Зеньковский В. А.

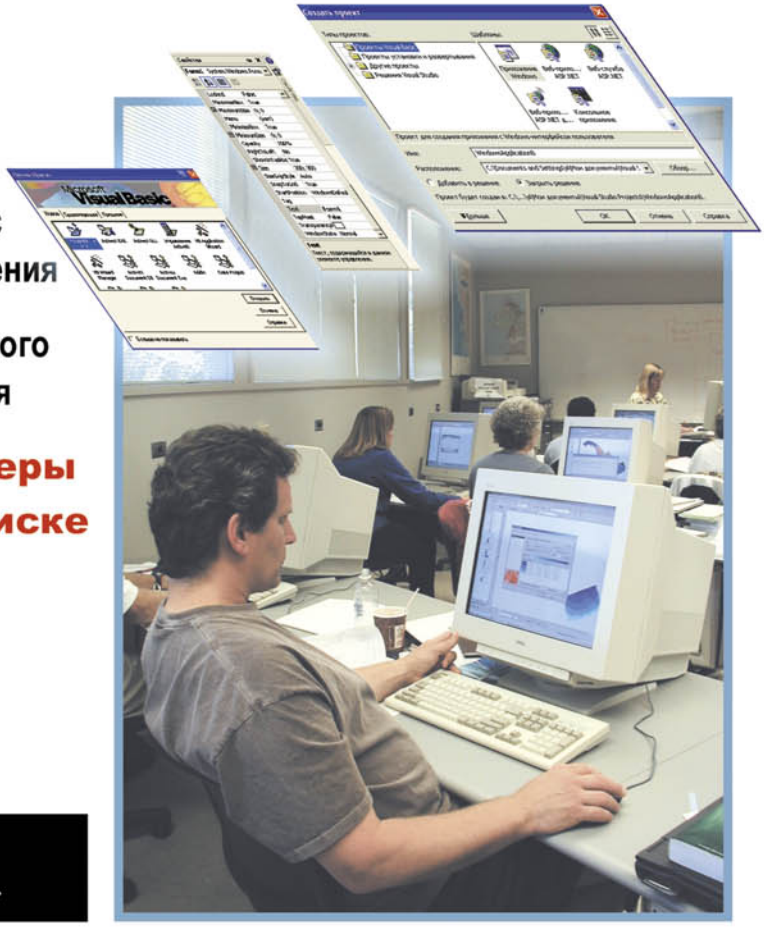
Программирование на Visual Basic 6.5 и Visual Basic.Net

«СОЛОН»

Практический курс ускоренного обучения

Секреты визуального программирования

БОНУС: примеры на компакт-диске



ПроПК

УДК 621.396.218
ББК 32.884.1
356

В. А. Зеньковский

356 Программирование на Visual Basic 6.5 и Visual Basic.Net — М.: СОЛОН-ПРЕСС, 2006. — 248 с.: ил. — (Серия «Про ПК»).

ISBN 5-98003-260-6

На большом количестве оригинальных примеров рассмотрены принципы объектно-ориентированного программирования в средах Visual Basic 6.5 и Visual Basic.Net. Приводимые программы снабжены подробными комментариями с детальными объяснениями используемого алгоритма и синтаксиса языка. Большое внимание уделено программированию графики, в частности, построению фрактальных изображений.

Книга адресована пользователям, имеющим начальный опыт программирования.

К книге прилагается **CD-ROM** с текстами программ.

УДК 621.396.218
ББК 32.884.1

По вопросам приобретения обращаться:
ООО «АЛЬЯНС-КНИГА КТК»
Тел: (495) 258-91-94, 258-91-95
www.abook.ru

Сайт издательства «СОЛОН-ПРЕСС» **www.solon-press.ru**.
E-mail: **solon-avtor@coba.ru**

ISBN 5-98003-260-6

Часть 1
Программирование в среде Visual Basic 6.5

1.1. Введение

В развитии программирования можно выделить следующие исторически сложившиеся этапы.

1. *Программирование в машинных кодах*. Чтобы посмотреть, как выглядит текст программы, записанный в машинных кодах, щелкните по кнопке **Пуск** (находясь в какой-либо из операционных систем семейства Windows) и выберите в **Главном меню** пункт **Выполнить**. В диалоговом окне **Запуск программы** (рис. 1.1) введите имя программы `debug.exe`, нажмите кнопку OK и в появившемся окне (рис. 1.2) введите команду просмотра содержимого ячеек памяти `-d 9876:5432`. Нажмите клавишу ENTER, и вам будет предъявлено со-

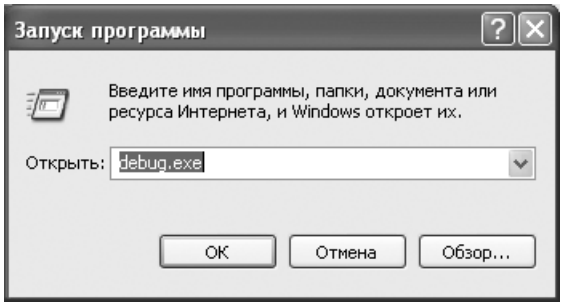


Рис. 1.1

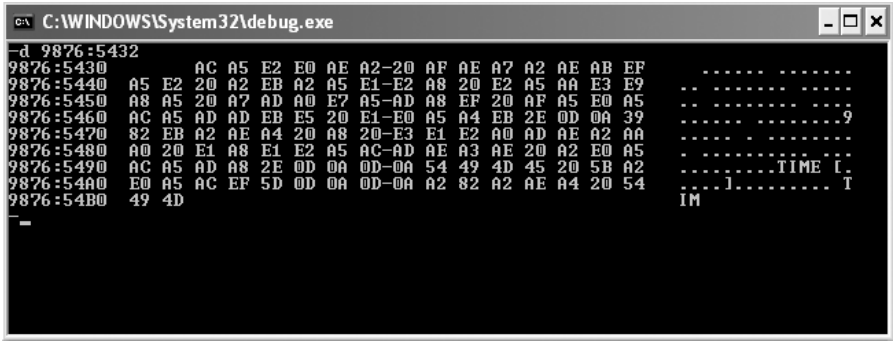


Рис. 1.2

держимое оперативной памяти по указанным адресам, записанное в машинных кодах в шестнадцатеричной системе счисления (рис. 1.2). Так выглядят программы, когда они загружены в компьютер. Для человека восприятие такого представления программ затруднено. В настоящее время машинные коды пишут не программисты, а программы-компиляторы.

2. *Программирование на Ассемблере.* Чтобы посмотреть, как выглядит программа на Ассемблере, проделайте те же действия, что и в предыдущем примере, только в окне программы `debug.exe` введите следующую команду `-u 9876:5432`. Результат представлен на рис. 1.3. Здесь команды процессора записаны не цифрами, а буквами. Например, команда **Переместить** записана как `MOV`. Такая форма записи более близка человеческому восприятию, чем машинные коды. Однако Ассемблер привязан к устройству конкретного процессора, а не к алгоритму программируемой задачи. Машинно-ориентированные языки такого типа не могли получить широкого распространения в среде пользователей вычислительной техники.

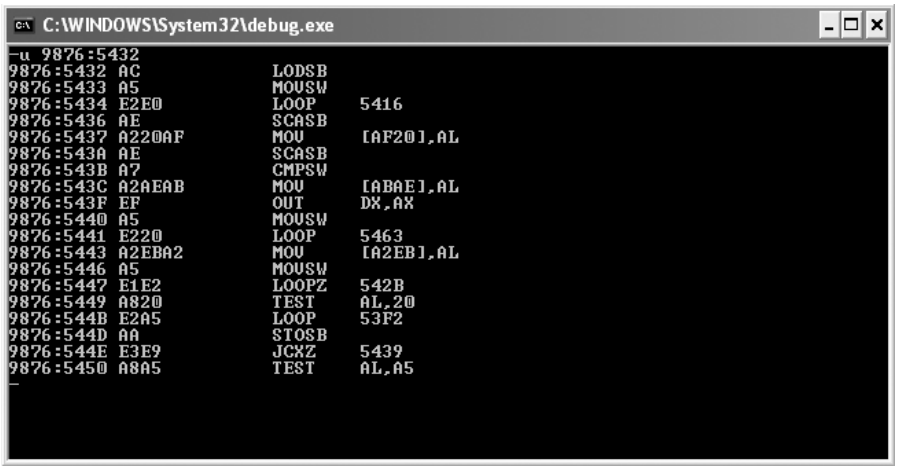


Рис. 1.3

3. *Алгоритмическое программирование.* В 1950-х годах появились языки, для написания программ в которых использовались общеупотребительные слова и простые правила синтаксиса. Перевод таких программ на машинные коды производился специальной программой-компилятором, поэтому программиста уже не волновало, в каких ячейках хранятся результаты расчета и какие регистры процессора используются для операций. Программист был сосредоточен на записи разработанного им алгоритма на алгоритмическом языке программирования. Таков, например, язык ФОРТРАН (Fortran — переводчик формул), разработанный для инженерных расчетов, или КОБОЛ для экономических расчетов. Во главу угла было поставлено понятие алгоритма, который, как известно, является последовательностью однозначно определенных действий, приводящих за конечное число шагов к результату.

4. *Процедурное программирование.* Было замечено, что в программах встречаются одинаковые группы операторов, отличающиеся только значениями входящих в них параметров. Повторяющиеся блоки операторов стали стандар-

тизировать, то есть выделять из общей программы в отдельные подпрограммы, процедуры и функции. В результате возникли такие процедурные языки программирования, как Паскаль, С. Это — универсальные языки. Они не ориентированы на решение конкретного типа задач, т. е. не были проблемно-ориентированными, как, например, Фортран.

5. *Объектно-ориентированное программирование.* В середине 1980-х годов был введен принцип повторного использования кода ранее написанных программ. Такие готовые программные блоки назвали **объектами**. При создании новой программы объекты просто перенастраиваются в соответствии с требованиями решаемой программистом задачи. Таковы встречающиеся в разных программах одинаковые по форме окна, командные кнопки, списки, похожие меню, одинаковые шрифты и т. д. Все они являются настраиваемыми объектами. Примерами объектно-ориентированных языков являются, например, языки C++ и Object Pascal.

6. *Визуальное программирование.* С разработкой в середине 1990-х годов операционной системы Windows компьютеры приобрели возможность графического управления. На экране монитора размещаются уже не просто картинки, а графические **элементы управления**, реагирующие на определенные действия, в частности, на действия со стороны пользователя (на события). Щелчком мыши по графическому элементу можно запустить на выполнение целую программу, не пользуясь при этом клавиатурой. Появление графических (визуальных) систем управления сделало программирование также визуальным. Теперь можно с помощью мыши выбирать из библиотек компоненты, размещать их в рабочем окне будущей программы (делая таким образом из компонентов объекты), и настраивать объекты с помощью свойств, добавляя процедуры для обработки событий (создавая таким образом из объектов графические элементы управления). На рис. 1.4 приведена схема «превращения» компонента через объект в элемент управления. Можно сказать, что компонент сродни чертежу, по которому изготавливают множество совершенно одинаковых объектов, например, одинаковых квартир. Люди, поселившись в квартире, обустраивают ее в соответствии со своими потребностями и вкусом, делают квартиру уникальным элементом своего образа жизни.

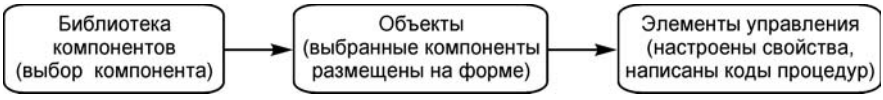
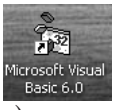


Рис. 1.4

1.2. Интерфейс среды программирования

После запуска системы Visual Basic (щелчком по соответствующей пик-

тограмме 

или через Главное меню Пуск=>Программы=>Microsoft Visual Basic) на экране появится диалоговое окно Новый проект (рис. 1.5).

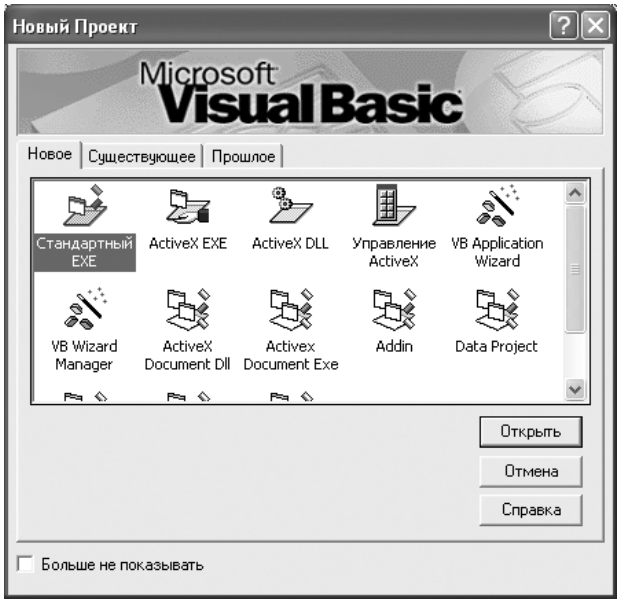

Новый Проект
Microsoft
Visual Basic
Новое
Существующее
Прошлое
Стандартный EXE
ActiveX EXE
ActiveX DLL
Управление ActiveX
VB Application Wizard
VB Wizard Manager
ActiveX Document Dll
Activex Document Exe
Addin
Data Project
Открыть
Отмена
Справка
Больше не показывать

Рис. 1.5

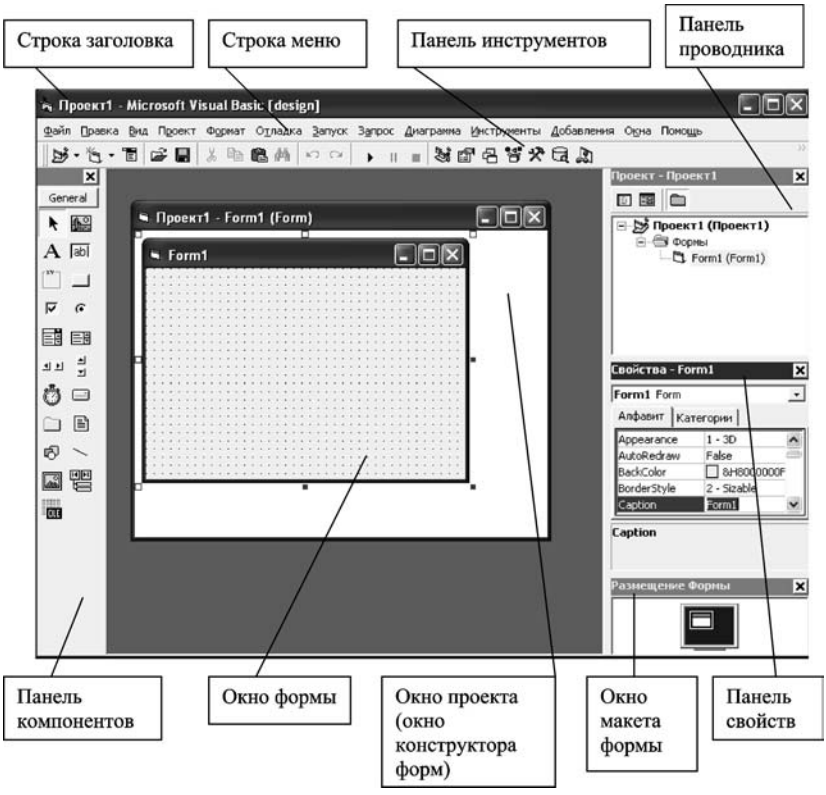

Строка заголовка
Строка меню
Панель инструментов
Панель проводника
Проект1 - Microsoft Visual Basic [design]
Файл Правка Вид Проект Формат Отладка Запуск Запрос Диаграмма Инструменты Добавления Окна Помощь
General
Проект1 - Form1 (Form)
Form1
Проект - Проект1
Проект1 (Проект1)
Формы
Form1 (Form1)
Свойства - Form1
Form1 Form
Алфавит
Категории
Appearance 1 - 3D
AutoRedraw False
BackColor &H8000000F
BorderStyle 2 - Sizable
Caption Form1
Caption
Размещение Формы
Панель компонентов
Окно формы
Окно проекта (окно конструктора форм)
Окно макета формы
Панель свойств

Рис. 1.6

Перейдите на вкладку Новое, выделите значок Стандартный EXE и щелкните на кнопке Открыть. Будет создан новый проект с именем Проект 1, а на экране появится основное окно системы (рис. 1.6).

Для создания интерфейса проекта служит **конструктор форм**. При открытии нового проекта в окно конструктора форм (в окно проекта) добавляется окно **формы**, представляющее собой заготовку стандартного окна Windows. Окно формы характеризуется наличием строки заголовка, в котором указывается имя формы (в данном случае Form1). Имеются также кнопки управления, системное меню, предусмотрена возможность управления с помощью мыши (рис. 1.6). Программист на этапе визуального проектирования помещает на форму компоненты, некоторые из которых представлены на **панели компонентов** (рис. 1.6). Компоненты являются заготовками будущих элементов управления.

В правой части системного окна находится **панель свойств**, содержащая список свойств и их значений для любого выделенного объекта. Выбрать объект можно на самой панели свойств в списке под заголовком (рис. 1.6).

На панели проводника отображаются все составные части проекта, а также имеются кнопки для переключения между окном редактора форм и окном редактора кода программ (рис. 1.7).

Рис. 1.7

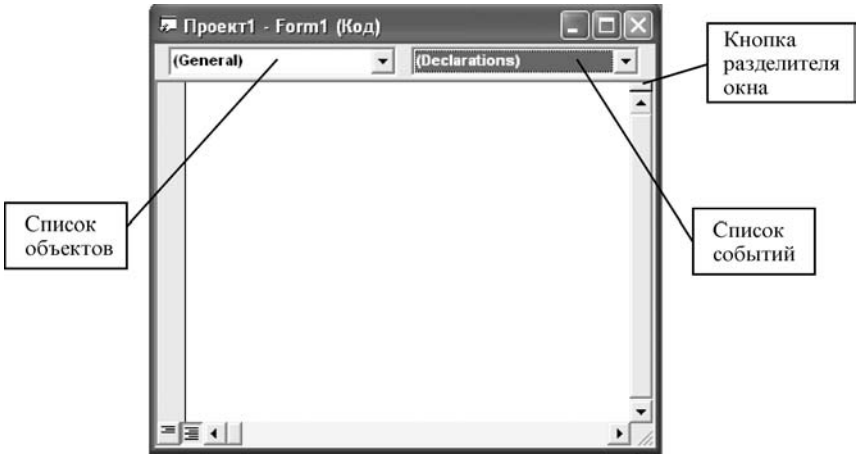


Рис. 1.8

Окно редактора кода программ открывается также двойным щелчком левой кнопкой мыши по форме. Редактор служит для записи кода программы, составленной по модульному принципу в виде отдельных процедур. Для создания процедуры надо выбрать объект и событие, на которое реагирует объект. Этот выбор производится с помощью левого и правого списков окна (рис. 1.8).

Кнопка разделителя окна позволяет разделить окно по вертикали на две части со своими полосами вертикальной прокрутки. Это позволяет одновременно видеть две части длинной программы, не умещающейся целиком в окне.

Окно макета формы демонстрирует, как будет выглядеть форма на экране в режиме выполнения проекта (рис. 1.6).

Для настройки окон служит пункт Главного меню **Вид**.

1.3. Визуальное программирование

Окно формы является главным элементом создаваемого приложения. Сама форма служит контейнером для элементов управления. У формы, как у любого объекта, есть **свойства**. Помимо свойств объектам в Visual Basic приписываются **события** и **методы**. События определяются действиями пользователя (например, одинарный или двойной щелчок левой кнопкой мыши на объекте), самими объектами (например, таймер сам генерирует событие) или операционной системой. События запускают на выполнение код программы, записанной в соответствующей процедуре. **Методы** это команды, которые может выполнить объект (например, метод Move, вызывающий перемещение объекта по форме).

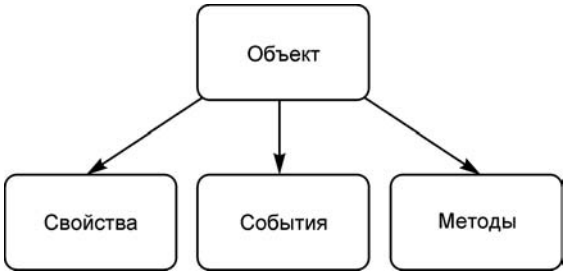


Рис. 1.9

Многие свойства оказываются общими для различных объектов. Рассмотрим некоторые из них на примере формы. Выделите форму и просмотрите перечень ее свойств на панели свойств (рис. 1.10).

Свойство `Name` позволяет изменять имя формы, под которым она фигурирует в коде программы. Это очень важное свойство. Если имя в коде программы будет указано неправильно, объект не будет найден. Поэтому при написании кода во избежание ошибок, которые могут возникнуть при наборе с клавиатуры (например, символ С выглядит одинаково в кириллице и в латинском алфавите), имена объектов лучше копировать из панели свойств и вставлять в код программы. Для формы ее имя указывается в заголовке окна проекта (рис. 1.11).

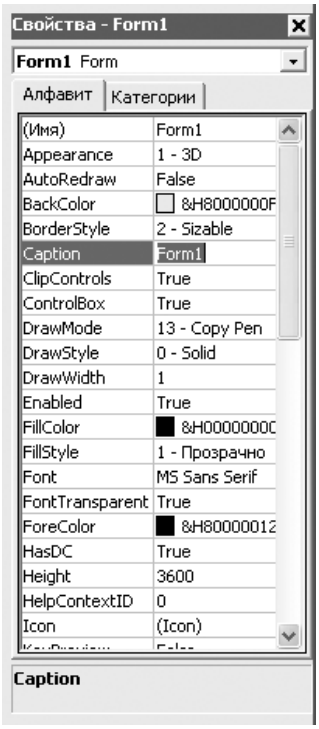


Рис. 1.10

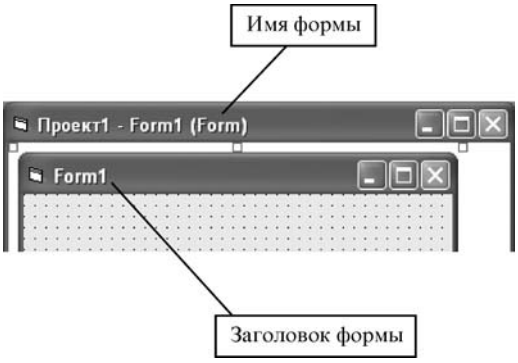


Рис. 1.11

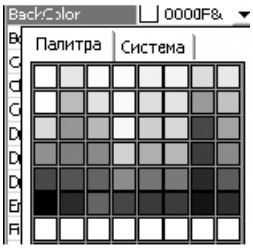


Рис. 1.12

Свойство `Caption` позволяет изменять заголовок формы. Для других объектов это просто надпись на их поверхности. Она несет декоративно-поясняющую функцию и не влияет на работу кода. Измените заголовок формы: в списке панели свойств для формы выделите двойным щелчком свойство Caption и введите вместо Form1 свое имя.

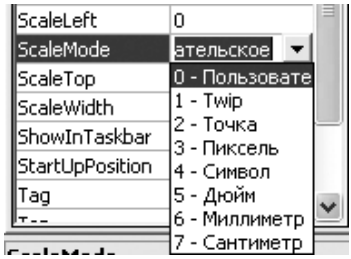


Рис. 1.13

Свойство `BackColor` позволяет изменить цвет рабочего поля формы. Выделите свойство BackColor и выберите в палитре свой любимый цвет (рис. 1.12).

Значения свойств `Left` и `Top` задают координаты левого верхнего угла формы на экране при запуске программы на выполнение. Размерность координат можно выбрать с помощью свойства `ScaleMode` из списка (рис. 1.13). Twip (твип) — единица измерения, введенная программистами Microsoft и равная 1/1440 логического дюйма (величина логического дюйма определяется разрешением экрана). С учетом того, что отсчет координат производится от левого верхнего угла экрана монитора (рис. 1.14), введите (в панели свойств) для свойств Left и Top значения **0**.

Рис. 1.14

Запустите проект на выполнение, нажав клавишу **F5** или щелкнув на кнопке **Запуск** на Панели инструментов ▶, и убедитесь, что форма будет расположена в левом верхнем углу экрана.

Попытайтесь перетащить форму за заголовок. Если свойство формы – `Movable` имело значение `True`, то передвинуть форму удастся. Для выхода из режима выполнения нажмите одновременно две клавиши **ALT** и **F4** или щелкните по кнопке **Остановка** на панели инструментов ■. Установите значение свойства Movable равным False, запустите проект на выполнение и попытайтесь передвинуть форму. Это вам не удастся. Форма останется на том месте, координаты которого были заданы в свойствах Left и Top. Положение формы можно также задать в окне макета формы, переместив мышью изображение формы на экране нарисованного монитора в требуемую позицию (рис. 1.15).

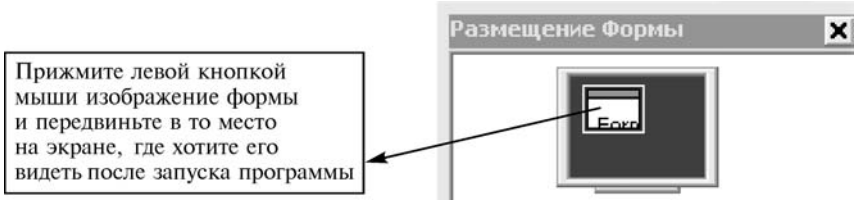


Рис. 1.15

Свойства `Height` и `Width` можно использовать для задания и отображения текущих значений соответственно высоты и ширины формы. Эти свойства можно задавать также мышью, протягивая левой кнопкой мыши прямоугольные маркеры по сторонам формы. При наведении указателя курсора на маркер указатель принимает вид двунаправленной стрелки.

Свойство `BorderStyle` задает различные варианты границы формы в режиме выполнения (рис. 1.16).

Рис. 1.16

Значение 0 — Нет убирает с формы строку заголовка и задает форме фиксированные размеры и положение.

Значение 1 — Fixed Single задает форме фиксированный размер.

Значение 2 — Sizable сохраняет форме все возможности изменения размера и перемещения.

Значение 3 — Fixed Dialog делает форму модальным диалогом, блокируя доступ к другим окнам программы и исключая возможность изменения размеров формы.

Значение 4 — Fixed ToolWindow оставляет на форме только одну кнопку Закрыть и задает форме фиксированный размер, сохраняя возможность перемещения.

Значение 5 — Sizable ToolWindow оставляет форме одну кнопку Закрыть и сохраняет возможность перемещения и изменения размеров. Проверьте работу каждого из перечисленных значений свойства BorderStyle.

Свойство `Picture` позволяет размещать на форме рисунок в качестве фона (подложки). Активизируйте свойство `Picture` и щелкните на кнопке

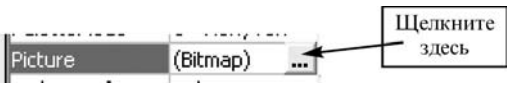


Рис. 1.17

вызова диалогового окна (рис. 1.17). Если вы работаете в Windows XP, зайдите, например, в папку **Общие документы\Рисунки (общие)\Образцы рисунков** и выберите одну из фотографий. Она будет размещена на форме. Остальные свойства формы будут рассмотрены по ходу изложения.

Чтобы наполнить форму конкретным содержанием, на ней размещают управляющие элементы: надписи, кнопки, списки и т. д. Управляющие элементы создаются из заготовок, называемых компонентами. Часть компонентов размещена на панели компонентов (рис. 1.6). Поместить компонент на форму можно двумя способами:

1. Дважды щелкнуть по изображению компонента на панели компонентов: компонент автоматически появится в центре формы. Далее его можно передвигать по форме и менять размеры.

2. Щелкнуть один раз по изображению компонента на панели компонентов и, нажав левую кнопку мыши, нарисовать его на форме.

После размещения на форме компонент становится объектом (экземпляром соответствующего класса), остается активным (выделенным) и приобретает собственный набор настраиваемых свойств, методов и событий. Чтобы удалить активный объект с формы, нажмите клавишу Del. К активному объекту применимы операции с буфером обмена.

Рассмотрим свойства некоторых компонентов и проиллюстрируем с их помощью возможности визуального программирования.

1.4. Компоненты Label (надпись), Shape (фигура), Line (линия)

Компонент `Label` (ярлык, этикетка, бирка, пометка) позволяет получить элемент управления `Label`. На панели компонентов он обозначен как  и служит для вывода информации, например текста, на свою поверхность.

Выберите на панели компонентов компонент `Label` (щелкните по нему левой кнопкой мыши) и разместите его на форме. В результате вы получите объект, имеющий порядковый номер `Label1`. Рассмотрим его основные свойства.

Свойство `Name` задает имя объекта, под которым он фигурирует в кодах программы.

Свойство `Caption` задает текст надписи.

Свойства `BackColor` и `ForeColor` задают соответственно цвет фона и цвет надписи.

Свойство `BackStyle` задает прозрачность заливки (рис. 1.18). При значении `Прозрачно` цвет фона не устанавливается.

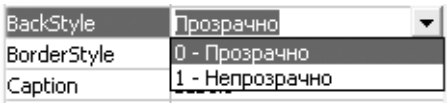


Рис. 1.18

Свойство `Font` позволяет установить атрибуты используемого шрифта: размер, начертание и т. д.

Компонент `Shape` (геометрическая фигура). На панели компонентов обозначен как  . Объект не имеет методов и воспринимается системой как декоративный элемент. Рассмотрим основные свойства компонента.

Свойства `BackColor` и `BackStyle` задают цвет и прозрачность фигуры. Если `BackStyle` имеет значение Прозрачно, то цвет `BackColor` не устанавливается.

Свойства `FillColor` и `FillStyle` задают цвет и стиль штриховки фигуры (рис. 1.19).

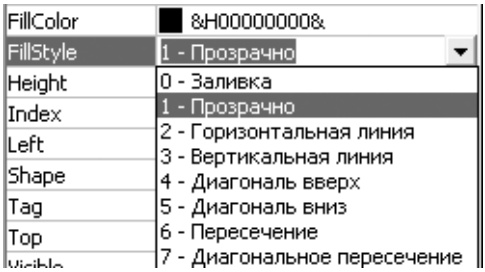


Рис. 1.19

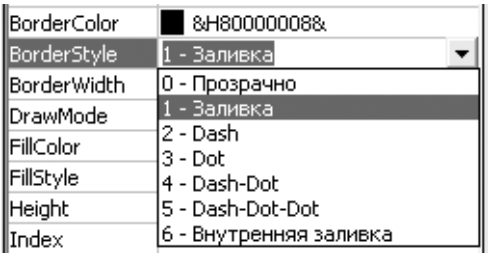


Рис. 1.20

Свойства `BorderColor` и `BorderStyle` задают цвет и тип линии границы (рис. 1.20), (Dash — пунктир, Dot — точка).

Свойство `BorderWidth` задает ширину линии границы.

Свойство `Shape` задает тип геометрической фигуры (рис. 1.21).

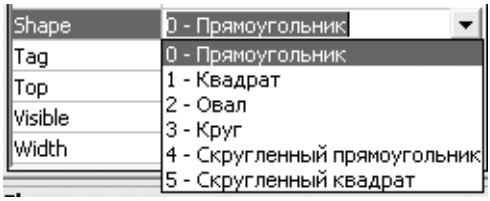


Рис. 1.21

Компонент `Line` (линия)  .

Свойства `BorderColor`, `BorderStyle` и `BorderWidth` определяют соответственно цвет, стиль и толщину линии.

Свойства `X1`, `X2`, `Y1`, `Y2` задают и отображают численные значения координат начальной и конечной точек линии.

Пример 1-1

Задача. Создать программу, выводящую на экран следующую фигуру (рис. 1.22). Для создания фигуры используйте объекты Shape, Line, Label. Покрасьте шляпу фигуры в желтый цвет, а рубаху — в красный. Используйте штриховку. Создайте ярлык программы и разместите его на Рабочем столе.

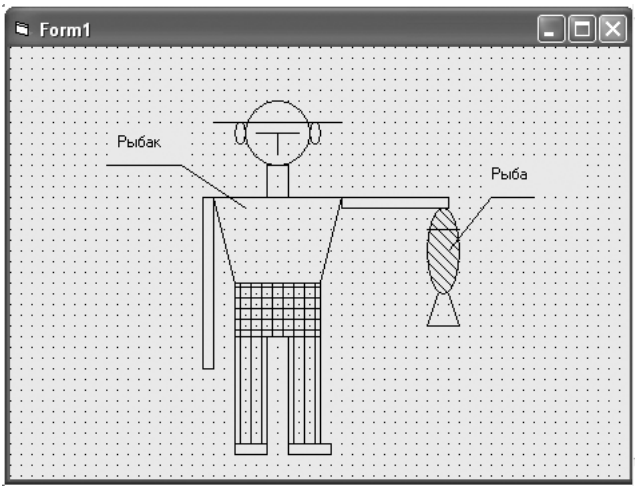


Рис. 1.22

Решение. Завершив построение фигуры, войдите в главное меню **Файл=>Сохранить проект как....** В диалоговом окне введите название Рыбак и щелкните по кнопке Сохранить. Снова откройте главное меню **Файл=>Создать** `Рыбак.exe` (рис. 1.23). Будет предъявлено диалоговое окно **Создать проект** (рис. 1.24). Введите имя Рыбак и щелкните на кнопке OK. Будет создан исполняемый файл **Рыбак.exe**. Еще раз откройте диалоговое окно **Создать проект**, щелкните правой кнопкой на значке **Рыбак**. В появившемся контекстном меню выберите Рабочий стол (создать ярлык) (рис. 1.25). В результате на рабочем столе появится ярлык вашей программы Рыбак (рис. 1.26). Вы только что создали приложение под Windows и теперь можете запустить его, щелкнув по ярлыку Рыбак на Рабочем столе.

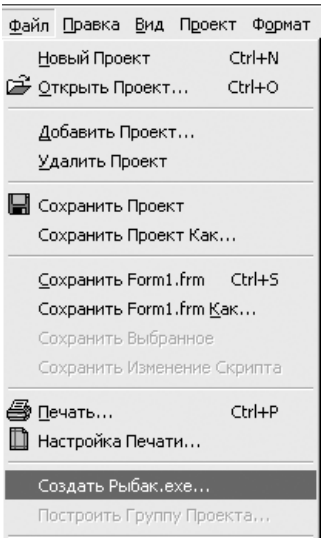


Рис. 1.23

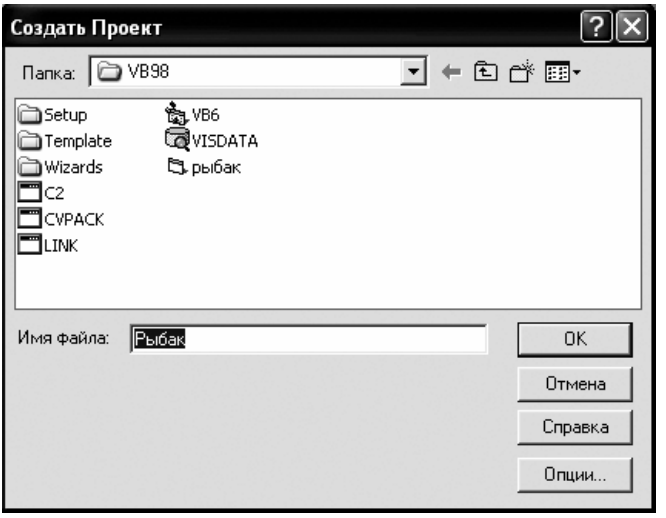


Рис. 1.24

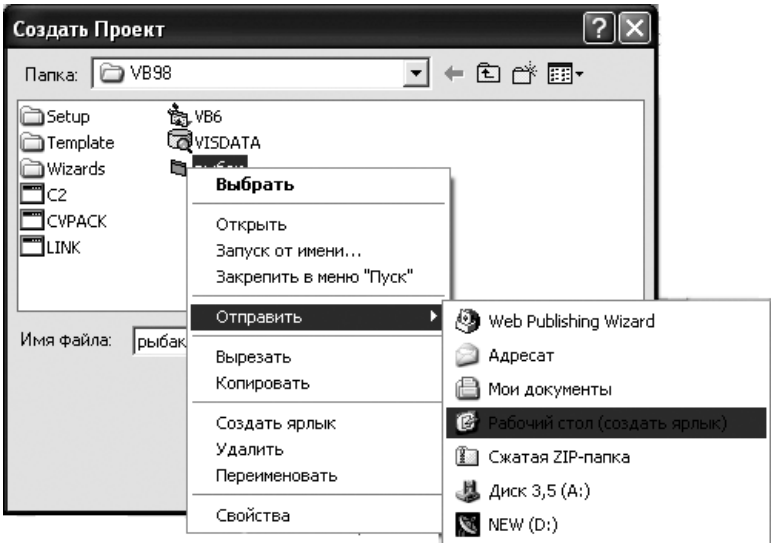


Рис. 1.25

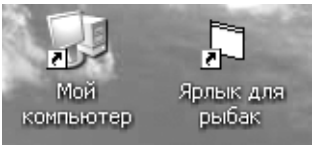


Рис. 1.26

1.5. Работа с событиями

Программы в Visual Basic управляются **событиями**. Например, на любое действие пользователя с мышью и клавиатурой (одинарный или двойной щелчок кнопкой мыши по объекту, перемещение мыши, нажатие клавиши на клавиатуре и т. д.) операционная система генерирует сообщение, которое описывает произведенное пользователем действие. Сообщение доставляется соответствующему объекту, который генерирует событие. Для каждого управляющего элемента имеется список относящихся к нему событий и его можно посмотреть в окне редактора кода (рис. 1.8). Реакцию на любое допустимое событие можно запрограммировать, создав в окне редактора кода **процедуру обработки события**.

Пример 1-2

Задача. При одинарном щелчке по красному полю на этом поле должна появиться надпись желтого цвета Один щелчок. При двойном щелчке должна появиться надпись желтого цвета `Двойной щелчок`.

Решение. На визуальном этапе программирования подготовим форму. Выберем на Панели компонентов компонент `Label` и нарисуем его на форме.

В результате получим объект Label1 с набором свойств. В свойстве Caption введем текст: Щелкни здесь. Выберем из палитры свойства ForeColor желтый цвет (для текста), а для свойства BackColor установим красный цвет (для фона) (рис. 1.27). Не забывайте о том, что значение свойства BackStyle должно быть Непрозрачно.

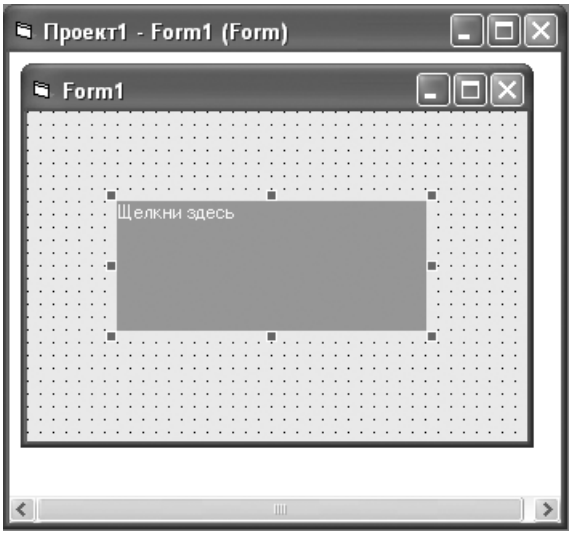


Рис. 1.27

Переходим к второму этапу — написанию кода. Дважды щелкаем на объекте Label1 и откроем **Окно редактора кода** (рис. 1.28). В верхней части окна находятся два раскрывающихся списка. Левый список содержит имена всех объектов на форме и имя самой формы. Правый список содержит набор событий, связанных с выбранным объектом. Посмотрите списки событий для объекта Label1 и Form и убедитесь, что содержимое списков различно. Выбрав имя объекта Label1 и имя события Click, получаем «заготовку» процедуры, состоящую из двух строк (операторных скобок): строки объявления процедуры Private Sub Label1_Click() и строки конца процедуры End Sub. Процедура Sub объявляется своим типом Private – локальная (то есть доступная только внутри текущей формы. Все процедуры по умолчанию имеют тип Private) и именем Label1_Click(), составленным из имени объекта и события (рис. 1.28).

Теперь надо запрограммировать изменение свойства Caption объекта Label1 в соответствии с заданием. На визуальном этапе мы изменяли свойства вручную, отыскивая их в списке на панели свойств. Так был задан первоначальный текст, цвета фона и букв. Теперь обратимся к свойствам объекта из программы. Для этого после имени объекта ставится точка, после которой записывается свойство. Если имя объекта набрано в окне редактора кода верно, то как только будет введена разделяющая точка, появится выпадающий список с перечнем свойств и методов, которыми обладает объект (рис. 1.29).

Выберем свойство Caption и завершим строку (рис. 1.30), не забывая, что присваиваемый свойству Caption текст заключается в кавычки. Напишем ана-

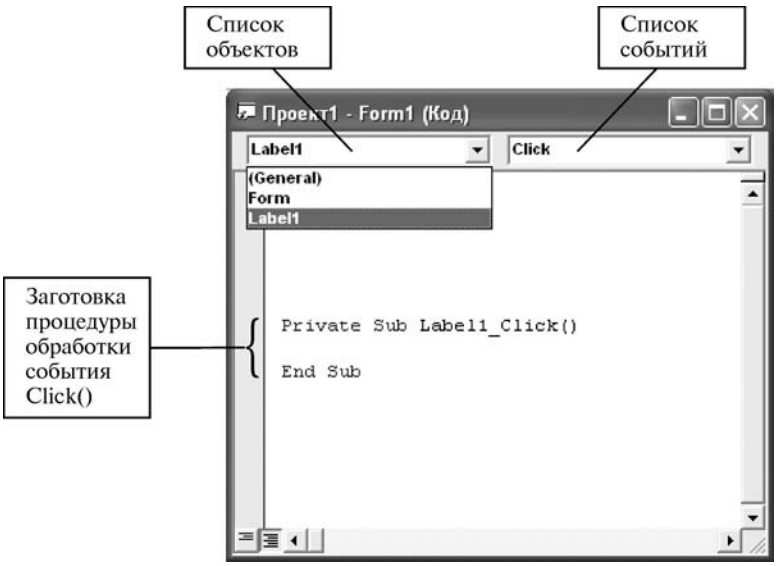


Рис. 1.28

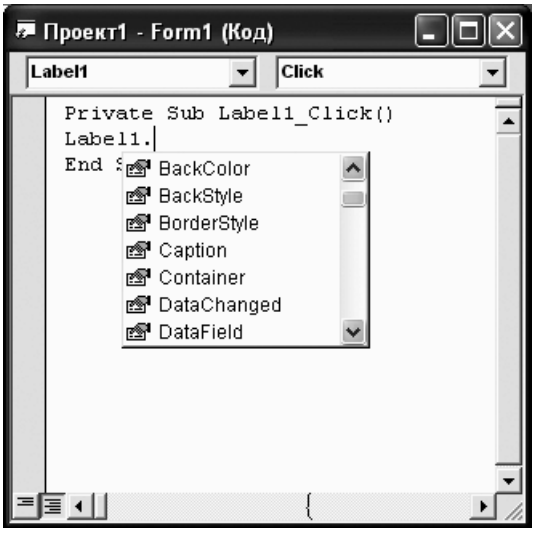


Рис. 1.29

логичную процедуру для двойного щелчка (рис. 1.30). Объект остается прежним Label1, а событие надо выбрать `DblClick` (DoubleClick — двойной щелчок). Запрограммируем изменение свойства Caption и запустим программу, нажав на клавишу **F5** или щелкните по пиктограмме Запуск ▶. Проверьте работу программы. Чтобы остановить выполнение программы, щелкните по пиктограмме Остановка ■ на панели инструментов или нажмите две клавиши **Alt** и **F4**.

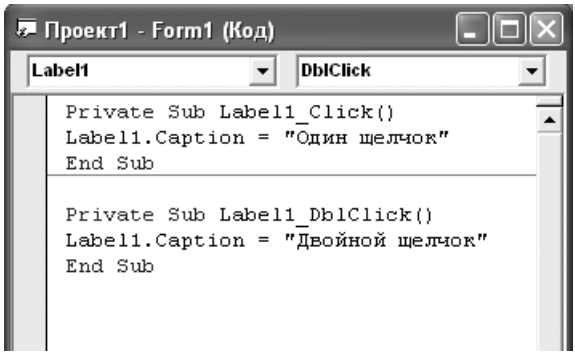


Рис. 1.30

Аналогичным образом можно программно изменять и другие свойства объектов в ответ на определенные события. Рассмотрим следующий пример.

Пример 1-3

Задача. При одинарном щелчке по форме фигура из красного квадрата преобразуется в желтый круг, а при двойном щелчке по форме фигура снова становится красным квадратом.

Решение. На этапе визуального программирования поместим компонент Shape на форму и настроим его цвет на красный (свойство BackStyle устанавливаем на 1 — Непрозрачно, в свойстве BackColor выбираем из палитры красный цвет).

Открываем окно редактора кода. Особенностью данного примера является то, что объект `Shape` создан разработчиками Visual Basic таким образом, что ему не приписано никаких событий и он интерпретируется средой программирования как декоративный элемент. Поэтому привяжем код, описывающий изменение цвета и формы объекта Shape, к процедурам «одинарный щелчок по форме» и «двойной щелчок по форме». На рис. 1.31 приведен текст кода. Изменение типа геометрической фигуры задается обращением объекта Shape1 к свойству Shape и присваиванием ему соответствующей цифры. Для преобразования в круг служит строка `Shape1.Shape = 3`, для преобразования в квадрат строка `Shape1.Shape = 1`.

Обратите внимание на то, как в программе задается изменение цвета объекта `Shape`. Для этого предусмотрено три способа.

Во-первых, можно использовать встроенные функции. В примере на рис. 1.31 цвет изменяется с помощью функции `QBColor(параметр)`, где в качестве параметра указываются числа, соответствующие цвету. Значения этого параметра приведены на рис. 1.32.

Чтобы цвет был виден на фигуре, необходимо установить свойство `BackStyle` на значение `Непрозрачно`. Эту операцию выполняет строка `Shape1.BackStyle = 1` (рис. 1.31).

Во-вторых, для программирования изменения цвета объекта можно использовать встроенные константы: vbYellow — желтый цвет, vbRed — красный, vbGreen — зеленый, vbBlue — голубой и т. д. На рис. 1.33 приведен текст кода с использованием встроенных констант цвета.

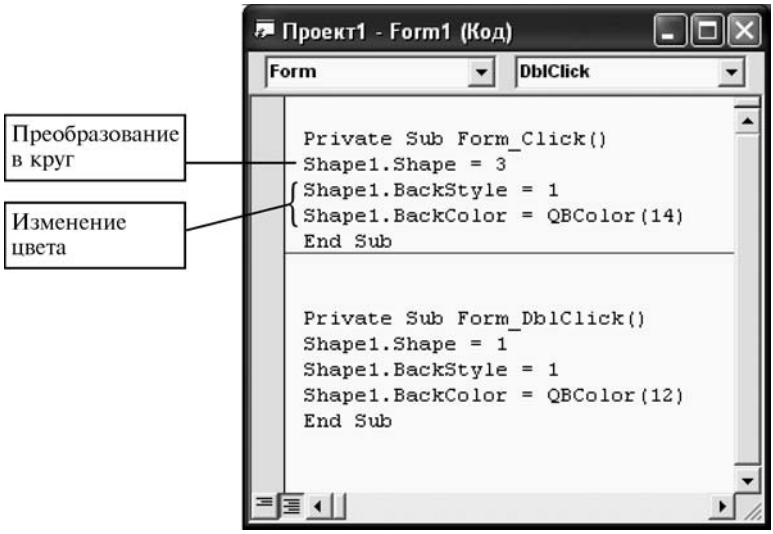


Рис. 1.31

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

Рис. 1.32

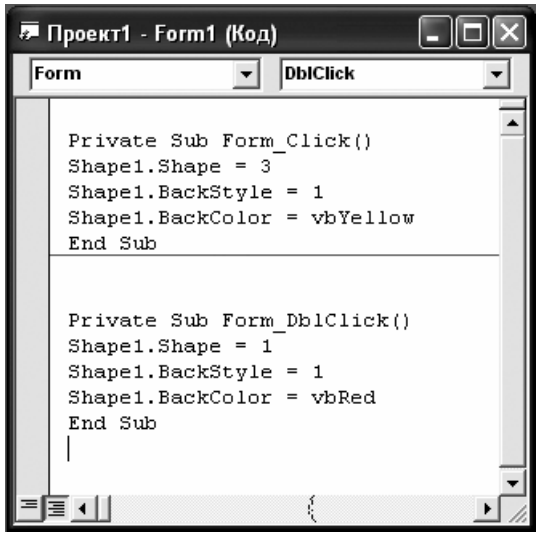


Рис. 1.33

В-третьих, для определения цвета можно применить функцию **RGB(r,g,b),** где значения параметров r, g, b задают соответственно доли основных цветов в результирующем цвете: красного (r означает red), зеленого (g означает green) и синего (b означает blue). Диапазон изменения параметров r, g, b от **0** до **255**. Например, чтобы запрограммировать чистый красный цвет, надо записать RGB(255,0,0), для чистого зеленого — RGB(0,255,0), для чистого синего — RGB(0,0,255). Остальные цвета можно получить, подбирая значения долей трех основных цветов в смеси.

Можно сделать еще одно усовершенствование кода программы, автоматизировав процесс задания начальных значений для цвета и геометрической формы фигуры — красный квадрат. Выше мы делали это вручную на этапе визуального программирования. Но эти действия можно запрограммировать и запускать на исполнение по событию `Load` – загрузка формы. Это событие активизируется при запуске программы на выполнение (клавиша **F5** или пиктограмма Запуск на панели инструментов ▸. Выберем в левом списке окна редактора кода объект `Form`, а в правом списке событие `Load`. В теле созданной процедуры запишем строчки кода для задания красного цвета и квадрата (рис. 1.34).

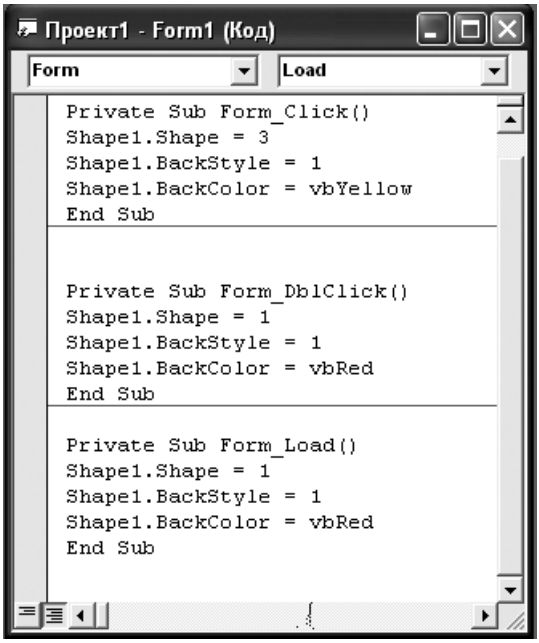


Рис. 1.34

Теперь достаточно лишь разместить на форме объект Shape, не заботясь о его начальном цвете и геометрии — после запуска программы на выполнение начальные условия будут выставлены автоматически. Проверьте это, задав вручную на визуальном этапе синий овал. Затем нажмите клавишу **F5** и убедитесь, что на форме в режиме выполнения изображен красный квадрат. Далее программа будет работать как и раньше. Заметьте, что щелкать

Содержание