***Тема 4.1. Интегрированная среда разработки***

## *Visual Studio .NET и основные понятия*

**[4.1.1. Технология визуального программирования и основные понятия](#_4.1.1._Технология_визуального)**

**[4.1.2. Главное окно VS и создание проекта](#_4.1.2._Главное_окно)**

**[4.1.3. Главное меню VS .NET](#_4.1.4._Стандартная_панель)**

[**4.1.4. Стандартная панель инструментов**](#_4.1.4._Стандартная_панель)

[**4.1.5. Окна панелей VS**](#_4.1.6._Настройка_параметров)

**[4.1.6. Настройка параметров интегрированной среды разработки](#_4.1.6._Настройка_параметров)**

[**4.1.7. Пример создания проекта**](#_4.1.7._Пример_создания)

[**4.1.8. Контрольные вопросы по теме «Интегрированная среда разработки   
 Visual Studio .NET и основные понятия»**](#_4.1.7._Контрольные_вопросы)

**4.1.9. Тестовые задания по теме «Интегрированная среда разработки   
 Visual Studio .NET и основные понятия»**

### 4.1.1. Технология визуального программирования и основные понятия

В разделе 2 дисциплины Информатика (Тема 2.2) уже были рассмотрены основные моменты технологии визуального программирования в интегрированной среде обработки **Visual Studio .NET**. Эту тему рекомендуется проработать перед изучением данной темы, в которой мы обращаемся к программированию уже на конкретном языке высокого уровня – языке **Visual Basic .NET**. По сравнению с предыдущими поколениями языков программирования, язык программирования высокого уровня – **Visual Basic .NET**(в дальнейшем **VB)** имеет ряд принципиальных отличий. Это мощная, профессиональная система программирования, позволяющая быстро и эффективно создавать приложения для **Microsoft Windows (MS Windows).** Поскольку система является *«****визуальной****»,* программист может создавать видимую часть приложения **MS Windows,** не написав ни строчки программного кода. Эта видимая часть является основой интерфейса пользователя *«****программа – пользователь****»,* с помощью которого осуществляется взаимодействие пользователя с программой. Разработка, как программного кода, так и программного интерфейса осуществляется на принципах ***объектно-ориентированного подхода****,* реализованного в **Visual Studio .NET (VS).** Приложения (программные проекты) выполняются под управлением **MS Windows** с помощью ***Интегрированной Среды Разработки (Integrated Development Environment*** –***IDE)*. *IDE*** – это программный комплекс, объединяющий в себе ряд средств, предназначенных для использования в процессе разработки ***программного обеспечения***. Как правило, **IDE** включает в себя редактор программного кода, компилятор и (или) интерпретатор программного кода, отладчик, средства для упрощения разработки пользовательского интерфейса и др.

Для таких приложений характерным является существование на экране в любой момент времени множества объектов: форм, окон, кнопок, меню, текстовых и диалоговых окон, линеек прокрутки и т.п. Пользователь имеет определенную (в зависимости от алгоритма решаемой задачи) свободу выбора в части использования этих объектов. Он может сделать ***щелчок*** *(****нажатие и затем отпускание****)* кнопкой мыши, перетащить объект, ввести данные в окно и т.п. В большинстве случаев не предполагается заранее установленная последовательность действий. Как правило, программист, создавая программу, не должен ограничивать действия пользователя. Он должен разрабатывать программу, которая адекватно реагирует на любое действие пользователя, даже неправильное (в хорошей программе непредусмотренных действий не должно быть). Для каждого объекта на экране существует ряд возможных событий. Одни из них производятся пользователем: щелчок или двойной щелчок по кнопке мыши, перетаскивание объекта, нажатие клавиши клавиатуры, ввод текста в окно и т.п. Некоторые события происходят в результате совершения других событий: например, окно открывается или закрывается, элемент управления становится активным (доступным) или теряет активность.

Каждое событие проявляется в определенных действиях (откликах, реакции, поведении) программы. Эти действия можно разделить на две группы.

Первая является следствием свойств объекта. Эти свойства устанавливаются программистом из некоторого стандартного перечня свойств, заданного системой программирования **VB** и самой системой **MS Windows.** Примером таких действий является свертывание окна после щелчка по кнопке ***Свернуть.*** Изменить поведение объекта в подобных случаях программист не может. Однако во многих случаях имеется определенная возможность вариации отклика объекта на событие (текст, шрифт, цвет, размер, место на экране и др.) путем задания определенных свойств.

Вторая группа действий (отклика) на события полностью определяется программистом. Это обеспечивается возможностью задания процедуры **VB** для каждого возможного для объекта события. Первоначально каждая такая процедура является пустой, и, следовательно, когда происходит событие, ничего не случается. Теоретически можно создать процедуру для любого события, но на практике программист заполняет кодом процедуры только для тех событий, которые в данной программе представляют интерес.

Таким образом, **VB** предоставляет пользователю объектно-ориентированную среду программирования, основанную на событиях. Упрощенно говоря, процесс разработки про­граммы в этой среде сводится к выбору набора объектов и их свойств, заданию событий и процедур их обработки, которые в совокупности обеспечивают решение поставленной задачи.

Основными терминами визуального программирования, как было сказано выше, являются ***объект*, *свойство, класс, событие, процедура*,** которые в любой среде визуальной разработки приложений имеют одинаковый смысл.

***Объект (Object)***– это некая отдельная сущность, отличающаяся от других сущностей особыми свойствами, поведением, взаимодействием с другими объектами приложения. При объектно-ориентированном подходе в программировании любое приложение представляет собой набор взаимосвязанных объектов, реализующих необходимые функциональные требования, предъявленные к приложению. Объект всегда конкретен и реально существует на форме (в качестве элементов управления) или в приложении (в виде блоков программ).

Основной единицей в объектно-ориентированном программировании является программный объект, который объединяет в себе как описывающие его данные (свойства), так и средства обработки этих данных (методы). То есть, программные объекты обладают свойствами, могут использовать методы и реагировать на события.

Основными объектами визуального программирования являются форма и элементы управления.

***Форма (Form)*** –это объект, представляющий собой окно, на котором размещаются элементы управления, например: ***Кнопка (Button); Метка (Label); ТекстовоеПоле (TextBox); Флажок (CheckBox); Список (ListBox); РаскрывающийсяСписок(ComboBox); Переключатель (RadioButton)* идр.**

Среда визуального программирования позволяет увидеть объекты на экране на этапе проектирования приложения, дает возможность настроить объекты, изменяя их расположение, размеры и другие свойства. Форма является ***контейнером объектов*,** в то же время форма также является объектом. При размещении объекта на форме основные параметры объекта сохраняются в исполняемом программном коде. При этом объектами также являются программные блоки или фрагменты программного кода, из которых «собирается» проектируемое приложение.

Таким образом, любой объект характеризуется такими основными понятиями, как:

* ***состояние*** – каждый объект всегда находится в определенном состоянии, которое определяется набором свойств объекта;
* ***свойство*** – признак, некоторое отдельное качество объекта.

***Свойство объекта (Properties)*** – это значение или характеристика, хранимая объектом. В то же время совокупность свойств объекта определяет его состояние. Например, *Текст (****Text****)* или *Цвет (****BackColor)*** формы, способ *Выравнивания текста (****Alignment****)* или *Цвет символов (****ForeColor)*** текстового поля – это свойства. В программе обращение к объекту происходит ***по имени объекта***. Свойства объектов можно задавать в окне **Свойства (Properties)** и менять свойства объектов в программном коде.

В программном коде существуют определенные правила (определенный формат) для установки свойства –***имя объекта и название свойства согласно правилам точечной нотации разделяются точкой***. Например, строка

|  |
| --- |
| **Form1.Text = "Компьютер"** |

вводится в программу для установки свойства *Text* объекта *Form1* в значение ***«Компьютер».***

***Класс(Class)***– это совокупность объектов, обладающих общими свойствами и поведением. Например, *Button1(Кнопка1)*на ***Форме*** со всеми своими конкретными свойствами действием является объектом класса ***Button.*** Класс характеризуется такими основополагающими понятиями объектно-ориентированного программирования, как:

* ***наследование***–объект класса обладает всеми свойствами, методами и событиями класса;
* ***инкапсуляция*** – сокрытие сложного механизма действия объекта;
* ***полиморфизм*** – объекты разных классов могут использовать одноименные методы, работающие по-разному для разных объектов.

***Коллекция объектов*** – это набор объектов, объединенных общим именем, причем не обязательно это объекты одного класса. Примером коллекций могут служить встроенные в **VB** коллекции, в числе которых:

* ***Forms (Формы)*** –множество загруженных форм приложения;
* ***Controls (Элементы управления)*** –множество всех элементов управления в форме.

***Пространство имен*** – это иерархическая библиотека классов, организованная под уникальными именами такими, например, как **System.Windows** или **System.Math*.*** Для доступа к классам и основанным на них объектам, входящим в пространство имен, в начале кода программы может быть помещен оператор Imports***.*** Каждый проект в **VS** также содержит корневое пространство имен. Пространство имен в документации по **VS** называется также ***библиотеками классов***.

***Событие***–это средство взаимодействия объектов друг с другом. Событие может  
создаваться пользователем или возникать в результате воздействия других программных объектов. Объекты могут генерировать события и выполнять действия вответ на заданные события. Примеры событий – **Load *(Загрузка)*** формы**, Click *(Щелчок)*** мышью по объекту, **DblClick *(Двойной щелчок)*** мышью и другие***.*** В качестве реакции на события вызывается событийная процедура, которая может изменять свойства объекта, вызывать его

В языке **VB,** как и во многих других языках программирования, предназначенных для написания приложений для **MS Windows,** используется ***событийно-управляемая модель программирования.*** Фактически это означает, что ни одно приложение не может функционировать само по себе, не взаимодействуя с другими приложениями и с операционной системой. Для примера возьмем самый простой случай, когда команда задана нажатием клавиши на клавиатуре. Она не может непосредственно воздействовать ни на одно из приложений, и **MS Windows** в данном случае выступает в роли посредника. Восприняв событие, связанное с нажатием клавиши, операционная система (**ОС**) решает, куда передать обработку этого события. Затем сообщение о нажатии клавиши посылается приложению. Приложение обрабатывает его, анализируя ввод с клавиатуры, и отображает сообщение, например, в активном текстовом поле.

Приложения, созданные **с** помощью **VB,** работают потому же принципу. При этом **VB** выполняет как бы внутреннюю работу. Например, когда в приложении происходит щелчок мышью по кнопке, возникает событие – **VB** перехватывает соответствующее сообщение **MS Windows** и вызывает событие ***Click*** объекта ***Button.***

Обратите внимание, что в программах, управляемых событиями, ***нет сплошного программного кода, который выполняется от начала до конца.*** Это значит, что после запуска программы у пользователя нет и четкого представления о конкретном пути выполнения алгоритма. То есть, можно в любое время щелкнуть по какой-либо кнопке в приложении или выполнить, например, ввод текста в поле, а можно при желании и прекратить обработку. Поэтому приложение вынуждено ждать, пока одно из этих событий не наступит, и только потом выполняет заданный программный код. Таким образом, ***для начала выполнения программного кода всегда необходимо событие*** – это одно из важнейших правил создания приложений в среде **MS Windows.**

Откликом на события является вызов ***событийной процедуры****,* которая была ранее создана программистом. Смысл такой процедуры – выполнение запрограммированных действий: изменение свойств объекта, вызов методов, выполнение вычислений по заданному алгоритму с выводом результата в определенные места приложения и т.п. Например, если пользователь нажимает кнопку, то в качестве отклика выполняется некоторая последовательность действий, которая определена в событийной процедуре в соответствии с заданным алгоритмом.

Кроме событийных процедур пользователь может создать общие процедуры (процедуры пользователя).

***Метод обработки события*** – это законченный программный код, который реализует отдельный алгоритм. Для каждого события можно запрограммировать отклик, т. е. реакцию объекта на произошедшее событие, в результате будет выполнена некоторая последовательность действий – метод обработки события. В событийной процедуре может участвовать несколько объектов.

Имя событийной процедуры состоит из трех частей (включая подчеркивание):

|  |
| --- |
| **ИмяОбъекта\_ИмяСобытия** |

Таким образом, каждый раз, когда вы создаете программный код, который должен выполняться в ответ на событие, вы создаете событийную процедуру, которая вызывается автоматически при возникновении соответствующего события (например, нажатие кнопки).

***Процедура пользователя*** – это также законченный программный код, который реализует отдельный алгоритм. Однако вызов такой процедуры происходит по имени этой процедуры, а не как результат какого-либо события.

***Метод*** – это процедура, которая реализует действия, возможные с объектом. В результате этих действий в объекте что-либо меняется.

**VB** предоставляет пользователю большое количество встроенных процедур, готовых к использованию. Эти процедуры являются неотъемлемой частью библиотеки классов, то есть методов классов, **Microsoft .NET Framework**.

Средства процедурного программирования (процедуры) – неотъемлемая часть любого языка программирования, которая является одной из основных частей парадигмы современного программирования. Но лишь при появлении языков, поддерживающих объектно-ориентированное программирование, были введены особые типы встроенных процедур, названные ***методами***. ***Методы*** «работают» как процедуры, но принадлежат конкретным объектам, так же, как и свойства. В программном коде обратиться к методу можно так же, как и к свойству ***–*** с использованием точечной нотации.

В среде **VB** можно создавать ***собственные* процедуры,** определяемые (создаваемые) программистом*.*

***Обратите внимание, что любая программа (программный код), написанная на языке VB состоит из множества процедур, то есть, весь программный код находится как бы внутри множества процедур – подпрограмм и функций.***

Процедуры-подпрограммы начинаются ключевым словом **Sub** и заканчиваются ключевыми словами **End Sub**, а процедуры-функции начинаются ключевым словом **Function** и заканчиваются ключевыми словами **End Function*,*** между которыми помещается программный код данной процедуры. Программный код каждой процедуры состоит из одного или нескольких операторов языка программирования **VB**.

***Оператор*** – это конструкция языка программирования, задающая одну или несколько ***операций****,* производимых над ***операндами****.* В качестве операндов могут выступать константы, переменные, выражения, функции и т.д. Любой оператор записывается в соответствии с синтаксическими правилами языка программирования. Например, в операторе присваивания знак "**=**" читается "присвоить" и означает, что в память компьютера записывается значение, равное значению выражения справа от знака "**=**". Например, строка **Form1.Text = "Форма с картинкой"** означает, что в памяти с именем ***Form1.Text*** записывается значение – текст ***«Форма с картинкой»****.* А строка **Р=А+В+С** используется в программе для вычисления значения выражения и присваивания его переменной ***Р.***

***Проект*** – это разрабатываемое приложение (программа) – набор файлов, в которых хранится информация обо всех компонентах, используемых в данном приложении. Поскольку **VB,** с одной стороны – это ***система объектно-ориентированного визуального программирования,*** так как позволяет программировать алгоритмы на этом языке, а с другой стороны, эта система является ***средой проектирования,*** то результатом процессов программирования и проектирования является проект, который объединяет в себе программный код и графический интерфейс. Поэтому, чтобы записать на диск или считать с диска какую – либо программу, необходимо записывать или считывать проект, в котором находится эта программа. Часто, оказывается, удобно сохранять в отдельном файле форму проекта, что позволяет использовать данную форму в других проектах. После того, как все необходимые элементы внесены в форму и написан весь программный код (программа), приложение считается созданным и при этом будет создан проект данного приложения. Приложение в **VB** состоит из множества файлов, включая файл проекта с расширением **.*vbproj*** .Файл проекта содержит информацию, относящуюся к одной программной задаче – одному приложению.

Проект является также единицей компиляции. Результатом компиляции проекта является сборка. Каждый проект содержит одно или несколько пространств имен. На начальном этапе создания проекта по заданному типу проекта автоматически строится каркас проекта, состоящий из классов, являющихся наследниками классов, входящих в состав библиотеки **FCL** (Framework Class Library- библиотеки классов платформы **.NET Framework** – см. Тему 2.2.5). Так, если разработчик указывает, что он хочет построить проект типа «**WindowsFormsApplication**», то в состав каркаса проекта по умолчанию войдет класс Form1 – наследник библиотечного класса Form. Разработчик проекта населит созданную форму элементами управления – объектами соответствующих классов, тем самым расширив возможности класса, построенного по умолчанию.

Каждый проект содержит всю информацию, необходимую для построения сборки. В проект входят все файлы с классами, построенные автоматически в момент создания проекта, и файлы с классами, созданные разработчиком проекта. В проект входят установки и ресурсы, требуемые для работы. Частью проекта является файл, содержащий описание сборки.

В зависимости от выбранного типа проект может быть выполняемым или невыполняемым. К выполняемым проектам относятся, например, проекты типа **Console** или **Windows**. При построении каркаса выполняемого проекта в него включается класс, содержащий статическую процедуру с именем Main(). В результате компиляции такого проекта создается **PE**-файл (PortableExecutablefile) - выполняемый переносимый файл с уточнением **.exe**. Такой **PE**-файл может выполняться только на компьютерах, где установлен **Framework .Net**, поскольку это файл с управляемым кодом.

К невыполняемым проектам относятся, например, проекты типа **DLL -** динамически связываемые библиотеки**,** т.е. проекты, предназначенные для включения(связывания) в другой проект. В результате компиляции такого проекта в сборку войдет файл с уточнением **.dll**. Такие проекты (сборки) непосредственно не могут быть выполнены на компьютере. Они присоединяются к выполняемым сборкам, откуда и вызываются методы классов, размещенных в невыполняемом проекте (**DLL**).

***Этапы реализации проекта*** – можно разделить на несколько шагов:

* создание графического интерфейса проекта (интерфейса пользователя), который должен обеспечить взаимодействие проекта с пользователем при работе проекта;
* установка значений свойств объектов графического интерфейса;
* создание, редактирование и отладка программного кода проекта;
* сохранение проекта для дальнейшего использования.

***Решение*** – это разрабатываемое приложение, файлы которого содержат информации об одном или нескольких проектах. Файл решения используется для управления несколькими взаимосвязанными проектами и имеет расширение **.*sln*.**

Каждый проект, создаваемый в **VS**, помещается в оболочку, называемую ***Решением*** – Solution. Решение может содержать несколько проектов, как правило, связанных общей темой.

Когда создается новый проект, то он может быть помещен в уже существующее Решение или может быть создано новое Решение, содержащее проект.

Решения позволяют придать структуру множеству проектов, что особенно полезно, когда проектов много.

***Сборка*** – результат компиляции проекта. Сборка представляет собой коллекцию из одного или нескольких файлов, помеченных номером версии. Каждая сборка разворачивается на компьютере как единое целое. Программист работает с проектами, **CLR** (общеязыковая среда выполнения - см. Тему 2.2.5 дисциплины Информатика) работает со сборками. Сборка позволяет решать вопросы безопасности, так как содержит описание требуемых ей ресурсов и права доступа к элементам сборки. Каждая сборка содержит манифест, содержащий полное описание сборки, ее элементов, требуемые ресурсы, ссылки на другие сборки, исполняемые файлы. Благодаря этому описанию **CLR** не требуется никакой дополнительной информации для развертывания сборки, трансляции промежуточного кода и его выполнения. Манифест идентифицирует сборку, специфицирует файлы, требуемые для реализации сборки, специфицирует типы и ресурсы, составляющие сборку, задает зависимости, необходимые в период компиляции для связи с другими сборками, специфицирует множество разрешений, необходимых, чтобы сборка могла выполняться на данном компьютере.

***Пространство имен*** – это оболочка, содержащая множество классов, объединенных, как правило, общей тематикой или группой разработчиков. Собственные имена классов внутри пространства имен должны быть уникальны. В разных пространствах могут существовать классы с одинаковыми именами. Полное или уточненное имя класса состоит из уникального имени пространства имен и собственного имени класса. В пространстве имен могут находиться как классы, так и пространства имен.

Пространства имен облегчают независимую разработку проекта большим коллективом разработчиков, каждая группа которого работает в своем пространстве имен.

Пространства имен придают структуру библиотеке **FCL** (общей библиотеке классов для всей платформы **Framework .NET** – т.е. для всех языков см. Тема 2.2.5 и Тема 2.2.6 дисциплины Информатика), содержащей большое число различных пространств имен, объединяющих классы определенной тематики. Центральным пространством имен библиотеки **FCL** является пространство **System**, содержащее другие пространства и классы, имеющие широкое употребление в различных проектах.

Итак, обратите внимание еще раз, что:

***Класс*** – это центральное понятие объектно-ориентированного программирования и языка **VB**. Разработчик проектов на **VB** использует стандартные классы из библиотеки **FCL** и создает собственные классы. У класса две различные роли:

***Класс*** – это модуль – архитектурная единица построения проекта по модульному принципу. Справиться со сложностью большого проекта можно только путем деления его на модули, - сравнительно небольшие единицы, допускающие независимую разработку и последующее объединение в большую систему.

***Класс*** – это тип данных то есть семантическая единица, описывающая свойства и поведение множества объектов, называемых экземплярами класса. Синтаксически класс представляет описание данных, называемых полями класса, описание методов класса и описание событий класса. Для класса, рассматриваемого как тип данных, поля определяют состояние объектов, методы – поведение объектов. События – это некоторые специальные состояния, в которых может находиться объект, и которые могут обрабатываться внешними по отношению к классу обработчиками события. Так, например, объект класса **ЧЕЛОВЕК** может иметь событие «**День рождения**» и каждый из обработчиков этого события может принести объекту свои поздравления по этому случаю.

Как правило, классы **VB** играют обе роли. Но можно определять классы, играющие только роль модуля. Это так называемые статические классы, для которых невозможно создавать объекты. В ходе выполнения программной системы создается единственный экземпляр класса, обеспечивающий доступ к полям и методам этого модуля.

Хороший стиль программирования требует, чтобы каждый класс сохранялся в отдельном файле, имя которого совпадало бы с именем класса. Это требование стиля, которое на практике может и не выдерживаться. В наших примерах будем стараться выдерживать этот стиль.

Определив класс, разработчик получает возможность динамически создавать объекты класса. Для программистов, начинающих работать в объектном стиле, типичной ошибкой является путаница понятий объекта и класса. Нужно с самого начала уяснить разницу. Класс, создаваемый разработчиком, представляет статическое описание множества объектов. Объект – это динамическое понятие, он создается в ходе выполнения программной системы, реально существует в памяти компьютера и обычно исчезает по завершении выполнения проекта. Программист может создать программную систему, включающую два - три класса, но в ходе работы такой системы могут динамически появляться сотни объектов, взаимодействующих друг с другом сложным образом.

### 4.1.2. Главное окно VS

**Запустить *Microsoft Visual Studio 2008***

На рис. 4.1.2-1 показано **Главное окно VS**, каким оно выглядит после запуска.

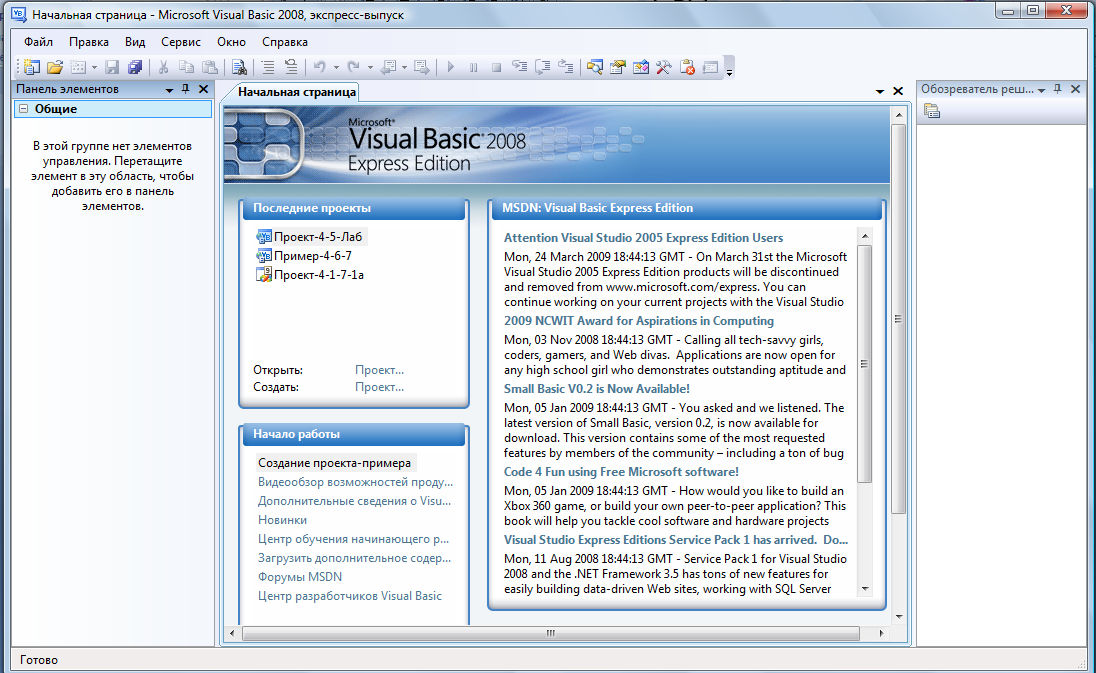


Рис. 4.1.2-1. Главное окно **Visual Studio 2008 Express Edition**

В этом окне можно выделить несколько основных панелей: окно **Стандартная панель инструментов (Standart Toolbars),** окно **Начальная страница (Start Page),** окно **Обозреватель ре­шений (Solution Explorer),** окно **Панель элементов (Toolbox)** и **др.**

**VS** предоставляет в распоряжение пользователя набор разнообразных панелей инструментов. Эти панели инструментов содержат кнопки, назначение которых зависит от функций конкретной панели инструментов. После запуска **VS** на экране всегда отображается **Стандартная панель инструментов (Standart Toolbars).**

Окно **Начальная страница (Start Page)** позволяет открывать использовавшиеся недавно проекты, осуществляет поиск примеров, содержит различные ссылки на сайты, которые могут помочь при работе с **VS.**

В окне **Обозреватель решений (Solution Explorer)** отображаются файлы текущего решения.

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

К средствам, призванным облегчить разработку приложений, относится также окно **Панель элементов (Toolbox),** отображающее элементы управления, используемые в проектах **VB.**

Для создания нового проекта используется диалоговое окно **Создание проекта  
(New Project)** (рис. 4.1.2-2). Для его открытия необходимо выполнить одно из следующих  
действий:

* выбрать ссылку ***Создать: проект...(Create:Project..****.)* в окне **Начальная страница (Start Page)***;*
* выполнить команду ***Создать проект…(New Project…)*** из элемента главного меню **Файл (File);**
* нажать кнопку ***Создать проект (New Project)*,** расположенную на стандартной панели инструментов.

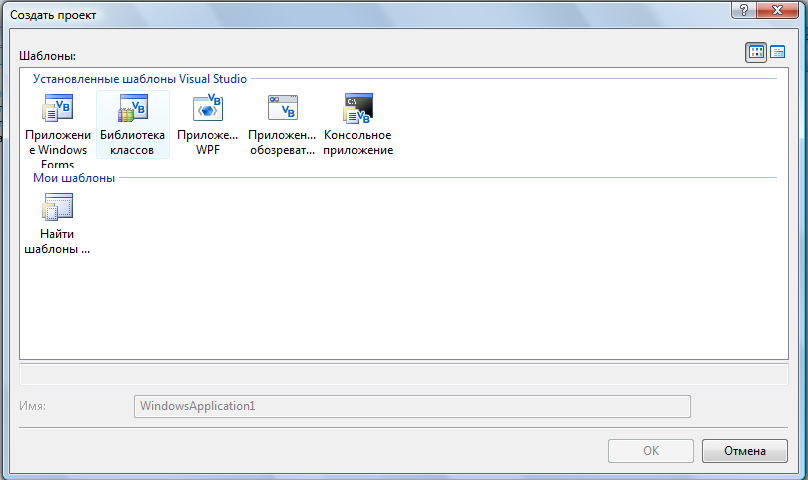


Рис. 4.1.2-2. Диалоговое окно **Создание проекта (New Project)**

В **Visual Basic** можно создавать различные приложения. Кратко охарактеризуем два основных шаблона проектов:

* **Приложение с Windows Forms (Windows Forms Application).** Предназначен для создания традиционного **Windows-приложения** и приложения "**клиент-сервер**", пользовательский интерфейс для которых проектируется с помощью форм **MS Windows.** Для формы можно задавать определенные характеристики и располагать на ней различные элементы управления.
* **Консольное приложение (Console Application).** Создает приложения, которые, как правило, не поддерживают пользовательские графические интерфейсы, и представляет собой отдельный исполняемый файл.

Рассмотрим на примере 4.1.2-1основные шаги создания проекта.

**Пример 4.1.2-1.Создать проект *Пример-4.1.2-1* с оконным графическим интерфейсом, состоящим из формы, на которой имеется одна кнопка.**

Форма проекта ***Пример-4.1.2-1*** может иметь вид как на рис. 4.1.2-3.



Рис. 4.1.2-3.Графический интерфейс (Форма) проекта ***Пример-4.1.2-1***

Написать методы (процедуры), необходимые для выполнения следующего задания: при нажатии на кнопку появляется сообщение **«Привет, МТУСИ!».**

Запустить полученное приложение, убедиться в его способности выполнять поставленную задачу.

Для решения поставленной задачи необходимо:

1. **Запустить VB.**
2. **Создать новый проект:**

* выбрать элемент главного меню **Файл (File)**, а затем выполнить команду ***Создать проект...(New Project…);***
* выбрать шаблон проекта **Приложение с Windows Forms*;***
* ввести в поле ***Имя*** (***Name)*** наименование проекта ***«Пример-4.1.2-1»(***по умолчанию среда **VB** присваивает проекту имя ***WindowsApplication1)****;*
* нажать кнопку ***ОК*;**
* в результате откроется окно **Конструктор форм (Designer)** –конструктор форм, позволяющий реализовать графический интерфейс приложения.

1. **Сохранить созданный проект в «своем» каталоге:**

* выбрать элемент главного меню **Файл (File)**, а затем выполнить команду **Сохранить все (Save All)**;
* в результате откроется диалоговое окно **Сохранить проект (Save Project)**;
* ввести в поле ***Расположение (Location)*** имя каталога, где будет располагаться проект, например ***F:\Информатика-1сем-СС0901-Иванов*** или воспользоваться кнопкой ***Обзор (Browse)*** для поиска необходимого каталога (рис. 4.1.2-4);
* щелкнуть на кнопке **Сохранить**;
* в результате среда разработки примет вид как на рис. 4.1.2-5.

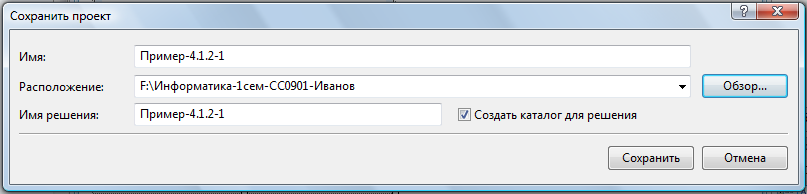


Рис. 4.1.2-4. Диалоговое окно **Сохранить проект (Save Project)**

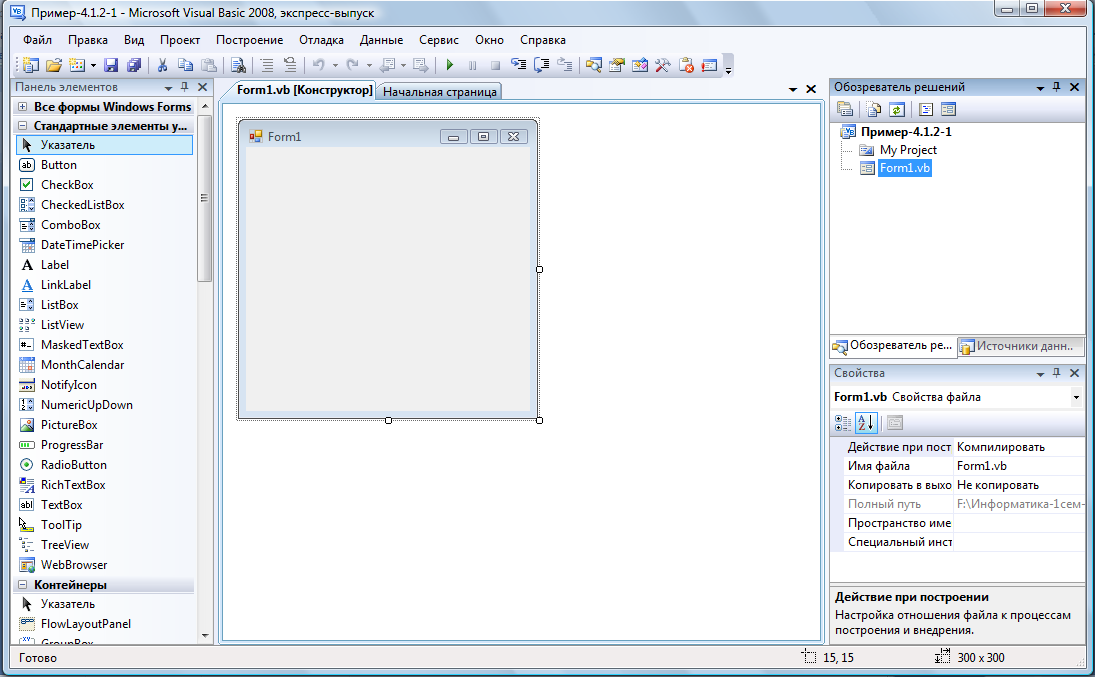


Рис. 4.1.2-5. Среда разработки

1. **Создать графический интерфейс проекта.**

Для создания графического интерфейса проекта на форму необходимо добавить управляющие элементы. Для этого необходимо:

* выбрать элемент главного меню **Вид (View)***,* а затем выполнить команду ***Панель элементов(Toolbox)***

в результате появится окно **Панель элементов (Toolbox)** с набором элементов управления, которые можно разместить на форме;

* выбрать необходимый элемент – например, ***Кнопку (Button)***(рис. 4.1.2-6);
* поместить выбранный элемент на форме – для этого необходимо просто кликнуть в  
  нужном месте формы;
* изменить при необходимости местоположение элемента управления на форме путем ***перетаскивания*** или ***растягивания***;
* изменить свойство ***Text*** элемента управления в окне **Свойства (Properties)** на нужное (рис. 4.1.2-7);
* повторить те же действия для всех необходимых элементов графического интерфейса, в том числе для формы.



Рис. 4.1.2-6. Фрагмент окна **Панель элементов**



Рис. 4.1.2-7. Фрагмент окна **Свойства**

1. **Написать (дописать) необходимый для выполнения задания программный код.**

Для того, чтобы написать код обработки события при нажатии на кнопку, необходимо:

* кликнуть дважды по кнопке – в результате откроется окно **Редактор кода (Code Editor**) – рис. 4.1.2-8.



Рис.4.1.2-8. Фрагмент окна **Редактор кода**

* ввести необходимый программный код обработки события (рис. 4.1.2-9);

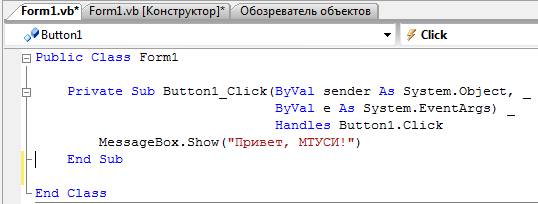


Рис. 4.1.2-9.Фрагмент окна **Редактор кода** с процедурой **Button1\_Click( )**

* вернуться в окно конструктора форм - щелкнув по вкладке ***Form1.vb [Design];***
  + повторить вышеперечисленные действия для остальных кнопок, если они есть.

1. **Нажать <F5> (компиляция и запуск программы производится нажатием<F5>).**

После нажатия на кнопку на экране появится результат работы (рис. 4.1.2-10).



Рис. 4.1.2-10. Результат работы проекта ***Пример 4.1.2-1***

Таким образом, программы **VB** хранятся в ***программных модулях*** в виде процедур (методов). Эти программные модули, являющиеся, по сути, ***контейнерами*** для данных и процедур, могут быть трех видов:

* ***модуль формы;***
* ***модуль класса.***

***В общем виде Приложения могут состоять из нескольких форм, нескольких стандартных модулей и нескольких модулей классов и включать другие конструкции и операторы:***

|  |
| --- |
| Option Explicit on  Option Strict on  Imports System.Math  Imports System.IO  … 'импортирование компонент других библиотек классов  Public Class Class1  … 'модулькласса 1  End Class  …  Public Class Class\_k  … 'модулькласса k  End Class  Public Class Form1  … 'модуль формы 1  End Class  …  Public Class Form\_m  … 'модуль формы m  End Class |

Каждое приложение **VB** должно содержать точку входа. Этой точкой входа является процедура с именем **Main( ).** В **VB** стартовым объектом всегда является процедура **Main()**. Платформа **.NET Framework** вызывает процедуру **Main( )** при загрузке приложения. **Main( )** содержит код, который будет выполняться в первую очередь. В процедуре **Main( )** пользователь может определить, какая форма загружается первой при запуске программы, установить, работают ли в системе копии данного приложения, ввести набор переменных для приложения или открыть необходимую для приложения базу данных. Причем основную работу выполняет метод **Run** - в процессе его вызова создается объект класса **Form1** и открывается форма – визуальный образ объекта, с которой может работать конечный пользователь проекта. Если, как положено, форма спроектирована и заполнена элементами управления, то конечному пользователю остается вводить собственные данные в поля формы, нажимать на кнопки, вообще быть инициатором возникновения различных событий в мире объектов формы. В ответ на возникающие события начинают работать обработчики событий, что приводит к желаемым (или не желанным) изменениям мира объектов. Типичной ситуацией является проведение вычислений по данным, введенным пользователем и отображение результатов этих вычислений в полях формы, предназначенных для этих целей.

Файл, запускаемый самостоятельно с расширением **.exe**, должен также содержать процедуру **Main( ).** В то же время библиотека классов (например, с расширением **\*.dll**) не может быть запущена самостоятельно и не требует процедуры **Main( )**.

Приложения **Windows Forms** выполняются самостоятельно. Для этого **VB** в таких приложениях создает процедуру **Main( )** автоматически, и поэтому ее не требуется создавать. Дело в том, что в **VB** поддерживается концепция стартового объекта, автоматически создаваемого при запуске приложения. По умолчанию стартовым объектом всегда является исходный класс, производный от класса **Form**. Имя этого объекта указывается на вкладке Приложение (Application) диалогового окна MyProject. Программа **VB** построена из стандартных блоков. **Решение** состоит из одного или нескольких проектов. **Проект** в свою очередь может содержать одну или несколько сборок. Каждая **сборка** компилируется из одного или нескольких исходных файлов. **Исходный файл** включает в себя определения и реализацию классов, структур, модулей и интерфейсов, и, в конечном счете, содержит весь код.

***Сборки*** образуют базовую единицу развертывания и принимают форму исполняемого файла (**.exe**) или файла библиотеки динамической компоновки (**DLL**); они являются стандартными блоками платформы **.NET Framework**. Среда **CLR** получает от них сведения, необходимые для реализации типов. Сборку можно представить как коллекцию типов и ресурсов, формирующих логическую единицу функциональности и собранных для совместной работы.

В **VB** используется содержимое сборок, и добавляются ссылки на них. Кроме того, каждая сборка содержит *манифест сборки*. Подобно содержанию манифест сборки содержит ***Идентификатор сборки*** (ее имя и версию); ***Таблицу файлов***, в которой описаны все другие файлы, образующие сборку, например, любые другие сборки, с которыми связаны **\*.exe**-файлы или библиотеки **DLL**, или даже точечные рисунки или текстовые файлы; ***Список ссылок на сборки***, который представляет собой список всех внешних зависимостей – файлов библиотек **DLL** или других необходимых приложению файлов, которые могли быть созданы кем-либо еще.

Поскольку сборки содержат сведения о содержимом, отслеживании версий и зависимостях, то правильность работы приложений, создаваемых с помощью **VB**, не зависит от значений реестра. Сборки уменьшают конфликты библиотек **DLL** и делают приложение более надежным и простым для развертывания.

Как нам уже известно, при создании проекта или файла и открытии редактора кода, пользователю предоставляется уже готовая часть кода, расположенная в правильном порядке. Любой код должен создаваться в следующей последовательности:

1. Операторы Option
2. Операторы Imports
3. Операторы Namespace и элементы уровня пространства имен

Размещение операторов в иной последовательности может вызвать ошибки компиляции.

**Операторы Option и Imports** уже известны

**Операторы пространства имен** помогают организовывать и классифицировать элементы программирования для облегчения их группировки и доступа к ним. [Оператор Namespace](ms-help://MS.VSCC.v90/MS.MSDNQTR.v90.ru/dv_vbalr/html/a31fbd95-9ace-4c3d-bbb1-51222a2272b2.htm) используется для классификации следующих операторов в определенном пространстве имен и объявляет имя пространства имен и вызывает исходный код ниже объявления для компиляции в пределах этого пространства имен:

|  |
| --- |
| Namespace { *name | name.name* }  [ *component types* ]  End Namespace |

Где **name**

Обязательный. Уникальное имя пространства имен. Должно быть допустимым идентификатором **Visual Basic**;

**component types**

Дополнительный. Элементы пространства имен. В пространство имен могут входить структуры, интерфейсы, классы, модули, делегаты, другие пространства имен и прочие элементы;

*End Namespace*

Завершает блок Namespace.

Пространства имен используются в качестве организационной системы. Они предоставляют возможность классификации и представления элементов программирования, которые предоставляются другим программам и приложениям. Обратите внимание, что пространство имен не является типом как класс или структура, поэтому нельзя объявлять элемент программирования с типом данных пространства имен.

Можно использовать инструкцию Namespace только на уровне файла или пространства имен. Это означает, что контекст объявления внутри пространства имен должен быть исходным файлом или другим пространством имен и не может быть классом, структурой, модулем, интерфейсом или процедурой.

Пространства имен обрабатываются так же, как если бы они имели уровень доступа Public. К пространству имен можно обращаться из кода в том же проекте, из других проектов, которые ссылаются на проект, и из любой сборки, построенной из проекта.

Рассмотрим поведение элементов программного кода на различных уровнях:

* Уровень доступа. Элементы программирования, объявленные на уровне пространства имен (то есть имеют смысл в пространстве имен, но не внутри какого-либо другого элемента), могут иметь доступ Public или Friend. Если данный параметр не определен, по умолчанию для элемента используется уровень доступа Friend. Элементы, которые можно объявить на уровне пространства имен, включают классы, структуры, модули, интерфейсы, перечисления и делегаты.
* Компиляция. Все элементы программирования, объявленные после оператора Namespace, относятся к пространству имен. **Visual Basic** продолжает компилировать элементы в последнем объявленном пространстве имен до тех пор, пока он не встречает оператор End Namespace или Namespace.
* Добавление к пространству имен. Если пространство имен уже определено (даже за пределами проекта), к нему можно добавить элементы программирования. Для этого следует использовать оператор Namespace, чтобы дать **Visual Basic** команду компиляции следующих элементов в этом пространстве имен.

В следующем примере определяется схема нового универсального списка класса и она добавляется к пространству имен System.Collections.Generic.

|  |
| --- |
| Namespace System.Collections.Generic  Class specialSortedList(Of T)  Inherits List(Of T)  ' Insert code to define the special generic list class.  End Class  End Namespace |

* Корневое пространство имен. Все имена пространства имен в проекте основаны на корневом пространстве имен. **VS** задает имя проекта как корневое пространство имен по умолчанию для всего кода в проекте. Например, если проект называется Payroll, его элементы программирования относятся к пространству имен Payroll. При объявлении Namespace funding полным именем пространства имен будет являться Payroll.funding.

При необходимости указать существующее пространство имен в операторе Namespace, можно установить корневое пространство имен в значение NULL. Это можно сделать на вкладке ***Приложение (Application)*** диалогового окна ***MyProject*** (рис.5.1.3-7 и рис.5.1.3-7). Если это не было выполнено с базовым списком классов, компилятор **VB** будет обрабатывать System.Collections.Generic в качестве нового пространства имен в пределах проекта Payroll, а полным именем будет являться Payroll.System.Collections.Generic.

Кроме того можно использовать ключевое слово [Global](ms-help://MS.VSCC.v90/MS.MSDNQTR.v90.ru/dv_vbalr/html/18c8ba14-40f6-4978-8096-6a5852324635.htm) для ссылки на элементы пространства имен, определенные вне проекта. Это позволяет сохранить имя проекта в качестве корневого пространства имен. Это уменьшает вероятность случайного слияния элементов программирования с соответствующими им элементами в существующем пространстве имен.

В следующем примере объявляются два пространства имен, причем одно из них вложено в другое.

|  |
| --- |
| Namespace n1  Namespace n2  Class a  ' Insert class definition.  End Class  End Namespace  End Namespace |

В следующем примере выполняется обращение к классу, определенному в предыдущих примерах.

|  |
| --- |
| Dim instance As New n1.n2.a |

Как известно, операторы Option создают основные правила обработки программного кода, предупреждающие синтаксические и логические ошибки

Операторы Imports…, импортируют в программу компоненты библиотек, определяемые в импортируемом пространстве имен.

Кроме того, в программный код проекта могут быть включены операторы описания типов данных с ключевыми словами Dim, Private и Public***,*** находящиеся внутри модулей или процедур; процедуры, генерируемые средой разработки внутри модулей формы; процедуры, написанные программистом, которые могут находиться внутри любых модулей.

Классы, структуры и модули содержат весь код исходного файла. Они являются ***элементами* уровня пространства имен** и могут отображаться в пространстве имен или на уровне исходного файла. Они содержат описания всех других элементов программирования. Интерфейсы, которые определяют подписи элементов, но не выполняют реализацию, также отображаются на уровне модуля.

* [Оператор Class](ms-help://MS.VSCC.v90/MS.MSDNQTR.v90.ru/dv_vbalr/html/f2664f38-eb5a-4d4b-a374-1d041521fb6c.htm)
* [Инструкция Structure](ms-help://MS.VSCC.v90/MS.MSDNQTR.v90.ru/dv_vbalr/html/9bd1deea-2a89-4cdc-812c-6dcbb947c391.htm)
* [Оператор Module](ms-help://MS.VSCC.v90/MS.MSDNQTR.v90.ru/dv_vbalr/html/a1243afc-14a5-45df-95d5-51118aeac362.htm)
* [Оператор Interface](ms-help://MS.VSCC.v90/MS.MSDNQTR.v90.ru/dv_vbalr/html/8997af73-bda3-4f79-bd41-ca396b610260.htm)

Элементы данных на уровне пространства имен являются перечислениями и делегатами.

**Элементы программирования уровня модуля**

Процедуры, операторы, свойства и события являются единственными элементами программирования, которые могут содержать исполняемый код (то есть выполняются во время выполнения). Они являются элементами уровня модуля программы

* [Оператор Function](ms-help://MS.VSCC.v90/MS.MSDNQTR.v90.ru/dv_vbalr/html/a4497077-0f46-4ede-a27f-9e8670df52b9.htm)
* [Оператор Sub](ms-help://MS.VSCC.v90/MS.MSDNQTR.v90.ru/dv_vbalr/html/e347d700-d06c-405b-b302-e9b1edb57dfc.htm)
* [Оператор Declare](ms-help://MS.VSCC.v90/MS.MSDNQTR.v90.ru/dv_vbalr/html/d3f21fb0-b804-4c99-97ed-583b23894cf1.htm)
* [Оператор Operator](ms-help://MS.VSCC.v90/MS.MSDNQTR.v90.ru/dv_vbalr/html/b12ec4af-1ad7-4a17-865b-c5ee96320ae5.htm)
* [Оператор Property](ms-help://MS.VSCC.v90/MS.MSDNQTR.v90.ru/dv_vbalr/html/3155edaf-8ebd-45c6-9cef-11d5d2dc8d38.htm)
* [Оператор Event](ms-help://MS.VSCC.v90/MS.MSDNQTR.v90.ru/dv_vbalr/html/306ff8ed-74dd-4b6a-bd2f-e91b17474042.htm)

Элементами данных на уровне модуля являются переменные, константы, перечисления и делегаты.

**Элементы программирования уровня процедуры**

К элементам уровня процедуры относятся в основном исполняемые операторы, которые составляют код времени выполнения программы. Весь исполняемый код должен находиться в пределах некоторой процедуры (Function, Sub, Get, Set, AddHandler, RemoveHandler, RaiseEvent).

Элементы данных на уровне процедуры ограничиваются локальными переменными и константами.

По умолчанию переменные ***локальны***внутри ***процедур***, т.е. переменные, которые описаны внутри процедур, могут быть использованы и изменены только в той процедуре, в которой они ***созданы (объявлены)***. Также можно объявить переменные в верхней части программного кода формы, до описания процедур пользователей и процедур событий, и тем самым при­дать переменным более широкую область видимости, распространяющу­юся на всю форму. Однако, при создании в проекте нескольких форм, переменные, объявленные в верхней части формы, будут доступны толь­ко в той форме, в которой они объявлены. Точно так же, процедуры по умолчанию объявляются как ***частные*** (Private***)*** и являются локальными для той формы, в которой они созданы. Например, нельзя из второй формы с именем Form2вызвать процедуру события Button1\_Click(), если она объявлена в Form1 как Private.

Чтобы сделать переменные и процедуры доступными для всех форм и процедур проекта, их следует объявить в одном или несколь­ких стандартных модулях, включенных в проект. Все вышесказанное касается и модулей классов.

Таким образом, чтобы эффективно управлять элементами, используемыми на этапе разработки, например ссылками, подключениями данных, папками и файлами, в **VS** предусмотрены два типа ***контейнеров***. Эти контейнеры называются ***решениями*** и ***проектами***. Также, **VS** предоставляет папки решений для того, чтобы структурировать связанные проекты по группам и затем выполнять действия над этими группами проектов. Частью интегрированной среды разработки (**IDE**) является интерфейс для просмотра и управления этими контейнерами и связанными с ними элементами – Обозреватель решений.

Решения и проекты содержат элементы, которые представляют ссылки, подключения к данным, папки и файлы, необходимые для создания приложения. Решение может содержать несколько проектов, тогда как проект обычно содержит несколько элементов. Эти контейнеры позволяют воспользоваться преимуществами интегрированной среды разработки, следующими способами:

* возможностью управления параметрами для решения в целом или для отдельных проектов
* возможностью использования Обозревателя решений для управления параметрами управлениями файлами при более тщательной работе с элементами, необходимыми для разработки приложения
* возможностью добавления необходимых элементов к нескольким проектам в решении или к решению без создания ссылки на эти элементы в каждом проекте
* возможностью работы с несколькими файлами, которые независимы от решений или проектов

Элементы могут представлять собой файлы и другие составляющие проекта, например ссылки, подключения к данным или папки. В Обозревателе решений элементы можно сгруппировать следующими способами:

* В виде элементов проекта, которые являются элементами, составляющими данный проект, например формами, исходными файлами и классами в пределах проекта в Обозревателе решений. Группировки и отображение зависят от выбранного шаблона проекта и от внесенных изменений.
* В виде элементов решения для файлов, применимых к данному решению в целом, в папке ***Элементы решения*** в Обозревателе решений.
* В виде прочих файлов для файлов, не связанных с проектом или решением, которые могут отображаться в папке Прочие файлы.

### 4.1.3. Главное меню VS 2008

При работе с **VB** в интегрированной среде разработки **VS**можно пользоваться как кнопками стандартной панели инструментов, так и элементами главного меню, расположенными в верхней части экрана (4.1.3-1).



Рис. 4.1.3-1.Стандартная панель инструментов

Главное меню может содержать следующие элементы: **Файл (File), Правка (Edit), Вид (View), Проект (Project), Построение (Build), Отладка (Debug), Данные (Data), Сервис (Tools), Окно (Window)** и **Справка (Help).**

Первоначально при запуске программы в меню присутствуют лишь некоторые из указанных элементов. Остальные элементы добавляются в меню при открытии дополнительных окон. Например, при открытии проекта в меню добавляются пункты **Проект (Project), Построение (Build), Отладка (Debug).** Настроить отображение элементов главного меню можно с помощью диалогового окна **Настроить(Customize),** которое открывается командой ***Настроить (Customize)***элемента главного меню **Tools(Сервис).**

Элемент главного меню **Файл (File)**содержит команды, связанные с доступом к файлам(4.1.3-2).

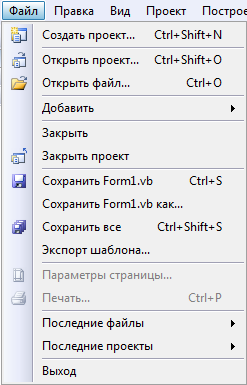


Рис. 4.1.3-2. Команды элемента главного меню **Файл**

Команды элемента главного меню **Правка (Edit)** используются при создании форм и редактировании программного кода (рис. 4.1.3-3).

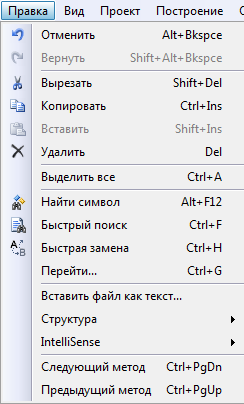


Рис. 4.1.3-3. Команды элемента главного меню **Правка**

С помощью команды ***Технология IntelliSense*** реализуются возможности технологии

***IntelliSense,*** а именно: можно просматривать список членов определенного класса, структуры, объединения или пространства имен и вставлять в программный код подходящий, получить информацию о числе, типах и именах параметров методов и свойств, дописать слова, являющиеся именами методов, команд (рис. 4.1.3-4).

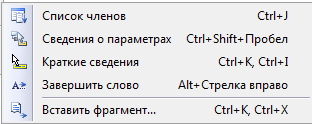


Рис. 4.1.3-4. Элементы команды ***Технология IntelliSense***

Элемент главного меню **Вид (View)**содержит команды вызова окон среды **VS .** С помощью этих команд могут открываться окна редактора программного кода, конструктора формы, свойств объектов, обозревателя решений и другие окна.

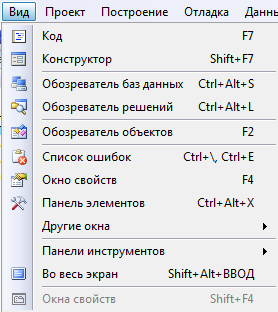


Рис. 4.1.3-5.Команды элемента главного меню **Вид**

Элемент главного меню **Проект (Project)**содержит команды, позволяющие добавлять в проект и удалять из него такие элементы, как форма, программный модуль, класс, а также команды, дающие возможность добавлять ссылки на подключаемые библиотеки (рис. 4.1.3-6).

Последней командой элемента меню **Проект(Project)**является команда ***Свойства: ... (…Properties...),*** позволяющая открыть окно свойств проекта.

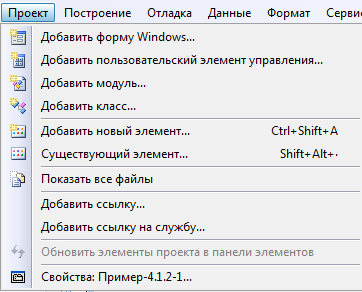


Рис. 4.1.3-6.Команды элемента главного меню **Проект**

Элемент главного меню **Построение (Build)**содержит команды, помогающие скомпоновать решение или проект (4.1.3-7).

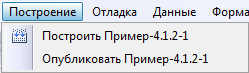


Рис. 4.1.3-7.Команды элемента главного меню **Построение**

Элемент главного меню **Отладка (Debug)**содержит команды, предназначенные для отладки и запуска приложения (рис. 4.1.3-8). С помощью команд этого меню можно запустить приложение на выполнение, установить точки останова программы, осуществить пошаговое вы­полнение приложения, открыть специальные окна для отладки.

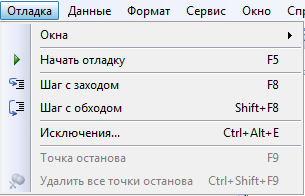


Рис. 4.1.3-8. Команды элемента главного меню **Отладка**

Элемент главного меню **Формат (Format)**доступен при работе в конструкторе форм (рис. 4.1.3-9).

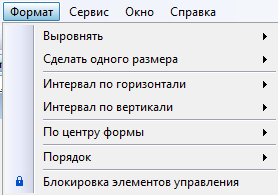


Рис. 4.1.3-9. Команды элемента главного меню **Формат**

Он содержит команды, управляющие выравниванием текста и объектов, заданием размеров объектов и определением интервалов между ними. Однако, при работе с различными конструкторами становятся доступными и дополнительные команды. Внизу этого меню располагается команда ***Блокировка элементов управления (LockControls),*** которая позволяет сделать недоступным перемещение элементов управления и сохранить их размер. С ее помощью фиксируются все элементы управления, включая саму форму.

Элемент главного меню **Сервис (Tools)**содержит средства для настройки среды разработки, создания макросов, а также команды запуска дополнительных утилит (рис. 4.1.3-10).

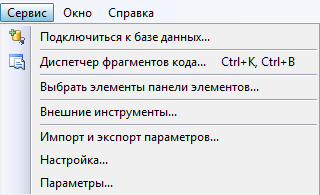


Рис. 4.1.3-10.Команды элемента главного меню **Сервис**

Элемент главного меню **Окно (Window)** содержит команды, которые управляют открытыми на экране окнами (рис. 4.1.3-11). С помощью этих команд можно упорядочивать, скрывать окна и переходить из одного окна в другое. Кроме того, команды данного меню позволяют активизировать любое открытое окно.

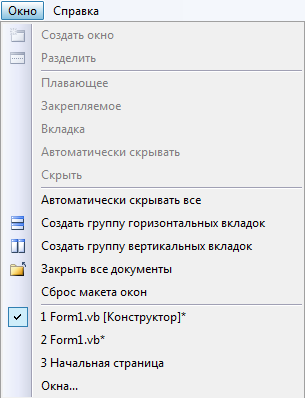


Рис. 4.1.3-11.Команды элемента главного меню **Окно**

Элемент главного меню **Справка (Help)**использует свои команды для вызова справочной системы с различными вариантами представления информации (рис. 4.1.3-12).

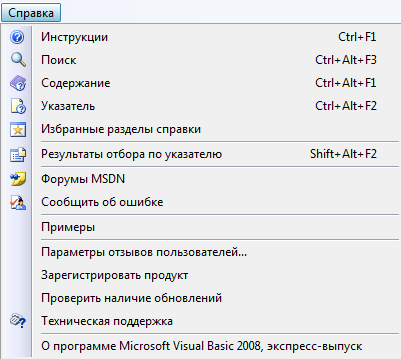


Рис. 4.1.3-12.Команды элемента главного меню **Справка**

### 4.1.4. Стандартная панель инструментов

**В VB** содержится большое количество панелей инструментов для отладки и запуска программ, задания расположения элементов на форме и многого другого. Познакомимся со стандартной панелью инструментов (рис. 4.1.4-1), которая используется во всех режимах работы.



Рис. 4.1.4-1. Стандартная панель инструментов

Назначение кнопок стандартной панели инструментов описано в табл. 4.1.4-1.

По умолчанию в главном окне программы **Visual Basic** всегда присутствует стандартная панель инструментов, если только вы не удалили ее с экрана. Если для работы необходима стандартная панель инструментов, выберите команду ***Стандартная (Standard)***элемента меню **Панели инструментов (Toolbars)** из элемента главного меню **Вид (View).**

Таблица 4.1.4-1

|  |
| --- |
|  |

### 4.1.5. Окна панелей VS

Окно панели **Начальная страница (Start Page)**(рис. 4.1.1-1) позволяет просмотреть последние использовавшиеся проекты, осуществить поиск примеров программ, как из справки, так и Интернета, а также просмотреть ссылки на сайты, содержащие новости о продукте **VS,** документацию, учебные пособия.

Начальная страница автоматически открывается при запуске **VS.** Если же окно **Начальная страница (Start Page)**не появилось, его можно вызвать с помощью команды **Начальная страница (Start Page)**элемента главного меню **Вид(View).**

Окно панели **Конструктор (Designer)**является основным рабочим окном, в котором выполняется визуальное проектирование приложения (рис. 4.1.5-1). Вызвать это окно можно из главного меню командой ***Конструктор(Designed)*** элемента главного меню **Вид (View)**или двойным щелчком на названии формы в окне **Обозреватель решений (Solution Explorer).**

В Окне конструктора форм визуально создаются все формы приложения с использованием инструментария среды разработки. Для точного позиционирования объектов на форме в окне можно использовать сетку.

Размер формы в окне можно изменять, используя маркеры выделения формы и мышь. Для изменения размера формы необходимо установить указатель мыши на маркер и, когда он примет вид двунаправленной стрелки, перемещать до требуемого размера.

Для работы в окне **Конструктор** можно использовать контекстное меню   
(рис. 4.1.5-1).

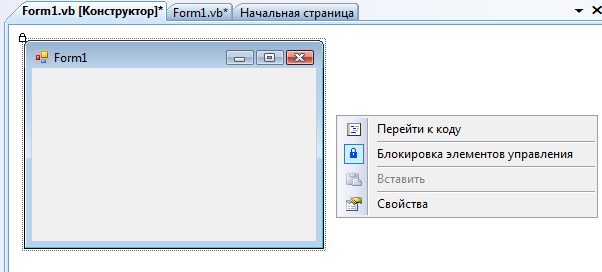


Рис. 4.1.5-1.Окно панели **Конструктор форм** и ее контекстное меню

Окно панели **Редактор кода (Code Editor)** –это мощный текстовый редактор с большим количеством возможностей, являющийся основным инструментом программиста для создания и отладки приложения.

В окне панели **Редактор кода (Code Editor)** (рис.4.1.5-2)расположены элементы:

* раскрывающийся список ***Имя класса (Class Name)*** содержит перечень объектов приложения; при выборе объекта в этом списке содержимое списка ***Имя метода (Method Name)*** изменяется;
* раскрывающийся список ***Имя Метода (Method Name)*** дает возможность выбора членов объекта (событий) и автоматического вывода в окно редактора процедуры или шаблона для выбранного члена.

Для работы в окне редактора можно использовать контекстное меню (4.1.5-2).

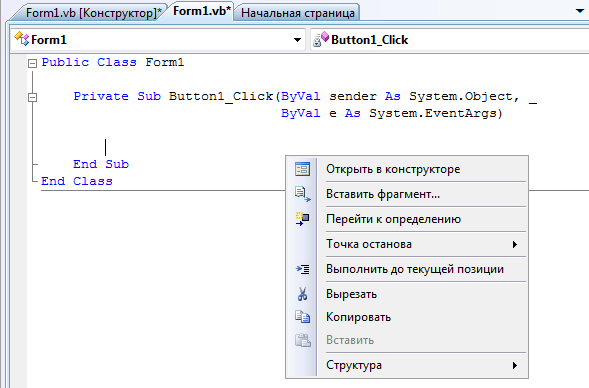


Рис. 4.1.5-2.Окно панели **Редактор кода** и ее контекстное меню

Окно панели **Обозреватель решений (Solution Explorer)**позволяет получить доступ к компонентам, входящим в проект (4.1.5-3). Если после создания или открытия проекта этого окна нет на экране, то можно его отобразить одним из следующих способов:

* в элементе главного меню **Вид (View)** выбрать команду ***Обозреватель решений (Solution Explorer);***
* нажать кнопку ***Обозреватель решений (Solution Explorer)*** стандартной панели инструментов;
* нажать комбинацию клавиш **<Ctrl>+<Alt>+<L>.**

Окно панели **Обозреватель решений (Solution Explorer)** содержит список всех компонентов решения (проекта). В верхней части окна размещается имя решения, а ниже располагаются входящие в него проекты и файлы. Также окно панели **Обозреватель решений (Solution Explorer)** содержит кнопки, отображение которых зависит от типа, выделенного в окне компонента.

Для работы в окне **Обозреватель решений (Solution Explorer)** можно пользовать контекстное меню (4.1.5-3).

Окно панели **Панель элементов (Toolbox)** - основной рабочий инструмент при визуальной разработке форм приложения (рис.4.1.5-4). Панель элементов управления вызывается из главного меню **Вид (View)** командой ***Панель элементов (Toolbox)*** или нажатием кнопки ***Панель элементов (Toolbox)*** на стандартной панели инструментов.

Панель элементов управления состоит из различных разделов, в которых расположены используемые в проектах элементы. В табл. 4.1.5-1 описаны основные разделы окна **Панель элементов (Toolbox).**

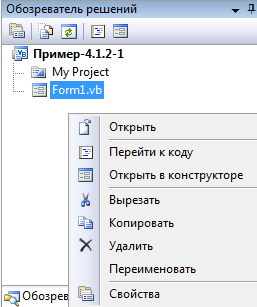


Рис. 4.1.5-3.Окно панели **Обозреватель решений** и ее контекстное меню

Таблица 4.1.5-1

|  |  |
| --- | --- |
| **Раздел** | **Описание** |
| ***Все формы***  ***(All Windows Forms)*** | В этом разделе располагаются все элементы управления формы из указанных ниже разделов. |
| ***Обычные элементы управления***  ***(Common Controls)*** | Хранит основные элементы управления, используемые для построения интерфейса.пользователя |
| ***Контейнеры***  ***(Containers)*** | В этом разделе хранятся элементы, которые могут содержать другие объекты. |
| ***Меню и панели инструментов***  ***(Menus &Toolbars)*** | Содержит такие элементы управления, как обычное и контекстное меню, строка состояния и панель инструментов. |
| ***Данные***  ***(Data)*** | Содержит компоненты, с помощью которых можно получить доступ к данным и источникам данных. |
| ***Компоненты***  ***(Components)*** | Хранит элементы, посредством которых можно выполнить мониторинг файловой системы. |
| ***Печать***  ***(Printing)*** | В этом разделе содержатся элементы, которые используются для организации печати. |
| ***Диалоговые***  ***(Dialogs)*** | Содержит список стандартных диалоговых окон: окна открытия и сохранения файла, настройки шрифтов текста и цветовой палитры. |
| ***Общие***  ***(General)*** | В этом разделе могут располагаться стандартные элементы управления проекта и специальные управляющие элементы. |

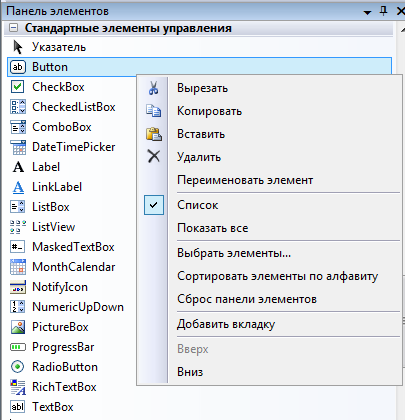


Рис. 4.1.5-4.Окно панели **Панель элементов** и ее контекстное меню

Окно панели **Свойства (Properties)**предназначено для отображения и настройки свойств объектов проекта, включая форму и размещенные в ней объекты. В этом окне, например, содержатся такие свойства выбранного объекта, как позиция в форме, высота, ширина, цвет (рис. 4.1.5-5).

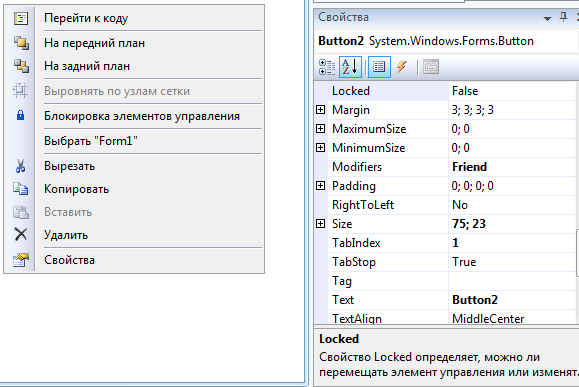


Рис. 4.1.5-5. Окно панели **Свойства** и ее контекстное меню

Для открытия диалогового окна **Свойства (Properties**) следует выполнить одно из следующих действий:

* в элементе главного меню **Вид (View)** выбрать команду ***Окно свойств (Properties Window);***
* нажать кнопку ***Окно свойств (Properties Window),*** расположенную на стандартной панели инструментов;
* выбрать команду ***Свойства (Properties)*** контекстного меню выделенного объекта;
* нажать клавишу **<F4>.**

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

В нижней части окна появляется подсказка, поясняющая назначение выбранного  
свойства объекта. Более подробное пояснение можно посмотреть в справочной системе. Также можно воспользоваться динамической справкой, выбрав нужный раздел в окне **Динамическая справка (Dynamic Help).**

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

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

Окно панели **Обозреватель объектов (Object Browser)** предназначено **д**ля просмотра всех объектов, входящих в состав проекта**.** В этом окне (рис. 4.1.5-6) можно получить доступ не только ко всем входящим в проект объектам, но и их свойствам, методам, событиям. Окно просмотра объектов обычно не визуализировано, и его можно вызвать командой ***Обозреватель объектов (Object Browser)*** из меню **Вид (View).**

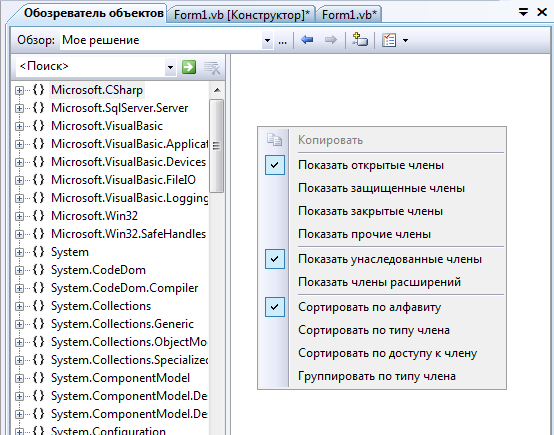


Рис. 4.1.5-6.Окно панели **Обозреватель объектов** и ее контекстное меню

С помощью раскрывающегося списка ***Обзор(Browse) (***рис. 4.1.5-7***)*** окна **Обозреватель объектов (Object Browser)** задается область просмотра.

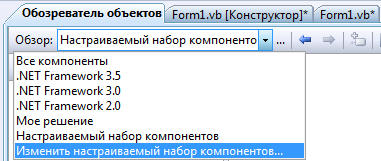


Рис. 4.1.5-7. Раскрывающийся список ***Обзор***

Этот список может содержать сле­дующие значения:

* ***Все компоненты (All Components) -*** в окне **Object Browser (Обозреватель  
  объектов)** будут отображаться все имеющиеся компоненты;
* ***Компоненты платформы .NET Framework (.NET Framework)*** - окно просмотра  
  объектов будет содержать все библиотеки классов платформы **.NET Framework;**
* ***Мое решение (My Solution)*** – в окне **Object Browser  
  (Обозреватель объектов)** будут отображаться содержащиеся в открытом решении пространства имен, классы, структуры, интерфейсы, типы и их свойства, методы, события, переменные, константы;
* ***Настраиваемый набор компонент (Custom Component Set)*** - в окне  
  просмотра объектов будет отображаться содержимое компонентов, указанных в диалоговом окне ***Изменить настраиваемый объект компонентов (Edit Custom Component Set)*,** которое открывается с помощью расположенной справа от списка кнопки ***Добавить*** (***Add).* В**качестве компонентов могут выступать проекты решения, **СОМ-объекты,** внешние библиотеки и исполняемые файлы.

Для задания типа отображаемых компонентов предназначена кнопка ***Обозреватель объектов: параметры(Object Browser Settings),*** при нажатии на которую открывается меню со списком типов компонентов: пространства имен, контейнеры, базовые, производные и скрытые типы, открытые, защищенные и скрытые члены классов. При выборе типа слева от его наименования появляется галочка.

Для работы в окне **Обозреватель объектов (Object Browser)** можно использовать контекстное меню (4.1.5-6).

Окно панели **Локальные (Locals)** предназначено **д**ля просмотра списка локальных переменных приложения и контроля их значений (рис. 4.1.5-8). Вызвать его можно подкомандой ***Локальные (Locals)*** команды ***Окна (Windows)*** элемента основного меню **Отладка (Debug)**или из панели элементов **Отладка (Debug).**

В окне **Локальные (Locals)**удобно просматривать имена локальных переменных, объявленных в текущей процедуре, тип и значения этих переменных. Указанная информация появляется в окне автоматически при его вызове. Данное окно используется для отладки и проверки работы приложения, однако глобальные переменные для просмотра в нем недоступны.

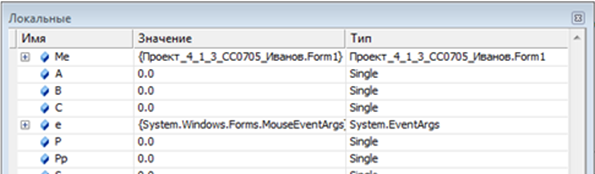


Рис. 4.1.5-8. Окно панели **Локальные**

Окно панели **Видимые (Watch)** предназначено для более полного контроля работы приложения (рис. 4.1.5-9). Это окно вызывается подкомандой ***Видимые (Watch)*** команды ***Окна (Windows)*** элемента основного меню **Отладка (Debug)**или из панели элементов **Отладка (Debug).** Оно предназначено для определения значений выражений и их изменения.



Рис. 4.1.5-9. Окно панели **Видимые**

### 4.1.6. Настройка параметров интегрированной среды разработки

Окна инструментов, справочная система, параметры компилятора среды разработки **VS** легко и очень гибко настраиваются.

Следующие шаги показывают, как изменить параметры среды разработки на **VB** в соответствии с рекомендациями для выполнения практических работ:

1. В элементе главного меню **Сервис (Tools)** выполнить команду ***Импорт и экспорт параметров...(Import and Export Settings).*** Для сохранения настроек среды с целью их использования на другом компьютере, загрузки настроек с другого компьютера или сброса настроек – опции, которую требуется выбрать сейчас – можно исполь­зовать появляющийся при этом мастер.
2. Выбрать ***Сбросить все настройки (Reset ALL Options)****,* а затем щелкнуть на кнопку ***Далее (Next****).* **VS** спросит, необходимо ли перед настройкой сохранить текущие настройки в файле. Следует всегда сохранять резервную копию своих текущих настроек с тем, чтобы была возможность вернуться к ним, если новые настройки окажутся «неудобными».
3. Проверить, что выбрана опция ***Да****,* и обратить внимание на ***имя файла*** и ***папки****,* в которых **VS**собирается сохранить эти настройки.

Если необходимо вернуться к этим настройкам, для их восстановления используется этот же мастер и опция ***Импорт выбранных настроек среды (Import Selected Environmental Settings).***

1. Чтобы увидеть список по умолчанию параметров, используемых для **VS,** следует щелкнуть на кнопке ***Далее (Next).***

Вы увидите список опций.

1. Щелкнуть на ***Параметры для разработки на Visual Basic (Visual Basic Development Settings),*** а за теми щелкнуть на кнопке ***Готово (Finish).***

Мастер переключит параметры **IDE,** включая команды меню, панели инструментов и настройки некоторых диалоговых окон, окон инструментов и **Редактор кода (Code Editor).**

Можно повторить этот процесс настройки в любой момент, когда потребуется сбросить текущие параметры (например, если вы сделали ошибку в настройках), или если потребуется настроить **VS 2008**для использования другого инструмента программирования.

1. Чтобы закрыть мастер, щелкнуть на кнопке ***Закрыть (Close).***

Если параметры среды для работы с **Visual Basic** настроены, то можно выполнять задания. Но если они были изменены, необходимо выполнить следующие действия, позволяющие проверить, что настройки, связанные с проектами и компилятором **VB** соответствуют тем, которые используются при проведении практических работ. Для проверки параметров проекта и компилятора необходимы следующие действия:

1. Щелкнуть на команде ***Параметры (Options)*** элемента меню **Сервис (Tools)** и открыть диалоговое окно **Параметры (Options).**

Диалоговое окно **Параметры (Options)**– это окно, в котором находятся многие из настроечных параметров **VS**. Чтобы увидеть все параметры, которые можно изменить, надо щелкнуть на флажке ***Показать все параметры(Show All Settings)***, расположенном в нижнем левом углу диалогового окна, чтобы установить его.

1. В диалоговом окне **Параметры (Options)** следует щелкнуть на категории ***Проекты и решения (Projects And Solutions)*** и затем на разделе **Общие (General).** Эта группа флажков и опций настраивает параметры проектов и решений **Visual Studio**. Чтобы программное обеспечение соответствовало параметрам, использованным в этом пособии необходимо настроить их так, как показано на рис. 4.1.6-1.

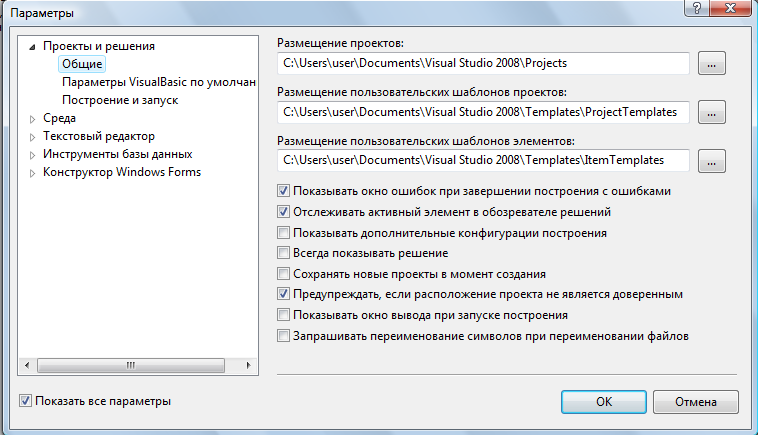


Рис. 4.1.6-1. Диалоговое окно **Параметры (Options)**

Чтобы инструкции, связанные с открытием проектов, соответствовали этому пособию, снимите или установите флажки с этих опций, в соответствии с рис. 4.1.6-1. В частности, рекомендуется снять флажки для опций ***Всегда показывать решение  
 (Always Show-Solution)***  и ***Сохранять новые проекты в момент создания (Save New Projects When Created)***. Первая опция создает вторую папку для файла решения программы, что не требуется для решений, которые содержат только один проект. Вторая опция приводит к тому, что **VS** откладывает сохранение проекта до того момента, когда вы щелкните на команде ***Сохранить все (SaveAll)*** в элементе меню **Файл (File)** и укажете место для сохранения. Это «***отложенное***» сохранение позволяет создавать тестовые программы, компилировать и отлаживать программы, и даже запускать их, не сохраняя проект на диск, что может оказаться полезным, если вы хотите быстро создать тестовую программу, которую по окончании ее использования можно удалить, не сохраняя. С этой опцией по умолчанию система будет просить сохранить проекты после их создания, хотя можно сохранять проекты заранее, установив флажок ***Сохранять новые проекты в момент создания (Save New Projects When Created).***

1. В диалоговом окне **Параметры (Options)** следует щелкнуть на элементе **VB** по умолчанию (**VB Defaults**) команды ***Проекты и Решения (ProjectandSolution)***.   
   ***VS*** отобразит список из трех параметров компилятора: ***Option Explicit, Option Strict u Option Compare.***
2. Студенты должны проверить, что опция ***Option Explicit*** установлена в значение ***On***, a ***Option Strict*** - в значение ***Off***- значения по умолчанию для программирования на **Visual Basic** в **Visual Studio**. ***Option Explicit On*** - это параметр, который требует явного объявления переменной до ее первого использования в программе - для хорошего стиля программирования, придерживаться которого настоятельно рекомендуется при изучении дисциплины **Информатика**. ***Option Strict Off*** позволяет при определенных обстоятельствах объединять переменные и объекты различных типов, не вызывая при этом ошибок компилятора. (Например, число может быть присвоено объекту текстового поля без возникновения ошибки.) Хотя это потенциально опасная практика программирования, ***Option Strict Off*** очень полезна для определенных типов демонстрационных программ.
3. Можно изучить другие параметры диалогового окна **Options**, связанные со средой  
   программирования и **VB**. Когда закончите, щелкните на ***ОК***, чтобы закрыть диалоговое окно **Параметры (Options).**

### 4.1.7. Пример создания проекта

1. ***Задание на разработку проекта Проект-4.1.7-1***

Создать приложение на **VB**с именем ***Проект-4.1.7-1,*** которое должно имитировать работу ***игрового автомата***, генерирующего три счастливых числа. Если хотя бы одно из чисел равно **7**, то считается, что играющий выиграл [8].

***Проект-4.1.7-1***должен иметь простой интерфейс пользователя (рис. 4.1.7-1):

* две ***Кнопки*** управления;
* три ***Поля,*** отображающие случайные числа;
* один ***Рисунок,*** отмечающий выигрыш;
* один ***Текст* «Проект-4.l.7-1».**

Получить эти элементы можно, создав на ***Форме*** проекта ***Проект-4.1.7-1***семь объектов, а затем изменить некоторые из свойств каждого объекта.

После создания интерфейса добавить код программы *для* кнопок ***Играть*** и ***Конец игры,*** который обрабатывает щелчки пользователя на этих кнопках.

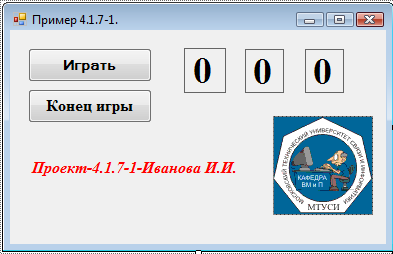


Рис. 4.1.7-1.Интерфейс пользователя проекта **Проект-4.1.7-1**

1. ***Разработка приложения***
2. **Создание нового проекта:**
3. Запустить **VS**.
4. Выполнить команду ***Создать проект (NewProject)*** элемента главного меню ***Файл (File***).
5. На экране появится диалоговое окно **Создание проекта (NewProject).**
6. Выбрать шаблон ***WindowsApplication*** для типа проектов **Visual BasicWindows** в диалоговом окне **Создание проекта (New Project).**  
   Ввести в текстовом поле ***Имя (Name)*** нижней части диалогового окна **Создание проекта (New Project)**  имя проекта ***Проект-4.1.7-1***.
7. Щелкнуть на кнопке **ОК** диалогового окна ***Создание проекта (New Project).***

**VS** присвоит проекту имя ***Проект-4.1.7-1*** и отобразит основное окно   
**VS** с различными панелями, среди которых присутствует окно **Конструктора форм (Designer)** с пустой формой *Форма1 (Form1).* Эта форма будет использоваться для создания интерфейса пользователя.

1. **Создание интерфейса пользователя:**
2. Изменить размер формы, для этого необходимо поместить указатель мыши над нижним правым углом формы так, чтобы он изменился и принял форму указателя изменения размера, а затем перетащить этот угол так, чтобы увеличить размер формы для размещения на ней объектов проекта. При изменении размера формы в окне **Конструктора форм (Designer)** могут появиться полосы прокрутки, которые позволяют получить доступ ко всей созданной форме.

Задать размер формы, примерно как показано на рис. 4.1.7-2.

Чтобы увидеть всю форму без помех, если это необходимо, можно изменить размер других панелей инструментов или закрыть их.

1. Добавить на форму объект *Кнопка1*, для чего дважды щелкнуть в окне панели **Панель элементов (ToolBox)** на ***Button (Кнопка)***. **VS 2005** создаст на форме объект кнопки с размерами по умолчанию и именем *Button1*. Причем объект новой кнопки выделен и, вокруг него имеются манипуляторы изменения размера.

Необходимо обратить внимание на это название кнопки - оно встретится при написании кода программы. Когда **VB** находится в режиме конструктора, объекты на форме перемещаются с помощью перетаскивания их мышью, а их размеры изме­няются с помощью манипуляторов изменения размера.

Необходимо попрактиковаться в перемещении и изменении размера кнопки.

1. Добавить на форму второй объект *Кнопка2*, для чего необходимо щелкнуть (теперь только один раз) в окне панели **Панель элементов (ToolBox)** на объекте ***Button***, а затем переместить указатель мыши в область формы *Form1*. Указатель мыши примет форму перекрестия с пиктограммой кнопки. Перекрестие предназначено для того, чтобы нарисовать прямоугольную фигуру кнопки на форме, и этот метод используется как альтернатива двойному щелчку, создающему элемент управления с размером по умолчанию.

Перетащить указатель вниз и вправо. Чтобы завершить образование второй кнопки, необходимо отпустить кнопку и посмотреть на ее привязку к форме.

Изменить размер объекта кнопки так, чтобы она стала такого же размера, как и первая кнопка, а затем переместить ее на форме, расположив ниже *Кнопка1*. В любой момент можно удалить объект и начать заново - выбрав этот объект на форме, а затем нажав на клавишу **<Del>.**

Попрактикуйтесь в создании и удалении объектов интерфейса пользователя. Форма должна выглядеть примерно так, как показано рис. 4.1.7-2.



Рис. 4.1.7-2. Форма *Form1* с объектами *Button1* и *Button2*

1. Добавить на форму числовые надписи, для чего дважды щелкнуть мышью в окне панели **Панель элементов (ToolBox)** на элементе управления ***Надпись (Label).***

**VS**  создаст на форме экземпляр объекта *Label1*.

Необходимо перетащить элемент *Label1* вниз от двух объектов кнопок.

1. Создать второй объект надписи, дважды щелкнув мышью в окне панели **Панель элементов (ToolBox)** на элементе управления ***Надпись (Label).* VS** создаст на форме элемент *Label2*.
2. Создать третий объект надписи *Label3*, снова дважды щелкнув мышью на элементе управления ***Label,*** чтобы создать третий объект надписи *Label3*.
3. Создать четвертый объект надписи *Label4* таким же образом.
4. Переместить третий и четвертый объекты надписей вправо от второго, оставив между этими тремя надписями небольшое расстояние, так как при работе программы они будут использоваться для отображения больших цифр.

Форма будет иметь вид примерно такой, как показано на рис. 4.1.7-3. Если объекты надписей выглядят не вполне правильно, то вы можете переместить их.

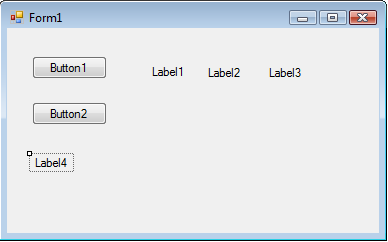


Рис. 4.1.7-3.Форма *Form1* с элементами управления ***Button*** и ***Label***

1. Добавить на форму рисунок, для этого необходимо щелкнуть в окне панели  
   **Панель элементов (ToolBox)** на элементе управления ***Изображение (PictureBox.)***

Используя Указатель рисования элемента управления, необходимо создать на форме большой прямоугольник, расположенный ниже второй и третьей надписей.

Область рисунка должна выглядеть примерно так, как на рис. 4.1.7-4.

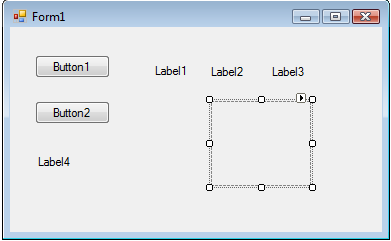


Рис. 4.1.7-4.Форма *Form1* с элементами управления ***Button***,***Label***и***PictureBox***

Этот объект в программе будет называться *PictureBox1****.*** Это имя будет использоваться в дальнейшем при написании кода программы.

1. **Установка свойств объектов**

Свойства изменяются при помощи выбора объектов на форме и изменения их значений в окне **Свойства (Properties).**

1. Установить свойства первой кнопки, для этого щелкнуть на кнопке *Button1****.***

Кнопка выделится и будет окружена манипуляторами изменения размера.

Затем необходимо сделать щелчок на строке заголовка окна ***Свойства (Properties).***

Если окно **Свойства (Properties)** скрыто, щелкните на команде ***Свойства (Properties)*** элемента главного меню **Вид (View)** или нажмите на клавишу **<F4*>.*** Изменить (если требуется) размер окна **Свойства (Properties)** так, чтобы в нем было достаточно места для отображения имен свойств и их текущих значений.

Когда вы привыкнете устанавливать свойства, то сможете использовать окно **Свойства (Properties)** без увеличения его размеров.

Окно **Свойства (Properties)** содержит список значений свойств для первой кнопки *Button1****.*** Они включают значения для цвета фона, текста, высоты шрифта и ширины кнопки и др. Так как имеется много свойств, **VS** организует их в категории и отображает в виде структуры. Если имеется желание увидеть свойства из какой-то категории, необходимо щелкнуть на знаке «плюс» (+) рядом с названием категории.

Прокрутить список окна **Свойства (Properties)** так, чтобы увидеть свойство *Text*, расположенное в категории ***Внешний вид (Appearance)***.

Дважды щелкнуть мышью в окне **Свойства (Properties)** в левом столбце свойства *Text*. В окне **Свойства (Properties)** будет выделено текущее значение свойства *Text* ***(«****Button1****»).***

Необходимо ввести текст *Играть* и нажать **<Enter>.**

В окне **Свойства (Properties)** свойство *Text* изменит свое значение на ***«****Играть****»,*** и этот же текст появится на самой кнопке формы.

1. Установить свойства второй кнопки, выделив вторую кнопку *Button2* другим способом. Для этого необходимо раскрыть список ***Объект (Object)*** в верхней части окна **Свойства (Properties)**.

Появится список объектов интерфейса программы (рис. 4.1.7-5).



Рис. 4.1.7-5. Список объектов *Form1*

В этом списке щелкнуть на *Button2* из списка всех используемых элементов управления ***System.Windows.Forms.***

В окне **Свойства (Properties)** появятся значения свойств второй кнопки, а **VS** выделит *Button2*на форме.

Дважды необходимо щелкнуть мышью на текущем свойстве *Text* ***(«****Button2****»***) и ввести текст «*Конец игры*», а затем нажать клавишу **<Enter>*.***

Название второй кнопки формы изменится на **«Конец игры**».

Использование списка ***Объект (Object***) является удобным способом переключения между объектами проекта. Можно переключаться между объектами формы, просто щелкая на них.

1. Установить свойства числовых надписей, для этого щелкнуть на надписи *Label1*, нажать на клавишу <**Shift**> и, не отпуская ее, щелкнуть на *Label2* и *Label3* надписях, а затем отпустить клавишу ***<*Shift*>.***

Вокруг каждой из надписей, на которых щелкнули, появятся пря­моугольники выделения и манипуляторы изменения размера. Теперь необходимо изменить свойства *TextAlign, BorderStyle* и *Font* так, чтобы числа, которые будут появляться в этих метках, находились в центре, имели рамку и отображались одинаковым шрифтом одинакового размера. Все эти свойства расположены в окне **Properties (Свойства)** в категории ***Appearance***. Необходимо также установить свойство *Autosize* в значение *False* с тем, чтобы можно было вручную изменить размер надписей. Свойство *Autosiz*e расположено в категории ***Layout.***

Обратите внимание, когда выделяется более одного объекта, в окне **Свойства (Properties)** отображаются только те свойства, которые могут быть изменены для всей группы выделенных объектов.

В окне **Свойства (Properties)** щелкнуть на свойстве *Autosize*, а затем щелкнуть на стрелке, появившейся справа от свойства.

Установить свойство *Autosize* в значение *False* так, чтобы можно было вручную изменить размер.

Щелкнуть на свойстве *TextAlign*, а затем щелкнуть на стрелке справа.

Появится список, содержащий набор вариантов выравнивания текста; эти настройки позволяют различным образом выравнивать текст в рамках объекта.

Щелкнуть на варианте центрирования -*MiddleCenter****.***

Свойство *TextAlign* для всех выделенных надписей изменится и примет значение *MiddleCenter.*

Щелкнуть на свойстве *BorderStyle* , а затем на стрелке справа.

В списке появятся доступные значения (*None, FixedSingle* и *Fixed3D*).

Щелкнуть в этом списке на *FixedSingle,* чтобы добавить вокруг каждой надписи тонкую рамку.

Щелкнуть на свойстве *Font*, а затем на кнопке многоточия (кнопка с тремя точками, которая расположена рядом с текущей настройкой шрифта).

Появится диалоговое окно **Шрифт (Font*).***

Необходимо изменить шрифт на *TimesNewRoman*, стиль шрифта – на *Жирный*, а его размер - на *24*, а затем нажать на **ОК**.

Отображаемый текст надписей будет иметь выбранный шрифт, стиль и размер.

Теперь необходимо изменить текст для всех трех надписей на цифру ***0*** - хороший заполнитель для цифр, которые будут выводиться в этих полях во время игры.

Щелкнуть на пустой области формы, чтобы снять выделение со всех трех надписей, а затем щелкнуть на *Label1*.

Дважды щелкнуть мышью на свойстве *Text,* ввести *0*, а затем нажать**<Enter>.**

Текст объекта *Label1*, будет содержать *0*. Код программы будет помещать в это свойство случайное число.

Далее необходимо изменить текст третьей и четвертой надписей формы так, чтобы он тоже содержал *0*.

Далее необходимо переместить и изменить размер надписей так, чтобы между ними было нормальное расстояние.

Форма выглядит примерно так, как показано на рис. 4.1.7-6.

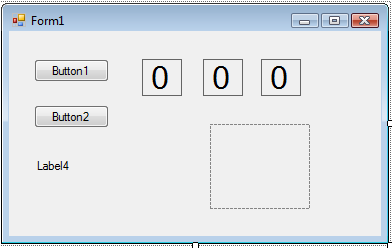


Рис. 4.1.7-6.Форма *Form1* с элементами управления ***Button***, ***Label*** и ***PictureBox*** и соответствующими надписями

1. Установить свойства описательной надписи, для этого щелкнуть на объекте надписи *Label1* формы. В окне ***Свойства (Properties)*** изменить свойство *Text* на значение - ***Проект-4.1.7-1*** – ***Иванова И.И*.**

Щелкнуть на свойстве *Font*, а затем на кнопке многоточия.

Использовать диалоговое окно **Шрифт (Font**) для изменения шрифта на *Arial*, стиля шрифта на жирный и его размера на *18*. Затем щелкните на **ОК**. Шрифт надписи объекта *Label4* обновится, и благодаря тому, что свойство *Autosize* объекта установлено в значение *True*, надпись автоматически изменит размер так, чтобы в нее поместился шрифт большего размера. Далее необходимо в окне **Properties (Свойства)** щелкнуть на свойстве *ForeColor****,*** а затем щелкнуть на стрелке во второй колонке.

**VS** для установки цвета переднего плана (цвета текста) объекта надписи отобразит список, содержащий вкладки ***Польз. (Custom)***, ***Web*** и ***Системные (System).*** Вкладка ***Польз. (Custom)*** предлагает многие из цветов, доступных в вашей системе.

Щелкнуть на *пурпурном (purple)* цвете на вкладке ***Польз. (Custom***).

Цвет текста надписи изменится на пурпурный цвет.

1. Установить свойства области изображения, для этого щелкнуть на объекте области изображения *PictureBox1*формы, щелкнуть в окне **Свойства (Properties)** на свойстве *SizeMode* категории ***Поведение (Behavior)***, щелкнуть на стрелке справа, а затем щелкнуть на *Stretchmage (Растянуть изображение****).***

Установка ***SizeMode*** в значение *Stretchmage (Растянуть изображение****)*** до открытия файла изображения заставляет **VS** изменить размер изображения так, чтобы он в точности совпал с размерами области изображения

Щелкнуть мышью в окне **Свойства (Properties)** на свойстве *Image*категории *Внешний вид (Appearance)****,*** а затем щелкнуть на кнопке ***многоточия*** во втором столбце. В результате появится диалоговое окно **Выбор ресурса (SelectResource).**

Щелкнуть на опции ***Локальный ресурс (LocalResource)***, а затем щелкнуть на кнопке **Импорт (Import).**

В диалоговом окне **Открыть (Open*)*** перейти в папку, например в **Проект-4-1-7-1.** Эта папка содержит рисунок ***Mtuci.gif***. Выделить ***Mtuci.gif*** затем щелкнуть на **Open**. В диалоговом окне **Select Resource** появится рис. 4.1.7-7.

Щелкнуть **ОК*.***

Рисунок ***Mtuci.gif*** загрузится в область вывода изображений.



Рис. 4.1.7-7. Диалоговое окно **Select Resource**

1. Изменить заголовок формы, щелкнув в области *Форма1 (Form1),* а затем в правом столбце свойства *Text* окна **Свойства (Properties)** и ввести текст   
   ***«Проект-4.1.7-1».***

Завершенная форма должна выглядеть, как на рис 4.1.7-7.

1. **Написание программного кода**

Теперь можно написать программный код для проекта ***Проект-4.1.7-1.***Так как большинство созданных ранее объектов уже «знают», как им надо работать, при запуске программы они будут готовы к приему ввода от пользователя и его обработке. Встроенная функциональность объектов является одной из самых сильных сторон **VS** и **VB**-после того как объекты будут размещены на форме и их свойства будут установлены, они готовы к запуску без дополнительного программирования (визуальное программирование). Однако основа игры ***Счастливая семерка*** - программный код, который реализует алгоритм вычисления случайных значений, отображает их в соответствующих полях и определяет выигрыш - в программе все еще отсутствует. Эта программная логика может быть встроена в приложение только с помощью операторов языка программирования **VB**-программного кода, который четко говорит, что программа должна делать на каждом шаге ее выполнения. Так как программой управляют кнопки ***Играть*** и ***Конец игры,*** программный код нашего проекта должен быть связан именно с этими кнопками. Для ввода и редактирования операторов программы на **VB**предназначено окно **Редактор программного кода (Code Editor).**

1. Представить значение свойств в таблице.

Свойства, которые были установлены для проекта ***Проект-4.1.7-1*** в пошаговом режиме, сведемв таблицу 4.1.7-2. Таблица 4.1.7-1

|  |  |  |
| --- | --- | --- |
| **Объект** | **Свойство** | **Значение** |
| *Button1* | *Text* | *" Играть"* |
| *Button2* | *Text* | *"Конец игры"* |
| *Label1, Label2, Label3* | *Autosize* | *False* |
| *BorderStyle* | *FixedSingle* |
| *Font* | *TimesNewRoman, Bold, 24* |
| *Text* | *"0"* |
| *TextAlign* | *MidleCetnter* |
| *Label4* | *Text* | *Проект-4.1.7-1-Иванова И.И.* |
| *Font* | *Ariel, Bold, 18-point* |
| *ForeColor* | *Purple* |
| *PictureBox1* | *Image* | *"d:\VB2008-Информатика-курс-1.Mtuci.gif"* |
| *SizeMode* | *Stretchmage* |
| *Visible* | *False* |

1. Дописать код программы для обработки события нажатия кнопки*Button2****,*** используя окно **Редактор кода (Code Editor**). Дважды щелкнуть мышью на кнопке **Конец игры** приложения***.***

В основном окне **VS** появится окно панели **Редактор кода (Code Editor),**имеющий вид документа с вкладкой и показанный на рис. 4.1.7-8.

Внутри окна **Редактора кода (CodeEditor)** находятся строки кода, связанного с текущей формой. Операторы программы, которые используются совместно для выполнения некоторого действия, обычно группируются в программные конструкции, называемые процедурами. Обычным типом процедуры является процедура Sub, называемая процедурой-подпрограммой. Процедуры Sub в первой строке содержат ключевое слово Sub и заканчиваются End Sub. Обычно процедуры выполняются при определенных обстоятельствах, например, таких, как щелчок по кнопке или вызов специального оператора. Когда процедура ассоциирована с конкретным объектом и событием, она называется ***обработчиком события*** или ***событийной процедурой***.

При двойном щелчке мышью на кнопке ***Конец игры* VB** автоматически добавляет первую и последнюю строки процедуры события Button2\_Click( ) , как показано на рис. 4.1.7-8. (Первая строка перенесена на следующую строку, чтобы не вылезти за поля.) На рис.4.1.7-8 видно, что в окне **Редактор кода (CodeEditor)** есть и другие строки кода, которые **VS** добавила для определения важных характеристик формы.

Тело процедуры, которое находится между заголовком процедуры и оператором, завершающим процедуру – End Sub, будет исполняться каждый раз, когда пользователь активизирует элемент интерфейса, ассоциированный с этой процедурой. В данном случае событие - это щелчок мышью на кнопке *Button2*, но как будет показано далее, это может быть и другой тип события.

Ввести оператор языка **VВ** End, а затем нажать на клавишу со стрелкой вниз.

После ввода оператора буквы становятся синими и создаются отступы, что указывает, что **VB** распознал End как одно из нескольких сотен зарезервированных - или ключевых слов языка **Visual Basic**. Ключевое слово End используется для остановки программы и удаления с экрана ее формы. В данном случае End также является законченным оператором программы, исполняемой компилятором **VB** - частью **VS**, обрабатывающей и выполняющей анализ каждой строки исходного кода **VB** и объединяющей результаты с другими ресурсами с целью создания исполняемого файла. Все операторы программы должны следовать грамматическим правилам компилятора. В**VS** операторы программ могут состоять из ключевых слов, свойств, имен объектов, переменных, чисел, специальных символов и других значений. При вводе операторов программы и внесении других изменений окно **Редактор кода (CodeEditor)** выполняет многие задачи, связанные с форматированием, включая настройку отступов, пробелов и добавление необходимых скобок. Точное написание, порядок и расположение пробелов оператора программы называется ***синтаксисом оператора***.

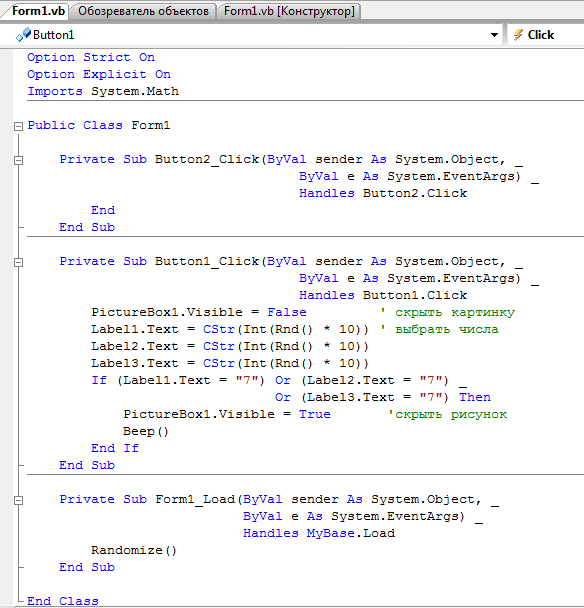


Рис. 4.1.7-8. Программный код проекта ***Проект-4.1.7-1***

При нажатии на клавишу ***«стрелка вниз»*** оператор End был сдвинут так, чтобы он был отделен от операторов Private Sub и End. Эта схема форматирования является одним из соглашений программирования и предназначена для того, чтобы программы выглядели ясно и читаемо. Соглашения относительно того, как организуется код в программе, часто называется ***стилем программирования***.

Теперь, когда написан код, связанный с кнопкой **Конец игры**, необходимо написать код для кнопки **Играть**.

1. Написать программный код, который будет выполняться при нажатии кнопки **Играть*,*** то есть ***событийную процедуру*** *для кнопки Button1*.

Чтобы отобразить форму, щелкнуть в окне **Обозреватель решений (Solution Explorer)** на кнопке **Конструктор(View Designer**). В режиме ***Редактор программного кода (CodeEditor)*** форма, с которой производится работа, не отображается. Кнопка **Конструктор (View Designer)** является одним из тех механизмов, которые используются для того, чтобы снова вывести форму на экран. Также можно щелкнуть на вкладке ***Form1.vb [Design].***

Дважды щелкнуть мышью на кнопке **Играть.**

Через некоторое время появится окно **Редактор программного кода (Code Editor),** и рядом с процедурой события, связанной с кнопкой Button2, появится процедура события, ассоциированная с кнопкой *Button1****.***

Хотя текст этой кнопки был изменен на ***«*Играть*»,*** ее имя в программе осталось *Button1*. (Имя и текст элемента интерфейса могут быть различными для удобства программирования.) Каждый объект может иметь несколько связанных с ним процедур - по одной для каждого распознаваемого события.

Между операторами Private Sub Button1\_Click( ) и End Sub необходимо ввести строки программы, показанные на рис. 4.1.7-8.

После ввода каждой строки необходимо нажимать ***<Enter>,*** а для создания отступа необходимо нажать на ***<Таb>*** и следить за вводом операторов программы в точности так, как они напечатаны на рис. 4.1.7-8. (При вводе длинных строк **Редактор программного кода (Code Editor)** будет прокручивать экран влево.) Если будет сделана ошибка, которую обычно легко определить по подчеркиванию волнистой лини­ей, удалите неверную строку и введите ее снова.

При вводе кода программы **VB** форматирует ее текст и отображает различные части программы различными цветами, которые помогают распознавать различные элементы. В момент ввода свойства отображается список, содержащий все допустимые свойства объекта, так что вместо набора свойства вручную можно дважды щелкнуть на свойстве мышью. Если **VB** отображает сообщение об ошибке, то оператор, скорее всего, введен с ошибкой. Сверьте строку с текстом в пособии, сделайте необходимые исправления и продолжайте набирать. Также можете удалить всю строку и начать вво­дить ее заново. Кроме того, **VB** может автоматически добавлять необходимый код.

Когда закончите ввод операторов, окно **Редактор программного кода (CodeEditor)** должно выглядеть так, как показано на рис. 4.1.7-8.

1. Сохранить введенный программный код, щелкнуть на команде ***Сохранить все (SaveAll)*** элемента главного меню **Файл (File).**

Команда ***Сохранить все (SaveAll)*** сохраняет все файлы, входящие в проект – собственно файл проекта, файл формы, все модули кода и все связанные с приложением компоненты. Если это первое сохранение проекта, появится диалоговое окно **Сохранить проект (SaveProject),** запрашивающее имя и место для сохранения проекта. Если **VS**  настроена так, чтобы запрашивать место сохранения проекта при его создании, то сейчас диалогового окна **Сохранить проект**  вы не увидите - **VS** просто сохранит все изменения.

Щелкнуть на кнопке **Обзор (Browse)** справа от текстового поля ***Место- положение (Location)*** и выбрать место для сохранения файлов.

Необходимо использовать папку (место, где находятся файлы лабораторных работ, созданные студентами). Если необходимо сохранить только элемент, с которым в настоящий момент работаете (форму, модуль кода или что-либо еще), то используется команда ***Сохранить (Save),*** также находящаяся в меню **Файл (File**). Если необходимо сохранить текущий элемент с другим именем, то используется команда ***Сохранить как (SaveAs).***

1. Обсудим программный код процедуры обработки события нажатие кнопки ***Играть,*** используя окно **Редактор программного кода (CodeEditor).**

Процедура Button1\_Click( ) выполняется тогда, когда пользователь щелкает на кнопке ***Играть*** формы *Form1*. Эта процедура содержит несколько довольно сложных операторов. Процедура **Button1\_Click( )** выполняет три задачи:

* скрывает рисунок;
* создает три случайных числа, отображаемые в числовых надписях;
* когда появляется цифра **7**, она отображает рисунок.

Сокрытие рисунка выполняется с помощью следующей строки:

|  |
| --- |
| **PictureBoxl.Visible=False' скрыть картинку** |

Эта строка состоит из двух частей: программного оператора и комментария. Оператор программы (PictureBox1.Visible=False) устанавливает свойство ***Visible*** объекта области вывода изображения (***PictureBox***) равным ***False*** (это одно из двух возможных значений). Вы, возможно, помните, что только что установили это свойство равным ***False*** при помощи окна **Свойства (Properties).** Сейчас это снова делается в коде программы, так как первая задача - это «вращение», и фото, которое может остаться на экране после предыдущей игры, требуется убрать. Так как это свойство будет изменено во время работы, а не во время разработки, нужно установить его с помощью кода программы. Это удобная функция **VB.**

Вторая часть первой строки (часть, отображаемая на экране зеленым цветом) называется ***комментарием***. ***Комментарии*** - это описательные пояснения, включаемые в программный код после символа апострофа ('). Программисты используют комментарии для описания действия операторов программы. **VB** при запуске программы не обрабатывает эти пояснения; они существуют только для документирования того, что делает программа. При написании программы часто придется использовать комментарии, чтобы оставить легко воспринимаемые записи о том, что же вы сделали. Следующие три строки выполняют вычисление случайных чисел. С помощью специальных действий можно на самом деле заставить **VB**  сгенерировать непред­сказуемые числа – другими словами, можно создать случайные числа для лотерей, игры в кости или других статистических задач. Функция Rnd( ) в каждой из строк создает случайное число в диапазоне между 0 и 1 (число с десятичной точкой и несколькими знаками после нее), а функция Int( ) возвращает целую часть результата произведения этого случайного числа на 10. Это вычисление дает программе случайные числа между 0 и 9 - именно то, что требуется для нашего приложения игрового автомата.

|  |
| --- |
| **Label1.Text=CStr(Int(Rnd( ) \* 10)) ' выбрать числа** |

Затем в коде требуется преодолеть небольшое препятствие. Вы должны скопировать эти три случайных числа в три поля надписей на форме, но сначала числа с помощью функции CStr( ) (преобразование в строку) должны быть преобразованы в текст. Обратите внимание, как CStr(), Int() и Rnd()объединены вместе в одном операторе программы – они работают совместно и дают результат аналогично математической формуле. После вычисления и преобразования значения присваиваются свойствам *Text* первых трех надписей формы, и в результате этого числа отображаются в этих трех надписях жирным шрифтом *TimesNewRoman* с размером *24* пункта. Последняя группа операторов программы проверяет, не равняется ли **7** одно из случайных чисел. Если одно или несколько из этих чисел равно **7**, то программа отображает изображение, а звуковой сигнал о победе:

|  |
| --- |
| **'если одна из цифр-7, то отобразить картинку и издать звук**  **If (Label1.Text = "7") Or (Label2.Text = "7") \_**  **Or(Label3.Text = "7") Then**  **PictureBoxl.Visible = True**  **Beep()**  **End if** |

Каждый раз, когда пользователь щелкает на кнопке **Играть,** выполняется (вызывается) процедура Button1\_Сlick( )***,*** и снова выполняются операторы программы, содержащиеся в этой процедуре (рис. 4.1.7-8).

1. Технология ***IntelliSense*** и точечная нотация

Теперь известно, как с помощью **VB** читать и записывать значения свойств форм и эле­ментов управления. Если у каждой формы и элемента управления много свойств, как за ними уследить? Не слишком удобно постоянно переключаться меж­ду редактором кода и окном **Свойства**, чтобы выяс­нить, какие свойства нам нужны. Имеется более удобный способ.

Можно использовать технологию ***IntelliSense***. Это часть **VS**. ***IntelliSense*** позволяет сохранять время при редактировании программного кода в окне редактора. ***Intelli­Sense*** знает, к какому элементу управления обра­щаются, и какие

у него есть свойства. Оно выводит список свойств и позволяет вам выбирать из него нужное. Используемое чаще всего свойство подсвечи­вается, когда ***IntelliSense*** отображает список. Напри­мер, для текстового поля изначально подсвечивается свойство *Text*. С помощью стрелок вверх и вниз на клавиатуре можно просматривать список и выбирать то свойство, которое вам требуется. При нажа­тии клавиши **<Tab>** выбранное свойство добавляется в код. Используя ***IntelliSense,*** не придется запо­минать все свойства всех элементов управления. Кроме того, данное средство заметно сокращает по­требность в наборе длинных имен на клавиатуре!

Рассмотрим это на примере.

Введем с клавиатуры PictureBox1 и нажмем клавишу «точка» - рис. 4.1.7-9. Как только вы введете точку, ***IntelliSense*** отобразит список свойств элемента PictureBox1. Поскольку PictureBox1 - изображение, этот список будет таким же, как и для любого другого изображения на форме. По умолчанию в списке ***IntelliSense*** подсвечивается чаще всего используемое свой­ство (рис. 4.1.7-9).



Рис. 4.1.7-9. Свойства PictureBox1, отображаемые с помощью IntelliSense

Если далее будет введена буква, например буква «V», ***IntelliSense*** про­крутит список до первого по алфавиту свой­ства, начинающегося с «V». По мере ввода ***IntelliSense*** ищет в списке свойство, наибо­лее соответствующее введенным буквам.

Когда свойство подсвечи­вается в списке ***IntelliSense***, рядом появляется под­сказка, кратко его описы­вающая. Эта подсказка, например, может сооб­щить, если свойство до­ступно только для чтения (***ReadOnly***, т. е. ему нель­зя присваивать значение). Кроме того, подсказка со­общает, какие значения можно присваивать свой­ству, например, строку (текст) или целое число.

1. ***Выполнение приложения VB***

Чтобы запустить программу **VB** из среды разработки, необходимо сделать одно из следующего:

* выполнить команду ***Начать отладку (StartDebugging)*** элемента меню **Отладка (Debug);**
* щелкнуть на кнопке ***Начать отладку (StartDebugging)*** на стандартной панели инструментов;
* нажать**<F5>.**

1. Запустить проект ***Проект-4.1.7-1***. Если **VB** выводит сообщение об ошибке, то при вводе кода программы, вероятно, была допущена одна или несколько ошибок. Следует исправить их, сравнив то, что было введено, с вариантом из этого пособия.

Если все было сделано правильно, то ***Проект-4.1.7-1*** будет компилироваться, и запустится в **VS 2008**. Через некоторое время появится созданный интерфейс пользователя, ожидающий действий от пользователя.

1. Далее необходимо щелкнуть на кнопке **Играть**.

Программа сгенерирует три случайных числа и отобразит их в надписях на форме, как показано на рис. 4.1.7-10. Так как в поле надписи появилась 7, появится рисунок, означающий выигрыш, и компьютер издает звуковой сигнал. Вы выиграли! (Звук, который вы услышите, зависит от настроек.)

|  |  |
| --- | --- |
|  |  |

Рис. 4.1.7-10. Результаты работы проекта Проект-4.1.7-1

Щелкните на кнопке **Играть** еще раз, наблюдая за результатом в числовых полях. Для окончания эксперимента с вашим новым проектом щелкните на кнопке ***Конец игры.*** Программа остановится и на экране снова появится среда разработки.

1. ***Внесение изменений в программу***

В любое время можно перезапустить **VS** и работать с программным проектом, сохраненным на вашем диске. Снова запустить **VS** и добавить в проект***Проект-4.1.7-1*** оператор Randomize***:***

1. Запустить **VS**.

На стартовой странице **VS** появится список недавно открывавшихся проектов. Так как только что закончили работать с ***Проект-4.1.7-1***, первым проектом в этом списке должен быть проект ***Проект-4.1.7-1***.

1. Чтобы открыть проект ***Проект-4.1.7-1***, щелкнуть на ссылке ***Проект-4.1.7-1*** .

Откроется проект ***Проект-4.1.7-1*** и появится форма.

Далее вы должны добавить в процедуру Form1\_Load( ), специальный оператор, который связан с формой и исполняется каждый раз при запуске программы - оператор Randomize

Чтобы отобразить на экране процедуру Form1\_Load( ), сделайте на форме (но не на одном из ее объектов) двойной щелчок мышью.

В окне **Редактор программного кода (Code Editor)** появится процедура Form1\_Load( ).

1. Ввести в программный код оператор Randomize, а затем нажать на клавишу со стрелкой вниз.

Оператор Randomize добавится в программу и будет исполняться каждый раз при ее запуске. Чтобы создать истинно случайную начальную точку для функции Rnd( ), используемой в процедуре Button\_Click( ), Randomize использует системные часы. Без оператора Randomize***Проект-4.1.7-1*** при каждом ее запуске производит одну и ту же последовательность случайных чисел. При наличии Randomize программа при каждом запуске генерирует новую последовательность случайных чисел.

1. 3апустить новую версию ***Проект-4.1.7-1***, а затем сохраните проект.
2. Когда закончите работу с проектом, выполните команду ***Закрыть проект (CloseProject)*** в элементе главного меню **Файл(File**).

Файлы, связанные с программой ***Проект-4.1.7-1***, будут закрыты.

1. ***Создать исполняемый файл***

Последней задачей этого примера является завершение процесса разработки и создание приложения для **MS Windows,** или исполняемого файла (при условии, что полученное решение соответствует заданию). Приложения для **MS Windows,** созданные в **VS,**имеют расширение ***.ехе*** и могут выполняться на любой системе, имеющей **MS Windows**и необходимые файлы поддержки. **(VB**устанавливает все эти файлы, включая файл **.NET Framework,** автоматически.)

В данный момент необходимо знать, что **VB** может создавать два типа исполняемых файлов: отладочный и окончательный.

Отладочные версии создаются **VS** автоматически при создании и тестировании программы. Они сохраняются в папке, которая расположена в папке проекта, и называются ***\bin\debug.*** Отладочный исполняемый файл обычно содержит отладочную информацию, из-за которой программа выполняется более медленно. Окончательные версии - это оптимизированные исполняемые файлы, сохраняемые в папке проекта ***bin\release****.*

Чтобы установить параметры для окончательной версии, выполнитькоманду ***Свойства: Проект-4.1.7-1...*** элемента главного меню **Проект (Project),** а затем щелкнуть на вкладке ***Компиляция (Compile),*** где находится перечень опций компилятора рис. 4.1.7-11.

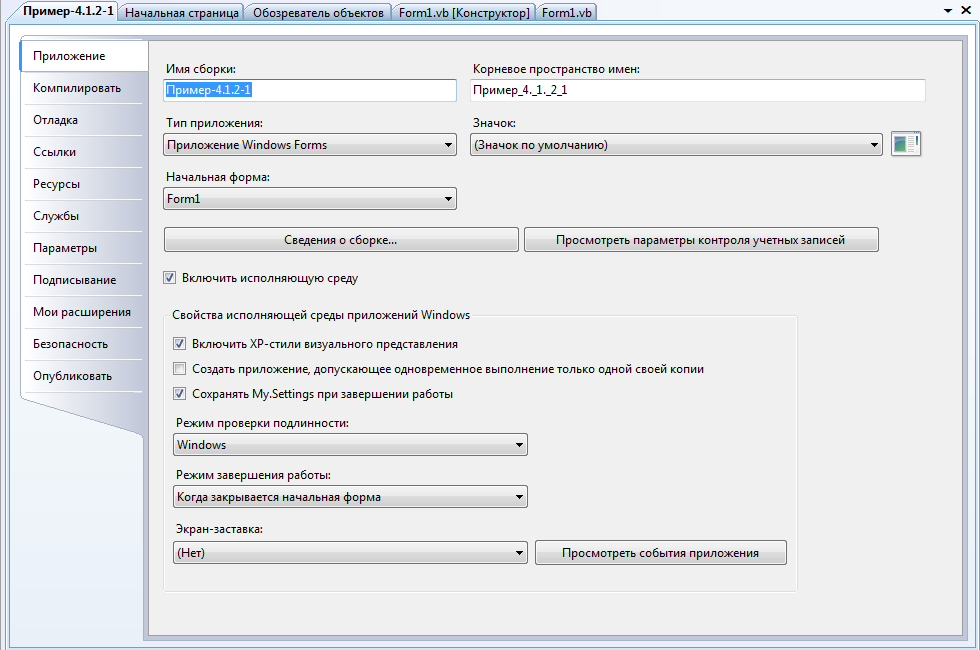


Рис. 4.1.7-11

Далее необходимо реализовать следующие шаги.

1. В элементе главного меню **Построение (Build)** выполнить команду ***ПостроитьПроект-4.1.7-1(Вuild Project-4.1.7-1).***

Команда ***Построить (Вuild)*** для сохранения файла создает папку с именем ***bin\release*** (если она еще не существует) и компилирует исходный код проекта.

Результатом этого является исполняемый файл с именем ***Проект-4.1.7-1***.***ехе.***

1. Теперь необходимо запустить эту программу вне **VS**, используя команду

***Run (Выполнить)*** из меню ***Start (Пуск )*MS Windows**.

Появится диалоговое окно **Run**.

1. Щелкните на **Browse (Обзор)**, а затем перейдите в папку ***\release.***
2. Щелкните на пиктограмме приложения***Проект-4.1.7-1***.***ехе***, щелкните на ***Open (Открыть),*** а затем щелкните на **ОК**.

Программа ***Проект-4.1.7-1*** загрузится и запустится в **Windows.**

1. Щелкните несколько раз на кнопке ***Играть*** формы, чтобы проверить

работоспособность проекта, а затем щелкните на кнопке ***Конец***.

(Можно запустить приложения для **MS Windows**, включая откомпилированные программы **VB**, открыв проводник **MS Windows**).

1. Чтобы закрыть **VS** и проект ***Проект-4.1.7-1***, в меню **Файл (File)** выполните команду ***Выход (Exit).***

Среда разработки **VS** закроется.

1. ***Выход из Visual Studio 2008***

Сохранить все изменения, которые вы сделали в вашей программе, можно щелкнув на кнопке ***Сохранить все (SaveAll)*** на стандартной панели инструментов.

Поведением по умолчанию **Visual Studio 2008** является ситуация, когда программе имя присваивается при создании проекта, но при этом не указывается местоположение файла и не сохраняется проект до тех пор, пока не щелкните на кнопке ***Сохранить все (SaveАll)***или не выполните команду ***Сохранить все (SaveАll)*** из элемента главного меню **Файл (File),** в связи с тем, что в проект внесено несколько изменений.

В меню **Файл (File)** выполните команду ***Выход (Exit).***

**Visual Studio** закроется.

### 4.1.8. Контрольные вопросы по теме «Интегрированная среда разработки Visual Studio .NET и основные понятия»

### 4.1.9. Тестовые задания по теме

### «Интегрированная среда разработки Visual Studio .NET и основные понятия»

1. **Проект - это**
2. разрабатываемое приложение, состоящее из набора файлов, в которых хранится

информация обо всех компонентах, используемых в приложении

1. семейство процедур
2. набор модулей
3. множество форм
4. **Проект VB хранится в файле с расширением**
5. **.frm**
6. **.vbproj**
7. **.bas**
8. **.fix**
9. **В окне Обозреватель решений отображаются**
10. форма и элементы управления
11. проекты и файлы текущего решения
12. модули и процедуры
13. программный код
14. **Окно Конструктор форм служит для**
15. создания форм приложения с использованием инструментария среды

разработки

1. конструирования модулей
2. написания программного кода
3. **Программный код проекта**
4. существует сам по себе
5. привязан к отдельным объектам и не оторван от формы
6. привязан к отдельным объектам и не связан с формой
7. привязан только к форме
8. **Свойства объекта в среде Visual Basic можно изменить**
9. только на этапе проектирования интерфейса
10. только на этапе кодирования текста программы
11. и на этапе проектирования интерфейса, и на этапе кодирования текста программы
12. **Форма – это**
13. философское понятие, противоположное "содержанию" программы
14. объект, появляющийся на экране при запуске Visual Basic и предназначенный для ввода данных
15. объект Visual Basic, появляющийся на экране при запуске программы, и

на котором размещаются элементы управления

1. объект, появляющийся на экране при запуске программы и предназначенный для вывода результатов работы программы
2. **Элемент управления – это**
3. объект, с помощью которого программа запускается на выполнение
4. объект, являющийся элементом графического интерфейса приложения и реагирующий на события
5. инструмент, который используется для создания объектов на форме VB
6. объект, появляющийся на экране при запуске программы и предназначенный для останова программы
7. **Текстовые поля могут быть использованы для**
8. заполнения окна формы
9. вывода информации на экран и для получения данных от пользователя
10. только для вывода информации на экран
11. только для получения данных от пользователя
12. лучше их не использовать
13. **В файле формы (с расширением .vb) хранится**
14. форма
15. программный код, связанный с формой
16. данные об объектах, расположенных на форме
17. данные о форме, объектах, расположенных на ней, и программный код, связанный с формой
18. **Оператор программы Visual Basic – это**
19. строка кода программы, завершенная конструкция, исполняемая

компилятором

1. присваивание, **For…Next, If**
2. арифметические операции
3. ключевые слова
4. **Ключевое слово – это**
5. зарезервированное слово языка VB, распознаваемое компилятором
6. специальные имена переменных
7. встроенные функции
8. объекты
9. **Свойство в Visual Basic – это**
10. особый элемент управления
11. имя текстового поля
12. окно Свойства (Properties)
13. значение или характеристика, хранимая объектом
14. **Именами свойств объ­ектов являются**
15. Text
16. Integer
17. Date
18. Names
19. String
20. Single
21. **В программе при обращении к объекту указывается свойство**
22. Textalign
23. Text
24. Name
25. Enabled
26. В строку заголовка формы выводится значение свойства
27. Textalign
28. Text
29. Name
30. Multiline
31. Выравнивание текста в метке определяется значением свойства
32. Textalign
33. Text
34. Name
35. Multiline
36. Вывод нескольких строк в текстовом окне разрешает (или запрещает) значение свойства
37. Caption
38. Alignment
39. Text
40. Name
41. Multiline
42. Именем событийной процедуры в программе является ###

|  |
| --- |
| **…**  **Dim intA As Integer**  **Dim intB As Integer**  **Dim intC As Integer**  **Private Sub Button1\_Click( )**  **intA =2**  **intB = 3**  **intC = intA/intB**  **Form1. TextBox1.Text=Chr(intC)**  **End Sub**  **…** |

1. Button1\_Click()
2. Cоответствие между событием и тем, когда оно происходит

L1: Load

L2: Click

L3: DBlclick

L4: Show

R1: при загрузке формы

R2: по щелчку кнопкой мыши в момент, когда указатель мыши находится на управляющем элементе

R3: по двойному щелчку кнопкой мыши в момент, когда указатель мыши находится на управляющем элементе

R4: при изменении значения свойства Visible поль­зовательского управляющего элемента или докумен­та на True