**Учреждения НПО, СПО**

**Учебно-методическое пособие** «Программирование в объектно-ориентированной среде»

**Автор: Солдатенко Мария Николаевна,** преподаватель специальных дисциплин ОГАОУ СПО «Белгородский индустриальный колледж».

**Рецензент: Старовойтов А.С.,** доцент кафедры естественно-математического образования и информационных технологий, кандидат физико-математических наук ОГАОУ ДПО БелИРО.

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

Основы программирования – это дисциплина, целью которой является изучение методов программирования для овладения знаниями в области технологии программирования; подготовка к осознанному использованию как языков программирования, так и методов программирования.

Программирование в объектно–ориентированной среде - это важный раздел дисциплины «Основы программирования», главной целью которого является формирование системы понятий, знаний, умений и навыков в области современного программирования, включающего в себя методы проектирования, анализа и создания программных продуктов, основанные на использовании объектно-ориентированной методологии. Современная объектно-ориентированная среда программирования позволяет легко создавать полноценные Windows-приложения. Это не только полезное, но и очень интересное занятие.

Учебно-методическое пособие «Программирование в объектно–ориентированной среде» по дисциплине «Основы программирования» предназначено для студентов 2 курса, обучающихся по специальности 230115 Программирование в компьютерных системах. Пособие составлено с учетом требований ФГОС СПО третьего поколения.

Актуальность составления данного пособия обусловлена необходимостью учебно-методического обеспечения дисциплины «Основы программирования», а также тем, что объектно-ориентированный подход к программированию является одной из ведущих и интенсивно развивающихся технологий программирования. При составлении пособия в течении учебного года проводились такие работы, как подбор теоретического материала в соответствии с программой дисциплины, создание методических указаний для выполнения лабораторных работ, создание и тестирование программных продуктов, анализ результатов и работа над ошибками.

Пособие направлено на повышение мотивации учащихся к изучению дисциплины «Основы программирования», развитие гибкого логического и пространственного мышления учащихся, развитие профессиональных компетенций учащейся молодежи.

В данном пособии предлагаются 10 лабораторных работ по разделу «Программирование в объектно–ориентированной среде».

Предполагается, что учащиеся уже ознакомлены с основами структурного программирования, синтаксисом языка программирования, знают основные принципы объектно-ориентированного программирования.

Лабораторные работы, приведенные в учебно-методическом пособии, включают в себя следующие элементы: название темы, цель занятия, ход работы, теоретическую часть, практическую часть (указания по выполнению) и контрольные вопросы.

В теоретической части приводится структурированный материал, необходимый для подготовки студентов к лабораторным работам. Практическая часть представляет собой логично выстроенный ход работы, включающий задания, этапы выполнения работы и примеры реализации программного кода. Вопросы к каждой лабораторной работе позволяют провести контроль и самоконтроль учащихся.

В учебно-методическом пособии содержатся задания для освоения основных приемов работы в объектно-ориентированной среде: работа с интерфейсом интегрированной среды Microsoft Visual Studio 2010, создание экземпляров классов, объявление класса, создание наследованного класса, перегрузка методов, создание проектов с использованием разнообразных визуальных компонентов, разработка оконного приложения с несколькими формами, разработка многооконного приложения.

Учебно-методическое пособие содержит лабораторные работы, которые обеспечивают формирование базовых умений и навыков разработки и отладки приложений, раскрывают практические вопросы принципов объектно-ориентированного программирования.

В лабораторных работах, приведенных в пособии, содержатся как задания с подробными указаниями к выполнению, так и задания без алгоритма работы. Таким образом, предлагаемые практические работы можно использовать для индивидуальной работы студентов на занятиях под руководством преподавателя, а также для самостоятельной работы обучающихся.

**Учебно-методическое пособие**

**Программирование в объектно–ориентированной среде**

по дисциплине **«Основы программирования»**

для студентов 2 курса СПО

Специальность **230115 Программирование в компьютерных системах**

Автор-составитель:

преподаватель специальных дисциплин ОГАОУ СПО «Белгородский индустриальный колледж»

Солдатенко Мария Николаевна

Белгород 2014

Рассмотрено

предметно-цикловой комиссией

Протокол заседания № \_\_\_\_

От «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2013 г.

Председатель цикловой комиссии

\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Шершнева М.А.

**Рецензент:**

Шершнева М.А., председатель предметно-цикловой комиссии информатики и ПОВТ, преподаватель специальных дисциплин ОГАОУ СПО «Белгородский индустриальный колледж».

**Составитель:**

Преподаватель специальных дисциплин ОГАОУ СПО «Белгородский индустриальный колледж» Солдатенко М.Н.

Учебно-методическое пособие представляет собой сборник лабораторных работ по программированию в объектно-ориентированной среде. Пособие предназначено для студентов второго курса специальности 230115«Программирование в компьютерных системах» для работы на занятиях по дисциплине «Основы программирования» под руководством преподавателя.

**СОДЕРЖАНИЕ**

|  |  |
| --- | --- |
|  |  |
| Лабораторная работа № 1. Изучение интегрированной среды разработчика. Создание простого проекта………………………………. | **5** |
| Лабораторная работа № 2. Создание экземпляров класса. Объявление класса………………………………………………………………………… | **15** |
| Лабораторная работа № 3. Создание наследованного класса…………… | **23** |
| Лабораторная работа № 4. Перегрузка методов………………………… | **31** |
| Лабораторная работа № 5. Создание проекта с использованием кнопочных компонентов………………………………………………….. | **36** |
| Лабораторная работа № 6. Создание проекта с использованием компонентов для работы с текстом……………………………………… | **44** |
| Лабораторная работа № 7. Создание проекта с использованием компонентов ввода и отображения чисел, дат и времени……………… | **54** |
| Лабораторная работа № 8. Создание проекта с использованием компонентов стандартных диалогов и системы меню…………………. | **63** |
| Лабораторная работа № 9. Разработка оконного приложения с несколькими формами……………………………………………………. | **79** |
| Лабораторная работа № 10. Разработка многооконного приложения... | **93** |
| Список использованных источников…………………………………….. | **99** |

Лабораторная работа № 1

**Изучение интегрированной среды разработчика. Создание простого проекта**

**Цель:** изучить интегрированную среду разработчика Visual Studio.NET, приобрести навыки работы с интегрированной средой разработчика в процессе создания простого проекта.

## Ход работы:

## Изучить теоретическую часть.

## Выполнить задания, следуя указаниям.

## Ответить на контрольные вопросы.

## Предъявить преподавателю результаты работы программы и исходные коды.

## Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, программный код, результаты тестирования программы, выводы).

**🕮Теоретическая часть**

**Интегрированная среда разработки Visual Studio.Net**

Интегрированная среда разработки (integrateddevelopmentenvironment,IDE) Visual Studio.NET, едина для всех языков программирования .NET от Microsoft. Таким образом, какой бы тип проекта вы ни создавали (ATL, MFC,C#, Visual Basic.NET, FoxPro, стандартный C++ и т. п.), вы все равно будете работать в одной и той же среде.

При запуске интегрированной среды разработки Visual Studio.NET на экране появляется ее окно, в котором предлагается выбрать язык программирования. После этого на экране появляется главное окно, основными элементами которого являются:

***строка заголовка*** содержит название программы и имя открытого файла, а также кнопки свертывания, восстановления и закрытия;

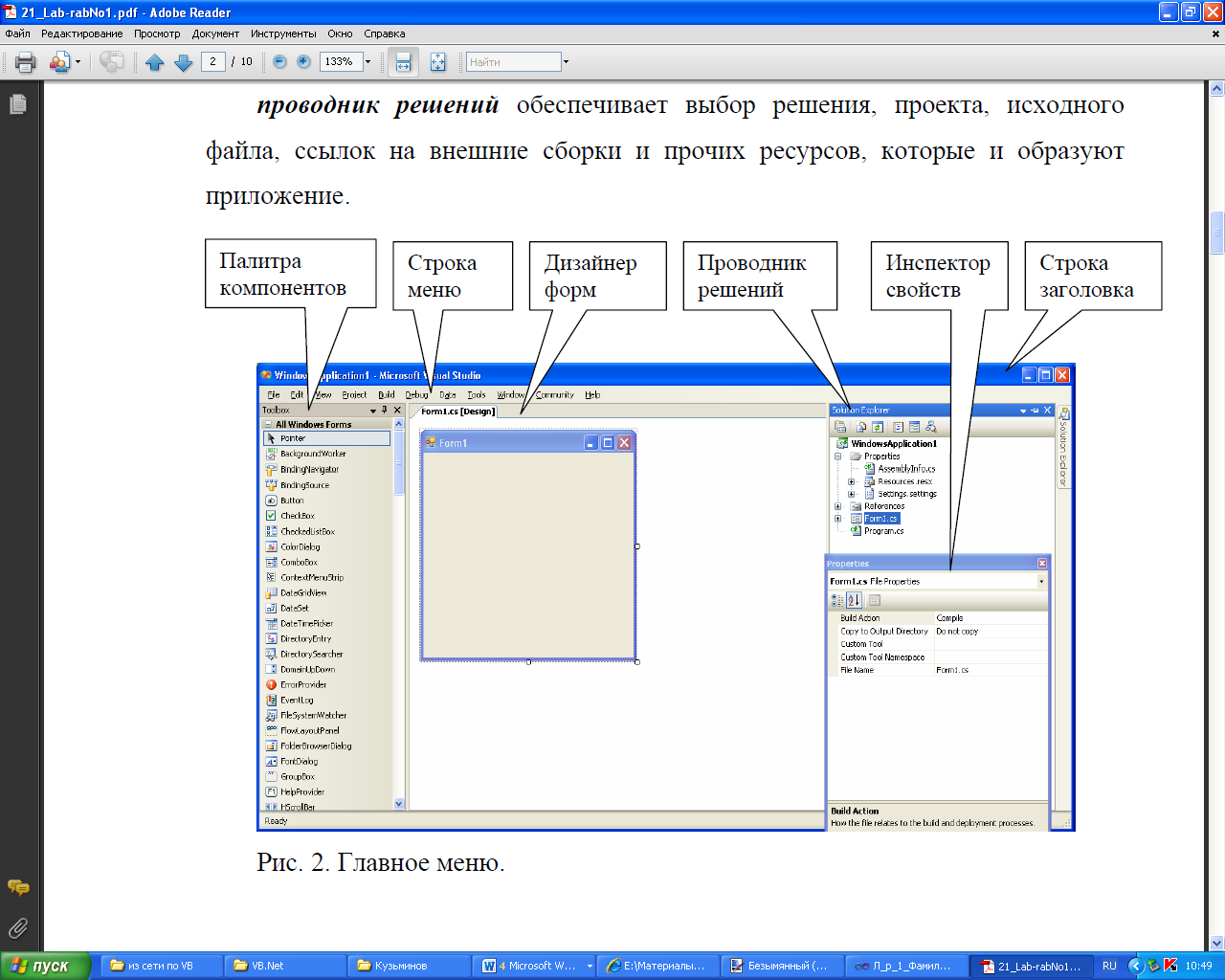
***строка меню***, которая представляет доступ ко всем функциям и командам программы;

***панель элементов (палитра компонентов)***она состоит из нескольких закладок, на которых располагаются визуальные компоненты, используемые при создании программ;

***окно свойств*** в этом окне производится настройка основных свойств визуальных компонентов;

***конструктор (дизайнер) форм*** содержит название и кнопки управления окном;

***обозреватель решений*** обеспечивает выбор решения, проекта, исходного файла, ссылок на внешние сборки и прочих ресурсов, которые и образуют приложение.



Меню и команды интегрированной среды VisualStudio 2010

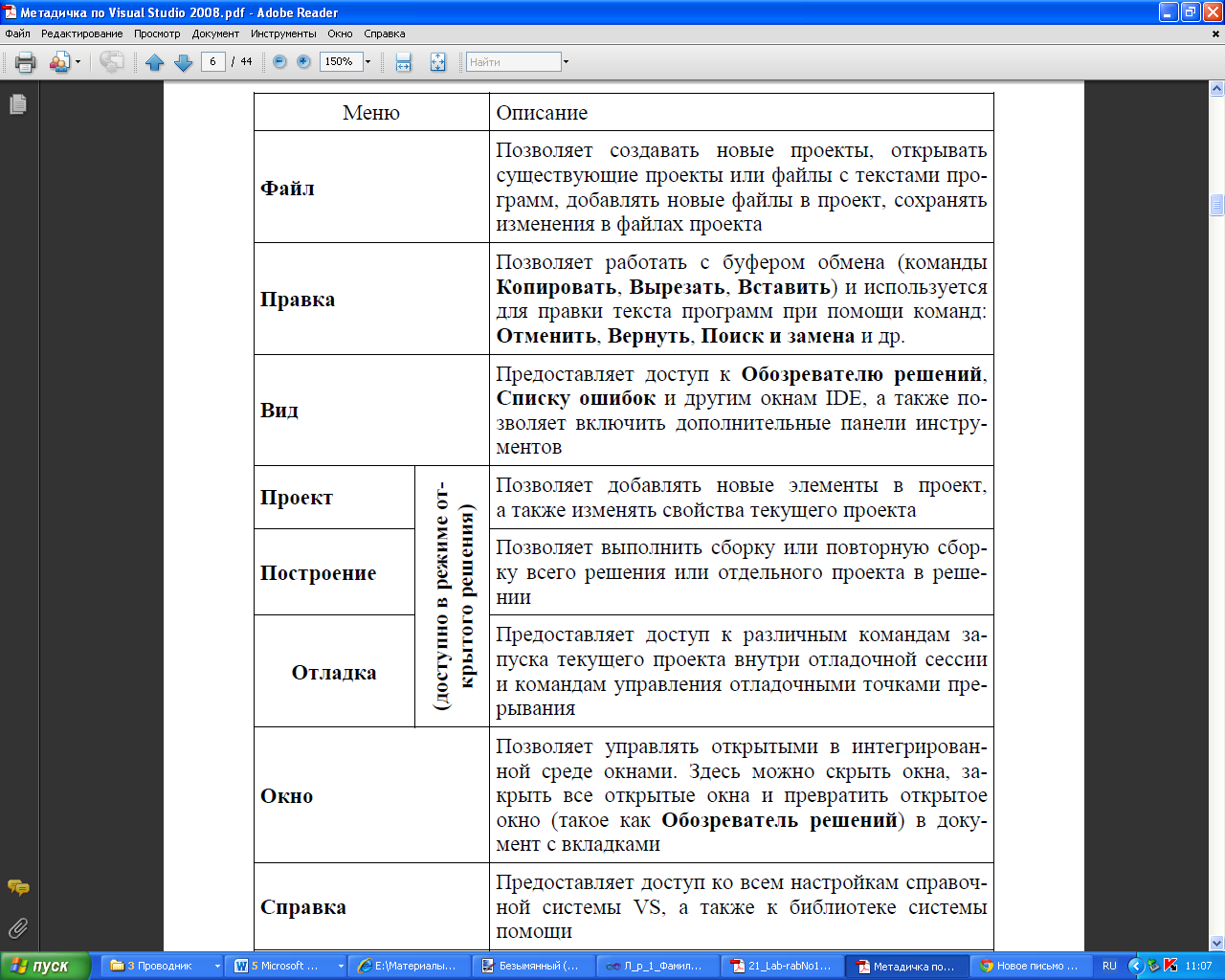
Чтобы выполнить команду в среде **Visual Studio**, можно воспользоваться тремя основными способами:

− с помощью главного меню,

− с помощью панелей инструментов (их видимость настраивается в меню **Вид /Панели инструментов** (**View \ Toolbars)**),

− с помощью контекстного меню, которое, как обычно, активизируется при нажатии правой кнопки мыши.

Меню Visual Studio 2010



Панели инструментов и палитра компонентов. Окно свойств

Панели инструментов скрываются или отображаются с помощью меню **Вид/Панели инструментов** (**View\Toolbars)**. Настройка панелей инструментов осуществляется с помощью команды **Вид/Панели инструментов** /Настройка (**View\Toolbars\Customize)**.

Окно **Панель элементов (Toolbox)** представляет собой палитру компонентов, распределённых по категориями:

*Все формы WindowsForms (AllWindowsForms)*– общий список компонентов,

*Commo nControl***s**– элементы управления общего назначения и т.д. Для добавления на форму элемента управления существует несколько путей:

− щёлкнуть мышью на нужном элементе управления в окне **Панель элементов (Toolbox)**, затем растянуть на форме рамку, которая и определит его размеры;

− дважды щёлкнуть на элементе управления в окне **Панель элементов (Toolbox)**, и он будет добавлен на форму, причем размеры и положение **Visual Studio** определит автоматически (по умолчанию);

− щёлкнуть мышью на нужном элементе управления, затем один раз щёлкнуть на форме.

Элемент будет добавлен так, чтобы его верхний левый угол совпадал с координатами курсора мыши, а размер будет задан по умолчанию. Если при этом всё время держать нажатой клавишу **Ctrl**, будут добавлено несколько экземпляров этого элемента управления (при каждом щелчке мыши).

Компоненты, не используемые для непосредственного создания пользовательского управления (невизуальные компоненты), добавляются аналогично, но вне зависимости от указанного места всегда располагаются в специальной области под формой (панели компонентов). При этом их местоположение не имеет значения.

**Окно свойств (Properties)** служит для установки свойств компонента. На вкладке С**войства (Properties)** редактируются свойства компонентов, на вкладке **События** (**Event)** задаются обработчики событий. В окне **Окно свойств (Properties)** отображены не все свойства компонентов, а лишь те, которые могут быть установлены на этапе проектирования. Все изменения в окне **Окно свойств (Properties)** вызывают изменения исходного

кода программы (для формы **Form1** – файла **Form1.Designer.cs**).

Работа с формами

Проектирование форм – ядро визуальной разработки в среде **Visual Studio.** Каждый помещаемый на форму элемент управления или любое задаваемое свойство вносит изменения в исходный код файла, связанного с формой.

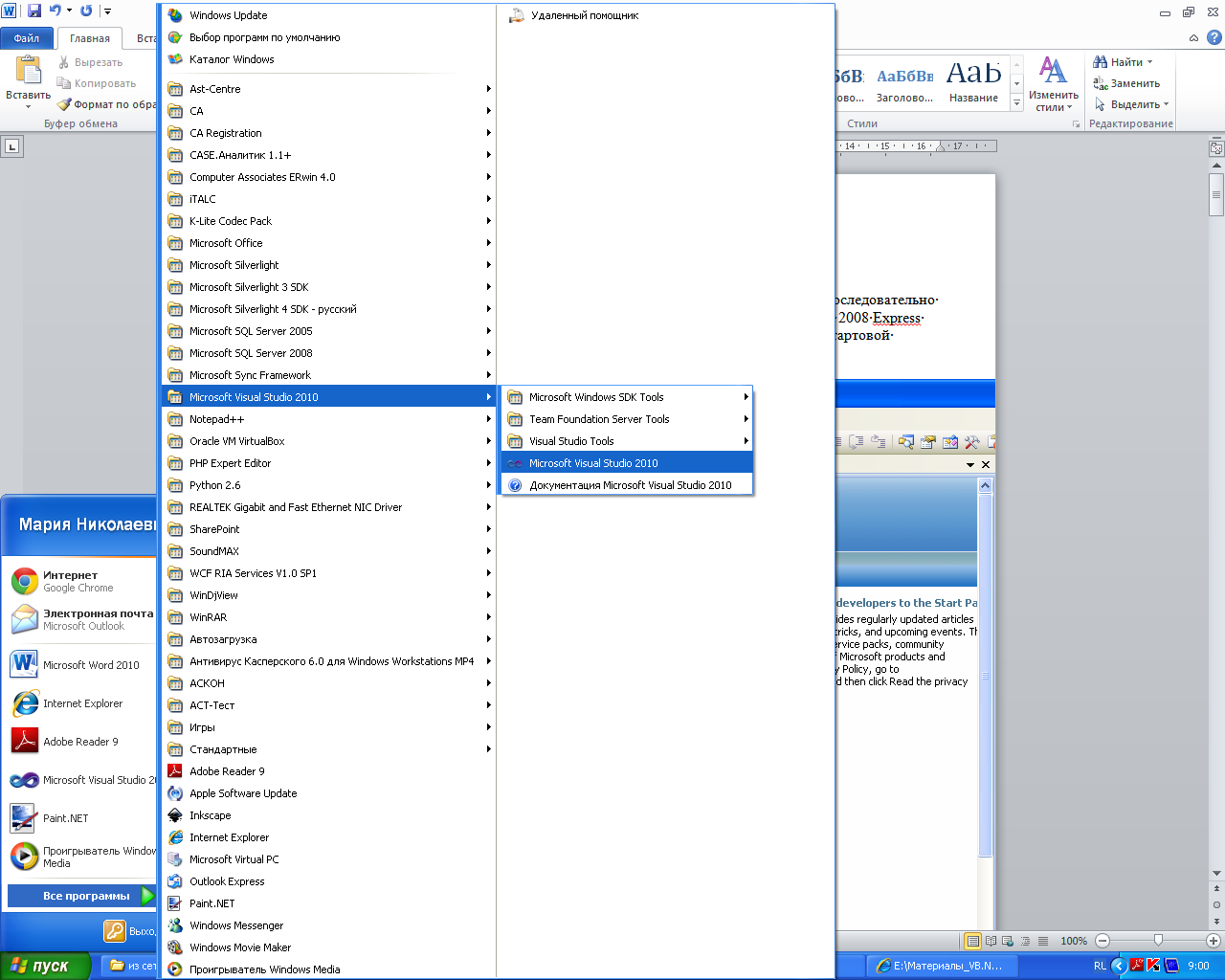
Можно начать новый проект, создав пустую форму, или начать с существующей формы или добавить в проект новые формы. Проект может иметь любое число форм, кроме того, возможно динамическое создание любого числа форм во время выполнения программы.

При работе с формой на этапе проектирования можно изменять её свойства, свойства одного или нескольких элементов управления одновременно. Чтобы выбрать форму или элемент управления, можно просто щёлкнуть по нему мышью. Возможен выбор нескольких элементов управления – с помощью растягивающейся рамки либо с помощью комбинации клавиш **Ctrl**+щелчок левой кнопкой мыши. Для управления взаимным расположением элементов управления используется выпадающее меню **Формат (Format)**.

В интегрированной среде разработки Visual Studio.NET проекты логически организуются в ***решения*** (solutions). Каждое решение состоит из одного или нескольких проектов. В свою очередь, каждый проект может состоять из любого количества исходных файлов, ссылок на внешние сборки и прочих ресурсов, которые и образуют приложение. Вы сможете открыть любой изданных ресурсов с помощью окна Solution Explorer - Проводника решений

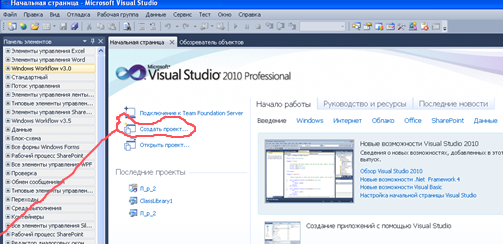
**Задание 1.** Создать программу, в которой щелчок на кнопке приводит к изменению цвета формы и выводит на экран окно с сообщением "Привет, Мир"

**Запуск и настройка VisualStudio .NET**



1. Щелкните на кнопке *Пуск.*
2. Выберите пункт *Программы.*
3. Выберитепункт*Microsoft Visual Studio 2010.*

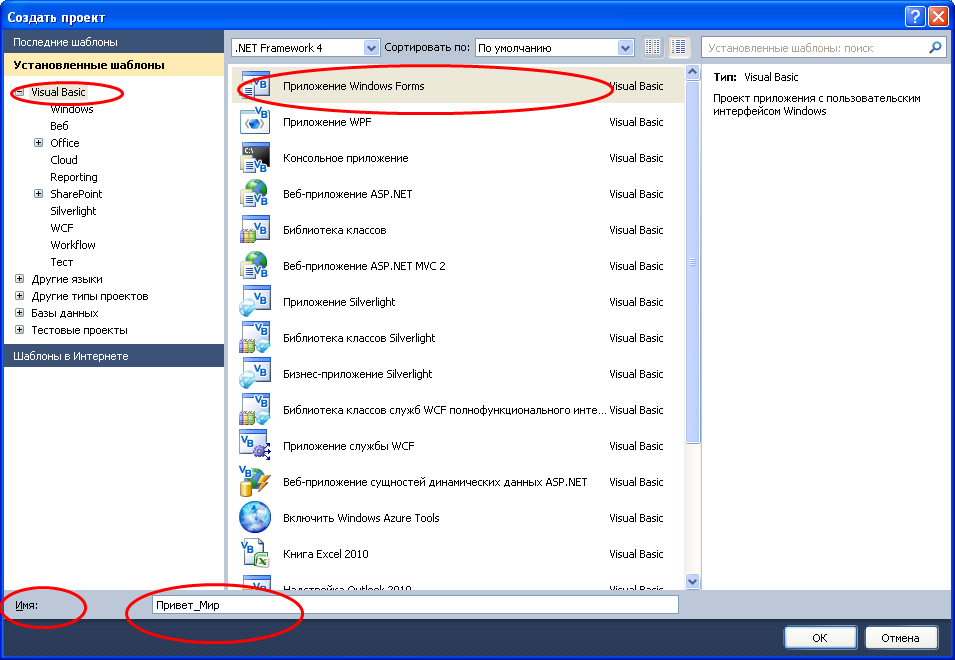
Первое, что вы увидите на экране, запустив Visual Studio .NET, — начальную страницу *{Начальная страница).*По умолчанию будет открыта вкладка *Проекты.* Обычно на ней приводится список недавно открывавшихся проектов*.* Для открытия существующего про­екта щелкнуть по кнопке *Открыть проект,* а для начала создания нового проекта — щелкнуть по кнопке *Создать проект.*



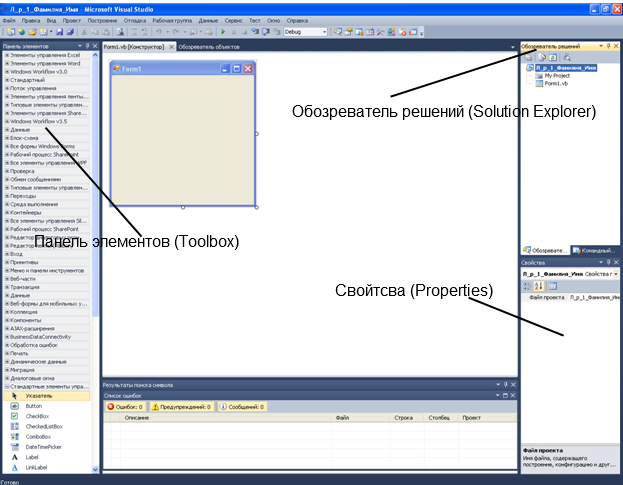
**Создание первого проекта**

Щелкните на кнопке *Создать проект,* чтобы создать новый проект. Откроется диалоговое окно *Создать проект.*

1. В окне *Создать проект* вы увидите, что в левой части, в панели *Типы проектов:* перечислено мно­жество типов проектов, которые можно создавать в Visual Studio .NET. С некоторыми из этих типов вы вскоре познакомитесь. Проект «Привет, Мир» будет приложением Windows, поэтому в правой части окна (панель *Шаблоны)* необходимо выбрать шаблон *Приложение Windows.* В окне *Создать проект* нуж­но выбрать не только тип проекта и шаблон, но и имя проекта и папку, в которой будут размещаться его файлы.
2. Выберите тип проекта *Проекты- VisualBasic*в панели *Типы проектов:.*
3. Выберите шаблон *Приложение Windows*в панели *Шаблоны:*



1. Введите *Привет\_Мир* в качестве имени проекта в текстовом поле *Имя* (в имени проекта лучше не использовать пробелы).
2. Нажмите кнопку *ОК.* Откроется новый созданный проект, и на экране будут отображены форма Forml.vb (в левой части) и Обозреватель решений (Solution Explorer) (в правой части) и др.



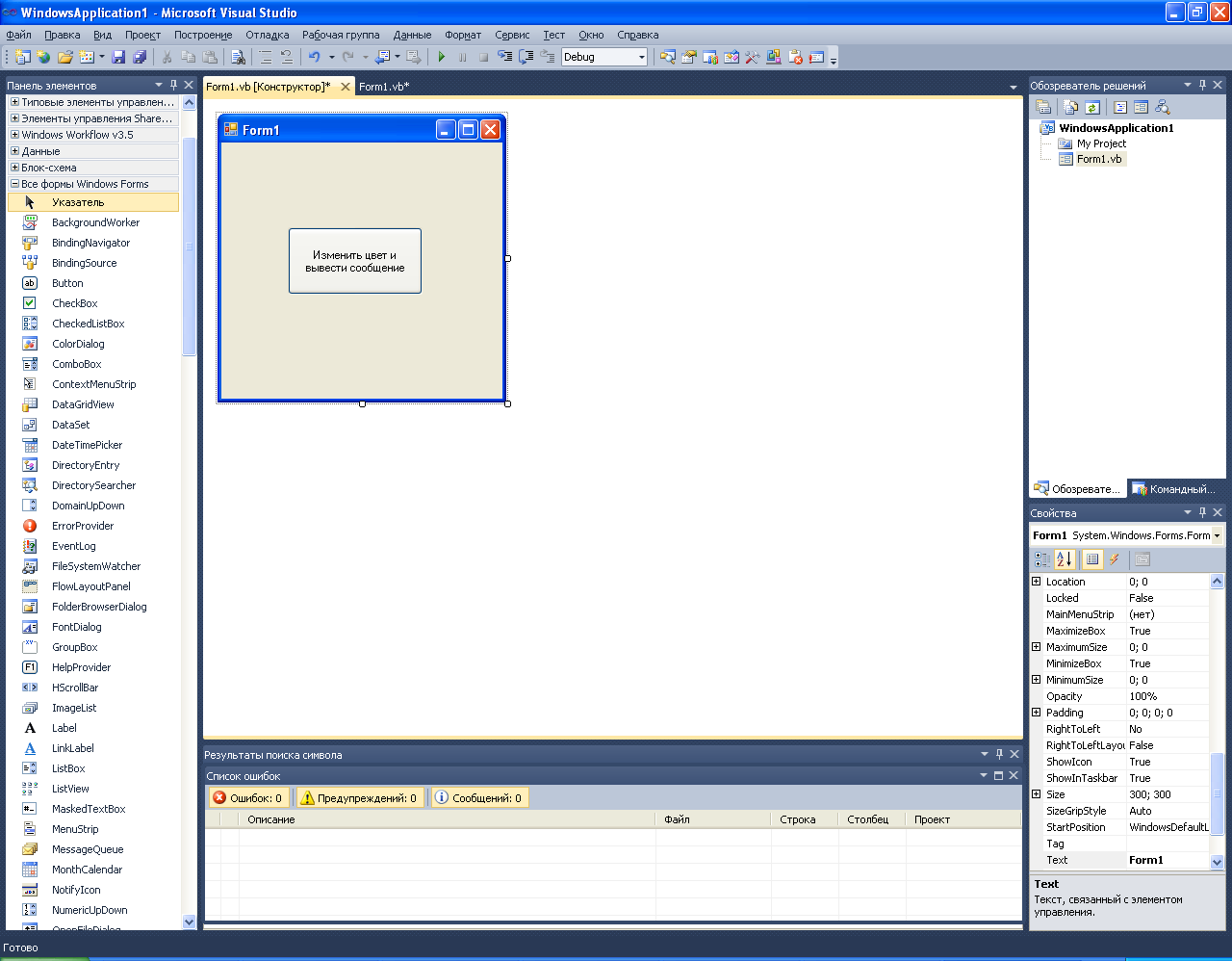
**Конструирование графического интерфейса проекта**

**Окно *Конструктор форм (FormDesignWindow).***

Форма — это элемент графического интерфейса, с помощью которого осу­ществляется взаимодействие пользователя с програм­мой. Форма представляет собой поле с размещаемыми на нем объектами разных типов — кнопками, тексто­выми полями. В основной рабочей области в левой части окна будет отображена пустая форма, на которой нет кно­пок и других объектов. Эта область называется ок­ном *Конструктор форм.* Ярлычок этого окна содер­жит надпись *Forml.vb [Конструктор].*

**Окно *Область элементов (Toolbox).*** В этом окне содержатся все объекты, которые можно поместить на форму, — кнопки, переключатели, текстовые поля, выпадаю­щие списки и т. д.

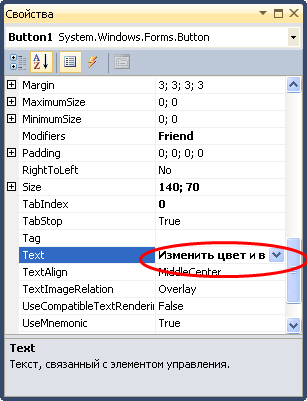
1. Откройте меню *Вид (View)* в строке меню.
2. Выберите пункт *Панель элементов (Toolbox)* — откроется окно *Панель элементов.*



1. Двойным щелчком по пункту *Button*поместите на форму кнопку.

**Перемещение кнопки и изменение ее размеров.**

1. Подведите курсор мыши к кнопке, нажмите левую кнопку мыши и, удерживая ее нажатой, перетащите кнопку в центр формы, после чего отпустите кнопку мыши.
2. Чтобы изменить размер кнопки, подведите курсор мыши к одному из белых квадратиков вокруг кнопки (это метки изменения размера).



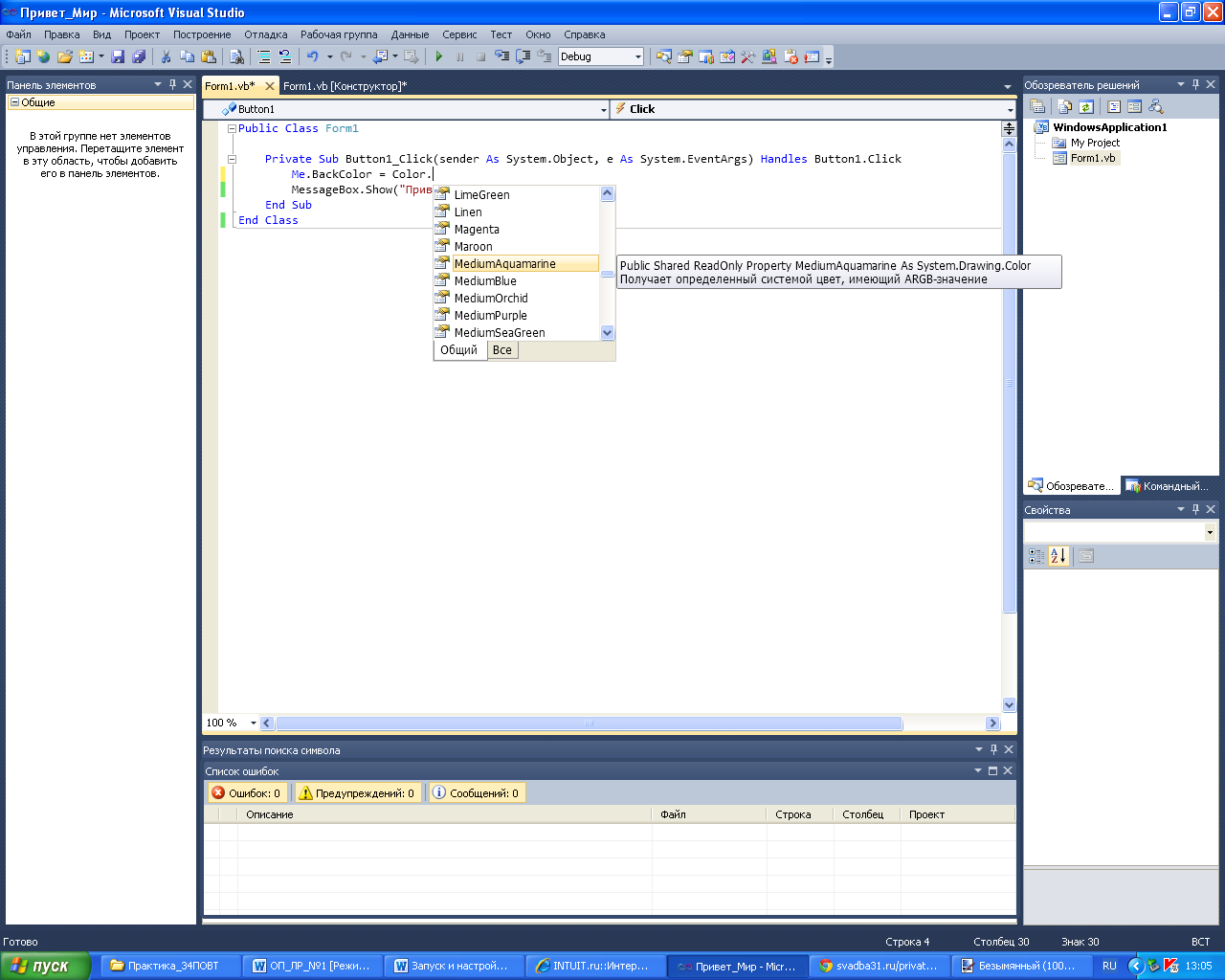
1. Нажмите левую кнопку мыши, и, удерживая ее нажатой, перемещайте метку, чтобы растянуть или сжать кнопку до требуемого размера.
2. В окне Свойства (Propertys) измените свойство *Text* кнопки *Button1* на *Изменить цвет и вывести сообщение*

**Создание программного кода проекта**

1. Сделайте двойной щелчок по кнопке Buttonl. Откроется окно *Редактор кода.*
2. Щелкните по пустой строке над строкой EndSub.
3. Введите в пустую строку с клавиатуры строку кода, которая вызывает процедуру обработки события (точно так, как показано ниже):



**Совет**. После того, как вы наберете имя объекта Color и точку, Visual Studio покажет список всех свойств объекта текстовое поле. Вы можете выбрать свойство из этого списка, дважды щелкнув на нем мышью, или набрать его самостоятельно.

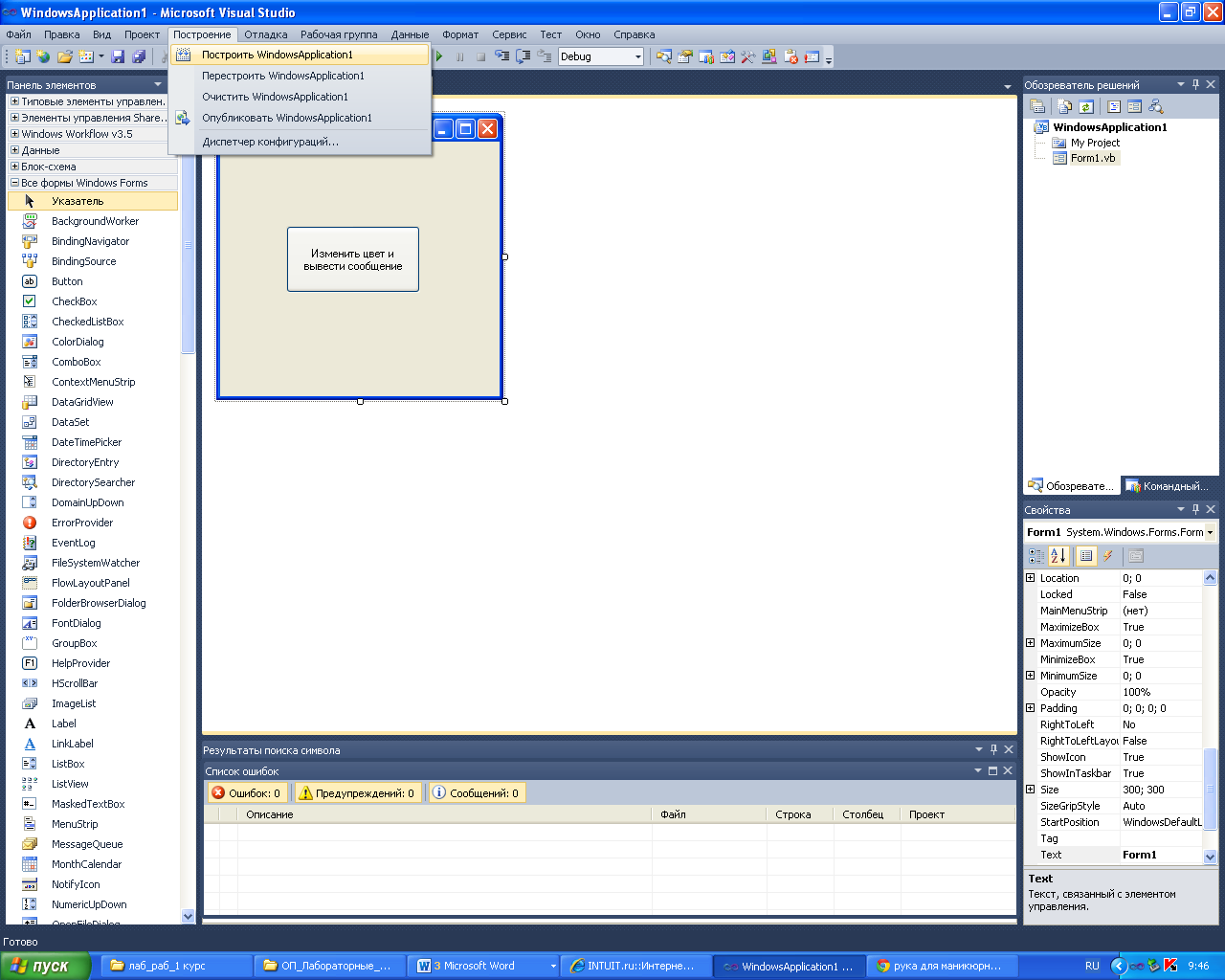


Этот код будет выполняться, когда пользователь будет щелкать по кнопке на форме и генерировать событие.

**Построение решения**

Теперь нужно построить решение. При построении решения код, написанный вами и Visual Studio.NET, компилируется в понятные компьютеру инст­рукции.

1. Откройте меню *Построение (Build)* в строке меню.



1. Выберите пункт *Построить решение (Build Solution)* — начнется построение решения.

Процесс построения решения будет отображаться в окне *Вывод.* Если в программном коде не было сделано ошибок, после завершения построения в окне *Вывод* будет выведено сообщение о том, что по­строение выполнено успешно.

В программном коде могли быть сделаны ошиб­ки, пусть, например, вместо MessageBox вы набрали mMessageBox. Если VisualBasic может определить, где произошла ошибка, он подчеркивает это место в программном коде синей волнистой линией. Кроме того, открывается окно *Список задач (TaskList),* со­держащее список ошибок, которые нужно испра­вить. Двойной щелчок по сообщению об ошибке перемещает курсор в коде к этой ошибке.

В окне *Вывод (Output)* появится сообщение о том, что построение выполнить не удалось.

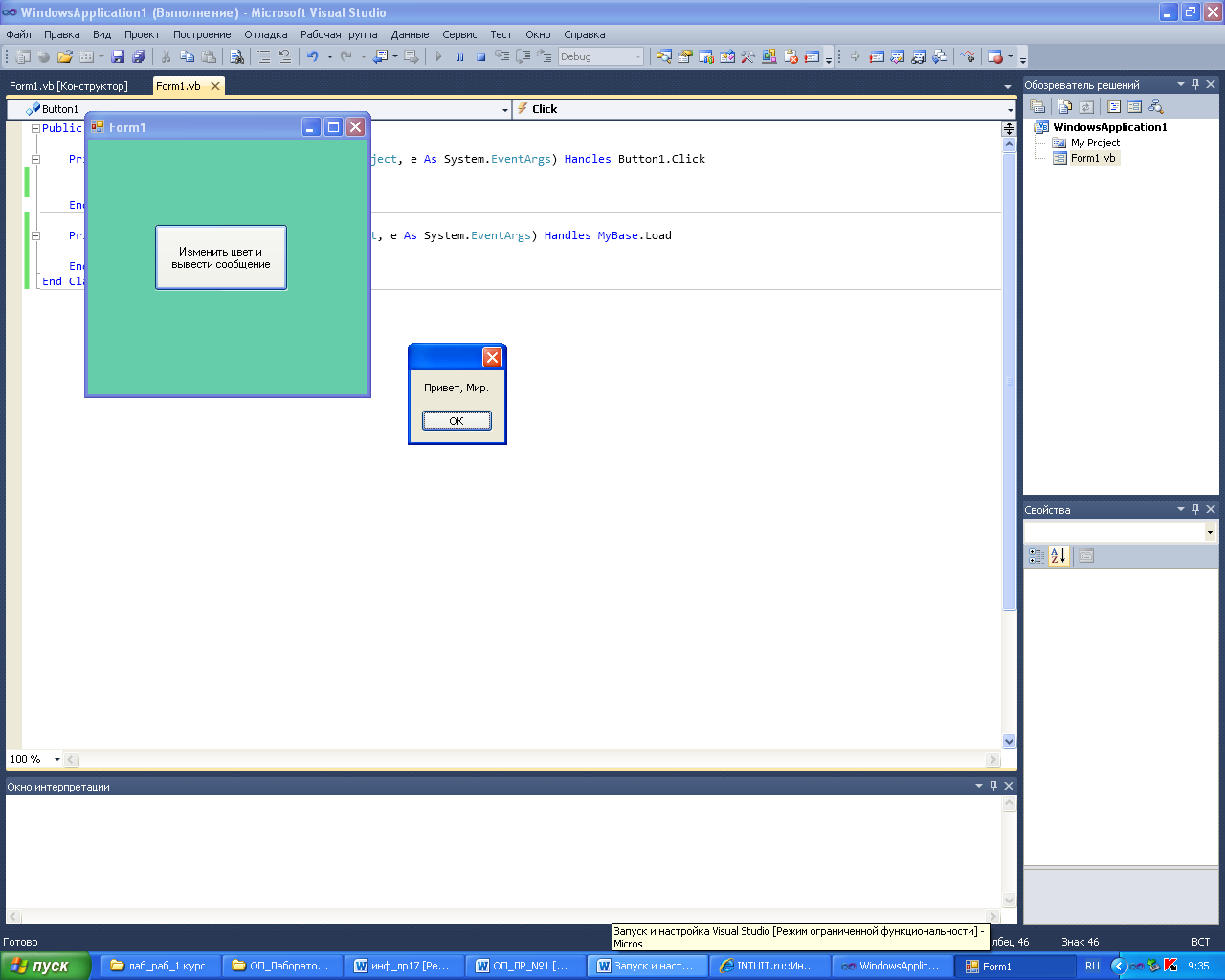
Если построение выполнить не удалось, верни­тесь назад и введите строку кода точно в том виде, в каком она приведена в этом курсе. Затем еще раз по­пытайтесь построить решение, как показано ранее.

**Запуск проекта**

Теперь скомпилированную программу можно запустить.

1. Откройте меню *Отладка (Debug).*

2. Выберите в этом меню пункт *Начать (Start).*(На экране появится форма проекта с кнопкой, которую вы на нее поместили.)



3. Щелкните на кнопке, изменится цвет формы и появится окно сообщения с текстом «Привет, Мир».

4. Нажмите кнопку *OK,* чтобы закрыть окно сообщения. (Окно сообщения закроется, но основное окно проекта останется открытым.Можете еще раз нажать кнопку, и окно сообщения появится снова.)

5. Нажмите кнопку со значком в верхнем правом углу формы, чтобы закрыть программу.



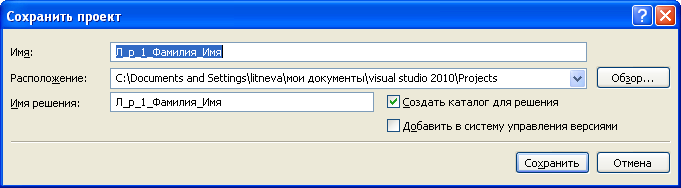
После завершения выполнения программы в окне *Вывод (Debug)* отображается отладочная информа­ция. Чтобы посмотреть выведенные сообщения пол­ностью, вам, вероятно, придется воспользоваться линейкой прокрутки.

**Сохранение проекта**

Сохранение выполняется так же, как и в других программах в операционной системе Windows.

1. Откройте меню *Файл (File).*
2. В меню *Файл (File)* выберите команду *Сохранить все (Save АН).*

Вы можете принять все значения по умолчанию для сохраняемого проекта, нажав кнопку *Сохранить(Save)*, (не забудьте только каталог в котором вы сохраняете проект, его имя указывается в поле *Расположение (Location:)*). Имеется также возможность определить свои значения. Для этого в поле *Имя (Name:)* введите имя сохраняемого проекта, в поле *Расположение (Location:)* задайте каталог, в котором должен быть сохранен проект или выберите этот каталог с помощью кнопки *Обзор (Browse...)*, а также задайте в поле *Имя решения (Solution Name:)* имя решения, содержащего ваш проект. Если вы хотите сохранить ваше решение в отдельном каталоге, оставьте галочку в поле *Создать каталог для решения (Create directory for solution)*, если же этого не требуется, то снимите галочку и нажмите кнопку *Сохранить (Save)* для сохранения.



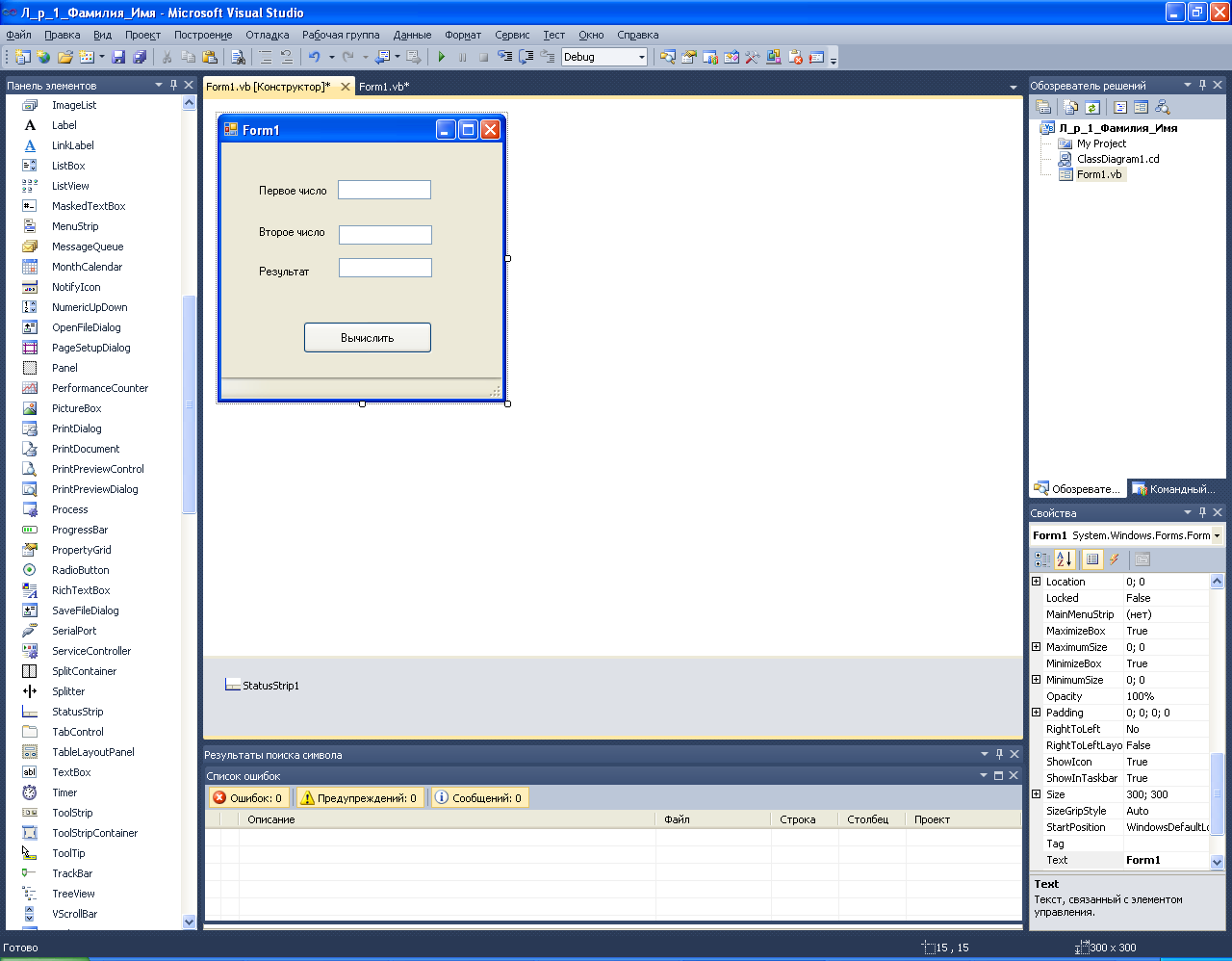
Файлы и **папки** решения. Когда вы создаете при­ложение на Visual Basic, создается целый набор файлов. Вы должны знать, какие это файлы и для чего они предназначены. Перейдите в папку , в которой вы создали проект «Привет, Мир». Откройте эту папку, и вы увидите, что для проекта была создана отдельная папка, названная Привет\_Мир. Зайдите в папку Привет\_Мир, чтобы посмотреть файлы, со­зданные для вашего проекта VisualStudio .NET. В папке будут следующие файлы:

Привет\_Мир.sln — это файл решения, в котором хранится описание всех файлов и настроек этого решения. Вообще-то в решение можно включить несколько проектов, но в решении «Привет, Мир» проект только один — это файл ПриветМир.vbproj. К файлу Привет\_Мир.sln система обращается, когда вы открываете решение. Файл Form1.vb содержит форму и связанный с ней код.

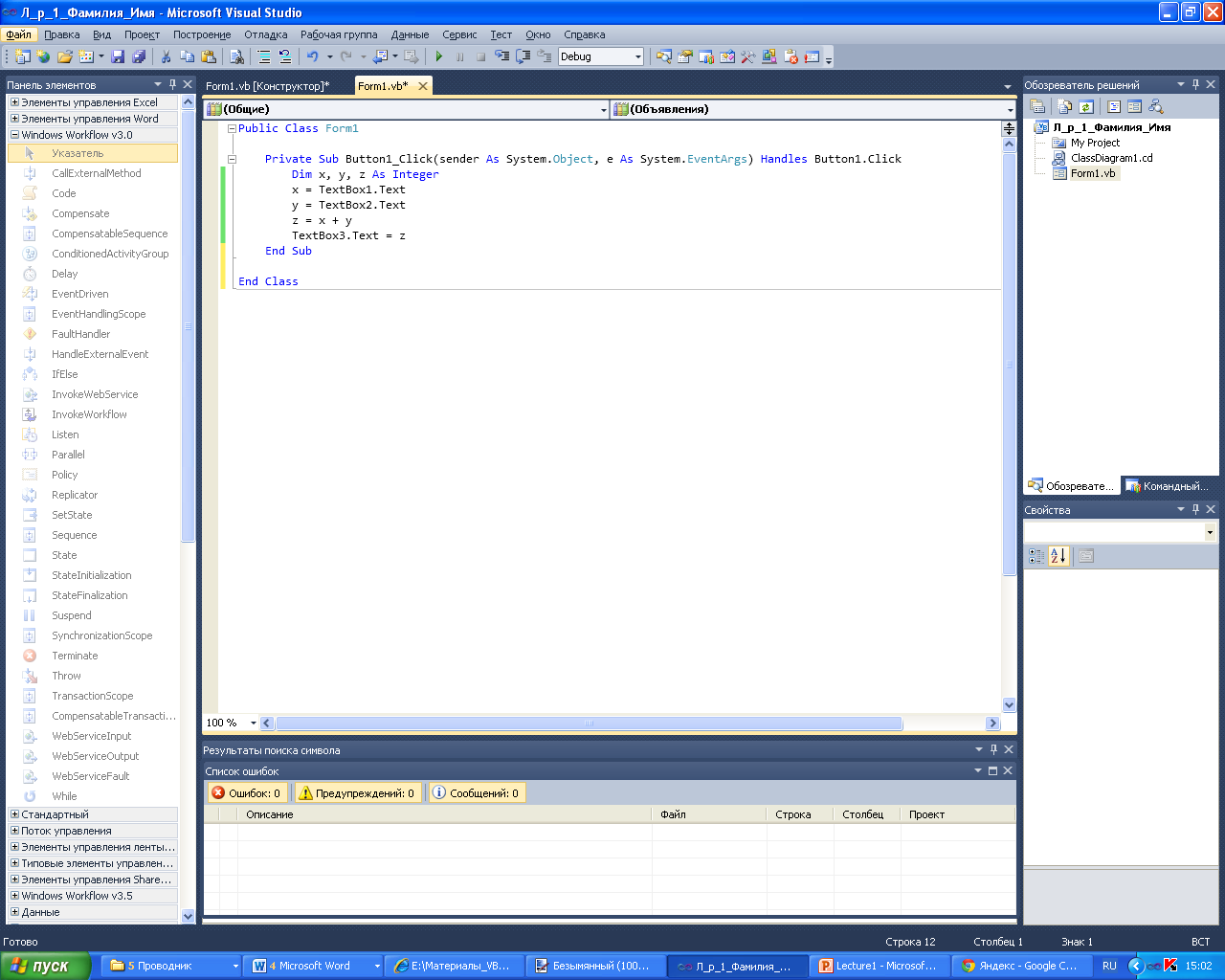
Откройте папку ..\bin\, сделав по ее значку двойной щелчок мышью. В папке ..\bin\ содержится ис­полняемый файл, полученный компиляцией программы на VisualBasic. Это файл Привет\_Мир.ехе. Его можно запустить на другом компьютере, даже если на нем не установлена система программирова­ния VisualStudio .NET.

**Задание 2.**Разработать программу, которая обеспечит возможности получения данных от пользователя, обработки полученных данных и выдачи результата. Для этого в первую очередь создайте графический интерфейс программы, примерно так, как показано на рисунке.

1. Выберите в левой области Начальной страницы пункт *Создать проект(Create:).*В появившемся окневыберите тип *Приложение Windows Forms(WindowsForms Application).*В поле ввода *Имя: (Name:)* введите имя своего проекта. Нажмите OK



1. Для статических строк текста, например "Первое число", поместите на форму три элемента *Label* с панели элементов. Для отображения текста измените значение свойства "Text" соответствующих элементов *Label1,Label2, Label3* на *Первое число, Второе число, Результат*.
2. Для создания полей ввода/вывода добавьте на форму три элемента *TextBox* с панели элементов.
3. Для написания программы нужно, как и в предыдущем примере, сделать двойной щелчок на кнопке*Вычислить* и написать в появившемся обработчике следующий код



**Пояснения к приведенному коду.**

В первой строке с помощью оператора Dim определены три целочисленных переменных x, y и z.

Вторая строка обеспечивает считывание числа, записанного пользователем в первое окошко ввода, в переменную x. Заметьте, что все окошки ввода различаются именами, заданными по умолчанию: *TextBox1*, *TextBox2*, *TextBox3*. Программист может изменить эти имена, меняя значения свойства "Name" для соответствующего элемента. Разумеется, после этого в программном коде ссылаться на эти элементы вы должны используя данные вами имена. Свойство "Text" поля ввода содержит текст, вводимый пользователем, и отображаемый на экране.

В третьей строке содержимое второго поля ввода копируется в переменную y.

Следующая строка реализует непосредственно вычисления, в данном случае сложение значений x и y и присвоение полученного результата переменной z.

Чтобы полученный результат стал известен пользователю, в последней строке кода производится присвоение свойству "Text" третьего окошка ввода значения переменной z, что и приводит к отображению ее значения в этом окошке.

1. Проверьте работу программы, нажав клавишу F5, введя числовые данные и нажав кнопку "Вычислить". Поэкспериментируйте с программой. Попробуйте вместо суммы вычислять значения каких-либо других арифметических выражений, например, умножения.

**Контрольные вопросы:**

1. Перечислите и охарактеризуйте основные элементы, которые появляются при запуске интегрированной среды разработки Visual Studio.NET.
2. Перечислите и опишите основные пункты меню и команды интегрированной среды Visual Studio 2010
3. Основные приемы настройки интегрированной среды Visual Studio 2010.
4. С помощью какого окна выбираются элементы управления для размещения их на форме.
5. Способы переключения в редактор кода.
6. Назначение оператора Dim.
7. Выполнение построения решения.
8. Выполнение отладки приложения.
9. Назвать основные файлы проекта.

Лабораторная работа №2

**Создание экземпляров класса. Объявление класса**

**Цель:**научиться объявлять классы, создавать экземпляры классов в среде VisualStudio.

## Ход работы:

## Изучить теоретическую часть.

## Выполнить задание, следуя указаниям.

## Ответить на контрольные вопросы (в устной форме).

## Предъявить преподавателю результаты работы программы и исходные коды.

## Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, программный код, результаты тестирования программы, выводы).

**🕮Теоретическая часть**

Процесс создания объектов (экземпляров класса) в VisualBasic.NET можно разделить на три этапа:

1. Создание файла классов.

2. Написание кодов в файле классов.

3. Объявление в программе переменных, которые представляют собой файлыклассов.

Если переменная представляет файл классов, она называется объектом —термином, взятым из концепции объектно-ориентированного программирования.

Файл классов является, по сути, шаблоном, который определяет, как должны вести себя объекты. После создания файла классов вам еще нужно объявить переменную — чтобы создать таким образом сам объект.

Определение объекта

После того как файл классов будет создан, приступайте к написанию кодов, определяющих, что из себя представляет новый объект. Обычно такие коды (они также называются модулем класса) состоят из трех частей:

1. объявление переменных;
2. объявление свойств (данных);
3. методы, являющиеся процедурами BASIC, которые манипулируют переменными и свойствами.

PublicClassClassl

Объявление переменных

Объявление свойств

Методы

EndClass

Каждый объект инкапсулирует (отделяет) свои данные (свойства) от остальной части программы. Эти данные могут быть изменены лишь командами (методами), сохраненными в томже файле классов. Таким образом, другие команды программы никогда непосредственно необращаются к данным, принадлежащим какому-то объекту.

Если возникает необходимость изменить способ обработки каких-либо данных, просматривать всю программу уже не нужно — достаточно изменить коды только одного объекта.

Таким образом, объектно-ориентированное программирование помогает изолировать команды, которые имеют доступ к определенным данным, что значительно снижает вероятностьвозникновения ошибок при внесении изменений в коды программы.

Объявление переменных

Объявление переменных в начале модуля класса является хорошей практикой (к тому жеобязательной), поскольку позволяет в любой момент видеть, какие данные этим классом используются. Если нужно объявить переменную, доступ к которой могут получить командытолько этого класса, объявите ее как локальную переменную:

Private Число AsInteger

Чтобы объявитьглобальную переменную, замените слово Private ключевым словом Public:

Public Число AsInteger

Объявление свойств объектов

Свойства объектов представляют те данные, доступ к которым имеют все коды вашейпрограммы. Любая часть программы может передать свойствам объектов новые значения иизвлечь хранимые ими данные. Коды объявления свойств выглядят приблизительно так:

Dim. ИмяПеременнойAsТипДанных

PublicPropertyНазваниеСвойства() AsТипДанных

Get

НазваниеСвойства = ИмяПеременной

EndGet

Set (ByVal Значение AsТипДанных)

ИмяПеременной = Value

End Set

End Property

Чтобы объявить свойство объекта, нужно создать локальную переменную (ИмяПеременной),в которой будет храниться значение свойства. Вместо имени ИмяПеременной можно присвоитьлюбое другое имя.Тип этой переменной (ТипДанных) должен совпадать с типом данных, объявленным длясвойства. Например, если объявлено, что свойство будет содержать значения типа String,переменная также должна иметь тип String.И, наконец, необходимо указать имя, посредством которого другие коды программы смогут иметь доступ к значениям свойств объектов (НазваниеСвойства).Например, если нужно объявить свойство объекта, именуемое словом Вектор и содержащее значения типа Integer, это можно сделать так:

Dim ЗначВекAs Integer

PublicProperty Вектор () As Integer

Get

Вектор = ЗначВек

End Get

Set

ЗначВек = Value

End Set

End Property

Вот как эти коды будут восприняты VisualBasic .NET.

В первой строке объявляется о создании переменной ЗначВек, которая может содержать значения типа Integer.

2. Вторая строка говорит: "Создай свойство Вектор, которое будет представлять данные типа Integer".

3. Третья, четвертая и пятая строки дают указание: "Когда другая часть программы хочет извлечь (Get) данные, представляемые свойством Вектор, передай ей значение переменной ЗначВек".

4. Шестая, седьмая и восьмая строки говорят: "Когда другая часть программыхочет передать (Set) новое значение свойству Вектор, сохрани его как значение локальной переменной ЗначВек".

5. А девятая строка сообщает: "На этом объявление свойства завершается"'.

Создание объектов

Модуль класса— это еще не объект. Он является лишь средством, с помощью которого объекты создаются. Согласно терминологии объектно-ориентированного программирования, создание объектов называется также созданием экземпляров классов.

Чтобы создать экземпляр класса, нужно создать объект, который будет представлять модуль класса. Для этого используется ключевое слово New:

DimИмяОбъектаAsNewНазваниеКласса

Вот что эта строка означает для VisualBasic .NET.

1. Слово Dim говорит: "Сейчас будет объявлено о создании объекта".

2. Словом ИмяОбъекта обозначается имя объекта.

3. Слово New дает указание: "Создай новый объект, который будет представлять модуль класса, именуемый НазваниеКласса".

НазваниеКласса — это то имя, которое вы дали файлу класса в момент его создания. Если вы сами не укажете это имя, VisualBasic .NET присвоит ему автоматически генерируемое имя наподобие C l a s s l .

Использование объектов

После того как объект создан, остается только использовать его для:

* сохранения значений в свойствах объектов;
* извлечения данных через свойства объектов;
* манипулирования данными объектов с помощью методов объектов.

Чтобы передать объекту данные, необходимо написать такой код:

ИмяОбъекта.Свойство = Значение

Извлечь информацию, хранящуюся в объекте, вам поможет код:

ИмяПеременной = ИмяОбъекта.Свойство

А чтобы запустить метод объекта, наберите следующее:

ИмяОбъекта.Метод

#### Задание. Следуя указаниям, создайте программу, которая запрашивает у нового сотрудника имя, фамилию и дату рождения. Вы будете хранить эту информацию в свойствах нового класса с именем Person, и создадите метод класса, который будет вычислять текущий возраст нового сотрудника. Этот проект научит вас создавать собственные классы, экземпляры классов (объекты) а также как использовать эти классы в процедурах событий вашей программы.

#### Добавление в ваш проект нового класса

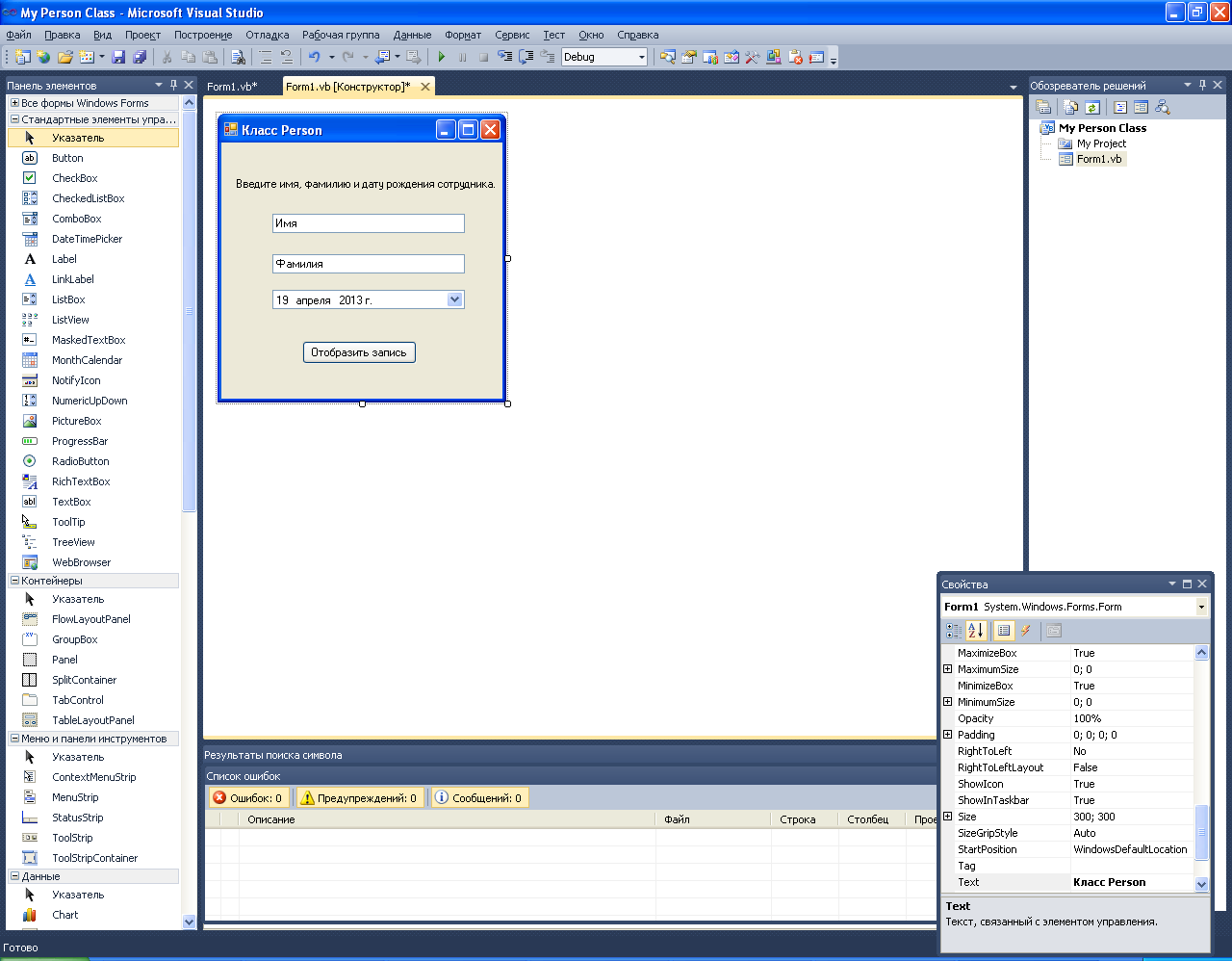
Класс, определенный пользователем, позволяет определить в программе ваши собственные объекты, которые имеют свойства, методы и события, точно так же, как объекты, создаваемые на формах Windows с помощью элементов управления из Области элементов. Чтобы добавить в ваш проект новый класс, щелкните в меню Проект (Project) на команде Добавить класс (AddClass), а затем определите этот класс с помощью кода программы и нескольких новых ключевых слов VisualBasic.

##### Создание проекта PersonClass

1. Запустите *MicrosoftVisualStudio2010,* затем создайте в своей папке новый проект с именем **MyPersonClass**.
2. Используйте элемент управления *Label* и добавьте в верхней части формы Form1 длинную метку.
3. Используйте элемент управления *TextBox* и нарисуйте под меткой два широких текстового поля.
4. Используйте элемент управления *DateTimePicker* и нарисуйте под текстовыми полями объект выбора даты и времени.
5. Используйте элемент управления *Button* и нарисуйте под объектом выбора даты и времени кнопку.
6. Установите для объектов формы следующие свойства:

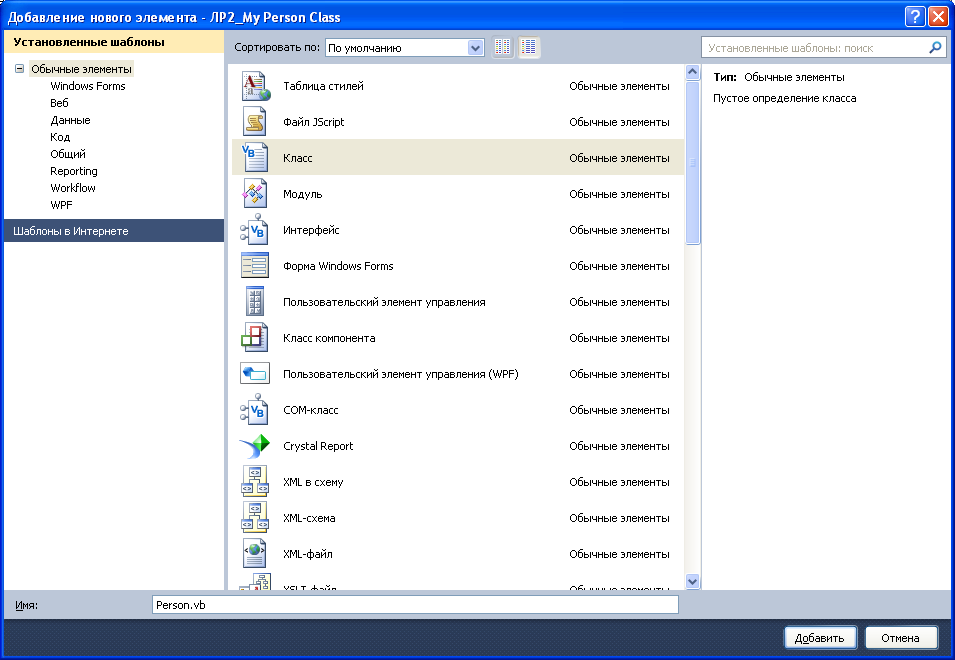
|  |  |  |
| --- | --- | --- |
| **Объект** | **Свойство** | **Установка** |
| Label1 | Text | Введите имя, фамилию и дату рождения сотрудника. |
| TextBox1 | Text | Имя |
| TextBox2 | Text | Фамилия |
| Button1 | Text | Отобразить запись |
| Form1 | Text | Класс Person |

1. Ваша форма должна выглядеть примерно так.

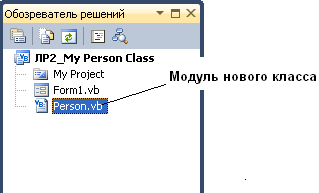


Это базовый интерфейс пользователя для формы, которая определяет запись нового сотрудника фирмы. (Эта форма не подключена к базе данных, так что храниться может только одна запись.) Теперь вы должны добавить в проект класс для хранения информации из этой записи.

1. Щелкните на команде*Добавить класс(AddClass)* в меню *Проект (Project)*. VisualStudio откроет диалоговое окно *Добавление нового элемента (AddNewItem)*, показанное ниже.



Диалоговое окно Добавление нового элемента дает возможность задать имя вашего класса. Когда вы присвоите имя, обратите внимание, что вы можете сохранить в новом модуле класса несколько классов и указать имя, которое будет для них общим.



1. Введите в текстовом поле *Имя(Name)* имя *Person.vb*, а затем щелкните*Добавить*. *VisualStudio* откроет в Редакторе кода пустой модуль класса и добавит имя файла *Person.vb* в ваш проект в *Обозревателе решений*, как показано на рисунке.

##### Объявление переменных класса

* Под оператором программы PublicClassPerson введите следующие объявления переменных:



Здесь вы объявляете две переменные, которые будут использованы исключительно в модуле класса для хранения значений двух строковых свойств. Переменные объявлены с помощью ключевого слова Private, так как по соглашению VisualBasic программисты должны держать внутренние переменные класса закрытыми - другими словами, недоступными для просмотра извне самого модуля класса.

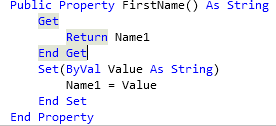
##### Создание свойств

1. Под объявлением переменных введите следующий оператор программы и нажмите клавишу (Enter):



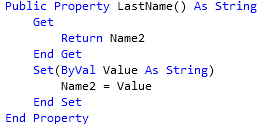
Этот оператор создает свойство вашего класса с именем FirstName, которое имеет тип String. Когда вы нажмете (Enter), VisualStudio немедленно создаст структуру кода для остальных элементов объявления свойства. Требуемыми элементами являются: блок Get, который определяет, что программисты увидят, когда будут проверять свойство FirstName, блок Set, который определяет, что произойдет, когда свойство FirstName будет установлено или изменено, и оператор EndProperty, который отмечает конец процедуры свойства.

1. Заполните структуру процедуры свойства так, чтобы она выглядела, как показано ниже.



Ключевое слово Return указывает, что при обращении к свойству FirstName будет возвращена строковая переменная Name1. При установке значения свойства блок Set присваивает переменной Name1 строковое значение. Обратите особое внимание на переменную Value, используемую в процедурах свойств для обозначения значения, которое присваивается свойству класса при его установке. Хотя этот синтаксис может выглядеть странно, просто поверьте мне - именно так создаются свойства в элементах управления, хотя более сложные свойства будут иметь здесь дополнительную программную логику, которая будет проверять значения и производить вычисления.

1. Под оператором EndProperty введите для свойства LastName вашего класса вторую процедуру свойства. Она должна выглядеть так, как показано ниже.



Эта процедура свойства аналогична первой, за исключением того, что она использует вторую строковую переменную (Name2), которую вы объявили в верхней части кода класса. Вы закончили определять два свойства вашего класса. Теперь перейдем к методу с именем Age, который будет определять текущий возраст нового сотрудника на основе даты рождения.

##### Создание метода

* Под процедурой свойства LastName введите следующее определение функции:



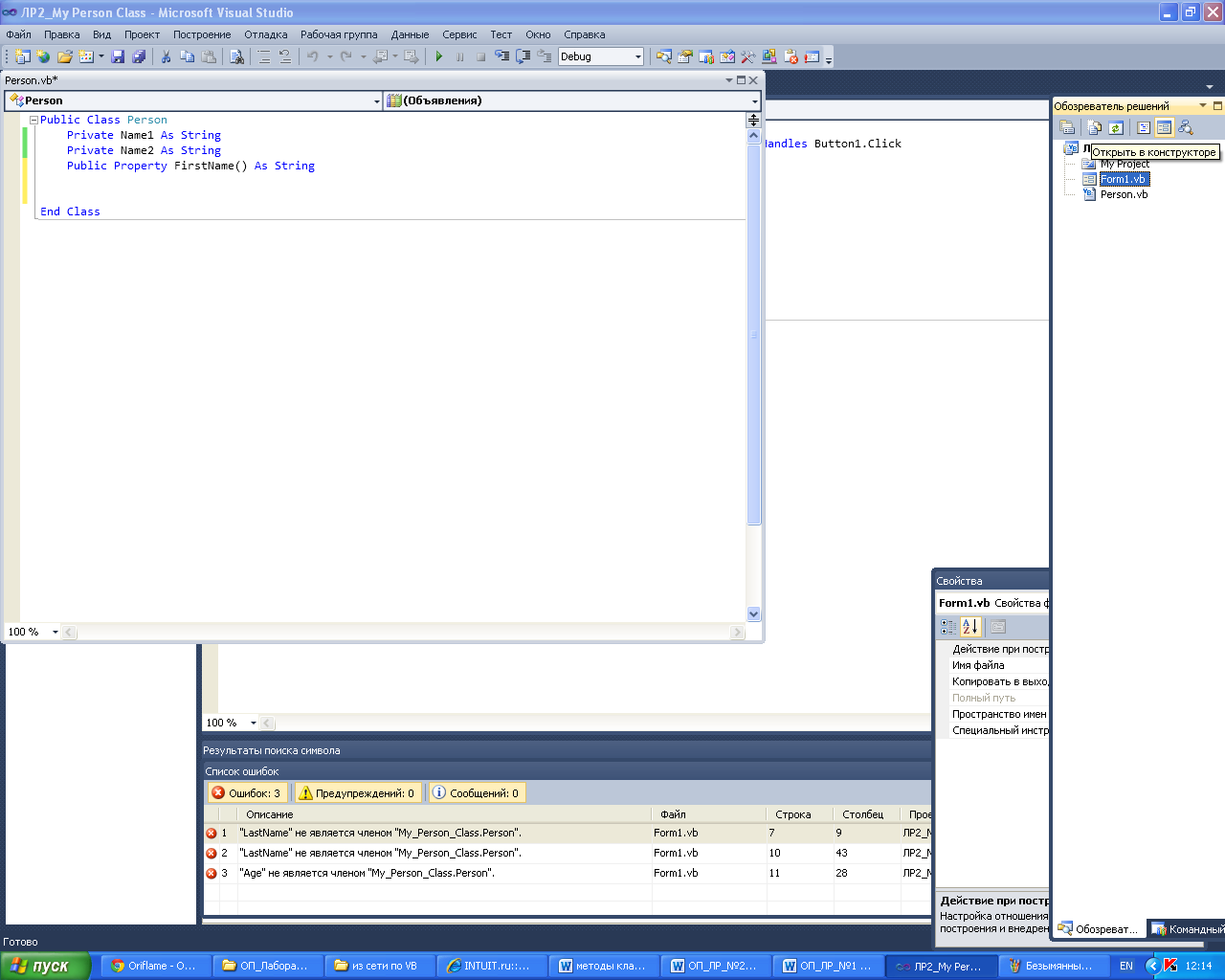
Чтобы создать метод класса, который выполняет некое действие, добавьте в ваш класс процедуру Sub. Хотя многие методы не требуют для выполнения своей работы аргументов, метод Age, определенный мной, требует для своих вычислений аргумент Birthday типа Date. Это метод использует для вычитания даты рождения нового сотрудника из текущей системной даты метод Subtract, и возвращает значение, выраженное в днях, деленных на 365.25 - примерную длину одного года в днях. Функция Int преобразует это значение в целое, и это число с помощью оператора Return возвращается в вызывающую процедуру - как и в случае с обычной функцией.

Определение класса закончено! Вернитесь к форме *Form1* и используйте новый класс в процедуре события.

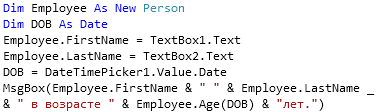
**Совет**. Хотя в данном примере это и не делалось, в реальном проекте полезно добавить в модуль класса логику для проверки типов данных. Это делается для того, чтобы неправильное использование свойств или методов, не приводило к возникновению ошибок времени исполнения, из-за которых выполнение программы может прерваться.

##### Создание объекта с помощью нового класса

1. Щелкните в Обозревателе решений на значке Form1.vb, а затем на кнопкеОткрыть в конструкторе. Появится интерфейс пользователя Form1.



1. Чтобы открыть в Редакторе кода процедуру события Button1\_Click, сделайте двойной щелчок мышью на кнопке**Отобразить запись**.
2. Введите следующие операторы программы:

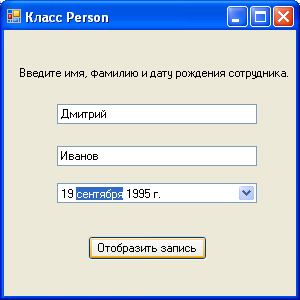


Эта процедура сохраняет в объекте с именем Employee, который имеет тип Person, значения, введенные пользователем. Ключевое слово New указывает, что вы хотите немедленно создать новый экземпляр объекта Employee. Теперь нужно объявить переменную с помощью класса, созданного вами самими! Затем процедура объявляет переменную с именем DOB типа Date. Она будет хранить дату, введенную пользователем, и устанавливает свойства FirstName и LastName объекта Employeeравными имени и фамилии, введенным в два объекта текстовых полей формы. Значение, возвращаемое объектом выбора даты и времени, сохраняется в переменной DOB, а последний оператор программы отображает окно сообщения, содержащее свойства FirstName и LastName, а также возраст нового сотрудника, определенный методом Age, который при передаче в него переменной DOB возвращает целое значение. Как только вы определили класс в модуле класса, его легко можно использовать в процедуре события.

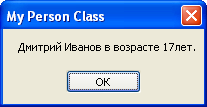
1. Чтобы запустить программу, щелкните на кнопке Начать отладку (F5).В среде разработки появится интерфейс пользователя, готовый к приему ваших данных.
2. Введите в текстовое поле FirstName ваше имя, а в текстовое поле LastName - фамилию.
3. Щелкните на раскрывающемся списке объекта выбора даты и времени, и прокрутите его до вашей даты рождения.

**Совет**. Вы можете быстро прокрутить список, щелкнув в открытом диалоговом окне объекта выбора даты на поле года. Появятся небольшие стрелки прокрутки, и вы сможете переходить сразу на год вперед или назад. Также можно быстро перейти на нужный вам месяц, щелкнув на поле месяца, а затем на месяце в появившемся меню.

Ваша форма будет выглядеть примерно так.



1. Щелкните на кнопке**Отобразить запись**. Ваша программа сохраняет значения имени и фамилии в свойствах и использует метод Age для вычисления текущего возраста нового сотрудника. Появится диалоговое окно с результатом.



1. Чтобы закрыть это окно сообщения, щелкните на **OK**, а затем поэкспериментируйте с несколькими различными значениями дат, щелкая на**Отобразить запись** каждый раз, когда вы меняете значение поля даты рождения.

**Контрольные вопросы:**

1. Определите понятия класс, экземпляр класса, объект.
2. На какие этапы можно разбить процесс создания экземпляров класса.
3. Как создать базовый файл класса, какой код он содержит.
4. Синтаксис объявления переменной класса.
5. Создание в классе нового свойства.
6. Создание в классе нового метода.
7. Объявление переменной объекта для использования в классе.
8. Обращение к свойствам и методам объекта.

Лабораторная работа №3

**Создание наследованного класса**

**Цель:**научиться создавать наследованные классы, реализовать наследование форм в среде VisualStudio.

## Ход работы:

## Изучить теоретическую часть.

## Выполнить задания, следуя указаниям.

## Ответить на контрольные вопросы (в устной форме).

## Предъявить преподавателю результаты работы программы и исходные коды.

## Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, программный код, результаты тестирования программы, выводы).

**🕮Теоретическая часть**

Оператор **Inherits** используется для объявления нового класса, называемого *производным классом*, который основан на существующем классе, называемом *базовым классом*.Производные классы наследуют и могут расширять свойства, методы и события, поля и константы, определенные в базовом классе.

* Все классы могут наследоваться по умолчанию, если только они не помечены зарезервированным словом **NotInheritable**.Классы могут наследовать из других классов проекта или из классов других сборок, на которые ссылается проект.
* В отличие от языков, которые позволяют множественное наследование, VisualBasic позволяет только единичное наследование в классах; то есть производные классы могут иметь только один базовый класс.Хотя в классах не поддерживается множественное наследование, они все же могут реализовывать множественные интерфейсы, которые способны эффективно выполнять те же самые задачи.
* Чтобы предотвратить предоставление элементов с ограничениями в базовом классе, тип доступа к производному классу должен быть таким же, как и тип доступа к базовому классу, или более строгим.Например, **Public** класс не может наследовать **Friend** или **Private** класс, а **Friend** класс не может наследовать **Private** класс.

[**Модификаторы наследования**](javascript:void(0))

Для поддержки наследования VisualBasic вводит следующие инструкции на уровне класса и модификаторы:

* Инструкция **Inherits** определяет базовый класс.
* Модификатор **NotInheritable**не позволяет использовать класс в качестве базового класса.
* Модификатор **MustInherit** определяет, что класс предназначен только для использования в качестве базового класса.Невозможно создать напрямую экземпляры классов **MustInherit**; их можно создать только как экземпляры базового класса в производном классе

### Наследование форм с помощью инструмента Выбор наследования

В терминологии объектно-ориентированного программирования наследование означает, что один класс получает объекты, свойства, методы и другие атрибуты другого класса. VisualBasic всегда использует это при создании в среде разработки новой формы. Первая форма проекта (*Form1* ) определена на основе класса System.Windows.Forms.Form и получает от него свои значения по умолчанию. На самом деле каждый раз, когда с помощью команды*Добавить форму Windows(AddWindowsForm)* из меню *Проект (Project)* создается новая форма, этот класс указывается в верхней части кода каждой формы с использованием ключевого слова Inherits, как показано ниже:

**Inherits**

**System.Windows.Forms.Form**

Вы неосознанно всегда использовали наследование для определения форм Windows, которые использовали при создании приложений VisualBasic. Хотя существующие формы могут наследоваться с помощью кода программы, разработчики VisualStudio .NET сочли эту задачу важной и разработали специальный инструмент среды разработки, облегчающий этот процесс. Этот инструмент называется Выбор наследования (InheritancePicker). Он доступен через команду Добавить производную форму (AddInheritedForm) в меню *Проект (Project)*. В следующем упражнении вы будете использовать Выбор наследования (InheritancePicker)для создания второй копии диалогового окна проекта.

#### Задание 1. Наследование простого диалогового окна

1. Запустите VisualStudio и создайте новый проект с именем **MyFormInheritance**.
2. Отобразите форму проекта и используйте элемент управления *Button* для добавления в нижнюю часть формы двух расположенных рядом объектов кнопок.
3. Измените свойства *Text* кнопок *Button1* и *Button2* на *"OK"* и *"Отмена"* соответственно.
4. Чтобы отобразить в Редакторе кода процедуру события Button1\_Click, сделайте двойной щелчок мышью на кнопке **OK**.
5. Введите следующий оператор программы:



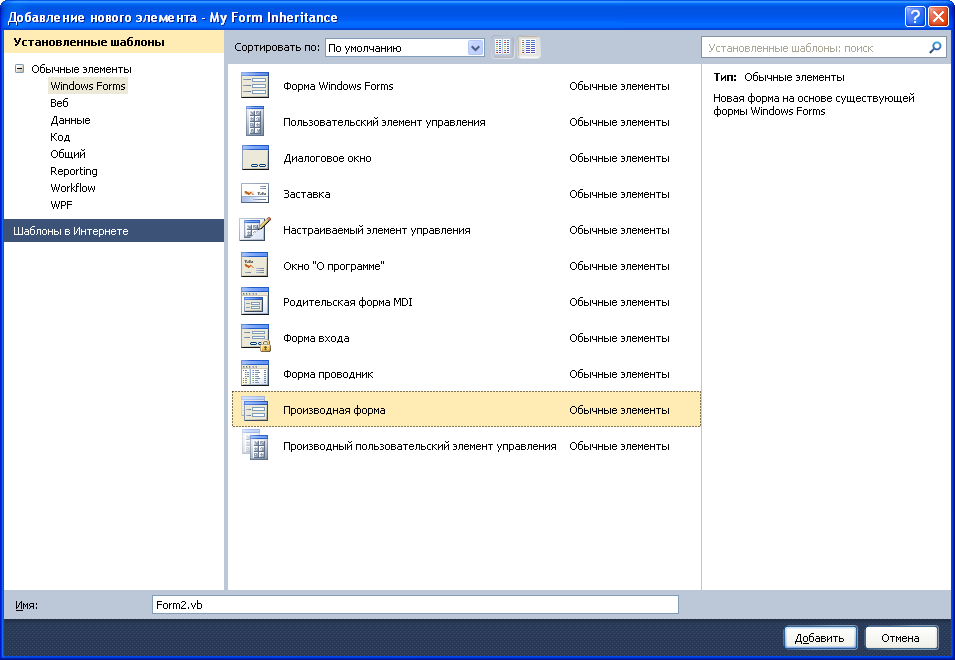
1. Снова отобразите форму, сделайте двойной щелчок мышью на кнопке **Отмена**, а затем введите в процедуре события Button2\_Click следующий оператор программы:



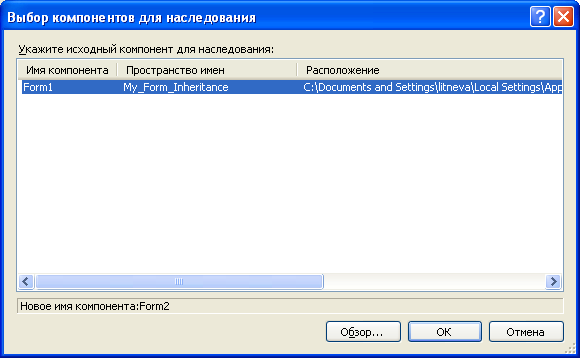
1. Снова отобразите форму, а затем установите свойство *Text* формы на значение *"Диалоговое окно"*. Теперь у вас есть простая форма, которую можно использовать как основу для диалогового окна программы. С помощью некоторых настроек вы можете использовать эту базовую форму для выполнения нескольких задач - просто нужно добавить на нее элементы управления, которые потребуются вашему приложению.

Попрактикуйтесь в наследовании форм. Первым шагом в этом процессе является сборка - или **компиляция** - проекта, так как наследовать можно только от тех форм, которые скомпилированы в виде файлов .exe или .dll. Каждый раз, когда компилируется базовая форма, изменения, сделанные в этой базовой форме, передаются в производную (наследованную) форму.

1. Щелкните на команде *Построить решение (BuildSolution)* в меню *Построение (Build)*. Visual Basic скомпилирует ваш проект и создаст .exe-файл.
2. Щелкните на команде *Добавить производную форму (Add Inherited Form)* в меню *Проект (Project)*. Вы увидите диалоговое окно, показанное ниже. Как обычно, Visual Studio приводит список всех возможных шаблонов, которые вы можете включить в проект. На панели Установленные шаблоны выберите Windows Forms, а справа-Производная форма (Inherited Form). Текстовое поле *Имя(Name)* в нижней части диалогового окна позволяет присвоить вашей производной форме имя; это имя, которое появится в *Обозревателе решений (Solution Explorer)* и в имени файла формы на диске.

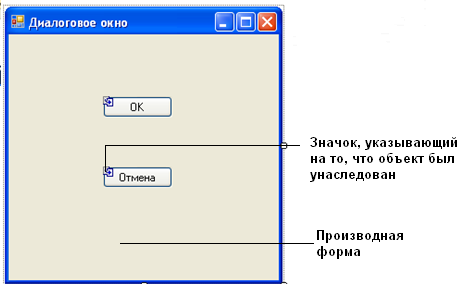


1. Щелкните на *Добавить (Add)*, чтобы принять для новой формы установки по умолчанию. Visual Studio отобразит диалоговое окно Выбор компонентов для наследования (Inheritance Picker), показанное ниже.



Это диалоговое окно содержит перечень всех форм текущего проекта, от которых можно наследовать. Если вы хотите просмотреть другие скомпилированные формы, щелкните на кнопке *Обзор (Browse)* и найдите на вашем жестком диске требуемый .dll-файл. (Если вы хотите наследовать от формы, которая не является компонентой текущего проекта, форма должна быть скомпилирована в .dll-файл.)

1. Щелкните в диалоговом окне Выбор наследования (Inheritance Picker) на *Form1*, а затем на **OK**. Visual Studio создаст в Обозревателе решений (Solution Explorer) элемент Form2.vb и отобразит в Конструкторе Windows Forms (Windows Forms Designer) производную форму. На следующем рисунке обратите внимание, что форма выглядит идентично окну Form1, созданному ранее, за исключением того, что две кнопки содержат маленькие значки, которые указывают, что объекты получены из наследуемого источника.



Иногда сложно отличить производную форму от базовой (маленькие значки наследования не так очевидны), так что используйте для различения этих форм Обозреватель решений (Solution Explorer) и закладки окон среды разработки.

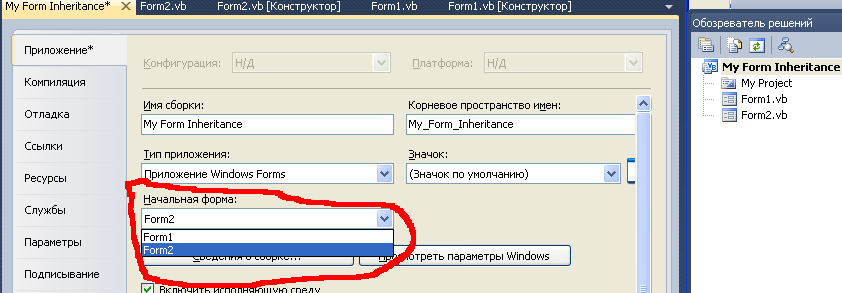
Теперь вы должны добавить к производной форме несколько новых элементов.

#### Настройка производной формы

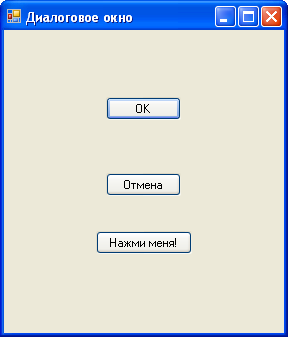
1. Используйте элемент управления *Button* и добавьте на Form2 (производная форма) третью кнопку.
2. Установите свойство *Text* этой кнопки равным "*Нажми меня!"*.
3. Сделайте двойной щелчок мышью на кнопке **Нажми меня**!
4. В процедуре события Button3\_Click введите следующий оператор программы:



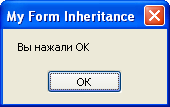
1. Снова отобразите форму Form2, а затем попробуйте сделать двойной щелчок мышью на кнопках **OK** и **Отмена**. Вы не можете отобразить или отредактировать процедуры событий для этих унаследованных объектов без дополнительных действий, которые не обсуждаются здесь. Однако вы можете добавить в форму новые объекты и настроить их.
2. Увеличьте форму. Вы также можете изменить другие характеристики формы, такие, как ее размер и расположение.
3. Щелкните в Обозревателе решений (Solution Explorer) на значке проекта My Form Inheritance, а затем на команде *Свойства (Properties)* в меню *Проект(Project)*. Появится диалоговое окно.
4. Щелкните на раскрывающемся списке Начальная форма, далее на Form2, а затем на **OK**. Запустите новый проект.



1. Щелкните на кнопке Начать(Start). Откроется производная форма, показанная ниже.



1. Щелкните на кнопке **OK**. Производная форма выполнит процедуру события наследуемой формы Form1, и она выведет следующее сообщение.



1. Щелкните на **Oтмена**, а затем на **Нажми меня**!. Form2 отобразит наследуемое сообщение формы. Производная форма настроена так, чтобы включить новый объект, а также два наследуемых объекта кнопки. Вы сделали первые шаги в освоении механизма наследования с помощью диалогового окна InheritancePicker (Выбор наследования).
2. Чтобы закрыть окно сообщения, щелкните на **OK**, а затем на **Закрыть** формы, чтобы завершить выполнение программы. Программа остановится, и вернется среда разработки.

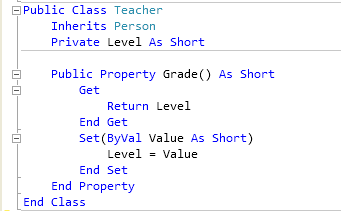
### Задание 2.Наследование базового класса

Точно так же, как при наследовании классов форм, вы можете наследовать обычные классы, которые вы сами определяете с помощью команды*Добавить класс (AddClass)* и модуля класса. Механизм наследования базового (родительского) класса состоит в использовании оператора **Inherits**, который включает ранее определенный класс в новый класс. Затем вы можете добавить в производный (дочерний) класс новые свойства или методы, которые будут отличать его от базового класса.

**Задание:**Изменить проект MyPersonClass (проект из Л\_Р\_№2), добавив в модуль класса Person второй класс, определенный пользователем. Этот новый класс с именем Teacher будет наследовать от класса Person свойство FirstName, свойство LastName и метод Age, и будет добавлять свойство с именем Grade, в которое будет записываться уровень, на котором обучает новый учитель.

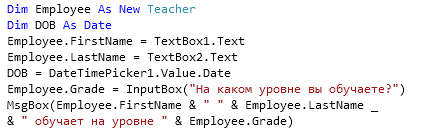
#### Использование ключевого слова Inherits

1. Щелкните в Обозревателе решений на классе Person.vb, а затем щелкните на кнопке*Просмотреть код (ViewCode).*
2. Прокрутите Редактор кода вниз так, что текстовый курсор окажется стоящим после оператора EndClass. Как упоминалось выше, вы можете включить в модуль класса более одного класса, при условии, что каждый класс отделен от остальных операторами PublicClass и EndClass. Вы создадите в этом модуле класса новый класс с именем Teacher, а для встраивания в него методов и свойств, определенных вами в классе Person, будете использовать ключевое слово Inherits.
3. Введите в Редактор кода следующее определение класса.



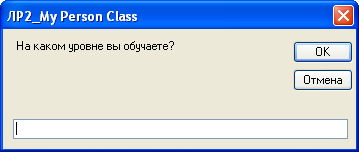
Оператор Inherits связывает класс Person с этим новым классом, встраивая в него все свои переменные, свойства и методы. Если бы класс Person находился в отдельном модуле или проекте, вы могли бы указать его расположение, задав пространство имен, точно так же, как вы указываете классы библиотеки .NET Framework с помощью оператора Imports. По существу класс Teacher определен как специальный тип класса Person - в дополнение к свойствам FirstName и LastName класс Teacher имеет свойство Grade, которое хранит уровень студента, обучаемого учителем. Теперь вы будете использовать этот новый класс в процедуре события Button1\_Click.

1. Отобразите процедуру события Button1\_Click формы Form1. Вместо создания новой переменной для хранения класса Teacher, просто используйте имеющуюся переменную Employee - единственное различие будет в том, что теперь вы можете установить свойство Grade нового сотрудника.
2. Измените процедуру события Button1\_Click в соответствии со следующим кодом.

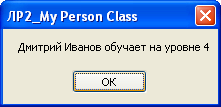


В этом примере мы удалили вычисление текущего возраста - метод Age не используется, но сделали это только для того, чтобы сократить до минимума информацию, выводимую в окне сообщения. Когда вы определяете свойства и методы класса, вам не требуется использовать их в коде программы.

1. Чтобы запустить программу, щелкните на кнопке Начать отладку. На экране появится форма для ввода данных о новом сотруднике.
2. Введите в текстовое поле **Имя** ваше имя, а в текстовое поле **Фамилия** - фамилию.
3. Щелкните на объекте выбора даты и прокрутите его до вашего дня рождения.
4. Щелкните на кнопке **Отобразить запись**. Ваша программа сохраняет значения имени и фамилии в свойствах, а затем отображает следующее окно ввода.



1. Введите 4, а затем щелкните на **OK**, чтобы закрыть окно ввода. Приложение сохраняет число 4 в новом свойстве Grade и использует свойства FirstName, LastName и Grade для отображения информации о новом сотруднике в подтверждающем окне сообщения. Вы увидите такое сообщение.



1. Если хотите, поэкспериментируйте еще с несколькими значениями, а затем щелкните на кнопке**Закрыть** формы. Программа остановится, и вернется среда разработки.

**Контрольные вопросы:**

1. Что означает наследование в терминологии ООП?
2. Какой оператор необходимо использовать для объявления производного класса?
3. На основе какого класса определена первая форма проекта?
4. Опишите действия по добавлению производной формы.
5. Назовите визуальные отличия производной формы от базовой.

Лабораторная работа №4

**Перегрузка методов**

**Цель:** научиться выполнять перегрузку методов в среде Visual Studio.

## Ход работы:

## Изучить теоретическую часть.

## Выполнить задания, следуя указаниям.

## Ответить на контрольные вопросы (в устной форме).

## Предъявить преподавателю результаты работы программы и исходные коды.

## Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, программный код, результаты тестирования программы, выводы).

**🕮Теоретическая часть**

[**Переопределение свойств и методов в производных классах**](javascript:void(0))

По умолчанию производный класс наследует свойства и методы от своего базового класса. Если наследуемое свойство или метод в производном классе должен работать другим образом, то оно может быть *переопределено*. То есть, в производном классе можно определить новую реализацию метода. Следующие модификаторы используются для управления переопределением свойств и методов.

* **Overridable** позволяет переопределение свойства или метода в производном классе.
* **Overrides** переопределяет **Overridable** свойство или метод, определенное в базовом классе.
* **NotOverridable** не позволяет переопределять свойство или метод в наследующем классе.По умолчанию **Public** методы являются **NotOverridable**.
* **MustOverride** требует обязательного переопределения свойства или метода производным классом.При использовании ключевого слова **MustOverride** определение метода состоит только из инструкций **Sub**, **Function** или **Property**.Другие инструкции не допускаются, и в частности отсутствуют инструкции **EndSub** и **EndFunction**.Методы**MustOverride** должны объявляться в классах **MustInherit**.

[Ключевое слово MyBase](javascript:void(0))

Ключевое слово **MyBase** ведет себя подобно объектной переменной, ссылающейся на базовый класс текущего экземпляра. Ключевое слово **MyBase** часто используется для обращения к членам базового класса, которые переопределены или скрыты в производном классе. В частности, можно воспользоваться вызовом MyBase.New для явного вызова конструктора базового класса из конструктора производного класса.

[Ключевое слово MyClass](javascript:void(0))

Ключевое слово **MyClass** ведет себя подобно объектной переменной, ссылающейся текущий экземпляр класса, который был изначально реализован. Ключевое слово **MyClass** напоминает слово **Me**, однако при каждом вызове метода или свойства **MyClass** предполагается, что метод или свойство имеет атрибут [NotOverridable (VisualBasic)](http://msdn.microsoft.com/ru-ru/library/ta771xh0(v=vs.100)). Следовательно, на метод или свойство не оказывает никакого влияния его переопределение в производном классе.

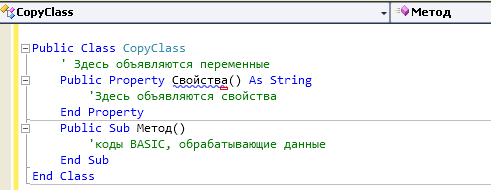
**Чтобы перегрузить какой-либо метод или свойство необходимо выполнить такие действия:**

1. В окне Обозреватель решений (SolutionExplorer) щелкните на названии файла классов, коды которого должны быть наследованы для создания новых классов.

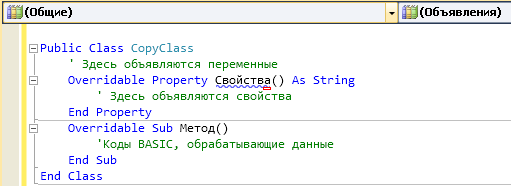
2. Нажмите клавишу <F7>, VisualBasic .NET отобразит на экране коды BASIC выбранного файла.

3. Для каждого свойства или метода, коды которого могут быть перегружены, замените слово **Public** ключевым словом **Overridable**.

Допустим, коды выбранного файла выглядят так:



Для свойств и методов замените слово Public словом Overridable:



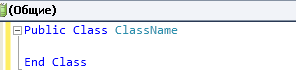
Коды тех свойств и методов, для которых вы оставите слово Public, в новых объектах не могут быть изменены.

4. Выберите команду *Проект/Добавить класс (Project/Add Class*).

Откроется диалоговое окно *Добавить новый элемент (Add New Item)*.

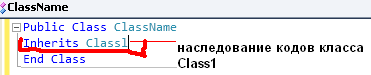
5. Наберите имя нового файла классов в поле Имя (Name) и щелкните на кнопке *Добавить (Add).*

Visual Basic .NET отобразит показанные ниже коды пустого класса, где вместо слова ClassName будет фигурировать указанное вами имя:



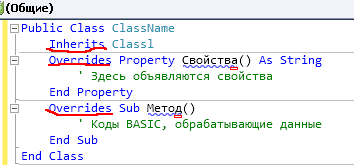
6. Наберите слово **Inherits** , а затем укажите название класса, коды которого должны быть наследованы.

Например, если нужно наследовать коды класса, названного именем Classl, наберите



7. Наберите коды свойств и методов, которые должны быть перегружены, но вместо слова Public используйте слово Overrides.

Слово Overrides используется при написании свойств и методов, коды которых будут отличаться от кодов свойств и методов исходного файла с теми же именами:



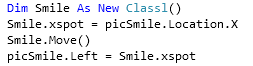
Таким образом, при создании нового класса вам нужно будет написать коды только тех свойств и методов, которые должны отличаться от свойств и методов исходного класса. Если приемы наследования и перегрузки кодов кажутся вам не очень понятными, не спешите сразу же отказываться от их применения. Помните, что они действительно могут помочь вам быстрее и более простым способом создавать корректно работающие программы, позволяя повторно использовать либо коды свойств и методов других объектов, либо только имена этих свойств и методов.

#### Задание. Чтобы реализовать перегрузку методов, прежде необходимо разработать приложение, в котором создаются файлы классов для представления местоположения улыбающегося и грустного лиц. Каждый раз, когда пользователь щелкает на кнопке Переместить улыбающееся лицо или Переместить грустное лицо, эти картинки начинают перемещаться. Воспользуйтесь графическим редактором для создания подобных рисунков.

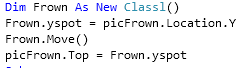
#### Ниже в таблице приведен список объектов пользовательского интерфейса, которые нужно создать для этой программы, и список значений, которые нужно присвоить их свойствам.

|  |  |  |
| --- | --- | --- |
| **Объект** | **Свойство** | **Значение** |
| PictureBox1 | Image  Name  SizeMode | указать URL рисунка  picSmile  StretchImage |
| PictureBox2 | Image  Name  SizeMode | указать URL рисунка  picFrown  StretchImage |
| Button1 | Text  Name | Переместить улыбающееся лицо  btnSmile |
| Button2 | Text  Name | Выход  btnExit |
| Button3 | Text  Name | Переместить грустное лицо  btnFrown |

1. Запустите VisualStudio и создайте новый проект.
2. Отобразите форму проекта и используйте элемент управления *PictureBox* и *Button* для добавления объектов боксов для рисунков и кнопок.
3. Измените свойства*Image*, *Name*, *SizeMode* элементов *PictureBox1* и *PictureBox2* на значения, указанные в таблице выше.
4. Измените свойства кнопок Button1, Button2 и Button3 на значения, указанные в таблице выше.
5. Дважды щелкните на кнопке формы **Переместить улыбающееся лицо**  и создайте такую процедуру обработки события:



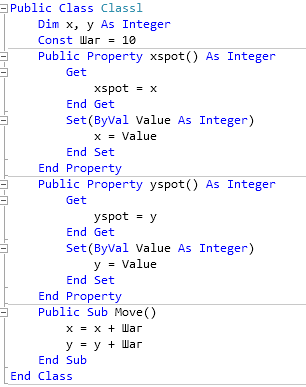
1. Отобразите форму, сделайте двойной щелчок мышью на кнопке **Переместить грустное лицо** и создайте такую процедуру обработки события:



1. Снова отобразите форму, сделайте двойной щелчок мышью на кнопке **Выход**, а затем введите в процедуре события следующий оператор программы:



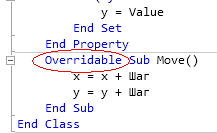
1. Создайте отдельный файл класса *(Проект/Добавить класс)* и наберите следующее:



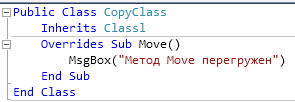
1. Проект готов, запустите его и протестируйте.

**Действия по перегрузке методов**

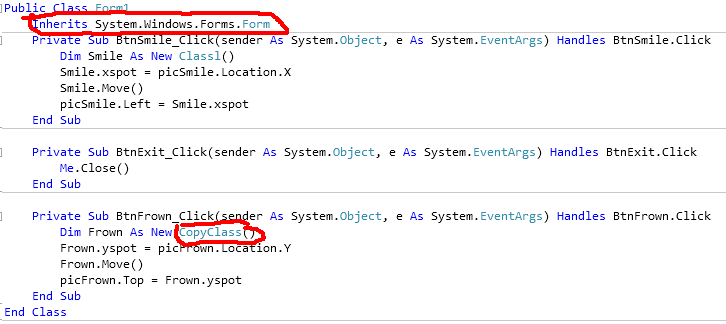
1. Измените коды класса Classl.vb, заменив для метода слово Public словом Overridabie:



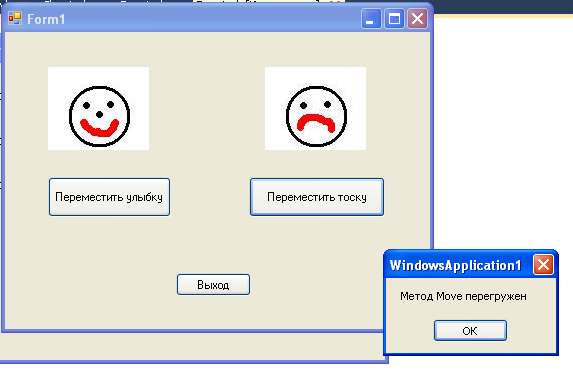
1. Затем создайте новый файл классов (*Проект/Добавить класс*), назовите его CopyClass и наберите для него код:



1. Измените коды в файле формы таким образом, чтобы был использован как оригинальный файл классов (Class1), так и наследованный (Copy Class):



Если теперь вы щелкнете на кнопке формы **Переместить улыбающееся лицо**, улыбающееся лицо, как и прежде, будет перемещаться по экрану. Если же вы щелкнете на кнопке **Переместить грустное лицо**, Visual Basic.NET обратится к наследованному классу CopyClass, коды которого были перегружены, и на экране появится сообщение "Метод Move перегружен".



**Контрольные вопросы:**

1. Что такое перегрузка методов.
2. Перечислить и охарактеризовать модификаторы, которые используются для управления переопределением свойств и методов.
3. Охарактеризуйте ключевые слова MyBase и MyClass.
4. Опишите действия, необходимые для перегрузки.

Лабораторная работа №5

**Создание проекта с использованием кнопочных компонентов**

**Цель:** создание Windows-приложения, аналогичного стандартному калькулятору Windows в среде Visual Studio.

## Ход работы:

## Выполнить задания, следуя указаниям (задание 1-го уровня на «3», задание 2-го уровня на «4», задание 3-го уровня на «5»,).

## Ответить на контрольные вопросы (в устной форме).

## Предъявить преподавателю результаты работы программы и исходные коды.

## Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, программный код, результаты тестирования программы, выводы).

#### Задание на лабораторную работу.

**Задание 1-го уровня**

1. Создать новый проект.

2. Составить эскиз интерактивной формы - калькулятора, способной выполнять простые арифметические действия (Рис. 1).

3. Задать значения свойств элементов управления, размещенных на интерактивной форме.

4. Для каждого элемента управления написать программный код, соответствующий событию активизации (нажатия) элемента управления.

5. Осуществить сборку и компиляцию модулей проекта.

6. Выполнить вычисления с помощью созданного калькулятора.

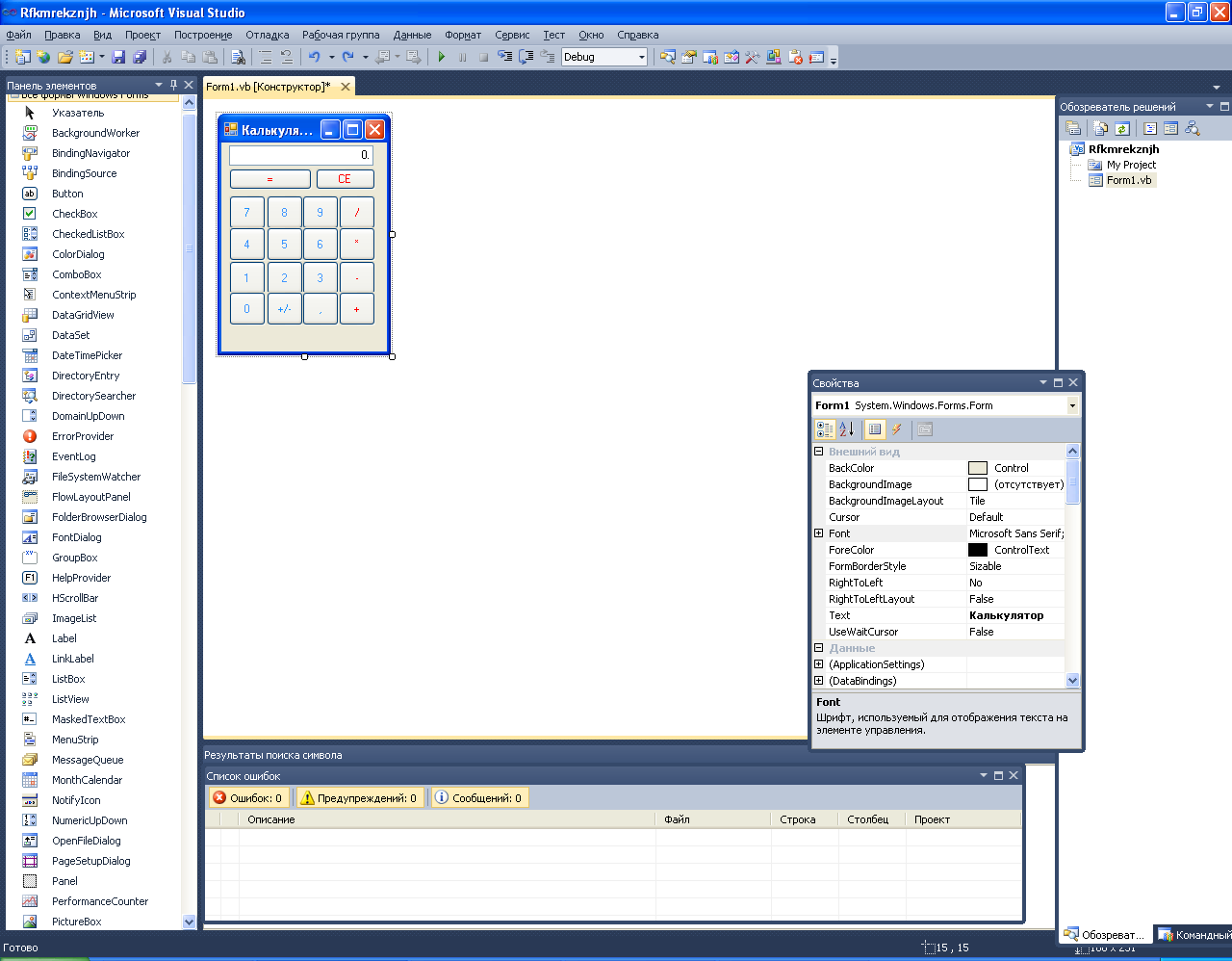


Рис. 1. Эскиз интерактивной формы

**Задание 2-го уровня.** Доработайте калькулятор из первого задания, добавив следующие возможности: вычисление квадратного корня, процентов, обратного числа, стирание одного символа, стирание числа (Рис. 2).



Рис. 2. Эскиз интерактивной формы (2-й уровень)

**Задание 3-го уровня.** Доработайте калькулятор из первого и второго заданий, добавив возможности работы с памятью: стереть память, вывести из памяти, записать в память, добавить в память (Рис. 3).

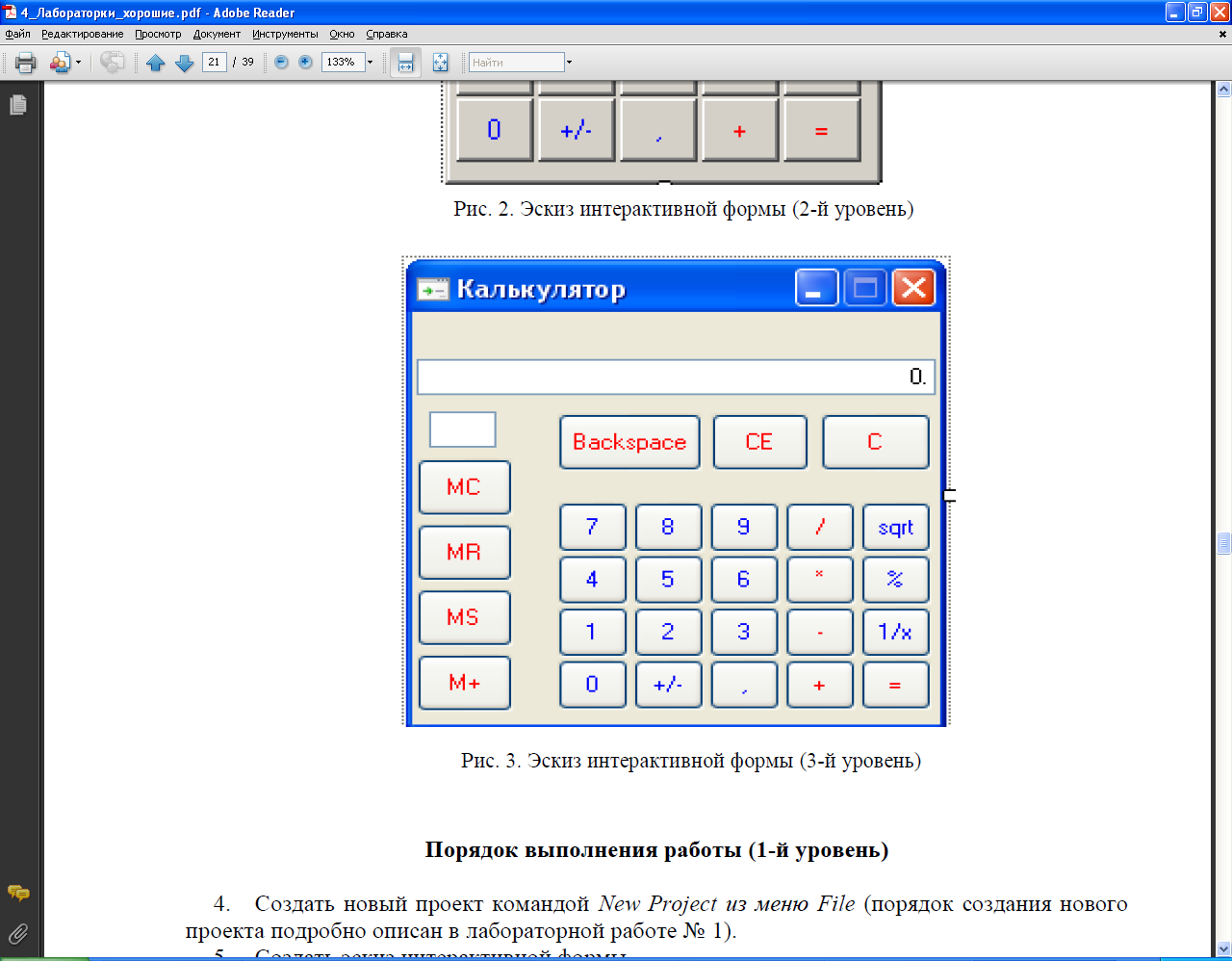


Рис. 3. Эскиз интерактивной формы (3-й уровень)

**Порядок выполнения работы (1-й уровень)**

1. Создать новый проект командой *Новый проект* из меню *Файл* (порядок создания нового проекта подробно описан в лабораторной работе № 1).

2. Создать эскиз интерактивной формы.

Используя панель инструментов *ToolBox*, разместить на форме элементы управления (кнопки - *Button1 - Button18* и текстовое поле - *TextBox1*), как показано на Рис. 4.

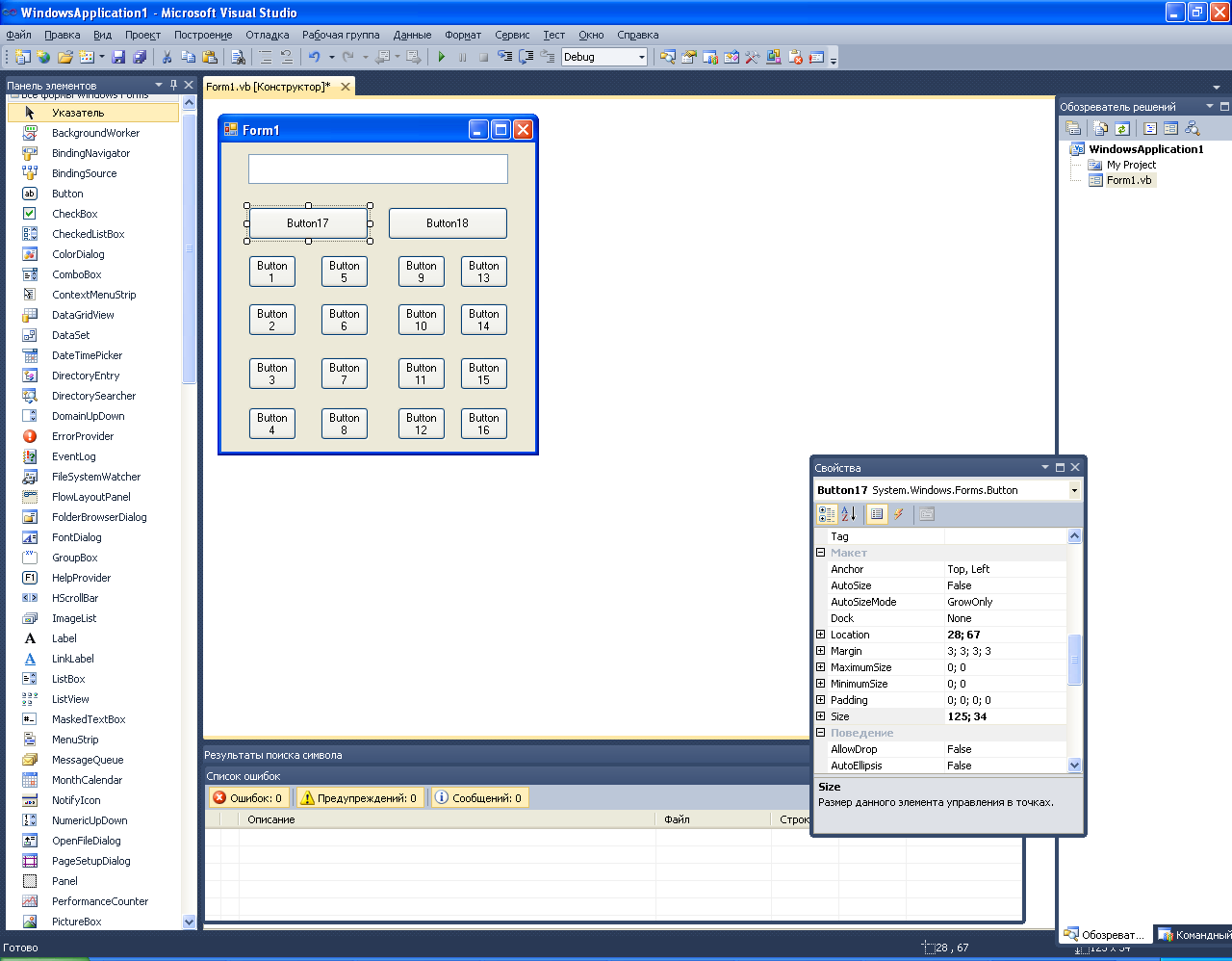


Рис. 4. Размещение элементов управления на форме

После размещения всех необходимых элементов управления на форме необходимо задать их свойства через панель Свойства (*Properties)*, которая появляется после одинарного щелчка мышью по нужному элементу управления, расположенному на форме. Каждый элемент управления имеет свой набор свойств. Свойства можно назначать не только элементам управления, но и форме.

3.1. Установите значения свойств *MaximizeBox*, *Size* и *Text* объекта *Form1*, как показано на Рис. 5.

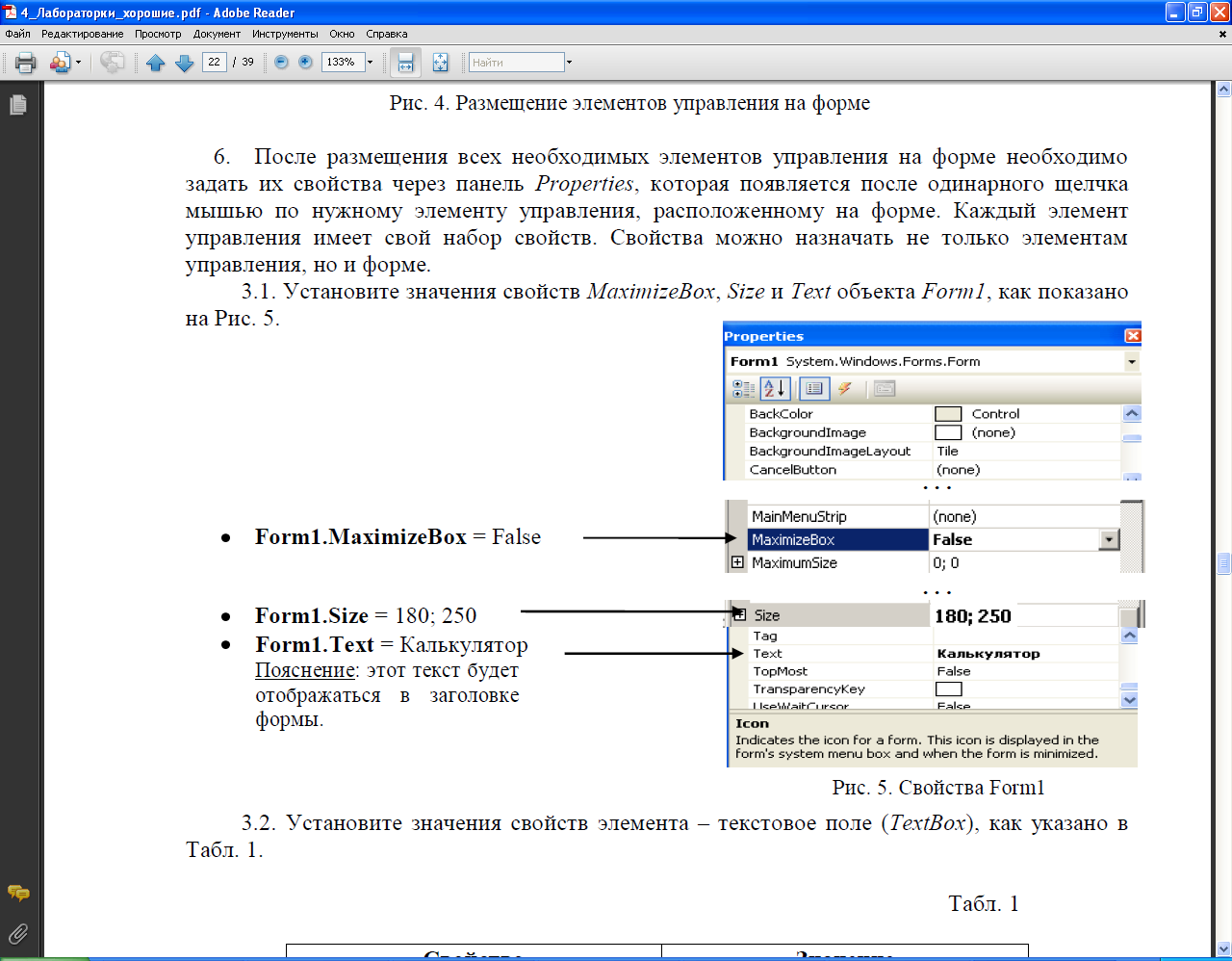


Рис. 5. Свойства Form1

3.2. Установите значения свойств элемента – текстовое поле (*TextBox*), как указано в Таблице 1.

|  |  |
| --- | --- |
| **Свойство** | **Значение** |
| TextBox1.Name | TextBox1 |
| TextBox1.Text | 0. |
| TextBox1.BorderStyle | Fixed3D  Пояснение: щелкнуть на кнопку в правом поле, затем с помощью окна настройки установить значение данного свойства |
| TextBox1.TextAlign | Right |

3.3. Установите значения свойств элементов – кнопок (*Button*) как указано в Таблице 2.

Таблица 2

|  |  |
| --- | --- |
| **Свойство** | **Значение** |
| Button1.Name | bt7 |
| Button1.ForeColor | Голубой |
| Button1.Text | 7 |
| Button2.Name | bt4 |
| Button2.ForeColor | Голубой |
| Button2.Text | 4 |
| Button3.Name | bt1 |
| Button3.ForeColor | Голубой |
| Button3.Text | 1 |
| Button4.Name | bt0 |
| Button4.ForeColor | Голубой |
| Button4.Text | 0 |
| Button5.Name | bt8 |
| Button5.ForeColor | Голубой |
| Button5.Text | 8 |
| Button6.Name | bt5 |
| Button6.ForeColor | Голубой |
| Button6.Text | 5 |
| Button7.Name | bt2 |
| Button7.ForeColor | Голубой |
| Button7.Text | 2 |
| Button8.Name | btpm |
| Button8.ForeColor | Голубой |
| Button8.Text | +/- |
| Button9.Name | bt9 |
| Button9.ForeColor | Голубой |
| Button9.Text | 9 |
| Button10.Name | bt6 |
| Button10.ForeColor | Голубой |
| Button10.Text | 6 |
| Button11.Name | bt3 |
| Button11.ForeColor | Голубой |
| Button11.Text | 3 |
| Button12.Name | btpoint |
| Button12.ForeColor | Голубой |
| Button12.Text | , |
| Button13.Name | btdel |
| Button13.ForeColor | Красный |
| Button13.Text | / |
| Button14.Name | btpr |
| Button14.ForeColor | Красный |
| Button14.Text | \* |
| Button15.Name | btmin |
| Button15.ForeColor | Красный |
| Button15.Text | - |
| Button16.Name | btplus |
| Button16.ForeColor | Красный |
| Button16.Text | + |
| Button17.Name | btis |
| Button17.ForeColor | Красный |
| Button17.Text | = |
| Button18.Name | btce |
| Button18.ForeColor | Красный |
| Button18.Text | CE |

В результате изменения свойств вышеперечисленных объектов форма *Form1* примет вид, указанный на Рис. 1.

4. Написание программы (кода) включает в себя разработку кода для обработки событий нажатия всех кнопок.

4.1. Выполните двойной щелчок левой кнопкой мыши на пустом месте формы. В появившемся окне головного модуля *Form1.vb* выберете блок Объявление, как показано на Рис. 6, и введите программный код, объявляющий переменные:

- *IsText* (для хранения содержимого текстовой строки);

- *IsNumber* (для хранения числа);

- *Point* (для указания разделителя дробной части);

- *op* (для хранения номера арифметической операции).

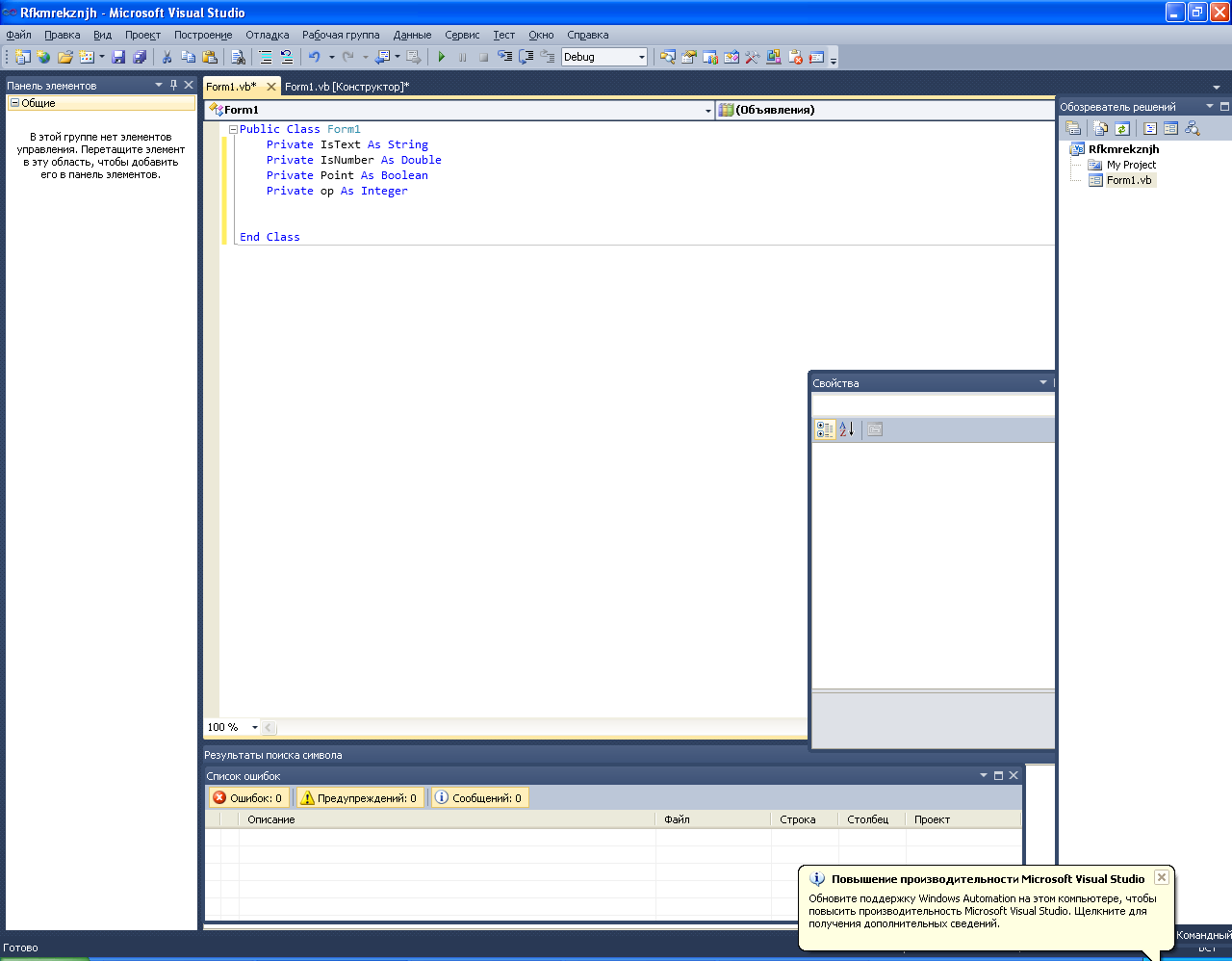


Рис. 6. Объявление переменных в блоке Form1 - Объявления

4.2. Обработка нажатия цифровых клавиш: 1, 2 … 9, 0.

4.2.1. Введите программный код для обработки события – нажатия кнопки «1» (*bt1\_Click*). Для этого необходимо выполнить двойной щелчок левой кнопкой мыши по кнопке *bt1* и ввести код:



**Пояснение:** данный фрагмент кода сначала проверяет, не является ли вводимая цифра первой в числе и не была ли нажата клавиша арифметической операции (+, -, /, \*), в этом случае вводимая цифра заменяет содержимое текстового поля (**TextBox1.Text**). В противном случае вводимая цифра добавляется к содержимому текстового поля (**TextBox1.Text**).

4.2.2. Введите программный код для обработки события – нажатия кнопки «2» (*bt2\_Click*). Для этого необходимо выполнить двойной щелчок левой кнопкой мыши по кнопке *bt2* и ввести код аналогичный коду п. 4.2.1:



4.2.3. По аналогии с п. 4.2.1, 4.2.2 введите программный код для обработки нажатия оставшихся цифровых кнопок (*bt3 – bt9,bt0*). Для экономии времени можете копировать повторяющиеся фрагменты кода.

4.3. Введите программный код для обработки события – нажатия кнопки «смена знака числа (+/-)» (*btpm\_Click*). Для этого необходимо выполнить двойной щелчок левой кнопкой мыши по кнопке *btpm* и ввести код:



**Пояснение:** функция **Val()** преобразует текстовый тип в числовой; функция **CStr()** преобразует числовой тип в текстовый.

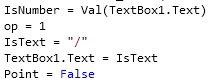
4.4. Введите программный код для обработки события – нажатия кнопки «запятая, отделяющая целую часть от дробной ( , )» (*btpoint\_Click*). Для этого необходимо выполнить двойной щелчок левой кнопкой мыши по кнопке *btpoint*и ввести код:



**Пояснение:** данный фрагмент кода через переменную *Point* сначала проверяет, не была ли кнопка «запятая» нажата ранее при вводе текущего числа.

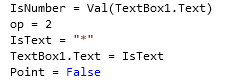
4.5. Обработка нажатия кнопок арифметических действий: /, \*, -, +.

4.5.1. Введите программный код для обработки события – нажатия кнопки «деление ( / )» (*btdel\_Click*). Для этого необходимо выполнить двойной щелчок левой кнопкой мыши по кнопке *btdel* и ввести код:



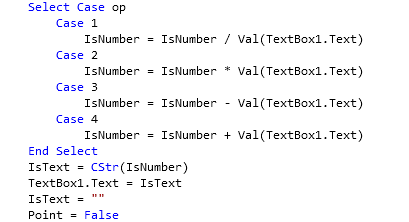
**Пояснение:** в данном фрагменте кода переменной *op* присваивается номер арифметической операции. При этом деление соответствует первому номеру, умножение – второму, вычитание - третьему, сложение – четвертому. Значение переменной *op* будет использоваться при вычислении результата (нажатие кнопки *btis*).

4.5.2. Введите программный код для обработки события – нажатия кнопки «умножение (\*)» (*btpr\_Click*). Для этого необходимо выполнить двойной щелчок левой кнопкой мыши по кнопке *btpr* и ввести код, аналогичный коду п. 4.5.1, изменив номер операции на второй:



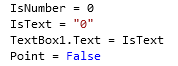
4.5.3. По аналогии с п. 4.5.1, 4.5.2 введите программный код для обработки событий – нажатия кнопок «вычитание ( - )» (*btmin\_Click*) и «сложение ( + )» (*btplus\_Click*), изменив соответственно номера операций и символы, их отображающие.

4.6. Введите программный код для обработки события – нажатия кнопки «=» (*btis\_Click*). Для этого необходимо выполнить двойной щелчок левой кнопкой мыши по кнопке *btis* и ввести код:



**Пояснение:** в данном фрагменте кода используется конструкция **SelectCase**, позволяющая выбирать выполняемую часть кода в зависимости от значения переменной *op*.

4.7. Введите программный код для обработки события – нажатия кнопки «CE» (*btce\_Click*). Для этого необходимо выполнить двойной щелчок левой кнопкой мыши по кнопке *btce* и ввести код:



1. Выполните сборку и компиляцию модулей проекта.
2. Запустите проект и протестируйте.

**Пояснения для выполнения задания 2-го уровня**

* 1. Для вычисления квадратного корня используйте функцию **Math.Sqrt()**.
  2. Для обработки события – нажатия кнопки «Backspace» (удаление последнего введѐнного символа) можно использовать функцию **Remove()**, удаляющую указанное число символов в текстовой переменной начиная с указанной позиции, и свойство **Length**, возвращающее число символов в значении переменной. Например, так:

**IsText = IsText.Remove(IsText.Length - 1, 1)**.

* 1. Нажатие кнопки «С» должно привести к стиранию только текущего набираемого числа, например если Вы ошиблись при вводе и хотите стереть число не посимвольно кнопкой «Backspace», а сразу целиком. Отличается от кнопки «СЕ» тем, что не стирает предыдущие набранные числа и операции.

**Пояснения для выполнения задания 3-го уровня**

В текстовом поле над кнопками по работе с памятью (М+, MS, MR, MC) должен отображаться символ «М», если в памяти содержится какое-либо число.

**Контрольные вопросы:**

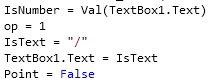
1. Назначение функций Val() и CStr().
2. Поясните данный фрагмент кода: .



1. Перечислите свойства кнопки, используемые для задания отображаемого текста и его цвета.
2. Поясните данный фрагмент кода:



1. Назначение конструкции **SelectCase**
2. Поясните данный фрагмент кода:



Лабораторная работа №6

**Создание проекта с использованием компонентов для работы с текстом**

**Цель:** создание Windows-приложения, обеспечивающего возможность решения уравнения и построения графика функции. Приобретение навыков использования графического элемента управления *PictureBox* при проектировании интерфейса Windows-приложения в среде Visual Studio.

## Ход работы:

## Выполнить задания, следуя указаниям (задание 1-го уровня на «3», задание 2-го уровня на «4», задание 3-го уровня на «5»,).

## Ответить на контрольные вопросы (в устной форме).

## Предъявить преподавателю результаты работы программы и исходные коды.

## Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, программный код, результаты тестирования программы, выводы).

#### Задание на лабораторную работу.

**Задание 1-го уровня**

1. Создать новый проект.

2. Составить эскиз интерактивной формы (Рис.1).

3. Задать значения свойств элементов управления, размещенных на интерактивной форме.

4. Составить программу для нахождения корней функции *f(x)* на интервале *[A, B ]*с шагом *E*, предусмотрев ввод исходных данных через текстовые поля интерактивной формы. Функцию *f(x)* выбрать из Табл. 6 в соответствии со своим вариантом.

5. Осуществить сборку и компиляцию модулей проекта.

6. Решить уравнение *f(x) = 0*.

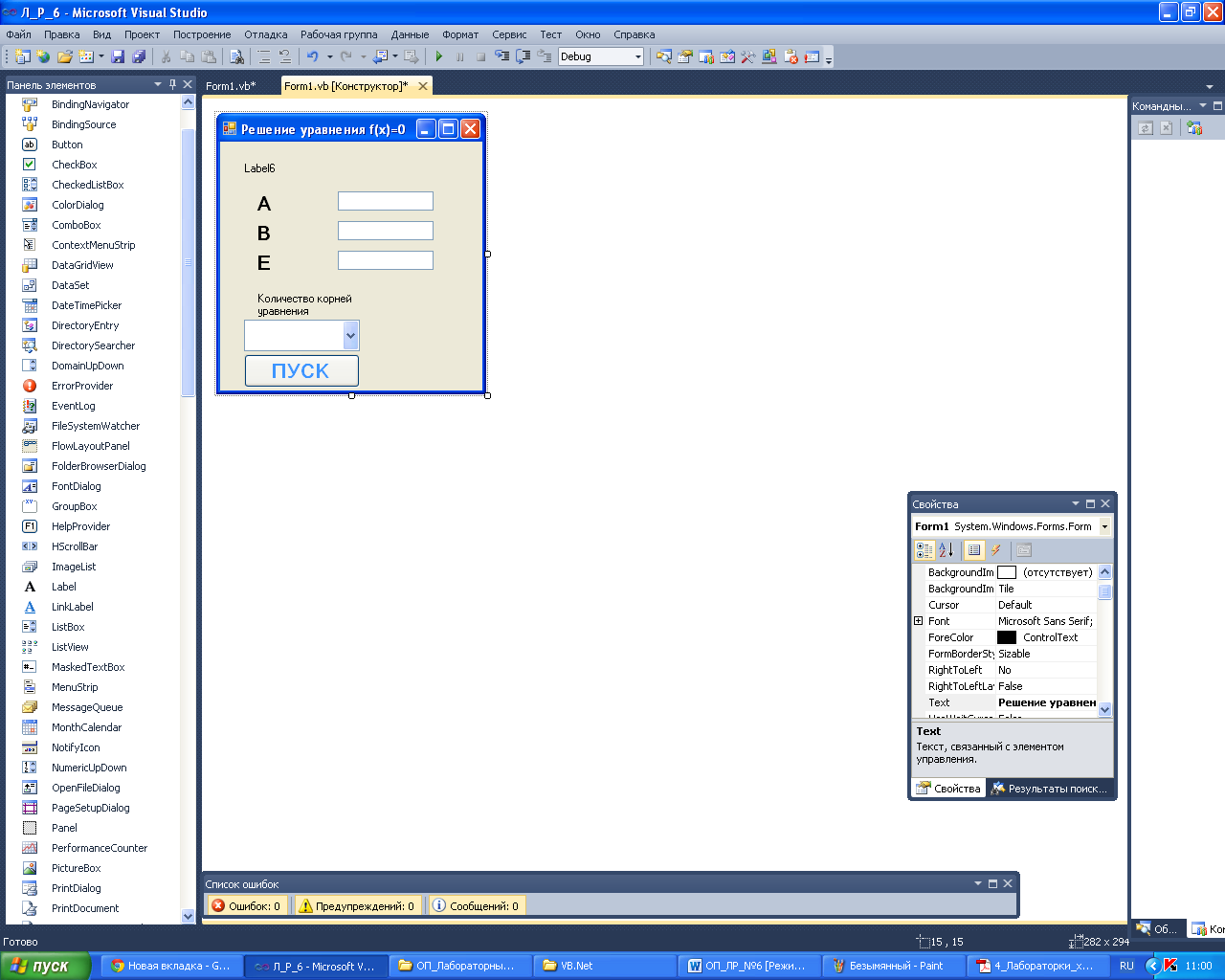


Рис. 1. Эскиз интерактивной формы

**Задание 2-го уровня.** Реализовать построение графика и отображение графика функции в элементе управления *PictureBox*(Рис. 2).

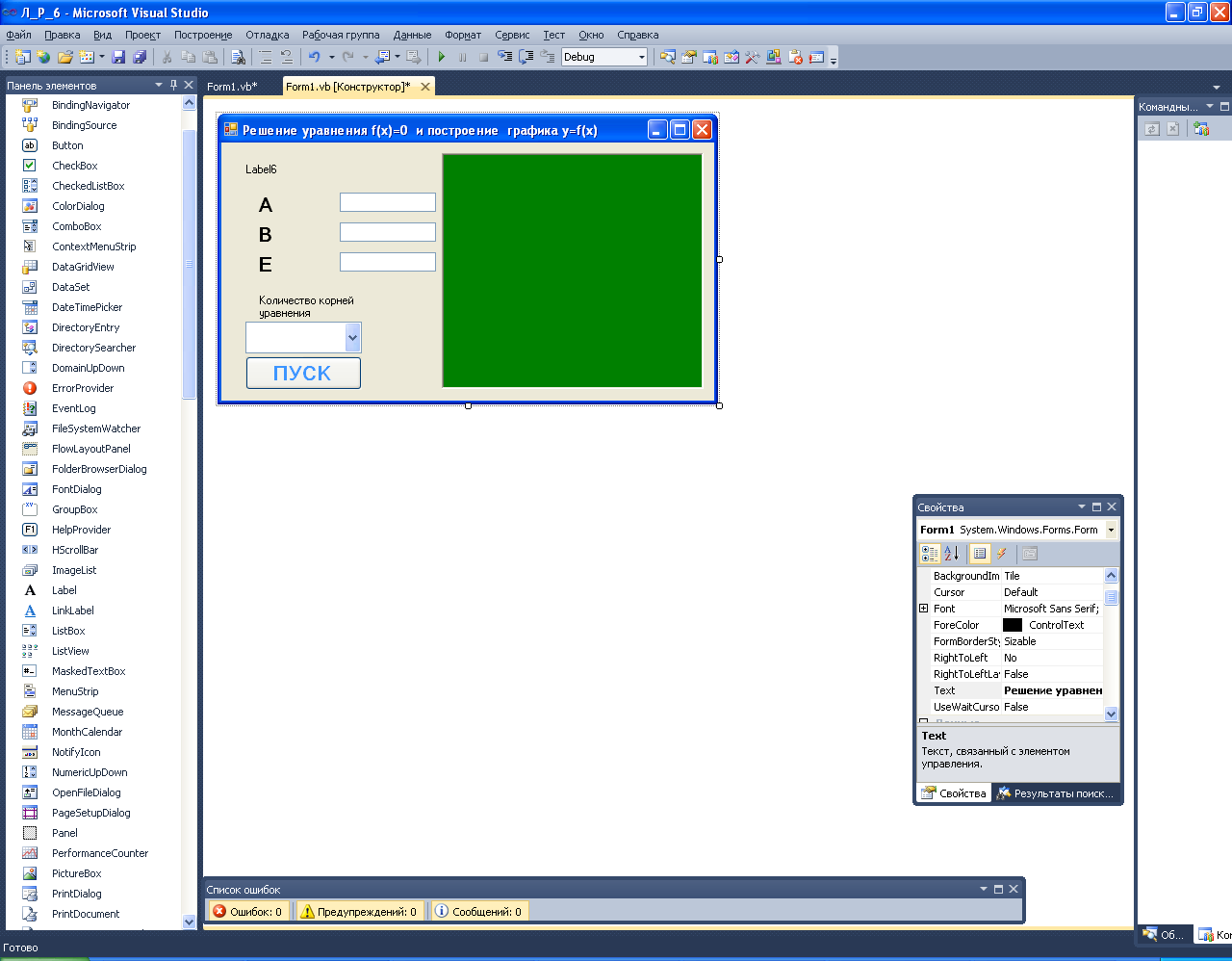


Рис. 2. Эскиз интерактивной формы с построением графика

**Задание 3-го уровня.** Реализовать возможность задавать пользователем функцию (полином до третьей степени), предусмотрев ввод параметров функции через текстовые поля интерактивной формы (Рис. 3).

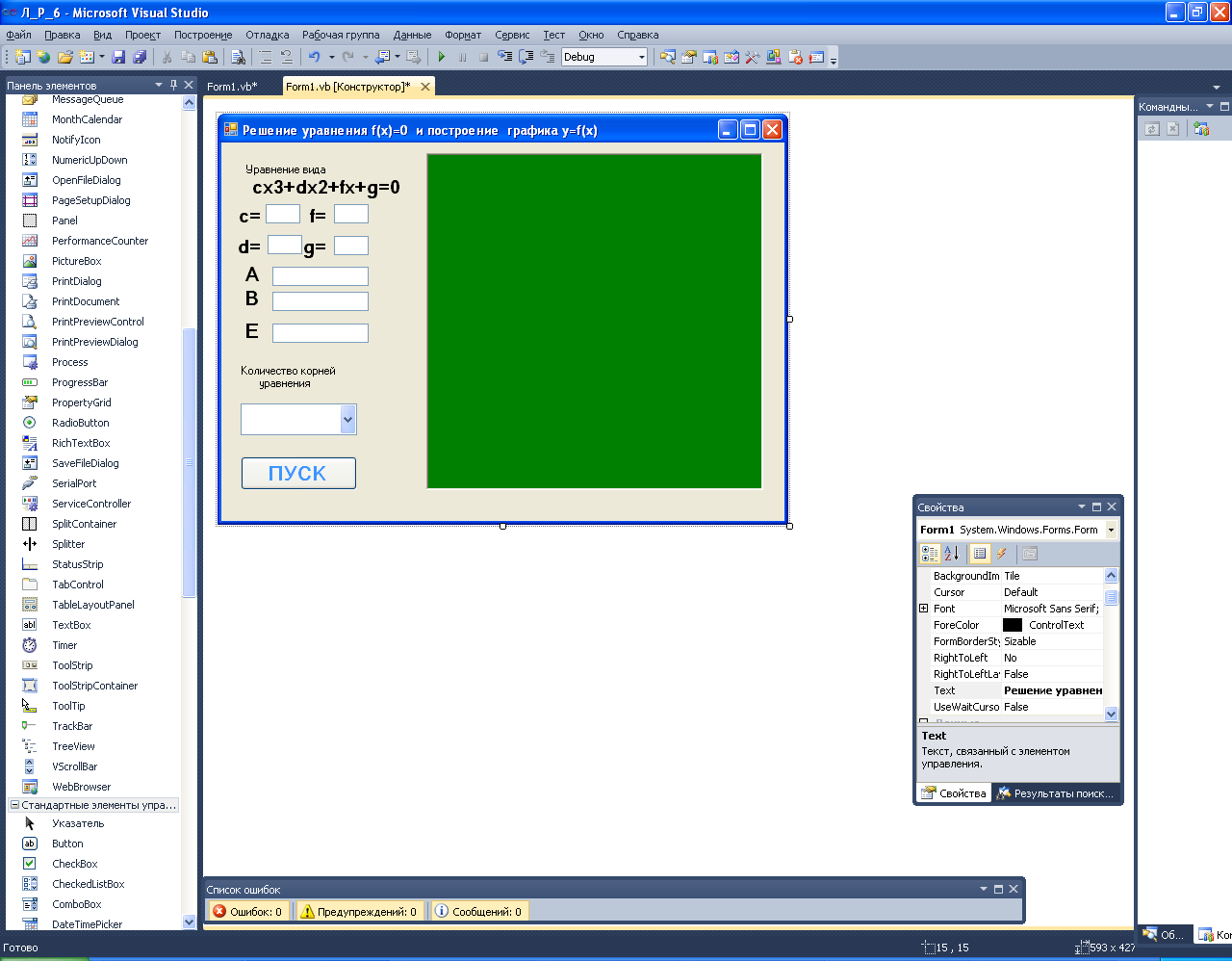


Рис. 3. Эскиз интерактивной формы с построением графика и пользовательским вводом функции

**Порядок выполнения работы (1-й уровень)**

1. Создать новый проект командой Создать проект (New Project) из меню Файл (File) (порядок создания нового проекта подробно описан в лабораторной работе № 1).

2. Создать эскиз интерактивной формы.

2.1. Используя панель инструментов *ToolBox*, разместить на форме элементы управления (кнопку - *Button1*, надписи *– Label1 – Label6*, текстовые поля - *TextBox1 TextBox3*, поле со списком - *ComboBox1* и графическое поле - *PictureBox1*), как показано на Рис. 4. Элемент управления *ComboBox*- текстовое поле с предопределѐнным списком значений, из которого можно выбрать одно из имеющихся значений. В данной работе в *ComboBox* будут отображаться значения вычисленных корней уравнения.

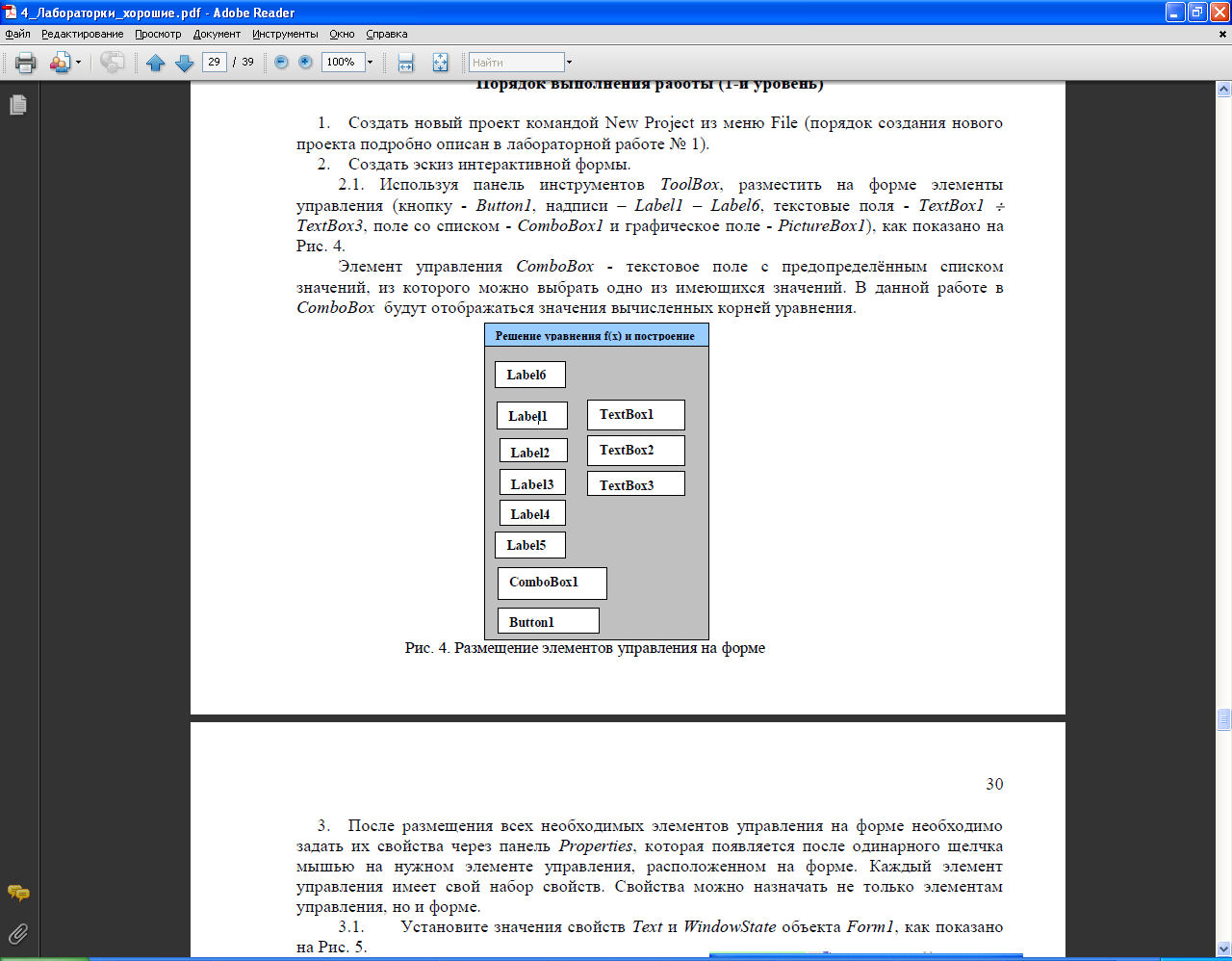
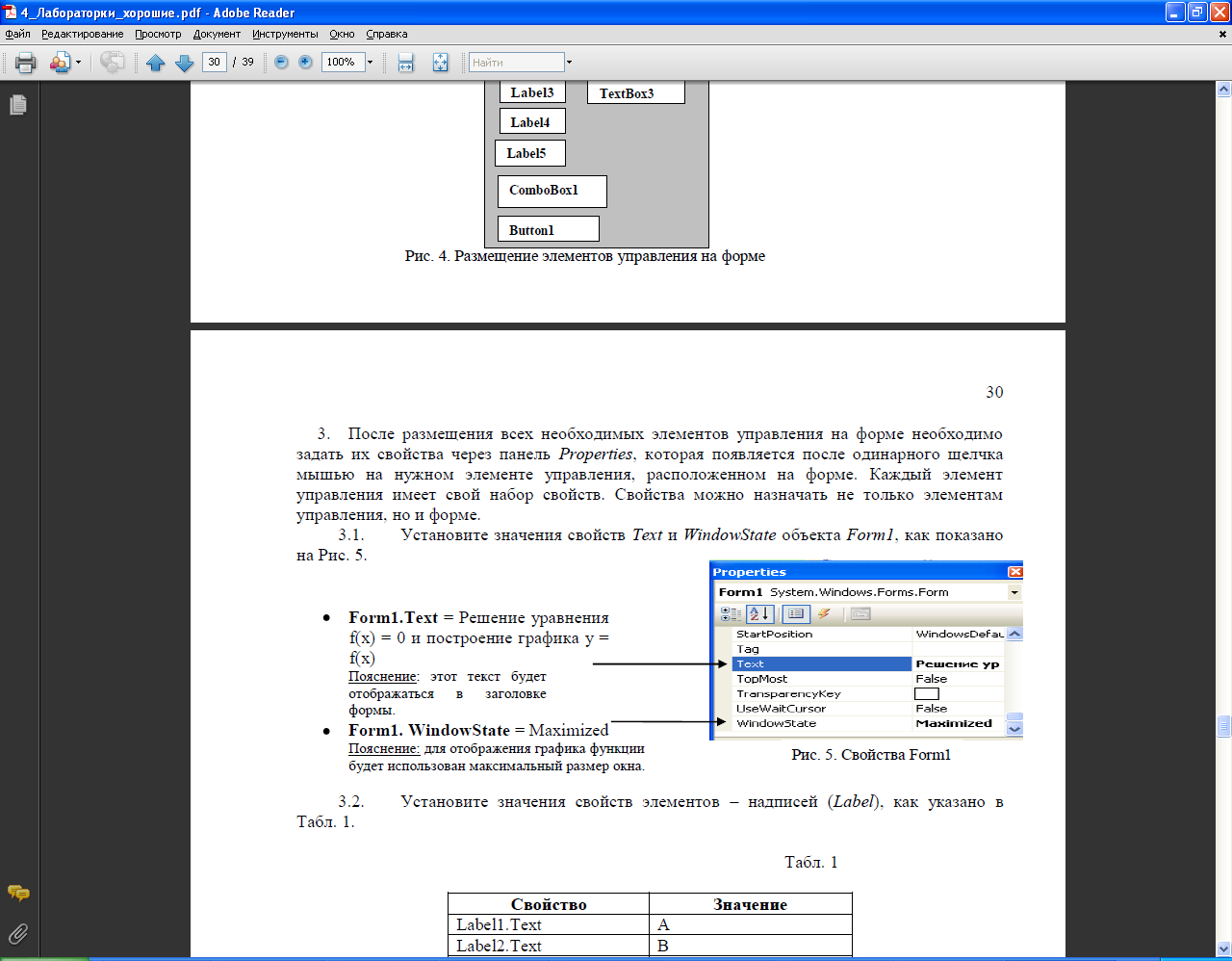


Рис. 4. Размещение элементов управления на форме

3. После размещения всех необходимых элементов управления на форме необходимо задать их свойства через панель *Свойства* (*Properties)*, которая появляется после одинарного щелчка мышью на нужном элементе управления, расположенном на форме. Каждый элемент управления имеет свой набор свойств. Свойства можно назначать не только элементам управления, но и форме.

3.1. Установите значения свойств *Text* и *WindowState* объекта *Form1*, как показано на Рис. 5.



3.2. Установите значения свойств элементов – надписей (*Label*), как указано в Таблице 1.

Таблице 1

|  |  |
| --- | --- |
| **Свойство** | **Значение** |
| Label1.Text | A |
| Label2.Text | B |
| Label3.Text | E |
| Label1.Font | жирный, 16 пт. |
| Label2.Font | жирный, 16 пт. |
| Label3.Font | жирный, 16 пт. |
| Label4.Text | Количество корней |
| Label5.Text | Уравнения |
| Label6.Text | |

3.3. Установите значения свойств элементов – текстовых полей (*TextBox*), как указано в Таблице 2.

Таблица 2

|  |  |
| --- | --- |
| **Свойство** | **Значение** |
| TextBox1.Font | жирный, 16 пт. |
| TextBox2.Font | жирный, 16 пт. |
| TextBox3.Font | жирный, 16 пт. |

3.4. Установите значения свойств элемента – кнопки (*Button*), как указано в Таблице 3.

Таблице 3

|  |  |
| --- | --- |
| **Свойство** | **Значение** |
| Button1.BackColor | СИНИЙ |
| Button1.Font | жирный, 16 пт. |
| Button1.Text | ПУСК |

Пояснение: для установки нужного цвета необходимо щелкнуть на кнопку в правом поле, перейти на вкладку Custom и выбрать из палитры цветов нужный цвет, например синий.

3.5. Установите значения свойств элемента – поля со списком (*ComboBox*), как указано в Таблице 4.

Таблице 4

|  |  |
| --- | --- |
| **Свойство** | **Значение** |
| ComboBox1.DropDownStyle | DropDownList |
| ComboBox1.Font | жирный, 16 пт. |

В результате изменения свойств вышеперечисленных объектов форма *Form1* примет вид, указанный на Рис. 1.

4. Написание программы (кода) включает в себя разработку кода для обработки событий формы и всех элементов управления. В качестве примера рассмотрим функцию *f(X) = X2 - 2X – 10*.

4.1. Для объявления глобальных переменных выполните двойной щелчок левой кнопкой мыши на форме. В появившемся окне головного модуля *Form1.vb* выберете блок Объявление*(Declarations*), как показано на Рис. 6, и введите программный код, объявляющий переменные:

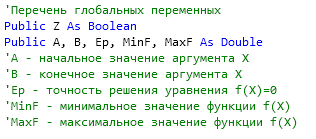


Рис. 6. Обработка события в блоке Общие (General) – Объявление (Declarations)

4.2. Для обработки события – загрузки формы (**Form1\_Load**) выберете блок Load (как показано на Рис. 7) и

ведите программный код:

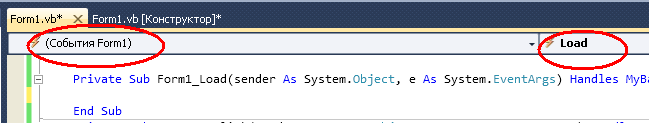
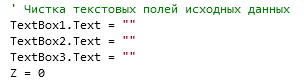
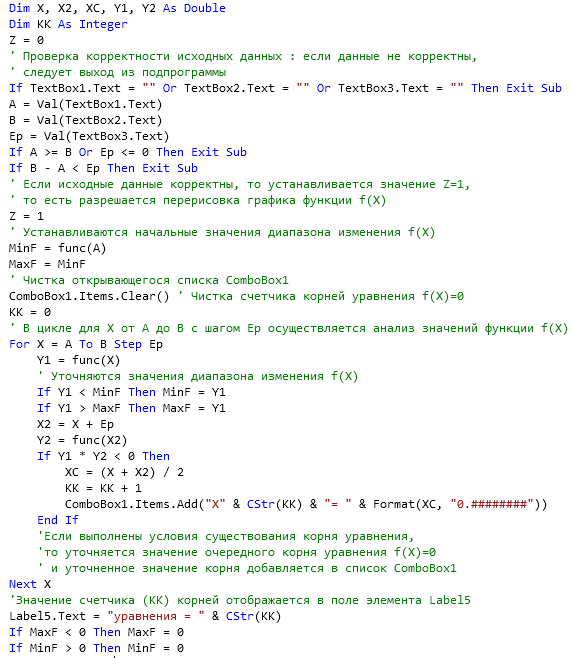


Рис. 7. Обработка события в блоке Form1 - Load

4.3. Написать программный код, обрабатывающий событие «нажатие кнопки пуск»(Button1\_Click).



1. Сборка и компиляция модулей проекта выполняется командой *Построить решение (Build Windows Application)*из меню *Построение* (*Build)*. Запустить приложение на выполнение можно командой Начать отладку (*Start Debugging)*из меню *Отладка* (*Debug).* В появившейся форме (Рис. 1) ввести с клавиатуры значения исходных данных: - **A** - начало интервала табулирования функции; - **B** - конец интервала табулирования функции; - **E** - шаг вычисления корней уравнения. Для выполнения вычислений нажать кнопку ПУСК. Покажите преподавателю результаты работы.

**Пояснения для выполнения задания 2-го уровня**

1. Для отображения графика функции можно использовать элемент управления *PictureBox*, позволяющий размещать графические примитивы (точку, отрезок, простые геометрические фигуры). Разместите элемент управления *PictureBox*на форме, как показано на Рис. 8.

2. Установите значения свойств элемента – графического поля (*PictureBox*), как указано в Таблице 5.

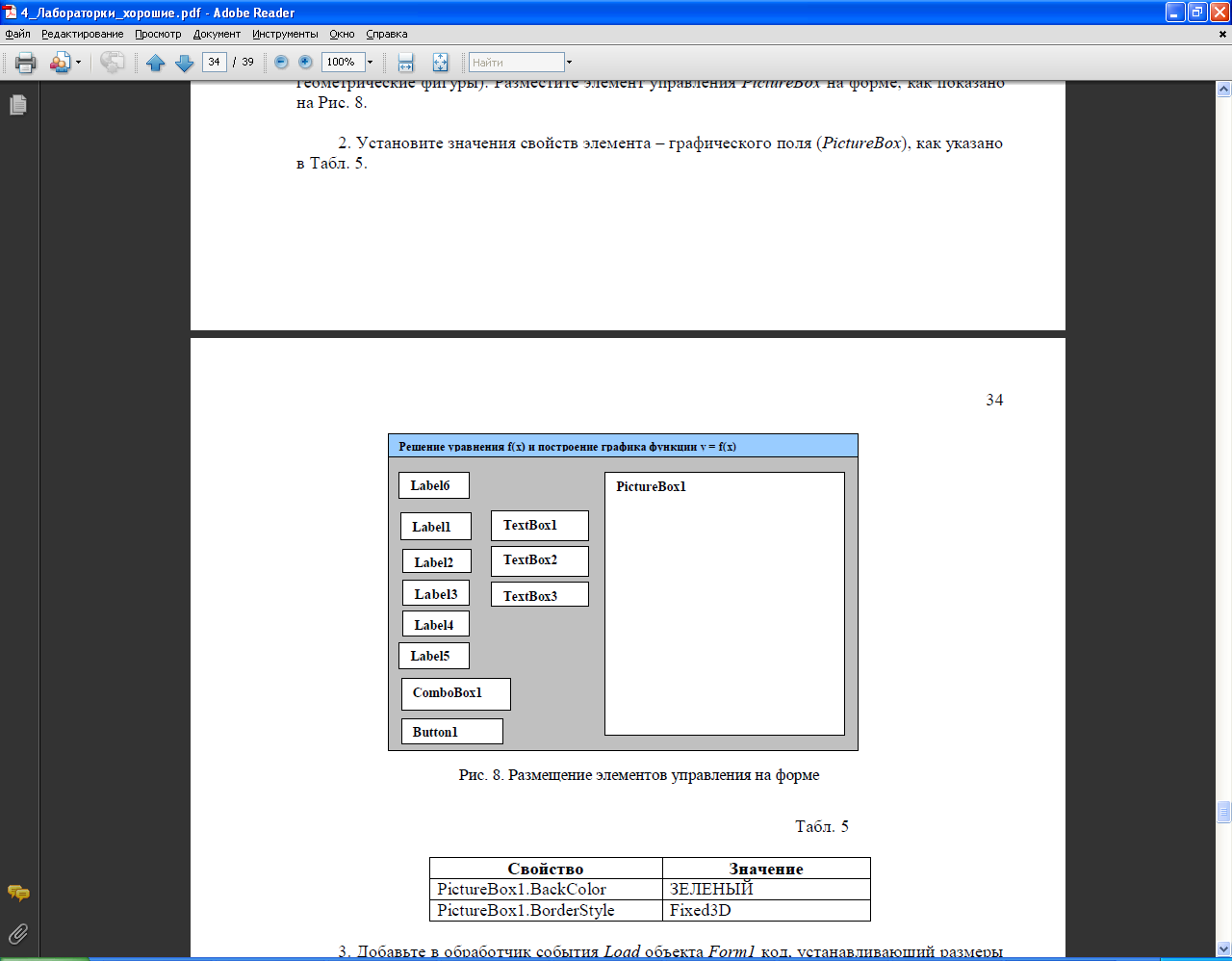


Рис. 8. Размещение элементов управления на форме

Таблица 5

|  |  |
| --- | --- |
| **Свойство** | **Значение** |
| PictureBox1.BackColor | ЗЕЛЕНЫЙ |
| PictureBox1.BorderStyle | Fixed3D |

3. Добавьте в обработчик события *Load* объекта *Form1* код, устанавливающий размеры *PictureBox*:

Добавьте в обработчик события *Click* объекта *Button1* код, выполняющий перерисовку содержимого в *PictureBox*:



5. Для обработки события **Paint**, возникающего при активизации графического элемента **PictureBox1**, необходимо выбрать блок **Paint**(Рис. 9).

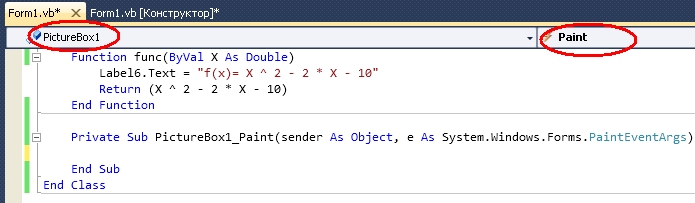
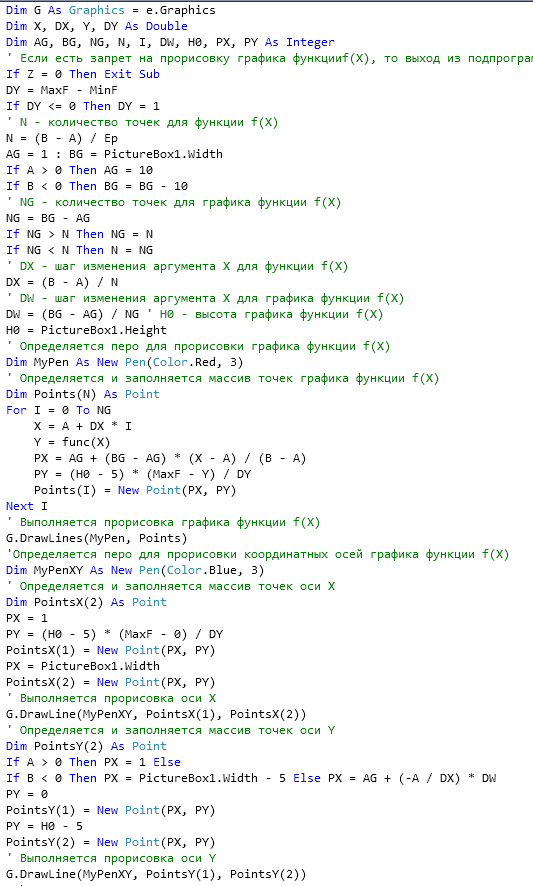


Рис. 9. Обработка события в блоке PictureBox1- Paint 35

И ввести код для подпрограммы **PictureBox1\_Paint**:



После сборки, компиляции и запуска приложения результат расчета в виде графика функции появится на форме в поле элемента **PictureBox1** (Рис. 10):

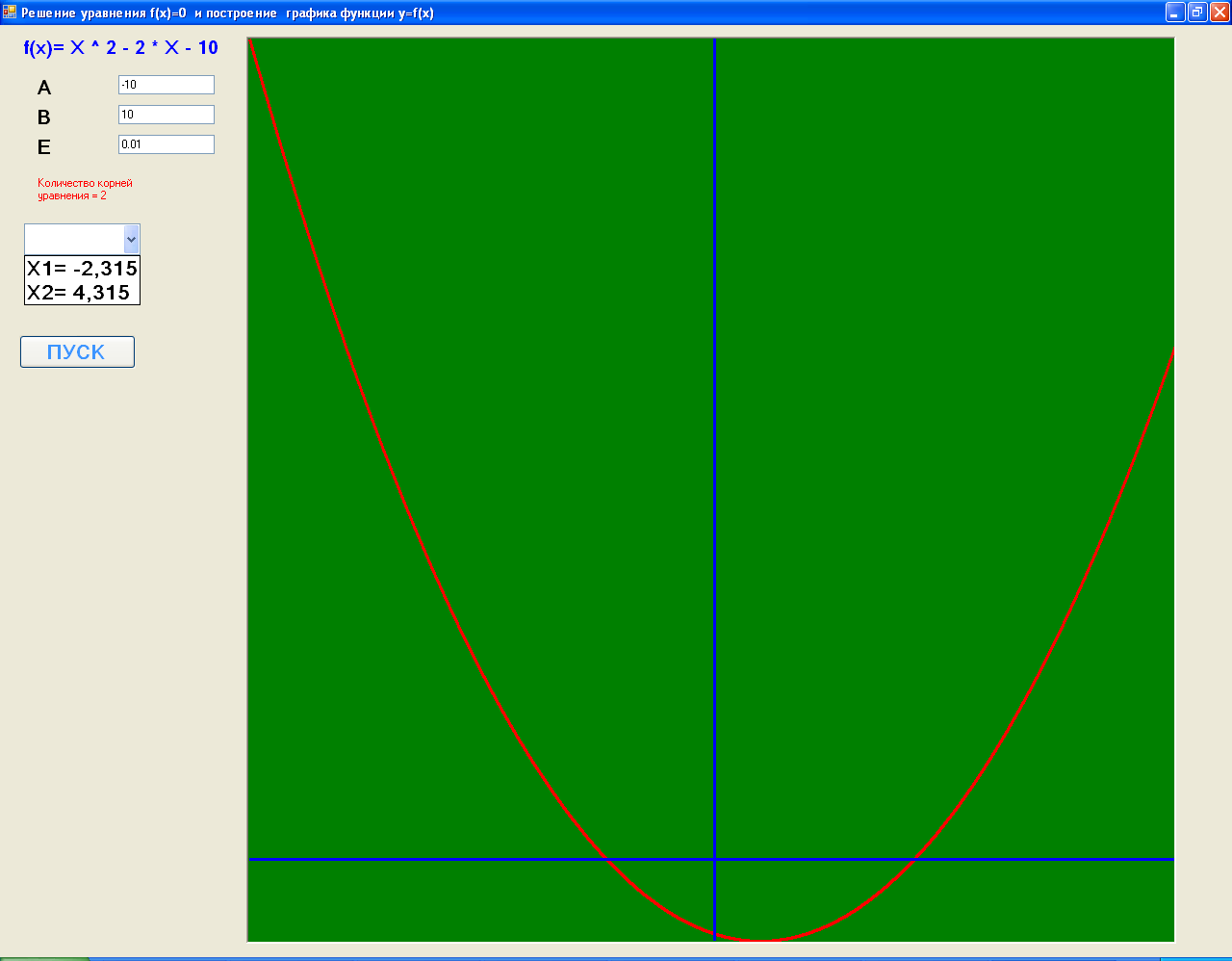
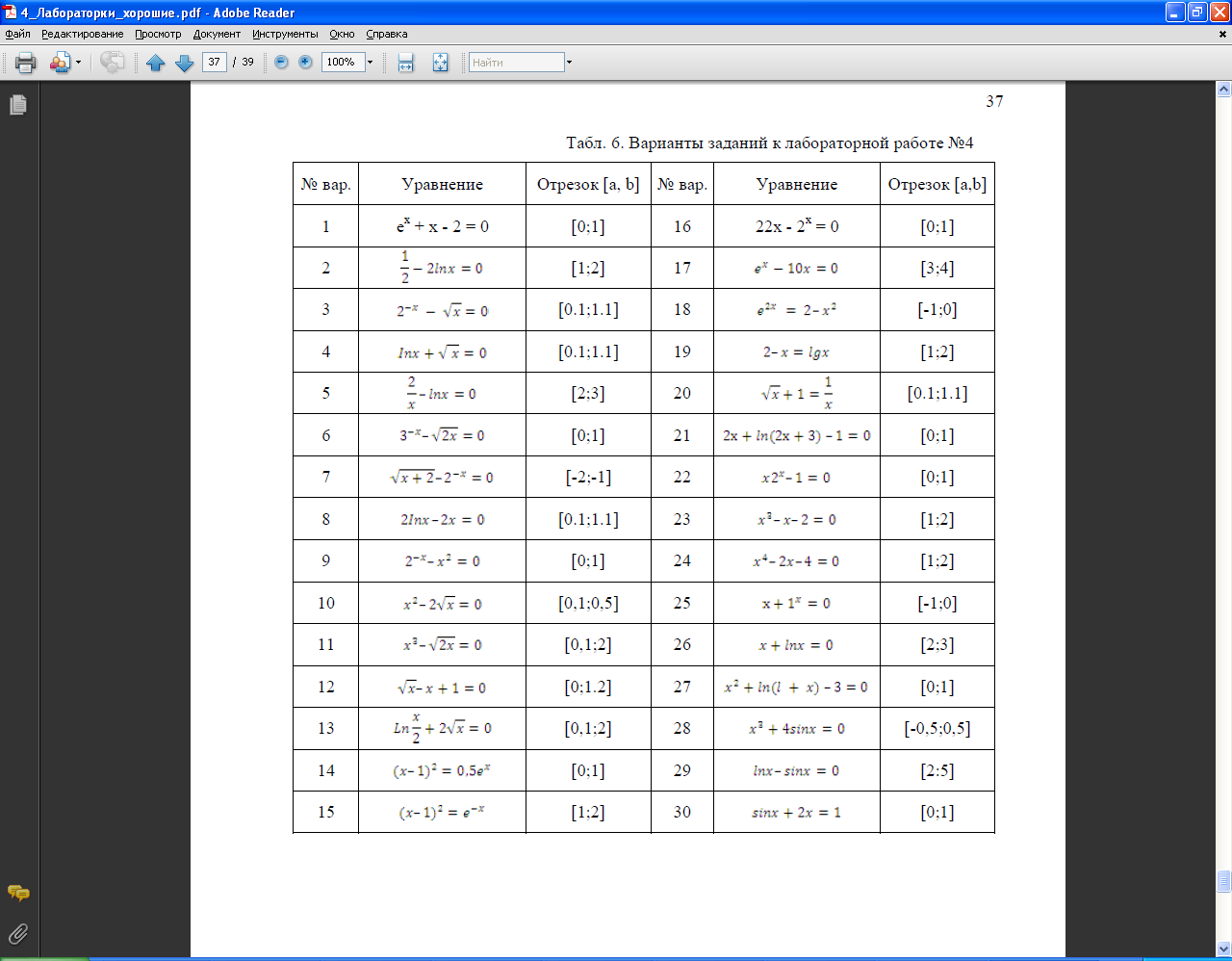


Рис. 10. Результат расчета функции f(x) = x2 - 2x - 10

Выполните несколько вычислений (на разных отрезках [*A, B*] с разным шагом *E*), используя спроектированное приложение.

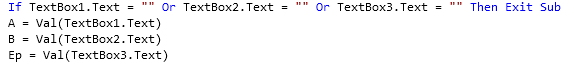
Таблица 6. Варианты заданий к лабораторной работе №6



**Контрольные вопросы:**

1. Перечислите элементы управления для работы с текстом.
2. Поясните данный фрагмент кода:

.



1. Перечислите свойства элементов управления, используемые для задания отображаемого текста и его цвета.
2. Укажите объявление глобальных переменных в коде программы**.**
3. Элемент управления для задания графических примитивов
4. Поясните данный фрагмент кода:



Лабораторная работа №7

**Создание проекта с использованием компонентов ввода и отображения чисел, дат и времени**

**Цель:** научиться извлекать даты, выбранные в элементах управления [*MonthCalendar*](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.monthcalendar%28v=vs.90%29.aspx) и [*DateTimePicker*](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.datetimepicker%28v=vs.90%29.aspx), и отображать их в форме *Windows Forms*, получить навыки отображения диапазона дат и форматирования извлеченных дат различными способами.

## Ход работы:

## Изучить теоретическую часть.

## Выполнить задания, следуя указаниям.

## Ответить на контрольные вопросы (в устной форме).

## Предъявить преподавателю результаты работы программы и исходные коды.

## Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, программный код, результаты тестирования программы, выводы).

**🕮Теоретическая часть**

**Отображение дат. Использование элементов управления MonthCalendar и DateTimePicker**

Использование элемента управления, отображающего календарь, значительно упрощает для пользователя выбор даты. Кроме того, такие элементы управления гарантируют, что дата будет отформатирована правильно. Календарь можно отобразить с помощью элемента управления [*MonthCalendar*](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.monthcalendar%28v=vs.90%29.aspx) или [*DateTimePicker*](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.datetimepicker%28v=vs.90%29.aspx).

Элемент управления [*MonthCalendar*](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.monthcalendar%28v=vs.90%29.aspx) позволяет отображать календарь для одного или нескольких месяцев. При этом пользователи могут выбирать отдельную дату или диапазон дат.

Элемент управления [*DateTimePicker*](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.datetimepicker%28v=vs.90%29.aspx) имеет два состояния. По умолчанию элемент управления [*DateTimePicker*](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.datetimepicker%28v=vs.90%29.aspx) выглядит как текстовое поле с раскрывающимся списком в виде стрелки. Когда пользователь нажимает на стрелку раскрывающегося списка, появляется календарь. При использовании этого элемента управления пользователь может выбрать только одну дату. Элемент управления [*DateTimePicker*](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.datetimepicker%28v=vs.90%29.aspx) также позволяет отображать время вместо дат.

Процесс, используемый для извлечения даты из этих элементов управления, зависит от конкретного используемого элемента. Используйте свойство [*Start*](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.selectionrange.start%28v=vs.90%29.aspx) для элемента управления [*MonthCalendar*](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.monthcalendar%28v=vs.90%29.aspx) и свойство [*Value*](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.datetimepicker.value%28v=vs.90%29.aspx) для элемента управления [*DateTimePicker*](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.datetimepicker%28v=vs.90%29.aspx).

Элемент управления [*MonthCalendar*](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.monthcalendar.aspx) позволяет отображать на экране одновременно до 12 месяцев. По умолчанию в этом элементе управления отображается только один месяц, однако имеется возможность указать количество месяцев, которые будут отображаться на экране, и их размещение в данном элементе управления. Чтобы обеспечить достаточное количество места в форме для новой размерности, при изменении диапазона календаря изменяются размеры элемента управления.

**Константы для указания формата даты**

|  |  |  |
| --- | --- | --- |
| **Константа** | **Описание** | **Пример** |
| DateFormat.GeneralDate | Отображает дату, время или оба значения. Если присутствует дата, она отображается в кратком формате. Если присутствует время, оно отображается в полном формате. Если присутствует и время, и дата, отображаются обе части. | 22/11/1963 12:00:00 PM |
| DateFormat.LongDate | Отображает дату в полном формате, который определяется установленными на компьютере региональными параметрами. | Пятница, 22 ноября, 1963 |
| DateFormat.ShortDate | Отображает дату в кратком формате, который определяется установленными на компьютере региональными параметрами. | 11/22/1963 |
| DateFormat.LongTime | Отображает время в полном формате, который определяется установленными на компьютере региональными параметрами. | 12:00:00 PM |
| DateFormat.ShortTime | Отображает время в 24-часовом формате (чч:мм). | 12:00 |

**Свойства и функции системных часов**

Чтобы получить от системных часов информацию о времени, можно использовать их различные свойства и функции. Информация о времени может потребоваться в программах при создании собственных календарей, часов или оповещений. В следующей таблице содержится перечень наиболее полезных функций системных часов. За дополнительной информацией обращайтесь к справочной системе VisualStudio.

|  |  |
| --- | --- |
| **Свойство или функция** | **Описание** |
| TimeString | Возвращает от системных часов текущее время. |
| DateString | Возвращает от системных часов текущую дату. |
| Now | Возвращает закодированное значение, содержащее текущие дату и время. Наиболее полезно как аргумент для других функций системных часов. |
| Hour (time) | Возвращает количество часов для указанного времени (от 0 до 24). |
| Minute (time) | Возвращает количество минут для указанного времени (от 0 до 59). |
| Second (time) | Возвращает количество секунд для указанного времени (от 0 до 59). |
| Day (date) | Возвращает целое число, представляющее собой день месяца (от 1 до 31). |
| Month (date) | Возвращает целое число, представляющее собой месяц (от 1 до 12). |
| Year (date) | Возвращает год для указанной даты. |
| Weekday (date) | Возвращает целое число, представляющее собой день недели (по американской системе: 1 - это воскресенье, 2 - это понедельник, и т.д.). |

#### Задание на лабораторную работу.

**Порядок выполнения работы**

1. Создать новый проект командой *Создать проект (New Project*) из меню *Файл (File)* (порядок создания нового проекта подробно описан в лабораторной работе № 1).
2. Выберите элемент Приложение WindowsForms и нажмите кнопку ОК.
3. Добавьте в форму элемент *Label*, оставив имя по умолчанию Label1.
4. Удалите текст из свойства *Text* элемента управления Метка.
5. Добавьте в форму элемент управления *MonthCalendar*, оставив имя по умолчанию *MonthCalendar1*.
6. Дважды щелкните элемент управления *MonthCalendar*, чтобы открыть обработчик событий по умолчанию в редакторе кода.
7. В обработчике событий MonthCalendar1\_DateChanged добавьте следующий код для добавления элементов в список.

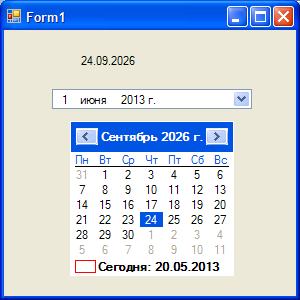


1. Вернитесь в режим конструктора и добавьте в форму элемент управления *DateTimePicker*, оставив имя по умолчанию *DateTimePicker1*.
2. Дважды щелкните элемент управления *DateTimePicker*, чтобы открыть обработчик событий по умолчанию в редакторе кода.
3. В обработчике событий DateTimePicker\_ValueChanged добавьте следующий код для добавления элементов в список.



1. Нажмите клавишу F5 для запуска программы.
2. Когда появится форма, выберите дату в элементе управления *MonthCalendar* и убедитесь, что она отображается в метке.
3. Щелкните стрелку раскрывающегося списка элемента управления *DateTimePicker* и выберите дату.

Дата и время отображаются в метке.

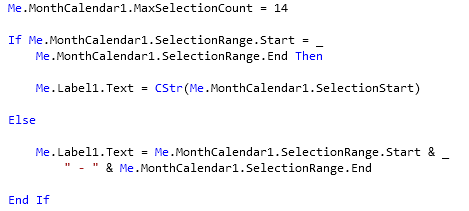


[**Извлечение нескольких дат**](javascript:void(0))

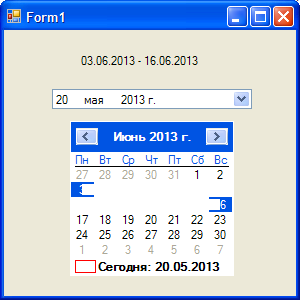
Диапазон дат, выбранных в элементе управления *MonthCalendar*, можно извлечь с помощью свойств [*Start*](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.selectionrange.start%28v=vs.90%29.aspx) и [*End*](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.selectionrange.end%28v=vs.90%29.aspx) свойства [*SelectionRange*](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.monthcalendar.selectionrange%28v=vs.90%29.aspx). По умолчанию максимальное число дней, которые можно выбрать, равно 7, но при необходимости этот параметр можно изменить, установив значение свойства [*MaxSelectionCount*](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.monthcalendar.maxselectioncount%28v=vs.90%29.aspx). Чтобы определить, выбран ли диапазон дат, просто проверьте, совпадают ли даты начала и конца.

**Извлечение диапазона дат из элемента управления календарем месяца**

1. Замените код в обработчике событий MonthCalendar1\_DateChanged следующим. Этот код устанавливает максимальное число дней (две недели), которые могут быть выбраны в элементе управления. Он отображает дату начала в метке, если выбран только один день, и отображает диапазон дат при выборе диапазона дней в элементе управления *MonthCalendar*.



1. Нажмите клавишу F5 для запуска программы.
2. Когда появится форма, выберите диапазон дат в элементе управления *MonthCalendar* и убедитесь, что диапазон дат появился в метке.

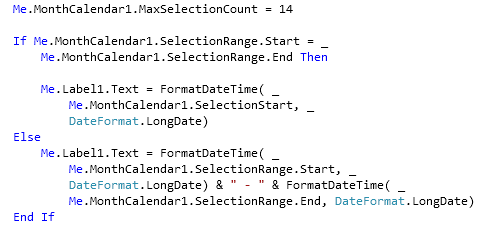


[**Форматирование дат**](javascript:void(0))

Даты, возвращаемые элементами управления [*MonthCalendar*](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.monthcalendar%28v=vs.90%29.aspx) и [*DateTimePicker*](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.datetimepicker%28v=vs.90%29.aspx), можно форматировать с помощью функции [*FormatDateTime*](http://msdn.microsoft.com/ru-ru/library/microsoft.visualbasic.strings.formatdatetime%28v=vs.90%29.aspx). Существует несколько констант, которые можно использовать для указания формата даты (см. теор. часть).

**Форматирование даты в метке**

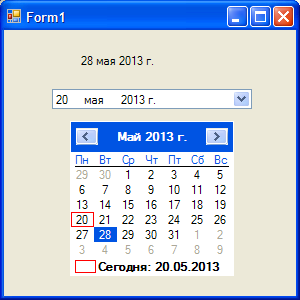
1. Замените код в обработчике событий MonthCalendar1\_DateChanged следующим. Этот код форматирует дату, возвращаемую в полном формате.



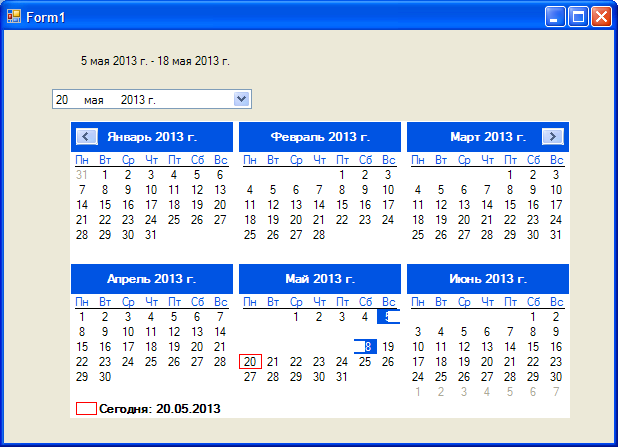
1. Замените код в обработчике событий DatePicker1\_ValueChanged следующим. Этот код форматирует дату, возвращаемую в полном формате.



1. Нажмите клавишу F5 для запуска программы.
2. Когда появится форма, выберите дату или диапазон дат в элементе управления *MonthCalendar*. Убедитесь, что дата или диапазон дат отображается в метке в полном формате.
3. Выберите дату в элементе управления *DateTimePicker* и убедитесь, что дата в метке отображается в полном формате.



**Чтобы отобразить несколько месяцев**



* Задайте для свойства [*CalendarDimensions*](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.monthcalendar.calendardimensions.aspx) значение, равное числу месяцев, отображаемых по горизонтали и вертикали.



#### Программа Birthday

В программе Birthday элементы управления *DateTimePicker* и *Button* используются, чтобы выяснить у пользователя дату его рождения и показать эту информацию в окне сообщения.

1. В области элементов выберите элемент управления *Button*, и ниже объекта выбора даты и времени добавьте объект кнопки. Эта кнопка будет использована для показа дня рождения и для проверки правильности работы объекта выбора даты и времени.
2. В окне *Свойства(Properties)* измените свойство *Text* объекта кнопки на **Показать день моего рождения**.
3. Дважды щелкните мышью на объекте кнопки, а потом наберите следующий фрагмент программы между операторами Private Sub и End Sub в процедуре события Button1\_Click:



Этот фрагмент программы показывает три последовательных окна сообщения (небольшие диалоговые окна), которые содержат информацию из объекта календаря. В первой строке используется свойство *Text* календаря для вывода информации о дате рождения, которую пользователь выберет в этом объекте после запуска программы. Функция MsgBox кроме текстового значения из свойства *Text* календаря показывает строку "Ваш день рождения". Эти два текстовых элемента объединяются в строку с помощью оператора конкатенации (слияния) строк &.

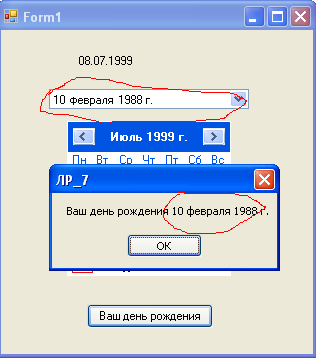
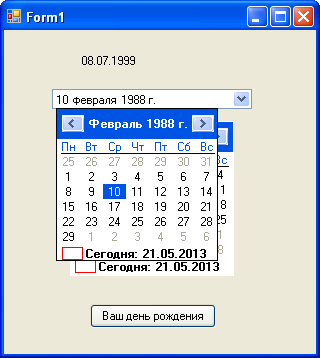
Во второй строке объект календаря используется для вычисления дня года, отсчитывая с 1 января. Это делается с помощью свойства DayOfYear и метода ToString, который переводит числовой результат вычисления даты в текстовое значение, которое гораздо проще показать с помощью функции MsgBox.



В третьей строке фрагмента, после перевода значения в строковое (или текстовое) представление, в окне сообщения показывается информация о точном времени.

#### Запуск программы Birthday

1. На стандартной панели инструментов нажмите кнопку Start (Начать). Программа Birthday запустится в среде разработки. В окне объекта выбора даты и времени появится текущая дата.
2. Нажмите стрелку раскрывающегося списка, чтобы вывести на экран представление этого объекта в виде календаря. Форма будет выглядеть как на следующей иллюстрации.
3. Выберите в элементе DatetimePicker1число, месяц и год Вашего рождения, пользуясь стрелками прокрутки. 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.



1. В окне сообщения нажмите OK. Появится второе окно сообщения, указывающее, в какой день года вы родились.
2. Нажмите OK, чтобы показать последнее окно сообщения. Появятся текущие дата и время. Вы обнаружите, что 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.

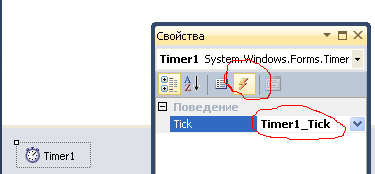
**Совет.** Чтобы настроить объект выбора даты и времени для показа времени, а не даты, установите свойство *Format* этого объекта равным *Time*.

**Работа с датами в VB.Net. (Дополнительно)**

Реализовать программу которая будет узнавать текущую дату и время.

Чтобы узнать текущее время в VB.Net есть функция - **TimeString**, чтобы узнать дату - **DateString**. Кроме того есть функции работы со временем: **Hour()**(часы), **Minute()**(минуты), **Second()**(секунды). Эти функции вырезают часы, минуты, секунды из указанно времени. Например, Minute(TimeString) - вырезает минуты из текущего времени. И функции для работы с датами: **Day()**(год), **Month()**(месяц), **Year**(год). Эти функции вырезают из текущей даты: день, месяц, год. Например, Month(DateString), вырезает из текущей даты месяц.

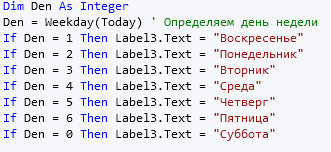
1. Создайте новый проект командой *Создать проект* из меню *Файл (File)*. Выберите элемент Приложение Windows Forms и нажмите кнопку ОК.
2. Добавьте в форму три элемента *Label*.
3. Чтобы размеры метки автоматически регулировались, в зависимости от текста в метке Задайте *AutoSize = True*.
4. Удалите текст из свойства *Text* элемента управления Метка:*Text* = "".
5. Добавьте в форму невизуальный элемент управления *Timer*(*Enabled* = True, *Interval* = 1000).
6. Напишите обработчик события Tick элемента Timer. Для этого выделите Элемент Таймер и в окне Свойства перейдите в раздел События, нажав на кнопку .



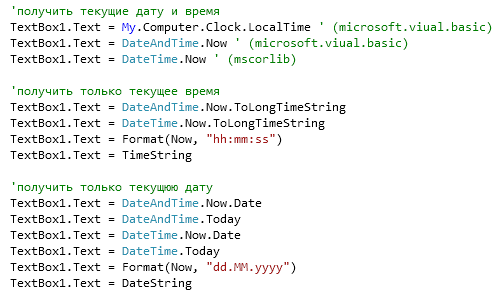
1. Дважды щелкните в строке и введите в открывшемся окне код:



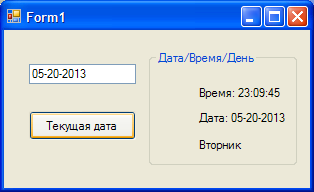
1. Для определения дня недели нужна функция **WeekDay**. Напишите обработчик события загрузки формы:



1. В области элементов выберите элемент управления *Button* и добавьте его, задайте свойство *Text* =Текущая дата/Время
2. Напишите обработчик события нажатия кнопки для вывода текущей даты или времени:



1. Запустите проект.



**Контрольные вопросы:**

1. Назначение элемента управления DateTimePicker
2. Назначение элемента управления [MonthCalendar](http://msdn.microsoft.com/ru-ru/library/system.windows.forms.monthcalendar%28v=vs.90%29.aspx)
3. Назовите свойства, используемые для извлечения даты из этих элементов.
4. Свойства элемента управления MonthCalendar для извлечения диапазона дат.
5. Назначение оператора конкатенации &, пример его использования
6. Отображение нескольких месяцев элемента управления MonthCalendar.
7. Функция форматирования дат.
8. Какие существуют константы для указания формата даты, опишите их.
9. Функции для работы с датой и временем в VB.Net.
10. Объясните работу фрагмента кода



**Лабораторная работа №8**

**Создание проекта с использованием компонентов стандартных диалогов и системы меню**

**Цель:** научиться использовать компоненты стандартных диалогов, системы меню и отображать их в форме Windows Forms.

## Ход работы:

## Изучить теоретическую часть.

## Выполнить задания, следуя указаниям.

## Ответить на контрольные вопросы (в устной форме).

## Предъявить преподавателю результаты работы программы и исходные коды.

## Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, программный код, результаты тестирования программы, выводы).

**🕮Теоретическая часть**

**Добавление меню с помощью элемента управления MainMenu**

Элемент управления *MainMenu* - это инструмент, с помощью которого в программу можно добавить меню и настроить их в окне Свойства. С помощью *MainMenu* можно добавлять новые меню, изменять и удалять существующие. В меню можно добавить специальные возможности: клавиши ускоренного доступа, отметки "включено/выключено" и сочетания клавиш. После того, как меню добавлено в форму, для обработки команд меню использовать процедуры обработки событий.

**Соглашения о пунктах меню**

По соглашению, в приложениях для Microsoft Windows каждое название и команды меню начинаются с заглавной буквы. Часто первыми двумя командами меню в строке меню являются Файл и Правка, а последним является Справка. Часто встречаются названия команд Вид, Формат и Окно. Не имеет значения, какие меню и команды вы используете в приложении - главное, чтобы они были понятны и соответствовали своим названиям. При создании элементов меню придерживайтесь следующих рекомендаций.

* Используйте короткие и ясные названия, состоящие из одного или максимум двух слов.
* Каждому элементу меню назначайте клавишу доступа. Если возможно, используйте для этого их первую букву.
* У элементов меню, расположенных на одном уровне видимости, клавиши доступа не должны совпадать.
* Если команда используется как переключатель "включено/выключено", то когда она активна, рядом с ней должна быть видна галочка. Чтобы добавить такую пометку, в окне Свойства для этой команды свойство Checked должно иметь значение True.
* Если команда при нажатии выполняется не сразу, а от пользователя потребуется дополнительные действия, поставьте после такой команды многоточие ( ). Оно указывает, что если пользователь выберет эту строку, то откроется диалоговое окно.



**Добавление клавиш доступа к командам меню**

В большинстве приложений команды меню можно вызывать с помощью клавиатуры. Например, чтобы в VisualStudio открыть меню Файл, нужно нажать клавишу Alt а затем Ф. Когда меню Файл откроется, то, чтобы выполнить команду Печать, достаточно нажать на клавиатуре П. Клавиши, которые вы нажимаете вместе с Alt или в открытом меню, называются **клавишами доступа** (или **клавишами быстрого доступа** ). Клавишу доступа можно определить по подчеркнутому символу.

Чтобы добавить в главное меню клавишу доступа, активизируйте Конструктор меню и введите перед требуемой буквой в имени меню символ "амперсанд" (&). Во время выполнения программы соответствующая клавиша на клавиатуре будет работать как клавиша быстрого доступ к меню.

С помощью элемента управления *MainMenu* можно назначать создаваемым меню сочетания клавиш. Сочетания клавиш - это комбинация клавиш на клавиатуре, нажав которую пользователь может вызвать команду меню не открывая его. Например, в обычном меню Правка в приложении Windows можно выделить текст и скопировать его в буфер обмена, нажав клавиши (Ctrl)+(C). Эти сочетания настраиваются в свойстве Shortcut элемента управления *MainMenu*.

**Использование элементов управления для диалоговых окон**

В Visual Studio на закладке WindowsForms окна области элементов имеется семь стандартных элементов управления для диалоговых окон. Во многих случаях нужно написать код, который подключает эти диалоговые окна к программе, но пользовательский интерфейс уже сделан, и он соответствует стандартам для общих задач в приложениях Windows. Все семь имеющихся элементов управления для стандартных диалоговых окон перечислены в следующей таблице.

**Процедуры обработки событий, которые управляют общими диалоговыми окнами**

Чтобы показать в программе диалоговое окно, в процедуре обработки события для соответствующей команды меню нужно ввести оператор, состоящий из имени диалогового окна и метода ShowDialog. Если это необходимо, то перед открытием диалогового окна нужно запрограммировать свойства этого окна. Наконец, в тексте программы должен присутствовать код, который после закрытия окна выполнит те или иные операции в зависимости от выбора или действий, которые пользователь выполнит в этом диалоговом окне.

|  |  |
| --- | --- |
| **Название элемента управления** | **Назначение** |
| OpenFileDialog | Получает названия диска, папки и файла для существующего файла. |
| SaveFileDialog | Получает названия диска, папки и файла для нового файла. |
| FontDialog | Позволяет выбрать новый шрифт и его стиль. |
| ColorDialog | Позволяет выбрать цвет из палитры. |
| PrintDialog | Позволяет задать параметры печати. |
| PrintPreviewDialog | Отображает диалоговое окно предварительного просмотра материала для печати(какв MSWord). |
| PageSetupDialog | Позволяет управлять параметрами страницы: полями, размером бумаги и ее ориентацией. |

**Управление выбором цвета с помощью установки свойств диалогового окна выбора цвета**

Диалоговое окно выбора цвета тоже можно настроить. Например, можно управлять тем набором цветов, которые будут предоставлены на выбор пользователя при открытии окна. Эти параметры можно настроить в окне Свойства в среде разработки или задать их в тексте программы перед открытием этого диалогового окна, используя метод ShowDialog. Следующая таблица содержит список наиболее часто используемых свойств элемента управления ColorDialog. Чтобы задействовать ту или иную настройку, соответствующему свойству должно быть присвоено значение True, чтобы отменить настройку - значение False.

|  |  |
| --- | --- |
| **Свойство** | **Значение** |
| AllowFullOpen | Если присвоено значение True, в диалоговом окне будет работать кнопка DefineCustomColors (Определить цвет). |
| AnyColor | Если присвоено значение True, пользователь сможет выбрать любой цвет из показанных в диалоговом окне. |
| FullOpen | Если присвоено значение True, при открытии диалогового окна будет видна область CustomColors (Дополнительные цвета). |
| ShowHelp | Присвойте True, если нужно включить в диалоговом окне кнопку Help (Справка). |
| SolidColorOnly | Если присвоено значение True, пользователь сможет выбрать только "чистые" цвета (цвета со смешением будут отключены). |

#### Задание на лабораторную работу.

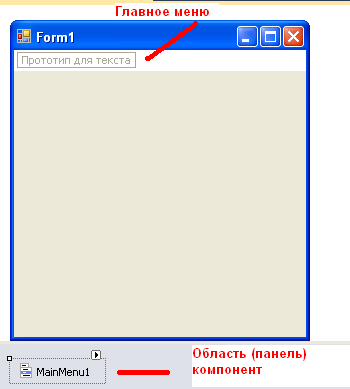
С помощью элемента управления *MainMenu* мы создадим меню Clock, команды которого показывает текущую дату и время.

**Порядок выполнения работы**

**Создание меню**

1. Создайте новый проект командой *Создать проект (New Project)* из меню *Файл (File)* (порядок создания нового проекта подробно описан в лабораторной работе № 1).
2. Выберите элемент *Приложение Windows Forms* и нажмите кнопку ОК.
3. Выберите элемент управления *MainMenu* на закладке WindowsForms окна области элементов и нарисуйте на поле формы элемент управления.

Форма будет выглядеть примерно так.



В Visual Studio .NET невидимые объекты, в том числе меню и таймеры, показываются в среде разработки на отдельной панели, которая называется **областью компонентов**. На этой панели их можно выделять, настраивать их свойства или удалять.

В дополнение к объекту меню, расположенному в области компонентов, создаваемое в Visual Studio.NET меню графически изображается в верхней части формы. Чтобы ввести название меню, нужно щелкнуть по полю Прототип для текста. После этого можно добавлять заголовки подменю и других меню, выбирая нужные поля с помощью стрелок и вписывая туда требуемые названия. Позже вы всегда сможете вернуться к этому встроенному Конструктору меню и отредактировать то, что уже сделано, или добавить новые пункты. Объект главного меню очень хорошо настраивается и позволяет создать решения с использованием меню, не уступающие лучшим Windows-приложениям.

1. Щелкните на Прототип для текста, введите **Дата**, а затем нажмите клавишу (Enter). Слово "Дата" стало названием первого меню, и появились два новых поля Прототип для текста, позволяющие создать элементы подменю или команды в меню Дата и дополнительные меню. В данный момент выберите подменю.

**Совет**. Если меню с формы исчезает, выберите объект *MainMenu1* в области компонентов, и оно снова появится.

1. Чтобы в меню **Дата** создать новую команду, введите слово **Число**, а затем нажмите клавишу (Enter). VisualStudio добавит команду к меню и выделит следующий элемент подменю.
2. Наберите слово **Время**, а затем нажмите клавишу (Enter). Теперь у вас есть меню **Дата** с двумя командами меню - **Число** и **Время**. Форма будет выглядеть примерно так.



1. Чтобы закрыть Конструктор меню, щелкните на поле формы. Конструктор меню закроется, и в среде разработки будет полностью видна форма, а меню исчезнет. Чтобы увидеть меню и начать с ним работать, нужно щелкнуть на его названии.
2. В поле формы нажмите слово Дата. Меню появляется снова, с уже знакомыми полями Прототип для текста. Теперь его опять можно настраивать.

**Добавление клавиш доступа**

1. На форме выберите меню **Дата**, а затем щелкните на нем еще раз. В названии появится курсор редактирования текста. Появление текстового курсора означает, что название меню можно изменить или добавить символ **&**, чтобы обозначить клавишу доступа.
2. Чтобы убедиться, что курсор находится перед первой буквой, нажмите на клавишу со стрелкой влево. Курсор будет мигать перед буквой "Д" в слове Дата.
3. Введите **&** (амперсанд), чтобы указать, что буква "Д" является клавишей доступа для меню Дата.
4. В списке меню выберите команду **Число**, а затем щелкните на ней еще раз, чтобы появился курсор редактирования текста.
5. Введите **&** (амперсанд) перед буквой "Ч". Теперь буква "ч" определена как клавиша доступа для команды **Число**.
6. В списке меню выберите команду **Время**, а затем щелкните на этой команде второй раз, чтобы появился курсор.
7. Введите **&** (амперсанд) перед буквой "В". Теперь буква "В" определена как клавиша доступа для команды Время.
8. Нажмите клавишу (Enter). Нажатие на (Enter) фиксирует изменения в тексте. Ваша форма будет выглядеть примерно так.



Теперь с помощью Конструктора меню давайте изменим порядок команд **День** и **Время**. Изменение порядка элементов меню - это важный навык, так как это требуется довольно часто.

**Изменение порядка элементов меню**

1. Выберите в форме меню **Дата**, чтобы развернуть элементы этого меню. Изменить порядок элементов очень просто - нужно просто перетащить элемент на новое место в меню.
2. Перетащите надпись **Время** на надпись **Число** и отпустите клавишу мыши. Перетаскивание элемента меню поверх другого элемента означает, что вы хотите разместить его перед вторым элементом. VisualStudio немедленно перемещает элемент меню **Время** на новое место перед элементом **Число**.

Мы закончили создавать пользовательский интерфейс для меню Дата. Теперь нужно запрограммировать процедуры, соответствующие строкам меню, для обработки выбора пользователя.

**Совет**. Чтобы удалить из меню ненужный элемент, щелкните на этом элементе, а затем нажмите клавишу (Delete).

**Обработка выбора меню**

После того, как меню и команды настроены с помощью объекта *MainMenu*, они также становятся объектами программы. Чтобы заставить объекты меню выполнять осмысленную работу, необходимо написать для них процедуры обработки событий. Процедуры обработки событий меню обычно содержат операторы, которые показывают информацию в пользовательском интерфейсе, обрабатывают ввод или изменяют одно или несколько свойств меню. Если для обработки команды требуется получить дополнительную информацию, то процедура обработки события обычно открывает диалоговое окно. Для этого используется один из элементов управления диалоговых окон WindowsForms или один из элементов управления для ввода.

**Добавление на форму объекта надпись**

1. В области элементов выберите элемент управления *Label*.
2. Нарисуйте в центре формы надпись среднего размера. На форме появится объект *Label*, и в коде программы у него будет имя *Label1*.
3. Задайте для этой надписи следующие свойства:

|  |  |  |
| --- | --- | --- |
| **Объект** | **Свойство** | **Значение** |
| Label1 | BorderStyle | FixedSingle |
| Font | Microsoft Sans Serif, жирный , 14 пунктов |
| Text | (empty) |
| TextAlign | MiddleCenter |

**Редактирование процедур событий меню**

1. Щелкните на меню **Дата**, чтобы его раскрыть.
2. Чтобы открыть в редакторе кода процедуру обработки событий для команды **Время**, дважды щелкните мышью на этой команде. В редакторе кода появится процедура события MenuItem3\_Click. Имя MenuItem3\_Click означает, что пункт **Время** был третьим из созданных в этом проекте (вслед за **Дата** и **Число**), а слово \_Click напоминает, что это процедура события, которая запускается при щелчке на этом элементе меню.
3. Добавьте в программу следующий оператор



Этот оператор присваивает текущее время (по системным часам) свойству *Text* объекта *Label1*, которое, собственно, и показывается в виде надписи.

1. Нажмите на клавишу со стрелкой вниз. VisualBasic интерпретирует строку и, если потребуется, изменит заглавные буквы и добавит или удалит пробелы. VisualBasic проверяет каждую строку в процессе ее ввода и ищет в них синтаксические ошибки. Набор строки можно закончить, нажав клавишу (Enter), стрелку вверх или стрелку вниз.
2. В Обозревателе решений нажмите кнопку*Просмотреть конструктор*, а затем дважды щелкните мышью на команде **Число** в меню **Дата**. В редакторе кода откроется процедура обработки событий MenuItem2\_Click. Эта процедура исполняется тогда, когда пользователь щелкает в меню **Дата** на команде **Число**.
3. Добавьте в программу следующий оператор



Этот оператор присваивает сегодняшнее число (по системным часам) свойству *Text* объекта *Label1*, которое показывается в виде надписи. Предыдущий текст в объекте *Label1*, если он имелся, будет заменен.

1. Чтобы закончить ввод строки, нажмите клавишу со стрелкой вниз.

**Запуск программы Menu**

1. На стандартной панели инструментов нажмите кнопку Start (Начать). Программа Menu запустится в среде разработки.
2. В строке меню выберите пункт **Дата**. Появится меню **Дата**.
3. Выберите команду **Время**. В поле надписи появится текущее системное время, как показано ниже.



Теперь посмотрим, какое сегодня число.

1. Нажмите и отпустите клавишу (Alt). В строке меню выделится меню **Дата**.
2. Чтобы раскрыть меню **Дата**, нажмите **Д**. Меню появится на экране.
3. Чтобы показать сегодняшнее число, нажмите **Ч**. В поле надписи появится дата.

**Добавление элементов управления OpenFileDialog и ColorDialog**

1. Добавьте в область компонентов, в которой уже находится объект главного меню *MainMenu1*, два элемента управления диалоговых окон. Элемент управления *OpenFileDialog* потребуется, чтобы открывать файлы с точечными изображениями, а элемент управления *ColorDialog* позволит изменять цвет для показа даты. В процессе разработки элементы управления диалоговых окон помещаются в области компонентов, а не на поле формы, так как во время выполнения они на форме не появляются.
2. В области элементов на закладке *WindowsForms* выберите элемент управления *OpenFileDialog*, а затем щелкните мышкой в области компонентов, где уже есть объект *MainMenu1*.

**Совет**. Если вы не видите *OpenFileDialog* в области элементов, то он может быть за пределами видимости. Чтобы промотать список в области элементов, щелкните на нижней стрелке прокрутки, находящейся рядом с закладкой *ClipboardRing (Буфер обмена)*.

В области компонентов появится объект диалогового окна для открытия файла.

1. В области элементов на закладке *WindowsForms* выберите элемент управления *ColorDialog*, а затем щелкните на области компонентов, расположенной ниже поля формы. Теперь область компонентов выглядит так.



Как и объект главного меню, объекты с диалогами открытия файла и выбора цвета можно настроить, задав их свойства.

Теперь с помощью элемента управления *PictureBox* создайте область для показа изображений. Этот объект показывает в поле формы содержимое из файлов изображений. На этот раз мы выведем на поле формы картинку, используя диалоговое окно для открытия файла.

**Добавление объекта области показа изображения**

1. В области элементов выберите элемент управления *PictureBox*.
2. В поле формы ниже надписи нарисуйте объект области показа изображений, и в окне Свойства для свойства *SizeMode* этого объекта выберите значение *StretchImage*.

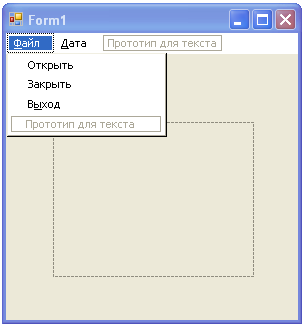
Теперь с помощью *Конструктора меню* добавьте в вашу программу меню Файл.

**Добавление меню Файл**

1. В поле формы щелкните на меню **Дата**, затем на ячейке Прототип для текста, расположенной справа от этого меню. Теперь нужно добавить в программу меню **Файл**, в котором будут команды **Открыть**, **Закрыть** и **Выход**.
2. Чтобы создать меню **Файл** с буквой **"Ф"** в качестве клавиши доступа, введите **&Файл**.
3. Нажмите клавишу со стрелкой вниз, а затем, чтобы создать команду Открыть с буквой **"O"** в качестве клавиши доступа, введите **&Открыть** . Команда Открыть будет использоваться для загрузки точечных изображений. Так как эта команда должна будет открывать диалоговое окно, добавьте к ее имени многоточие.



1. Нажмите клавишу со стрелкой вниз, а затем, чтобы создать команду **Закрыть** с буквой **"З"** в качестве клавиши доступа, введите **&Закрыть**. Команда **Закрыть** будет использоваться в программе для закрытия файла с изображением.
2. Нажмите клавишу со стрелкой вниз, а затем, чтобы создать команду **Выход** с буквой **"ы"** в качестве клавиши доступа, введите **В&ыход**. Команда **Выход** будет использоваться для закрытия программы. Обратите внимание, что в этом случае в качестве клавиши доступа для команды Выход была использована третья буква, как это сделано во многих приложениях для Windows.
3. Чтобы передвинуть меню **Файл** на первое место, просто перетащите его на меню **Дата**. В Конструкторе меню целые меню можно перемещать точно так же, как и отдельные команды внутри меню. Имеет смысл сделать меню **Файл** первым меню программы. Ваша форма должна выглядеть примерно так.



В следующем упражнении мы отключим команду **Закрыть** в меню **Файл**. (Команда **Закрыть** может использоваться только после того, как файл уже был открыт в программе). Далее в этой лекции мы добавим в процедуру обработки события команды **Открыть** оператор, который в нужный момент включает команду **Закрыть**.

**Отключение команды Закрыть**

1. В меню **Файл** программы *Menu* выберите команду **Закрыть**.
2. В окне Свойства для свойства *Enabled* объекта *MenuItem7* выберите значение *False*.

Теперь, чтобы продемонстрировать, как работает диалоговое окно выбора цвета, необходимо добавить в меню **Дата** команду **Цвет текста**. Диалоговое окно выбора цвета c помощью свойства *Color* возвращает в программу вновь выбранный цвет. Это свойство будет использоваться для изменения цвета текста в объекте *Label1*.

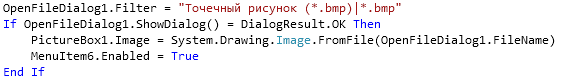
**Добавление команды Цвет текста в меню Дата**

1. Выберите меню **Дата**, а затем щелкните на нижней ячейке Прототип для текста.
2. Чтобы добавить в это меню команду **Цвет текста** с клавишей "Ц" в качестве клавиши доступа, введите **&Цвет текста** . Команда будет добавлена в меню **Дата**. Эта команда заканчивается многоточием, которое указывает, что при ее выборе откроется диалоговое окно.



**Редактирование процедуры обработки событий команды Открыть**

1. В меню **Файл** нашей формы дважды щелкните мышью на команде **Открыть**. В редакторе кода появится процедура обработки событий mnuOpenItem\_Click.
2. Между операторами Private Sub и End Sub добавьте следующие ниже операторы. Убедитесь, что вы набрали каждую строку в точности так, как они здесь напечатаны, а после набора последней строки нажмите на клавиатуре стрелку вниз.



Первые три оператора в этой процедуре обработки события ссылаются на три свойства объекта диалога открытия файла. В первом операторе свойство Filter используется для определения списка допустимых файлов. В нашем случае этот список содержит только один элемент: \*.bmp. Это достаточно важно для диалогового окна Открыть, так как объект показа изображения поддерживает шесть типов файлов: точечные изображения (файлы .bmp), метафайлы Windows (файлы .emf и .wmf), значки (файлы .ico), формат JointPhotographicExpertsGroup (файлы .jpg и .jpeg), формат PortableNetworkGraphics (файлы .png) и формат GraphicsInterchangeFormat (файлы .gif). Попытка показать в объекте изображения файл .txt приведет к возникновению ошибки в момент выполнения. Чтобы добавить в список Filter дополнительные элементы, между ними нужно ввести символ (|). Например, при следующем значении фильтра

*OpenFileDialog1.Filter = "Точечный рисунок (\*.bmp)|\*.bmp|Метафайл Windows (\*.wmf)|\*.wmf"*

в диалоговом окне **Открыть** будут показаны как файлы с точечными изображениями, так и метафайлы Windows.

Второй оператор в процедуре обработки события показывает в программе диалоговое окно **Открыть**. Метод *ShowDialog* - это новый метод Visual Basic .NET, он похож на метод Show из VisualBasic 6, но может использоваться для любой формы *WindowsForms*. Метод *ShowDialog* возвращает результат с именем *DialogResult*, который указывает, какую кнопку диалогового окна нажал пользователь. Чтобы определить, щелкнул ли пользователь на кнопке **Открыть**, используется оператор If...Then, который проверяет, равно ли возвращенное значение *DialogResult.OK*. Если оно равно этому значению, то в свойстве *FileName* диалога открытия должен храниться путь к существующему файлу на диске.

В третьем операторе используется имя файла, которое было выбрано в диалоговом окне. Когда пользователь выбирает диск, папку и имя файла, а затем нажимает кнопку **Открыть**, полный путь передается в программу через свойство *OpenFileDialog1.FileName*. Затем используется метод *System.Drawing.Image.FromFile*, который копирует указанное точечное изображение в объект показа изображений. (Этот оператор разбит на две строки при помощи символа продолжения строки, так как он получился очень длинным.)

В четвертом операторе активируется команда **Закрыть** из меню **Файл**. Теперь, когда файл в программе был открыт, команда **Закрыть** должна быть доступна, чтобы пользователи могли закрыть этот файл.

Теперь введите код программы для процедуры обработки события *mnuCloseItem\_Click*, которая выполняется при выборе команды **Закрыть** из меню **Файл**.

**Редактирование процедуры события команды Закрыть**

1. Снова откройте форму, а затем дважды щелкните мышью на команде **Закрыть** из меню **Файл**. В редакторе кода появится процедура обработки события для команды **Закрыть**.
2. Между операторами PrivateSub и EndSub добавьте следующие операторы программы:



Первый оператор закрывает открытое точечное изображение, удаляя информацию, хранящуюся в свойстве *Image*. Ключевое слово *Nothing* используется здесь для того, чтобы отменить связь между текущим объектом точечного изображения и свойством *Image*, другими словами, *Nothing* задает для этого свойства нулевое значение, и изображение исчезает. (Далее в этой книге *Nothing* будет использоваться для сброса значений и других объектов и свойств.) Во втором операторе команду **Закрыть** из меню **Файл** отключается, так как открытых файлов больше нет. Использование этого оператора в программе равнозначно настройке свойства *Enabled* в окне *Properties (Свойства)*.

**Редактирование процедуры событий команды Выход**

1. Снова перейдите в конструктор формы, а затем дважды щелкните мышью на команде **Выход** из меню **Файл**. В редакторе кода появится процедура обработки событий для команды **Выход**.
2. Между операторами Private Sub и End Sub добавьте следующий оператор программы



Оператор End останавливает программу, когда пользователь заканчивает с ней работать.

**Редактирование процедуры обработки событий команды Цвет текста**

1. Перейдите в конструктор формы, а затем дважды щелкните мышью на новой команде **Цвет текста** из меню **Дата**. В редакторе кода появится процедура обработки событий для команды **Цвет текста**.
2. Добавьте в нее следующие операторы



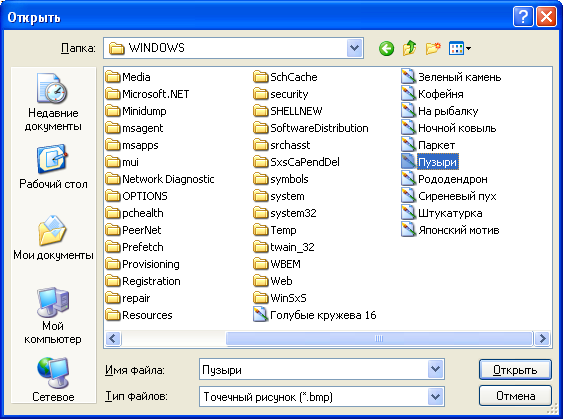
**Совет**. Диалоговое окно Color (Цвет) может использоваться для установки цвета любого элемента пользовательского интерфейса, который работает с цветом. Например, это может быть фоновый цвет формы, цвета геометрических фигур на форме, основной или фоновые цвет для различных объектов.

В первом операторе используется метод *ShowDialog*, который открывает диалоговое окно выбора цвета. Метод *ShowDialog* используется для открытия любой формы как диалогового окна, в том числе и формы, созданные стандартными диалоговыми окнами, которые предоставляет Visual Studio. Во втором операторе цвет, выбранный пользователем в диалоговом окне, присваивается свойству *ForeColor* объекта *Label1*. *Label1* - это поле надписи, которое используется в форме для показа текущих даты и времени. При работе программы вы будете выбирать в диалоговом окне цвет, и он будет использован для текста в этой надписи.

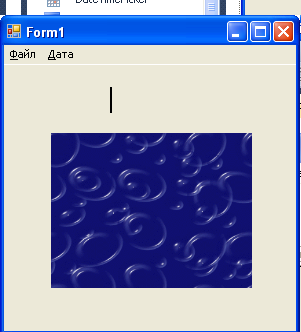
Теперь давайте запустим программу Menu и поэкспериментируем с созданными нами меню и диалоговыми окнами.

**Запуск программы Menu**

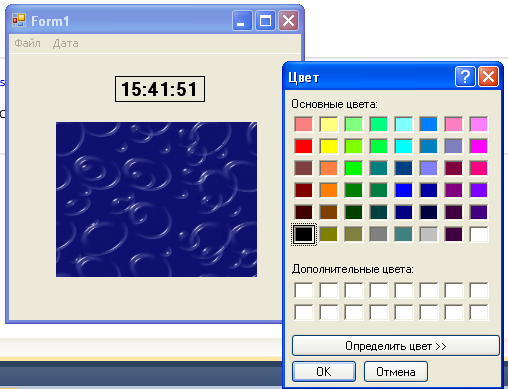
1. На стандартной панели инструментов нажмите кнопку Start (Начать). Программа запустится, и в строке меню появятся меню **Файл** и **Дата**.
2. В меню **Файл** запущенной программы выберите **Открыть**. Появится диалоговое окно **Открыть**. Обратите внимание на текст Точечный рисунок (\*.bmp) в поле Тип файлов. Выберите подходящий файл с изображением.



1. Выберите один из файлов \*.bmp, а затем нажмите кнопку **Открыть**. Изображение из этого файла появится в поле показа изображений. Форма будет выглядеть примерно так.



1. В меню **Дата** выберите строку **Время**. В поле надписи появится текущее время.
2. В меню **Дата** выберите команду **Цвет текста**. Появится диалоговое окно Color (Цвет), показанное ниже.



Диалоговое окно Color (Цвет) содержит элементы, которые позволяют изменить цвет надписи в программе. По умолчанию в этом окне выбран текущий цвет - черный.

1. Щелкните на синем поле, а затем на кнопке **OK**.

Диалоговое окно Color (Цвет) закроется, а цвет текста в надписи изменится на синий.



1. В меню **Дата** щелкните на команде **День**.
2. Откройте меню **Файл**.

Обратите внимание, что команда **Закрыть** включена. (Мы включили ее с помощью свойства *Enabled* = True.)

1. Нажмите **З** (клавишу доступа для **Закрыть**, при необходимости переключите клавиатуру в русский регистр), чтобы закрыть изображение.

Файл закроется, и точечное изображение Windows исчезнет (это сработало ключевое слово *Nothing*.)

1. Откройте меню **Файл**. Теперь команда **Закрыть**отключена, так как в области вывода изображений картинки нет.
2. Выберите команду **Выход**. Программа закроется, и появится среда разработки Visual Studio.

**Следующий шаг: привязка сочетаний клавиш к пунктам меню**

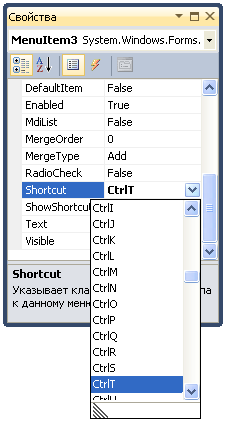
**Определение сочетания клавиш для меню Дата**

1. В меню **Дата** выберите команду **Время**.

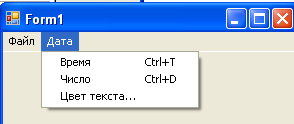
Сочетание клавиш определяется в свойстве *Shortcut* выбранной команды, в окне Свойства.

1. Откройте окно Свойства, выберите свойство *Shortcut*, затем нажмите стрелку раскрывающегося списка в столбце справа, прокрутите этот список и выберите CtrlT. Окно Свойства будет выглядеть так, как показано на рисунке справа.

**Совет**. Обычно Visual Basic при выполнении программы показывает в меню сочетания клавиш, чтобы подсказать пользователям, какие клавиши следует нажимать. Чтобы скрыть от пользователя эти комбинации клавиш (например, если для них не хватает места), задайте для свойства ShowShortcut значение False. Сочетания клавиш по-прежнему будут работать, но подсказок в меню пользователи не увидят.



1. Выберите команду **Число**, а затем измените ее свойство *Shortcut* на значение CtrlD. Теперь давайте запустим программу и попробуем использовать сочетания клавиш.
2. На стандартной панели инструментов нажмите кнопку Начать.
3. Нажмите (Ctrl)+(T), чтобы выполнить команду **Время**. В программе появится текущее время.
4. Нажмите (Ctrl)+(D), чтобы выполнить команду **Число**. В программе появится текущая дата.



1. Щелкните на меню **Дата**. Рядом с командами **Время** и **Число** будут показаны сочетания клавиш. Visual Basic дописывает эти комбинации клавиш, когда они определены с помощью свойства *Shortcut*.

**Контрольные вопросы:**

1. Назовите элемент управления для добавления меню.
2. Рекомендации по созданию меню.
3. Добавление клавиш доступа. Как визуально отличаются пункты меню, для которых назначена клавиша доступа.
4. Назначение сочетаний клавиш для пунктов меню, свойство.
5. Перечислить элементы управления для создания диалоговых окон.
6. Метод, используемый для отображения диалогового окна.

Лабораторная работа №9

**Разработка оконного приложения с несколькими формами**

**Цель:** научиться создавать оконное приложение с несколькими формами.

## Ход работы:

## Изучить теоретическую часть.

## Выполнить задания, следуя указаниям.

## Ответить на контрольные вопросы (в устной форме).

## Предъявить преподавателю результаты работы программы и исходные коды.

## Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, программный код, результаты тестирования программы, выводы).

**🕮Теоретическая часть**

**Добавление в программу новых форм**

Во многих случаях для взаимодействия с пользователем одной формы вполне достаточно. Но если вам требуется обмениваться с пользователем большим объемом информации, Visual Basic позволяет добавить в программу дополнительные формы. Каждая новая формы рассматривается как объект, который наследует свои свойства от класса System.Windows.Forms.Form. Первая форма программы называется Form1.vb. Последующие формы называются Form2.vb, Form3.vb и т.д. Следующая таблица содержит список способов практического применения дополнительных форм в вашей программе.

Каждая новая форма имеет уникальное имя и свой собственный набор объектов, свойств, методов и процедур событий.

|  |  |
| --- | --- |
| **Форма** | **Описание** |
| "Титульная" форма | Форма, которая при запуске программы отображает сообщение приветствия, графику или информацию об авторских правах. |
| Инструкции программы | Форма, которая отображает информацию и подсказки о том, как работать с программой. |
| Диалоговые окна | Дополнительные диалоговые окна, которые принимают данные и отображают результаты работы программы. |
| Содержимое документов | Форма, которая отображает содержимое одного или более файлов и изображений, используемых в программе. |

#### Как используются формы

VisualBasic позволяет пользоваться формами достаточно гибко. Вы можете сделать все формы программы видимыми одновременно, а можете загружать и выгружать формы по мере их необходимости. Если вы выводите сразу несколько форм, то можете разрешить пользователю переключаться между ними, а можете управлять порядком, в котором эти формы используются. Форма, которой при ее отображении на экране передается фокус ввода, называется **диалоговым окном.** Диалоговые окна (называемые в VisualBasic 6 **модальными формами** ) сохраняют фокус ввода до тех пор, пока пользователь не нажмет на **OK**, на Cancel ( **Отмена** ) или не закроет ее другим способом. Чтобы в VisualBasic .NET отобразить существующую форму как диалоговое окно, откройте ее с помощью метода ShowDialog.

Если вы хотите отобразить форму, на которую пользователь сможет переключиться, используйте вместо метода ShowDialog метод Show. В Visual Basic 6 формы, которые могли терять фокус ввода, назывались немодальными формами, и вы по-прежнему можете услышать такое их название. Большинство приложений для Microsoft Windows использует для отображения информации обычные немодальные формы, так как они дают пользователю большую гибкость, и при создании новой формы в Microsoft Visual Studio этот стиль является стилем по умолчанию. Так как формы - это просто члены класса System.Windows.Forms.Form, вы также можете создать и отобразить формы с помощью кода программы.

Чтобы узнать, как в приложении на Visual Basic определяется форма по умолчанию, изучите код в разделе, сгенерированном Windows Form Designer, расположенном в верхней части каждой новой формы.

#### Использование свойства DialogResult в вызывающей форме

Вы можете эффективно использовать в программе на VisualBasic свойство DialogResult, которое присваивается в диалоговом окне. Более сложное диалоговое окно может предлагать пользователю дополнительные кнопки – Cancel (**Отмена** ), Yes (Да), No (Нет), Abort (Прервать) и так далее. Каждая кнопка диалогового окна может быть ассоциирована со своим типом действия в главной программе. А в каждой процедуре события кнопки диалогового окна можно присваивать свойству DialogResult формы значение, соответствующее названию кнопки, как это сделано в следующем операторе программы:

**Задание на лабораторную работу.**

Создать приложение для Windows "Семерка" (LuckySeven). Эта программа имитирует игровой автомат со "счастливыми" Для отображения справочной информации о программе «Семерка» использовать вторую форму. Добавьте вторую форму с помощью команды Вторая форма должна отображать файл Readme.txt, который должен содержать справочную информацию.

**Порядок выполнения работы**

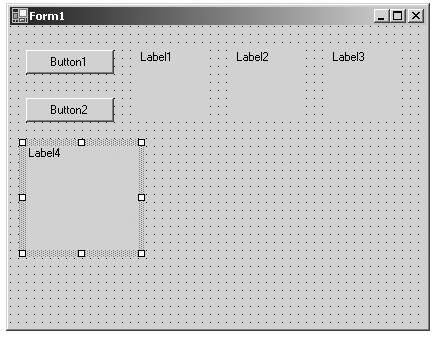
Графический интерфейс "Семерки" имеет две кнопки, три поля для показа "счастливых" чисел, цифровую фотографию, символизирующую выигрыши, и текст "Счастливая семерка". Чтобы сконструировать этот интерфейс, нужно создать семь объектов на форме Семерка, а затем изменить некоторые свойства в каждом из них. Затем нужно добавить код программы для кнопок Крутить и Выход, который обрабатывает нажатия пользователя на этих кнопках и генерирует случайные числа. Чтобы создать эту программу с самого начала, необходимо выполнить в Visual Basic три основных шага по разработке программы: создать интерфейс пользователя, настроить свойства и написать код программы. В таблице описан этот процесс для "Семерки".

|  |  |
| --- | --- |
| Таблица 2.1. | |
| **Шаг программирования** | **Число элементов** |
| Создать графический интерфейс пользователя | 7 объектов |
| Настроить свойства | 12 свойств |
| Написать код программы | 2 объекта |

**Создание пользовательского интерфейса**

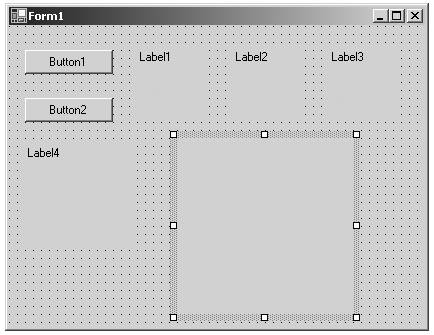
1. Создайте новый проект командой *Создать проект (New Project)* из меню *Файл (File)* (порядок создания нового проекта подробно описан в лабораторной работе № 1).
2. Выберите элемент Приложение Windows Forms и нажмите кнопку ОК.
3. В области элементов выберите элемент управления *Button* и добавьте на форму две кнопки.
4. Добавьте надписи, которые используются в программе для показа чисел.В области элементов выберите элемент управления *Label*. И создайте на форме четыре надписи с именами *Label1*, *Label2*, *Label3* и *Label4*.

Ваши надписи будут выглядеть примерно так, как показано на иллюстрации ниже. (Если надписи выглядят не вполне правильно, их размер можно изменить.)



**Добавление изображения**

1. Теперь добавьте на форму рамку для рисунка, чтобы показать графический приз, когда пользователь выбросит "7".В области элементов выберите элемент управления *PictureBox*.
2. Нарисуйте большой прямоугольник прямо под тремя надписями для цифр. В результате поле рисунка должна выглядеть так.



Этот объект в программе будет называться *PictureBox1*, и это имя в дальнейшем используется при написании кода программы.

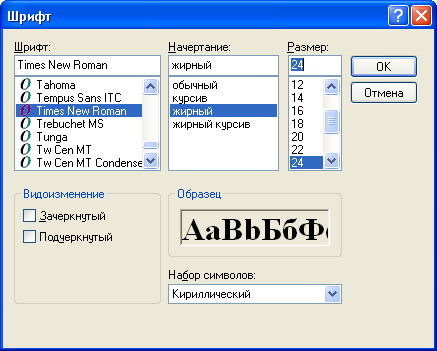
Теперь нужно настроить интерфейс, задав несколько свойств.

**Настройка свойств кнопки**

1. Щелкните на первой кнопке *Button1*.Текущее значение свойства *Text* измените на слово *"Крутить"* и нажмите клавишу (Enter).
2. Измените свойство *Text* для второй кнопки так, чтобы там было написано *"Выход".*

**Настройка свойств надписей для чисел**

1. Выберите первую надпись, предназначенную для чисел (*Label1* ), а затем, удерживая нажатой клавишу (Shift), щелкните на второй и третьей надписях. Вокруг каждой из надписей, по которой вы щелкаете, появляются прямоугольники выделения и маркеры изменения размера. Выделите все три надписи и отпустите клавишу (Shift). Теперь нужно изменить свойства *TextAlign*, *BorderStyle* и *Font* так, чтобы числа, которые будут появляться в этих надписях, находились в центре, были обведены рамкой и показывались одним и тем же шрифтом одинакового размера. Все эти свойства перечислены в окне Свойства в категории Внешний вид.
2. В окне Свойства выберите *TextAlign*, а затем нажмите справа на стрелку раскрывающегося списка. Появляется схема, показывающая варианты центровки текста в пределах поля надписи. Выберите вариант форматирования по центру MiddleCenter.
3. Теперь измените свойство *BorderStyle*. Выберите его в списке и нажмите справу стрелку раскрывающегося списка. Выберите FixedSingle, чтобы добавить вокруг каждой надписи тонкую рамку.
4. Теперь измените шрифт для надписей, изменив параметры свойства *Font*. Выберите его в окне Свойства, а затем нажмите справа кнопку многоточия. Появится диалоговое окно Шрифт, показанное ниже.



1. Измените шрифт на Times New Roman, стиль шрифта - на жирный, а его размер - на 24, а затем нажмите OK. Шрифт, стиль и размер текста на надписях изменятся на новые.
2. Теперь удалите текст для этих трех надписей, чтобы при запуске программы эти поля были пустыми. Выбор шрифта для надписей сохранится, так как он является отдельным свойством. Чтобы это сделать, необходимо выбрать каждую из надписей по отдельности и с помощью клавиши Delete очистить свойство Text.

**Установка свойств описательной надписи**

1. Выберите объект четвертой надписи *Label4*.
2. В окне Свойства измените свойство Text на *"Счастливая семерка"*.
3. Щелкните на свойстве Font, а затем на кнопке многоточия.
4. В диалоговом окне Шрифт измените шрифт на Arial, стиль шрифта - на жирный и размер - на 18. Нажмите OK, при этом шрифт надписи обновится.
5. Теперь измените цвет отображаемого текста. В окне Свойства выберите ForeColor и во втором столбце нажмите стрелку раскрывающегося списка. Выберите пурпурный цвет на закладке Польз.(Custom). Цвет текста в поле надписи изменится на пурпурный.

**Свойства поля для изображения**

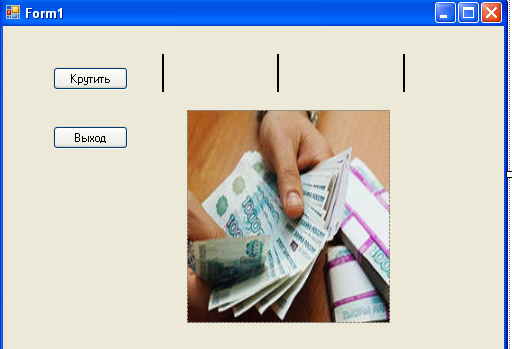
Объект для показа изображения будет содержать картинку с человеком, платящим игроку деньги при выигрыше (то есть когда в числовых надписях формы есть хотя бы одна семерка). Чтобы задать размеры картинки, нужно изменить свойство SizeMode, а в свойстве Image следует указать имя JPEG-файла, который будет загружен в поле изображения. Следует также задать свойство Visible, которое определяет состояние изображения в начале работы программы.

**Установка свойств поля изображения**

1. На поле формы щелкните на объекте поля изображения.
2. В окне Свойства выберите свойство *SizeMode*, которое находится в категории Поведение. Нажмите на стрелке раскрывающегося списка и выберите StretchImage. Если для *SizeMode* выбрано значение StretchImage, то прежде чем показать файл изображения, VisualStudio изменит его размер так, чтобы оно в точности совпадало с размерами поля.
3. В окне Свойства выберите свойство *Image*, а затем щелкните на кнопке многоточия во втором столбце. Появится диалоговое окно Открыть и выберите файл jpg с нужным изображением.
4. В окне Properties Свойства в категории *Behavior (Поведение)* выберите свойство *Visible*. Чтобы сделать изображение невидимым при запуске программы, выберите False.

Готовая форма должна выглядеть примерно так, как показано на рисунке ниже.

Дважды щелкните мышью на строке заголовка окна Properties (Свойства) , чтобы вернуть его в его закрепленное положение.



**Написание кода**

**Таблицы значений свойств**

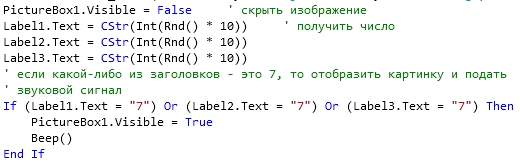
|  |  |  |
| --- | --- | --- |
| Таблица 2.2. | | |
| **Объект** | **Свойство** | **Значение** |
| Button1 | Text | "Крутить" |
| Button2 | Text | "Выход" |
| Label1, Label2, Label3 | BorderStyle | FixedSingle |
| Font | Times New Roman, жирный, 24 пункта |
| Text | (пусто) |
| TextAlign | MiddleCenter |
| Label4 | Text | "Счастливая семерка" |
| Font | Arial, жирный, 18 пунктов |
| ForeColor | Purple |
| PictureBox1 | Image | "абсолютный адрес файла с изображением" |
| SizeMode | StretchImage |
| Visible | False |

**Работа в Редакторе кода**

1. В поле формы дважды щелкните мышью на кнопке Выход. В центральном окне среды разработки VisualStudioпоявится редактор кодаМежду строками PrivateSub и EndSub введите



1. Дважды щелкните мышью на кнопке Крутить. Между строками PrivateSub и EndSub введите следующие ниже строки программы:



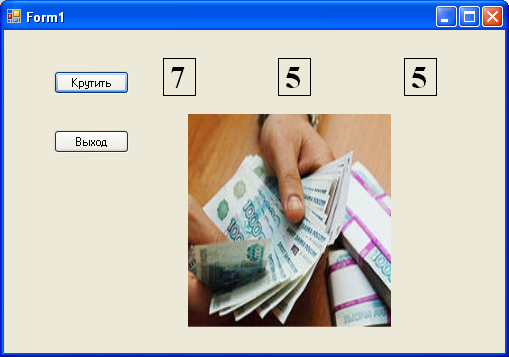
Функция Rnd в каждой строке возвращает случайное число в интервале между 0 и 1 (число с десятичной запятой и несколькими знаками после нее), а функция Int возвращает целую часть произведения этого случайного числа на 10. Это вычисление дает нам случайные числа в интервале между 0 и 9 - именно то, что нам требуется для нашего приложения игрового автомата.

Затем необходимо выполнить в нашем коде одно обязательное действие - скопировать эти случайные числа в три поля надписей на форме, но прежде, чем это сделать, эти числа должны быть преобразованы в текст с помощью функции CStr (converttostring - преобразовать в текст). Обратите внимание, как CStr, Int и Rnd объединены вместе в одно выражение в программе - они работают совместно и дают результат аналогично математической формуле. После вычисления и преобразования эти значения присваиваются свойствам Text первых трех надписей формы.

Последняя группа операторов программы проверяет, не равняется ли одно из случайных чисел "7". Если одно или несколько из этих чисел равны "7", то программа показывает средневековое изображение выплаты денег, а звуковой сигнал провозглашает о победе.

**Запуск программы "Семерка"**

1. Нажмите кнопку Начать отладку(Start) на стандартной панели инструментов. Программа "Семерка" будет скомпилирована и запущена в среде программирования. Появится созданный вами графический интерфейс.
2. Нажмите кнопку Крутить. Программа найдет три случайных числа и покажет их в надписях на форме.



Так как в первом поле надписи выпала "7", появляется цифровое фото, изображающее выплату денег, и компьютер издает звуковой сигнал. Вы выиграли! Звук, который вы услышите, зависит от установок в окне Soundsand Devises (Звуки и аудиоустройства) в панели управления Windows - чтобы сделать звук в этой игре интереснее, измените звук, установленный по умолчанию, на что-нибудь более динамичное.

1. Нажмите кнопку Крутить еще несколько раз, наблюдая за результатом "вращений" в числовых полях.

Почти в половине ваших экспериментов вы будете выигрывать (реальные шансы - 2,8 раза из 10; просто вам поначалу везет). В дальнейшем можно будет усложнить игру, выводя цифровое фото только тогда, кода появятся две или три семерки, или подсчитывая текущую сумму выигрышей.

1. Когда вы закончите экспериментировать с вашим новым детищем, нажмите кнопку Выход. Программа остановится, и на экране снова появится среда программирования.

**Совет**. Если вы снова запустите программу, то заметите, что "Семерка" показывает ту же самую последовательность случайных чисел. Здесь нет никакой ошибки - функция Rnd в VisualBasic спроектирована так, чтобы сначала отображать повторяющуюся последовательность чисел, чтобы программа выдавала одни и те же предсказуемые результаты в целях отладки. Чтобы создать истинно "случайные" числа, нужно использовать функцию Randomize.

**Следующий шаг: внесение дополнений в программу**

Добавьте в программу "Семерка" специальный оператор с именем Randomize.

1. Теперь вы должны добавить в процедуру Form\_Load - специальную процедуру, которая ассоциирована с формой и исполняется каждый раз при запуске программы - оператор Randomize.
2. Чтобы перейти к тексту процедуры Form\_Load, дважды щелкните мышью на форме (но не на одном из ее объектов). В редакторе кода появится процедура Form\_Load.
3. Введите , а затем нажмите на клавишу со стрелкой вниз. Оператор Randomize добавлен в программу и будет исполняться каждый раз при ее запуске. Чтобы создать истинно случайную начальную точку для функции Rnd в процедуре Button1\_Click, в Randomize используются системные часы. При использовании Randomize программа будет каждый раз выдавать новую последовательность, и числа не будут повторяться.



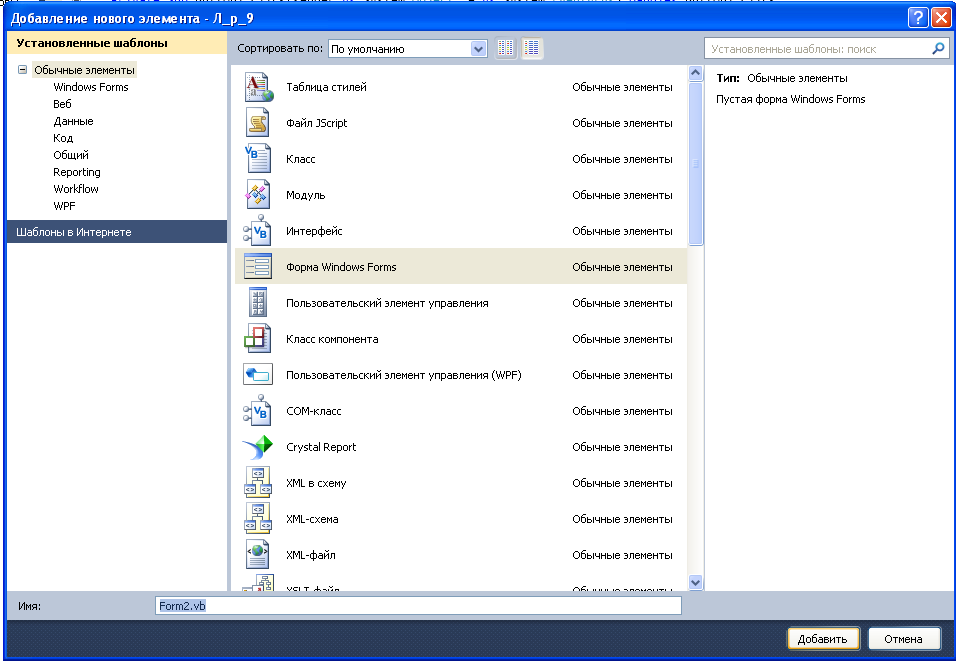
1. Запустите новую версию "Семерки".

### Работа с несколькими формами

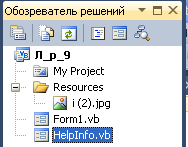
Использование второй формы для отображения справочной информации о программе «Семерка». Добавьте вторую форму с помощью команды *Добавить форму Windows* из меню *Проект*, а отображать эту форму на экране будете в коде программы с помощью метода *ShowDialog*. Вторая форма будет отображать файл Readme.txt, который должен содержать справочную информацию.

#### Добавление второй формы

1. Чтобы добавить в проект вторую форму, щелкните на команде Добавить форму Windows в меню Проект. Вы увидите такое диалоговое окно.



1. Введите в текстовом поле *Имя (Name)* имя файла HelpInfo.vb, а затем щелкните на*Добавить (Open)*. В проект Семерка будет добавлена вторая форма с именем HelpInfo.vb. Она появится в Обозревателе решений, как показано здесь.

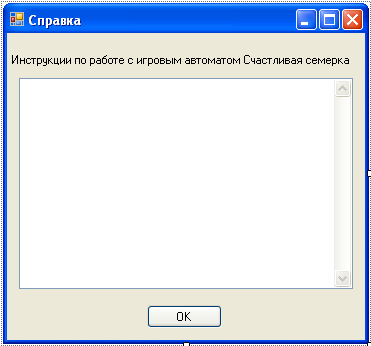


Теперь необходимо добавить в форму HelpInfo.vb несколько элементов управления.

1. Используйте элемент управления *Label* и нарисуйте в верхней части формы HelpInfo.vb метку. Сделайте длину этой метки равной ширине формы так, чтобы в ней уместился длинный текст.
2. Используйте элемент управления *TextBox* и создайте объект текстового поля.
3. Установите свойство *Multiline* этого объекта текстового поля на значение *True*, чтобы можно было изменить размер этого объекта.
4. Измените размер объекта текстового поля так, чтобы он заполнял почти все пространство формы.
5. Используйте элемент управления *Button* и создайте в нижней части формы объект кнопки.
6. Установите для объектов формы HelpInfo.vb следующие свойства:

|  |  |  |
| --- | --- | --- |
| **Объект** | **Свойство** | **Установка** |
| Label1 | Text | "Инструкции по работе с игровым автоматом Счастливая семерка" |
| TextBox1 | Scrollbars | Vertical |
| Text | пустой (empty) |
| Button1 | Text | "OK" |
| HelpInfo.vb | Text | "Справка" |

1. Форма HelpInfo.vb выглядит так.



1. Теперь необходимо ввести строку кода программы в процедуру события Button1\_Click формы HelpInfo.vb.
2. Чтобы отобразить в Редакторе кода процедуру события Button1\_Click, сделайте двойной щелчок мышью на кнопке **OK**.
3. Введите следующий оператор программы:



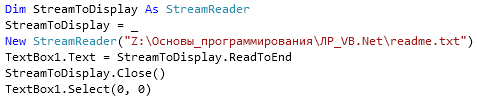
Форма HelpInfo.vb действует в этом проекте как диалоговое окно, так как она открывается в процедуре события Form1 с помощью метода *ShowDialog*. После того, как пользователь изучил справочную информацию в этом диалоговом окне, кнопка **OK** закрывает форму, устанавливая свойство *DialogResult*текущей формы (Me) на значение *DialogResult.OK*. Это значение является константой VisualBasic, указывающей, что диалоговое окно закрыто, при этом вызывающую процедуру должно вернуться значение "OK". Более сложное диалоговое окно может с помощью параллельных процедур событий кнопок возвращать и другие значения, такие, как *DialogResult.Cancel, DialogResult.No, DialogResult.Yes и DialogResult.Abort*. Однако когда устанавливается свойство *DialogResult*, форма автоматически закрывается.

1. Прокрутите код программы в Редакторе кода в начало. Введите следующий оператор Imports:



Этот оператор создает в проекте ссылку на библиотеку классов, содержащую класс StreamReader. Класс StreamReader не связан непосредственно с определением или использованием дополнительных форм - просто используем его как быстрый способ добавления в новую используемую форму текстовой информации.

1. Снова отобразите форму HelpInfo.vb, а затем сделайте двойной щелчок мышью на фоне формы. В Редакторе кода появится процедура события HelpInfo\_Load.
2. Введите следующие операторы программы:



Вместо того чтобы вводить содержимое справочного файла в свойство *Text* объекта текстового поля, для открытия, чтения и отображения в объекте текстового поля файла Readme.txt используйте класс *StreamReader*. Файл Readme.txt является документом поддержки продукта, содержащим информацию об использовании программы Счастливая семерка. Он содержит инструкции по работе, информацию по деинсталляции и контактную информацию. К сожалению, выводить русские символы с помощью этого метода затруднительно. Поэтому мы выведем содержимое англоязычного файла Readme.txt, который вы самостоятельно создадите**.**

*StreamReader* - это альтернативный по отношению к использованию функции Visual Basic FileOpen способ открывать текстовый файл, предоставляемый .NET Framework. Чтобы использовать StreamReader, необходимо включить в начало кода вашей формы библиотеку классов System.IO. Затем объявите переменную (StreamToDisplay) типа StreamReader, которая будет хранить содержимое текстового файла, и откройте этот текстовый файл, указав путь к нему. Наконец, с помощью метода ReadToEnd, который получает из файла весь текст, начиная с текущей позиции (начало текстового фала) и до конца этого файла, вы считываете содержимое текстового файла в переменную StreamToDisplay, и присваиваете ее свойству *Text* текстового поля. Оператор StreamReader.Close закрывает текстовый файл, а метод *Select* удаляет выделение из текста, размещенного в объекте текстового поля.

Вы закончили создание формы HelpInfo.vb. Теперь вы добавите объект кнопки и некий код в первую форму.

#### Отображение второй формы с помощью процедуры события

1. Щелкните в Обозревателе решений на Form1.vb, а затем щелкните на кнопке *Просмотреть конструктор*. В среде разработки появится форма Семерка. Теперь необходимо добавить в нижний правый угол формы кнопку **Справка**.
2. Используйте элемент управления *Button* и нарисуйте в нижнем правом углу формы небольшую кнопку.
3. Используйте окно Свойства, чтобы установить свойство *Text* кнопки на значение **Справка**. Ваша форма должна выглядеть примерно так.
4. Чтобы отобразить в Редакторе кода процедуру события Button3\_Click, сделайте двойной щелчок мышью на кнопке **Справка**.
5. Введите следующие операторы программы:



Эти операторы демонстрируют, как объявлять и отображать в коде программы вторую форму. VisualBasic .NET требует, чтобы перед использованием второй формы явно объявили переменную типа формы. Добавив в проект форму HelpInfo.vb, вы создали класс с именем HelpInfo; теперь вы используете его для объявления с помощью оператора Dim переменной с именем frmHelpDialog.

Второй оператор программы использует переменную frmHelpDialog, открывая форму HelpInfo.vb как диалоговое окно с помощью метода ShowDialog. Также можно использовать для открытия этой формы метод Show, но в этом случае VisualBasic не рассматривал бы HelpInfo.vb как диалоговое окно; форма была бы немодальной, и пользователь мог бы переключаться между этой и главной формами по мере необходимости. В дополнение к этому свойство DialogResult в процедуре события Button1\_Click формы HelpInfo.vb не закрывало бы форму HelpInfo.vb - вместо этого потребовался бы оператор программы Me.Close

При создании собственных проектов помните о различиях между модальными и немодальными формами. Между этими типами форм существуют заметные различия, и вы обнаружите, что каждый из них дает пользователю определенные выгоды.

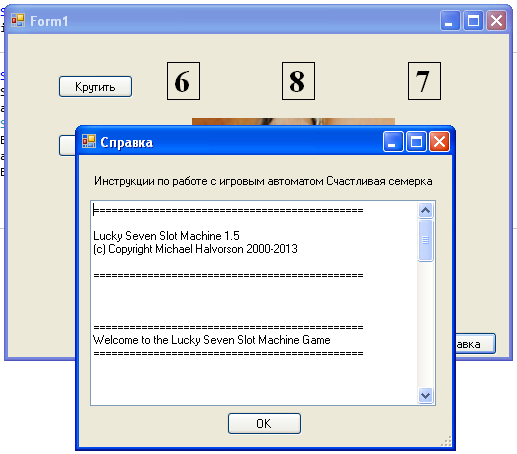
Теперь запустите программу, чтобы увидеть, как работает приложение, содержащее несколько форм.

#### Запуск программы

1. Щелкните на кнопке *Начать отладку* стандартной панели инструментов.

Появится главная форма LuckySeven.

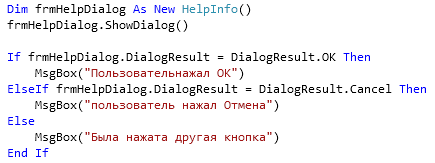
1. Чтобы немного поиграть в игру, щелкните семь или восемь раз на кнопке **Крутить**.
2. Щелкните на кнопке **Справка** первой формы. VisualBasic откроет вторую форму проекта (HelpInfo.vb) и отобразит в объекте текстового поля файл Readme.txt. Форма будет выглядеть примерно так.



1. Чтобы просмотреть весь файл Readme, используйте вертикальную полосу прокрутки.
2. Чтобы закрыть форму HelpInfo.vb, щелкните на кнопке **OK**. Форма закроется, и снова станет активной первая форма.
3. Еще несколько раз щелкните на кнопке **Крутить**, а затем снова щелкните на кнопке **Справка**. Снова появится форма HelpInfo.vb, работающая точно так же, как и в первый раз. Обратите внимание, что пока активна вторая форма, вы не можете активизировать первую. Это происходит из-за того, что вторая форма является диалоговым окном или модальной формой, и прежде, чем вы сможете продолжить работу с программой, необходимо закрыть ее.
4. Щелкните на кнопке **OK**, а затем на кнопке **Закрыть** первой формы. Программа остановится, и вернется среда разработки.

#### Использование свойства DialogResult в вызывающей форме

В вызывающей процедуре события - другими словами, в процедуре события Button3\_Click формы Form1 - вы напишите дополнительный код программы, который будет определять, на какой кнопке диалогового окна щелкнул пользователь. Эта информация передается обратно в вызывающую процедуру через свойство *DialogResult*, которое содержится в имени переменной, использованном для объявления и создания экземпляра второй формы. Например, следующий код в Form1 может быть использован для того, чтобы проверить, щелкнул ли пользователь на кнопке **OK**, Cancel или какой-либо еще кнопке диалогового окна. (Первые две строки у вас есть, они показывают имя переменной, которое необходимо использовать.)



Подобные процедуры событий, которые объявляют переменную, открывают диалоговое окно и обрабатывают выбор, сделанный в этом диалоговом окне, позволяют добавлять в программы любое количество форм и создавать интерфейс пользователя, который выглядит профессионально, является гибким и дружественным к пользователю.

**Контрольные вопросы:**

1. Назвать класс, свойства которого наследует каждая форма.
2. Характеристики модальных и немодальных окон.
3. Назначение функций CStr, Int и Rnd.
4. Назначение и различия методов Show и ShowDialog.
5. Назначение оператора в программе.



1. Что нам дает использование класса StreamReader.
2. Использование свойства DialogResult в вызывающей форме

Лабораторная работа №10

**Разработка многооконного приложения**

**Цель:** научиться создавать многооконные приложения (MDI-приложения).

## Ход работы:

## Изучить теоретическую часть.

## Выполнить задания, следуя указаниям.

## Ответить на контрольные вопросы (в устной форме).

## Предъявить преподавателю результаты работы программы и исходные коды.

## Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, программный код, результаты тестирования программы, выводы).

**🕮Теоретическая часть**

**Что такое MDI?**

MDI-форма, которую также называют родительской, представляет собой контейнер для дочерних форм приложения.MDI-форма имеет ряд характеристик:

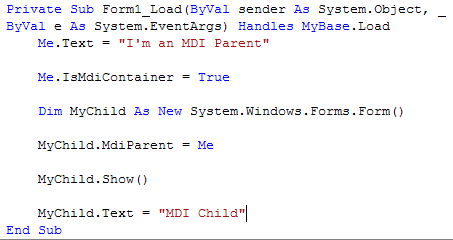
* Создаваемое приложение может иметь только одну MDI-форму.
* На панели задач Windows представляется лишь исходное MDI-окно, а все остальные дочерние окна на панели задач не представляются.
* При минимизации MDI-окна, минимизируются и все дочерние окна. При восстановлении - восстанавливаются.

MDI расшифровывается как **multipledocumentinterface** (многодокументный интерфейс). В приложениях с MDI, в основном (родительском) окне можно окрыть более одного дочернего окна. Данная возможность обычно используется в электронных таблицах или текстовых редакторах.

В прежних версиях VB при программировании приложений с интерфейсом **MDI** (**MultipleDocumentInterface**) родительская форма MDI выбиралась на стадии конструирования. В .NET эта задача решается иначе – свойству **IsMdiContainer** формы задается значение True. Программист создает дочерние формы MDI на стадии конструирования или выполнения, а затем заносит в их свойство **MdiParent** ссылку на форму со свойством **IsMdiContainer**, равным True.

Таким образом, в программах VB.NET можно сделать то, что было практически нереально в предыдущих версиях VB, – изменять связи MDI во время работы программы. Кроме того, приложение может содержать несколько родительских форм MDI; в VB6 такая возможность не поддерживалась.

Рассмотрим пример. Создайте приложение со следующей процедурой **Forml\_Load**:



Конечно, это весьма жалкое подобие приложения MDI. В приложениях MDI обычно присутствует меню **Window**, позволяющее расположить открытые дочерние окна "черепицей" или "мозаикой", а также активизировать любое дочернее окно.

**Задание на лабораторную работу.**

Создать программу, позволяющую просматривать графические файлы, таким образом, чтобы каждый выбранный графический файл открывался в новом дочернем окне.

Для этого нам потребуются следующие компоненты:

* компонент *MainMenu*, который мы будем использовать для открытия диалоговой панели для выбора файлов;
* компонент *OpenFileDialog*, который будет использоваться для выбора графического файла;
* компонент *PictureBox*, в котором мы будем отображать содержимое графического файла.

**Создание пользовательского интерфейса**

1. Создайте новый проект командой *Создать проект (NewProject)* из меню *Файл (File)* (порядок создания нового проекта подробно описан в лабораторной работе № 1).
2. Выберите элемент *Приложение Windows Forms* и нажмите кнопку ОК.
3. Добавьте на форму компоненты *MainMenu*, *OpenFileDialog*, *PictureBox*.

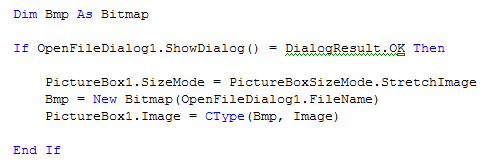
**Настройка свойств**

* 1. Задайте для формы свойство *Text* равным *GraphicsView*.
  2. Далее значение свойства *StartPosition*измените на *CenterScreen* — в результате наше приложение всегда будет отображаться в середине экрана.
  3. Если вы хотите, чтобы размер окна нельзя было изменить, установите значение *FormBorderStyle* равным *FixedSingle* или *Fixed3D*. В противном случае при изменении размера окна нарушится расположение элементов внутри него.
  4. Для элемента *PictureBox* установите трехмерную рамку (*BorderStyle* = *Fixed3D*), а также измените свойство *Anchor* — с его помощью вы укажете, как должны изменяться размеры элемента в зависимости от размеров окна.
  5. С помощью компонента *MainMenu* создайте элемент File и два подэлемента — Open и Exit. Для этого воспользуйтесь редактором меню, предоставляемым VisualStudio .NET и вызываемым при щелчке мышью по компоненту *MainMenu*.

**Написание кода**

После того как меню готово, необходимо написать код, который будет выполняться при выборе той или иной команды. Для этого дважды щелкните мышью по соответствующей команде и попадете в редактор кода.

1. Для команды File | Open напишите следующий код:



В первой строке мы создаем переменную типа Bitmap, в которую будем загружать содержимое выбранного нами графического файла. Затем мы вызываем метод ShowDialog объекта *OpenFileDialog*, чтобы отобразить на экране панель выбора файлов. Если файл выбран и нажата кнопка Open, мы выполняем следующие действия:

* задаем режим отображения графики в компоненте *PictureBox* — в нашем примере графика должна быть «растянута» во весь размер компонента;
* создаем новый экземпляр объекта Bitmap, в который загружаем содержимое выбранного нами файла;
* отображаем содержимое объекта Bitmap в компоненте PictureBox.

1. Для команды File | Exit мы напишите следующий код:

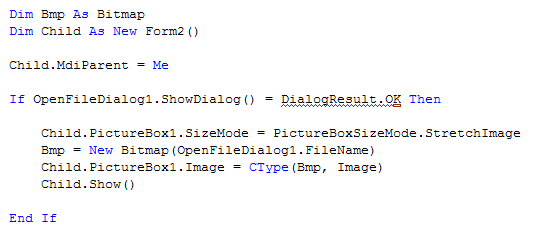


Выполнение этого кода приведет к завершению приложения и к закрытию его окна.

**Создание MDI-приложений**

После того как вы освоили создание меню, измените функциональность приложения таким образом, чтобы каждый выбранный графический файл открывался в новом дочернем окне, то есть превратите однооконное приложение в MultipleDocumentInterface (MDI)-приложение.

1. Создание MDI-приложения начинается с того, что изменяется свойство IsMDIContainer формы и присваивается ему значение True.
2. После этого мы используйте команду *Проект/Добавить форму (Project | AddWindowsForm)* для добавления к проекту еще одной формы, которая будет служить в качестве дочерней формы. В диалоговой панели*Добавить новый элемент (AddNewItem)* выберите элемент *WindowsForm* и нажимите кнопку Открыть (Open).
3. Перенесите компонент *PictureBox* из главной формы приложения в дочернюю форму. Для этого следует выбрать компонент *PictureBox*, нажать клавишу Ctrl-X, затем перейти на дочернюю форму и нажать клавишу Ctrl-V.
4. Ваше MDI-приложение практически готово —осталось лишь изменить код, выполняющийся при выборе команды File|Open:



Сначала мы создаем переменную Child, которая имеет тип Form2 и будет использоваться для создания дочерних форм. Затем указываем родительское окно дочерней формы — ключевое слово Me используется в VisualBasic для указания текущего класса. Поскольку компонент *PictureBox* теперь располагается в дочернем окне, мы изменяем способ обращения к нему. После того как графика загружена, мы показываем нашу дочернюю форму на экране.

1. Для того чтобы сделать приложение более удобным, давайте показывать в заголовке дочернего окна полное имя отображаемого в нем файла. Для этого добавьте в код обработчика события следующую строку:



1. Если же мы вы хотите отобразить только имя файла, нам надо написать такой код:



и добавить ссылку на пространство имен System.IO в самом начале кода нашей программы:



1. Теперь добавьте код, который позволит управлять дочерними окнами. Для этого создайте еще один пункт меню — Childs и добавьте в него следующие элементы: TileHorizontal; TileVertical; Cascade; ArrangeIcons.
2. Затем напишите код для каждой из команд:

КомандаTileHorizontal



КомандаTileVertical



Команда Cascade



Команда ArrangeIcons



1. Включите еще одну возможность — меню для переключения между окнами. Добавьте к главному меню элемент *Window* и измените значение его свойства *MdiList* на *True*.
2. Теперь используйте еще один компонент — *ContextMenu*, который позволит изменять способ отображения графики в компоненте PictureBox. Добавьте компонент *ContextMenu* к дочерней форме и создайте в нем следующие элементы: Normal; Center; Stretch; Autosize, которые повторяют все возможные значения свойства *SizeMode* компонента *PictureBox*.
3. Установите свойство *ContextMenu* компонента *PictureBox* в ContextMenu1. Затем напишите код, который будет выполняться при вызове каждой из команд контекстного меню:

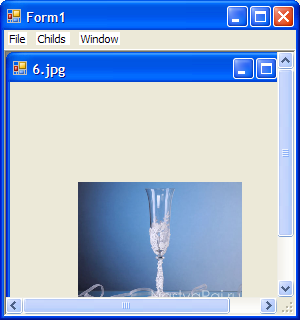


После этого надо удалить строку:



из обработчика события выбора команды File | Open в главном меню нашего приложения.

1. Запустите и протестируйте MDI-приложение



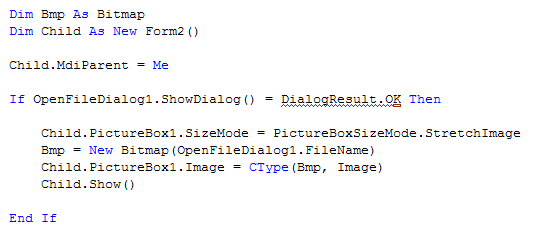
**Контрольные вопросы:**

1. Опишите форму MDI.
2. Какое свойство необходимо изменить для родительской формы MDI.
3. Назначение оператора в программе.



1. Поясните следующий код

.



1. Что дает использование строки кода :



**Список использованных источников**

# <http://msdn.microsoft.com/ru-ru/library/vstudio/jj159353.aspx> Msdn.microsoft.com: Средства и языки разработки

1. <http://www.intuit.ru/studies/courses/78/78/info> [Intuit.ru: Visual Basic.NET](http://www.intuit.ru/studies/courses/78/78/info)
2. Аляев Ю., Козлов О. Алгоритмизация и языки программирования Pascal, C++, Visual Basic. – М.: Финансы и статистика, 2011.
3. Голицына О.Л., Попов И.И. Основы алгоритмизации и программирования: Учебное пособие.– М.: Форум: Инфра-М, 2009.
4. Грейди Буч. Объектно-ориентированный анализ и проектирование с примерами приложений на C++ /Пер. с англ. – 2-е изд.– М.: Бином, 2010.
5. Давыдова Н.А., Боровская Е.В., Программирование. Учебное пособие. [Интуит.ру](http://www.booka.ru/search?q=%D0%98%D0%BD%D1%82%D1%83%D0%B8%D1%82.%D1%80%D1%83&st=publisher), 2011г.
6. Джеймс Фокселл. Освой самостоятельно Visual Basic.NET за 24 часа. – М.: Вильямс, 2012.
7. Дэн Кларк. Объектно-ориентированное программирование в Visual Basic .NET. – СПб.: Питер, 2009.
8. Зак Д. Самоучитель Visual Basic.Net. – Киев:ВНV; СПб.: Питер, 2010.
9. Кетков Ю., Кетков А. Практика программирования: Visual Basic, С++Builder. – СПб.: BHV, 2010.
10. Маккей А., «Введение в .NET 4.0 и Visual Studio 2010 для профессионалов», 2010
11. Ник Рендольф, Дэвид Гарднер, Крис Андерсон, Майкл Минутилло, Visual Studio 2010 для профессионалов, «ДИАЛЕКТИКА», 2011
12. Пономарев В. Visual Basic.NET: – Экспресс-курс. – СПб.: BHV - Санкт - Петербург, 2010.
13. Семакин И.Г., Шестаков А.П. Основы программирования: Учебник. – М.: Мастерство, 2011.

## Федоров А., Microsoft Visual Studio 2010: Первое знакомство, «Microsoft», 2009