МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

С.П. МИНЕЕВ,
Э.Г. ЧЕБОТКОВ

ИНФОРМАТИКА. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА VISUAL BASIC В СРЕДЕ РАЗРАБОТКИ VISUAL STUDIO 2005

Утверждено редакционно-издательским советом университета в качестве учебного пособия

Самара
Самарский государственный технический университет
2015

УДК 004.43(075.8)
М-616

Р е ц е н з е н т ы: д-р техн. наук В.Е. В ы с о ц к и й,
канд. техн. наук А.Н. П р о ц е н к о

Минеев С.П.

М 61 **Информатика. Основы программирования на Visual Basic в среде разработки Visual Studio 2005**: учеб.пособ. / *С.П. Минеев, Э.Г. Чеботков.* – Самара: Самар.гос.техн.ун-т, 2015. – 97 с.:ил.

ISBN

Даны указания для работы в среде разработки Visual Studio 2005 на языке программирования Visual Basic. Рассмотрены основные понятия визуального программирования. Читателю предлагается создать несколько программ с помощью подробных пошаговых процедур.

Предназначено для студентов, обучающихся по направлению «Электротехника, электромеханика и электротехнология» (140600), для глубокой и самостоятельной проработки и самоконтроля

УДК 004.43(075.8)
М-616

ISBN

ПРЕДИСЛОВИЕ

Обычно язык программирования Basic ассоциируется с чем-то очень простым в освоении и использовании для программирования. Это действительно так. На заре компьютерных технологий язык Basic был создан для простых программ и использовался в качестве учебного языка для первых шагов при изучении основ программирования с последующим переходом на более сложные и универсальные языки. Это было заложено в название языка *Basic (Beginners All-purpose Symbolic Instructional Code),* т.е. многоцелевой код символьных инструкций для начинающих. С прогрессом компьютерных технологий развивался и Basic. В настоящее время версия Visual Basic 2005 дает возможность решать любые современные задачи разработки приложений. При этом Visual Basic 2005 остался достаточно простым в освоении, став в то же время мощным современным языком программирования.

Visual Basic 2005 поставляется в виде отдельного пакета Microsoft Visual Basic 2005 Express Edition, а также входит в состав Microsoft Visual Studio 2005.

С помощью Visual Basic 2005 можно создавать приложения практически для любой области современных компьютерных технологий: бизнес-приложения, игры, мультимедиа, базы данных, интернет-приложения. При этом приложения могут быть как простыми, так и очень сложными, в зависимости от поставленной задачи.

Простота и мощность языка Visual Basic 2005 позволили сделать его встроенным языком для приложений Microsoft Office. В настоящее время Visual Basic уже не считается учебным языком – знание Visual Basic и его диалектов (VBA, VBScript) становится необходимостью для современного программиста любого уровня.

В Visual Basic 2005 используются все самые современные методы программирования: объектно-ориентированная модель, включая наследование визуальных классов; модель составных объектов COM (Component Object Model); технология программных компонентов

ActiveX и др. Суть этих подходов и их реализацию на примерах можно изучить, прочитав посвященные им главы из настоящего издания.

Учебное пособие предназначено для студентов, обучающихся по электротехнической специальности, однако может быть полезно для студентов других специальностей.

Авторы с благодарностью примут все замечания и пожелания читателей и просят направлять их по адресу: г. Самара, ул. Первомайская, д. 18, ауд. 134.

ВВЕДЕНИЕ

Авторы постарались в первую очередь доступно и подробно показать, как начинать работать в среде разработки Visual Studio – Visual Basic.

Учебное пособие состоит из двух частей.

Изучив первую часть, вы узнаете основные команды меню и способы программирования в Visual Studio; откроете и запустите простую программу на Visual Basic; измените настройки программирования; попрактикуетесь в перетаскивании, изменении размера и закреплении окон инструментов.

Изучив вторую часть, вы узнаете, как создать простой, но привлекательный графический интерфейс с помощью элементов управления из области элементов Visual Studio; как настраивать их поведение, изменяя свойства; напишете код, который определит, что будет делать ваша программа.

Упражнения показаны в виде пошаговых процедур. Каждый шаг – это элементарное действие, которое легко понять, так как каждое действие подробно описано и проиллюстрировано. Упражнения научат обращаться с большим количеством объектов, свойствами и кодом программы.

Каждая часть пособия заканчивается краткой справочной информацией по пройденному материалу. Кроме того, в конце каждой ча-

сти имеется большое количество контрольных вопросов, отвечая на которые, можно эффективно изучить материал.

Так как настоящее учебное пособие посвящено программированию, то здесь встречается примеры текстов программ, которые выглядят следующим образом:

```
    Public Class frmFirstProgram

    Private Sub btnExit_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnExit.Click
        End
    End Sub


    Private Sub btnWelcome_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnWelcome.Click
        lblMessage.Text = "Здравствуй, Мир!"
    End Sub

    Private Sub frmFirstProgram_Load(ByVal sender As Object, ByVal
e As System.EventArgs) Handles Me.Load
        lblMessage.Text = ""
    End Sub

    End Class
```

Так как строки программы могут быть длинными, то они могут не уместиться на одной строке. В этом случае они переносятся на другую строку. Обратите внимание, что строка программы, начинающаяся не с красной строки, является продолжением предыдущей строки. Например:

```
    Private Sub frmFirstProgram_Load(ByVal sender As Object, ByVal
e As System.EventArgs) Handles Me.Load
```

Следует обратить внимание на текст, выделенный **полужирным шрифтом**. Таким способом отмечены понятия, на которые нужно обратить особое внимание. К таким понятиям относятся определения, объекты, свойства, методы.

Для обозначения клавиш в тексте используются скобки < >. Например, если в тексте встречается **<Ctrl>**, то это означает, что

нужно нажать данную клавишу на клавиатуре. Если в тексте встречается **<Ctrl+Пробел>**, то это означает, что нужно нажать эти клавиши в сочетании. Знак «+» говорит о том, что клавишу **<Ctrl>** нужно нажать и, удерживая ее, нажать **Пробел**.

1. ОТКРЫТИЕ И ЗАПУСК ПРОГРАММЫ В VISUAL BASIC 2005

1.1. ЦЕЛЬ ЗАНЯТИЯ

Цель занятия состоит в следующем:

– запускать Microsoft Visual Studio 2005;

– использовать среду разработки Visual Studio;

– открыть и запустить программу на Visual Basic;

– настроить свойства среды;

– перемещать, изменять размер и закреплять окна инструментов или делать их всплывающими;

– использовать справочную систему и завершать работу в Visual Studio.

Microsoft Visual Basic 2005 – это обновленная и расширенная версия популярной системы разработки Visual Basic.Net, которая в свою очередь стала революционной в плане изменений по сравнению с предыдущей системой разработки Visual Basic 6. Далее рассказано об основных приемах быстрой и эффективной работы с программами в этой среде. Научившись свободно ориентироваться в среде разработки Visual Studio, вы сможете использовать те же самые инструменты для написания программ в **Microsoft Visual C++ 2005**, **Microsoft Visual C# 2005**, **Microsoft Visual J# 2005**, а также компиляторы и инструменты третьих фирм.

Вы узнаете, как запустить Visual Basic и как в интегрированной среде разработки открыть и запустить простую программу. Вы изучите основные команды меню и способы программирования в Visual

Studio; откроете и запустите простую программу на Visual Basic, которая называется МузВопрос; измените настройки программирования; попрактикуетесь в перетаскивании, изменении размера и закреплении окон инструментов. Вы также узнаете, как получить дополнительную информацию с помощью справочной системы и как корректно завершить работу в среде разработки.

Важно! Чтобы выполнить упражнения из этого пособия, нужно установить одну из редакций Visual Studio 2005. Данное упражнение и все последующие можно выполнять также и в предшествующей системе разработке Visual Studio.Net, однако в еще более ранней системе разработки Visual Basic 6 упражнения выполнить не удастся. Как уже говорилось выше, Visual Studio.Net содержит революционные изменения по сравнению с Visual Basic 6.

1.2. СРЕДА РАЗРАБОТКИ VISUAL STUDIO. ЗАПУСК VISUAL STUDIO

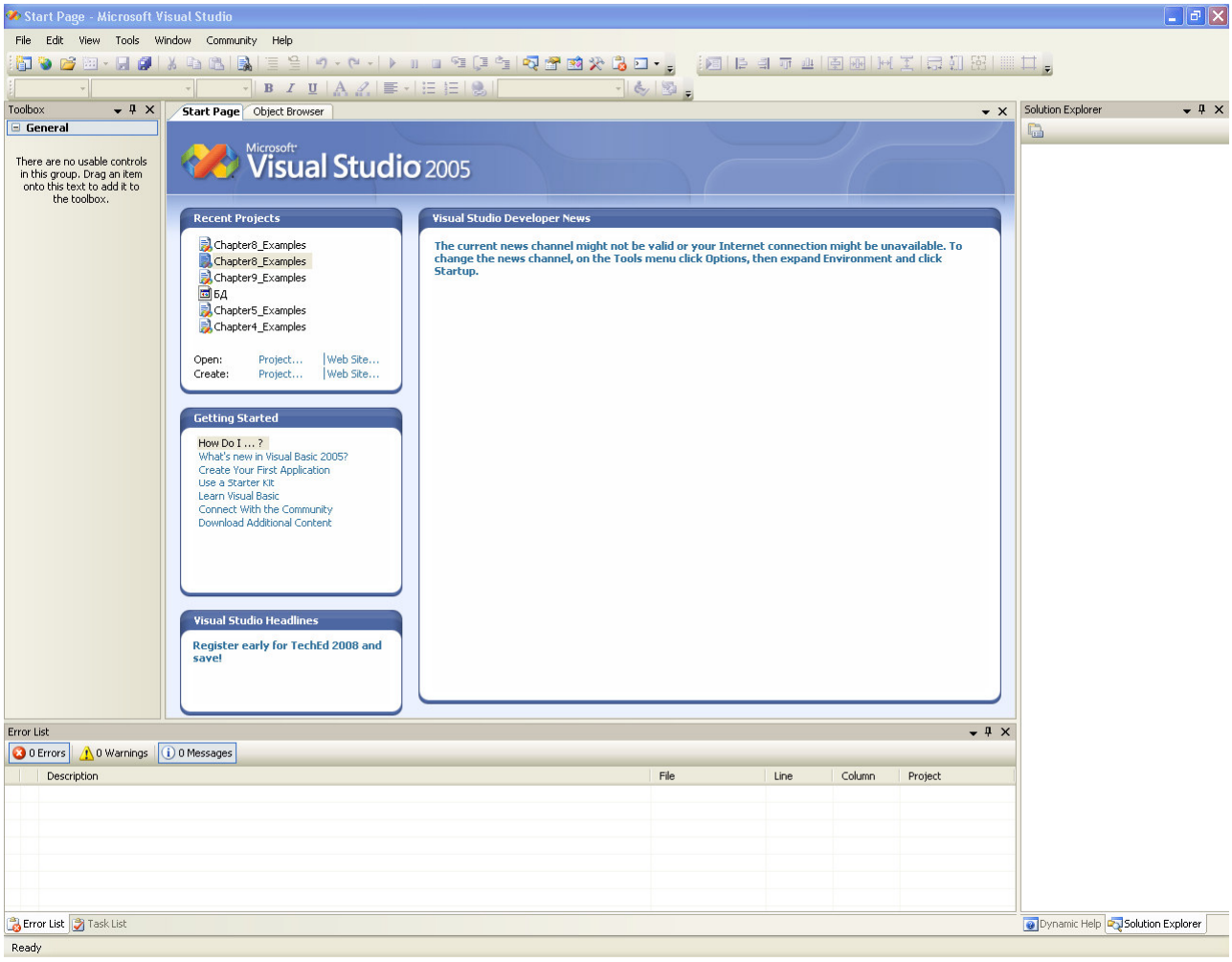
Хотя язык программирования, который вы будете изучать, называется Visual Basic, среда разработки программ называется Microsoft Visual Studio 2005. Это мощная и настраиваемая среда программирования, которая содержит все инструменты, необходимые для быстрого и эффективного создания надежных программ для Windows. Большинство возможностей Visual Studio можно одинаково использовать в Visual Basic 2005, Visual C++ 2005, Visual C# 2005 и Visual J# 2005.

Для запуска Visual Studio выполните следующие действия.

1. На панели задач Windows нажмите кнопку Start (Пуск), наведите указатель на All Programs (Все программы), а затем – на папку Microsoft Visual Studio 2005. Появится список программ, находящихся в папке Microsoft Visual Studio 2005.

2. Выберите значок Microsoft Visual Studio 2005. Запустится Visual Studio, и вы увидите на экране среду разработки (рис. 1.1) с ее многочисленными меню, инструментами и окнами с компонентами (иногда эти окна также называются окнами инструментов.) Если это первый запуск Visual Studio после новой установки, то должна по-

явиться Start Page (Начальная страница). Если Start Page (Начальная страница) не видна, ее можно вызвать из меню View (Вид) – Other Windows (Другие окна) по команде Start Page (Начальная страница).



Р и с. 1.1. Среда разработки Visual Studio 2005

На рис. 1.1 показано главное окно Visual Studio 2005, каким оно выглядит после запуска программы.

В нем можно выделить несколько основных объектов: стандартная панель инструментов, окна **Start Page** (Начальная страница), **Solution Explorer** (Обозреватель решений), **Toolbox** (Инструментарий) и **Dynamic Help** (Динамическая справка).

Visual Studio 2005 предоставляет в распоряжение пользователя набор самых разнообразных панелей инструментов. Эти панели инструментов содержат кнопки, назначение которых зависит от функций конкретной панели инструментов. После запуска Visual Studio 2005 на экране отображается стандартная панель инструментов.

Окно **Start Page** (Начальная страница) позволяет открывать недавно использовавшиеся проекты; осуществляет поиск примеров как из справочной системы, так и из Интернета; содержит различные ссылки на сайты, которые могут помочь при работе с Visual Studio 2005.

В окне **Solution Explorer** (Обозреватель решений) размещаются проекты и файлы текущего решения.

Для получения подробной информации об объектах используется диалоговое окно **Object Browser** (Просмотр объектов). Оно позволяет искать и исследовать элементы, их свойства, методы, события, находящиеся в проектах и ссылках на них, как бы представляя собой внутреннюю библиотеку.

Для удобства разработки используется окно **Dynamic Help** (Динамическая справка). Во время работы оно постоянно обновляется, и предлагается справочная информация, относящаяся к текущим действиям.

К средству, призванному облегчить разработку, относится также окно **Toolbox** (Инструментарий), отображающее элементы, используемые в проектах Visual Basic. К таким элементам могут относиться компоненты .NET и COM, HTML-объекты, фрагменты кода, текст.

1.3. ОТКРЫТИЕ ПРОЕКТА VISUAL BASIC

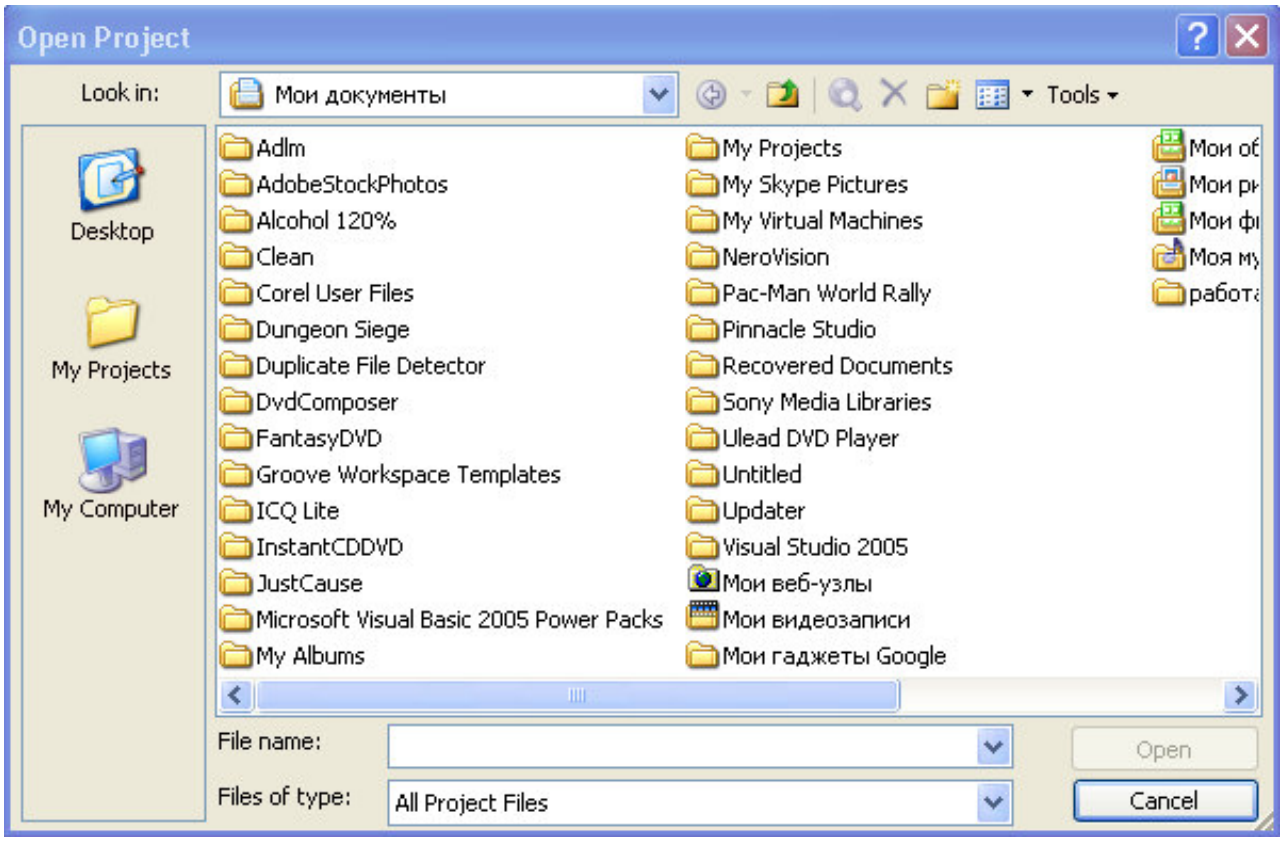
1. Чтобы открыть существующий проект, выполните одно из следующих действий:

– в окне **Start Page** (Начальная страница) выберите ссылку **Open Project** (Открыть проект);

– из меню **File** (файл) выберите пункт **Open Project** (Открыть проект).

На экране появится диалоговое окно Open Project (Открытие проекта) (рис. 1.2).

2. В раскрывающемся списке Look in в верхней части окна Open Project (Открытие проекта) выберите диск C (если учебные файлы установлены на другом диске, выберите свой диск).



Р и с. 1.2. Окно Open Project (Открыть проект)

3. Дважды щелкните на папке с именем VB2005 (если папка с упражнениями имеет другое имя, дважды щелкните на ней). Вы увидите подпапки с именами Занятие1, Занятие2, Занятие3, Занятие4 и т.д.

4. Откройте папку Занятие1\МузВопрос, а затем дважды щелкните мышью на файле проекта МузВопрос (МузВопрос.sln). Visual Studio загрузит форму МузВопрос, свойства и код программы проекта. Возможно, все еще будет видна начальная страница, но в Обозревателе решений в верхнем правом углу экрана уже будет виден список файлов проекта. В этом случае в окне Solution Explorer (окно решений) дважды щелкните по ссылке MusicTrivia.vb. После этого форма МузВопрос должна отобразиться.

1.4. ПРОЕКТЫ И РЕШЕНИЯ

В Visual Studio разрабатываемые программы обычно называются **проектами** или **решениями**, так как они содержат не один-

единственный файл, а много отдельных компонентов. Для программ на Visual Basic 2005 всегда имеются файл проекта (.vbproj) и файл решения (.sln). Файл проекта содержит информацию, относящуюся к одной программной задаче. Файл решения содержит информацию об одном или нескольких проектах. Примеры, прилагаемые к этому пособию, содержат по одному проекту для каждого решения, так что открытие файла проекта (.vbproj) будет иметь такой же эффект, как и открытие файла решения (.sln). Но для решений, содержащих несколько проектов, вы должны открывать файл решения.

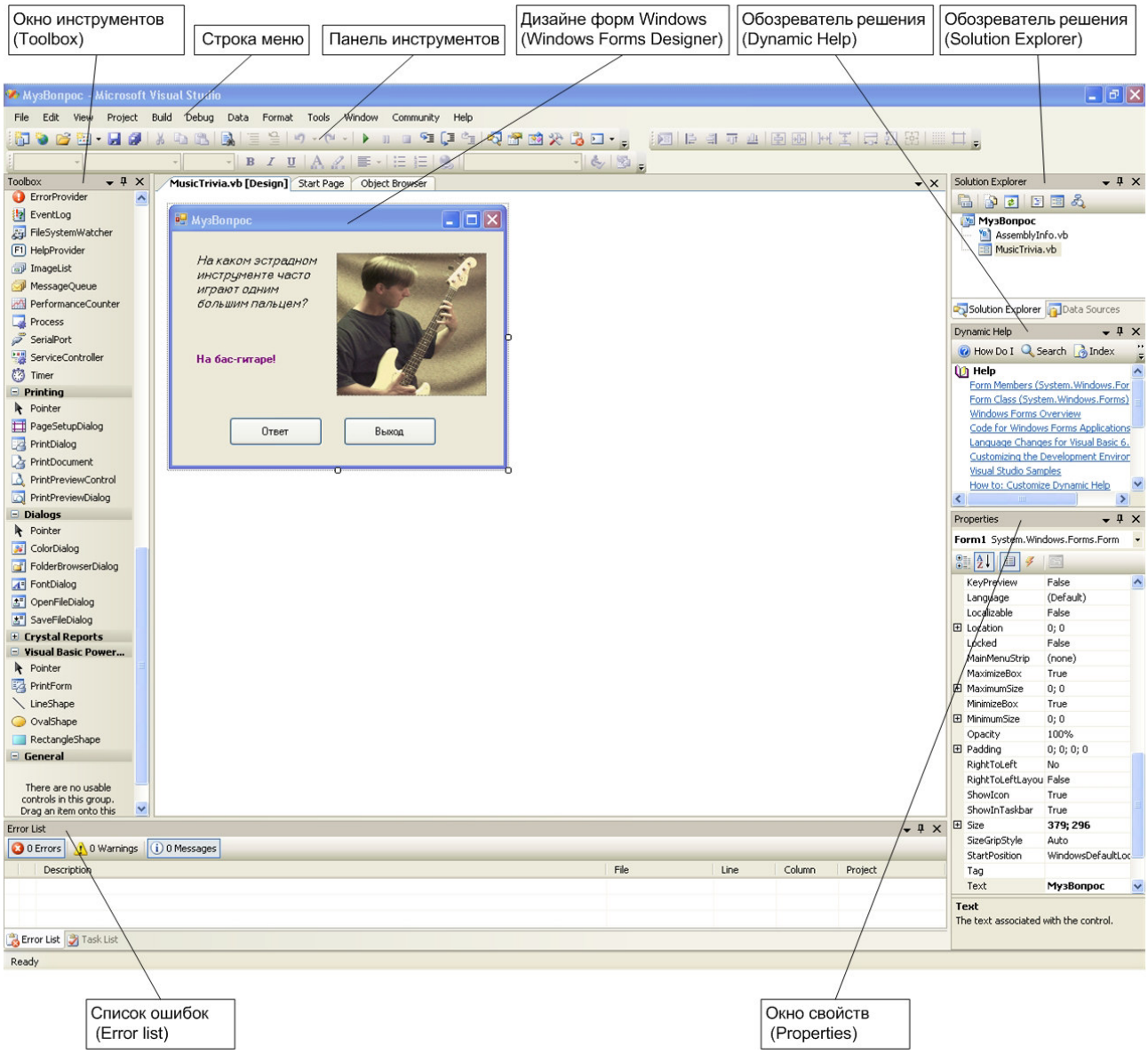
1.5. ИНСТРУМЕНТЫ VISUAL STUDIO 2005

Теперь подробнее познакомимся со средой разработки, инструментами программирования и окнами, которые нам понадобятся при изучении этого курса. Если вы писали программы на Visual Basic, то вам уже знакомы многие инструменты разработки. В целом **инструменты** – это средства или функции, которые используются при создании, настройке и тестировании программ на Visual Basic. Некоторые инструменты помогут получить дополнительную информацию о ресурсах системы, включая доступные базы данных и соединения с веб-сайтами. Имеется также несколько мощных инструментов для работы со справочной системой.

На рис. 1.3 показаны некоторые инструменты, окна и другие элементы среды разработки Visual Studio. Не удивляйтесь, если эта иллюстрация выглядит не совсем так, как среда разработки на вашем компьютере – внешний вид может отличаться. Далее вы познакомитесь с этими элементами подробнее.

Большинство команд, которые управляют средой разработки, доступны в верхней **строке меню**. Меню и команды работают точно так же, как и во всех остальных программах для Windows, и к ним можно получить доступ как с помощью клавиатуры, так и с помощью мыши. Под строкой меню расположена **стандартная панель инструментов**,

представляющая собой набор кнопок, запускающих те или иные команды или управляющих средой разработки Visual Studio. Чтобы выполнить команду, связанную с кнопкой на панели инструментов, щелкните по этой кнопке мышью.



Р и с. 1.3. Инструменты Visual Studio

Основные инструменты в среде разработки Visual Studio – это **Solution Explorer** (Обозреватель решений), окно **Properties** (Свойства), **Dynamic Help** (Динамическая справка), **Windows Forms Designer** (Конструктор Windows Forms), **Toolbox** (Область элементов) и окно **Output** (Вывод). Есть также несколько специализированных инструментов, в числе которых **Server Explorer** (Обозреватель серверов) и

Class View (Представление классов). Они чаще всего представлены в виде закладок вдоль границ среды разработки, или в нижней части окон инструментов, или вообще невидимы. Если какой-либо инструмент не виден и он вам потребовался, откройте меню **View** (Вид) и выберите инструмент, который вы хотите использовать. Если вы не видите то, что вам нужно, выберите строку **Other Windows** (Другие окна).

Точный размер и форма окон и инструментов зависят от настроек вашей среды разработки. Visual Studio позволяет располагать и закреплять окна так, чтобы сделать видимыми все необходимые элементы. Вы также можете скрыть инструменты так, что они примут форму закладок по краям среды разработки и останутся невидимыми до тех пор, пока они снова вам не понадобятся. Понять, какие инструменты важны вам в данный момент, а какие можно изучить позже, – непростая проблема при знакомстве с насыщенным интерфейсом Visual Studio.

Среда разработки содержит множество полезных инструментов и имеет при этом простые механизмы для приведения их в порядок. Для этого используются такие механизмы, как закрепление окон, автоскрытие окон и другие, которые будут описаны несколько позже. Если вы просто запускаете Visual Basic, наилучший способ справиться с перегруженностью инструментами – скрыть инструменты, которые вы не будете использовать часто, и освободить место для наиболее нужных инструментов. Основные инструменты для программирования на Visual Basic – это **Solution Explorer** (Обозреватель решений), окно **Properties** (Свойства), **Windows Forms Designer** (Конструктор Windows Forms), **Toolbox** (Область элементов), **Dynamic Help** (Динамическая справка). Вам вряд ли скоро потребуются окна **Server Explorer** (Обозреватель серверов) , **Class View** (Представление классов), **Resourse View** (Просмотр ресурсов), **Object Browser** (Обозреватель объектов) или **Debug** (Отладка).

В следующих упражнениях вы начнете экспериментировать с основными инструментами среды разработки Visual Studio. Вы также узнаете, как скрывать инструменты, которые пока не нужны.

1.6. WINDOWS FORMS DESIGNER (КОНСТРУКТОР WINDOWS FORMS)

Если вы выполнили предыдущее упражнение, то в среде разработки Visual Studio будет загружен проект МузВопрос, однако пользовательский интерфейс и основная графическая форма проекта могут быть все еще не видны в Visual Studio. (Более сложные проекты могут содержать несколько графических форм, но в этой простой программе требуется только одна.) Чтобы сделать форму проекта МузВопрос видимой в среде разработки, вы должны вывести ее на экран с помощью Solution Explorer (Обозревателя решений).

Для того чтобы отобразить Windows Forms Designer (Конструктора Windows Forms), выполните следующие действия.

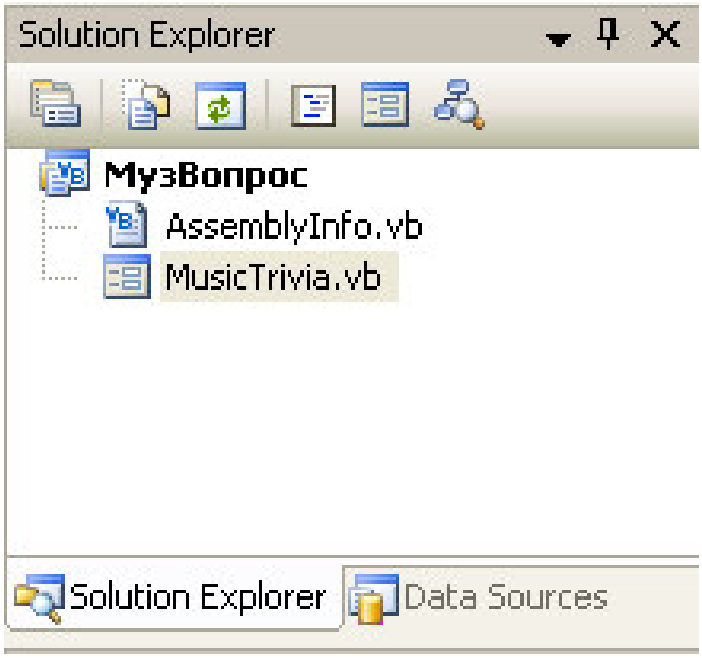
1. Найдите панель Solution Explorer (Обозреватель решений), расположенную в верхнем правом углу среды разработки Visual Studio. Если вы ее не видите (она может быть скрыта в виде закладки так, что ее не видно, или вообще отключена), то в меню View (Вид) выберите Solution Explorer (Обозреватель решений).

Если проект МузВопрос загружен, Solution Explorer (Обозреватель решений) выглядит так, как показано на рис. 1.4.

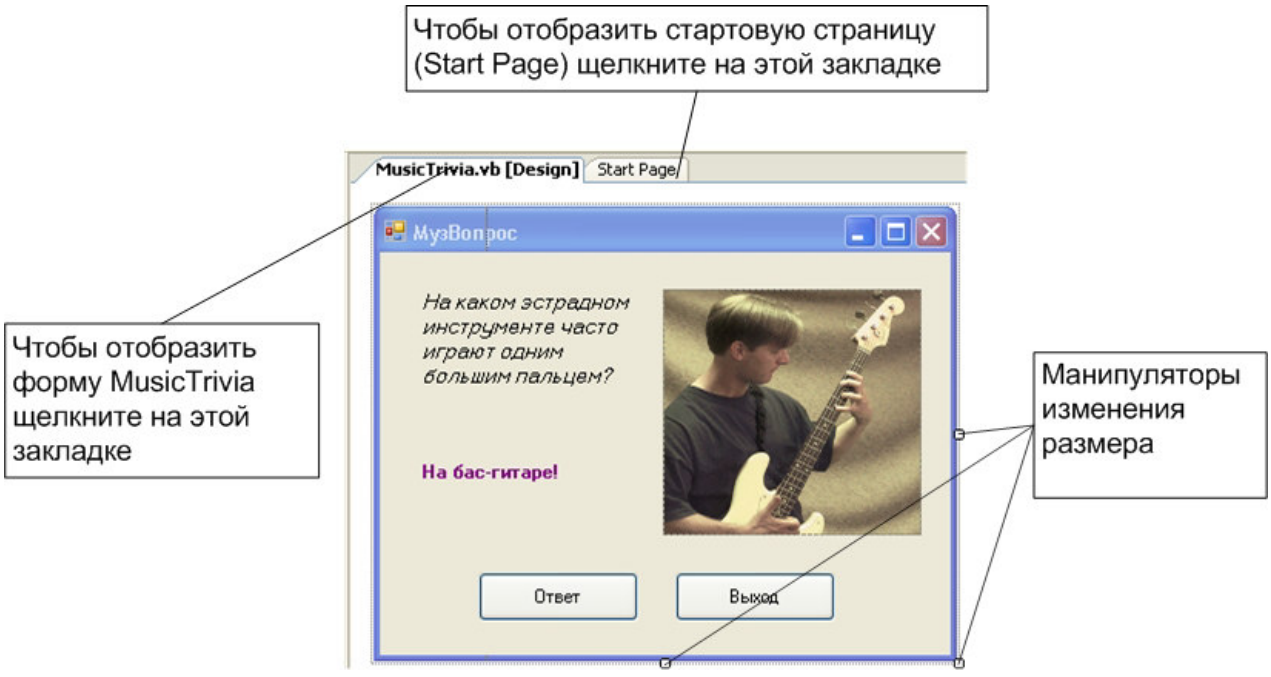
2. В окне Solution Explorer (Обозреватель решений) выберите форму MusicTrivia.vb. У всех файлов форм, включая и этот, рядом находится маленький значок формы, так что их легко идентифицировать. Когда вы щелкаете на файле формы, Visual Studio выделяет его в Solution Explorer (Обозревателе решений), а в окне Properties (Свойства), если оно отображено на экране, появляется информация об этом файле.

3. Чтобы показать рабочее поле, щелкните в Solution Explorer (Обозревателе решений) на кнопке View Designer (Просмотреть конструктор). Форма МузВопрос отображается в Windows Forms Designer (Конструкторе Windows Forms), как показано на рис.1.5.

Обратите внимание, что в верхней части Windows Forms Designer (Конструктора Windows Forms) по-прежнему видна закладка Start Page (Начальная страница). Вы можете щелкнуть на этой закладке, чтобы открыть Start Page (Начальную страницу), изменить настройки своего профиля или открыть дополнительные файлы проектов. Чтобы вернуться в окно Конструктора Windows Forms, щелкните на закладке с меткой MusicTrivia.vb [Design], расположенной над формой МузВопрос.



Р и с. 1.4. Панель Solution Explorer



Р и с. 1.5. Конструктор форм Windows (Windows Forms Designer)

Совет. Если вы не видите закладок Start Page (Начальная страница) и MusicTrivia.vb [Design], возможно, ваша среда разработки находится в режиме Среда MDI, а не в режиме Tabbed documents (Вкладки). Чтобы изменить эту настройку, в меню Tools (Сервис) выберите строку Options (Параметры). В левой части диалогового окна Options (Параметры) выберите папку Environment (Среда), а затем General (Общие). В правой части в разделе Options (Параметры) щелкните на переключателе Tabbed documents (Вкладки), а затем – на OK. Когда вы запустите Visual Studio в следующий раз, открытые вами окна будут иметь закладки, с помощью которых можно переключаться между окнами.

1.7. ЗАПУСК ПРОГРАММЫ НА ЯЗЫКЕ VISUAL BASIC

МузВопрос – это простая программа на языке Visual Basic, разработанная специально, чтобы дать вам представление об инструментах программирования Visual Studio. В форму, которую вы сейчас видите, добавлено пять объектов (две надписи, изображение и две кнопки), а в код программы – три строки, которые заставляют эту простую программу задавать простой вопрос и затем выдавать ответ. (Сейчас ответ виден, так как вы находитесь в режиме разработки, но когда вы запустите программу, ответ будет скрыт.) О создании объектов и добавлении кода программы вы узнаете далее, а сейчас с помощью среды разработки Visual Studio попробуйте запустить эту программу.

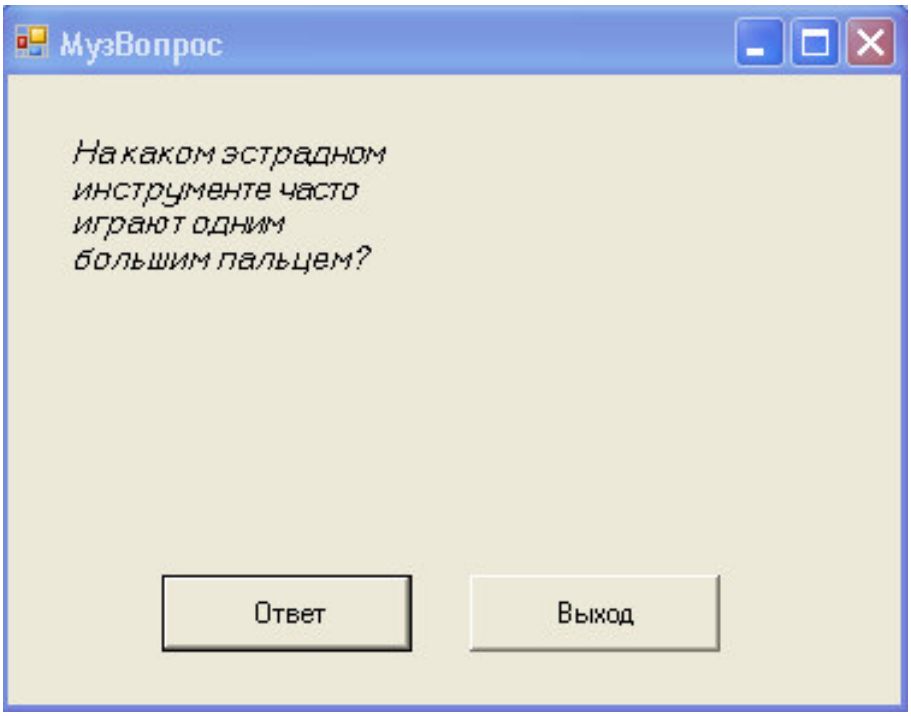
Чтобы запустить и работать с программой МузВопрос из Visual Studio, выполните следующие действия.

1. Щелкните на кнопке **Start Debugging** (Начать Отладку)  , расположенной на стандартной панели инструментов.

Совет. Чтобы запустить программу в среде разработки Visual Studio, можно также нажать на клавиатуре <F5> или выбрать команду **Start Debugging** (Начать Отладку) из меню **Debug** (Отладка).

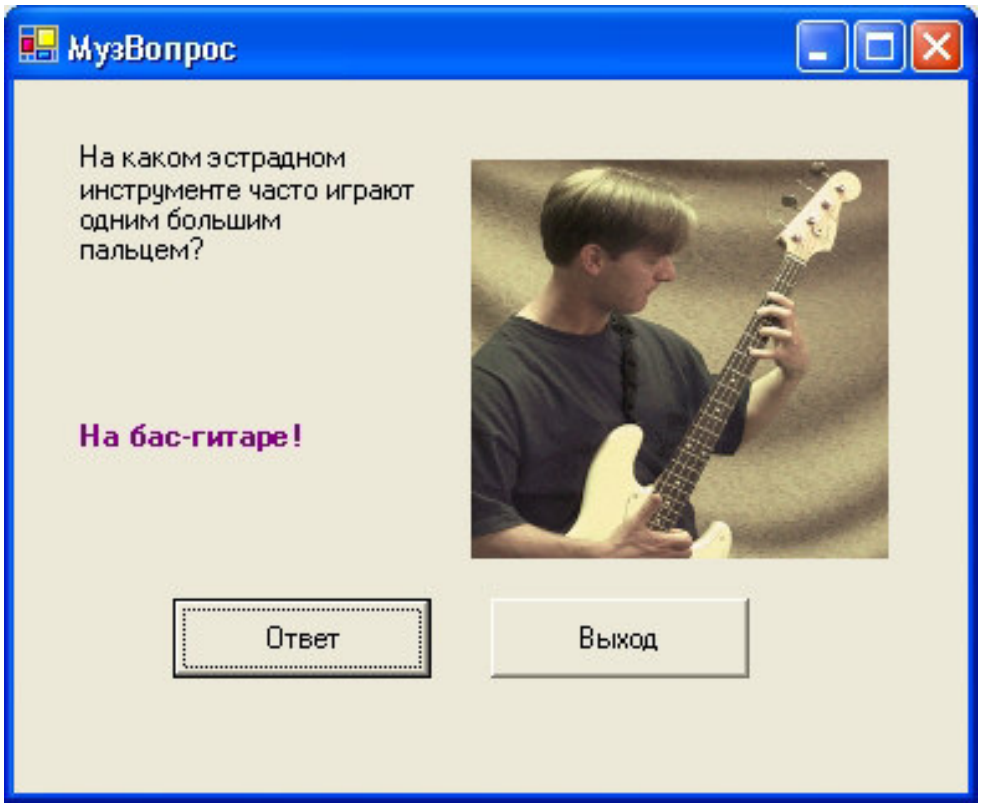
Visual Studio загружает и компилирует проект в сборку, а затем запускает программу в среде разработки. Значок программы появля-

ется на панели задач Windows. Во время компиляции в окне **Output** (Вывод) печатаются некоторые сведения об этапах загрузки и компиляции и об ошибках, если они возникают. Через некоторое время вы снова увидите форму МузВопрос, причем фотография и текст ответа сначала не видны.



Р и с. 1.6. Окно программы после запуска

2. Чтобы увидеть ответ на вопрос, нажмите кнопку Ответ. Ниже вопроса программа выведет текст ответа («На бас-гитаре!»), а затем и фотографию гитариста. Тестовая программа работает (рис. 1.7).



Р и с. 1.7. Окно программы после нажатия на кнопку Ответ

3. Чтобы закрыть программу, нажмите **Выход**. Форма закроется, а среда разработки Visual Studio снова станет активной.

1.8. ОКНО PROPERTIES (СВОЙСТВА)

Окно **Properties** (Свойства) предназначено для отображения и настройки **свойств** объектов решения, включая форму и размещенные в ней объекты. В нем, например, содержатся такие свойства выбранного объекта, как позиция в форме, высота, ширина, цвет.

Чтобы открыть диалоговое окно Properties (Свойства), нужно выполнить одно из следующих действий:

– в меню View (Вид) выберите команду Properties Window (Окно свойств);

– нажмите кнопку Properties Window (Окно свойств) , расположенную на стандартной панели инструментов;

– выберите команду Properties (Свойства) контекстного меню выделенного объекта;

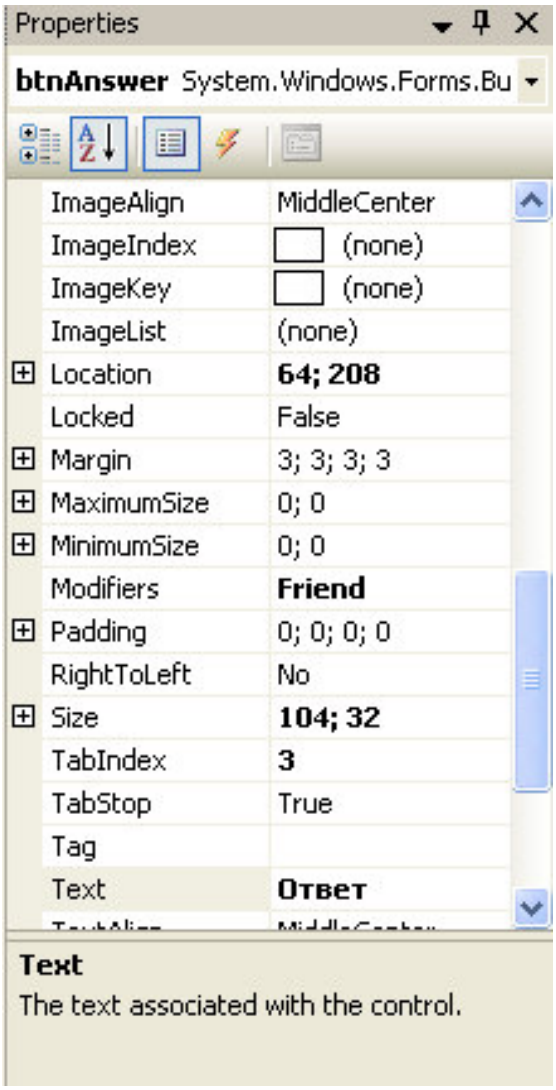
– нажмите клавишу <F4>.

Окно Properties показано на рис. 1.8.

Поскольку форма и элементы управления каждый сам по себе являются объектами, то набор свойств в этом окне меняется в зависимости от выбранного объекта. С помощью кнопок **Alphabetical** (По алфавиту)  и **Categorized** (По категориям)  свойства объекта можно просмотреть в алфавитном порядке или по группам (категориям) соответственно.

В нижней части окна появляется подсказка, поясняющая назначение выбранного свойства объекта. Более подробное пояснение можно посмотреть в справочной системе Visual Basic, выделив интересующее свойство и нажав клавишу <F1>. Можно также воспользоваться динамической справкой, выбрав нужный раздел в окне **Dynamic Help** (Динамическая справка).

Окно **Properties** (Свойства) содержит раскрывающийся список всех элементов интерфейса пользователя (объектов формы). Для каждого из объектов в окне появляется полный перечень всех свойств, доступных для изменения.



Р и с.1.8. Окно Properties (Окно свойств)

Используя диалоговое окно **Properties** (Свойства), можно изменить установленные по умолчанию свойства объектов. Часть свойств объекта, например размеры и расположение, можно задать перемещением объекта и изменением его размеров с помощью мыши в конструкторе форм. Свойства, установленные в окне свойств, можно изменять при выполнении приложения, написав соответствующие коды в процедурах, создаваемых с помощью **Code Editor** (Редакторе кода).

Как правило, форма содержит много объектов. Если выбрать сразу несколько объектов, то в окне свойств можно увидеть общие для этих объектов свойства.

Для объекта, содержащего вопрос в программе МузВопрос, можно изменить шрифт или размер шрифта, или текст может появляться

в другом месте. (При работе с Visual Studio можно задать для текста любой шрифт из тех, что установлены в вашей системе.)

Чтобы изменить свойства объектов в программе МузВопрос, выполните следующие действия свойства.

1. Щелкните на объекте `lblQuestion` формы МузВопрос (объект `lblQuestion` содержит текст «На каком эстрадном инструменте часто играют одним большим пальцем?»). Чтобы начать работать с объектом формы, необходимо сначала выбрать этот объект. Когда вы выбрали объект, вокруг него появляется рамка и маркеры для изменения размера, а в окне Properties (Свойства) отображаются его свойства.

2. Откройте окно Properties (Свойства), если оно не открыто, одним из вышеперечисленных способов.

Обычно оно появляется ниже Solution Explorer (Обозревателя решений) в правой части среды разработки. В окне Properties (Свойства) перечислены все свойства первой надписи, имеющейся на форме. **Названия свойств** перечислены **в левой части** таблички, а **текущие настройки** для каждого свойства показаны **в правом столбце**. Так как существует множество свойств (и большая часть из них изменяется редко), Visual Studio организует их по категориям и отображает в виде структуры. Если категория содержит рядом с собой знак (+), то ее можно раскрыть двойным щелчком и увидеть все свойства, входящие в эту категорию. Если рядом с категорией стоит знак (-), то все свойства видны, а чтобы скрыть этот список и оставить только имя категории, щелкните по знаку «минус».

3. Используя полосу прокрутки, спуститесь вниз по списку свойств, пока не появится свойство `Font`. Полоса прокрутки в окне Properties (Свойства) работает так же, как и для обычных списков.

4. Щелкните на названии свойства `Font` в левом столбце.

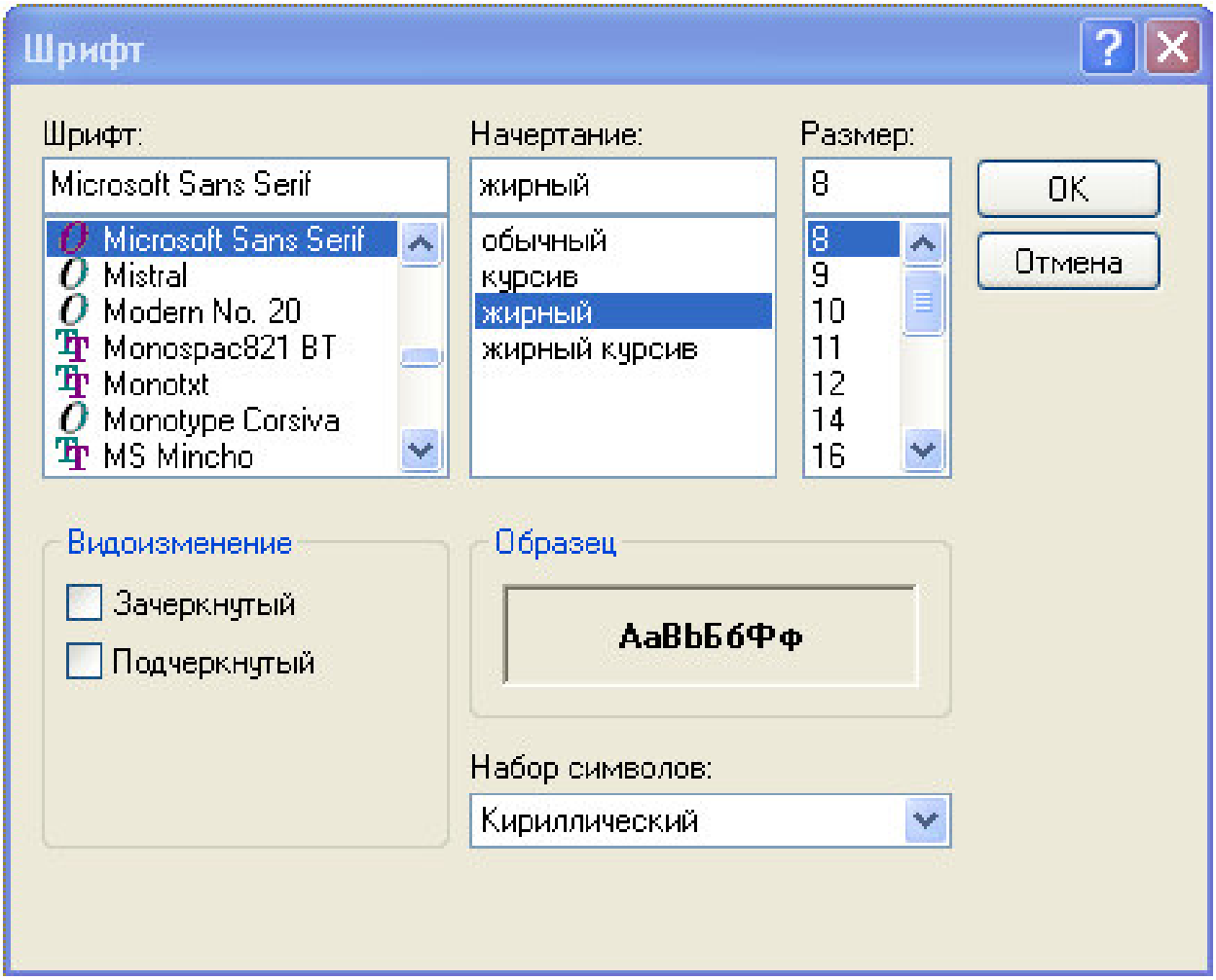
В правом столбце частично видно имя текущего шрифта (Microsoft Sans Serif), а рядом с ним появляется кнопка с многоточием. Эта кнопка означает, что для настройки данного свойства существует дополнительное окно.

5. Нажмите кнопку многоточия в строке `Font`.

Visual Studio покажет диалоговое окно `Font` (Шрифт) (рис. 1.9), в котором можно уточнить параметры шрифта для выбранной надписи. Каждому выбранному вами параметру соответствует отдельная строка в окне Properties (Свойства), и ее значение будет изменено соответствующим образом.

6. Измените размер шрифта с 10 до 12 точек, а затем измените начертание с обычного на **курсив**. Чтобы подтвердить сделанные изменения, нажмите кнопку OK.

Visual Studio записывает изменения и в соответствии с ними изменяет значения свойств. Вы можете изучить изменения, сделанные Visual Studio, посмотрев форму в конструкторе Windows Forms или раскрыв категорию `Font` в окне Properties (Свойства).



Р и с. 1.9. Диалоговое окно Шрифт (Font)

7. В Solution Explorer (Обозревателе решений) щелкните на второй надписи объект `lblAnswer` (объект `lblAnswer` содержит надпись «На бас-гитаре!»).

Когда вы выберете этот объект, вокруг него появятся рамка и маркеры для изменения размера.

8. Снова щелкните в окне Properties (Свойства) на свойстве `Font`.

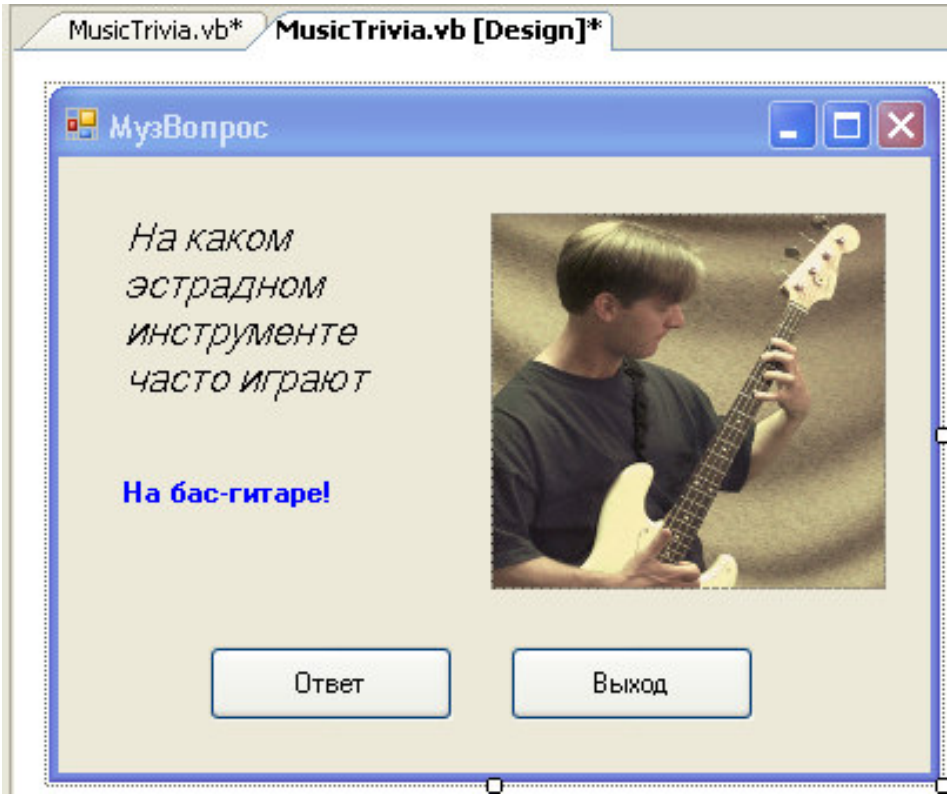
Объект `lblAnswer` содержит свой собственный уникальный набор свойств. Хотя названия свойств такие же, как и для объекта `lblQuestion`, значения свойств отличаются, поэтому вид и поведение объекта `lblAnswer` не зависят от остальных объектов формы.

9. Нажмите кнопку с многоточием свойства `Font`, установите стиль шрифта «жирный» и нажмите OK.

10. Прокрутите окно Properties (Свойства) до свойства `ForeColor`, а затем щелкните мышкой в левом столбце.

11. В правом столбце нажмите кнопку раскрытия списка, а в нем выберите закладку `Custom` (Польз.) и на ней синий цвет.

Текст надписи `lblAnswer` поменяется на жирный и станет синего цвета. Форма после изменения свойств показана на рис. 1.10.



Р и с. 1.10. Объекты формы после изменения свойств

1.9. РАЗМЫШЛЕНИЕ О СВОЙСТВАХ

В Visual Basic каждый элемент графического интерфейса программы (включая саму форму) имеет набор свойств, которыми можно управлять. На стадии проектирования вы можете задать их значения с помощью окна Properties (Свойства). К свойствам можно обращаться из кода и изменять их так, как это требуется по ходу работы программы. Те элементы интерфейса пользователя, которые предназначены для ввода текста, часто используют свойства для передачи введенной информации в программу. Поначалу концепция свойств может показаться сложной для понимания. Рассмотрите их на примере чего-нибудь из обычной жизни.

Например, велосипед – это объект, который используется для поездок из одного места в другое. Так как велосипед – это физический объект, у него есть качества, общие для объектов подобного рода: название модели, цвет, тормоза, колеса. По конструкции это может быть дорожный, гоночный или двухместный велосипед. В терминологии Visual Basic эти характеристики являются свойствами объекта «велосипед». Большинство свойств конкретного велосипеда определяются при его изготовлении, но некоторые (шины, скорость, возраст и дополнительные детали – отражатели или зеркала) могут изменяться в процессе его использования. При работе с Visual Basic вы встретитесь со свойствами обоих видов.

1.10. ПЕРЕМЕЩЕНИЕ И ИЗМЕНЕНИЕ РАЗМЕРОВ ОКОН ИНСТРУМЕНТОВ

Когда на экране одновременно видно много инструментов программирования, среда разработки Visual Studio выглядит очень перегруженной. Visual Studio позволяет перемещать, изменять размеры, закреплять и использовать функцию автоматического скрытия для большинства элементов интерфейса, которые используются для создания программ.

Чтобы передвинуть любое из окон инструментов Visual Studio, просто щелкните на его строке заголовка и перетащите объект в дру-

гое место. Если вы придвинете одно окно к краю другого окна, то оно закрепится. Преимущество закрепляемых окон в том, что они всегда видны (они не скрываются за другими окнами). Если вы хотите увеличить закрепляемое окно, просто раздвиньте его границу.

Чтобы полностью закрыть окно, нужно нажать кнопку **Close** (Закрыть) в верхнем правом углу этого окна. Вы всегда сможете открыть это окно снова, выбрав соответствующую команду в меню View (Вид).

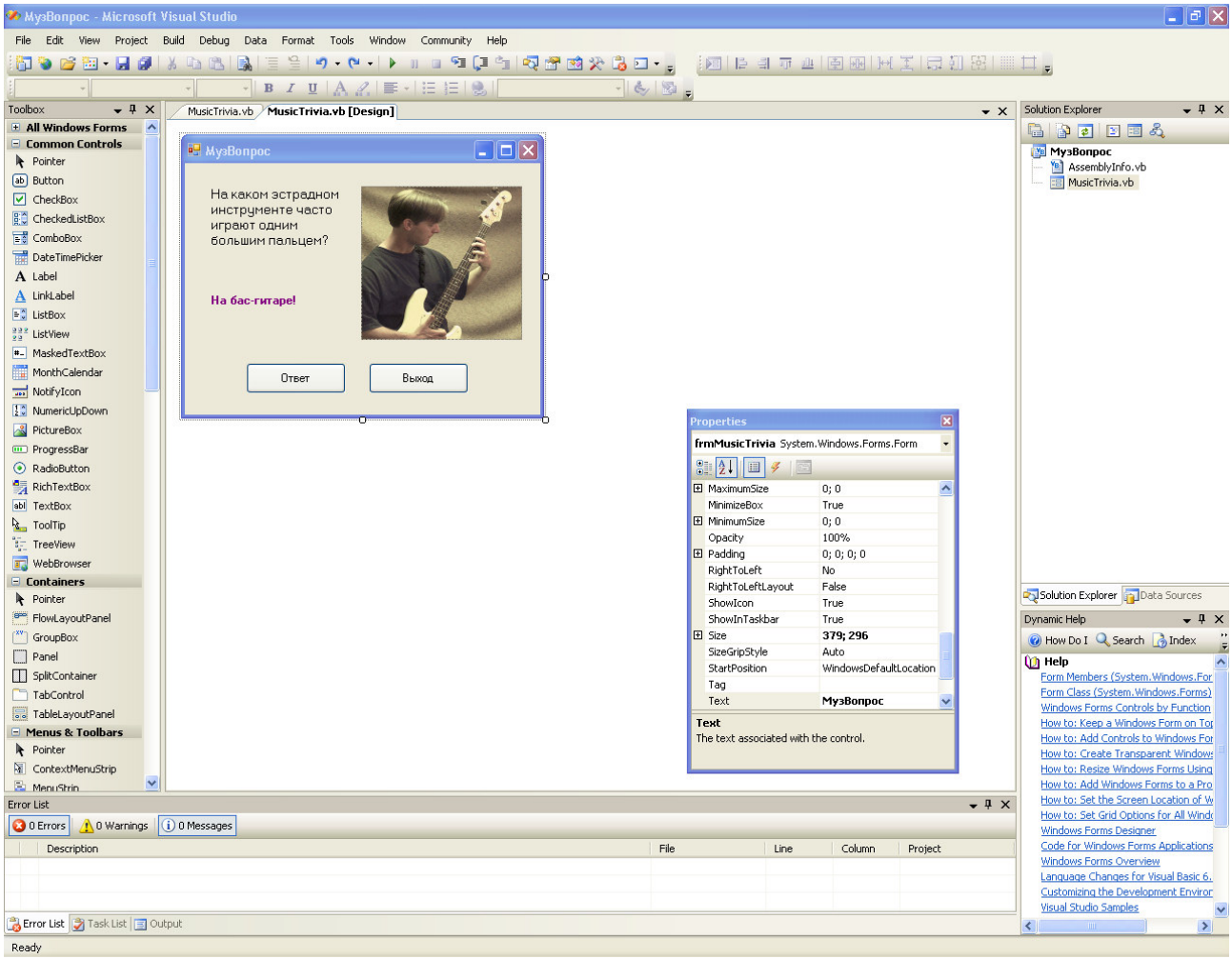
В среде разработки Visual Studio имеется функция автоскрытия окна инструмента, которую можно задействовать, щелкнув в правой части заголовка окна по канцелярской кнопке **Auto Hide (**Автоскрытие). Это действие убирает окно с его закрепленной позиции, а заголовок окна остается на краю среды разработки на закладке, которая не занимает много места. Выполняя автоскрытие окна, вы заметите, что окно инструмента будет оставаться видимым до тех пор, пока вы держите на нем указатель мыши. Как только мышь перемещается в другую часть среды разработки, это окно свернется и превратится в закладку.

Чтобы восстановить окно, для которого активизировано автоскрытие, нажмите на его закладку на краю среды разработки или некоторое время подержите над ней указатель мыши. Определить, включена ли функция автоскрытия для данного окна, можно по канцелярской кнопке в его правом углу. Удерживая указатель мыши над заголовком, вы можете использовать инструменты в режиме, который можно назвать «игрой в прятки». Другими словами, вы можете быстро вызвать скрытое окно, щелкнув на его закладке, проверить или задать требуемую информацию и затем передвинуть мышь, чтобы окно исчезло. Если вам нужно, чтобы окно было открыто постоянно, снова нажмите кнопку **Auto Hide** (Автоскрытие), чтобы острие канцелярской кнопки смотрело вниз  . Окно останется видимым.

Чтобы передвинуть или изменить размеры одного из окон инструментов Visual Studio, выполните нижеследующие действия. В этом упражнении использовано окно Properties (Свойства), но вы можете передвинуть любое окно по выбору.

1. Если окно Properties (Свойства) не отображается в среде разработки, нажмите кнопку Properties Window (Окно свойств) на стандартной панели инструментов. При этом окно появится, а его заголовок будет подсвечен.

2. Чтобы сделать окно Properties (Свойства) плавающим (незакрепленным), дважды щелкните мышью на заголовке окна. Окно будет выглядеть так, как показано на рис. 1.11.



Р и с. 1.11. Плавающее окно свойств (Properties)

3. Используя строку заголовка окна Properties (Свойства), перетащите его в другое место в среде разработки, но не позволяйте ему закрепиться. Возможность перемещать окна инструментов в среде разработки Visual Studio позволяет организовать ее удобным для вас образом. Теперь давайте изменим размеры окна Properties (Свойства), чтобы увидеть больше свойств объекта.

4. Передвиньте указатель мыши к нижнему правому углу окна Properties (Свойства) так, чтобы он принял форму стрелки для изменения размера.

Размеры окон Visual Studio можно изменить тем же способом, что и окна других приложений в операционной системе Microsoft Windows.

5. Чтобы увеличить окно Properties (Свойства), потяните нижний правый угол окна вниз и вправо. Теперь окно стало больше. Окно большего размера позволяет работать быстрее и легче находить нужную информацию. Вы можете свободно изменять размеры окна, если необходимо увидеть большую часть его содержимого.

1.11. ЗАКРЕПЛЕНИЕ ИНСТРУМЕНТА В VISUAL STUDIO

Если инструмент в среде разработки представлен плавающим окном, вы можете вернуть его в его первоначальное закрепленное положение, дважды щелкнув мышью на строке его заголовка. (Эта же техника применялась в последнем упражнении для раскрытия закрепленного окна: двойной щелчок на строке заголовка работает в ту или другую сторону между двумя стандартными положениями окна.) Когда окно находится в плавающем незакрепленном режиме, его можно закрепить в новом месте. Это может понадобиться, чтобы освободить место в Visual Studio для конкретной задачи программирования, например для работы над графическим интерфейсом в Windows Forms Designer (Конструкторе Windows Forms).

Чтобы закрепить окно Properties (свойства), выполните следующие действия.

1. Убедитесь, что окно Properties (Свойства) (или другой инструмент, который вы хотите передвинуть) представлено в среде разработки Visual Studio плавающим окном и не закреплено.

Если вы выполнили предыдущее упражнение, то окно Properties (Свойства) будет именно в таком виде.

2. Перетащите окно Properties (Свойства) за его строку заголовка к верхнему, нижнему, правому или левому краю среды разработки, пока граница окна не «прилипнет» к краю окна среды разработки.

Эта привязка указывает, что, когда вы отпустите кнопку мыши, окно будет закреплено в этом месте. Заметьте, что существует несколько допустимых мест закрепления окон инструментов Visual Studio, так что вы можете попробовать использовать два или три различных положения, пока не найдете то, которое устроит вас больше всего. (Окно должно быть расположено в удобном месте, но не мешать другим используемым окнам инструментов.)

3. Чтобы закрепить окно Properties (Свойства), отпустите кнопку мыши. Окно будет находиться в том месте, где вы его оставили.

Совет. Чтобы предотвратить закрепление при перетаскивании окна, нажмите и удерживайте при перетаскивании клавишу <Ctrl>. Если вы хотите, чтобы передвигаемое окно было совмещено с другим окном в виде закладки, перетащите это окно непосредственно на строку заголовка другого окна. Когда окна связаны друг с другом подобным образом, в нижней части их общего окна появится строка с закладками для каждого из них, и вы сможете переключаться между этими окнами, выбирая их по закладкам. Окна с закладками позволяют эффективно использовать пространство одного окна для двух и более задач. Например, часто объединяют в виде закладок окна Solution Explorer (Обозреватель решений) и Class View (Представление классов).

4. Попробуйте еще несколько раз закрепить окно Properties (Свойства) в различных местах, чтобы окончательно разобраться в том, как работают закрепление и закладки.

Возможно, некоторые из этих процедур с окнами сначала покажутся странными, но через некоторое время они станут для вас привычны. Вы сможете создать такое пространство окон, в котором хватит места для информации, необходимой для работы с более важными задачами в Конструкторе Windows Forms) и редакторе кода.

1.12. СКРЫТИЕ ИНСТРУМЕНТА В VISUAL STUDIO

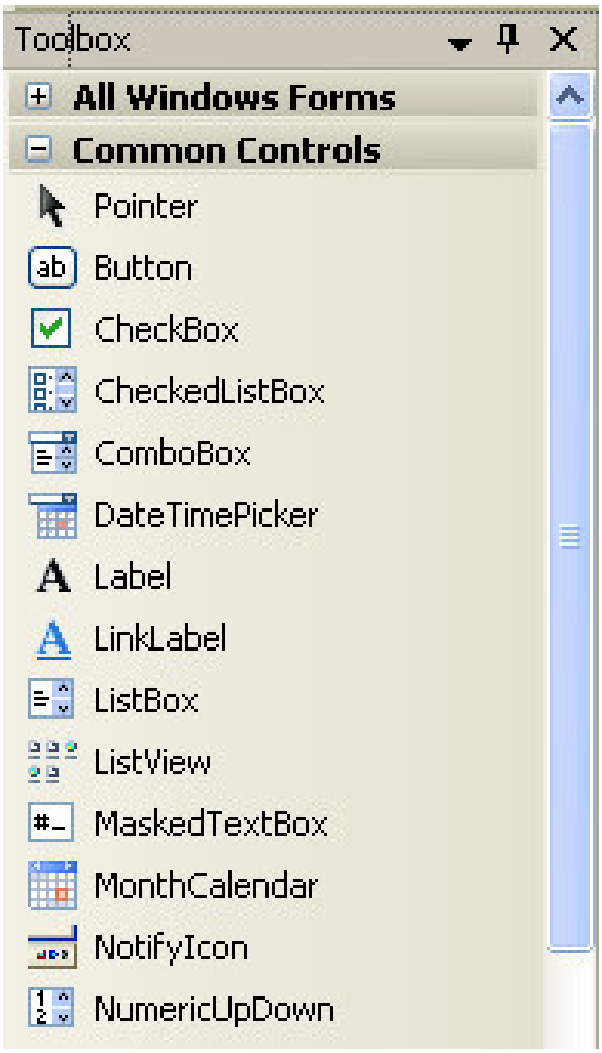
В Visual Studio имеется механизм для быстрого скрытия и отображения инструментов, который называется автоскрытие. Функция автоскрытия доступна для большинства окон инструментов. Чтобы скрыть окно инструмента, нажмите кнопку Auto Hide (Автоскрытие) в правой части заголовка окна, что приведет к сворачиванию его в закладку на краю среды разработки, а чтобы закрепить окно, нажмите кнопку автоскрытия еще раз. Те же действия можно выполнить с помощью команды Auto Hide (Автоскрытие) из меню Window (Окно). Заметьте, что функция и кнопка автоскрытия доступны только для закрепленных окон. Вы не увидите команды Auto Hide (Автоскрытие) или этой кнопки для активного окна, плавающего поверх среды разработки.

Чтобы использовать функцию автоскрытия, выполните следующие действия.

1. Найдите в среде разработки Toolbox (Область элементов) (ее окно обычно открыто в левой части Конструктора Windows Forms). В области элементов находятся элементы управления, которые можно использовать для создания приложений на Visual Basic. Например, для создания объектов в программе МузВопрос использованы элементы управления `Label` (надпись), `Button` (кнопка) и `PictureBox` (поле для изображения). В области элементов есть и другие наборы элементов управления, доступ к которым можно получить, нажав на закладки в ее окне.

Если область элементов не видна, выберите соответствующую строку в меню View (Вид). На рис. 1.12 показано, как она выглядит.

2. Найдите кнопку Auto Hide (Автоскрытие)  на заголовке окна Toolbox (Область элементов). Канцелярская кнопка на ней нарисована в вертикальном, или приколотом, положении, что означает, что область элементов «приколота» в открытом положении и автоскрытие отключено.



Р и с. 1.12. Окно ToolBox (панель элементов)

3. Нажмите кнопку автоскрытия в строке заголовка и удерживайте указатель мыши в окне области элементов. Кнопка автоскрытия теперь выглядит как канцелярская кнопка, которая смотрит влево . Это говорит о том, что область элементов больше не «приколота» в открытом положении, и будет работать автоскрытие. На левой стороне среды разработки появилась закладка с надписью Toolbox (Область элементов). Также обратите внимание, что Конструктор Windows Forms сдвинулся влево, однако если указатель мыши все еще находится в рамках окна области элементов, в самой области элементов ничего не изменится. Разработчики Visual Studio решили, что будет лучше, если окно с включенным автоскрытием не будет исчезать до тех пор, пока пользователь не передвинет указатель мыши в другую область среды разработки.

4. Передвиньте мышь из окна Toolbox (Область элементов). Как только вы это сделаете, Toolbox (Область элементов) скроется за пределы экрана и останется только маленькая закладка. (Над закладкой области элементов также видна закладка Server Explorer (Обозреватель серверов) – указание на то, что автоскрытие включено еще для одного окна инструмента. В зависимости от настроек Visual Studio могут быть видны и другие закладки для окон среды разработки, для которых включено автоскрытие.)

Преимущество автоскрытия для окон в том, что при этом они освобождают значительное место в рабочей области Visual Studio, оставаясь при этом доступными.

5. Подержите указатель мыши над закладкой Toolbox (Область элементов). (При желании можно щелкнуть на этой закладке.)

Ее окно немедленно выдвинется обратно в пределы видимости, и вы сможете использовать элементы управления для создания интерфейса пользователя.

6. Передвиньте указатель мыши за пределы области элементов, и инструмент снова исчезнет.

7. Наконец, снова раскройте область элементов, а затем нажмите кнопку автоскрытия в строке ее заголовка. Область элементов вернется в знакомое закрепленное положение, и вы сможете использовать ее постоянно.

Потратьте некоторое время на перемещение, изменение размеров, закрепление и автоскрытие окон инструментов Visual Studio, чтобы создать удобную для вас рабочую среду.

1.13. ПОЛУЧЕНИЕ СПРАВКИ

В комплект Visual Studio входит оперативное справочное руководство, в котором можно найти необходимые сведения о среде разработки Visual Studio, языке программирования Visual Basic, ресурсах Framework и остальных инструментах, входящих в набор Visual Studio. Прежде чем перейти к следующей лекции, в которой вы со-

здадите свою первую программу, потратьте некоторое время на изучение справочной информации.

Совет. Оперативная справка Visual Studio 2005 поставляется на нескольких компакт-дисках или DVD вместе с Visual Studio2005. Если у вас достаточно места на диске, вы можете установить всю документацию по Visual Studio 2005 на свою систему.

В этом разделе вы узнаете, как в Visual Studio 2005 пользоваться Dynamic Help (Динамической справкой). Этот инструмент, в зависимости от того, над чем вы работаете в данный момент, заранее подбирает справочные разделы и показывает их в своем окне. Для ограничения выбора материалов используется анализ контекста, и вы увидите только те из них, которые относятся к конкретному компилятору или инструменту. (Другими словами, в **Dynamic Help** (Динамической справке) не появятся заголовки, относящиеся к Visual C++ или Visual C#, если вы не работаете с этими инструментами.)

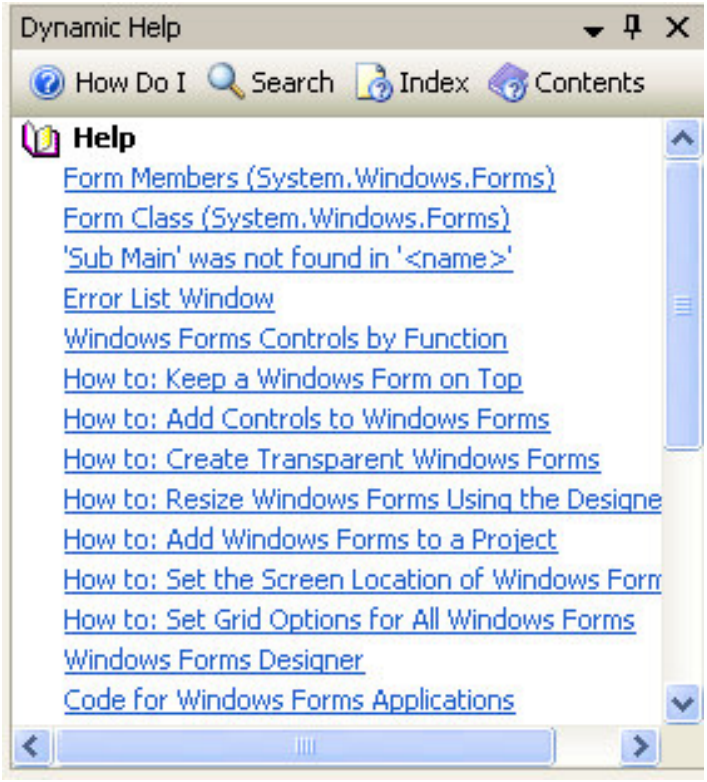
Здесь вы также узнаете, как выполнять полнотекстовый поиск по справочной системе Visual Studio. Полнотекстовый поиск полезен, когда нужно разыскать текст по отдельным ключевым словам.

Так как вы только что выполняли упражнение по автоскрытию области элементов, скорее всего система **Dynamic Help** (Динамической справки) подбирала для вас информацию об области элементов и других ваших действиях. Откройте **Dynamic Help** (Динамическую справку) и посмотрите.

1.14. ПОЛУЧЕНИЕ СПРАВКИ С ПОМОЩЬЮ DYNAMIC HELP

1. Щелкните на закладке Dynamic Help (Динамическая справка) в среде разработки или выберите строку Dynamic Help (Динамическая справка) в меню Help (Справка). Появится окно динамической справки (рис. 1.13).

Окно Dynamic Help (Динамическая справка) входит в состав Visual Studio. Его можно перемещать, изменять размер, закреплять или скрывать. Вы можете оставить его всегда открытым или открывать только тогда, когда оно нужно.



Р и с. 1.13. Окно Dynamic Help (Динамическая справка)

2. Выберите какой-нибудь пункт в окне динамической справки. При этом загружается справочная система, которая содержит справочную информацию по выбранному пункту справки.

Совет. Если вы не установили документацию по вашей системе, то в окне динамической справки никакие заголовки не появятся. Чтобы установить документацию, еще раз запустите программу установки Visual Studio 2005 и выберите пункт Product Documentation (Документация продукта).

3. Просмотрите несколько других статей из списка динамической справки. Обратите внимание, насколько эти темы соответствуют реальным действиям, которые вы выполняете.

Совет. Отбор и показ справочной информации можно настроить. В меню Tools (Сервис) выберите команду Options (Параметры) и в окне свойств раскройте папку Environment (Среда). Затем выберите пункт Dynamic Help (Динамическая справка) и определите, какие темы вы хотите видеть в справке и сколько ссылок будет показано одновременно. Эти настройки помогут вам управлять количеством информации, которую предоставляет справка.

Важно. Кроме Dynamic Help (Динамическая справка) существует множество других способов получить нужную информацию. Другие способы получения информации можно найти в разделе Help строки меню.

1.15. ВЫХОД ИЗ VISUAL STUDIO

1. Сохраните все изменения, которые вы сделали в вашей программе, щелкнув на кнопке SaveAll (Сохранить все) на стандартной панели инструментов  . Также эту операцию можно совершить из меню File, выбрав пункт меню SaveAll (Сохранить все).

2. В меню File (Файл) щелкните на команде Exit (Выход).

Среда разработки Visual Studio закроется.

1.16. КРАТКАЯ СПРАВКА ПО ТЕМЕ

В табл. 1.1 приведена краткая справка по теме.

Таблица 1.1

Краткая справка по теме

Чтобы	Сделайте следующее
Запустить Visual Studio 2005	На панели задач щелкните на кнопке Start (Пуск), укажите на Programs (Программы), укажите на папку Microsoft Visual Studio 2005, а затем щелкните на значке Microsoft Visual Studio 2005
Открыть готовый проект	Запустите Visual Studio 2005. В меню File (Файл) укажите на Open (Открыть), а затем щелкните на Project (Проект), или на Start Page (Начальной странице) щелкните на Open Project (Открыть проект)
Запустить программу	Щелкните на кнопке Start (Начать) стандартной панели инструментов или нажмите (F5)
Установить свойства	Щелкните на объекте формы, содержащем свойства, которые вы хотите установить, а затем щелкните на кнопке Properties Window (Окно свойств) стандартной панели инструментов, чтобы вывести на экран окно Properties (Свойства) (если оно не открыто)
Отобразить и изменить размеры окон инструментов	Дважды щелкните мышью на строке заголовка окна, чтобы перевести его в режим плавающего окна. Измените размер окна, перетащив его края
Переместить окна инструментов	Дважды щелкните мышью на строке заголовка окна, чтобы перевести его в режим плавающего окна, а затем перетащите строку заголовка

Окончание табл. 1.1

Чтобы	Сделайте следующее
Прикрепить окна инструментов	Дважды щелкните мышью на строке заголовка или перетащите инструмент к краю другого инструмента так, чтобы он прикрепился в нужном месте
Включить автосокрытие для окон инструментов	Щелкните на кнопке Auto Hide (Автосокрытие) в правой части строки заголовка окна инструмента. Окно инструмента скроется за небольшой закладкой на краю среды разработки до тех пор, пока вы не подержите над ней указатель мыши
Выключить автосокрытие для окон инструментов	Щелкните на закладке инструмента, а затем снова щелкните на кнопке Auto Hide (Автосокрытие)
Выйти из Visual Studio 2005	В меню File (Файл) выберите пункт Exit (Выход)

Контрольные вопросы

1. Какие языки программирования входят в состав Visual Studio?
2. Как запустить Visual Studio?
3. Как вызвать окно Start Page, если его не видно в среде разработки?
4. Какие окна являются основными в главном окне Visual Studio?
5. Для чего используется окно Start Page?
6. Для чего используется окно Solution Explorer?
7. Для чего используется окно Object Browser?
8. Какую информацию предоставляет окно Dynamic Help?
9. Как можно открыть существующий проект?
10. Что собой представляет файл решения?
11. Какое расширение имеет файл решения?
12. Что собой представляет файл проекта Visual Basic?
13. Какое расширение имеет файл проекта Visual Basic?
14. Какие инструменты являются основными в среде разработки?
15. Как открыть инструмент, если его не видно в среде разработки?
16. Перечислите специализированные инструменты в среде разработки Visual Studio.
17. Как можно открыть Windows Forms Designer?
18. Что нужно сделать, чтобы запустить программу в среде разработки Visual Studio?
19. Для чего используется нажатие клавиши <F5>?
20. Что делает Visual Studio при выборе команды Start Debugging?
21. Для чего используется окно Output?

22. Для чего предназначено окно Properties?
23. Какие способы существуют, чтобы запустить окно Properties?
24. Для чего используются кнопки **Alphabetical**  и **Categorized**  в окне свойств?
25. Где можно менять значения свойств элементов при создании интерфейса пользователя?
26. Как получить справку по нужному свойству?
27. Какими способами можно изменить размеры объектов на форме?
28. Можно ли изменять свойства с помощью окна Code Editor?
29. С помощью какого свойства меняется шрифт?
30. Где указаны названия свойств объектов?
31. Где указаны текущие настройки для каждого свойства?
32. Что означает знак (+) в левом столбце окна Properties?
33. Как изменить свойство объекта в окне Properties?
34. Для чего используется свойство ForeColor?
35. Могут ли свойства элементов графического интерфейса программы изменяться в процессе работы программы?
36. Что нужно сделать, чтобы окно инструмента стало плавающим (незакрепленным)?
37. Какую кнопку нужно нажать, чтобы окно инструмента оставалось постоянно открытым?
38. Для чего предназначена кнопка Auto Hide?
39. Что нужно сделать, чтобы инструмент в среде разработки, представленный плавающим окном, вернуть в первоначальное закрепленное положение?
40. Какую кнопку нужно нажать, чтобы свернуть окно закладки на краю среды разработки?
41. Как увеличить размер окна инструмента?
42. Какую клавишу нужно удерживать при перетаскивании окна инструмента, чтобы предотвратить закрепление?
43. Окно Dynamic Help (Динамическая справка) входит в состав Visual Studio?
44. Что нужно сделать для получения справки при работе в Code Editor (Редакторе кода)?
45. Что нужно сделать для получения справки о задаче, с которой вы сейчас работаете?
46. Что нужно сделать для получения справки по заголовку или действию?
47. Что нужно сделать, чтобы появилось окно динамической справки?
48. Где можно найти справку по интересующим вас разделам?
49. Как сохранить изменения в проекте?
50. Как выйти из среды проектирования?

2. СОЗДАНИЕ ПРОСТЕЙШИХ ПРОГРАММ

2.1. ЦЕЛЬ ЗАНЯТИЯ

Цели занятия:

– создать новый проект;

– создать пользовательский интерфейс для новой программы;

– ознакомиться с элементами управления **TextBox** (Текстовое поле), **PictureBox** (Поле рисунка) **DateTimePicker** (Поле выбора даты и времени), **LinkLabel** (Ссылка) и использовать в своих программах;

– настроить свойства для каждого объекта пользовательского интерфейса;

– создать код программы;

– сохранить и запустить программу;

– создать исполняемый файл;

– объявлять переменные типа **String** (строковые), **Integer** (целые числовые) с помощью инструкции **Dim** и использовать в своих программах;

– получить представление о понятиях **инструкция**, **элемент управления**, **объект**, **переменная**, **событие**, **процедура обработки события**, **свойство**, **метод**, **присваивание**.

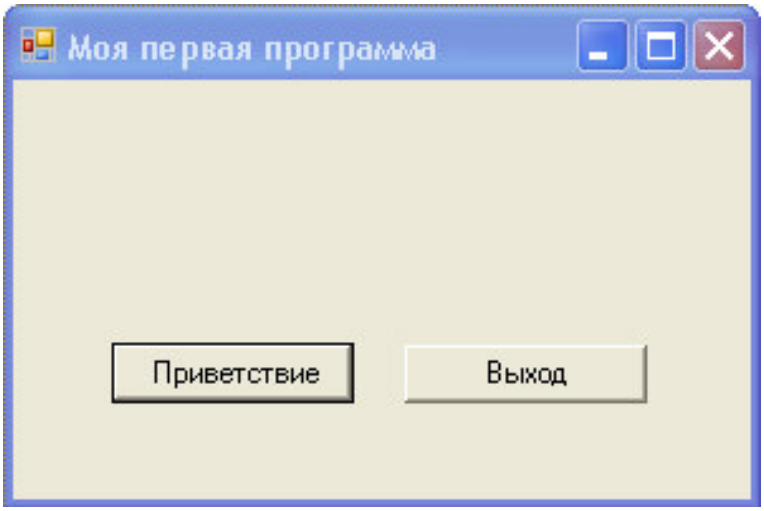
В среде Microsoft Visual Studio 2005 имеются мощные инструменты для запуска программ и работы с ними. В Visual Studio также содержится все необходимое для создания с нуля приложений для Microsoft Windows. Вы узнаете, как создать простой, но привлекательный графический интерфейс с помощью элементов управления из области элементов Visual Studio. Затем вы узнаете, как настраивать поведение этих элементов управления, изменяя их свойства, и напишете код, который определит, что будет делать ваша программа. Вы также узнаете, как сохранить и запустить вашу программу и как скомпилировать исполняемый файл.

2.2. ВАША ПЕРВАЯ ПРОГРАММА «ЗДРАВСТВУЙ, МИР!»

Использование программы под названием «Здравствуй, Мир!» («Hello, World!») стало традицией для книг по программированию. На примере программы «Здравствуй, Мир!» демонстрируется, как со-

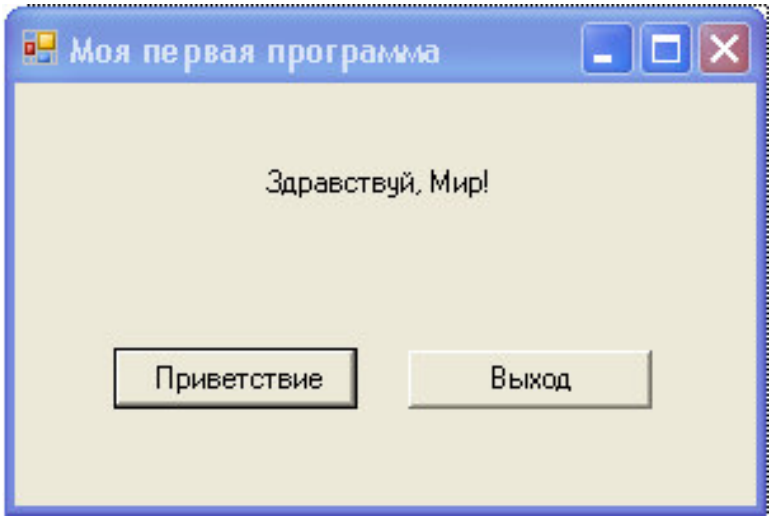
здать и запустить с помощью данного языка программирования простейшую программу.

После того, как пользователь запустит программу «Здравствуй, Мир!», на экране должна появиться форма, представленная на рис.2.1.



Р и с. 2.1. Программа «Здравствуй, Мир!» после запуска

Пользователь увидит форму с двумя кнопками. После нажатия на кнопку «Приветствие» на форме должна появиться надпись «Здравствуй, Мир!» (рис. 2.2).



Р и с. 2.2. Программа «Здравствуй, Мир!»
после нажатия на кнопку «Приветствие»

После нажатия на кнопку «Выход» программа должна завершить свою работу.

Создадим данную программу.

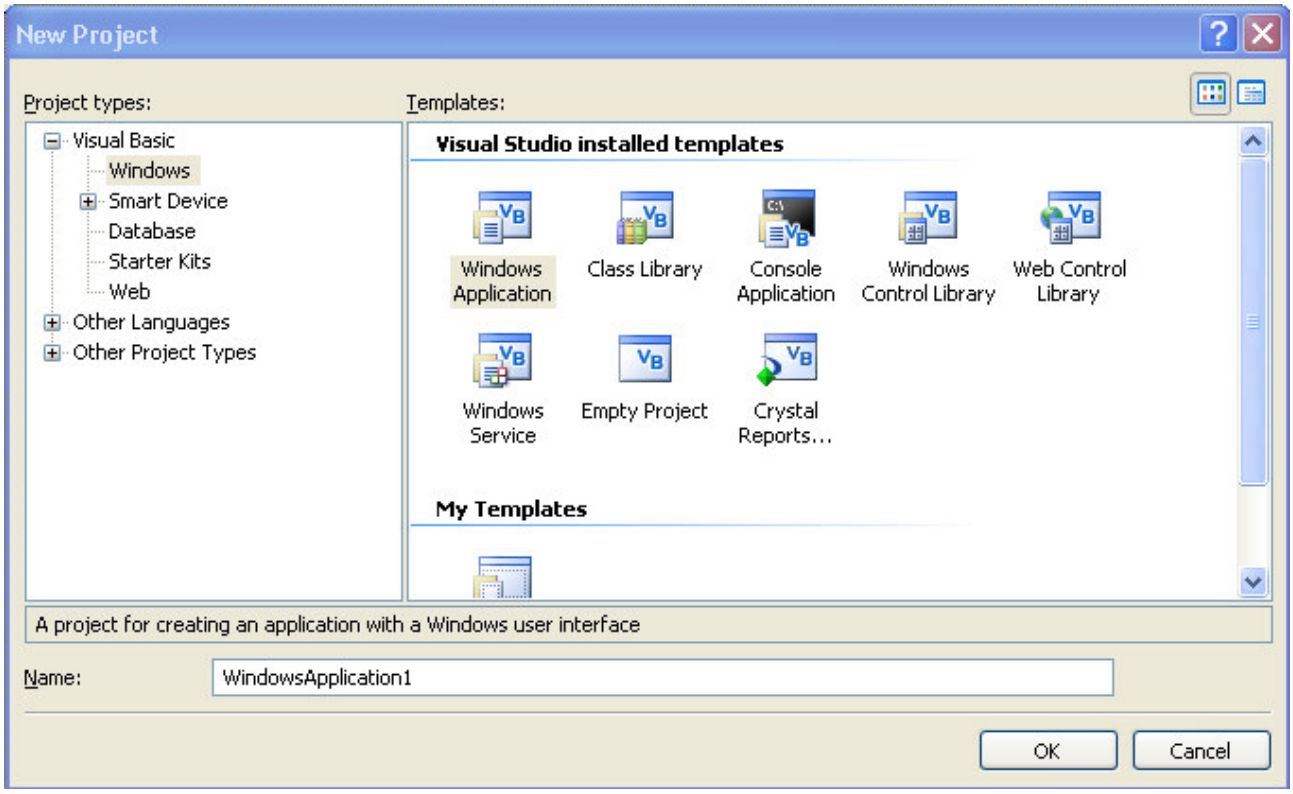
2.3. СОЗДАНИЕ НОВОГО ПРОЕКТА

1. Запустите Visual Studio.

2.На начальной странице (Start Page) Visual Studio нажмите на ссылку **Create Project** (Создать проект).

Совет. Новый проект можно начать и в меню **File** (Файл), выбрав в нем **New Project** (Новый проект).

Появится диалоговое окно **New Project** (Новый проект) (рис.2.3).



Р и с. 2.3. Диалоговое окно New Project (Новый проект)

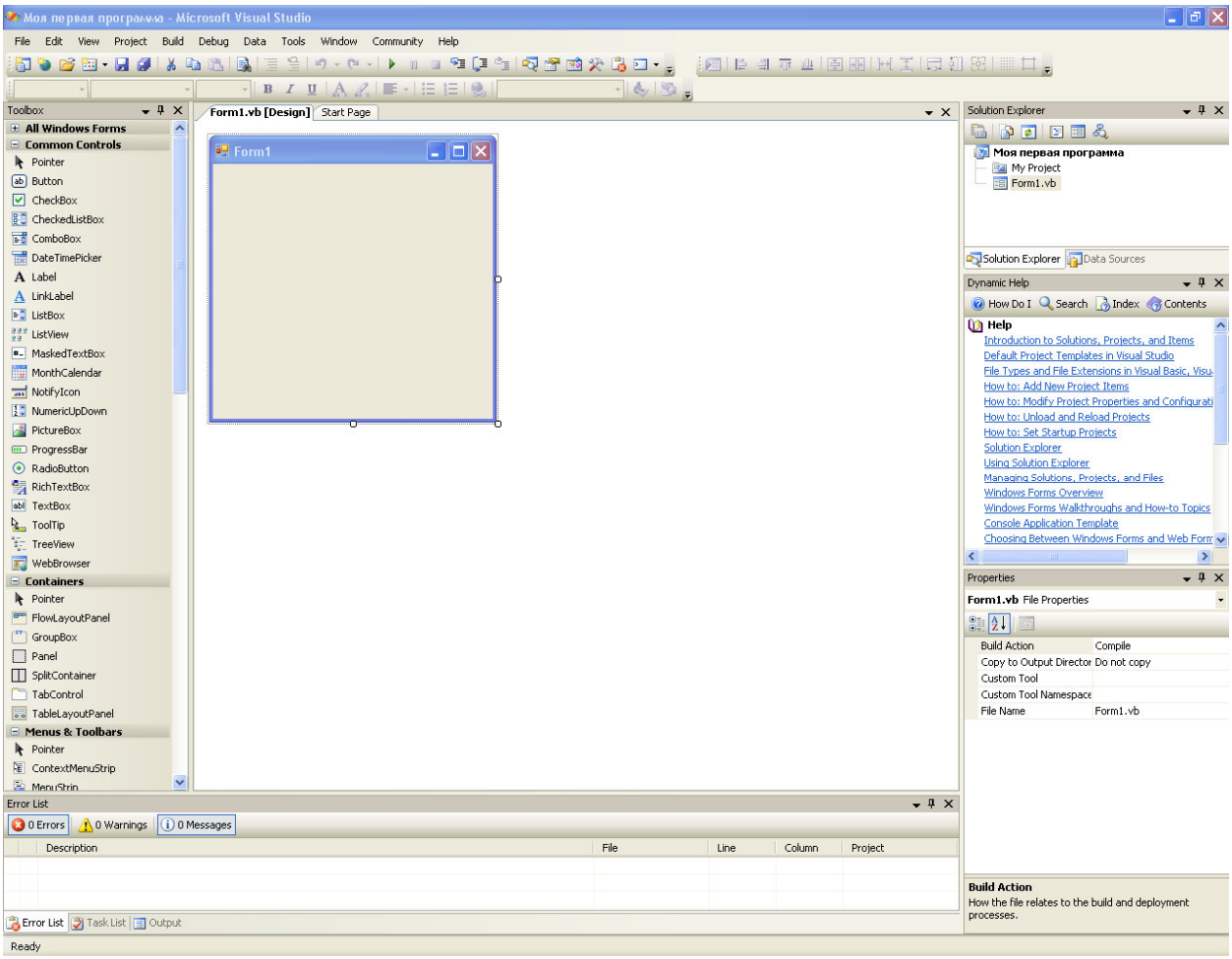
В Visual Studio можно создавать не только стандартные приложения для Windows, но и веб-приложения, и компоненты операционной системы. В диалоговом окне **New Project** (Новый проект) можно указать тип программы или компонента, который необходимо создать; язык, который будет использоваться; имя проекта и расположение файлов.

3. Убедитесь, что в списке **Project Types** (Типы проектов) выбрана ветка **Visual Basic – Windows**, а затем в окне **Templates** (Шаблоны) выберите **Windows Application** (Приложение для Windows). Visual Studio настроит среду разработки для программирования Windows-приложения на Visual Basic 2005.

4. В текстовом поле **Name** (Имя) введите «Моя первая программа» (без кавычек). Visual Studio присвоит вашей программе имя «Моя первая программа».

5. Нажмите OK, после чего новый проект будет создан.

Visual Studio отображает пустую графическую форму Windows (обычно имеющую название Form1), на основе которой можно создавать интерфейс для пользователей вашего приложения (рис. 2.4).



Р и с. 2.4. Новый проект в среде разработки

2.4. СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Теперь наша задача состоит в том, чтобы разместить элементы управления на форме, настроить форму и элементы управления с помощью окна Properties (окна Свойств).

Элементы управления находятся на **панели ToolBox** (панели элементов). Данная панель по умолчанию находится в левой части среды разработки (см. рис. 2.4).

Поместить элемент управления на форму можно несколькими способами.

Способ 1 (рисованием):

– найдите нужный элемент и наведите на него указателем мыши;

– нажмите указателем мыши на элемент управления;

– указатель мыши поместите над формой в том месте, где необходимо поместить элемент управления (указатель мыши примет форму перекрестия со значком кнопки);

– нажмите левую кнопку мыши, а затем потащите ее вниз и вправо до тех пор, пока не будет нарисован прямоугольник необходимого размера;

– остановите мышь и отпустите кнопку мыши (вместо прямоугольника будет нарисован элемент управления).

Способ 2 (двойным щелчком):

– найдите нужный элемент и наведите на него указателем мыши;

– дважды щелкните по нужному элементу управления мышью в области элементов, и новый объект с размерами по умолчанию будет добавлен на поле формы;

– указатель мыши наведите на помещенный элемент (указатель мыши примет форму двух перекрещивающихся стрелок);

– нажмите и не отпускайте левую клавишу мыши над элементом управления (на элементе управления появятся манипуляторы изменения размера);

– не отпуская левую клавишу мыши, перетащите элемент управления на нужное место формы;

– отпустите левую клавишу.

Способ 3 (с помощью буфера обмена):

– найдите нужный элемент и наведите на него указателем мыши;

– нажмите правой клавишей мыши на нужном элементе управления (на экране появится контекстное меню);

– в контекстном меню выберите пункт Copy (Копировать);

– переместите указатель мыши на нужное место формы;

– нажмите правую клавишу мыши (появится контекстное меню);

– из контекстного меню выберите пункт Paste (Вставить) (на форме в указанном месте появится элемент с размерами по умолчанию).

Способ 4 (перетаскиванием):

– найдите нужный элемент и наведите на него указателем мыши;

– нажмите и не отпускайте левую клавишу мыши над элементом управления;

– не отпуская левую клавишу мыши, перетащите элемент управления на нужное место формы;

– отпустите левую клавишу.

2.4.1. РАЗМЕЩЕНИЕ ОБЪЕКТОВ НА ФОРМЕ

1. В **ToolBox** (Панели элементов управления) найдите элемент управления **Button** (Кнопка) 

.

2. Поместите элемент управления **Button** (Кнопка) на форму одним из вышеуказанных способов.

На форме появится кнопка с маркерами изменения размера. Эта кнопка называется **Button1**, и это первая кнопка в программе.

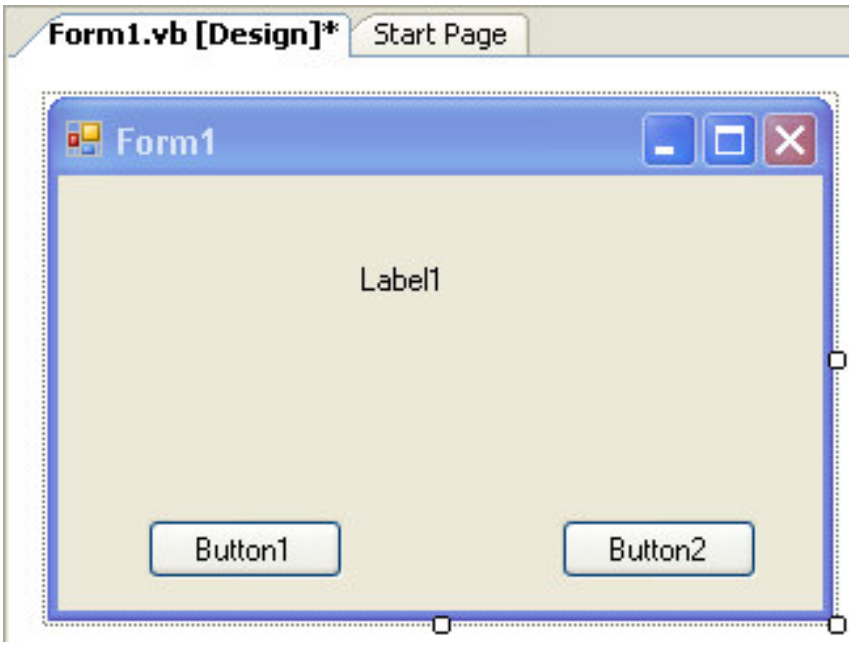
3. Поместите вторую кнопку на форму.

Надпись на кнопке будет **Button2**.

4. На панели элементов найдите элемент управления **Label** (Надпись).

5. Поместите элемент **Label** (Надпись) на форму.

Окно формы должно принять вид, показанный на рис. 2.5.



Р и с. 2.5. Форма с двумя кнопками

Когда Visual Basic находится в режиме разработки (т.е. тогда, когда активна среда программирования Visual Studio), элементы управления можно передвигать по полю формы с помощью мыши и изменять их размер при помощи маркеров изменения размера. Однако когда программа запущена и выполняется, пользователь не сможет перемещать элементы интерфейса, если только эта возможность не задана в специальном свойстве.

2.4.2. НАСТРОЙКА СВОЙСТВ ОБЪЕКТА

Изменять параметры объектов будем с помощью окна **Properties** (окна свойств). Начнем с формы. Если ваша форма активна, на ней должны быть видны манипуляторы изменения размера.

1. Указателем мыши нажмите на форму. Тем самым вы сделаете форму активной.

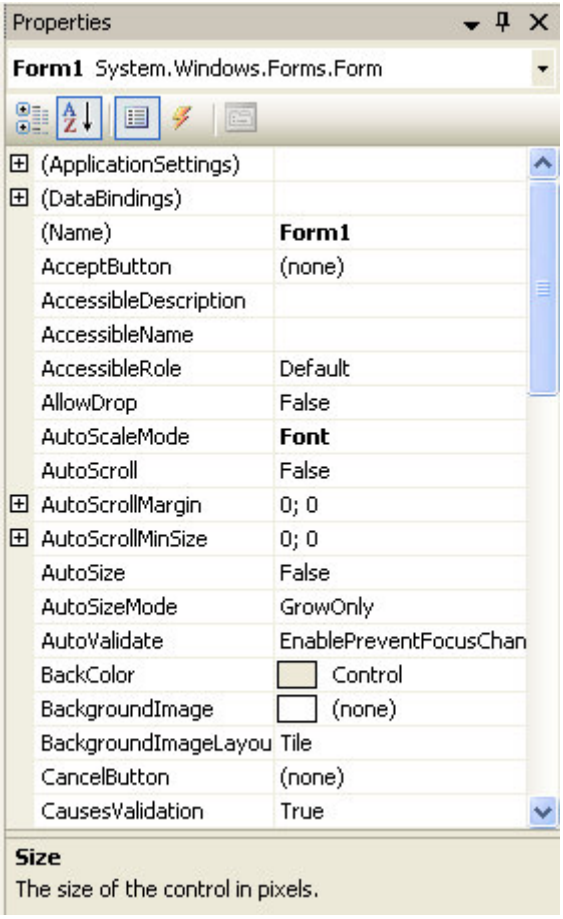
Признаком активности формы, как и любого другого объекта формы, является наличие на объекте манипуляторов изменения размера.

2. Выделите окно свойств. Если окно свойств не включено, нажмите на клавиатуре кнопку <F4>.

Когда активна форма, окно свойств будет выглядеть, как на рис. 2.6. В окне содержится список настроек (свойств) формы. Так как свойств очень много, Visual Studio организует их в категории и показывает в виде структуры. Если вы хотите увидеть свойства из какой-то категории, нажмите знак «плюс» (+) рядом с названием категории. Как работать с окном свойств было рассказано в первом занятии.

3. В окне свойств в левой колонке найдите свойство **Name**. Если у вас включена сортировка по алфавиту, то это свойство будет третьим в списке. Свойству **Name** формы по умолчанию присвоено имя `Form1`.

Свойство **Name** – одно из важнейших для элементов управления, в том числе и для формы. С помощью этого свойства идет обращение к любому из объектов.



Р и с. 2.6. Окно свойств

4. Измените свойство **Name** с **Form1** на **frmFirstProgram**. Первые три буквы в новом имени формы **frm** – это префикс, а **FirstProgram** (с англ. «Первая программа») – название.

Имя для элементов управления обычно начинается с **префикса**, состоящего из трех букв, и собственно имени объекта. **Префикс** – это сокращенное название элемента.

Представьте, что в вашем проекте много форм (с именами Form1, Form2, Form3, Form4 и т.д.), кнопок (с именами Button1, Button2, Button3, Button4 и т.д.), надписей (с именами Label1, Label2, Label3, label4 и т.д.) и других элементов с практически одинаковыми именами. Когда таких элементов много, то не мудрено запутаться, какой объект за что отвечает, поэтому прежде чем писать код программы, **настоятельно рекомендуется** давать осмысленные имена объектам формы, в том числе и самим формам.

В нашем примере мы дали осмысленное имя с наличием префикса (**frm**), которое явно говорит, что этот объект является формой.

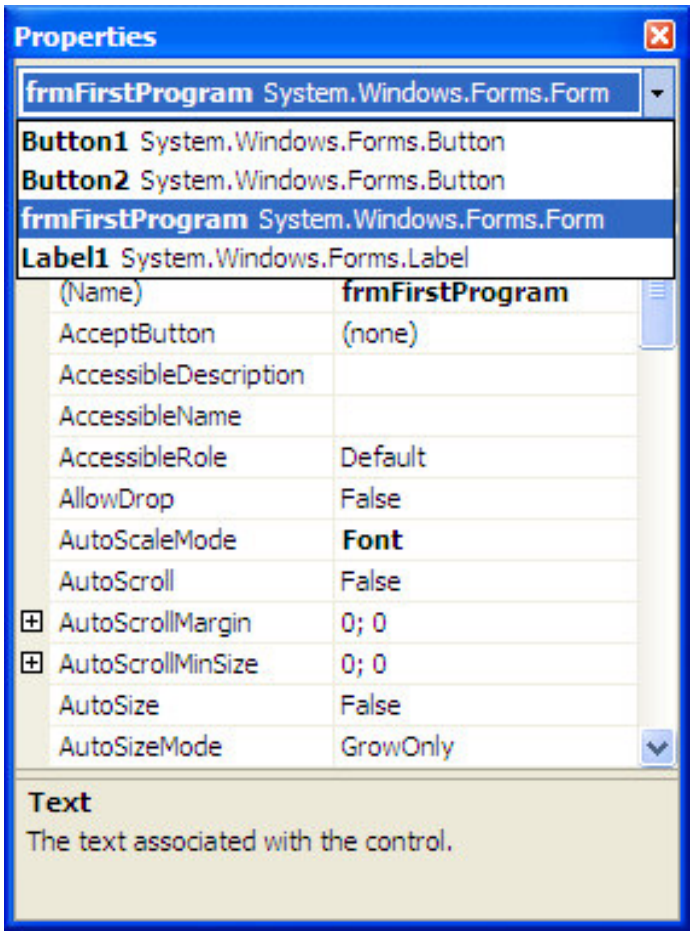
Использование префикса элементов в программах улучшает читаемость кода программы (понятно, с каким типом элемента управления (объектом) идет работа в данный момент), что в свою очередь является хорошим стилем программирования. Есть так называемое «**венгерское соглашение»,** которое унифицирует сокращение для всех объектов. Для других объектов сокращения будут даны дальше, по мере их появления в наших программах.

5. Прокрутите список окна Properties (Свойства) до тех пор, пока не увидите свойство **Text**.

6. Измените свойство **Text** с «**Form1**» на «**Моя первая программа»**

В заголовке формы вместо безликого **«Form1»**, появится надпись **«Моя первая программа»**. Обратите внимание, что практически в любой программе заголовок формы носит осмысленное значение, говорящее о том, для чего предназначена данная форма.

7. В верхней части окна Properties (Свойства) раскройте список объектов. Появится список объектов интерфейса программы (рис.2.7).



Р и с. 2.7. Выбор объекта из раскрывающегося списка

8. В этом списке выберите **Button1 System.Windows.Forms. Button** (это первая кнопка). В окне **Properties** (Свойства) появятся значения свойств для первой кнопки, а Visual Studio выделит объект `Button1` на форме. Объект можно также выбрать, нажав указателем мыши на объект на форме. В окне **Properties** (Свойства) также появятся значения свойств для первой кнопки.

9. Измените свойство Name кнопки с **Button1** на **btnWelcome**. Здесь **btn – префикс элемента управления кнопки**, Welcome – приветствие (с англ.).

10. В окне свойств найдите свойство кнопки **Text**. Измените это свойство с «**Button1**» на «**Приветствие**». На кнопке появится надпись «Приветствие».

11. Выберите вторую кнопку на форме.

12. С помощью свойства Name задайте имя второй кнопки **btnExit**. (Exit по англ. Выход).

13. Поменяйте надпись на кнопке с помощью свойства Text. Надпись на кнопке: «**Выход**».

14. Выберите элемент управления (объект) **Label1**.

Объект **Label** (Надпись) предназначен для размещения текста, чисел. Текст в объект Label (надпись) можно поместить как во время разработки интерфейса программы, так и во время выполнения программы.

15. Измените свойство Name объекта надпись с **Label1** на **lblMessage**. Здесь **lbl – префикс элемента управления (объекта) Label**, Message – сообщение (с англ.)

16. Надпись на объекте Label оставьте без изменения, т.е. «**Label1**».

После нажатия кнопки «Приветствие» в надписи **lblMessage** должен появиться текст «Здравствуй, Мир!». Было бы неплохо совсем убрать текст «**Label1**» из объекта **lblMessage**. Ведь при запуске программы данный текст останется на надписи, и это можно считать некрасивым, так как для пользователя непонятна надпись «Label1». Но если мы очистим свойство Text, то объект **lblMessage** станет невидимым на форме в дизайнере форм, а это неудобно программисту. Объект

будет существовать на форме, но его будет трудно найти, поэтому мы оставили свойство Text без изменения. Свойство Text будет очищено с помощь кода программы. Когда пользователь запустит программу, свойство Text будет очищено во время появления формы на экран. Пользователь даже не заметит, что надпись «Label1» исчезнет.

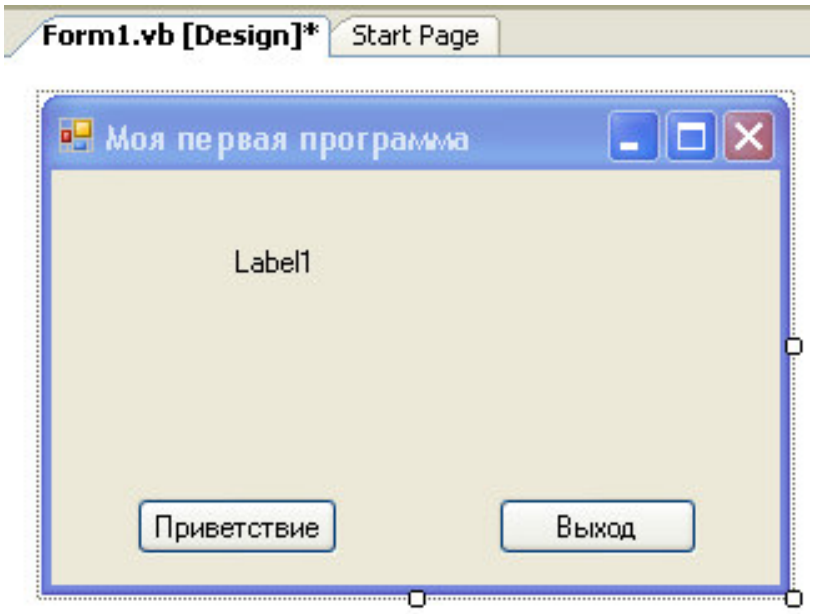
В этом примере мы шаг за шагом задали свойства для программы «Здравствуй, Мир!». В последующих занятиях инструкции для настройки свойств будут представлены в форме таблиц, за исключением случаев, когда их установка потребует особых действий. Свойства, которые вы задали для программы «Здравствуй, Мир!», представлены в табл.2.1.

Таблица 2.1

Таблица значений свойств программы «Здравствуй, Мир!»

Объект	Свойство	Значение
Form1	Name	frmFirstProgram
	Text	«Моя первая программа»
Button1	Name	btnWelcome
	Text	«Приветствие»
Button2	Name	btnExit
	Text	«Выход»
Label1	Name	lblMessage

Вид формы после изменения свойств ее объектов приведён на рис. 2.8.



Р и с. 2.8. Форма после изменения свойств

2.5. РАБОТА В РЕДАКТОРЕ КОДА

1. В поле формы дважды щелкните мышью на кнопке **Выход** (btnExit). В центральном окне среды разработки Visual Studio появится редактор кода, как показано на рис. 2.9.

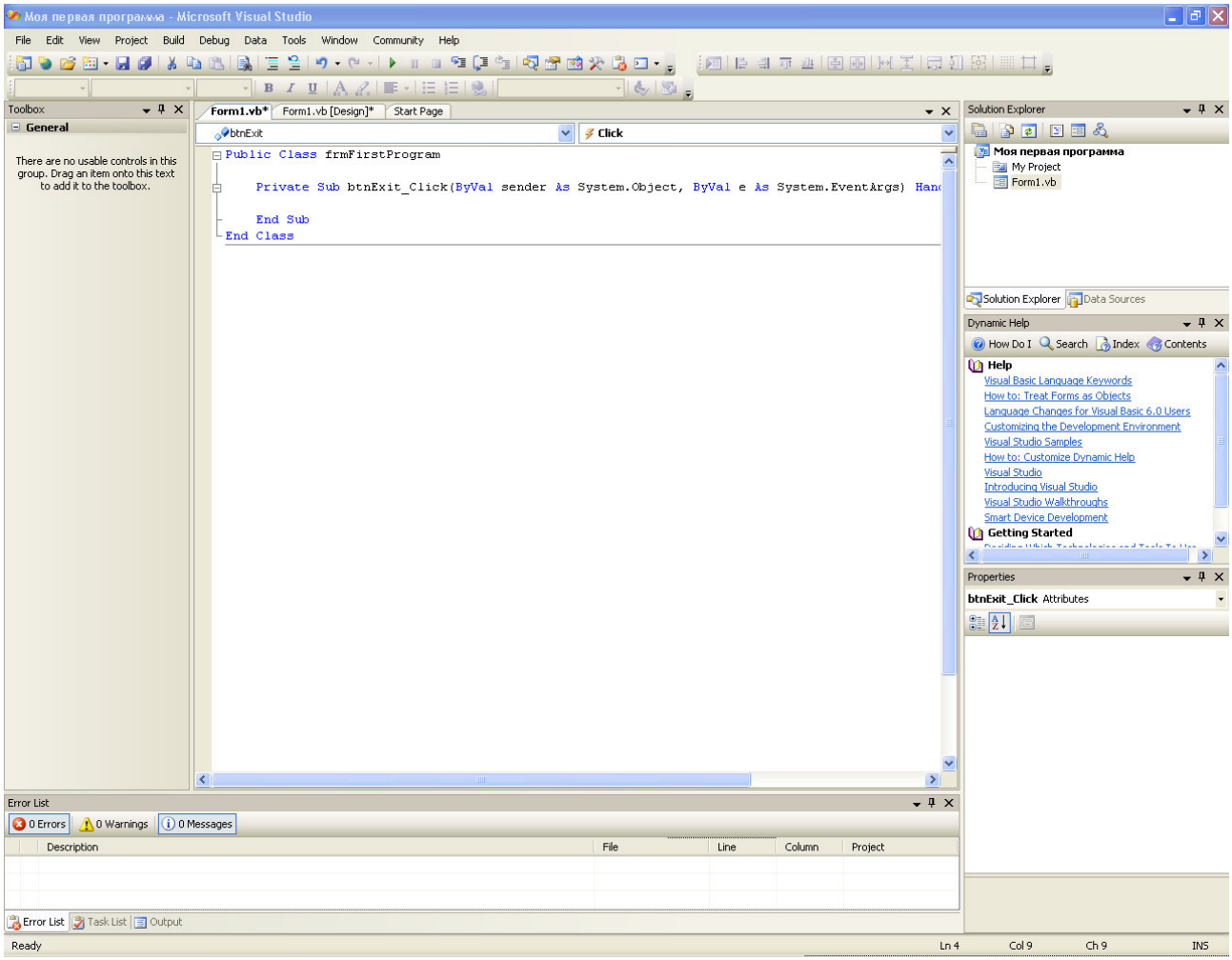


Рис. 2.9. Окно редактора кода

В окне редактора находится текст программы, связанный с текущей графической формой. Те выражения, которые вместе выполняют некоторое действие, обычно группируются в программные конструкции под названием **процедуры**. Обычным типом процедуры является **процедура Sub**, ее также называют **подпрограммой** (subroutine). **Процедуры Sub** начинаются с ключевого слова **Sub** в первой строке и заканчиваются с **End Sub**. Процедуры обычно выполняются при определенных обстоятельствах, например по нажатию кнопки на графическом интерфейсе. Когда процедура ассоциирована с конкрет-

ным объектом и событием, она называется **обработчиком события** или **процедурой события**. Когда по кнопке Выход (объект btnExit) был совершен двойной щелчок, среда Visual Studio автоматически добавила первую и последнюю строки процедуры для обработки события кнопки Выход. Строки процедуры показаны ниже.

```
    Private Sub btnExit_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnExit.Click

    End Sub
```

Тело процедуры находится между этими строками и выполняется всегда, когда пользователь совершает действие над элементом интерфейса, связанного с этой процедурой. В данном случае событие – это щелчок мышью по кнопке.

2. Наберите **End** внутри тела процедуры, а затем нажмите клавишу со стрелкой вниз. После того, как вы ввели команду, буквы стали синими. Это значит, что Visual Basic распознал введенный текст как допустимое выражение – **ключевое**, или **зарезервированное**, **слово** программы.

Инструкция (Statement) **End** используется для остановки программы и удаления с экрана ее формы. Язык программирования Visual Basic содержит большое количество зарезервированных слов и связанных с ними **инструкций** и символов. При написании кода программы очень важно написать их правильно с учетом пробелов, иначе они не будут распознаваться компилятором Visual Basic. При вводе ключевых слов и любой другой правке редактор кода автоматически выполняет действия по форматированию текста, добавляет отступы, пробелы и необходимые скобки.

Точное написание ключевых слов, их последовательность с учетом пробелов называется **синтаксисом выражения**.

При нажатии на клавишу «стрелка вниз» и переходе на другую строку выражение **End** было размещено так, чтобы оно было отделено от выражений **Private Sub** и **End Sub**. Такая схема форматирования является общепринятой в программировании и предназначена

для того, чтобы программы выглядели ясно и читаемо. Группа соглашений относительно того, как организуется код программы, часто называется **стилем программирования**.

3. Чтобы снова увидеть графическую форму, в окне Solution Explorer (Обозреватель решений) нажмите кнопку View Designer (Просмотреть конструктор). Когда виден редактор кода, форма, с которой вы работаете, не видна. Ее можно снова вывести на экран с помощью кнопки View Designer (Просмотреть конструктор). Если в Solution Explorer (Обозревателе решений) загружено более одной формы, выберите ту из них, которая вам нужна.

Совет. Чтобы снова показать окно с графической формой, можно выбрать закладку с текстом Form1.vb [Design] у верхнего края редактора кода.

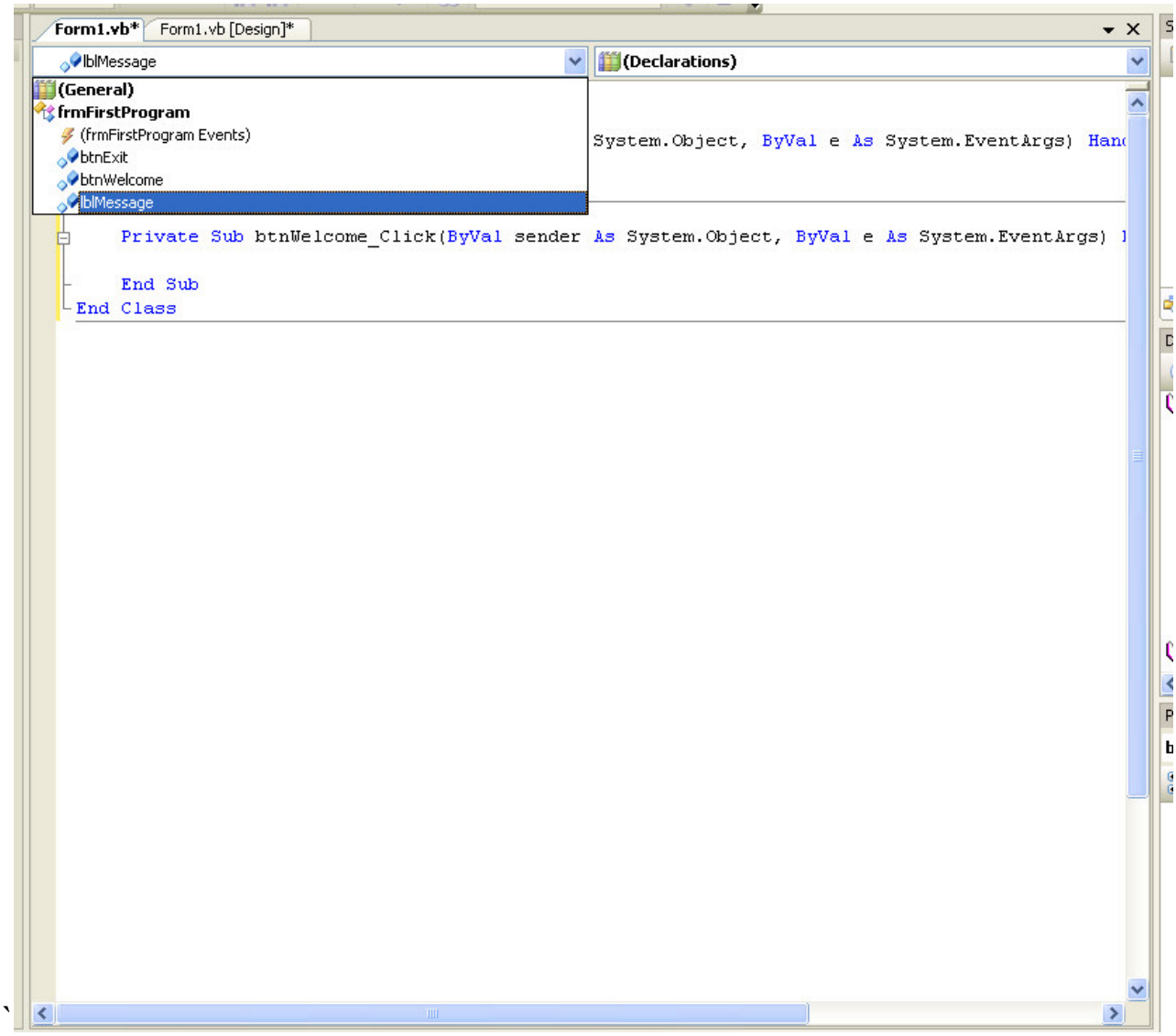
4. Дважды щелкните мышью на кнопке **Выход**. Через некоторое время появится редактор кода, и рядом с процедурой события **btnExit** появится процедура события, ассоциированная с кнопкой **btnWelcome**.

```
    Private Sub btnWelcome_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnWelcome.Click

    End Sub
```

Важно. Переключения между объектами формы и их событиями возможны с помощью редактора кода. В левой верхней части редактора есть выпадающее поле со списком существующих объектов. Названия объектов берутся из свойства **Name**. В нашем случае этих объектов всего четыре: форма, две кнопки и надпись (рис. 2.10).

Выбирая объект из списка, мы сообщаем редактору кода, с обработкой события какого объекта будем в данный момент работать.

В правой верхней части есть выпадающее поле со списком, в котором содержатся варианты событий для текущего элемента. Если мы в данный момент обрабатываем **событие** нажатия кнопки и при этом нажмем на кнопку выбора события, то появятся варианты событий именно для этой кнопки (рис. 2.11).



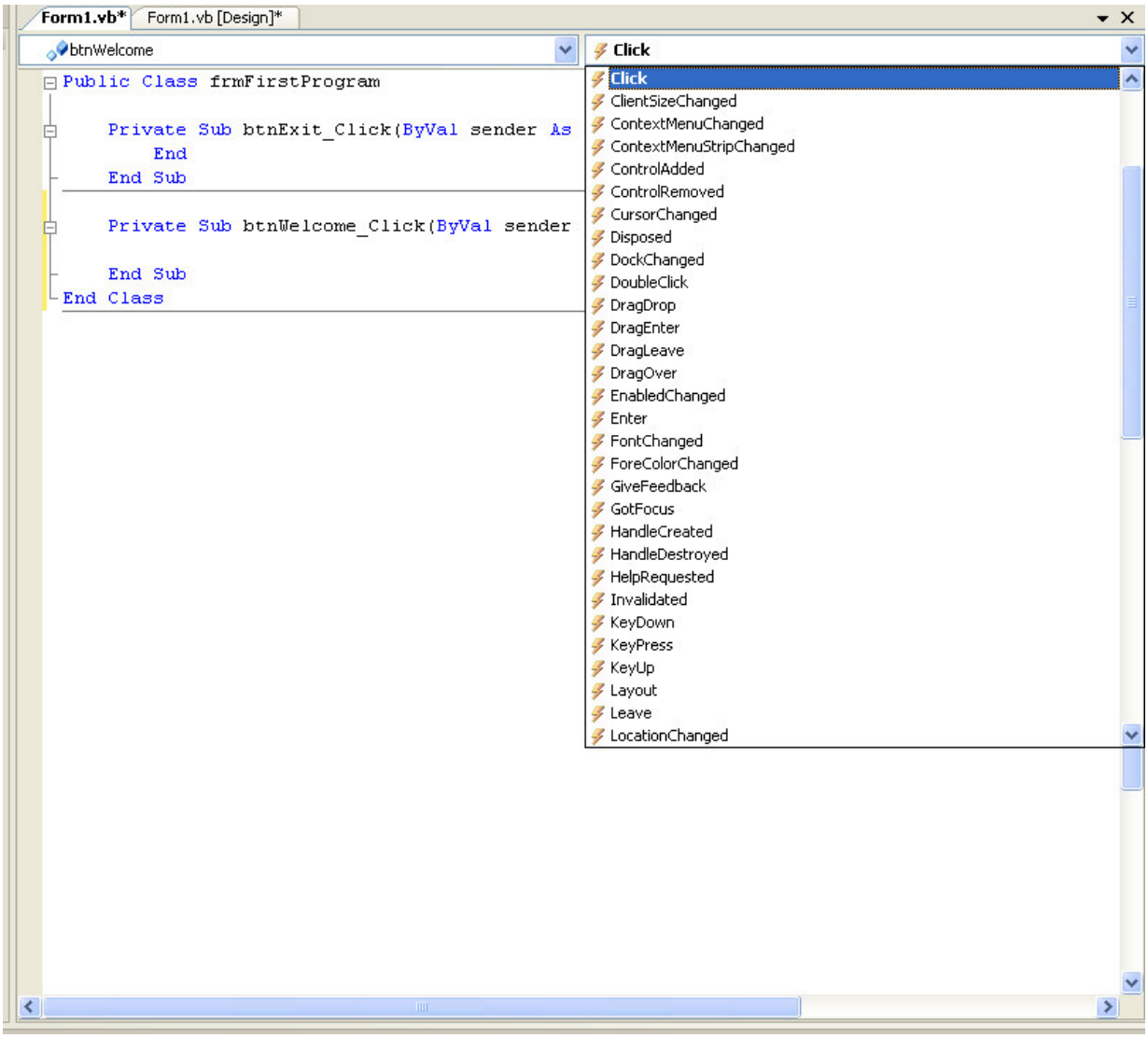
Р и с. 2.10. Окно кода – список объектов

Как видно из рисунка, кроме события нажатия курсором мыши на кнопку (событие **Click**), имеется множество других событий: двойное нажатие указателем мыши на кнопку (**DblClick**), перемещение указателем мыши над кнопкой (**MouseMove**) и др. Основным событием для кнопки, конечно же, является именно событие **Click**, которое мы и обрабатываем в нашей программе.

Для каждого элемента, в том числе и для формы, имеются свои возможные события, часто одинаковые, но есть и свои отличия.

Поэтому, используя редактор кода, пп. 3, 4 можно заменить следующим образом:

– не покидая редактор кода, нажмите на стрелку выпадающего списка, содержащего имена объектов (левый список);



Р и с. 2.11. Окно кода – список событий

– в раскрытом списке выберите объект **btnWelcome**, в результате чего в редакторе кода появится процедура обработки события:

```
    Private Sub btnWelcome_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnWelcome.Click

    End Sub
```

– если вам нужна другая процедура обработки события, в списке событий (правый список) выберите нужное вам событие, после чего в редакторе кода появится процедура обработки нужного вам события и нужного вам объекта.

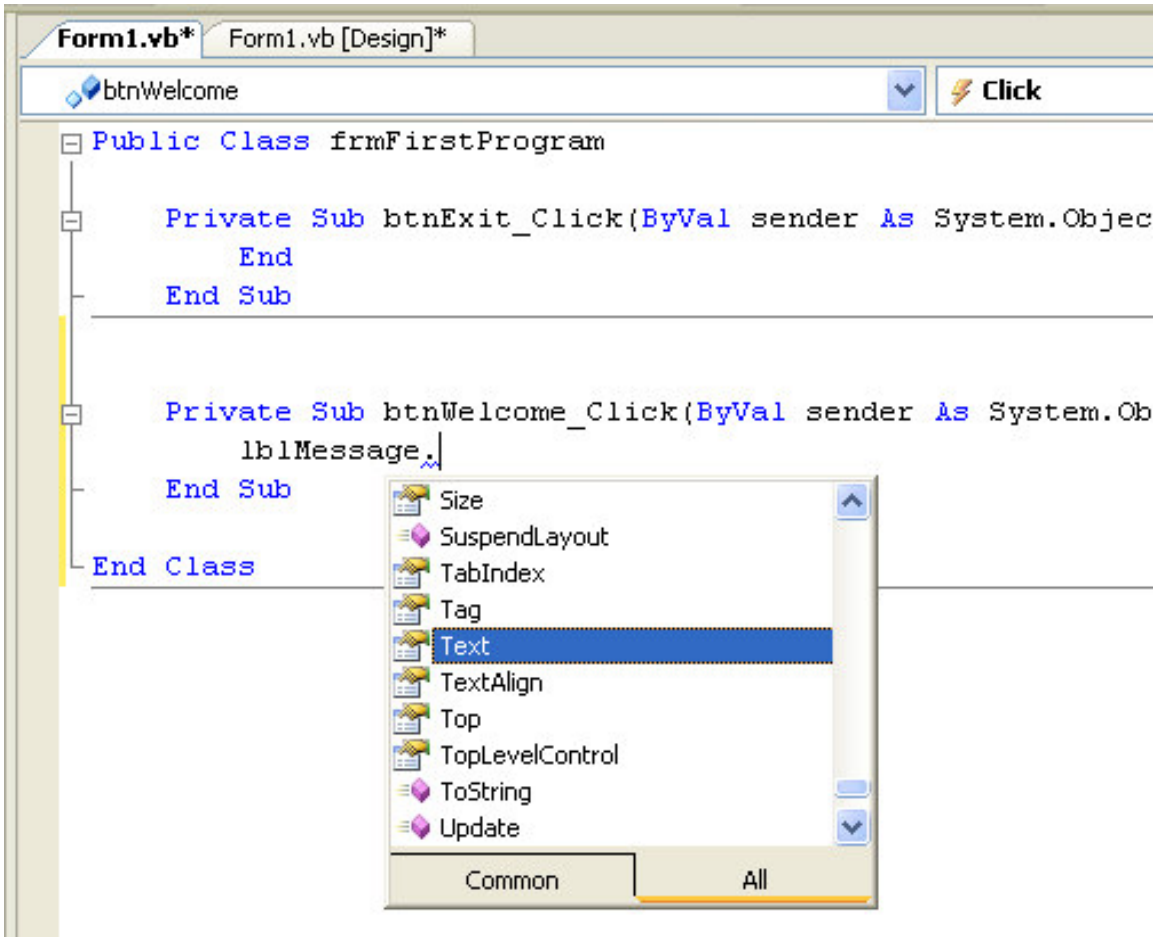
5. В теле процедуры обработки события нажатия на кнопку btn-Welcome наберите следующую строку:

```
lblMessage.Text = "Здравствуй, Мир!"
```

Этой строкой мы помещаем надпись «Здравствуй, Мир!» на объект **lblMessage**, расположенный на нашей форме. Так выглядит типичная команда Visual Basic. Читается она следующим образом. Свойству **Text** объекта **lblMessage** присваивается (знак «=») величина **"Здравствуй, Мир".** Эта величина имеет тип **String** (строка, текст), поэтому заключена в кавычки (""). Строковые данные всегда заключаются в кавычки. Знак («=») – это **оператор присвоения**. Величине, стоящей слева от знака «=» (оператора присвоения), присваивается величина, стоящая справа от знака «=» (оператора присвоения).

После выполнения команды присваивания свойство **Text объекта lblMessage** становится равным **"Здравствуй, Мир!"** .

Совет. Набирая текст программы, вы должны были обратить внимание на то, что после того, как вы ставите точку, появляется список, из которого можно выбрать мышкой или стрелками клавиатуры нужное **свойство** или **метод** (рис. 2.12).



Р и с. 2.12. Выбор свойства или метода

Это очень облегчает работу, поскольку не нужно набирать все буквы команды. Список возможных свойств и методов часто может оказаться очень большим, но чтобы отыскать нужное свойство или метод, не нужно листать весь список. Достаточно набрать начальные буквы нужного свойства или метода, как список отобразит свойства и методы, начинающиеся на буквы, которые вы набрали.

Также облегчает ввод команд **служба завершения слова**: можно не писать целиком слово **lblMessage**, а набрать часть команды, а затем нажать сочетание клавиш на клавиатуре <**Ctrl+Пробел**> (Знак + говорит, о том, что клавишу <**Ctrl**> нужно нажать и, удерживая ее, нажать <**Пробел**>). Редактор допишет слово до конца сам или предоставит возможность выбора слова, если его однозначно нельзя определить по первым буквам. Эта помощь позволяет сосредоточиться на решаемой программной задаче, а не на синтаксисе, и не нужно держать в памяти правильность написания каждой команды.

Для нашей программы можно поступить следующим образом:

– наберите внутри тела процедуры буквы «**lbl**» (без кавычек);

– и нажмите клавишу **Ctrl** и удерживая ее нажмите П**робел**.

Буквы **Message** будут дописаны автоматически.

На экране, соответствующем рис. 2.12, также видно, что одни предлагаемые варианты имеют слева значок  , а другие –  .

Значком  обозначаются **свойства** объекта, значком  – **методы** объекта. В нашем примере мы меняем свойство объекта.

6. В редакторе кода в левом раскрывающемся списке выберите строку **frmFirstProgram Events**. В правом списке появятся события, связанные с формой.

7. В правом раскрывающемся списке выберите событие Load. В редакторе кода появится процедура обработки события:

```
    Private Sub frmFirstProgram_Load(ByVal sender As Object,
ByVal e As System.EventArgs) Handles Me.Load

    End Sub
```

Эта процедура отвечает за событие, которое возникает после запуска программы, когда форма загружается в память компьютера и появляется на экране.

В этот момент (момент загрузки формы) программа должна очистить надпись **lblMessage** от содержимого – текста «Label1». Когда форма появится на экране, пользователь не увидит этого текста.

8. В тело процедуры **frmFirstProgram_Load** введите следующую строчку:

```
lbl
```

9. Нажмите сочетание клавиш **Ctrl+Пробел**. К буквам **lbl** добавится **Message**. Строка примет вид

```
lblMessage
```

10. Наберите знак точки.

11. Из списка предлагаемых свойств и методов выберите свойство **Text.** Строка примет вид

```
lblMessage.Text
```

12. После свойства **Text** наберите: « = ""». Строка должна принять вид

```
lblMessage.Text = ""
```

В этой строке свойству **Text** объекта **lblMessage** присваивается ничего не содержащая (пустая) строковая величина. Кавычки без пробела ("") говорят о том, что строковая величина пустая.

Текст нашей программы должен принять вид

```
Public Class frmFirstProgram

    Private Sub btnExit_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnExit.Click
        End
    End Sub


    Private Sub btnWelcome_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnWelcome.Click
```

```
        lblMessage.Text = "Здравствуй, Мир!"
    End Sub

    Private Sub frmFirstProgram_Load(ByVal sender As Object, ByVal
e As System.EventArgs) Handles Me.Load
        lblMessage.Text = ""
    End Sub

End Class
```

Добавим **комментарии** к тексту нашей программы, чтобы оставить понятные записи о том, что сделано. **Комментарии** – это пояснения, включаемые в программный код после апострофа ('). Программисты используют комментарии для описания того, что делает программа. Visual Basic при запуске программы не обрабатывает эти пояснения (на экране они показаны зеленым цветом); они существуют только для документирования.

13. Добавьте к коду программы комментарии таким образом, чтобы текст программы принял следующий вид:

```
Public Class frmFirstProgram

    'Процедура обработки события нажатия на кнопку Выход
    Private Sub btnExit_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnExit.Click
        End 'Инструкция завершающая работу программы
    End Sub

    'Процедура обработки события нажатия на кнопку Приветствие
    Private Sub btnWelcome_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnWelcome.Click
        lblMessage.Text = "Здравствуй, Мир!" 'помещение надписи на
объект lblMessage
    End Sub

    'Процедура обработки события загрузки формы
    Private Sub frmFirstProgram_Load(ByVal sender As Object, ByVal
e As System.EventArgs) Handles Me.Load
        lblMessage.Text = "" 'Очистка содержимого надписи
lblMessage
    End Sub

End Class
```

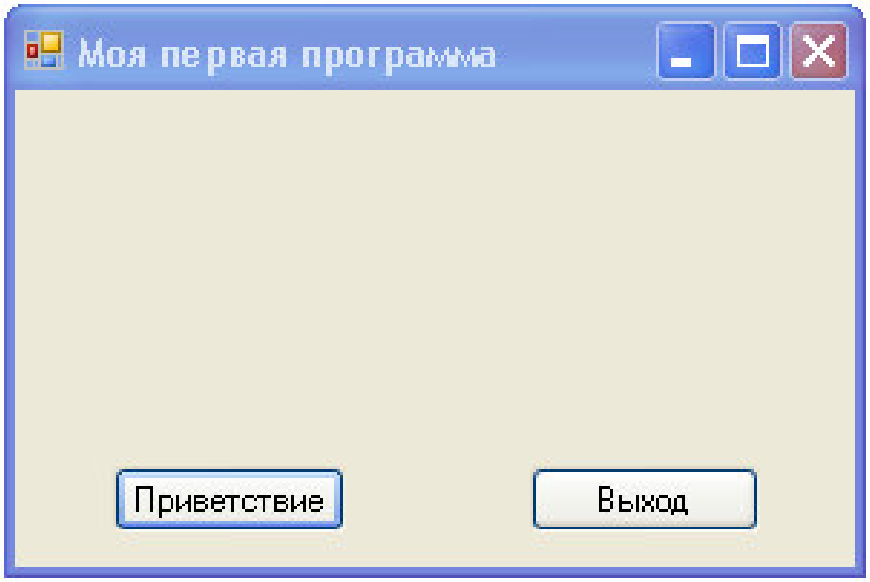
Это окончательный текст нашей программы.

Видно, что текст программы состоит из трех процедур обработки событий: нажатия (щелканье) на кнопке btnExit, нажатия на кнопке btnWelcome, загрузки формы frmFirstProgram. Данные процедуры работают не последовательно, а в момент, когда совершается одно из событий.

2.6. ЗАПУСК ПРОГРАММЫ «ЗДРАВСТВУЙ, МИР!»

1. На стандартной панели инструментов нажмите кнопку **Start Debugging** (Начать Отладку). Программа «Здравствуй, Мир!» будет скомпилирована и запустится в среде разработки Visual Studio.

Программа очистит надпись lblMessage, и на экране появится форма (рис. 2.13).



Р и с. 2.13. Программа после запуска

2. Нажмите кнопку **Приветствие**. Программа отобразит на форме приветствие "Здравствуй, Мир!".

Когда вы нажали **Приветствие**, код программы изменил свойство **Text** пустой надписи **lblMessage** на текст "Здравствуй, Мир!" и показал этот текст в надписи на форме. Если вы не получили этого результата, проверьте, все ли вы сделали правильно. Возможно, вы неправильно задали свойство или сделали опечатку при вводе кода программы. (Синтаксические ошибки подчеркиваются в редакторе кода волнистой линией.)

3. Чтобы остановить программу «Здравствуй, Мир!», щелкните на кнопке **Выход**.

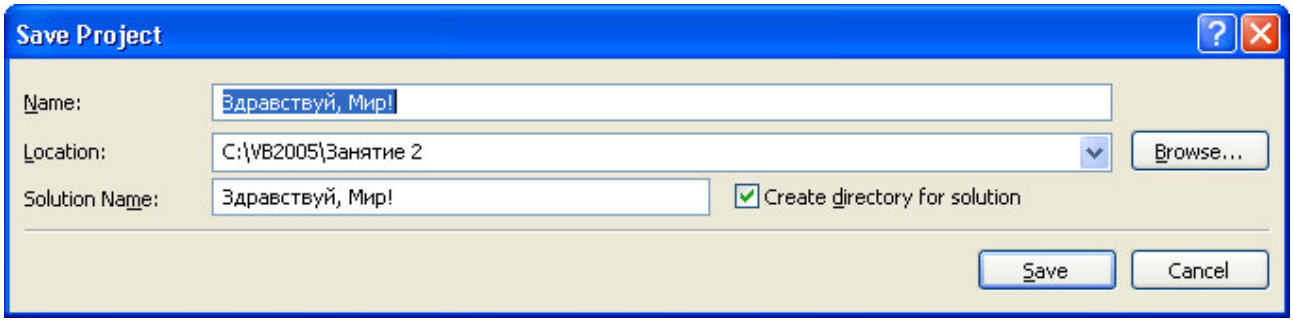
Программа выполнит команду завершения работы. Форма закроется и закончит свою работу.

Совет. Чтобы остановить программу, работающую в Visual Studio, можно также нажать кнопку **Stop Debugging** (Остановить отладку) на панели инструментов **Debug** (Отладка) среды Visual Studio, что приведет к закрытию этой программы.

2.7. СОХРАНЕНИЕ ПРОЕКТА

1. Нажмите кнопку **Save All** (сохранить все) на стандартной панели инструментов.

Появится диалоговое окно **Save Project** (Схранение проекта) (рис.2.14).



Р и с. 2.14. Диалоговое окно Сохранения проекта (Save Project)

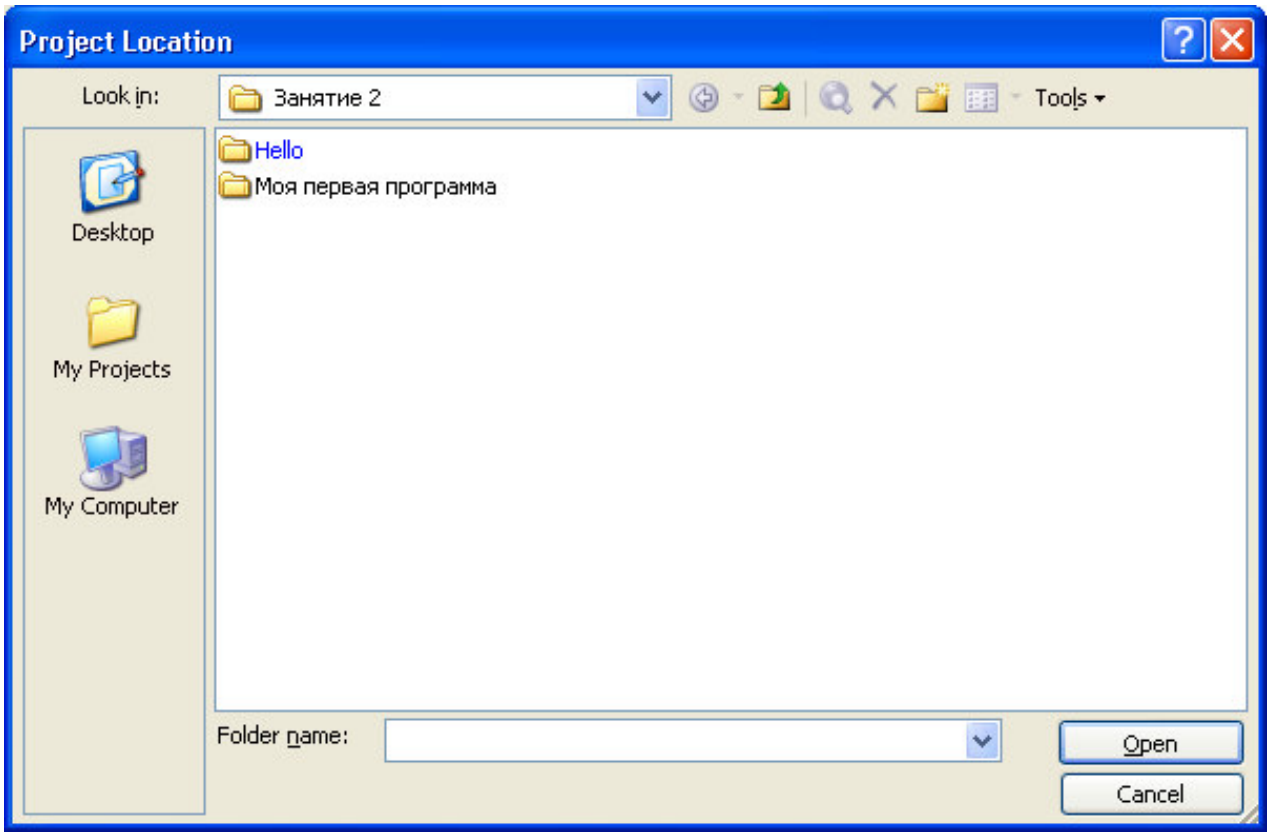
2. В поле **Name** (имя) наберите имя проекта.

3. В поле **Location** (путь) наберите путь, где будет сохранен ваш проект, или нажмите кнопку **Browse…**, чтобы выбрать путь с помощью диалогового окна **Project Location** (рис. 2.15).

4. В поле **Solution** наберите имя решения.

5. Нажмите кнопку **Save**. Ваш проект будет сохранен

В дальнейшем, если вы будете нажимать на кнопку **Save All**, диалоговое окно **Save Project** появляться не будет, а проект будет сохраняться в той папке, которую вы выбрали впервые для сохранения проекта.



Р и с. 2.15. Диалоговое окно Project Location

Совет. Не забывайте сохранять проекты время от времени, чтобы не потерять результаты работы из-за различных аварийных ситуаций (прекращение подачи электричества, сбой системы). Также сохраняйте проекты перед запуском.

2.8. СОЗДАНИЕ ИСПОЛНЯЕМОГО ФАЙЛА

Последней задачей будет завершение процесса разработки и создание приложения для Windows или исполняемого файла. Приложения для Windows, созданные в Visual Basic, имеют расширение **.exe** и могут выполняться на любом компьютере с Microsoft Windows и необходимыми файлами поддержки. Visual Basic устанавливает эти файлы, включая динамические библиотеки и файлы Framework, автоматически.

В данный момент вы должны знать, что у Visual Studio есть возможность создавать для вашего проекта два типа исполняемых файлов – **отладочный** и **окончательный**. В Visual Studio по умолчанию создаются отладочные версии, которыми пользуются при тестировании и отладке программ. Отладочные версии содержат отладочную

информацию, которая делает их медленнее, тем не менее компилятор Visual Basic выполняет их сборку довольно быстро. Отладочная версия исполняемого файла создается автоматически, после того как вы нажимаете кнопку **Start Debugging**  .

Когда разработка программы завершена, можно скомпилировать приложение в варианте конечной сборки, которая включает различные оптимизации и не содержит ненужной отладочной информации.

Чтобы создать исполняемый файл выполните следующие действия.

1. В меню Build (Построение) выберите команду Build Solution (Построить решение). По этой команде в папке, где расположен ваш проект, будет создана подпапка с именем **bin** (если ее еще нет), а внутри нее еще одна папка **Release**, и исходный код проекта будет скомпилирован именно в этой папке. Результатом этого является исполняемый файл с именем **Моя первая программа.exe**. Для экономии времени Visual Studio часто создает эти файлы во время разработки приложения, однако всегда, когда вы заканчиваете какой-либо важный промежуточный этап, следует перекомпилировать ваше приложение вручную с помощью команды Build Solution (Построить решение). В частности, приложение необходимо перекомпилировать, если вы переключаетесь с отладочного режима на окончательный.

1. Загрузите Проводник Windows (Explorer). Откройте папку с вашим проектом.

2. В папке проекта откройте папку **bin**, а затем папку **Release**.

3. Выберите значок приложения Моя первая программа.exe и запустите его.

Программа "Семерка" загрузится и запустится в операционной системе – вы запустили программу вне среды разработки Visual Studio.

4. Нажмите кнопку **Приветствие**. Программа отобразит на форме приветствие "Здравствуй, Мир!".

5. Чтобы остановить программу «Здравствуй, Мир!», щелкните на кнопке **Выход**.

Программа выполнит команду завершения работы. Форма закроется и закончит свою работу.

2.9. ВЫХОД ИЗ VISUAL STUDIO

1. Сохраните все изменения, которые вы сделали в вашей программе, щелкнув на кнопке **SaveAll** (Сохранить все) на стандартной панели инструментов  . Эту операцию можно также совершить из меню **File**, выбрав пункт меню **SaveAll** (Сохранить все).

2. В меню **File** (Файл) щелкните на команде **Exit** (Выход).

Среда разработки Visual Studio закроется.

2.10. ПРОГРАММА «ПРИВЕТСТВИЕ ПОЛЬЗОВАТЕЛЯ»

Создадим еще одну простую программу. Эта программа должна позволить пользователю ввести свое имя с клавиатуры и приветствовать его с помощью сообщения и появления приветственной картинки.

1. Запустите Visual Studio.

2. Создайте новый проект с именем «Приветствие пользователя». Подробно о создании нового проекта рассказано при разработке проекта «Здравствуй, Мир!».

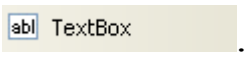
Начнем создавать интерфейс программы. Форма после размещения элементов управления и изменения их свойств должна выглядеть, как на рис. 2.16.

3. Из панели **Toolbox** поместите на форму сверху элемент управления **Label** (надпись). Данная надпись будет использоваться для приветствия пользователя.

4. Найдите на панели **Toolbox** элемент управления **PictureBox** (Поле рисунка) 

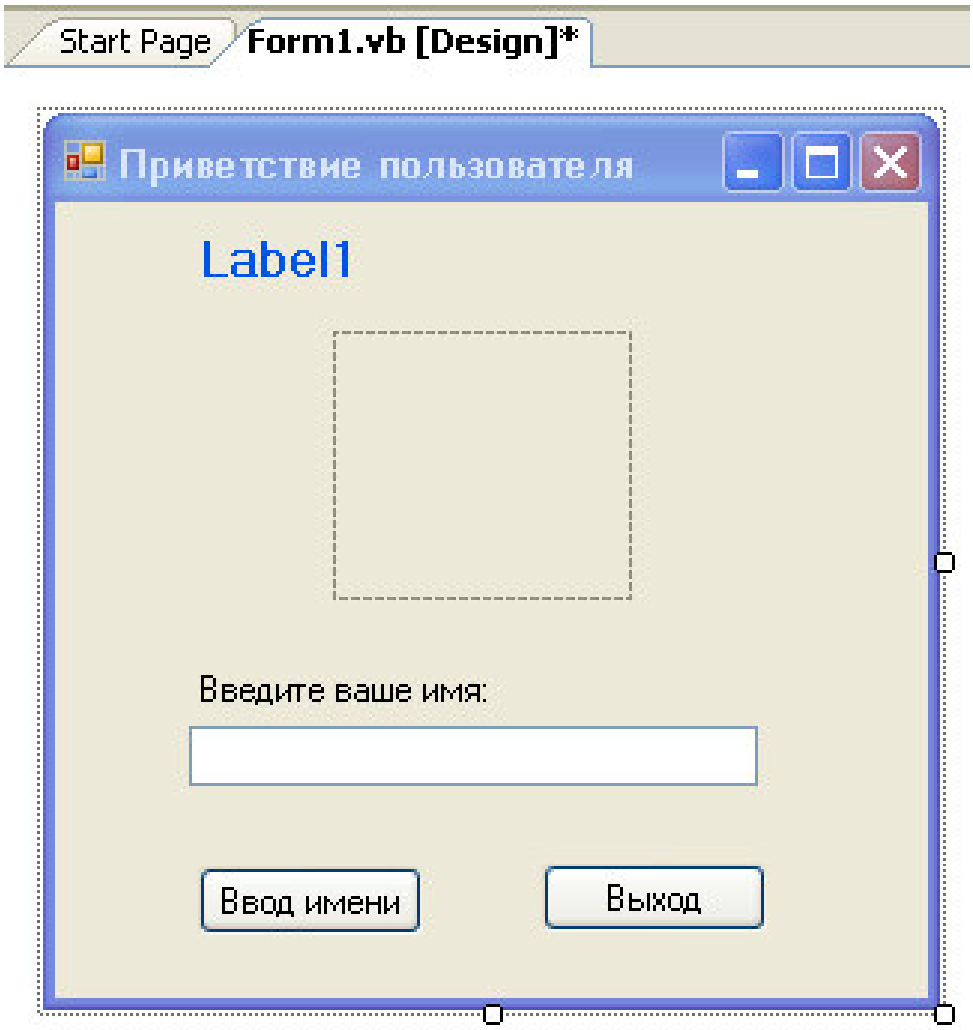
.

5. Поместите на форму элемент управления **PictureBox** (Поле рисунка). Элемент управления разместите ниже приветственной надписи. В этот объект будет загружен из файла приветственный рисунок.

6. Найдите на панели **Toolbox** элемент управления **TextBox** (Текстовое поле) 

.

7. Поместите элемент управления **TextBox** (Текстовое поле) на форму. Текстовое поле будет предназначено для ввода текста, а в нашей программе – для ввода имени пользователя.



Р и с. 2.16. Проект формы для программы «Приветствие пользователя»

8. Из панели **Toolbox** поместите на форму элемент управления **Label** (надпись). Расположите ее над текстовым полем. Данная надпись будет пояснять, для чего нужно текстовое поле.

9. Из панели **Toolbox** поместите на форму две кнопки (элемент управления **Button**). Одна кнопка будет предназначена для подтверждения ввода имени, другая – для завершения работы программы.

10. Измените свойства объектов формы согласно табл. 2.2.

Обратите внимание, что имя (свойство Name) текстового поля (TextBox) – **txtUserName**. Здесь **txt – префикс** для элемента управления **TextBox**.

Имя поля для рисунка (PictureBox) – **picFace**. Здесь **pic** – префикс для объекта **PictureBox** (Поля для рисунка). Свойство **Image** используется для того, чтобы указать путь к точечному рисунку. Свойство

SizeMode указывает, как вести себя точеному рисунку в объекте **PictureBox**:

– если свойство установлено в **Normal**, то размеры рисунка не зависят от размеров объекта **PictureBox**;

– если свойство установлено в **StretchImage**, то рисунок изменяется (растягивается или уменьшается) до размеров объекта **PictureBox**, при этом **не соблюдается** пропорциональность рисунка;

– если свойство установлено в **AutoSize**, то размер объекта **PictureBox** изменяется (растягивается или уменьшается) до размеров рисунка, при этом пропорциональность рисунка не соблюдается;

– если свойство установлено в **CenterImage**, то размеры рисунка не зависят от размеров объекта **PictureBox**, сам рисунок центрируется внутри объекта **PictureBox**;

– если свойство установлено в **Zoom**, то рисунок изменяется (растягивается или уменьшается) до размеров объекта **PictureBox**, при этом **соблюдается** пропорциональность рисунка.

Свойство **Visible** отвечает за видимость объекта на форме во время работы программы. В нашей программе свойство **Visible** отвечает за видимость рисунка в объекте **PictureBox** во время работы программы. Если свойство установлено как **False (Ложь)**, то объект (в нашей программе **PictureBox**) будет невидим. Если свойство установлено как **True (Истина)**, то объект будет видим.

В нашем случае для объекта **PictureBox** мы устанавливаем **False** – невидимый. Это делается для того, чтобы пользователь сразу не увидел рисунка. Рисунок станет видимым, когда пользователь нажмет кнопку **Ввод имени**.

Обратите внимание, что у объекта **Label2** мы не стали менять имя (свойство **Name**). Это связано с тем, что данный объект не участвует в коде программы и никак не изменяется: не меняет своих свойств и не использует свои методы во время работы программы, а служит только для информативных целей.

Таблица 2.2

Таблица значений свойств программы «Приветствие пользователя»

Объект	Свойство	Значение
Form1	Name	frmWelcomeUser
	Text	«Приветствие пользователя»
Button1	Name	btnEnterName
	Text	«Ввод имени»
Button2	Name	btnExit
	Text	«Выход»
Label1	Name	lblMessage
	Font	Microsoft Sans Serif, жирный, 12 пунктов
	ForeColor	Blue (Синий)
Label2	Text	«Введите ваше имя:»
TextBox1	Name	txtUserName
PictureBox1	Name	picFace
	Image	"C:\VB2005\Занятие 2\Приветствие пользователя"
	SizeMode	StretchImage (вписать)
	Visible	False

Переходим к созданию кода программы.

11. Дважды щелкните по форме в дизайнере форм. Загрузите редактор кода.

12. Убедитесь, что вы работаете с процедурой **frmWelcomeUser_Load**. Данная процедура обрабатывает событие загрузки формы.

```
    Private Sub frmWelcomeUser_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load

    End Sub
```

13. Наберите текст в теле процедуры:

```
    lblMessage.Text = ""
    txtUserName.Select()
```

Метод **Select** для объекта **txtUserName** делает активным сам объект **txtUserName**. Это нужно для того, чтобы при загрузке формы

курсор клавиатуры мигал в текстовом поле и пользователь понял, что в это поле надо что-то ввести.

Методы – это специальные выражения, которые выполняют действие или работу для конкретного объекта, например преобразуют числа в строку или добавляют еще один элемент к списку. Методы отличаются от свойств, которые содержат значение, и процедур событий, которые выполняются при манипуляциях пользователя с объектом. Методы могут использоваться совместно в нескольких объектах, так что когда вы узнаете, как использовать один метод, вы сможете использовать его в различных условиях.

14. Создайте процедуру обработки события щелчка по кнопке **btnEnterName**:

```
    Private Sub btnEnterName_Click(ByVal sender As Object, ByVal e
As System.EventArgs) Handles btnEnterName.Click

    End Sub
```

15. В теле процедуры **btnEnterName_Click** наберите следующий текст:

```
        Dim strUserName As String
        strUserName = txtUserName.Text
        lblMessage.Text = "Здравствуйте, " & strUserName & "!"
       picFace.Visible = True
```

В первой строке

```
Dim strUserName As String
```

инструкция **Dim** объявляет переменную **strUserName** как **строковой тип (String).** Ключевое слово **As** переводится с англ. словом **«Как»**. После ключевого слова **As** указывается тип переменной. Переменные типа **String** содержат в себе строковые (текстовые) данные, например «Здравствуй, Мир!». Переменные типа **String** могут содержать приблизительно до 2 миллиардов (2^31) знаков.

Переменная – это специальный контейнер, используемый для временного хранения данных в программе. Переменные могут быть

различного типа: **строковые** (текстовые), **числовые**, **логические** (булевы), **дата** и др.

Во второй строке

```
strUserName = txtUserName.Text
```

переменной **strUserName** присваивается величина, содержащаяся в свойстве Text текстового поля **txtUserName**, т.е. текст, который ввел пользователь.

В третьей строке

```
lblMessage.Text = "Здравствуйте, " & strUserName & "!"
```

свойству **Text** объекта **lblMessage** присваивается сразу три текстовые величины: 1) "Здравствуйте"; 2) переменная strUserName; 3)"!". Знак & – это оператор объединения текстовых величин. Этот оператор также называется **конкатенацией**. В результате объединения строк получается одна текстовая строка.

В четвертой строке

```
picFace.Visible = True
```

свойству **Visible** объекта **picFace** присваивается величина **True**, вследствие чего картинка в объекте **picFace** становится видимой.

16. Создайте процедуру обработки события щелчка по кнопке **btnExit**:

```
    Private Sub btnExit_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnExit.Click

    End Sub
```

17. В теле процедуры **btnExit_Click** напишите команду

```
End
```

Инструкция **End** завершает работу программы.

18. Напишите комментарий к программе.

Текст нашей программы вместе с комментариями должен принять вид

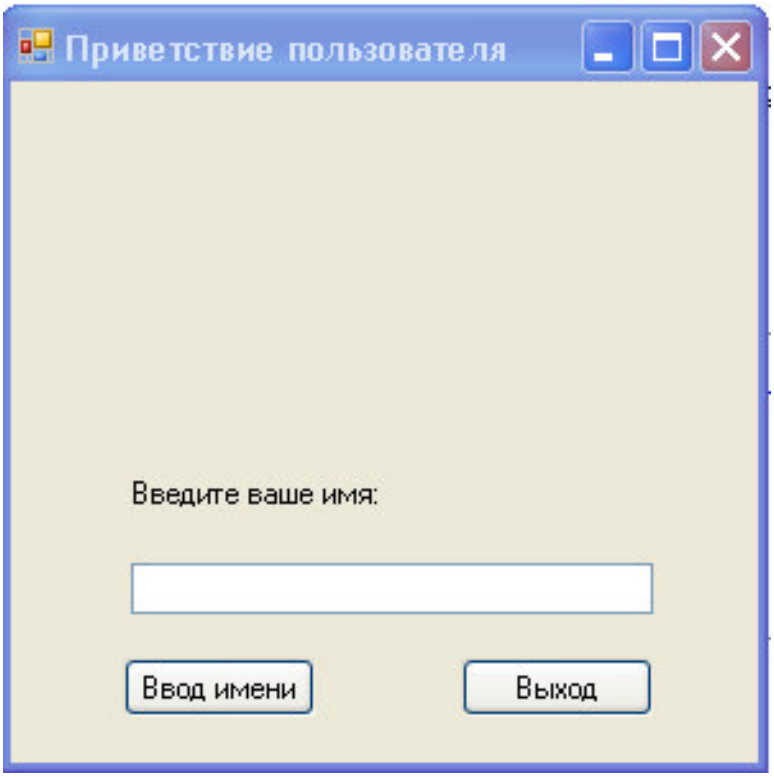
```
Public Class frmWelcomeUser

    Private Sub btnEnterName_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnEnterName.Click
        'Объявление переменной типа String
        Dim strUserName As String
        'Переменной присваиваем текст из текстового поля
        strUserName = txtUserName.Text
        'Помещаем приветствие в надпись
        lblMessage.Text = "Здравствуйте, " & strUserName & "!"
        'делаем картинку видимой
        picFace.Visible = True
    End Sub

    Private Sub frmWelcomeUser_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        'очистка надписи от содержимого
        lblMessage.Text = ""
        'Делаем текстовое поле активным
        txtUserName.Select()
    End Sub

    Private Sub btnExit_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnExit.Click
        'Завершаем работу программы.
        End
    End Sub

    End Class
```

19. Сохраните проект.

20. Запустите программу. На экране должна появиться форма (рис.2.17).

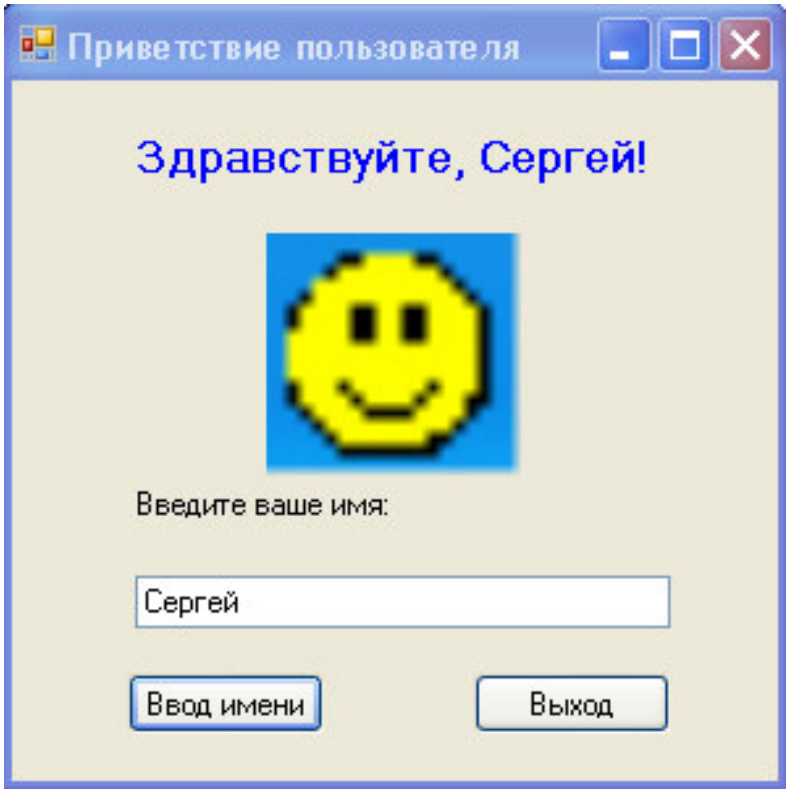
21. Введите имя в текстовое поле.

22. Нажмите кнопку **Ввод имени.**



Р и с. 2.17. Программа «Приветствие пользователя» после запуска

На форме должны появиться приветственная надпись и отобразиться рисунок (рис. 2.18).



Р и с. 2.18. Программа «Приветствие пользователя»
после нажатия кнопки Ввод имени

23. Нажмите кнопку Выход. Программа должна завершить свою работу.

24. Закройте среду разработки Visual Studio.

2.11. ИСПОЛЬЗОВАНИЕ ЭЛЕМЕНТА УПРАВЛЕНИЯ DATETIMEPICKER

Некоторые элементы управления Visual Basic показывают информацию, а другие собирают информацию от пользователя или обрабатывают данные "за кулисами". В следующем упражнении мы будем работать с элементом управления **DateTimePicker**, который с помощью графического календаря со стрелками прокрутки спрашивает пользователя о дате или времени. Хотя мы посмотрим только самые простые возможности элемента управления **DateTimePicker**, эксперименты с ним дадут представление о том, как много всего могут автоматически сделать элементы управления Visual Basic и как можно обрабатывать информацию, которая от них поступает.

2.12. ПРОГРАММА «ДЕНЬ РОЖДЕНИЯ»

В программе «День рождения» элемент управления **DateTimePicker** используется, чтобы выяснить у пользователя дату его рождения и показать эту информацию в окне сообщения.

1. Запустите Visual Studio.

2. Создайте новый проект с именем «День рождения».

Начнем создавать интерфейс программы. Форма после размещения элементов управления и изменения их свойств должна иметь вид, показанный на рис. 2.19.

3. На панели элементов (**ToolBox**) найдите элемент управления **DateTimePicker** и поместите его на форму.

Объект выбора даты и времени **DateTimePicker** по умолчанию показывает текущую дату, но эту настройку можно изменить в свойстве **Value**. Показ даты при проектировании очень удобен – это позволяет при создании объекта правильно задать его размеры.

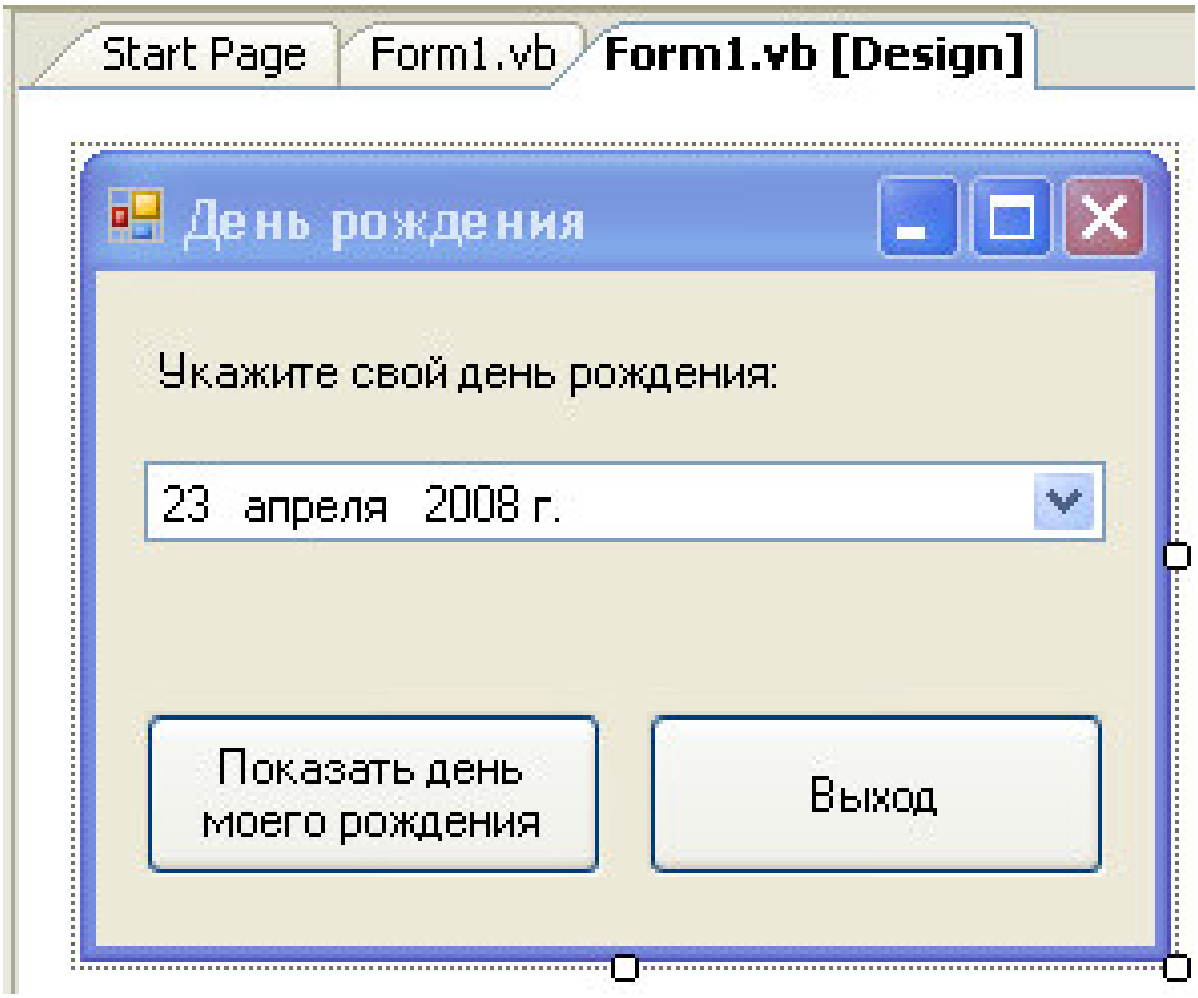


Рис. 2.19. Проект формы программы «День рождения»

4. Из панели **Toolbox** поместите на форму элемент управления **Label** (надпись). Расположите ее над элементом **DateTimePicker**. Данная надпись показывает, для чего пользователю нужен объект выбора даты и времени.

5. Из панели **Toolbox** поместите на форму две кнопки (элемент управления **Button**). Одна кнопка будет использована для показа дня рождения и для проверки правильности работы объекта **DateTimePicker**, другая – для завершения работы программы.

6. Измените свойства объектов формы согласно табл. 2.3.

Обратите внимание, что имя (свойство **Name**) объекта **DateTimePicker** – **dtpBirthDay**. Здесь **dtp – префикс** для элемента управления **DateTimePicker**.

Таблица 2.3

Таблица значений свойств программы «День рождения»

Объект	Свойство	Значение
Form1	Name	frmBirthDay
	Text	«День рождения»
Label1	Text	«Укажите свой день рождения:»
DateTimePicker1	Name	dtpBirthDay
Button1	Name	btnShow
	Text	«Показать день моего рождения»
Button2	Name	btnExit
	Text	«Выход»

Переходим к созданию кода программы.

7. Дважды щелкните по кнопке **btnShow** (show – показать, англ.) в дизайнере форм. Загрузите редактор кода.

8. Убедитесь, что вы работаете с процедурой **btnShow_Click**. Данная процедура обрабатывает событие нажатия на кнопку **btnShow**.

9. Наберите следующий фрагмент программы между инструкциями **Private Sub** и **End Sub** в процедуре события **Button1_Click**:

```
    MsgBox("Ваш день рождения " & dtpBirthDay.Text)
    MsgBox("День года: " & dtpBirthDay.Value.DayOfYear.ToString)
    MsgBox("Номер дня в неделе:" &
dtpBirthDay.Value.DayOfWeek.ToString)
```

Этот фрагмент программы показывает три последовательных окна сообщения (небольшие диалоговые окна, вызываемые с помощью **функции MsgBox**), которые содержат информацию из объекта календаря.

В первой строке

```
MsgBox("Ваш день рождения " & dtpBirthDay.Text)
```

используется свойство **Text** объекта **dtpBirthDay**. Это свойство получает содержимое объекта **dtpBirthDay**, является текстовой величиной (String). Содержимое объекта в нашем случае – это информация о дате рождения, которую пользователь выберет в этом объекте после запуска программы. Содержимое объекта **dtpBirthDay** объединяется

со строкой "Ваш день рождения" в одну общую строку с помощью оператора **конкатенации** (слияния) строк &. **Функция MsgBox** отображает диалоговое окно, в котором отображена текстовая строка. В нашей программе это общая строка, полученная с помощью слияния двух текстовых величин. Когда появляется диалоговое окно, программа приостанавливается и ждет, когда пользователь нажмет кнопку ОК, расположенную ниже выводимого текста. Когда пользователь нажмет кнопку ОК программа переходит ко второй строке.

Вторая строка

```
MsgBox("День года: " & dtpBirthDay.Value.DayOfYear.ToString())
```

показывает число дней, прошедших с 1 января до даты, указанной в календаре. Это делается с помощью свойства **DayOfYear** и свойства **Value**. Свойство **Value** содержит в себе различные величины, а свойство **DayOfYear** является содержимым (частью) свойства **Value** и возвращает (определяет) число дней, прошедших с начала года до указанной даты в объекте **dtpBirthDay**. Свойство **DayOfYear** возвращает числовое значение типа **Integer**. Тип **Integer** является целым типом, т.е. может содержать в себе только целое число. Переменные типа **Integer** хранятся как 32-битные (4-байтные) целые числа со знаком, значения которых лежат в диапазоне от -2 147 483 648 до 2 147 483 647.

В общем случае метод **ToString** переводит данные различных типов в строковое (текстовое) значение (в тип **String**). В нашей программе метод **ToString** переводит числовой результат (тип **Integer**) вычисления даты в строковое (текстовое) значение (в тип **String**), которое гораздо проще показать с помощью функции **MsgBox**, так как эта функция отображает данные именно в строковом виде (тип **String**)

Третья строка

```
 MsgBox("Номер дня в неделе:" &
dtpBirthDay.Value.DayOfWeek.ToString)
```

выводит в окно сообщений номер дня в неделе, указанной даты.

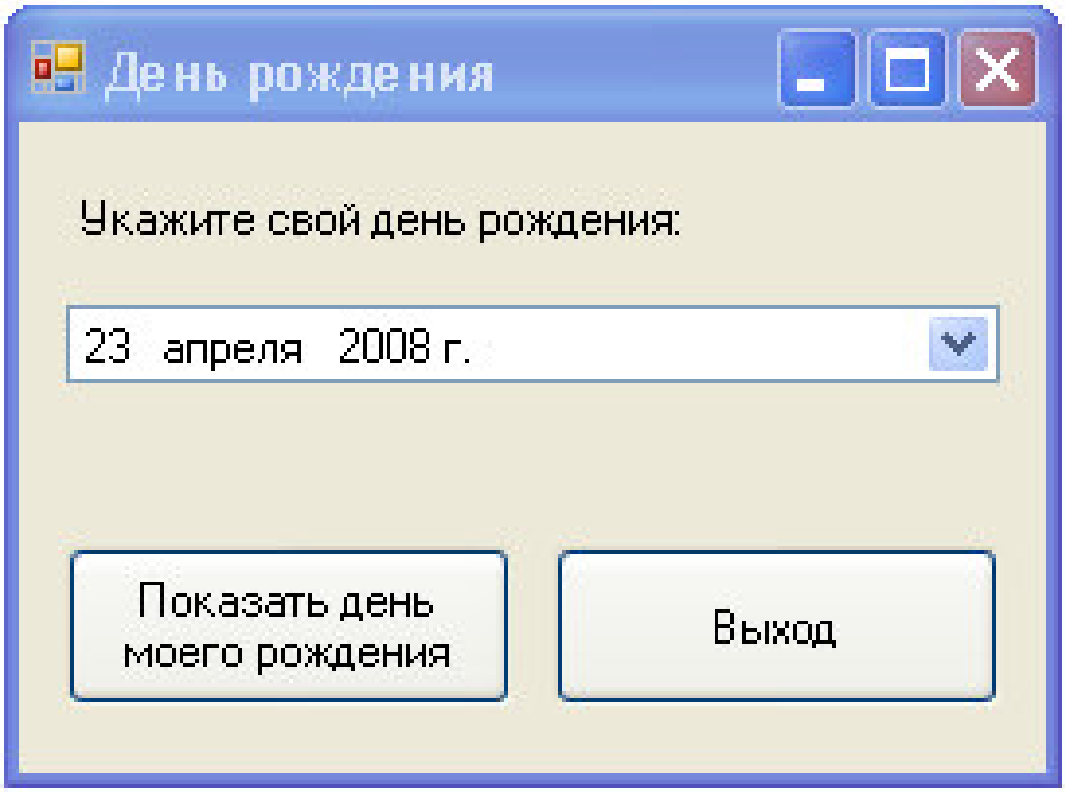
10. Создайте процедуру обработки события щелчка по кнопке **btnExit**, которая будет завершать работу программы:

```
    Private Sub btnExit_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnExit.Click
        End
    End Sub
```

11. Чтобы сохранить внесенные изменения на диске, нажмите кнопку Save All (Сохранить все) .

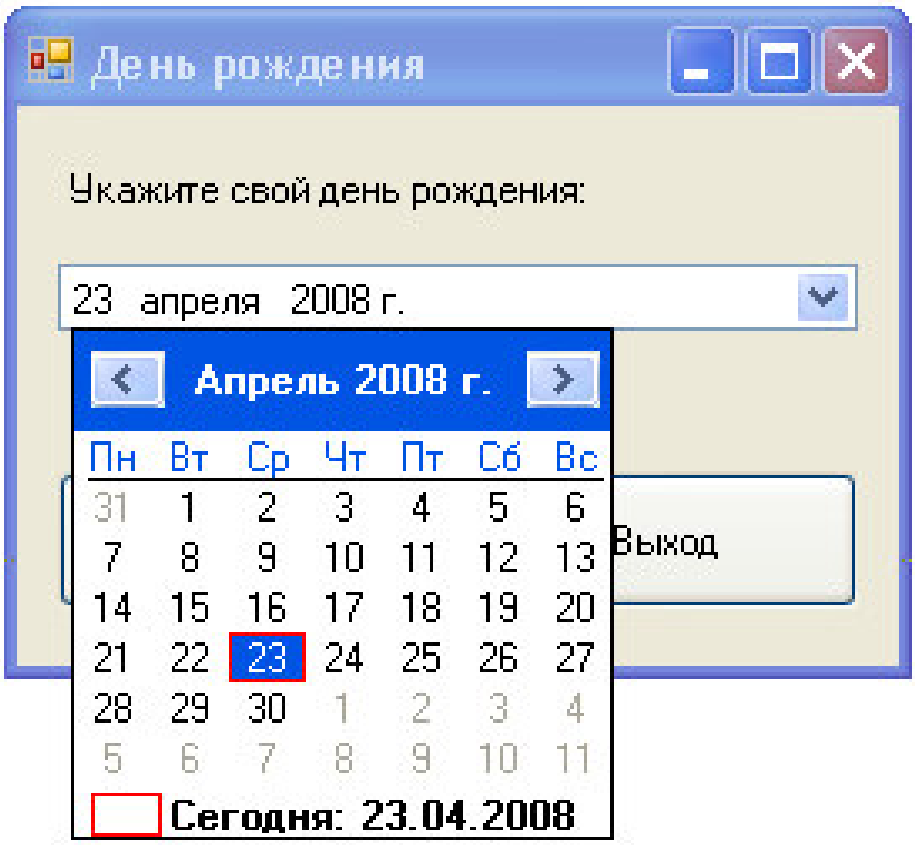
Теперь можно запустить программу «День рождения».

12. На стандартной панели инструментов нажмите кнопку Start (Начать). Программа «День рождения» запустится в среде разработки. В окне объекта выбора даты и времени появится текущая дата (рис. 2.20).



Р и с. 2.20. Программа «День рождения» после запуска

13. Нажмите стрелку раскрывающегося списка, чтобы вывести на экран представление этого объекта в виде календаря. Форма будет выглядеть, как на рис. 2.21 (дата, разумеется, будет другая).



Р и с. 2.21. Календарь

14. Чтобы увидеть предыдущие месяцы календаря, нажмите на левой стрелке прокрутки. Обратите внимание, что текстовое поле объекта при прокрутке дат также изменяется, однако значение Сегодня в нижней части календаря остается неизменным.

Хотя календарь можно промотать назад до даты вашего рождения, вряд ли у кого хватит терпения прокручивать месяц за месяцем. Чтобы быстрее выбрать год, подведите указатель мыши к выбору года, нажмите на текущее значение года и с помощью появившихся стрелок выберите нужный вам год.

Чтобы выбрать месяц, подведите указатель мыши к выбору месяца, нажмите на месяц и из появившегося меню выберите нужный месяц.

Далее останется выбрать только дату.

Выбрать нужную дату можно и не заходя в календарь. Для этого выделите в текстовом поле нужное значение (год, месяц, число) и введите новое значение. Нужное значение в текстовом поле можно выбрать и с помощь клавиш: <стрелка вверх> и <стрелка вниз>.

15. Вместо выделенного года введите год вашего рождения, а затем снова щелкните на стрелке раскрывающегося списка. Появится календарь, который будет открыт на годе вашего рождения.

16. В календаре выберите месяц вашего рождения

17. Нажмите кнопку **Показать день моего рождения**. Visual Basic исполнит введенный вами код программы и покажет окно с сообщением, содержащим день и дату вашего рождения. Обратите внимание на соответствие двух дат (рис. 2.22).

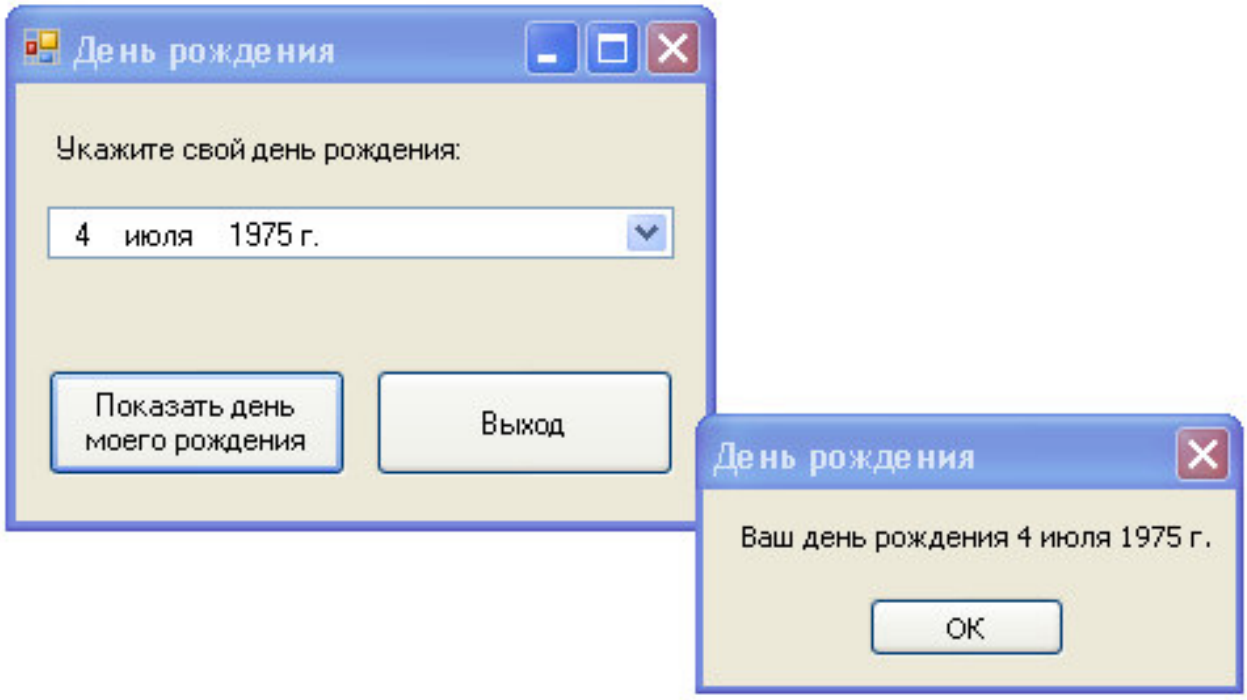


Рис. 2.22. Диалоговое окно, содержащее дату рождения

18. В окне сообщения нажмите **OK**. Появится второе окно сообщения, указывающее, в какой день года вы родились.

19. Нажмите **OK**, чтобы показать последнее окно сообщения. Появтся номер дня в неделе, когда вы родились.

20.Чтобы закрыть окно сообщения, щелкните на **OK**, а затем на кнопке **Закрыть**.

21. Закройте среду разработки Visual Studio.

Объект выбора даты и времени очень удобен: он не только помнит новую, введенную вами информацию о дате или времени, но

также отслеживает текущие дату и время и может показывать эту информацию в различных форматах.

Совет. Чтобы настроить объект выбора даты и времени для показа времени, а не даты, установите свойство **Format** этого объекта равным **Time**.

2.13. ИСПОЛЬЗОВАНИЕ ЭЛЕМЕНТА УПРАВЛЕНИЯ LINKLABEL

Во многих Windows-приложениях уже давно предоставляется доступ в интернет, и это уже становится стандартной функцией. В Visual Studio добавить такую функцию к создаваемой программе стало намного проще. Используя шаблоны библиотеки Web Forms и другие возможности, можно создавать сложные приложения для работы по сети Интернет, а чтобы открыть веб-страницу в веб-обозревателе, потребуется всего лишь несколько строк кода.

2.14. СОЗДАНИЕ ПРОГРАММЫ WEBLINK

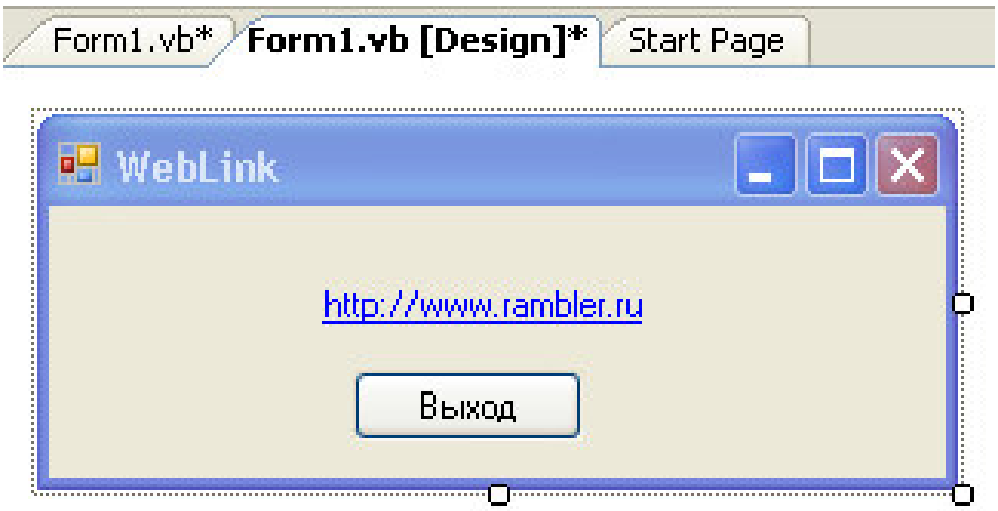
В этом упражнении вы научитесь применять элемент управления **LinkLabel**, который в поле формы показывает текст, являющийся ссылкой на адрес в Интернете.

Используя элемент управления **LinkLabel** вместе с методом **Process.Start**, можно открывать ссылки на форме с помощью обозревателей Internet Explorer, Opera или других. В нашем примере с помощью элемента управления **LinkLabel** мы подключимся к веб-странице Rambler – информационно-поисковой системе.

1. Запустите Visual Studio.

2. Создайте новый проект с именем **WebLink**.

Начнем создавать интерфейс программы. Форма после размещения элементов управления и изменения их свойств должна выглядеть так же, как на рис. 2.23.



Р и с. 2.23. Проект формы программы WebLink

3. В области элементов найдите элемент управления **LinkLabel** LinkLabel.

4. Поместите элемент управления **LinkLabel** на форму. Объект **LinkLabel** выглядит как обычные объекты надписи (неизменяемый текст), но отображается на форме синим цветом с подчеркиванием.

5. Измените свойства объектов формы согласно табл. 2.4.

Обратите внимание, что имя (свойство **Name**) объекта **LinkLabel** – **lnkRambler**. Здесь **lnk – префикс** для элемента управления **LinkLabel**. http://www.rambler.ru -это веб-сайт информационно-поисковой системы Rambler.

Таблица 2.4

Таблица значений свойств программы «День рождения»

Объект	Свойство	Значение
Form1	Name	frmWebLink
	Text	«WebLink»
LinkLabel1	Name	lnkRambler
	Text	«http://www.rambler.ru»
Button1	Name	btnExit
	Text	«Выход»

Переходим к созданию кода программы.

6. Дважды щелкните мышью на объекте ссылки (lnkRambler). В коде программы должна появиться процедура **lnkRambler_ LinkClicked**

7. В процедуре **lnkRambler_LinkClicked** наберите следующий код вместе с комментариями:

```
        ' Изменяем цвет ссылки, установив LinkVisited в значение
True.
        lnkRambler.LinkVisited = True

        ' Используем метод Process.Start, который запускает
обозреватель,
        ' указанный в системе по умолчанию с URL
"http://www.rambler.ru/":
        System.Diagnostics.Process.Start("http://www.rambler.ru/")
```

В строке

```
lnkRambler.LinkVisited = True
```

присвоение свойству **LinkVisited** значения **True** делает ссылку бледно-фиолетовой, что во многих обозревателях указывает на то, что ассоциированный со ссылкой HTML-документ уже был вызван раньше. Хотя для показа веб-страницы настройка этого свойства необязательна, считается хорошим тоном предлагать пользователю информацию в согласовании с другими приложениями.

Строка программы

```
System.Diagnostics.Process.Start("http://www.rambler.ru/")
```

запускает веб-обозреватель по умолчанию (например Internet Explorer), если он еще не запущен. (Если обозреватель уже запущен, то в нем немедленно загружается этот URL-адрес.) Метод **Start** класса **Process** выполняет важную работу, запуская в памяти процесс или сеанс для исполнения программы обозревателя. Класс **Process** управляет многими аспектами исполнения программ и входит в библиотеку объектов **System.Diagnostics**, которую программисты на Visual Basic называют "пространство имен System.Diagnostics". Передав в метод **Start** интернет-адрес (или URL-адрес), мы сообщили программе на Visual Basic, что хотим посмотреть веб-сайт, и Visual Basic оказался

достаточно умным, чтобы определить, что наилучшим инструментом для просмотра этого URL-адреса является обозреватель по умолчанию, хотя мы и не указывали его имя.

Метод **Process.Start** замечателен тем, что с его помощью можно запустить и другие Windows-приложения. Если для просмотра URL-адреса нужно определить конкретный обозреватель по его имени, это можно сделать так (в данном случае вызывается Internet Explorer):

```
    System.Diagnostics.Process.Start("IExplore.exe",
"http://www.rambler.ru)
```

Здесь в методе **Start** используются два аргумента, разделенные запятой. Точное расположение в системе программы с именем IExplore.exe не указано, но Visual Basic при запуске программы будет искать ее по открытым путям.

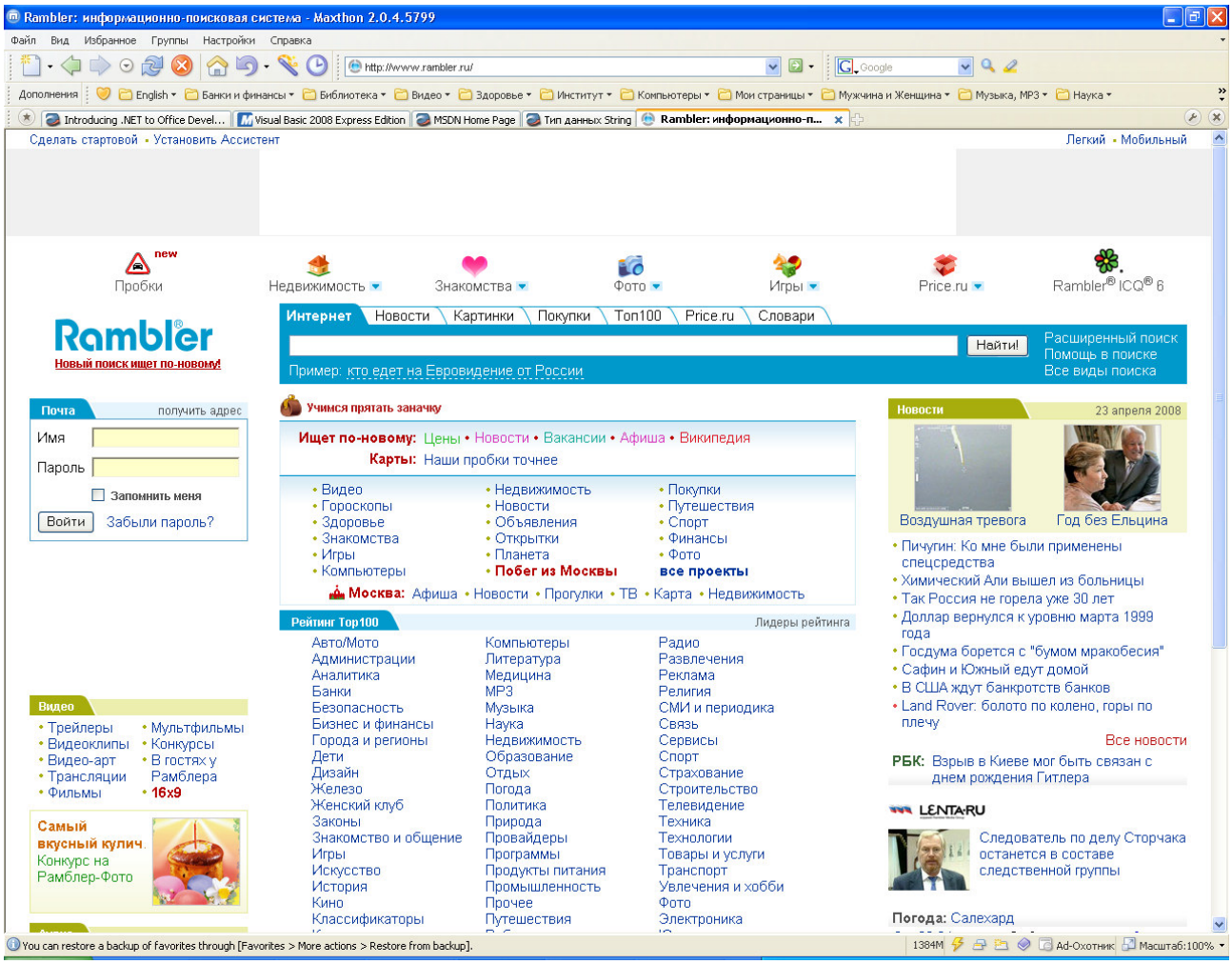
Если бы требовалось запустить с помощью метода **Start** другое приложение, например Microsoft Word (и открыть в нем документ **c:\myletter.doc**), то это можно было бы сделать так:

```
    System.Diagnostics.Process.Start("Winword.exe",
"c:\myletter.doc")
```

8. Сохраните проект. Чтобы сохранить изменения, на стандартной панели инструментов нажмите кнопку Save All (Сохранить все) .

9. Запустите проект. Для этого на стандартной панели инструментов нажмите кнопку Start Debugging (Начать отладку). Приложение начнет выполняться, откроется форма, показывающая ссылку на веб-сайт и имеющая подходящий текст в строке заголовка.

10. Чтобы открыть указанный веб-сайт (www.rambler.ru), щелкните по ссылке. Обратите внимание, что это просто удачное совпадение – то, что свойство **Text** объекта ссылки **LinkLabel** содержит тот же URL-адрес, что указан в коде программы. В качестве ссылки можно ввести любой текст, который лучше подходит. Для текста ссылки можно использовать свойство **Image**, и связанное с ним изображение будет показано как фон объекта **LinkLabel**. На рис. 2.24 показана веб-страница Rambler.ru, которую вызывает программа WebLink с помощью Internet Explorer.



Р и с. 2.24. Веб-страница Rambler: информационно-поисковая система

11. Снова переключитесь на форму (если она не видна, нажмите значок формы WebLink на панели задач Windows).

Обратите внимание, что теперь ссылка окрашена другим цветом. Как и для обычной веб-ссылки, изменение цвета означает, что эта ссылка уже использовалась (но все еще активна).

12. Чтобы выйти из программы, нажмите на кнопку **Выход**.

13. Закройте Visual Studio.

2.15. НЕСКОЛЬКО СЛОВ О ТЕРМИНОЛОГИИ

К настоящему моменту авторами было использовано несколько терминов, которые описывают элементы программы на Visual Basic. Так как до сих пор мы не дали строгих определений этим терминам, давайте сделаем это сейчас.

Инструкция. Инструкция – это зарезервированное или ключевое слово в коде, которое выполняет работу в программе. Инструкции программы на Visual Basic выделяют память для хранения данных, открывают файлы, делают вычисления и выполняют множество других важных задач. Большинство ключевых слов в редакторе кода имеют синий цвет.

Переменная. Переменная – это специальный контейнер, используемый для временного хранения данных в программе. С помощью инструкции **Dim** программист создает переменные для хранения результатов вычислений, создания имен файлов, отслеживания ввода и т.п. В переменных можно хранить числа, имена и значения свойств.

Элемент управления. Элемент управления – это инструмент, который используется для создания объектов в графической форме Visual Basic. Вы выбираете элементы управления в области элементов и рисуете с помощью мыши объект в поле формы. Большинство элементов управления используется для создания элементов интерфейса пользователя: кнопок, надписей, текстовых полей, списков и полей вывода изображений и многих других. Элементы управления явления на форме являются **объектами** формы.

Объект. Объект – это какая-либо сущность, имеющая свойства, методы, события. **Объектами формы** являются элементы пользовательского интерфейса, который вы создаете в поле формы Visual Basic с помощью элемента управления из области элементов. Меняя значения свойств, **объекты** можно перемещать, настраивать и изменять их размер. Объекты обладают встроенной функциональностью – они "знают", как выполнять те или иные действия, и могут самостоятельно реагировать при возникновении определенных обстоятельств. (Например, кнопка надавливается, когда пользователь нажимает на нее.) Объекты являются членами классов, которые служат шаблонами, определяющими вид и поведение объекта. Объекты в Visual Basic можно программировать, создавая для них процедуры обработки событий, предназначенных для реагирования на те или иные возникающие в программе ситуации. Сама форма в Visual Basic также является объектом.

Кроме объектов формы существует множество других типов объектов. Например, документ Word является объектом. Составляющие документы приложения Word (текст, формулы, рисунки, таблицы) также являются объектами. С помощью Visual Basic можно управлять этими объектами.

Свойство. Свойство – это значение или характеристика, принадлежащая объекту. Например, объект **Button** имеет свойство **Text**, в котором хранится текст, и свойство **Image**, определяющее путь к файлу с изображением, которое появляется на поверхности этой кнопки. В Visual Basic эти свойства можно задавать как при проектировании в окне Properties (Свойства), так и во время выполнения программы, с помощью операторов в коде программы. В тексте программы команда настройки свойств имеет вид

```
Object.Property = Value
```

где **Object** – это имя настраиваемого объекта; **Property** – характеристика (свойство), которую нужно изменить; **Value** – новое значение свойства. Например, строка

```
lblMessage.Text = "Здравствуй, Мир!"
```

присвоит значение "Здравствуй, Мир!" свойству **Text** объекта **lblMessage**.

Присваивание. Операция присваивания осуществляется с помощью оператора «=». Этот оператор используется для присвоения значения переменной или свойству. Общий вид операции присваивания

```
variable = value
```

где **variable** – любая переменная или любое перезаписываемое свойство; **value** – любое литерал, константа или выражение.

Имя с левой стороны от знака равенства может быть простой скалярной переменной, свойством или элементом массива. Свойства с левой стороны знака равенства могут быть только перезаписываемы-

ми во время выполнения свойствами. Во время использования значение, находящееся справа от знака равенства, присваивается переменной, которая находится слева от знака равенства.

Например:

```
myInt = 42
```

Здесь переменной myInt присваивается целое значение, равное 42. После выполнения операции присваивания переменная myInt становится равной 42.

```
myString = "Здравствуй, мир"
```

Здесь переменной myString присваивается строковая (текстовая) величина "Здравствуй, мир". После выполнения операции присваивания переменная myString становится равной "Здравствуй, мир".

```
myObject = myInt
```

Здесь переменной myObject присваивается содержимое переменной myInt. После выполнения операции присваивания переменная myObject становится равной величине, которая содержится в переменной myInt.

```
strMessage = lblMessage.Text
```

Здесь переменной strMessage присваивается содержимое свойства Text объекта lblMessage. После выполнения операции присваивания переменная strMessage становится равной величине, которая содержится в свойстве Text.

```
lblMessage.Text = strMessage
```

Здесь свойству Text объекта lblMessage присваивается содержимое переменной strMessage. После выполнения операции присваивания содержимое свойства Text становится равным содержимому переменной strMessage.

Событие. Событие – это действие, совершенное над объектом, пользователем или каким-либо процессом (например таймером). В ка-

честве события могут быть нажатие указателем мыши на объекте, двойной щелчок, передвижение мыши над объектом, нажатие кнопок клавиатуры, срабатывание таймера и т.д.

Процедура обработки события. Процедура обработки события – это блок кода, который исполняется тогда, когда с объектом программы происходят какие-либо действия (события). Например, при щелчке мышью на объекте **btnWelcome** исполняется процедура обработки события **btnWelcome_Click.** Процедура обработки событий выполняет работу, которую создал в виде текста программы программист.

Метод. Метод – это специальный оператор, который выполняет действие или работу для конкретного объекта программы. В тексте программы для метода используется запись

Object.Method(Value)

где **Object** – это имя объекта, с которым вы работаете; **Method** – это действие, которое вы хотите выполнить; **Value** – это необязательный аргумент, используемый методом. Например, в выражении

```
System.Diagnostics.Process.Start("http://www.rambler.ru/")
```

используется метод **Start**, который запускает браузер с указанной ссылкой. Названия методов и свойств обычно зависят от их расположения в коллекции или библиотеке объектов, поэтому не удивляйтесь, если увидите длинные ссылки, например **System.Drawing.Image.FromFile** (Файл). Подобное название можно прочитать как "метод **FromFile** (Файл), являющийся членом класса **Image**, который, в свою очередь, является членом библиотеки объектов (или пространства имен) **System.Drawing**".

2.16. РАЗЛИЧИЯ МЕЖДУ СВОЙСТВАМИ И МЕТОДАМИ

Рассмотрим две строчки программы:

```
txtUserName.Select()
txtUserName.Text = strUserName
```

В начале каждой строки стоит имя объекта. Дальше стоит оператор «.», который ставится между именем объекта и его свойством либо методом.

Можно сказать, что эти строчки похожи. В обоих случаях вначале **имя объекта**, затем **точка** и только потом следует либо **свойство**, либо **метод**. Как различить, что идет после точки: **метод** или **свойство**?

Во-первых, различить можно исходя из определения, что является свойством, а что методом. **Свойство** – это характеристика объекта, а **метод** – это действие для объекта. **Свойство** является характеристикой, следовательно, если мы переведем название свойства с английского на русский, то должны получить в результате перевода существительное.

Например:

Name – имя;

Text – текст;

Font – шрифт;

ForeColor – передний цвет.

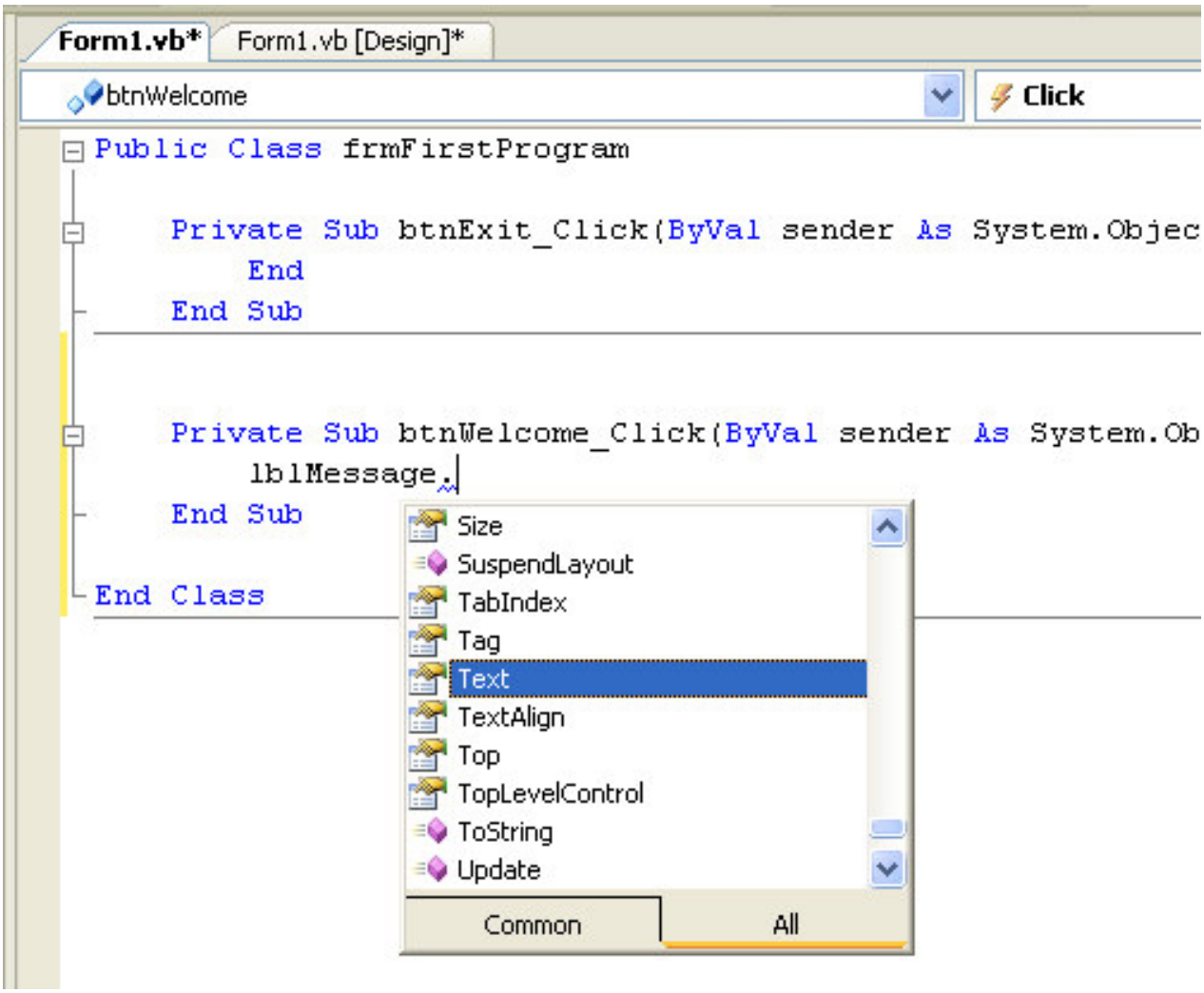
Если **метод** является действием, то с английского на русский название метода переведется как глагол.

Например:,

Select – выделить;

Start – начать, запустить.

Во-вторых, различить можно при написании программы. Когда вы набираете точку после названия объекта, то вам предлагается выбрать свойство или метод, соответствующие данному объекту (рис. 2.25).



Р и с. 2.25. Выбор свойства или метода

Из рис. 2.25 видно, что одни предлагаемые варианты имеют слева значок , а другие – . Значком обозначаются **свойства** объекта, значком – **методы** объекта.

В-третьих, различить можно по наличию в строке программы оператора присвоения (=). Если используется свойство, то оператор присвоения, скорее всего, присутствует в этой строке.

Например:

`strMessage = lblMessage.Text` – переменной strMessage присваивается содержимое свойства Text объекта lblMessage;

`lblMessage.Text = strMessage` – свойству Text объекта lblMessage присваивается содержимое переменной strMessage.

Оператор присваивания, скорее всего, отсутствует в строке программы, где используется метод.

Например:

```
System.Diagnostics.Process.Start("http://www.rambler.ru/")
```

или

```
txtUserName.Select()
```

В-четвертых, методы чаще всего содержат в себе параметры, которые указываются в круглых скобках. Эти параметры могут быть пропущены, но тогда около метода появляются пустые скобки.

2.17. КРАТКАЯ СПРАВКА ПО ТЕМЕ

Краткая справка по теме приведена в табл. 2.5.

Таблица 2.5

Краткая справка по теме

Чтобы	Сделайте следующее
Создать интерфейс пользователя	Для размещения объектов на форме используйте элементы управления из области элементов, а затем установите необходимые свойства этих объектов. Измените размеры формы и объектов на ней
Переместить объект	Поместите мышь над объектом так, чтобы она приняла форму четырех стрелок, направленных в разные стороны, а затем перетащите этот объект
Изменить размер объекта	Щелкните на объекте, чтобы его выбрать, а затем перетащите находящиеся по его сторонам манипуляторы изменения размера
Удалить объект	Щелкните на объекте, а затем нажмите клавишу (Delete)
Открыть Code Editor	Двойным щелчком кнопкой мыши на объекте формы (или на самой форме) выберите форму или модуль в Solution Explorer (Обозреватель решений), а затем щелкните на кнопке View Code (Просмотреть код)
Написать код программы	Введите в редакторе кода строки программы на Visual Basic, ассоциированные с объектом, который вы хотите запрограммировать
Сохранить программу	В меню File (Файл) щелкните на команде Save All (Сохранить все) или на кнопке Save All (Сохранить все) стандартной панели инструментов

Окончание табл. 2.5

Чтобы	Сделайте следующее
Сохранить файл формы	Убедитесь, что форма открыта, а затем щелкните на команде Save (Сохранить) из меню File (Файл) или на кнопке Save (Сохранить) стандартной панели инструментов
Создать .exe-файл	В меню Build (Построение) щелкните на команде Build Solution (Построить решение).
Создать текстовое поле	Щелкните на элементе управления TextBox, а затем нарисуйте это поле
Создать кнопку	Щелкните на элементе управления Button, а затем нарисуйте эту кнопку
Изменить свойство во время выполнения	Измените значение свойства с помощью кода программы. Например: Label1.Text = "Привет!"
Отобразить веб-страницу	Создайте ссылку на эту веб-страницу с помощью элемента управления LinkLabel, а затем откройте ссылку в браузере с помощью метода Process.Start в коде программы.

Контрольные вопросы

1. Как создать новый проект?
2. Что находится на панели ToolBox?
3. Как поместить элемент управления на форму?
4. Что собой представляет элемент управления Button?
5. Как называется на панели ToolBox элемент управления Кнопка?
6. Что собой представляет элемент управления Label?
7. Как называется на панели ToolBox элемент управления Надпись?
8. Можно ли передвигать элементы управления по форме в режиме разработки?
9. Можно ли передвигать элементы управления по форме в режиме отладки программы?
10. Как сделать элемент управления на форме активным?
11. Какой клавишей можно вызвать окно свойств?
12. Как можно понять, что нужный элемент управления на форме активен?
13. Для чего используется свойство Name?
14. Для чего нужно давать осмысленные названия объектам?
15. Что такое префикс?
16. Для чего используется префикс?
17. Что регламентирует «венгерское соглашение»?
18. Какой префикс используется для элемента управления Form (Форма)?

19. Какой префикс используется для элемента управления Label (Надпись)?
20. Какой префикс используется для элемента управления Button (Кнопка)?
21. Что означает frm в имени объекта frmFirstProgram?
22. Что означает btn в имени объекта btnWelcome?
23. Что означает lbl в имени объекта lblMessage?
24. Для чего используется свойство Text?
25. Как изменить текст на объекте формы (элементе управления)?
26. Как в окне свойств изменить редактируемый объект?
27. Для чего используется редактор кода?
28. Как перейти в редактор кода?
29. Что такое процедура?
30. С какого ключевого слова начинается процедура?
31. Какими ключевыми словами заканчивается процедура?
32. При каких обстоятельствах выполняется процедура?
33. Что такое обработчик события?
34. Что такое процедура события?
35. Какое событие обрабатывает процедура Private Sub btnExit_Click?
36. Для чего используется инструкция End?
37. Что такое синтаксис выражения?
38. Что такое стиль программирования?
39. Как выбрать нужный объект в редакторе кода (Code Editor)?
40. Как выбрать нужное событие в редакторе кода (Code Editor)?
41. Для чего используется левый выпадающий список в верхней части редактора кода (Code Editor)?
42. Для чего используется правый выпадающий список в верхней части редактора кода (Code Editor)?
43. Какое событие чаще всего используется для объекта Button (кнопка)?
44. Как правильно прочитать строку программы `lblMessage.Text = "Здравствуй, Мир!"`?
45. Для чего используется знак равенства (=) в строке программы `lblMessage.Text = "Здравствуй, Мир!"`?
46. Для чего используются кавычки (" ") в строке программы `lblMessage.Text = "Здравствуй, Мир!"`?
47. Что такое оператор присвоения?
48. Для чего используется оператор присвоения?
49. Какой тип имеет часть строки, заключенной в кавычки «" "» в строке программы `lblMessage.Text = "Здравствуй, Мир!"`?

50. Что происходит в редакторе кода, когда вы ставите точку после названия объекта lblMessage в строке программы `lblMessage.Text = "Здравствуй, Мир!"`?
51. Для чего используется служба завершения слова?
52. Какое сочетание клавиш используется для ускорения ввода команды?
53. Какое событие обрабатывается процедурой `frmFirstProgram _Load`?
54. Что происходит в строке программы `lblMessage.Text = ""`?
55. Для чего используются комментарии в программе?
56. Как ввести комментарии в программу?
57. Для чего используется знак апострофа (')?
58. Каким цветом в тексте программы обозначаются комментарии?
59. Каким цветом в тексте программы обозначаются ключевые слова?
60. Как запустить программу, созданную в Visual Studio?
61. Для чего нужен значок  на стандартной панели инструментов?
62. Как сохранить проект программы?
63. Для чего нужен значок  на стандартной панели инструментов?
64. Как создать исполняемый файл проекта?
65. Какие типы исполняемых файлов может создавать Visual Studio?
66. Какое расширение имеют исполняемые файлы?
67. Для чего используются отладочные версии исполняемых файлов?
68. Для чего используется окончательная версия исполняемых файлов?
69. Как запустить окончательную версию исполняемого файла?
70. В какой папке хранится окончательная версия исполняемого файла?
71. Для чего используется элемент управления PictureBox?
72. Какой префикс у элемента управления PictureBox?
73. Что означает pic в имени объекта picFace?
74. С помощью какого элемента управления можно вставить рисунок на форму?
75. Для чего используется элемент управления TextBox?
76. Какой префикс у элемента управления TextBox?
77. Что означает txt в имени объекта txtUserName?
78. С помощью какого свойства можно вставить рисунок в PictureBox?
79. Для чего используется свойство Image?
80. Для чего используется свойство SizeMode?
81. Если значение свойства SizeMode установлено как Normal, как это влияет на объект PictureBox?
82. Если значение свойства SizeMode установлено как StretchImage, как это влияет на объект PictureBox?

83. Если значение свойства SizeMode установлено как AutoSize, как это влияет на объект PictureBox?
84. Если значение свойства SizeMode установлено как CenterImage, как это влияет на объект PictureBox?
85. Если значение свойства SizeMode установлено как Zoom, как это влияет на объект PictureBox?
86. Что нужно сделать, чтобы размеры рисунка в объекте PictureBox не зависели от размера объекта PictureBox?
87. Что нужно сделать, чтобы размеры рисунка в объекте PictureBox изменялись до размеров объекта PictureBox, при этом пропорциональность рисунка не соблюдалась?
88. Что нужно сделать, чтобы размеры рисунка в объекте PictureBox не зависели от размера объекта PictureBox, а сам рисунок центрировался внутри объекта PictureBox?
89. Что нужно сделать, чтобы размеры рисунка в объекте PictureBox изменялись до размеров объекта PictureBox, при этом пропорциональность рисунка соблюдалась?
90. Для чего используется свойство Visible?
91. Что нужно сделать, чтобы объект формы становился невидимым во время работы программы?
92. Какое значение свойства Visible делает объект видимым?
93. Для чего используется метод Select?
94. Какой метод позволяет сделать нужный объект активным во время работы программы?
95. Что такое метод?
96. Какая инструкция объявляет переменные?
97. Для чего используется инструкция Dim?
98. Что такое переменная?
99. Что означает ключевое слово String в строке программы `Dim strUserName As String`?
100. Что содержат в себе строковые переменные?
101. Для чего нужен знак (`&`) в строке программы `lblMessage.Text = "Здравствуйте, " & strUserName & "!"`?
102. Что такое конкатенация?
103. Как объединить две текстовые строки?
104. Для чего используется объект DateTimePicker?
105. Какой префикс у элемента управления DateTimePicker?

106. Что означает dtp в имени объекта dtpBirthDay?
107. Для чего используется функция MsgBox?
108. Для чего используется свойство Text в строке программы `MsgBox("Ваш день рождения " & dtpBirthDay.Text)`?
109. Для чего используется свойство Text в строке программы `MsgBox("День года: " & dtpBirthDay.Value.DayOfYear.ToString)`?
110. Для чего используется метод ToString в строке программы `MsgBox("День года: " & dtpBirthDay.Value.DayOfYear.ToString)`?
111. Для чего используется свойство Text в строке программы `MsgBox("День года: " & dtpBirthDay.Value.DayOfWeek.ToString)`?
112. Как узнать с помощью объекта календаря (DateTimePicker) число дней, прошедших с начала года от даты, указанной в календаре?
113. Как узнать с помощью объекта календаря (DateTimePicker) число дней, прошедших с начала недели от даты, указанной в календаре?
114. Как перевести свойство, содержащее числовую величину, в строковой тип?
115. Что собой представляет числовое значение типа Integer?
116. Какого типа данные отображает функция MsgBox?
117. Какое свойство объекта DateTimePicker нужно использовать, чтобы отображать на календаре время вместо даты?
118. Для чего используется свойство Format в объекте DateTimePicker?
119. Для чего используется объект LinkLabel?
120. Какой префикс у элемента управления LinkLabel?
121. Что означает lnk в имени объекта lnkRambler?
122. Какое событие обрабатывается процедурой `lnkRambler_LinkClicked`?
123. Для чего используется свойство LinkVisited в объекте LinkLabel?
124. Что нужно сделать, чтобы показать, что ссылка (объект LinkLabel) была посещена?
125. Для чего используется строка программы `System.Diagnostics.Process.Start("http://www.rambler.ru/")`?
126. Для чего используется метод Start класса Process?
127. Что такое инструкция?
128. Что такое элемент управления?
129. Является ли элемент управления объектом?
130. Что такое свойство объекта?
131. Как в программе присвоить свойству какое-либо значение?

132. Какой оператор используется для операции присваивания?
133. Какая часть строки программы `variable = value` (правая или левая от оператора присвоения) получает присваиваемую величину?
134. В строке программы `lblMessage.Text = strMessage` свойству Text присваивается значение переменной strMessage или переменой strMessage присваивается значение, содержащееся в свойстве Text?
135. В строке программы `strMessage = lblMessage.Text` свойству Text присваивается значение переменной strMessage или переменой strMessage присваивается значение, содержащееся в свойстве Text?
136. Что такое событие?
137. Что такое процедура обработки события?
138. Что такое метод?
139. Как в тексте программы отличить метод от свойства?
140. Название свойства переводится с английского как существительное или как глагол?
141. Название метода переводится с английского как существительное или как глагол?
142. Что означает значок  при выборе метода или свойства при написании кода программы?
143. Что означает значок  при выборе метода или свойства при написании кода программы?
144. В строке программы `strMessage = lblMessage.Text` наличие оператора присваивания является признаком использования свойства или метода?
145. В строке программы `txtUserName.Select()` наличие скобок является признаком использования свойства или метода?

ЗАКЛЮЧЕНИЕ

Были рассмотрены основы создания программ в Visual Studio на языке Visual Basic: запуск Visual Studio, использование окон инструментов, создание интерфейса пользователя, работа с объектами, их свойствами, методами и событиями, написание кода программы.

Естественно, что в рамках настоящего пособия была рассмотрена лишь малая часть возможностей программирования в Visual Studio. Оно и не претендует на полноту охвата и было направлено на то, что-

бы показать, как начать работать в среде проектирования программ и что создавать программы совсем не сложно.

Хочется обратить внимание, что, имея под рукой такой инструмент, как Visual Basic, инженер может решать огромный комплекс задач. При этом на изучение этого языка программирования не придется тратить большое количество времени, поскольку он прост в освоении. В дальнейшем затраченное время окупится за счет ускорения решения рутинных задач, а высвобожденное время уйдёт на решение других задач.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Хальворсон, Майкл.* Microsoft Visual Basic.NET 2003: Пер. с англ. / *Майкл Хальворсон*. – М.: ЭКОМ, 2004. – 672 с.
2. *Ляхович, В.Ф.* Основы информатики /*В.Ф. Ляхович, С.О. Крамаров*. – Ростов н/Д: Феникс, 2005.
3. *Семакин, И.Г.* Информационные системы и модели / *И.Г. Семакин, Е.Г. Хеннер.* – М.: Бином; Лаборатория знаний, 2005. – 303 с.
4. *Мозговой, М.В.* Занимательное программирование: Самоучитель / *М.В. Мозговой*. – СПб.: Питер, 2005. 208 с.
5. *Угринович, Н.Д.* Информатика и информационные технологии / *Н.Д. Угринович.* – М.: Бином; Лаборатория знаний, 2005. – 512 с.
6. *Шевякова, Д.А.* Самоучитель Visual Basic 2005 / *Д.А.Шевякова, А.М. Степанов, Р.Г. Карпов*. СПб.: БХВ-Петербург, 2006. – 576 с.
7. Visual Basic .Net. Справочник программиста: практ. пособ.: пер. с англ. – М.: ЭКОМ, 2002. – 352 с.
8. *Мак-Манус, П.* Обработка баз данных на Visual Basic .NET: пер. с англ. / *Мак-Манус, П., Джеффри Голдштейн, Джеки Прайс, Т. Кевин*. – 3-е изд. – М.: Издат. дом «Вильямс», 2003. – 416 с.
9. *Долженков, В.* Visual Basic.NET: учеб. курс / *В. Долженков, М. Мозговой*. – СПб.: Питер, 2003. – 464 с.
10. *Франклин, Кит.* VB.NET для разработчиков: Пер. с англ./ *Кит Франклин.* – М. : Издат. дом «Вильямс», 2002. – 272 с.
11. *Понамарев, В.А.* Visual Basic .NET: Экспресс-курс / *В.А. Понамарёв*. – СПб.: БХВ-Петербург, 2003. – 304 с.
12. *Вонг, Уоллес.* Visual Basic .NET для «чайников»: Пер. с англ./ *Уоллес Вонг.* – М. : Издат. дом «Вильямс», 2002. – 336 с.
13. *Фокселл, Джеймс Д.* Освой самостоятельно Visual Basic .NET за 24 часа: Пер. с англ./ *Джеймс Д. Фокселл.* – М.: Издат. дом «Вильяме», 2002. – 416 с.
14. *Орлов, А.* VBA для тех, кто любит думать / *А. Орлов*. – М.: Солон, 2002. – 240 с.
15. *Васильков, Ю.В.* Компьютерные технологии вычислений в математическом моделировании: учеб. пособ./ *Ю.В. Васильков, Н.Н. Василькова.* – М.: Финансы и статистика, 2002. – 256 с.
16. *Минеев, С.П.* VBA: Создание простейших программ, создание отчетов в документе Word: учеб. пособ. / *С.П. Минеев*. – Самара: СамГТУ, 2004.

17. *Минеев, С.П.* VBA: управление ходом программы, действия над массивами данных: учеб. пособ. / *С.П. Минеев, П.В. Тулупов, Ю.А. Макаричев, Ю.А. Кальянова.* – Самара: СамГТУ, 2005.
18. *Минеев, С.П.* Основы программирования в AutoCAD. Технологии Activex Aautomation и VBA в среде проектирования AutoCAD для решения задач электромеханики: учеб. пособ. / *С.П. Минеев*. – Самара: СамГТУ, 2007. – 78 с.

ОГЛАВЛЕНИЕ

Учебное издание

МИНЕЕВ Сергей Петрович
ЧЕБОТКОВ Эдуард Галактионович

Информатика. Основы программирования на Visual Basic в среде разработки Visual Studio 2005

Редактор С. И. К о с т е р и н а
Компьютерная вёрстка И.О. Миняева
Выпускающий редактор Н.В. Беганова

Подп. в печать 21.11.15.
Формат 60×84 1/16. Бумага офсетная.
Усл. п.л.5,60. Уч.-изд. л. 5,58.
Тираж 100 экз. Рег. № 280.

Государственное образовательное учреждение
высшего профессионального образования
«Самарский государственный технический университет»
443100. г. Самара, ул. Молодогвардейская, 244. Главный корпус

Отпечатано в типографии
Самарского государственного технического университета
443100. г. Самара, ул. Молодогвардейская, 244. Корпус №8