (3) —

вертикальная штриховка; *Transparent* (1) —

закраска "прозрачным" цветом. Цвет линий штриховки определяет свойство *FillColor*

Вид контура графических объектов, вычерчиваемых в поле компонента соответствующими методами:

Solid (0) — сплошная линия; *Dash* (1)— пунктир-

ная линия; Dot (2)— линия из точек; Dash-Dot (3) — линия вида "точка-тире"; Dash-Dot-Dot (4) - линия вида "тире-точка-точка"; Transparent (5) - "прозрачная" линия

Толщина линий для графических объектов

Ширина рабочей области компонента, то есть без учета ширины левой и правой границ. Единицу измерения задает свойство ScaleMode

Высота рабочей области компонента, то есть без учета ширины нижней и верхней границ компонента. Единицу измерения задает свойство ScaleMode

Часть II. Краткий справочник 248

Таблица 7.13 (окончание)

Свойство

ScaleMode

Left

Top

Height

Width

Visible

Задаёт единицу измерения размеров компонента и объектов на его поверхности. Значение этого свойства не влияет на единицы измерения свойств Width и Height (независимо от него их значения измеряются в пикселях)

Расстояние от левой границы компонента до левой границы формы

Расстояние от верхней границы компонента до верхней границы формы

Высота компонента

Ширина компонента

Позволяет скрыть компонент (значение свойства — False) или сделать его видимым (значение свойства — True)

Image

Компонент *image* (рис. 7.13) обеспечивает отображение иллюстраций. Отличается от компонента *pictureBox* тем, что поверхность компонента недоступна методам вычерчивания графических примитивов (на поверхности компонента рисовать нельзя).

Свойства компонента *image* приведены в табл. 7.14.

Рис. 7.13. Компонент *Image*

Таблица 7.14. Свойства компонента *image*

Свойство Описание

Name Имя компонента. Используется для доступа к компоненту и его свойствам

Глава 7. Компоненты 249

Таблица 7.14 (окончание)

Свойство

Picture Картинка (объект *Bitmap*), отображаемая в поле компонента. Задать картинку можно во время разработки формы или загрузить из файла во время работы программы (функция *LoadPicture*)

BorderStyle Стилль границы компонента: *FixedSingle* (1) — стандартная граница (тонкая линия); *None* (0) — границы нет

Stretch Признак масштабирования (сжатия или растяжения) иллюстрации в соответствии с реальным размером компонента. Значение *True* — иллюстрация масштабируется в соответствии с размером компонента (если размер компонента не пропорционален размеру иллюстрации, то иллюстрация будет искажена). Значение *False* — масштабирование не выполняется

Left Расстояние от левой границы компонента до левой границы формы

Top Расстояние от верхней границы компонента до верхней границы формы
Height Высота компонента
width Ширина компонента
visible Позволяет скрыть компонент (значение свойства — *False*) или сделать его видимым (значение свойства — *True*)

Shape

Компонент *shape* (рис. 7.14) — графический объект (прямоугольник, овал или круг, прямоугольник со скругленными углами), который можно поместить на поверхность формы. Компонент может использоваться только в качестве декоративного элемента, т. к. он не может воспринимать события. Свойства компонента приведены в табл. 7.15.

Часть II. Краткий справочник 250

Рис. 7.14. Компонент *Shape*

Таблица 7.15. Свойства компонента *shape*

Свойство Описание

Name Имя компонента. Используется для доступа к компоненту и его свойствам

Shape Вид геометрической фигуры: *Rectangle* (0) — прямоугольник; *Square* (1) — квадрат; *Oval* (2) — овал;

Circle (3) — круг; *RoundedRectangle* (4) — прямоугольник со скругленными углами; *RoundedSquare* (5) — квадрат со скругленными углами

BackColor Цвет фона компонента. Цвет можно задать, выбрав его из палитры цветов или указав привязку к текущей цветовой схеме операционной системы

BackStyle Стиль фона компонента: *Transparent* (0) — прозрачный, *Opaque* (1) — непрозрачный

BorderColor Цвет границы объекта (контура геометрической фигуры)

BorderStyle Вид контура объекта (геометрической фигуры): *Solid* (0) — сплошная линия; *Dash* (1) — пунктирная линия;

Dot (2) — линия из точек; *Dash-Dot* (3) — линия "тире-точка-тире"; *Dash-Dot-Dot* (4) — линия "тире-точка-точка", *Transparent* (5) — "прозрачная" линия

DrawWidth Толщина линии контура объекта (геометрической фигуры)

FillColor Цвет закрашки внутренней области объекта (геометрической фигуры)

Fillstyle Стиль закрашки внутренней области объекта (геометрической фигуры): *Transparent* (1) — "прозрачный" цвет (нет закрашки), *Solid* (0) — сплошная заливка;

HorizontalLine (2) — горизонтальная штриховка;

VerticalLine (3) — вертикальная штриховка. Цвет линий штриховки определяется значением свойства

FillColor

Глава 7. Компоненты 251

Таблица 7.15 (окончание)

Свойство

Left Расстояние от левой границы компонента до левой границы формы

Top Расстояние от верхней границы компонента до верхней границы формы

Height Высота компонента

width Ширина компонента

Visible Позволяет скрыть компонент (значение — *False*) или сделать его видимым (значение — *True*)

Line

Компонент *Line* (рис. 7.15) — графический объект "линия"

Компонент может использоваться только в качестве декоративного элемента, т. к. он не может воспринимать события. Свойства компонента приведены в табл. 7.16.

Рис. 7.15. Компонент *Line*

Таблица 7.16. Свойства компонента *line*

Свойство

Name Имя компонента. Используется для доступа к компоненту и его свойствам

Border-Color Цвет линии

Border Style Вид линии: *Solid* (0)— сплошная; *Dash* (1)— пунктирная _____; *Dot* (2)— линия из точек; *Dash-Dot* (3)— линия "точка-тире"; *Dash-Dot-Dot* (4)— линия "тире-точка-точка"; *Transparent* (5)— прозрачная линия

Часть II. Краткий справочник 252

Таблица 7.16 (окончание)

Свойство Описание

DrawWidth Толщина линии

X1 Горизонтальная координата точки начала линии

Y1 Вертикальная координата точки начала линии

X2 Горизонтальная координата точки конца линии

Y2 Вертикальная координата точки конца линии

visible Позволяет скрыть компонент (значение— *False*)

или сделать его видимым (значение — *True*)

UpDown

Компонент *UpDown* (рис. 7.16) представляет собой две кнопки, используя которые, можно изменить (увеличить или уменьшить) значение внутренней переменной-счетчика. Компонент *UpDown* можно связать с другим компонентом и использовать в качестве индикатора значения переменной-счетчика. Свойства компонента *UpDown* приведены в табл. 7.17.

Рис. 7.16. Компонент *UpDown*

Таблица 7.17. Свойства компонента *UpDown*

Свойство Описание

Name Имя компонента. Используется для доступа к компоненту и его свойствам

Value Счетчик. Значение счетчика изменяется в результате щелчка на кнопке *Up* (увеличение) или *Down* (уменьшение)

Глава 7. Компоненты 253

Таблица 7.17 (продолжение)

Свойство Описание

Increment

Величина изменения значения переменной-счетчика

Верхняя граница изменения значений переменной-счетчика *Value*

Нижняя граница изменения значений переменной-счетчика *Value*

Компонент, который используется в качестве индикатора значения переменной-счетчика (например, *Label* ИЛИ *TextBox*)

Свойство компонента, указанного в *BuddyControl*, обеспечивающее индикацию значения переменной-счетчика (*caption*, если индикатор—компонент *Label*)

Автоматическое определение свойства компонента-индикатора, используемого для индикации состояния переменной-счетчика. Если в качестве индикатора используется компонент *Label*, то автоматически в качестве значения свойства *BuddyProperty* устанавливается *Caption*

Синхронизация (*True*) изменением значения

Value и значения свойства компонента-индикатора

Ориентация кнопок (стрелок на кнопках) компонента: *OrientationVertical* (0) — по вертикали

(вверх, вниз); *OrientationHorizontal* (1)—по горизонтали (влево, вправо)

Если значение свойства равно *False*, то при достижении максимального значения переменной *Value*,

ее значение не изменяется при последующих нажатиях кнопки Up. Аналогично для кнопки Down. Если значение свойства равно True, при аналогичных действиях, максимальное значение переменной Value изменяется на минимальное и наоборот
Доступность(значение свойства—True) или недоступность компонента (значение свойства —False)

Max

Min

BuddyControl

BuddyProperty

AutoBuddy

SyncBuddy

Orientation

Wrap

Enabled

Часть II. Краткий справочник 254

Таблица 7.17 (окончание)

Свойство Описание

Left Расстояние от левой границы компонента до левой границы формы

Top Расстояние от верхней границы компонента до верхней границы формы

Height Высота компонента

width Ширина компонента

visible Позволяет скрыть компонент (значение свойства— False) или сделать его видимым (значение свойства — True)

CommonDialog

Компонент *CommonDialog* (рис. 7.17) представляет собой стандартное диалоговое окно Windows (**Открыть**, **Сохранить**, **Цвет**, **Шрифт**, **Печать** или **Справка**). Свойства компонента приведены в табл. 7.18. Тип окна определяет метод, обеспечивающий отображение диалога (табл. 7.19).

Рис. 7.17. Компонент *CommonDialog*

Таблица 7.18. Свойства компонента *CommonDialog*

Свойство

Name Имя компонента. Используется для доступа к компоненту и его свойствам

DialogTitle Заголовок окна

Глава 7. Компоненты 255

Таблица 7.18 (окончание)

Свойство

FileName Полное имя файла, выбранного в диалоге **Открыть**.

Если файл не был выбран, т. е. диалог завершен нажатием кнопки **Cancel**, значение свойства — пустая строка

Filter Фильтр. Задаёт файлы, отображаемые в окнах (диалогах) **Открыть** и **Сохранить**. Если фильтр не задан, то отображаются все файлы

Filter index Номер выбранного фильтра. Фильтры нумеруются с 1

Flags Флаги спецификации для диалоговых окон

HelpFile Путь к help-файлу, который нужно отобразить

Таблица 7.19. Методы отображения объекта *CommonDialog*

Метод

ShowOpen

ShowSave

ShowColor

ShowFont

ShowPrinter

ShowHelp

Диалог

Открыть

Сохранить

Цвет

Выбор шрифта

Печать

Справка

MMControl

Компонент *MMControl* (рис. 7.18, 7.19) обеспечивает воспроизведение звуковых и видеофайлов. Свойства компонента *MMControl* приведены в табл. 7.20.

Рис. 7.18. Значок компонента *MMControl*

Часть II. Краткий справочник 256

Рис. 7.19. Кнопки компонента *MMControl*

Таблица 7.20. Свойства компонента *MMControl*

Свойство Описание

Name Имя компонента. Используется в программе для доступа к компоненту и его свойствам

DeviceType Тип устройства: *CDAudio* — проигрыватель звуковых CD; *WaveAudio* — проигрыватель WAV-файлов;

AVIVideo — проигрыватель AVI-файлов и др.

Command Команда

TimeFormat Формат измерения времени

FileName Имя файла, который нужно воспроизвести

PlayEnabled Делает кнопку *Play* данного компонента доступной

(значение — *True*) или недоступной (значение — *False*). Аналогичное свойство есть для всех остальных кнопок компонента

PlayVisible Позволяет скрыть кнопку *Play* (значение — *False*)

компонента или сделать ее видимой (значение — *True*). Аналогичное свойство есть для всех остальных кнопок компонента

Left Расстояние от левой границы компонента до левой границы формы

Top Расстояние от верхней границы компонента до верхней границы формы

Height Высота компонента

Width Ширина компонента

Visible Позволяет скрыть компонент (значение — *False*)

или сделать его видимым (значение — *True*)

Глава 8

Графика

В этой главе приведено описание методов, обеспечивающих вывод графики на поверхность формы или компонента

PictureBox.

Инструкции вызова метода (обращения к свойству) в общем виде выглядят следующим образом:

Объект.Метод

Объект.Свойство

Здесь следует обратить внимание то, что объект, в инструкции вызова метода или обращения к свойству, можно не указывать.

Если объект не указан, то используется значение "по умолчанию", а именно — форма.

В приводимом ниже описании методов их необязательные параметры заключены в квадратные скобки.

Print

[объект.]Print String

Метод *Print* выводит на поверхность объекта строку *string* от текущей точки (узнать координаты текущей точки можно, обратившись к свойствам *currentx* и *currenty*). Шрифт определяется свойством *Font* графической поверхности (например, формы), на которую выводится текст, цвет символов - свойством *ForeColor* той же поверхности.

Пример:

```
Font.Name = "Arial"
```

```
Font.Size = 12
```

Часть II. Краткий справочник 258

```
ForeColor = RGB(0, 0, 255)
```

```
ScaleMode = 3 ' координаты отсчитывать в пикселах
```

```
CurrentX = 10
```

```
CurrentY = 20
```

```
Print "Microsoft Visual Basic"
```

Line

[Объект.]Line [Step] (x1, y1) [Step]-(x2, y2), [Color], [B][F]

Метод *Line* рисует линию или прямоугольник (если указан параметр *v*).

Параметры *x1*, *y1* и *x2*, *y2* задают координаты точки начала и конца линии или находящихся на одной диагонали углов прямоугольника (если указан параметр *v*).

Цвет линии (контура прямоугольника) определяет свойство *ForeColor* графической поверхности (если не указан параметр *Color*).

Толщину и вид определяют соответственно свойства *Drawwidth* и *Drawstyle* объекта, на поверхности которого метод рисует.

Параметр *step* показывает, что реальные координаты отсчитываются от указателя текущей точки.

Параметр *color* задает цвет линии или, если задан параметр *v*, границы прямоугольника (по умолчанию цвет контура определяет свойство графической поверхности).

Параметр *v* определяет, что надо нарисовать прямоугольник.

Параметр *F* определяет, что прямоугольник должен быть закрашен тем же цветом, что и граница.

Пример:

```
ScaleMode = 3 ' единица измерения координат - пиксели
```

```
Line (10, 20)-(100, 20) ' линия
```

```
Line (10, 30)-(100, 30), RGB(255, 0, 0) ' линия красного цвета
```

```
Line (10, 40)-(100, 50), , B ' прямоугольник
```

```
Line (10, 60)-(100, 70), RGB(0, 127, 0), B ' прямоугольник
```

```
Line (10, 80)-(100, 90), RGB(0, 127, 0), BF
```

Глава 8. Графика 259

Circle

[объект.]Circle [Step] (x,y), Radius, [Color, Start, End, Aspect]

Метод *circle* позволяет нарисовать окружность, эллипс (если задан параметр *Aspect*) или дугу. Цвет контура определяет свойство *ForeColor* графической поверхности, на которой метод рисует (если не задан параметр *color*).

Вид контура (толщина и стиль линии) определяют соответственно свойства *Drawwidth* и *Drawstyle* графической поверхности, на которой метод рисует.

Параметр *step* показывает, что реальные координаты отсчитываются от указателя текущей точки.

Параметр *Radius* задает радиус окружности или, если задан параметр *Aspect*, больший радиус эллипса.

Параметр *color* задает цвет контура (по умолчанию цвет контура определяет свойство *ForeColor* графической поверхности).

При вычерчивании дуги параметр *start* задает угловую координату точки начала дуги, а параметр *End* — угловую координату точки конца дуги. Дуга вычерчивается от точки *start* против часовой стрелки. Угловые координаты измеряются в радианах.

Для пересчета величины угла из градусов в радианы можно воспользоваться формулой:

$$r = 2\pi / (a / 360),$$

где: r — величина угла в радианах, a — величина угла в градусах, π — число "Пи" (3.1415926).

Параметр *Aspect* задает вид эллипса. Если значение параметра *Aspect* меньше 1, то эллипс получается путем сжатия окружности по вертикали (если значение параметра 0, то эллипс вырождается в вертикальную линию). Если значение параметра *Aspect* больше 1, то эллипс получается путем сжатия окружности по горизонтали.

Пример:

$\pi = 3.1415926$ ' число "Пи"

ScaleMode = 3 ' единица измерения координат — пиксели

Circle (100, 50), 30 ' окружность

Часть II. Краткий справочник 260

Circle (50, 50), 30, RGB(255, 0, 0)

Circle (150, 50), 30, , 0, π ' дуга

Circle (150, 50), 30, , 0, $\pi / 2$ ' дуга

Circle (150, 50), 30, RGB(255, 0, 0), $\pi / 2$, 0 ' дуга

Circle (100, 120), 30, , , , 0.5 ' эллипс

Circle (100, 120), 30, , , , 2 ' эллипс

RGB

RGB(r,g,b)

Функция RGB возвращает код цвета. Параметры r , g , b задают долю красной (r — read), зеленой (g — green) и синей (b — blue) составляющей. Диапазон изменения параметров r , g , b — от 0 до 255.

В табл. 8.1. приведены значения параметров r , g , b и указан цвет, соответствующий сочетанию значений параметров.

Таблица 8.1. Кодирование цвета

Цвет RGB-кодировка

Оливково-зеленый, темный 79 79 47

Оранжево-красный 255 36 0

Оранжевый 255 127 0

Аквамарин 112 219 147

Васильковый 66 66 111

Весенне-зеленый, средний 127 255 0

Светло-лиловый, средний 147 112 219

Сине-фиолетовый 159 95 159

Средне-синий 50 50 205

Средний лесной зеленый 107 142 35

Багряный 140 23 23

Глава 8. Графика 261

Таблица 8.1 (продолжение)

Цвет RGB-кодировка

Бирюзовый 173 234 234

Бирюзовый, средний 112 219 219

Бледно-зеленый 143 188 143

Бронзовый 140 120 83

Бронзовый 2 166 125 61

Весенне-зеленый 0 255 127

Голубой 35 107 142

Голубой кадет 95 159 159

Древесный, темный 133 94 66

Древесный, светлый 233 194 166

Древесный, средний 166 128 100

Дымчато-серый 84 84 84

Желтовато-зеленый 153 204 50

Зеленовато-желтый 147 219 112

Зеленовато-медный 82 127 118

Зеленовато-медный, темный 74 118 110

Зеленый морской, средний 66 111 66
Золотарниковый 219 219 112
Золотарниковый, средний 234 234 174
Золотой 205 127 50
Индийский красный 78 47 47
Кварц 217 217 243
Кирпич 142 35 35
Коралловый 255 127 0
Коричневато-зеленый 50 205 50
Коричневый 166 42 42
Латунный 181 166 66
Часть II. Краткий справочник 262

Лесной зеленый
Лиловый
Мандариновый (оранжевый)
Медный
Морской волны
Насыщенный синий
Небесно-голубой
Неоновый розовый
Новый полуночно-синий
Осеннее небо
Острый розовый
Охотничий зеленый
Охра
Очень светло-серый
Очень темно-коричневый
Перванш
Перванш, средний
Перванш, темный
Полевой шпат
Полуночно-голубой
Прохладный медный
Пшеничный
Розовый
Розовый, с серовато-коричневым
оттенком
Рыжевато-коричневый
Рыжевато-коричневый, новый
35 142 35
153 50 205
228 120 51
184 115 51
35 142 104
89 89 171
50 153 204
255 110 199
0 0 156
56 176 222
255 28 174
33 94 33
142 107 35
205 205 205
92 64 51
0 127 255
127 0 255
107 35 142
209 146 117
47 47 79
217 135 25
216 216 191
188 143 143
133 99 99
219 147 112
235 199 158

Таблица 8.1 (продолжение)
Цвет RGB-кодировка
Глава 8. Графика 263

Рыжевато-коричневый, темный
 Светло-голубой
 Светло-лиловый
 Светло-серый
 Светло-синий
 Серебряный
 Серый
 Сливовый
 Сомон
 Средний аквамарин
 Старое золото
 Темно-бирюзовый
 Темно-бордовый
 Темно-зеленый
 Темно-коричневый
 Темно-пурпурный
 Темно-серый, Аспидный
 Темно-синий
 Фиолетово-красный
 Фиолетово-красный, средний
 Фиолетовый
 Хаки
 Чертополох
 Шоколадный Баркера
 Шоколадный, полусладкий
 Яркий золотой
 151 105 79
 143 143 189
 219 112 219
 168 168 168
 192 217 217
 230 232 250
 192 192 192
 234 173 234
 111 66 66
 50 205 153
 207 181 59
 112 147 219
 142 35 107
 47 79 47
 92 64 51
 135 31 120
 47 79 79
 35 35 142
 204 50 153
 219 112 147
 79 47 79
 159 159 95
 216 191 216
 80 51 23
 107 66 38
 217 217 25
Таблица 8.1 (окончание)
Цвет RGB-кодировка

Глава 9

Функции

В этой главе дается краткое описание наиболее часто используемых функций. Подробную характеристику этих и других функций можно найти в справочной системе.

В описании необязательные параметры заключены в квадратные скобки.

Ввод и вывод

Ввод исходных данных может быть реализован при помощи функции *inputBox*, а вывод результата — при помощи *MsgBox*.

InputBox

InputBox(*Prompt*[, *Title*][, *Default*][, *X*][, *Y*][, *HlpFile*, *Cnt*])

Функция *inputBox* выводит на экран диалоговое окно, в поле редактирования которого пользователь может ввести исходные данные — строку символов. Значением функции является введенная строка.

Параметр *Prompt* задает строку-подсказку — сообщение, которое отображается в диалоговом окне.

Параметр *Title* задает текст заголовка окна. Если этот параметр не указан, то в заголовке будет имя приложения — программы, которая запрашивает данные.

Параметр *Default* (выражение строкового типа) задает текст, отображаемый в поле ввода (пользователь может ввести исходные данные путем редактирования этого текста). Если параметр не указан, то, при появлении окна на экране, поле ввода будет пустым.

Параметры *x* и *y* задают положение окна ввода. Параметры задаются в тейпах. Если параметры не указаны, окно будет выведено в центр экрана.

Параметры *x* и *y* задают положение окна ввода. Параметры задаются в тейпах. Если параметры не указаны, окно будет выведено в центр экрана.

MsgBox

MsgBox(*Prompt*[, *MsgType* [, *Title*][, *HFile*, *Cnt*])

Функция *MsgBox* выводит на экран окно с сообщением. Значение функции — код кнопки, щелчком на которой пользователь закрыл окно.

Параметр *Prompt* (выражение строкового типа) задает текст сообщения.

Параметр *MsgType* (целого типа) задает тип сообщения и командные кнопки, которые отображаются в окне сообщения. Необходимое значение параметра *MsgType* можно вычислить по формуле:

$Msg + Btn$

где: *Msg* — тип сообщения, *Btn* — код кнопки (кнопок), которую надо отобразить в окне сообщения. Тип сообщения:

vbInformation (64) — информационное, *vbCritical* (16) —

сообщение об ошибке. Код кнопки (кнопок): *vbOKOnly* (0) —

отображается кнопка *Ok*; *vbOKCancel* (1) — отображаются кнопки *Ok* и *Cancel*; *vbYesNo* (4) — отображаются кнопки *Yes* и *No*.

Если параметр не указан, в окне сообщения отображается только кнопка *Ok*.

Параметр *Title* задает заголовок окна сообщения. Если этот параметр не указан, то в заголовке отображается имя приложения — программы, которая вывела сообщение.

Параметр *HlpFile* задает файл справочной информации, а параметр *cnt* — номер раздела справочной информации. Чтобы получить доступ к справочной информации, пользователь должен нажать клавишу <F1>.

Часть II. Краткий справочник 266

Математические функции

В табл. 9.1 приведены наиболее часто используемые математические функции.

Таблица 9.1. Математические функции

Функция Значение

Abs(N) Абсолютное значение (модуль) *N*

Sqr(N) Квадратный корень *N*

Exp(N) Экспонента *N*

Sgn(N) Знак числа *N*. Если значение выражения *N* меньше нуля, то значение функции равно 1, если больше, то 0

Rnd(N) Случайное число в диапазоне от 0 до (*N*-1). Если параметр *N* не указан, то значение функции — случайное число в диапазоне [0, 1[. Перед первым обращением к функции *Rnd* необходимо инициализировать генератор случайных чисел — вызвать функцию *Randomize*

int(N) Целая часть числа. Значение получается путем "отбрасывания" дробной части. Если *N* отрицательное, то значение функции — ближайшее отрицательное целое число, меньшее либо равное *N*. Например:

Int(5.85) = 5, Int(-5.85) = -6

Fix(N) Функция отбрасывает дробную часть числа и возвращает целое значение. Для отрицательных чисел функция возвращает ближайшее отрицательное целое число, большее либо равное *N*. Например:

Fix(5.85) = 5, Fix(-5.85) = -5

isNumeric(S) Функция проверяет, является ли строка *S* изображением числа. Если строка *s* (или подстрока от первого символа) является изображением числа, то значение функции — *True*. Если строка не является изображением числа, то значение функции — *False*. Например:

IsNumeric("5.85") — возвращает *True*,

IsNumeric("5,85") — возвращает *True*,

IsNumeric("hello") — возвращает *False*

Глава 9. Функции 267

Таблица 9.1 (окончание)

Функция Значение

Log(N) Функция вычисляет натуральный логарифм (логарифм по основанию *e*). Десятичный логарифм можно вычислить по формуле:

$Log(N) / Log(10)$

Sin(a) Синус угла *a*

Cos(a) Косинус угла *a*

Tan(a) Тангенс угла *a*

Atn(x). Арктангенс угла *a*

Величина угла тригонометрических функций (*Sin*, *Cos*, *Tan*, *Atan*) должна быть указана в радианах. Для преобразования величины угла из градусов в радианы можно воспользоваться формулой

$(g * 3.14159265358979) / 180$

где: *g* — величина угла в градусах; 3.14159265358979 — число *π*.

Преобразование данных

В табл. 9.2 приведены наиболее часто используемые функции преобразования.

Таблица 9.2. Функции преобразования данных

Функция Описание

CBool(Expression) Преобразует выражение в тип *Boolean*.

Пример:

CBool(5 > 4) — возвращает значение

True,

CBool(5 = 4) — возвращает *False*

CDBl(Expression) Преобразует выражение в тип *Double*

CInt(Expression) Преобразует выражение в тип *Integer*

CLng(Expression) Преобразует выражение в тип *Long*

Часть II. Краткий справочник 268

Таблица 9.2 (окончание)

Функции Описание

CSng (Expression) Преобразует выражение в тип *Single*
CVar (Expression) Преобразует выражение в тип *Variant*
CDate (Expression) Преобразует выражение в тип *Date*
cstr (Expression) Преобразует числовое выражение в строку

Работа со строками

В табл. 9.3 приведены наиболее часто используемые функции, обеспечивающие операции со строками.

Таблица 9.3. Функции работы со строками

Функция

Chr(Code) Функция *Chr* возвращает ANSI-символ, код которого равен *Code*. Значение *Code* должно лежать в промежутке от 0 до 255

Asc(ch) Функция возвращает код символа *Ch*. Если берется строка символов, то функция возвращает код первого символа строки

InStr([start,] String1, String2 [, Compare]) Функция *InStr* выполняет поиск подстроки *String1*, *String2* (в частном случае – символа) в строке. Параметр *Compare* – опциональный, определяет способ сравнения символов. Значение функции – позиция подстроки (символа) в строке. Если искомой подстроки в строке нет, то значение функции равно 0.

Необязательный параметр *start* задает позицию символа в строке, от которого надо выполнить поиск. Если параметр не указан, то поиск начинается от первого символа.

Параметр *String1* – строка, в которой ведется поиск.

Параметр *String2* – подстрока (символ), который надо найти в строке *String1*.

Глава 9. Функции 269

Таблица 9.3 (продолжение)

Функция

Параметр *Compare* задает режим сравнения строк: *TextCompare* (1) – текстовое сравнение, *BinaryCompare* (0) – побитовое сравнение. В режиме сравнения строк прописные и строчные символы считаются одинаковыми, а в режиме побитового сравнения – разными. Например:

InStr(1, "Hh", "h", vbBinaryCompare) – равно 2,

InStr(1, "Hh", "h", vbTextCompare) – равно 1

InStrRev(String1, String2 [, Start]) Функция *InStrRev* выполняет поиск подстроки *String2* (символа) в строке *String1*. Параметр *Start* – опциональный, определяет положение найденной подстроки отсчитывается от первого левого символа. Если искомой подстроки в строке нет, то значе-

ние функции равно 0.
 Необязательный параметр *start* задает позицию символа в строке, от которого надо выполнить поиск. Если параметр не указан, то поиск начинается от последнего символа.
 Параметр *String1*—строка, в которой ведется поиск.
 Параметр *String2* — подстрока (символ), который надо найти в строке *String1*.
 Параметр *Compare* задает режим сравнения строк: *TextCompare* (1)— текстовое сравнение, *BinaryCompare* (0)— побитовое сравнение. В режиме сравнения строк прописные и строчные символы считаются одинаковыми, а в режиме побитового сравнения разными. Например:
InStrRev (1, "Hht", "ht", vbBinaryCompare) — равно 0,
InStr (1, "Hht", "ht", vbTextCompare) — равно 2
 Часть II. Краткий справочник 270

Таблица 9.3 (продолжение)

Функция

Len(S) Функция *Len* возвращает длину строки S (количество символов в строке)
LCase(S) Функция *LCase* преобразует прописные символы строки s в строчные. Цифры и строчные буквы остаются без изменения
UCase(S) Функция *UCase* преобразует строчные символы строки s в прописные. Цифры и прописные буквы остаются без изменения
Left (S, L) Функция *Left* возвращает первые (отсчет от начала строки, то есть слева) L символов строки S. Если значение L больше, чем количество символов в строке s, то значение функции — строка s
Right (S, L) Функция *Right* возвращает последние (отсчет от конца строки, то есть справа) L символов строки s. Если значение L больше, чем количество символов в строке s, то значение функции — строка s
LTrim(S) Функция *LTrim* удаляет пробелы в начале строки
RTrim(S) Функция *RTrim* удаляет пробелы в конце строки
Trim(S) Функция *Trim* удаляет пробелы в начале и конце строки
Mid (Str, Start) Значение функции *Mid* — подстрока, выделенная из строки str. Параметр *Start* задает позицию подстроки, а *Len* — ее длину (число символов). Например:
Mid ("Ms Visual Basic",4,6)—равно
 Visual
Space (N) Значение функции —строка из N пробелов
String (N, Ch) Значение функции— строка, состоящая из N символов ch
strReverse(S) Функция "переворачивает" строку s. Например:
StrReverse("Hello") —равно olleH
 Глава 9. Функции 271

Таблица 9.3 (окончание)

Функция

Val (S) Функция *Val* возвращает числовое значение, изображением которого является строка s.
 Если в строке есть недопустимые символы,

то будет обработана только часть строки, которую можно преобразовать в число. Пробелы игнорируются. При обработке строки, являющейся изображением дробного числа, правильным символом-разделителем является точка. Если строку преобразовать в число нельзя, то значение функции `val` равно нулю. Примеры: `Val(123 45) = 12345;`
`Val(123,45) = 123;`
`Val(123.45) = 123.45;`
`Val("Text") = 0`

`CDbl(S)` Функция `CDbl` возвращает числовое значение, изображением которого является строка `S`.

Если в строке есть недопустимые символы, то будет обработана только часть строки, которую можно преобразовать в число. Пробелы игнорируются. При обработке строки, являющейся изображением дробного числа, правильным символом-разделителем является символ, заданный в настройке операционной системы (для России это запятая). Если строку преобразовать в число нельзя, то возникает ошибка (исключение).

Примеры:

`CDbl(123 45) = 12345;`
`CDbl(123,45) = 123,45;`
`CDbl(123.45) = 123`

Работа с датами и временем

В табл. 9.4 приведены наиболее часто используемые функции для работы с датами и временем.

Часть II. Краткий справочник 272

Таблица 9.4. Функции работы с датами и временем

Функция Значение

`Date` Текущая дата по системному календарю компьютера

`Time` Текущее _____ время по системным часам компьютера

`Now` Текущее время и дата по системным часам и календарю компьютера

`Year (Date)` Год для заданной даты

`Month (Date)` Номер месяца для заданной даты

`MonthName (Month [, Функция возвращает полное или сокращенное (3 символа) название месяца по его номеру (аргумент Month должен быть`

числом в диапазоне от 1 до 12; если значение аргумента `Abbreviate` равно

`True`, возвращается сокращенное название месяца, если `False`— возвращается полное название)

`Day (Date)` День месяца (число от 1 до 31) для заданной даты

`Weekday (Date, Функция возвращает номер дня недели [FirstDayOfWeek])` ли по `date Date`. Аргумент

`FirstDayOfWeek` определяет первый день недели: `vbUseSystem (0)`— используются значения системных установок, `Sunday (1)`— воскресенье,

Monday (2) — понедельник и т. д.
 WeekdayName (Weekday, Функция возвращает полное или сокращенное название дня недели (2 символа) по его номеру. Аргумент Weekday должен быть числом в диапазоне от 1 до 7; если значение аргумента Abbreviate равно True, возвращается сокращенное название месяца, если False— возвращается полное название. Аргумент FirstDayOfWeek определяет первый день недели: vbUseSystem (0)— используются значения системных установок, Sunday (1)—воскресенье, Monday (2) — понедельник и т. д.
 Глава 9. Функции 273

Таблица 9.4 (окончание)

Функция

Hour (Time)

Minute (Time)

Second (Time)

Timer

Значение

Количество часов из выражения Time
 Количество минут из выражения Time
 Количество секунд из выражения Time
 Количество секунд с точностью до одной сотой, прошедших от полуночи до текущего момента

Работа с файлами

В табл. 9.5 приведены наиболее часто используемые функции, обеспечивающие операции с файлами.

Таблица 9.5. Функции работы с файлами

Функция

Open PathName

For Mode

[Access Am]

[Lock]

As

#FileNumber

[Len = reclen]

Функция Open открывает файл для выполнения операций чтения/записи. Параметр PathName задает имя файла, к которому надо получить доступ. Параметр Mode задает режим доступа к файлу: input — ввод данных (чтение), Output— вывод данных (запись), Binary— чтение/запись файла прямого доступа, Random— чтение/запись текстового двоичного файла. Параметр Am задает операции, разрешенные для открытого файла: Read (чтение), Write (запись), Read Write (чтение/запись). Параметр FileNumber— номер файла (число в диапазоне от 1 до 511) используется в файловых операциях в качестве идентификатора файла.

Часть II. Краткий справочник 274

Таблица 9.5 (продолжение)

Функция

Seek #FileNumber,

Position

Seek(#FileNumber

FreeFile[(Range)

Get

#FileNumber

,

```
[RecNumber]
,
VarName
Put
#FileNumber
,
[RecNumber]
,
VarName
```

Параметр *recLen* задает длину записи файла (размер буфера), если файл открывается в режиме прямого доступа (*Binary*)

Функция *Seek* устанавливает указатель текущей позиции для выполнения операции чтения/записи файла, открытого в режиме прямого доступа (*Binary*). Параметр *FileNumber* — идентификатор файла. Параметр *Position* задает позицию (номер байта или записи), которую надо прочитать или перезаписать

Функция *Seek* возвращает текущую позицию указателя чтения/записи для файла

Функция *FreeFile* возвращает число, которое можно использовать в качестве идентификатора файла (параметра *FileNumber* в функции *Open*)

Функция *Get* считывает данные из файла: *FileNumber* — номер (идентификатор) файла, *RecNumber* — позиция (номер байта или номер записи, если файл открыт в режиме *Binary*), в которую надо установить указатель чтения перед выполнением операции, *VarName* — переменная, в которую надо поместить данные

Функция *Put* записывает данные в файл: *FileNumber* — номер (идентификатор) файла, *RecNumber* — позиция (номер байта или номер записи, если файл открыт в режиме *Binary*), в которую надо установить указатель чтения перед выполнением операции, *VarName* — переменная, в которой находятся данные

Глава 9. Функции 275

Таблица 9.5 (продолжение)

Функция

```
Line Input
#FileNumber,
VarName
Input #FileNumber,
VarList
Input(Number,
#FileNumber)
Print #FileNumber,
[OutputList]
Write #FileNumber,
[OutputList]
FileLen(PathName)
LOF(FileNumber)
EOF(FileNumber)
```

Функция *Line* с параметром *Input* считывает строку из файла *FileNumber* и записывает ее в переменную *VarName*. Чтение происходит до тех пор, пока не будет обна-

ружен символ "новая строка" (код 13)
 Функция `input` считывает данные из файла. `FileNumber`— номер файла, `VarList`— список переменных, значение которых надо прочитать из файла. Например:
`Input #1, a,b,c`
 Функция `input` считывает символьные или байтовые данные из файла, открытого в режиме `Input` или `Binary`. `Number`— число считываемых символов или байтов, `FileNumber` — номер файла. Например:
`IDChar = Input(1,#1)`
 Функция записывает в заданный параметром `FileNumber` текст. Параметр `OutputList` (список вывода) — список выражений символьного типа. Например:
`Print #1, "a="+Str(a),"b="+Str(b)`
 Функция записывает данные в файл. `OutputList`— записываемые данные (список переменных). Символьные _____ данные в файле будут заключены в кавычки. Например:
`Write #1, a, Ъ`
 возвращает длину
 Функция `FileLen`
 файла (в байтах)
 Функция `LOF` возвращает длину файла (в байтах)
 Функция `EOF` проверяет положение указателя чтения/записи. Значение функции равно `True`, если достигнут конец файла (прочитан последний элемент данных)
 Часть II. Краткий справочник 276

Таблица 9.5 (продолжение)

Функция

`Dir[, (Path
 [, Attributes
])]`

`CurDir`

`ChDir Path`

Функция `Dir` возвращает имя файла или папки, соответствующее критерию, заданному параметрами `Path` и `Attributes`.

Если файлов (каталогов), удовлетворяющих указанным параметрам нет, то значение функции — "пустая" строка (" "). Если в качестве параметра `Path` задан шаблон имени файла (например, `c:\temp*.bmp`), то значение функции — имя файла, соответствующее шаблону. Чтобы получить имена остальных файлов, соответствующих шаблону, надо вызвать функцию `Dir` еще раз, но без параметров. Например:

```
fn =Dir("c:\temp\*.bmp")
```

```
fn = fn + Chr(13) + Dir
```

Параметр `Attributes` задает (уточняет)

тип файла: `Normal` (0), `Readonly` (1),

`Hidden` (2), `System` (4), `Directory`

(16) — каталог.

Примеры:

`Dir ("e:\test.txt")` — возвращает "test.txt", если файл `test.txt` существует на диске `e:`;

`Dir ("e:\t*.txt")` — возвращает имя первого найденного в каталоге `e:\t` файла

с расширением txt;
Dir ("e:\", vbDirectory) — возвращает имя первого (по порядку) подкаталога корневого каталога диска e:
Функция *CurDir* (без параметров) возвращает полное имя текущей (рабочей) папки. Сразу после запуска программы текущая папка — это папка, из которой запущена программа
Функция *ChDir* задает текущий (рабочий) каталог
Глава 9. Функции 277

Таблица 9.5 (окончание)

Функция Описание

MkDir Path Функция *MkDir* создает новый каталог.

Параметр *Path* задает путь к новому каталогу и его имя. При попытке создать каталог в несуществующей папке возникнет ошибка

Rmdir path Функция *Rmdir* удаляет каталог. Параметр *Path* задает полное имя каталога, который надо удалить. При попытке удалить каталог, в котором есть файлы, возникнет ошибка. В этом случае нужно сначала из него удалить файлы (функция *Kill*), и только после этого можно удалить сам каталог

Kill PathName Функция *Kill* удаляет файл. Параметр *PathName* (полное имя файла) задает файл, который надо удалить. Если в качестве имени задан шаблон, то будут удалены все файлы, имена которых соответствуют указанному шаблону. Например:
Kill "c:\temp.tmp"*

Глава 10

События

В этой главе дано краткое описание основных событий *Windows*.

Таблица 10.1. События

Событие

Click Происходит при щелчке кнопкой мыши

Dblick Происходит при двойном щелчке кнопкой мыши

MouseDown Происходит при нажатии кнопки мыши

MouseUp Происходит при отпускании кнопки мыши

MouseMove Происходит при перемещении мыши .

KeyPress Происходит при нажатии клавиши клавиатуры

KeyDown Происходит при нажатии клавиши клавиатуры. События *KeyDown* и *KeyPress* — чередующиеся или повторяющиеся, которые происходят до тех пор, пока не будет отпущена удерживаемая клавиша (в этот момент происходит событие *KeyUp*)

KeyUp При отпускании нажатой клавиши клавиатуры

Load Происходит при создании объекта (формы, элемента управления). Процедура обработки этого события обычно используется для инициализации переменных или выполнения подготовительных действий

Unload Происходит при закрытии формы или выгрузке какого-либо объекта

Глава 9. Функции 279

Таблица. 10.1 (окончание)

Событие Характеристика

Paint Происходит при появлении окна на экране в начале

работы программы, после появления части окна, которая, например, была закрыта другим окном и в других случаях. Событие сообщает о необходимости обновить (перерисовать) окно
GotFocus Происходит при получении элементом управления фокуса
LostFocus Происходит при потере элементом управления фокуса
Resize Происходит при изменении размеров формы

Приложение

Содержание компакт-диска, прилагаемого к книге Культина Н. Б. "Visual Basic. Освой на примерах"

CD-ROM содержит исходные тексты программ, выполняемые файлы и необходимые для работы программ файлы данных. Каждая программа находится в отдельном каталоге. Большинство программ не требуют для своей работы никаких дополнительных программных компонентов (библиотек) и могут быть запущены непосредственно с *CD-ROM*. Некоторые программы, например программы работы с базами данных, требуют, чтобы база данных была зарегистрирована в системе как источник данных *ODBC*. Создать источник данных можно при помощи Администратора источников данных *ODBC* — стандартной утилиты *Windows* (команда **Пуск** / **Настройка** / **Панель управления** / **Администрирование** / **Источники данных**). Для активной работы, чтобы иметь возможность вносить изменения в программы, скопируйте каталоги проектов на жесткий диск компьютера.

Предметный указатель

К

Компонент:

CheckBox 236

ComboBox 239

CommandButton 234

Common Dialog 254

DirListBox 243

DriveListBox 242

FileListBox 244

Image 248

Label 231

Line 251

ListBox 238

MMControl 255

OptionButton 237

PictureBox 246

Shape 249

TextBox 233

Timer 241
UpDown 252

М

Математические функции 266
Метод Print 257

П

Программа:

"Будильник" 180
"Калькулятор" 25
"CD-плеер" 101
"ID3v1 Tag Editor" 118
"Аудио-плеер" 107
"Будильник" 176
"Ежедневник" 217
"Записная книжка" 215
"Звуки Windows" 99
"Игра 15" 141
"Игра Puzzle" 147
"Игра Парные картинки" 154
"Игра Сапер" 165
"Поиск файла" 135
"Таймер" 43, 46, 50
"Тир" 58
"Чтение файла" 129
"Экзаменатор" 195
"Электронные часы" 39, 40
анимации изображений 85
вывода анимационного ролика 87
пересчета веса 9
пересчета скорости ветра 6, 7, 8
просмотра иллюстраций 92
просмотра текстовых файлов 133
расчета:
дохода по вкладу 14
силы тока напряжения или сопротивления эл. цепи 16, 18
скорости бега 11
стоимости поездки на автомобиле 22

формирования:

изображения оцифрованной координатной сетки 66
"бегущей" строки 91
графика функции 69
изображения гистограммы 72
изображения

флага РФ 61
изображения
часов 77, 81
контура пятиконечной
звезды 55
кривой Гильберта 62
фонового рисунка 97
прокрутки текста 89

С

Свойства:

компонента:

CheckBox 236

ComboBox 239

CommandButton 234

CommonDialog 254

DirListBox 243, 244

DriveListBox 242

Image 248

Label 231

Line 251

ListBox 238

MMControl 255

OptionButton 237

PictureBox 246

Shape 249

TextBox 233

Timer 241

UpDown 252

формы 229

Т

Твип 229

Ф

Форма 229

Функции:

даты и времени 271

для работы

с файлами 273

для работы

со строками 268

преобразования 267

InputBox 264

MsgBox 265