Министерство общего и профессионального образования
Российской Федерации

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

А.С. ТРЕТЬЯКОВ

ПРИКЛАДНОЕ ПРОГРАММИРОВАНИЕ

Ч. 1: Системы быстрой разработки приложений

Утверждено
Редакционно-издательским советом
в качестве учебного пособия
для студентов дневного отделения МТФ

НОВОСИБИРСК
1999

Третьяков А.С. Прикладное программирование: Учеб. пособие. – Новосибирск: Изд-во НГТУ, 1999. – Ч. 1: Системы быстрой разработки приложений. – 50 с.

Пособие состоит из двух частей, в первой приводится краткое изложение основных принципов разработки прикладных программ с использованием среды RAD (быстрой разработки приложений) для решения инженерных задач.

Материал излагается на примере системы Visual Basic 5 (VB). Хотя данное издание нельзя рассматривать как систематическое введение в программирование на VB, материала настоящего пособия вполне достаточно для разработки прикладных программ инженерных расчетов.

В приложении приводится список ключевых слов языка, а также рассматриваются примеры программирования на Visual Basic for Application – диалекте, предназначенном для разработки приложений, работающих в среде Microsoft Office.

Рецензенты: *Е.В. Фрейдина*, д-р техн. наук, проф.
Б.Р. Долгин, канд. техн. наук, доц.

Работа подготовлена на кафедре
автоматизации производственных процессов в машиностроении

ОГЛАВЛЕНИЕ

Стр.

Глава 1. БЫСТРОЕ ЗНАКОМСТВО С VISUAL BASIC

Visual Basic (далее – VB) представляет собой типичный пример системы быстрой разработки приложений, включает в себя как язык программирования, наиболее доступный для освоения, так и интуитивно понятую среду разработки.

Обладая всеми свойствами современных систем программирования, а именно объектно-ориентированным подходом к программированию, обработкой событий, проектированием экранного интерфейса с помощью предопределенных визуальных объектов, этот язык тем не менее сохранил простоту и доступность Basic"а, с которым читатель, возможно, знакомился в школе.

Создание программы на VB состоит из трех этапов:

- *разработка экранной формы, состоящая в размещении элементов интерфейса;*
- *определение свойств каждого элемента интерфейса;*
- *определение программ по обработке событий.*

Терминологическое замечание

Справочная система VB íà ðóññêîì ÿçûêå содержит следующее определение: "***Элемент управления*** – это графический объект, например поле, флажок, прямоугольник или кнопка, размещаемый пользователем в форме или отчете в режиме конструктора для отображения данных или упрощения чтения формы или отчета." Автор предлагает вместо термина "***элемент управления***" использовать термин "***элемент интерфейса***", как более соответствующий содержанию данного понятия.

Запуск VB

Запуск системы VB, как и любой другой программы в системе Windows, производится путем щелчка мышью по соответствующей иконке на рабочем столе или выбором VB в системе меню "Программы". После запуска VB на экране компьютера, как правило, появляется новая форма, не содержащая ни одного элемента интерфейса. При повторном запуске VB для возвращения к разработке какого-либо уже сохраненного приложения следует воспользоваться меню "Файл".

Экран VB содержит строку горизонтального меню, может содержать одну или более панелей инструментов, а также несколько окон, служащих для разработки программ пользователя. При первом запуске VB на экране компьютера появляется так называемая стандартная конфигурация среды разработки. Элементами стандартной конфигурации являются панель элементов Standart (стандартная), окна Project (проект), содержащие при запуске VB

пустую экранную форму, Tollbox (инструментарий), Project Explorer (просмотр проекта), Properties (свойства), Form Layout (рамещение формы). Во время работы со средой VB вы можете изменить состав и расположение окон на экране, сделав его более удобным для выполнения каждой конкретной работы. VB запоминает конфигурацию и при последующих запусках вы получите не стандартное, а свое, измененное расположение окон на экране.

Создание простейшей формы

Для того чтобы создать форму, реализующую простейшую программу суммирования двух чисел, на пустую экранную форму следует поместить перечисленные ниже элементы интерфейса:

- TextBox (текстовое поле). Этот элемент интерфейса служит для ввода информации в программу и вывода на экран значений, получаемых в результате выполнения программы. Создаваемая форма должна содержать два элемента класса TextBox, служащих в данном случае для ввода чисел, подлежащих сложению.
- Label (метка). Предназначен для размещения на форме различных поясняющих надписей и вывода результатов работы программы; в данном случае этот элемент интерфейса служит для вывода результата суммирования.
- CommandButton (командная кнопка). Предназначен для запуска подпрограмм обработки данных, в данном случае – подпрограммы суммирования двух чисел.

Размещение на форме элементов интерфейса производится при помощи мыши, при этом используется содержимое окна Toolbox. Если этого окна нет на экране, откройте его, нажав на иконку “Toolbox”, на которой изображены молоток и гаечный ключ, или через главное меню VB, пункты меню View – Toolbox.

Для помещения на форму конкретного элемента следует выбрать его в окне Toolbox, нажать левую кнопку мыши (при этом пиктограмма выбранного элемента станет “включенной”), отпустить кнопку мыши, позиционировать курсор в точку на форме, где будет располагаться левый верхний угол создаваемого элемента, вновь нажать кнопку мыши и задать требуемый размер элемента интерфейса.

Для изменения местоположения элементов интерфейса следует расположить курсор внутри требуемого элемента, нажать кнопку мыши (“выделить объект”) и, не отпуская кнопки, переместить.

Для изменения размера следует выделить объект, затем поместить курсор на один из ограничивающих элемент интерфейса маркеров (небольших квадратов) и придать ему требуемый размер.

Обратите внимание: форма также является элементом интерфейса и к ней применима общая процедура изменения размеров, однако для изменения местоположения формы относительно экрана при выполнении программы следует использовать окно Form Layout (расположение формы).

Задание свойств элементам интерфейса

После включения в форму элементов интерфейса следует задать свойства (параметры) как каждого объекта, так и формы в целом. Каждый класс элементов интерфейса имеет свой собственный набор свойств, всю совокупность свойств можно посмотреть и изменить любое свойство, используя одно из упомянутых выше окон – Properties (свойства).

Самый простой способ вывести на экран окно Properties (если его нет на экране) – нажать клавишу F4. Для того чтобы работать со свойствами конкретного элемента интерфейса, необходимо выделить этот элемент.

Рассмотрим подробнее окно свойств. Вверху находится собственное имя выделенного объекта и наименование класса элементов интерфейса. Окно свойств состоит из двух колонок, в первой приведены наименования свойств, которыми обладает объект данного класса, во второй – установленные значения. Список свойств может появляться в двух видах – упорядоченный по алфавиту и сгруппированный по категориям, однако при алфавитном порядке первым является свойство Name – имя объекта. Имя употребляется в подпрограммах обработки событий для обращения к свойствам данного объекта и никак не влияет на его видимое изображение. При создании каждого нового элемента интерфейса VB автоматически создает уникальное имя для вновь созданного объекта, поэтому для небольших программных проектов свойство “имя” можно не устанавливать, однако для облегчения работы с объектами им следует задавать имена, несущие смысловую нагрузку. Например, для главной формы можно задать имя Main (главная).

Каждой свойство имеет свое значение по умолчанию (таким значением, в частности, может быть “пустое”).

Важно запомнить, что свойство Name по умолчанию состоит из наименования класса объектов (возможно, сокращенного) и порядкового номера, например Label1.

По способу изменения свойства можно разделить на три группы: свойства, значения которых выбираются из заранее определенного списка; свойства, произвольно задаваемые пользователем, и свойства, для установления значения которых используется специальное окно диалога.

Задать значение свойства можно, выделив строку с его наименованием в окне свойств.

При этом если свойство задается произвольным образом, путем его задания набором значения на клавиатуре, то поле с его значением (рядом с наименованием) при выделении строки остается без изменений. Примером такого свойства может быть свойство Caption-надпись. Этим свойством, например, обладает форма.

Если свойство устанавливается путем выбора из списка, то в поле значения после его активизации появляется треугольник, направленный вниз. После нажатия на это треугольник раскрывается список, из которого и выбирается значение.

Если при выборе значений свойств требуется произвести более сложные действия, например открыть файл или выбрать шрифт (это означает выбрать начертание шрифта, размер знаков и т.д.), то в поле значения появляется кнопка с многоточием, после нажатия на которую раскрывается соответствующее данному свойству диалоговое окно.

Рассмотрим некоторые свойства, которыми обладают объекты используемых в рассматриваемом примере классов. Подчеркнем: каждый класс объектов обладает своим собственным набором свойств, однако множества свойств для различных классов имеют, как сказал бы математик, непустое пересечение.

Caption-надпись – значением этого свойства является текст, выводимый на объекте или в заголовке объекта. Применяется для объектов классов Form, Label, CommandButton.

Font-шрифт – устанавливает шрифт, используемый для вывода символов. Свойством обладают объекты классов Form, Label, CommandButton, TextBox.

ToolTipText-подсказка – устанавливает текст подсказки, выводимой на экран в случае, если курсор мыши задерживается на объекте (вывод текста производится только во время выполнения программы). Свойством обладают объекты классов Label, CommandButton, TextBox.

Итак, для создания простейшей программы, после расположения на форме элементов интерфейса следует, во время разработки программы, установить следующие свойства объектов:

Объект класса Form с именем Form1 (это имя, как и другие имена рассматриваемого примера, автоматически установлено системой VB):

```
Caption = “Âû÷èñëåíèå ñóììû’.
```

Объект класса TextBox с именем Text1:

```
Text=0;
TollTipText= “Ââåäèòå ïåðâîå ñëàãàåìîå”.
```

Объект класса TextBox с именем Text2:

```
Text=0;
TollTipText= “Ââåäèòå âòîðîå ñëàãàåìîå”.
```

Объект класса Label с именем Label1:

```
Caption = “Сумма”;
Font = “MS San Serif, полужирный, размер 18”
‘( установлено с использованием окна диалога).
```

Объект класса CommandButton с именем Command1:

```
Caption =”Нажми – получишь результат”;
Font = “MS San Serif, полужирный, размер 14”
‘( установлено с использованием окна диалога).
```

Задание подпрограмм обработки событий

Все действия программа, разрабатываемая в среде VB, выполняет в ответ на различные события. События могут быть как внешними, например нажатие кнопки пользователем, так и внутренними, возникающими при выполнении программы.

Разработка нашей простейшей программы будет завершена, если мы зададим подпрограмму – процедуру обработки нажатия командной кнопки, по которой будет вычислена сумма двух чисел.

Исходная информация для программы – слагаемые – вводится в поля, образованные элементами интерфейса TextBox. Доступ к содержимому осуществляется при помощи обращения к свойству Text, т.е., чтобы использовать число, введенное в поле ввода объекта Text1, следует написать Text1.Text. Заметим, что по умолчанию содержимое этого поля имеет тип “строка”, и для использования его как числа следует произвести преобразование при помощи функции VAL().

Подпрограмма обработки записывается в так называемом окне кода. Для открытия этого окна и записи подпрограммы дважды нажмите кнопку мыши на объекте (произведите “DubleCick”).

В окне кода автоматически появятся первая и последняя строки подпрограммы обработки – оператор заголовка.

```
Private Sub Command1_Click()
End Sub
```

Подпрограмма обработки нажатия кнопки мыши совершает вычисление суммы чисел из полей ввода Text1 и Text2, помещает значение в поле Label (переустанавливая значение свойства Captoin), затем изменяет заголовок формы. Приведенные ниже две строки, которые и составляют собственно подпрограмму обработки, размещаются между приведенными выше операторами Sub è End Sub.

```
Label1.Caption = Val(Text1.Text) + Val(Text2.Text)
Form1.Caption = "Работа выполнена"
```

Следует принять за правило не записывать самому (вручную) операторы начала и конца подпрограмм обработки событий. Это за нас быстрее и, что более важно, абсолютно правильно сделает система VB.

Из рассмотренного примера следует, что свойство Caption можно задавать как при разработке программы, так и изменять его динамически во время выполнения программы. Заметим, что это относиться не ко всем свойствам объектов.

Запуск программы на выполнение

Для выполнения программы следует нажать кнопку Start (треугольник острием направо)на панели инструментов. Для прекращения работы программы и возвращения в среду разработки следует нажать кнопку End (небольшой квадрат) на панели инструментов или стандартную кнопку закрытия окна на главной форме выполняемой программы.

Глава 2. БАЗИСНЫЕ ПОНЯТИЯ RDA

В настоящее время широкое распространение получили системы быстрой разработки приложений (Rapid Development of Application или RDA), экранный интерфейс прикладных программ в которых создается на основе некоторого множества базисных элементов. Состав этого множества базисных элементов интерфейса определен операционной системой Windows и практически одинаков во всех RDA. Ознакомившись с элементами VB, вы сможете в дальнейшем без особых проблем работать как с прикладными программами в среде Windows, так и с другими системами RDA.

В гл. 1 было показано, что каждый конкретный объект, принадлежащий к конкретному классу элементов интерфейса, обладает набором переопределяемых свойств.

Кроме того, на объекты каждого класса можно воздействовать набором процедур (подпрограмм), называемых в системах RDA методами. Метод-процедура, действующая на конкретный программный объект и изменяющая его свойства. Например, каждый рассмотренный ранее элемент интерфейса, отображаемый на экранной форме, обладает свойствами, характеризующими его положение на форме, а именно: Height (высота), Width (ширина), Top (положение по вертикали), Left (положение по горизонтали). Для изменения расположения объекта может быть использован метод Move (переместить), применяемый к конкретному объекту и изменяющий два его свойства – Top и Left.

В основе программ , разрабатываемых в системах RDA, лежит концепция программы, управляемой событиями. По сути, это означает, что программный код, написанный разработчиком, не выполняется последовательно, а состоит из отдельных процедур обработки событий. Примером программного события может служить нажатие пользователем программной кнопки из примера предыдущей главы. Ряд наиболее важных программных событий будет рассмотрен далее в этой главе.

Процесс “жизни” программы состоит из двух главных этапов: разработка или конструирование программы (design time) и время выполнения программы (run time). Большинство свойств объектов можно изменять как при конструировании, так и при выполнении программы. Но существуют и исключения, имеются свойства, доступные только для чтения (Read Only) âî âðåìÿ âûïîëíåíèÿ, è свойства, âîâñå íå äîñòóïíûå ïðè êîíñòðóèðîâàíèè. (Ðèñóíîê ïðèâîäèòñÿ â ýòîé ãëàâå.)

Метод, по своей сущности, применим к объекту только во время выполнения программы.

Состав основных элементов интерфейса

Основные элементы интерфейса, стандартно представленные в Tollbox, можно разделить на несколько групп.

- Поля ввода-вывода информации: TextBox (текстовое поле) и Label (надпись).
- Списки: ComboBox (поле со списком), ListBox (список).
- Полосы прокрутки HScrollBar (горизонтальная полоса), VScrollBar (вертикальная полоса).
- Элементы оформления: Frame (рамка), PictureBox (графическое поле), Image (рисунок), Line (линия), Shape (фигура).
- Элементы управления процессом выполнения программы: CommandButton (кнопка), CheckBox (флажок), OptionBox (переключатель).
- Элементы интерфейса, связанные с файловой системой компьютера: DriveListBox, DirListBox, FileListBox.
- Прочие элементы: Timer (таймер), Data (доступ к базам данных), OLE (вставка объектов других приложений).

В этой главе мы рассмотрим элементы интерфейса из групп 1...5, а также таймер. Группу 6 рассматривать не будем, так как функции входящих в нее элементов выполняет один новый элемент интерфейса Common Dialog (стандартный диалог), не входящий стандартно в ToolBox и рассматриваемый ниже. Элемент интерфейса Data рассматривается в гл. 6.

Приведем ряд свойств, которыми обладают все или почти все рассматриваемые в главе элементы интерфейса. Список свойств будет соответствовать порядку вывода в окно свойств "по категориям", описание дается в краткой форме и предполагает последующее обращение к справочной системе.

0Группа свойств Appearance - внешний вид

Appearance – стиль вывода объекта на экран, трехмерный или плоский *(доступно только для чтения во время выполнения)*.

BackColor – öâåò ôîíà îáъåêòà

BackStile – ñòèëü çàïîëíåíèÿ. Ïðè óñòàíîâêå çíà÷åíèÿ ýòîãî ñâîéñòâà transpsrent объект становится полупрозрачным.

Caption – надпись на объекте.

ForeColor – цвет выводимого изображения, в том числе текста.

1Группа свойств Behavior - ïîâåäåíèå

Enabled определяет, отвечает ли объект на события, генерируемые пользователем, например на нажатие кнопки мыши на объекте.

TabIndex определяет последовательность обхода объектов формы как во время разработки, так и при выполнении программы. Для перехода от одного объекта к другому можно использовать клавишу табуляции.

Visible – âèäèìîñòü îáúåêòà âî âðåìÿ âûïîëíåíèÿ ïðîãðàììû. Âî âðåìÿ ðàçðàáîòêè âñå îáúåêòû âèäèìû.

2Группа свойств Data -источник данных

DataField – ïîëå äàííûõ.

DataSource – источник данных.

Эти свойства употребляются для связи элемента интерфейса с базами данных, подробно рассматриваются в соответствующей главе.

3Группа свойств из единственного свойства Font - шрифт

4Группа свойств Position -позиция

Left, Top, Width, Height устанавливают размеры объекта. Единицы измерения будут рассмотрены позже.

Использование ComboBox

Этот элемент (поле со списком) позволяет во время выполнения программы выбирать одно значение из списка предлагаемых значений. Активизация элемента производится нажатием мышью на треугольник, находящийся в поле элемента. Список значений задается статически при разработке программы или динамически во время выполнения.

Свойства ComboBox:

`Text` – текст, выводимый в поле вывода объекта при разработке формы и при выполнении программы, но до выбора какого-либо элемента из предлагаемого списка.

`List(n)` – элемент списка с номером n. Нумерация элементов в списке начинается с нуля.

`ListIndex` – номер выбранного элемента списка.

Методы:

`AddItem` добавляет в список новый элемент.

В качестве примера рассмотрим форму со следующими элементами интерфейса: ComboBox с именем (name) Combo1; CommandButton с именем Command1; два элемента Label.

Программный код состоит из двух подпрограмм-обработчиков событий.

Напомним: программный код записывается в специальном окне – окне программного кода. Для перехода в это окно следует выполнить команду меню View-Code, или начать непосредственный ввод текста подпрограммы обработки (см. ниже).

Первое событие, возникающее при выполнении программы, – загрузка на экран главной формы. Это событие имеет наименование Load, во время его обработки удобно производить действия по инициализации переменных, используемых в программе, устанавливать те свойства объектов, которые невозможно установить во время конструирования. Соответствующая подпрограмма имеет имя Form_Load, для получения ее заголовка и завершающего оператора End Sub следует просто щелкнуть мышью на форме в месте, свободном от других элементов интерфейса. Пример подпрограммы обработки события

иллюстрирует также синтаксис использования методов: при обращении к методам следует указывать объект и через точку – название метода.

Второе событие возникает при нажатии командной кнопки.

Приведем текст подпрограмм с комментариями, начало комментария обозначается апострофом.

```
Private Sub Form_Load() 'заголовок подпрограммы,
'указывающий, что она выполняется при загрузке формы
Combo1.AddItem "Я знаю VB  хорошо”'добавление в спи-
               сок элемента
Combo1.AddItem "Я знаю VB плохо"
Combo1.AddItem "Я понимаю только английский язык"
Combo1.Text = Combo1.List(0)'при запуске программы
               на экран
'выводится значение первого элемента из списка
End Sub

Private Sub Command1_Click()'заголовок подпрограммы,
'указывающий, что она выполняется при нажатии кнопки
Label1.Caption = Combo1.List(Combo1.ListIndex)
'вывод выбранного из списка 'элемента
Label2.Caption = Combo1.ListIndex
'вывод номера выбранного из списка элемента
End Sub
```

Приведенный пример иллюстрирует динамическое задание списка. Для статического задания просто перечислите все требуемые значения в окне, открывающемся при изменении свойства List.

Обратите внимание: Свойство ListIndex отсутствует в списке окна свойств для объектов класса ComboBox во время разработки программы. Это означает, что данное свойство доступно только во время выполнения программы.

Использование HScrollBar

Свойства HScrollBar:

Value – значение, определяющее позицию бегунка на полосе прокрутки.

Max, Min – ìàêñèìàëüíîå è ìèíèìàëüíîå çíà÷åíèÿ, êîòîðûå ìîæåò ïðè-íèìàòü Value.

LargeChange, SmallChange – шаг изменения значения Value.

Горизонтальная полоса прокрутки позволяет изменять значение свойства Value с помощью позиционирования бегунка на полоске. Изменять положение бегунка с использованием мыши можно тремя способами:

• Позиционировать курсор мыши на бегунок, нажать кнопку мыши и перетащить бегунок. При этом значение Value изменяется с шагом 1.

• Позиционировать курсор мыши на треугольник полосы прокрутки и нажать кнопку мыши. Значения Value изменяется с шагом, задаваемым свойством SmallChange.

• Позиционировать курсор мыши на полосу прокрутки слева или справа от бегунка и нажать кнопку мыши. Значения Value изменяется с шагом, задаваемым свойством LargeChange.

Для изучения свойств полосы прокрутки рассмотрим форму с элементами HScrollBar и TextBox и следующим программным кодом:

```
Private Sub Form_Load()
HScroll1.Min = 0
HScroll1.Max = 1000
HScroll1.LargeChange = 100
HScroll1.SmallChange = 1
End Sub

Private Sub HScroll1_Change()
Text1.Text = HScroll1.Value
End Sub
```

Вторая подпрограмма обрабатывает событие Change – изменение, возникающее при изменении значения, сопоставленного элементу управления. Для элемента Hscroll это событие произойдет при перемещении движка.

Заметим, что первая подпрограмма в данном случае не является необходимой, значения свойств можно задать и в режиме конструирования.

Работа с вертикальной полосой прокрутки (VScollBar) аналогична работе с горизонтальной полосой.

Примечание: для некоторых приложений полезно не использовать бегунок для задания значения свойства Value, à позиционировать его на полосе прокрутки при помощи изменения Value программным путем.

Элементы оформления

Элемент интерфейса Frame (рамка) служит для объединения нескольких других элементов в единое целое. Все размещенные в рамке объекты могут обрабатываться (например, передвигаться по форме) как единое целое.

Элементы интерфейса PictureBox и Image предназначены для вывода графических изображений. Эти элементы интерфейса во многом подобны, работа программы с элементом Image ïðîèçâîäèòñÿ áûñòðåå, но Image поддерживает лишь подмножество свойств объектов класса PictureBox. Íàçíàåì îäíî ðàðàêòåðíûå äëÿ ýòèõ ýëåìåíòîâ ñâîéñòâî – Picture, óêàçûâàþùåå íà ôàéë, содержимое которого будет выводиться.

Следующие два элемента интерфейса выводят геометрические фигуры. Line выводит на экран отрезок прямой, Shape ðèñóåò геометрический примитив, устанавливаемый его же свойством Shape. Могут быть нарисованы следующие фигуры: прямоугольник, квадрат, овал, окружность, прямоугольник со скругленными углами, квадрат со скругленными углами.

Использование CheckBox и OptionButton

Данные элементы интерфейса предназначены для выбора одного или нескольких вариантов из множества предложенных. Эти элементы (по-русски – “флажок” и “переключатель”) функционально подобны, но имеют одно принципиальное отличие: любое количество элементов “флажок” может быть выбрано одновременно, но в каждой группе переключателей может быть выбран только один элемент “переключатель”.

Для создания группы переключателей несколько элементов располагаются в контейнере, которым могут служить Frame (рамка). Для того чтобы сгруппировать переключатели в рамке, разместите первоначально на форме рамку, и только затем разместите внутри ее переключатели.

Следующие примеры иллюстрируют использование флажков и переключателей. Создайте форму, использующую флажки. Разместите на форме рамку, в рамку поместите три флажка, кроме того, разместите на форме три элемента Shape è îäíó êîìàíäíóþ êíîïêó. Текст программы, которую следует ввести в окно программного кода, содержит условные операторы If…Then…Else, реализующие действия, устанавливаемые флажками.

```
Private Sub Command1_Click()
If Check1.Value = 1 Then
    Shape1.Visible = True
Else
    Shape1.Visible = False
End If
If Check2.Value = 1 Then
    Shape2.Visible = True
Else
    Shape2.Visible = False
End If
If Check3.Value = 1 Then
   Shape3.Visible = True
Else
   Shape3.Visible = False
End If
End Sub
```

Для формы с переключателями соответствующая подпрограмма обработки нажатия клавиши приведена ниже.

```
Private Sub Command1_Click()
If Option1.Value = True Then
    Shape1.Visible = True
Else
    Shape1.Visible = False
End If
If Option2.Value = True Then
    Shape2.Visible = True
```

```
Else
    Shape2.Visible = False
End If
If Option3.Value = True Then
    Shape3.Visible = True
Else
    Shape3.Visible = False
End If
End Sub
```

Элемент Timer

Элемент интерфейса Timer относится к так называемым невизуальным элементам, которые не видны на форме во время выполнения программы. Назначение таймера – выполнять заданную подпрограмму каждый раз по истечении определенного интервала времени, который устанавливается свойством Interval. Если значение свойства Interval равно нулю, то таймер "не срабатывает", т.е. программное событие не возникает и программа обработки не вызывается.

Для ознакомления с элементом таймер приведем пример программы, один из элементов формы в которой динамически меняет цвет, причем частота изменения задается во время выполнения программы.

Создайте форму с элементами интерфейса `Timer, HScollBar, PictureBox`. Обратите внимание на способ задания цвета вызовом функции `RGB` – задание цвета в виде тройки чисел – интенсивностей красного (RED), зеленого (GREEN) и голубого (BLUE) цвета. Каждая цветовая компонента выражается целым числом от 0 до 255.

```
Private Sub Form_Load()'установим начальное значение
                       интервала
Timer1.Interval = 900'параметр – минимальное значение
HScroll1.Min = 100
HScroll1.Max = 900
End Sub

Private Sub HScroll1_Change()
'установка интервала согласно местоположению бегунка
Timer1.Interval = 1000 - HScroll1.Value
End Sub

Private Sub Timer1_Timer()
'переключение цвета фона между красным и синим
If Picture1.BackColor = RGB(255, 0, 0) Then
    Picture1.BackColor = RGB(0, 0, 255)
Else
    Picture1.BackColor = RGB(255, 0, 0)
End If
End Sub
```

Элементы управления и программные события

Подытожим сведения о программных событиях, изложенные в этой главе. Каждый элемент интерфейса, при воздействии на него со стороны пользователя, может генерировать некоторые программные события. К числу наиболее часто используемых относятся события:

- Click – нажатие кнопки мыши на объекте;
- DblClick – двойное нажатие кнопки мыши на объекте;
- Change – изменение значения, связанного с объектом.

Кроме того, для инициализации программы используется событие Load, свеязанное с формой.

Для ввода текста подпрограммы обработки события следует открыть окно программного кода, в левом элементе поле со списком выбрать объект, а в правом элементе – наименование события.

Глава 3. ВВЕДЕНИЕ В СРЕДУ РАЗРАБОТКИ VB

В этой главе будет кратко перечислены особенности среды разработки Visual Basic версии 5.0. Несмотря на то, что каждая из систем RAD имеет свои особенности по интерфейсу, они в целом подобны, особенно на уровне первоначального освоения, и поэтому, настоящую главу можно назвать "Введение в среду разработки RAD".

Состав файлов проекта

Итак, в процессе разработки в среде VB создается так называемый проект, представляющий собой совокупность файлов на диске, создаваемых в процессе разработки программы. Проект включает в себя файлы экранных форм, файлы текстов программных модулей и файл, содержащий информацию о проекте в целом.

В простом случае, при использовании одной формы, совокупность файлов проекта состоит из трех файлов – файла с расширением FRM и именем, совпадающим с именем формы, а также двух файлов с именем, совпадающим с именем проекта, и имеющих расширения имени VBP и VBM.

Если форма использует какие-либо графические элементы (рисунки, иконки и т.д.), дополнительно создается файл с расширением FRX и именем, совпадающим с именем формы.

При переписывании файлов проекта с одного компьютера на другой следует переписывать все относящиеся к проекту файлы.

Создание выполняемого (EXE) файла программы

Создать выполняемый файл достаточно просто. Для этого следует открыть соответствующий проект и вызвать окно Make Project, используя команды меню File – Make.

В этом окне можно задать имя файла и некоторые описывающие его опции, которые будут доступны при выполнении программы.

Exe-файл может выполняться на любом компьютере, на котором установлен VB. Для того чтобы файл мог выполняться на компьютере без VB, следует записать на него еще и системные файлы. Для создания программы Setup, которая производит все необходимые для установки вашего программного обеспечения действия, следует воспользоваться Мастером создания программ установки (Application Setup Wizard). Этот мастер вызывается из программной группы Visual Basic 5.0 и, как все мастера среды Winows, обладает понятным интерфейсом и не требует дальнейших комментариев.

Отладка программ

В таблице приведены описания инструментов, доступных в панели Отладка. Для открытия панели инструментов Debug (Отладка) следует либо щелкнуть по панели Стандартная и в появившемся меню выбрать Debug (отладка), либо вызвать ее через меню View – Toolbars – Debug.

Пиктограммы инструментов, размещенные на панели, дублируют команды, доступные через панель Стандартная и через меню, но в панели Отладка они собраны вместе.

Èíñòðóìåíò	Клавиши вызова	Описание
Run Запуск	**F5**	Запускает текущий проект
Break Îñòàíîâêà	**Ctrl + Break**	Останавливает выполнение программы, после остановки можно использовать инструменты отладки
End Êîíåö		Завершает работу
Toddle Breakpoint Òî÷êè îñòàíîâêè	**F9**	Устанавливает или удаляет точку остановки в строке, на которой находится курсор
Step Into Øàã âíóòðü	**F8**	Выполняет один оператор. Если это оператор вызова процедуры, выполнение останавливается после входа в процедуру
Step Over Øàã ÷åðåç	**Shift + F8**	Выполняет один оператор. Если это оператор вызова процедуры, выполняется вся процедура
Step Out Øàã íàðóæó	**Ctrl + Shift + F8**	Выполняет все операторы текущей процедуры

Окончание таблицы

Èíñòðóìåíò	Клавиши вызова	Описание
Locals Window Îêíî ëîêàëüíûõ ïåðåìåííûõ		Открывает окно, в котором во время выполнения выводятся значения всех локальных переменных процедуры
Immediate Window Îêíî âûðàæåíèé	**Ctrl + G**	Открывает окно, в которое можно ввести любое выражение и выполнить его
Watch Window Îêíî ïðîñìîòðà		Открывает окно, в котором можно просмотреть значения выбранных переменных. Редактировать список выводимых переменных можно нажав правую кнопку мыши на окне
Quick Watch Áûñòðûé ïðîñìîòð	**Shift + F9**	Позволяет быстро просмотреть значение программного элемента, на котором находится курсор, и, при желании, добавить этот элемент в окно Watch
Call Stack Ñòåê	**Ctrl + L**	Âûâîäèò ñïèñîê ïîñëåäîâàòåëüíîñòè âûçîâà ïðîöåäóð

Еще две команды, доступные в меню Debug, не вошли в панель Отладка, но очень полезны в работе.

Run to Cursur Запуск	**Ctrl + F8**	Запускает выполнение программ и останавливает его в строке, на которой находится курсор
Clear All Breakpoints Удаление всех точек îñòàíîâêè	**Ctrl + Shift + F9**	Удаляет все ранее введенные точки остановки

Примечания:

1. Точку остановки можно задать или удалить, щелкнув по панели окна программного слева от текста соответствующего оператора.

2. Во время остановки программы при установке курсора мыши на программный объект выводится его текущее значение.

Глава 4. ТИПЫ ДАННЫХ И УПРАВЛЯЮЩИЕ СТРУКТУРЫ ПРОГРАММЫ VB

Данная версия VB позволяет использовать переменные различных типов, формально описывая их с помощью специальных операторов. Имеется также возможность не описывать переменные, при этом тип переменной определяется из контекста. В VB, как и в любом другом языке программирования, существуют управляющие конструкции, задающие порядок выполнения операторов. К таким конструкциям относятся процедуры, циклы, ветвления.

Процедуры, во-первых, позволяют разбивать текст на небольшие, ясные для понимания фрагменты, во-вторых, дают возможность выделять часто используемые группы операторов с целью устранения их повторения. В VB имеются два типа процедур: подпрограммы (Sub) и функции (Function).

Циклы позволяют многократно выполнять определенную последовательность операторов, проверка условия продолжения цикла происходит перед или после каждой итерации.

Ветвления позволяют, в зависимости от выполнения условий, производить те или иные действия, выполнять альтернативные последовательности операторов.

Описания переменных, все процедуры обработки событий, возникающих в конкретной форме или в ее элементах управления, объединены в так называемый Модуль формы. Кроме того, в модуль формы включают другие вспомогательные процедуры Sub и Function, относящиеся к данной форме.

Терминологическое замечание

Справочная система VB на русском языке использует слово "**инструкция**" для английского термина **Statment.** По мнению автора, здесь более пригоден традиционный термин "**оператор**".

Переменные

Переменные в VB имеют идентификатор и тип. Идентификатор должен начинаться с буквы и может иметь любую комбинацию букв и цифр. Возможно использование символов кириллицы. Тип переменной может следующим:

- **Byte** – целое от 0 до 255;
- **Integer** – целое от -32 768 до 32 767;
- **Long** – целое от –2 147 483 648 до 2 147 483 647;
- **Single** – действительное число, примерно 8 значащих цифр;
- **Double** – действительное число, примерно 15 значащих цифр;
- **Currency** – для работы с денежными числовыми величинами от -922 337 203 685 477.5808 до 922 337 203 685 477.5807;
- **Date –** дата;
- **String** – строка символов;
- **Boolean** – логические величины;

• **Object** – ссылка на объект, этот тип будет рассмотрен в дальнейшем;

• **Variant** – **для** работы с любыми данными.

Объявление переменных производится оператором **Dim**, перед наименованием типа ставится ключевое слово As, например

```
Dim Date1 As Date, i As Byte
```

Переменные, определяемые внутри подпрограмм обработки с помощью оператора Dim, èëè ÿâíî íå îïðåäåëÿåìûå, являются локальными, их значения можно использовать только в процедуре, в которой они определены. Для создания глобальных переменных следует использовать секцию описания глобальных переменных окна программного кода, которая отделяется от текста первой процедуры горизонтальной чертой. Выбрав в левом поле со списком окна кода строку General, вы сможете перейти непосредственно к вводу описания глобальных переменных.

Операции

VB обладает стандартным набором следующих арифметических операций: умножить; разделить; сложить; вычесть. Кроме того, имеются операции: возведения в степень “ ^ ”; целочисленного деления “ \ ” – обратная косая черта; вычисления остатка от деления двух целых чисел “Mod”. Пример: значение выражения 17 Mod 3 равно 2.

Для слияния двух строк в одну используется операция конкатенации “&“. При использовании этой операции для работы с нестроковыми переменными происходит предварительное преобразование их к строковому виду.

Логические выражения

Логическое выражение в VB возвращает результат типа Bollean, может принимать два значения: True и False. Имеются следующие операции сравнения, результатом которых также является значение типа Bollean:

< , > , <= , >= , = , <> (неравно).

Операндами операций сравнения служат числовые или строчные константы и выражения.

Логические операции работают с операндами логического типа. Имеется унарная логическая операция

• Not – Не – возвращает противоположное операнду значение.

Пять бинарных логических операций:

• And – И;

• Or – Или;

• Xor – равно True, если значения операндов различны, и равно False, если они равны;

• Eqv – равно False, если значения операндов различны, и равно True, если они равны;

• Imp – равно False, если первый операнд равен True, а второй – False.

Пример: значение выражения (5>3) And (5<3) равно False.

Структура IF

Структура If может иметь несколько видов. Заметим, что для описания этой структуры используются кроме ключевого слова IF также ключевые слова Then, Else, ElseIf, End If в различных сочетаниях. Приведем наиболее часто встречающиеся варианты использования этого типа ветвления.

```
If условие Then оператор1 Else оператор2
```

Пример: If I > 10 Then S1 = 10 Else S1 = I

```
If условие Then оператор1
```

Пример: If I > 10 Then S1 = 10

```
If условие Then
     Блок_операторов1
Else
    Блок_операторов2
End If
If условие1 Then
    Блок_операторов1, в случае истинности условия1
ElseIf условие2 Then
    Блок_операторов2, в случае истинности условия2
ElseIf условие3 Then
    Блок_операторов3, в случае истинности условия3
Else
    Блок_операторов4, в случае ложности всех условий
End If
```

Структура Select Case

Подобно структуре If данная структура позволяет выполнить ту или иную последовательность операторов в зависимости от истинности ряда условий. Для проверки значения управляющего ***контрольного выражения*** используются специальные ***Case-выражения***. После каждого Case - выражения находятся операторы, выполняющиеся в том и только в том случае, когда соответствующее Case – выражение истинно.. Общий вид структуры:

```
Select Case контрольное выражение
Case Case-выражение1    Операторы1
Case Case-выражение2
    Операторы2
…
Case Else
    Операторы- Else
End Select
```

Case-выражения могут быть трех видов:

- список значений, которые должно принимать контрольное выражение, через запятую;
- диапазон значений, задаваемых при помощи ключевого слова To;

• сравнение, в котором вместо контрольного выражения используется ключевое слово Is.

Пример использования оператора оператора Case:

Пусть I числовая переменная.

```
Select Case I
Case 0
    Label1="ноль"
Case 1
    Label1="один"
Case Else
    Label1="больше одного"
End Select
```

Êðîìå òîãî, Case-выражение может быть комбинацией всех указанных трех видов выражений, при этом выражения перечисляются через запятую.

Пример сложного Case-выражения:

Case 0,1, 3 To 5, Is> 10.

Это выражение будет истинно, если контрольное выражение принимает значения 0,1,3,4,5,10,11 и т. д.

Цикл For

```
For счетчик = выр1 To выр2 Step приращение
     Операторы цикла
Next счетчик
```

Здесь: счетчик – переменная, выр1, выр2 – начальное и конечное значения переменной, приращение – величина, на которую изменяется счетчик после выполнения каждой итерации.

В отличие от цикла For, во многих других языках программирования цикл For в VB – достаточно гибкая конструкция:

- в качестве счетчика можно использовать переменные любого числового типа, а не только целые;
- приращение может быть отрицательным.

Цикл Do While с предусловием

```
Do While условие
     Операторы цикла
Loop
```

Этот цикл выполняется до тех пор, пока (While) условие истинно. Истинность условия проверяется до выполнения операторов цикла.

Цикл Do Until с предусловием

```
Do Until условие
     Операторы цикла
Loop
```

Слово Until переводится с английского “до тех пор пока не”, в соответствии с этим цикл Do Until выполняется, пока условие ложно.

Циклы Do с постусловием

Условие проверяется после выполнения операторов цикла, они выполняются, по крайней мере, один раз.

```
Do
     Операторы цикла
Loop {While|Until} условие
```

Цикл Do без условия

Цикл выполняется неограниченное число раз, пока не выполнится в теле цикла оператор **Exit Do** (выход из цикла), который, например, может быть частью конструкции If…Then…Else.

Процедуры

В VBA замкнутыми программными единицами являются процедуры. Процедура содержит набор операторов, с помощью которых выполняются действия или рассчитывается значение. Примерами процедур могут служить рассматриваемые выше процедуры обработки событий.

Существуют процедуры двух типов:

Процедура-подпрограмма Sub, которая аналогично операторам VB выполняет действие или набор действий, но не возвращает значение. Примером простейшей процедуры может служить процедура суммирования двух чисел:

```
Private Sub Summa()
Label1.Caption = Val(Text1.Text) + Val(Text2.Text)
End Sub
```

Процедура-функция **Function** (часто такие процедуры называют просто функциями) возвращает значение, например полученное в результате расчетов. VB включает ряд встроенных функций; например функция **Now** возвращает текущее значение даты и времени. В дополнение к встроенным функциям пользователь имеет возможность самостоятельно создавать функции, которые называют специальными функциями.

Функции, возвращающие значения, могут использоваться в выражениях.

Ниже приводится пример процедуры Function с именем Lmax, которая возвращает максимальное расстояние между болтами в зависимости от давления (см. пример программы – расчет крышки цилиндра)

```
Private Function Lmax(Pp As Integer)
Select Case Pp
Case Is <= 15
    Lmax = 150
Case Is <= 25
    Lmax = 120
Case Is <= 50
    Lmax = 100
Case Is <= 100
```

```
        Lmax = 80
    Case Else
        Lmax = 0
    End Select
    End Function
```

Существенным отличием `Function от Sub` является обязательное присваивание в теле процедуры значения имени процедуры, последнее присвоенное имени значение является значением функции, передаваемым в вызывающую процедуру. Необязательное ключевое слово `As` определяет тип возвращаемого значения по умолчанию `Variant`**.**

```
Private Function Lmax(Pp As Integer) As Single
```

Процедуры `Sub` и `Function` могут принимать аргументы. Как правило, количество аргументов фиксировано, но существует возможность использовать переменное число аргументов. Для этого служит ключевое слово Optional, помещаемое перед первым необязательным аргументом. Переменное число аргументов можно передавать также через массив аргументов (ParamArray).

Глава 5. РАБОТА С ГРАФИЧЕСКИМИ ПРИМИТИВАМИ

Предварительные сведения

5Массивы

Массивы в VB определяются оператором DIM. Массив может иметь размерность до 60. Интервал изменения каждого индекса может задаваться явно, например (1 to 10) – индекс принимает значения от 1 до 10. Если нижняя граница изменения индекса не указана явно, то ее значение определяется значением, указанным в операторе **Option Base**, если же и этого оператора нет в программе, то нижняя граница принимается равной 0.

Формат оператора Option Base следующий:

```
Option Base { 0|1}
```

0 указывает, что нижняя граница равна нулю, 1 – что равна единице.

Пример использования **Option Base** и описания массива:

```
Option Base 1
Dim AR (0 TO 10,20,7 TO 10)
```

Описан трехмерный массив, нижняя граница изменения второго индекса равна 1.

6Цвет. Функция RGB

Цвет в VB описывается числовым значением типа Long. Для задания цвета объекта в VB используются несколько способов:

• функция QBColor используется для совместимости со старыми версиями языка Basic (Qbasic), например QBColor(3);

• непосредственное задание числа с типом Long, например &HFF&

• задание с помощью тройки чисел – цветовых компонент и функции RGB.

RGB(red, green, blue)

Каждое целое может принимать значение от 0 до 255, например красный цвет задается **RGB(255,0,0).**

Свойства и методы группы Scale

Эта группа позволяет работать с координатами на объекте. Объектом может являться форма или графическое поле. Свойства позволяют задавать координатную систему на объекте, получать данные о размере объекта в различной системе координат.

Метод Scale позволяет установить координатную систему.

```
[Объект.]Scale (x1,y1)-(x2,y2)
```

Если имя объекта не указано, предполагается, что работа производится с активной формой. (x1, y1) - (x2, y2) – координаты верхнего левого и нижнего правого углов объекта. Внимание! Это не координаты на экране, а внутренние координаты объекта, предназначенные для адресации точек внутри объекта. Если метод Scale не используется, координаты внутри объекта выражаются в единицах твип (см. ниже).

Свойство ScaleMode позволяет устанавливать или получать значение, определяющее меру, в которой измеряются координаты на объекте

```
[Объект.]ScaleMode [= value]
```

Значения, которые может принимать свойство, следующие:

0 показывает, что установлена координатная система пользователя (использован метод Scale);

1 – (по умолчанию) единица измерения – твип (Twip, 1440 единиц на логический дюйм);

2 – точка (Point, 72 на логический дюйм);

3 – пиксель (Pixel) – наименьшая единица, выводимая не экран;

4 – символ (Character) (горизонтально = 120 твип на знак; вертикально = 240 твип на знак);

5 – Дюйм (Inch);

6 – милиметр (Millimeter);

7 – сантиметр (Centimeter).

Свойства **ScaleHeight è ScaleWidth** позволяют установить или прочитать внутренний размер объекта в установленных ScaleMode единицах.

```
[Объект.].ScaleHeight [= value]
[Объект.].ScaleWidth [= value]
```

Графические методы и соответствующие свойства

В группу графических (graphic) входят методы, выполняющие рисование во время выполнения программы. Свойства этой группы не доступны во время разработки программы.

Метод Line рисует отрезок прямой с заданными координатами точек начала и конца или прямоугольник с заданными координатами левого верхнего и правого нижнего углов.

```
[ Объект.] Line[ Step][ (x1,1)] -[ Step] (x2,y2),[ color] ,[ B][ F]
```

Если указано ключевое слово Step, то первая точка отсчитывается относительно текущей позиции вывода (см. свойства CurrentX, CurrentY), а вторая – относительно первой точки. Параметр color устанавливает цвет линии, можно применять RGB-функцию. Прямоугольник рисуется, если указано “B”, при указании “F” прямоугольник заливается цветом линии.

Метод Circle рисует окружность, эллипс или дугу окружности или эллипса.

```
[ Объект.] Circle[ Step] (x,y),radius,[ color,start,end,aspect]
```

Ïàðàìåòðû start, end используются при рисовании дуг, устанавливают углы начала и конца рисования дуг, задаются в радианах. По умолчанию рисуется полная окружность или эллипс.

Параметр aspect устанавливает степень сжатия эллипса: при значении 1 (по умолчанию) рисуется окружность.

Ìåòîä **PSet** выводит на экран точку с заданными координатами заданного цвета.

```
[ Объект.] PSet [ Step]  (x, y), [ color]
```

Размер точки зависит от значения свойства DrawWidth.

Метод **Cls** очищает объект от нарисованого во время выполнения программы.

```
[ Объект.] Cls
```

На объекты, созданные во время разработки не действует.

Прочие графические методы: **Point** – возвращает цвет указанной точки; **PaintPicture** – выводит на объект содержимое графического файла;

Свойства **CurrentX, CurrentY** устанавливают или возвращают графические координаты текущего графического вывода.

```
[ Объект.] CurrentX [ = x]
[ Объект.] CurrentY [ = y]
```

Метод Print

```
Объект.Print [ списокВыражений]
```

Выражения, заданные аргументом списокВыражений, печатаются в объекте, начиная с позиции, указанной значениями свойств CurrentX и CurrentY.

СписокВыражений содержит числовые или символьные выражения . При использовании нескольких выражений они разделяются пробелом, точкой с запятой (;), или запятой. Пробел или точка с запятой указывают, что следующее выражение следует выводить непосредственно за предыдущим. Кроме того, СписокВыражений может содержать функции:

- Spc(n) – вставляет n пробелов.
- Tab(n) – печать начинается с позиции n печати, каждая позиция печати-14 знаков.

Пример работы с графикой

```
Private Sub Command1_Click()
Pi = 3.1415926
Picture1.Scale (0, -1.1)-(2 * Pi, 1.1) st = 0.01
Picture1.CurrentX = 0
Picture1.CurrentY = Sin(Picture1.CurrentX)
For i = 0 To 10 - st Step st
    Picture1.Line -(i, Sin(i)), &HFF&
Next i
End Sub
```

Глава 6. РАБОТА С БАЗАМИ ДАННЫХ

*Работа с базами данных присуща любой серьезной программе. Действительно, любой расчет базируется на **исходных** данных, в результате получаются **выходные** данные.*

Хранение данных следует производить в базах данных общепринятых структур, а не разрабатывать собственные "структуры файлов". Это целесообразно, по крайней мере, по двум причинам:

- *во-первых для стандартных типов баз данных разработаны стандартные процедуры обработки, например сортировка данных;*
- *во-вторых данные в таких базах будут легко доступны при разработке других программ.*

Предварительные сведения

Терминология, связанная с реляционными базами данных (РБД)

Данные в РБД представлены в виде таблиц (table), каждая таблица имеет произвольное количество строк-записей (record) и фиксированный состав столбцов (column), т.е. таблица состоит из упорядоченных элементарных ячеек (cell) с информацией. Столбец имеет тип (целый, текстовый и т.д.), определяющий, какого рода информацию могут содержать принадлежащие ему ячейки. Информация в таблице обрабатывается построчно, в каждый момент времени в непустой таблице имеется одна и только одна запись, к элементам (употребляется также термин к полям) которой может производиться доступ. Эта запись называется текущей (current record).

Для упорядочивания записей в таблице с целью установления последовательности обработки создаются так называемые индексы. В VB индексы используются так же, как указатели (указатель – это index по-английски) в книгах: чтобы найти данные, осуществляется их поиск в индексе. Можно создать индексы, основанные на одном или нескольких полях таблицы.

Если предполагается, что сортировка или поиск двух и более полей одновременно будет часто выполняться, можно создать составной индекс. Например, если для одного и того же запроса часто устанавливается критерий для полей “Имя” и “Фамилия”, то для этих двух полей имеет смысл создать составной индекс.

При сортировке таблицы по составному индексу сначала осуществляется сортировка по первому полю, определенному для данного индекса. Если в первом поле содержатся записи с повторяющимися значениями, то сортировка осуществляется по второму полю, определенному для данного индекса, и так далее.

1. Оператор With

```
With объект
   Операторы
End With
```

Конструкция With – End With позволяет выполнить последовательность операторов над указанным объектом, не повторяя задание имени объекта. Например, если имеется несколько свойств, которые необходимо изменить для одиночного объекта, то удобнее поместить инструкции присвоения свойств внутрь управляющей структуры With, указав ссылку на объект один раз, вместо того чтобы ссылаться на объект при каждом присвоении его свойств.

```
With Label1
 ·   Height = 2000
 ·   Width = 2000
 ·   Caption = "Объект MyLabel"End With
```

Внимание! Нельзя выполнять переходы внутрь или из блоков `With`. Если не выполнены инструкции `With` или `End With`, возможно возникновение ошибок или непредсказуемое поведение объектов.

2. Visual Data Manager

Программа-`утилита` Visual Data Manager вызывается с помощью команды из меню Add-Ins. Интерфейс программы достаточно прост и требует лишь некоторый пояснений. Меню File рассматриваемой утилиты предоставляет стандартные возможности по работе с файлами. Для открытия существующей БД следует выполнить Open – Database и в появившемся дополнительном меню выбрать тип открываемой базы данных. Выберите Microsoft Access. При создании новой БД (пункт меню New) дополнительно укажите версию 7.0.

После открытия базы данных можно создать новую таблицу, расположив курсор мыши на свободном месте в окне Database Window и нажав правую кнопку мыши.

Окно “Table structure” (структура таблицы) позволяет работать с полями таблицы, в том числе добавлять в таблицу поля. Каждое поле таблицы имеет собственный тип. Эти типы в целом совпадают по наименованиям с типами данных VB, но есть и некоторые отличия. Например, вместо типа String (VB) имеется тип Text, введен тип Memo для текстовых данных произвольной (очень большой) длины.

После ввода всей необходимой информации о полях таблицы следует присвоить ей имя и создать (выполнить команду Create Table).

Для работы с существующей таблицей следует нажать правую кнопку мыши, расположив курсор на ее имени. В появившемся меню следует выбрать одну из команд: Open (открыть, т.е. рассмотреть записи таблицы), Design (модифицировать структуру таблицы), Rename (переименовать), Delete… . Для работы со структурой таблиц (т.е. для изменения состава полей таблицы, для редактирования их параметров) выберите Design, для просмотра и редактирования содержимого – Open.

Элемент интерфейса Data

Данный элемент интерфейса используется для установления связи между программой и базой данных. Data позволяет выводить записи и манипулировать ими, передвигаться от записи к записи. Вы можете выполнять большинство операций по доступу к данным без написания программного кода. Для организации доступа к данным VB создает из физической базы данных новый объект Recordset – набор записей. В простейшем случае, при использовании "родных" баз данных, набор записей содержит все записи соответствующей таблицы. После размещения на экранной форме элемента Data следует установить два обязательных свойства, которые и определяют источник данных:

- Свойство DatabaseName óêàçûâàåò ïóòü ê áàçå äàííûõ è åå èìÿ.
- Свойство RecordSource – имя таблицы базы данных.

Кроме того, связью с физической базой данных управляет свойство Connect**,** определяющее тип базы данных, к которой производится подключение. По умолчанию используются базы данных Microsoft Access, èìåþùèå ðàñøèðåíèå .mdb. Далее мы будем рассматривать, если не будет сказано иное, только этот тип баз данных и не будем упоминать данное свойство.

Упомянем ряд свойств, которые устанавливают режимы работы с Recordset:

- ReadOnly – устанавливает режим "только чтение", åñëè ïðåäïîëà-ãàåòñÿ ìîäèôèöèðîâàòü äàííûå, òî ýòî ñâîéñòâî ñëåäóåò óñòàíîâèòü в False
- EOFAction – обработка ñîáûòèÿ "êîíåö ôàéëà", ïðè íåîáõîäèìîñòè äîáàâëåíèÿ íîâîé çàïèñè при äîñòèæåíèè êîíöà ôàéëà ñëåäóåò óñòàíîâèòü EOFAction=Add New.

• BOFAction – обработка ñîáûòèÿ "íà÷àëî ôàéëà".

Свойства DataBaseName и RecordSource элемента управления Data можно устанавливать динамически, используя для ввода значений различные элементы интерфейса, например TextBox. Однако при этом следует для установления связи с новой базой данных и/или таблицей применять метод Refresh, напомним, что при обращении к методам следует указывать объект, и, через точку название метода, например:

```
Data1.Refresh
```

Использование стандартных элементов интерфейса для отображения информации таблиц

Элементы интерфейса Picture, Label, TextBox, CheckBox, Image, OLE, ListBox è ComboBox могут быть привязаны к полю данных **Recordset,** управляемому элементом **Data.** Для этого следует установить значения свойств.

DataSource = [имя элемента интерфейса DATA]

DateField = [имя поля].

Ïîñëå ñîîòâåòñòâóþùåé óñòàíîâêè ïðè èçìåíåíèè òåêóùåé çàïèñè áóäóò èçìåðÿòüñÿ è çíà÷åíèÿ, îòîáðàæàåìûå â ýòèõ ýëåìåíòàõ èíòåðôåéñà.

Дополнительные элементы интерфейса

Дополнительные стандартные элементы интерфейса полезны как при работе с базами данных, так и во многих других случаях. Получить доступ к ним можно после добавления соответствующих иконок в окно ToolBox. Для этого следует открыть диалоговое окно Components, нажав CTRL+T или, воспользовавшись меню Project – Components, в закладке Controls установить флажок напротив требуемого компонента и нажать кнопку “Ok”. Например, чтобы добавить в TollBox DBGrid, следует отметить “Microsoft Data Bound Grid Control”.

Элемент интерфейса DBGrid

Grid в переводе с английского означает “решетка”. Элемент **DBGrid** дает адекватное интуитивным представлениям о таблицах баз данных их отображение, позволяет отображать несколько записей **Recordset**, причем достаточно гибким образом, а также требуемое количество столбцов и передвигаться по базе данных как “вверх-вниз” по записям, так и “влево - вправо” по полям.

Для привязки к данным используется свойство DataSource**,** в котором следует выбрать из списка одно из имен элементов Data**.**

С целью наиболее полного редактирования записей таблицы следует установить свойства AllowAddNew (позволять добавлять новые записи) и AllowDelete (позволять удалять записи) равными True.

Каждый столбец данных может иметь свое имя, позицию, ширину, вне зависимости от его свойств в физической таблице. Установить заголовок и вид

вывода “решетки с данными”, порядок вывода столбцов, их заголовки и прочие характеристики можно при помощи так называемого “Property page” – окна, которое выводится на экран при обращении к свойству `Custom`. Заметим, что решетка существенно привязывается к источнику данных, устанавливать вид вывода данных следует только после задания `DataSource`, если при работе источник данных изменился (другая таблица или другая структура), следует нажать правую кнопку мыши на элементе **DBGrid** è âûïîëíèòü îïåðàöèþ Retrieve Fields (восстановить поля), выбрав в появляющемся меню соответствующую строку.

Элементы интерфейса DBCombo, DBList

Для использования этих дополнительных элементов следует подключить компонент “Microsoft Data Bound List Control” (см. подключение DBGrid).

Данные элементы позволяют выводить значение установленного поля некоторого набора записей (Recordset) по всем записям. Кроме того, с помощью элементов можно устанавливать непосредственную связь между данными двух наборов записей.

Для вывода списка, сформированного из значений поля в наборе записей, следует установить свойства группы List: `Rowsource` (имя элемента Data) `Listfild` (имя поля).

Для связи двух наборов записей, кроме того, следует установить следующее:

- `DataSource` – имя управляющего элемента Data.
- `RowSource` – имя управляемого элемента Data, поле из которого будет использовано для вывода информации.
- `DataField` – имя поля, значение которого будет передаваться (поле принадлежит элементу набора записей, установленному свойством `DataSource`).
- `BoundColumn` – имя поля из набора записей, установленного свойством `RowSource`. Используется для установления однозначного соответствия между двумя íàáîðàìè çàïèñåé. При изменении текущей записи в íàáîðå, указанному в `DataSource` производится поиск записи в `RowSource`, такой, ÷òî значения полей, установленных в `DataField` è `BoundColumn` совпадают.
- `ListField` – имя поля из набора записей, установленного свойством RowSource, значение которого будет указываться в поле вывода DBCombo.

Приведем пример задания свойств DBCombo. В качестве управляющей таблицы используем **Books**, управляемой – **Authors**, содержащиеся в базе данных Biblio.mdb, поставляемой в качестве примера с системой VB. Обычное место расположения этой базы данных.

Program Files\DevStudio\VB\Biblio.mdb

Разместите на форме два элемента Data, соедините их с таблицами базы данных, разместите на форме элемент DBCombo è óñòàíîâèòå ñëåäóþùèå åãî

ñâîéñòâà (ïðèâîäèì èõ â òîì ïîðÿäêå, â êîòîðîì îíè âûâîäÿòñÿ íà ýêðàí â ðåæèìå "Categorized" окна Properties):

```
DataField = AuthorCode
DataSource = Data2 (Data2 связан с таблицей Books)
BoundColumn =Code
ListField = Name
RowSource = Data1  (Data1 связан с таблицей Authors)
```

Выполняя программу, Вы увидите, что при изменении текущей записи в наборе записей, связанном с таблицей **Books**, автоматически меняется запись в окне вывода элемента DBList.

Глава 7. ОБЪЕКТЫ ДОСТУПА К ДАННЫМ (DAO)

Новейшие программные продукты Microsoft производят доступ к информации из физических баз данных посредством так называемой "реактивной машины доступа к базам данных" (Microsoft Jet database engine), которую будем называть кратко - MJ.

Visual Basic предоставляет два способа доступа к данным: с помощью MJ - доступ при помощи элемента управления Data (Data control) и объектов DBGrid, DBCombo, DBList, рассмотренных ранее, и при помощи использования DAO (data access objects). В то время как первый способ - использование элемента управления - Data дает ограниченные возможности по доступу к существующим базам данных без программирования, DAO-модель представляет собой программный интерфейс полного управления базами данных. Эти два способа не являются взаимоисключающими, фактически во многих ситуациях следует использовать их оба вместе. В дальнейших примерах по использованию DAO мы и будем придерживаться такого "смешения" способов.

Существуют три категории баз данных, к которым VB может получить доступ посредством DAO и MJ:

- *Базы данных Visual Basic, называемые родными (native) базами данных. Эти базы данных используют тот же формат, что и Microsoft Access, и дают максимум гибкости и производительности.*
- *Внешние базы данных - базы данных всех распространенных форматов, включая Btrieve, dBASE III, dBASE IV, Microsoft FoxPro 2.0, 2.5, and Paradox 3.x и 4.0. Кроме того, аналогичными методами Вы можете получить доступ к рабочим листам Microsoft Excel или Lotus 1-2-3.*

И, наконец, базы данных, используемые в системах клиент-сервер. Это базы данных, соответствующие ODBC-стандарту, такие как Microsoft SQL Server.

Общие сведения о DAO

DAO-модель является совокупностью классов объектов, моделируюших структуру РДБ. Для описания операций, которые выполняются над РДБ ис-

пользуются многочисленные методы и свойства. MJ транслирует эти операции над DAO в физические операции над файлами баз данных.

Программирование работы с базами данных состоит в создании DAO-объектов, таких как Database (база данных), TableDef (описание таблицы), Field (поле) и т.д., соответствующих различным частям физической базы данных и в последующем использовании свойств и методов для выполнения операций.

Механизм работы с объектами DAO упрощает программирование и придает большую гибкость программам, так как вы используете одни и те же объекты, методы, свойства, работая с базами данных совершенно различных физических форматов.

Далее в этой части пособия будут рассматриваться только родные (native) базы данных, напомним, что это базы данных Microsoft Access, соответственно синтаксис методов и свойств DAO будет приводиться в упрощенном виде.

Изучение темы начнем с объекта Recordset (в переводе – “набор записей”), используя рассмотренный в предыдущем разделе элемент управления Data. Это элемент управления при задании свойств Databasename и RecordSource. Первым свойством устанавливается база данных, вторым используемая таблица. При корректном задании всех свойств элемент интерфейса Data создает набор записей.

Объекты Recordset

Объект класса Recordset – набор записей – автоматически создается при инициализации формы, если свойства элемента интерфейса Data установлены верно. В дальнейшем, при использовании метода Refresh – обновить – MJ делает попытку создать новый набор записей, используя новые свойства элемента Data. Данная особенность позволяет работать последовательно с несколькими таблицами, использую только один элемент Data..

Объект Recordset может быть нескольких типов, различающихся по возможностям редактирования данных исходных физических баз данных и по используемым методам. Далее будем рассматривать, если не оговорено иное, наборы записей только так называемого динамического типа (dynaset type), кстати, этот тип устанавливается VB по умолчанию. (Свойство RecordsetType элемента Data.) Для обращения к набору записей, представляемых конкретным элементом Data, следует указать имя этого элемента, свойство RecordSet и, используя в качестве связующего элемента точку, имя конкретного метода или свойства.

Следующий пример показывает, как подсчитать количество записей в наборе. Здесь используются метод `MoveLast` – перейти к последней записи – и свойство `RecordCount` – количество прочитанных записей.

```
Data1.Recordset.MoveLast
MsgBox "Records: " & Data1.Recordset.RecordCount
```

Приведем методы для работы с набором записей.

1. Перемещение текущей записи

Перемещение на начало, в конец набора, на одну запись вперед, на одну запись назад, на заданное число записей соответственно

```
MoveFirst, MoveLast, MoveNext, MovePrevious
Move строки
```

2. Ïîèñê çàïèñåé

Поиск соответственно первой записи в наборе, последней, следующей и предыдущей. Условие задается в виде выражения или строки, формат записи условия будет приведен ниже.

```
FindFirst, FindLast, FindNext, FindPrevious
```

Общий для всех формат:

```
FindFirst условие
```

3. Óäàëåíèå òåêóùåé çàïèñè

```
Delete
```

4. Ðåäàêòèðîâàíèå òåêóùåé çàïèñè

Для редактирования текущей записи следует выполнить метод Edit, затем модифицировать информацию, затем сохранить изменения (Update).

```
With Data1.Recordset
 .Edit
 .Author = "Третьяков"
 .Update
End With
```

При модификации информации применяют упрощенное обращение к полям набора данных, используя формат Recordset.имя_поля. Это связано с тем, что объекты Recordset по умолчанию помещаются в семейство Fields (поля базы данных). Используемым по умолчанию свойством объекта Field является свойство Value (значение). Использование значений по умолчанию позволяет упростить программу. При полной записи строка будет выглядеть так:

```
.Fields("Author ").Value = "Третьяков"
```

И, наконец, выполнение той же модификации поля при программировании без использования оператора With будет выглядеть еще сложнее:

```
Data1.Recordset.Fields("Author").Value = "Третьяков"
```

5. Äîáàâëåíèå íîâîé çàïèñè

```
AddNew
```

Использование этого метода подобно использованию метода Edit.

6. Некоторые свойства набора записей

RecordCount – возвращает число записей, к которым был осуществлен доступ в объекте Recordset, для подсчета количества записей предварительно следует использовать Movelast.

AbsolutePosition – задает или возвращает относительный номер текущей записи в объекте Recordset.

Bookmark – устанавливает или возвращает закладку, которая однозначно определяет текущую запись в Recordset.

Sort – задает или возвращает порядок сортировки записей в объекте Recordset

Filter – задает или возвращает значение, определяющее записи, которые будут включены в открываемый объект Recordset.

Последние два свойства требуют пояснений. Задаваемое или возвращаемое значение является выражением типа String, оно может содержать имена полей, числовые и строчные константы, знаки операций. Например, для того, чтобы рассматривать только записи с кодом автора более 500 и фамилией, начинающейся только с буквы Z, следует установить фильтр "Author>'YY'And Au_ID>500". Из примера видно, что строчные константы при задании фильтра следует заключать в одиночные кавычки (апрстрофы).

Для выполнения установленных сортировок и/или фильтров следует выполнить метод OpenRecordSet, подробное описание которого здесь не приводится. Примеры:

Сортировка

```
With Data1.Recordset
     .Sort = " Author"
     Set Data1.Recordset =.OpenRecordset
End With
```

Фильтр

```
Data1.Recordset.Filter = "Au_Id>5"
Set Data1.Recordset = Data1.Recordset.OpenRecordset
```

Легко просмотреть результат сортировки и фильтрации можно, разместив на форме DBGrid и посмотрев на содержимое RecordSet до и после воздействия на него.

Наиболее часто употребляется динамическое задание фильтра и сортировки посредством программного формирования соответствующих строк. Пусть в программе имеется строчная переменная SS. Приведем несколько примеров создания фильтра с ее использованием. Обратите внимание, что знак операции + используется для конкатенации (слияния) строчных переменных.

```
ss = "'XX'"
Data1.Recordset.Filter = "Author>" + ss
```

В следующем примере используется значение, предварительно введенное в текстовое поле с именем Text3. Обратите внимание: значение текстового поля окружается апострофами.

```
ss = "'" + Text3 + "'"
Data1.Recordset.Filter = "Author>" + ss
```

И, наконец, в последнем примере фильтр полностью формируется в строчной переменной.

```
ss = "Author>'" + Text3 + "'"
Data1.Recordset.Filter = ss
```

Программирование с использованием объектов DAO

Для объектов доступа к данным любых типов можно описать объектные переменные. Объектная переменная, так же как, впрочем, и любая другая переменная, перед использованием требует инициализации. Ниже приведен пример описания двух объектных переменных: базы данных и набора записей, а затем и открытия базы данных Biblio и создания набора записей на основе таблицы Authors.

```
Dim Db As Database
Dim Rs As Recordset

Set Db = OpenDatabase("Biblio.mdb")
Set Rs = Db.OpenRecordset("Authors")
```

Как видно из примера, инициализация объектных переменных производится оператором Set. Создать подобный набор записей (т.е. открыть таблицу для использования в программе) можно, используя элемент интерфейса Data с указанием свойств

```
Databasename = Biblio.mdb
Recodrsource = Authors.mdb.
```

7Семейства DAO

Каждый объект доступа к данным имеет соответствующее ему семейство. Семейство содержит все существующие объекты этого типа. Например, семейств Recordsets содержит все открытые объекты Recordset. Каждое семейство содержится в другом объекте более высокого уровня в иерархии. Например, объект Database содержит семейство Recordsets.

Большинство объектов доступа к данным имеют используемые по умолчанию семейства и свойства. Например, для объектов Recordset по умолчанию используется семейство Fields, а для объектов Field по умолчанию – свойство Value. Используемые по умолчанию семейства и свойства позволяют упростить текст программ (см. пример в разделе “Объекты Recordset”).

8Открытие базы данных

Чтобы открыть базу данных, следует открыть существующий объект Database (используя метод Opendatabase) или создать новый объект, выполнив, например, метод Createdatabase.

3. Обработка данных

Объекты доступа к данным обеспечивают широкий набор средств обработки данных. Создание объекта Recordset позволяет выполнять запросы к базе данных и обрабатывать результирующие наборы записей. Метод OpenRecordset принимает в качестве аргумента, указывающего источник данных, имя таблицы, строку SQL или имя объекта QueryDef (сохраненный запрос). Объект Recordset можно также открыть из объекта QueryDef, который в этом случае является источником данных. Результирующий объект Recordset обладает богатым набором свойств и методов, обеспечивающих поиск и изменение данных.

4. Объекты Database

Метод OpenDatabase

Этот метод позволяет открыть базу данных без использования элементов интерфейса Data. Открытие производится посредством оператора Set, используется предварительно описанная объектная переменная типа Database.

Синтаксис:

```
Set базаДанных = OpenDatabase (имяБД, [монопольно],
[толькоЧтение])
```

• базаДанных – объектная переменная, которая будет представлять открываемый объект Database;

• имяБД – выражение или переменная типа String, задающая имя существующего файла базы данных;

• монопольно – выражение или переменная типа Variant, задающая способ доступа к базе данных, True – открытие базы данных для монопольного доступа, False (по умолчанию) – открытие базы данных для общего доступа;

• толькоЧтение – значение типа Variant, True, если база данных открывается только для чтения, и значение False (по умолчанию) при открытии базы данных с доступом для чтения и записи.

Работа с открытой базой данных ведется с помощью объекта Database и его методов и свойств, в том числе использовать метод `OpenRecordset`; использовать метод `CreateTableDef` для создания таблиц; использовать метод `CreateQueryDef` для создания описания постоянного или временного запроса. С помощью свойства `CollatingOrder` указать язык, по правилам которого выполняется сортировка символьных полей.

Метод OpenRecordset

Этот метод применяется к объектам двух различных категорий: первая – объекты Database, вторая – уже существующие объекты Recordset и объекты QueryDef.

В первом случае используется синтаксис:

```
Set наборЗаписей = объект.OpenRecordset (источник, [ тип] , [ параметры] )
```

• источник – выражение или переменная типа String, определяющая источник записей для нового объекта Recordset – имя таблицы в базе данных;

• тип – константа, указывающая тип открываемого объекта Recordset, может принимать значения dbOpenTable – табличный объект Recordset; dbOpenDynaset – объект Recordset типа динамического набора записей;

• параметры – произвольная комбинация констант, задающих характеристики нового объекта Recordset, некоторые из них перечисленны ниже;

• DbAppendOnly – пользователям разрешается только добавление новых записей в объект Recordset и запрещается изменение или удаление существующих записей;

• DbDenyWrite – Запрет другим пользователям изменять или добавлять записи;

• DbReadOnly – Запрет пользователям на внесение изменений в объект Recordset.

Пример:

```
Dim NewRS AS RecordSet
Set NewRS = OldBD.OpenRecordSet("Customers", _
dbOpenDynaset, DbAppendOnly + DbDenyWrite)
```

Во втором случае при модификации существующего набора записей или при открытии набора записей на основании запроса (QueryDef) используется следующий синтаксис:

```
Set наборЗаписей = объект.OpenRecordset ([ тип] , [ параметры] )
```

Метод Close

```
объект.Close
```

Этот метод применяется к объектам Database и Recordset.

При открытии существующего объекта Database база данных автоматически добавляется в семейство Databases. Объект Database, закрытый с помощью метода **Close**, удаляется из семейства Databases, но остается на диске.

При завершении выполнения процедуры, в которой описан объект Database, этот локальный объект Database закрывается вместе со всеми открытыми объектами Recordset. Любые незавершенные изменения при этом теря-

ются. Чтобы избежать потери изменений, пользователь должен в явном виде до выхода из процедуры завершить все операции редактирования, закрыть объекты Recordset и объекты Database, описанные в этой процедуре на локальном уровне.

Создание новой таблицы

Ограничимся примером

```
Sub CreateTableDefX()
 Dim Bd As Database
 Dim Nt As TableDef
 Dim prpLoop As Property
 Set Bd = OpenDatabase("Книги.mdb")
' Создает новый объект TableDef.
   Set Nt = Bd.CreateTableDef("Контакты")
   With Nt
' Создает поля и добавляет их в новый объект TableDef.
' Это необходимо сделать до добавления объекта TableDef
' в семейство TableDefs базы данных "Книги".
   .Fields.Append .CreateField("Имя", dbText)
   .Fields.Append .CreateField("Фамилия", dbText)
   .Fields.Append .CreateField("Телефон", dbText)
   .Fields.Append .CreateField("Примечания", dbMemo)
' Добавляет новый объект TableDef в базу данных "Книги".
   Db.TableDefs.Append tdfNew
   End With
' Удаляет новый объект TableDef
   Bd.TableDefs.Delete "Контакты"
   Bd.Close
End Sub
```

Запросы. Метод CreateQueryDef

Приведем определение запроса, предоставленное в справочной системе:

Запрос – это требование на отбор данных, хранящихся в таблицах, или на выполнение определенных действий с данными. Запрос позволяет создать общий набор записей из данных, находящихся в разных таблицах, и использовать этот набор как источник данных для формы или отчета.

Запросы содержат строку, написанную на языке SQL, которая, собственно, и определяет суть запроса. Язык SQL – специальный язык программирования для баз данных – используется в программировании с 1970 г. Исторически сложившееся произношение этого термина – "сиквел", хотя в последнее время произносят и "эс-кю-эль".

В качестве примера простейшего запроса ниже рассматривается запрос, состоящий в выборке из таблицы Авторы базы данных Книги всех записей, у которых код автора меньше 12.

Метод CreateQueryDef, применяемый к базе данных, создает объект типа "запрос", выполняется же он позднее, при создании соответствующего ему набора записей (Recordset).

Синтаксис CreateQueryDef:

```
Set запрос = объект.CreateQueryDef ([имя], [строкаSQL])
```

• Запрос – объектная переменная, типа QueryDef.

• Объект – объектная переменная, представляющая открытый объект Database.

• Имя – выражение или переменная типа String, содержащая уникальное имя нового объекта QueryDef.

• СтрокаSQL – выражение или переменная типа String, содержащая инструкцию SQL.

Как видно из синтаксиса, обязательными являются только имена двух объектных переменных, переменной типа QueryDef, которая будет соответствовать создаваемому объекту-запросу и переменной типа Database. Параметры "имя" и "строкаSQL" можно при создании запроса не задавать, а задать эти значения позднее, модифицировав свойства `Name` и `SQL` созданного объекта QueryDef.

Имя (Name) обязательно только для сохраняемых запросов, которые (сам запрос, т.е. строка SQL, а не результат запроса), хранятся в базе данных. Если в качестве имени указана пустая строка, то запрос считается временным и не сохраняется в базе данных.

Рассмотрим упомянутый выше простейший пример.

```
Private Sub Command4_Click()
    Dim Base As Database
    Dim QNew As QueryDef
    Dim RTemp As Recordset
    Set Base = OpenDatabase("C:\vs\vb\Книги.mdb")
    With Base
       Set QNew = .CreateQueryDef("", _
       "SELECT * FROM Авторы where Au_Id <12")
        Set RTemp = _
        QNew.OpenRecordset(dbOpenSnapshot)
        Set Data2.Recordset = Rtemp
        RTemp.Close
        .Close
    End With
End Sub
```

Первый оператор Set открывает базу данных. Второй оператор Set создает временный запрос (параметр Name равен пустой строке), строка SQL указывает, что из таблицы Авторы должны быть выбраны те строки, у которых значение AU_ID меньше 12. Подробно синтаксис SQL будет рассмотрен да-

лее. Третий оператор Set открывает набор записей, которые будут выбраны из таблицы согласно запросу Qnew. Для визуализации полученных данных используются включенные в форму элементы интерфейса Data и DBGrid. Четвертый оператор Set "обманывает" DBGrid, замещая набор записей, созданный соответствующим ему Data, новым, созданным с помощью QueryDef набором записей.

Приложение 1

Дополнительные элементы интерфейса

В этом разделе будут рассмотрены несколько дополнительных элементов интерфейса, подробно не описанных ранее, но весьма полезных при программировании.

Заголовок раздела содержит наименование компоненты, загружаемой для доступа к рассматриваемым в разделе элементам интерфейса. Компонента выбирается из списка, вызываемого нажатием Ctrl-T.

1. Microsoft Chart Control

Для построения диаграмм требуется загрузить компоненту Microsoft Chart Control. Ниже приведены два примера работы с этим элементом интерфейса, первый пример иллюстрирует только свойства MSChart, второй – совместную работу построителя диаграмм с базой данной.

Диаграмма обладает следующими свойствами:

- **ChartType** – тип диаграммы. Может задаваться либо константой “ChartType” либо целым числом от 0 до 8.
- **ColumnCount** – количество столбцов данных
- **RowCount**– количество строк данных

В первом примере производится вывод диаграмм различных типов, заполненных случайными числами.

```
Public TypeChart
Private Sub Form_Load()
    TypeChart = -1
End Sub
Private Sub Command1_Click()
    With MSChart1
        TypeChart = TypeChart + 1
        If TypeChart = 9 Then TypeChart = 0
        .chartType = TypeChart
        .ColumnCount = 8
        .RowCount = 8
        For Column = 1 To 8
            For Row = 1 To 8
                .Column = Column
                .Row = Row
```

```
                .Data = Rnd() * 10
            Next Row
        Next Column
    End With
End Sub
```

Во втором примере данные для диаграммы вводятся из таблицы, содержащей столбцы с именами beta, sigma, name. Форма должна содержать соответствующим образом настроеный элемент интерфейса Data.

```
Private Sub Command1_Click()
  Data1.Recordset.MoveLast
    i = Data1.Recordset.RecordCount
    Data1.Recordset.MoveFirst
    MSChart1.rowCount = i
    With MSChart1
        .chartType = 6
        .columnCount = 2
         For row = 1 To i
                .column = 1
                .row = row
                .Data = Data1.Recordset!beta
                .column = 2
                .row = row
                .Data = Data1.Recordset!sigma
                .RowLabel = Data1.Recordset!Name
                Data1.Recordset.MoveNext
         Next row
    End With
End Sub
```

2. Microsoft Windows Common Controls-2 5.0

Содержит два класса элементов интерфейса Animation и UpDown. Мы рассмотрим толькоAnimation, корторый служит для воспроизведения видеороликов в формате AVI.

Метод `Open` открывает файл.

Свойство `AutoPlay` позволяет запустить ролик непосредственно после открытия файла.

Пример использования Animation

```
Private Sub Form_Load()
    With Animation1
        .AutoPlay = True
        .Open "c:\Cool.avi"
    End With
End Sub
```

3. Microsoft Common Dialog Control 5.00

Весьма полезный элемент управления, позволяющий выполнять различные стандарные для Windows процедуры, в том числе:

- открытие и сохранение файлов;
- выбор цвета для визуальных элементов;
- выбор параметров шрифта.

Открытие файла

Выполните метод **ShowOpen.** Вы также можете установить фильтр, позволяющий выводить список файлов только необходимых типов, установив свойство `Filter`. `FilterIndex` дает возможность из всех заданных фильтров выбрать используемый по умолчанию.

Пример:

```
Private Sub Command1_Click()
  CommonDialog1.Filter="All Files (*.*)|*.*|Text Files"&_
  "(*.txt)|*.txt|Batch Files (*.bat)|*.bat"
  CommonDialog1.FilterIndex = 2
  CommonDialog1.ShowOpen
  MsgBox CommonDialog1.filename
End Sub
```

Пример использования CommonDialog1 для открытия файла-видеоролика:

```
Private Sub Animation1_Click ()
 With CommonDialog1
      .Filter = "avi (*.avi)|*.avi"
      .ShowOpen
 End With
 With Animation1
      .Autoplay = True
      .Open CommonDialog1.Filename
 End With
End Sub
```

Выбор цвета

Выполните метод `ShowColor`.

```
Private Sub Command1_Click()
    CommonDialog1.ShowColor
    Form1.BackColor = CommonDialog1.Color
End Sub
```

Установка параметров шрифта

```
Private Sub Command1_Click()
   CommonDialog1.ShowFont
    Text1.Font.Name = CommonDialog1.FontName
    Text1.Font.Size = CommonDialog1.FontSize
```

```
        Text1.Font.Bold = CommonDialog1.FontBold
        Text1.Font.Italic = CommonDialog1.FontItalic
       Text1.Font.Underline = CommonDialog1.FontUnderline
        Text1.ForeColor = CommonDialog1.Color
End Sub
```

Приложение 2

Создание макросов в Microsoft Word

В этой главе Вы познакомитесь с языком Visual Basic for Application - некоторой разновидность. VB, предназначенной для использования совместно с текстовым редактором Word.

Часть примеров, помещенных в главе, опубликованные на www.citforum.ru, в статье Марины Миланиной "WordBasic и макровирусы".

Запись последовательности действий

Проще всего создать макрос с помощью команды Сервис-Макрос-Начать запись. Все действия пользователя до нажатия кнопки "Стоп" записываются в макрос и воспроизводятся при запуске этого макроса. Такой способ не позволяет организовывать циклы и выдавать сообщения пользователю, но и он бывает полезен при автоматизации часто повторяющихся действий. Например, при написании текста, содержащего как русские, так и английские слова, Word "теряет" установку русского языка и некорректно проверяет орфографию.

Запишем макрос, устанавливающий русский язык для всего текста, при записи макросу можно назначить сочетание клавиш на клавиатуре для его быстрого вызова. Последовательность действий:

- Выполним команду Сервис-Макрос-Начать запись.
- В появившемся окне назначим макросу имя -Русский.
- Назначим макрос клавиатуре – укажем клавиши Ctrl+Shift+H.
- Выполним команду Правка-Выделить Все.
- Выполним команду Сервис-Язык-Выбрать Язык-Русский-Ok.
- Нажмем кнопку Стоп.

Записанный макрос можно посмотреть. Для этого необходимо выполнить команду СервисМакрос-Редактор VisualBasic.

Текст макроса приведен ниже.

```
Sub Русский()'
' Русский Макрос
' Макрос записан 15.12.98
Selection.WholeStory
Selection.LanguageID = wdRussian
End Sub
```

Собственно текст макроса уместился в двух строчках. В первой размер выделенного текста определяется как размер всего текста (за исключением некоторых фрагментов), во второй для выделенного текста устанавливается язык =русский язык.

Создание макросов в редакторе VB

Выполним команду СервисМакрос-Редактор VisualBasic - откроем редактор VB. В окно программного кода запишем строки:

```
Sub Hello()
 MsgBox "Hello Word", vbInformation, "Мое первое сообщение"
End Sub
```

Приведенный текст является текстом макроса с именем Hello. Первый параметр функции MsgBox задает текст сообщения, второй - тип сообщения, т.е. значок и кнопки, а третий – заголовок окна сообщения. Для задания клавиши вызова макроса следует выполнить команду Сервис-Настройка-Клавиатура, выбрать категорию макросы и присвоить макроса Hello некоторую комбинацию клавиш.

Усложним программу. Пусть она выводит на экран сообщение с надписью "Закрыть Word?" и кнопками "Ok" и "cancel". Кроме того, пусть программа закрывает Word по нажатии Ok.

```
Sub Hello()
If MsgBox("Закрыть Word?", vbOKCancel,
"Мое первое сообщение") = vbOK Then
    Application.Quit
End If
End Sub
```

Здесь мы используем возвращенное функцией MsgBox значение для того, чтобы определить, на какую кнопку нажал пользователь. Если функция возвратила vbOK, т.е. пользователь выбрал кнопку OK, мы вызываем метод Quit объекта Application (объектом Application является сам Word).

Продолжим усложнение программы. При выходе Word выдает предупреждения, если изменения в файлах не сохранены. Модифицируем программу так, чтобы эти сообщения не появлялись. Для этого установим свойство DisplayAlerts объекта Application, управляющее выводом сообщений на экран в false и укажем параметр wdDoNotSaveChanges (не сохранять изменения) для метода Application.Quit.

```
Sub Hello()
Application.DisplayAlerts = False
If MsgBox("Закрыть Word?", vbOKCancel,
 "Мое первое сообщение") = vbOK Then
Application.Quit wdDoNotSaveChanges
End If
```

```
End Sub
```

Как защититься от макровирусов

Макровирус – это макроопределение, размножающееся подобно обычному компьютерному вирусу, но только в среде файлов Word (и еще Excel). Не следует пренебрегать средствами зашиты от макровирусов, так как с помощью VB можно написать вирус, портящий документы Word, или даже форматирующий жесткий диск. Особенность макровирусов состоит в том, что обычные антивирусные программы их не распознают, по крайней мере, вирусы, написанные любителями.

В Word97 макросы могут содержаться в обычных документах. Для автоматического запуска макроса при том или ином событии макрос должен иметь одно из следующих имен:

AutoExec – запускается при старте Word или загрузке глобального шаблона •

- AutoNew – запускается при создании нового документа
- AutoOpen – запускается при открытии документа •
- AutoClose – запускается при закрытии документа
- AutoExit – запускается при выходе из Word или при закрытии глобального шаблона.

Приведем макрос, препятствующий выполнению автомакросов.

```
Sub Autoexec()
MsgBox "Не допустим размножения вирусов!"
WordBasic.DisableAutoMacros
End Sub
```

Для предотвращения заражения документов макровирусами необходимо хорошо предсталять себе их принцип работы. Создатели Microsoft Office облегчили задачу злоумышленников тем, что ввели возможность подменять команды Word макрокомандами пользователя. Это значит, что если в Вашем документе есть макрос с именем, скажем, FileOpen, он будет исполняться всякий раз при открытии другого документа, причем макрос может выполнять свою основную функцию – открывать файл – и производить побочные действия.

Приложение 3

Список ключевых слов по категориям

В данном перечне содержатся ключевые слова VB как рассмотренные подробно в настоящем пособии, так и не рассмотренные, приведено также краткое описание их применения.

9Перечень ключевых слов, связанных с вводом и выводом *

Открытие или создание файла	– Open
Закрытие файла	– Close, Reset
Управление выводом	– Format, Print, Print #, Spc, Tab, Width #
Копирование файла	– FileCopy

Получение сведений о файле	–	EOF, FileAttr, FileDateTime, FileLen, FreeFile, GetAttr, Loc, LOF, Seek
Работа с файлами	–	Dir, Kill, Lock, Unlock, Name
Чтение файла	–	Get, Input, Input #, Line Input #
Определение размера файла	–	FileLen
Работа с атрибутами файла	–	FileAttr, GetAttr, SetAttr
Изменение положения указателя	–	Seek
Запись в файл	–	Print #, Put, Write #

10Перечень ключевых слов, связанных с датами и временем

Определение текущей даты или времени	–	Date, Now, Time
Вычисления над датами	–	DateAdd, DateDiff, DatePart
Определение даты	–	DateSerial, DateValue
Определение времени	–	TimeSerial, TimeValue
Установка даты или времени	–	Date, Time
Хронометраж процесса	–	Timer

11Перечень ключевых слов-директив компилятора

Определение константы компилятора	–	#Const
Компиляция выделенных блоков программы	–	#If...Then...#Else

12Перечень ключевых слов, связанных с каталогами и файлами

Изменение каталога или папки	–	ChDir
Изменение диска	–	ChDrive
Копирование файла.	–	FileCopy
Создание каталога или папки	–	MkDir
Удаление каталога или папки	–	RmDir
Переименование файла или папки	–	Name
Определение текущего пути	–	CurDir
Определение даты и времени последнего изменения файла	–	FileDateTime
Определение атрибутов файла, каталога или метки	–	GetAttr
Определение размера файла	–	FileLen
Определение имени файла или метки тома	–	Dir
Установка атрибутов файла	–	SetAttr

13Перечень ключевых слов, связанных с массивами

Проверка наличия массива	–	IsArray

Создание массива – Array
Изменение используемой по умолчанию
нижней границы индексов массива – Option Base
Описание и инициализация массива – Dim, Private, Public, ReDim, Static
Определение границ массива – LBound, UBound
Повторная инициализация массива – Erase, ReDim

6. Перечень ключевых слов для выполнения математических вычислений

Тригонометрические вычисления – Atn, Cos, Sin, Tan
Общие вычисления – Exp, Log, Sqr
Генерация случайных чисел – Randomize, Rnd
Вычисление абсолютного значения – Abs
Определение знака выражения – Sgn
Выделение целой части числа – Fix, Int

7. Перечень операций

Арифметические операции – ^, – , *, /, \, Mod, +, &
Операции сравнения – =, <>, <, >, <=, >=, Like, Is
Логические операции – Not, And, Or, Xor, Eqv, Imp

8. Перечень ключевых слов, связанных с ошибками при выполнении

Генерация ошибок выполнения – Clear, Error, Raise
Получение сообщений об ошибках – Error
Получение сведений об ошибках – Err
Перехват ошибок на стадии
выполнения – On Error, Resume

9. Перечень ключевых слов, связанных с переменными и константами

Присвоение значения – Let
Описание переменных и констант – Const, Dim, Private, Public, New, Static
Объявление модуля личным – Option Private Module
Получение сведений о переменной – IsArray, IsDate, IsEmpty, IsError, IsMissing, IsNull, IsNumeric, IsObject, TypeName, VarType
Ссылка на текущий объект – Me
Требование явного объявления

переменных	– Option Explicit
Задание используемого по умолчанию типа данных	– Def тип

10. Преобразование данных

Кода ANSI в строку	– Chr
Регистра букв	– Format, LCase, UCase
В другую систему счисления	– Hex, Oct
Числа в строку	– Format, Str
Одного типа данных в другой	– CBool, CByte, CCur, CDate, CDbl, CDec, CInt, CLng, CSng, CStr, CVar, CVErr, Fix, Int
Строки в код ASCII	– Asc
Строки в число	– Val
Формирование времени из компонент или строки	– TimeSerial, TimeValue
Формирование даты из компонент или строки	– DateSerial, DateValue
Извлечение из даты компонент	– Day, Month, Weekday, Year Hour, Minute, Second

11. Перечень ключевых слов, связанных с обработкой строк

Сравнение двух строк	– StrComp
Преобразование строк	– StrConv
Изменение регистра	– Format, Lcase, UCase
Создание строк, содержащих повторяющиеся символы	– Space, String
Определение длины строки	– Len
Форматирование строки.	– Format
Выравнивание строки	– Lset, Rset
Обработка строк	– InStr, Left, LTrim, Mid, Right, RTrim, Trim
Выбор типа сравнения строк	– Option Compare
Работа с кодами ASCII и ANSI	– Asc, Chr

12. Перечень ключевых слов, связанных с передачей управления в программе

Ветвление	– GoSub...Return, GoTo, On Error, On...GoSub, On...GoTo
Завершение и остановка программы	– DoEvents, End, Exit, Stop
Цикл	– Do...Loop, For...Next, For Each...Next,

	hile...Wend, With
Принятие решения	– Choose, If...Then...Else, Select Case, Switch
Использование процедур	– Call, Function, Property Get, Property Let, Property Set, Sub

13. Прочие ключевые слова

Запуск других программ	– AppActivate, Shell
Передача нажатий клавиш приложению	– SendKeys
Подача звукового сигнала	– Beep
Параметры среды	– Environ
Получение командной строки	– Command
Цвет	– QBColor, RGB