## Тема 4.1

## Интегрированная среда разработки Visual Studio .NET

### 4.1. Технология визуального программирования и основные понятия

По сравнению с предыдущими поколениями языков программирования, язык **программирования высокого уровня** – Visual Basic .Net (VB)имеет ряд принципиальных отличий. Это мощная, профессиональная система программирования, позволяющая быстро и эффективно создавать приложения для MS Windows**.** Поскольку система является *«****визуальной****»,* программист может создавать видимую часть приложения MS Windows**,** не написав почти ни строчки программного кода. Эта видимая часть является основой интерфейса пользователя *«****программный проект – пользователь****»,* с помощью которого осуществляется взаимодействие пользователя с программным проектом. Разработка, как программного кода классов и методов для решения поставленной задачи, так и программного интерфейса приложения осуществляется на принципах ***объектно-ориентированного подхода****,* реализованного в Visual Studio .Net (VS)**.** Приложения (проекты) выполняются под управлением MS Windowsс помощью ***Интегрированной Среды Разработки (IDE)*.**

Для приложений, созданных с помощью IDE, характерной особенностью 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***кнопкой мыши, перетащить объект, ввести данные в окно и т.п. В большинстве случаев не предполагается заранее установленная последовательность действий. Как правило, разработчик **программного обеспечения** (ПО), создавая программный проект, не должен ограничивать действия пользователя приложения. Он должен разрабатывать ПО, которое адекватно реагирует на любое действие пользователя, даже неправильное (в хорошей программе непредусмотренных действий не должно быть). Поэтому 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. другие события происходят в результате совершения других событий: например, окно открывается или закрывается, элемент управления становится активным (доступным) или теряет активность. Причем каждое событие проявляется в определенных действиях (откликах, реакции, поведении) программного проекта. Эти действия можно разделить на две группы.

Первая является следствием свойств объекта. Эти свойства устанавливаются разработчиком ПО из некоторого стандартного перечня свойств, заданного системой программирования и/или самой 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 разработчиком ПО. Это обеспечивается возможностью задания процедуры VBдля каждого возможного события. Первоначально каждая такая процедура является пустой, без тела процедуры, и, следовательно, когда происходит событие, ничего не случается. Теоретически можно создать процедуру для любого события, но на практике программист заполняет программным кодом процедуры только для тех событий, которые представляют интерес.

Таким образом, VBпредоставляет пользователю объектно-ориентированную среду программирования, основанную на событиях. Упрощенно говоря, процесс разработки программного проекта в этой среде сводится к выбору набора объектов и их свойств, заданию событий и процедур их обработки, и создания ПО для решения задачи, которые в совокупности обеспечивают функционирование создаваемого приложения.

Основными терминами визуального программирования, как было сказано выше, являются ***объект*, *форма*, *свойство, класс, событие, процедура*, *метод др.*,** 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 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 экземпляров классов).

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 ее 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**)и др**.**

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*,** ив то же время сама 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 (классы, процедуры, методы, модули), из которых «собирается» проектируемый проект (приложение).

***Свойство объекта*** – 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** 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***.

***Класс*** – это совокупность объектов, обладающих общими свойствами и поведением. Например, **Button1** *(*Кнопка1*)* на **Form1** (Форма1***)*** со всеми своими конкретными свойствами и 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 VBколлекции, 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 программного кода может быть помещен оператор **Imports*.*** Каждый проект в VSтакже содержит корневое пространство имен. Пространство имен в документации по VSназывается 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 Windowsв данном случае выступает в роли посредника. Восприняв событие, связанное с нажатием клавиши, операционная система 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 VBвыполняет как бы внутреннюю работу. Например, когда в приложении происходит щелчок мышью по кнопке, возникает событие – VBперехватывает соответствующее сообщение MS Windowsи 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 разработчиком ПО. 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).

***Процедура пользователя*** – это законченный программный код, который реализует отдельный алгоритм решаемой задачи. 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. VBпредоставляет пользователю большое количество встроенных процедур, находящихся в библиотеке, и, готовых к использованию. Эти процедуры являются неотъемлемой частью библиотеки классов, то есть являются методами классов, Framework .NET.

Средства процедурного программирования (процедуры) – неотъемлемая часть любого языка программирования высокого уровня, 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 VBможно создавать ***собственные* процедуры,** создаваемые разработчиком ПО*.*

***Обратите внимание, что любой программный код проекта, написанный средствами VB состоит из множества процедур, то есть, весь программный код находится как бы внутри множества процедур-подпрограмм и процедур-функций, а сами процедуры внутри класса (модуля).***

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.

***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.

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).

***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

**Form4.1.Text = "Форма с картинкой"** означает, что в памяти с именем **Form4.1.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 (программный код приложения) – набор файлов, в которых хранится информация обо всех программных 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 ПО,*** то результатом процессов программирования и проектирования является проект, который объединяет в себе программный код и графический интерфейс. Поэтому, чтобы записать на диск или загрузить с диска какой – либо программный проект, необходимо записывать или загрузить проект, в котором находятся все компоненты (файлы). Обычно приложение в VBсостоит 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. Так, если разработчик указывает, что он будет создавать проект типа «**Приложение** **Windows Forms**», то в состав каркаса проекта по умолчанию войдет объект **Form1** – наследник библиотечного класса **Form**. Разработчик программного проекта дополнит созданную форму элементами управления – объектами соответствующих классов, тем самым расширив возможности формы, построенной 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 также ***установки и ресурсы***, требуемые для работы. Частью проекта также является файл, содержащий описание сборки (манифест).

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

К невыполняемым проектам относятся, например, проекты типа   
**Библиотека классов (DLL)**, которая имеет расширение .**dll,** и является ***динамически связываемая библиотека*,** т.е. эти проекты, предназначены для включения (связывания) в другие проекты. В результате компиляции проекта типа   
***Библиотека классов*** в сборку войдет файл с расширением **.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 простейших проектов*** – 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;
* выполнение проекта.

***Решение*** – это разрабатываемое приложение, файлы которого содержат информации об одном или нескольких проектах, как правило, связанных общей темой. Файл решения используется для управления несколькими взаимосвязанными проектами и имеет расширение **.sln.** Причем каждый проект, создаваемый в VS, помещается в одну оболочку, называемую **Решением**.

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

Итак, обратите внимание еще раз на следующее:

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

* ***Класс*** – во-первых, 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*** – во-вторых, 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 играют обе роли. Но в VB можно определять классы, играющие только роль модуля. Это ***модули***, для которых невозможно создавать объекты.

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 проекта 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 объектов, 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. Разработчик проекта 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.

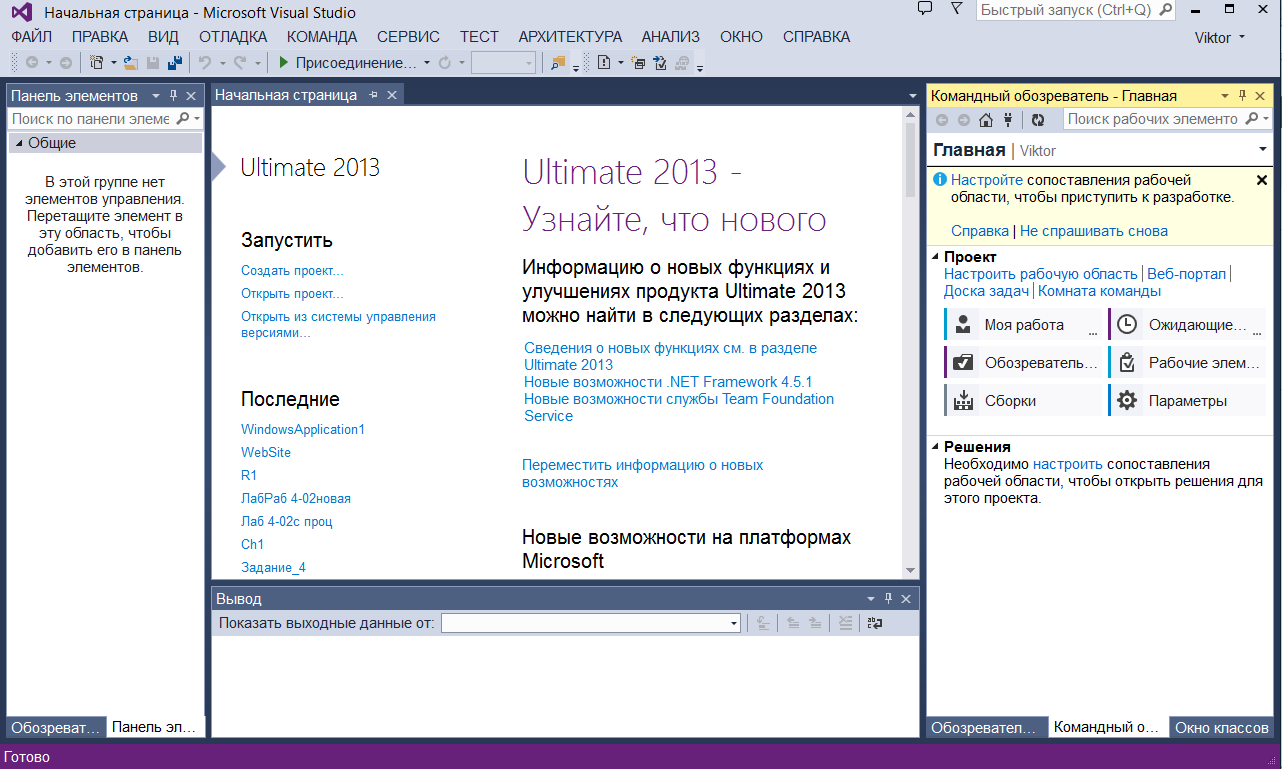
### 4.1.2. Главное окно VS

Отметим, что процесс создания простейшего приложения в среде VS подробно будет описан в следующей **Теме 4.2**, а также в лабораторной работе по **Теме 4.1.** «Интегрированная среда разработки Visual Studio .NET». Сейчас познакомьтесь с интерфейсом среды VS.

**Запустить *MS Visual Studio.***

На экране появится **Начальная страница** - MS VS **(**рис. 4.1.2-1).

В этом окне можно выделить несколько основных панелей: **Стандартное меню**, окно **Стандартная панель инструментов,** окно **Начальная стр**аниц**а,** окно **Обозреватель решений,** окно **Панель элементов.**



*Рис. 4.1.2-1. Начальная страница -* ***Microsoft Visual Studio***

Необходимо отметить, что при работе с VB в интегрированной среде разработки VSможно пользоваться как кнопками стандартной панели инструментов, так и элементами главного меню, расположенными в верхней части экрана.

VSпредоставляет в распоряжение пользователя набор разнообразных панелей инструментов. Эти панели инструментов содержат кнопки, назначение которых зависит от функций конкретной панели инструментов.

Окно **Начальная страница** позволяет открывать использовавшиеся недавно проекты, осуществляет поиск примеров, содержит различные ссылки на сайты, которые могут помочь при работе с VS**.** Кроме тогов процессе работы с проектом на месте начальной страницы будут отображаться различные окна, например, окно **Конструктора формы**, окно **Редактора программного кода** и др.

В окне **Обозреватель решений** отображаются файлы текущего решения (проектов). Это окно используется для получения подробной информации об объектах**.** Оно позволяет искать и исследовать элементы, их свойства, методы, события, находящиеся в проектах и ссылках на них.

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

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

* выбрать ссылку ***Создать проект...***в окне **Начальная страница**;
* выполнить команду ***Создать проект*** из элементаГлавного меню – **Файл;**
* нажать кнопку ***Создать проект*,** расположенную на стандартной панели инструментов.

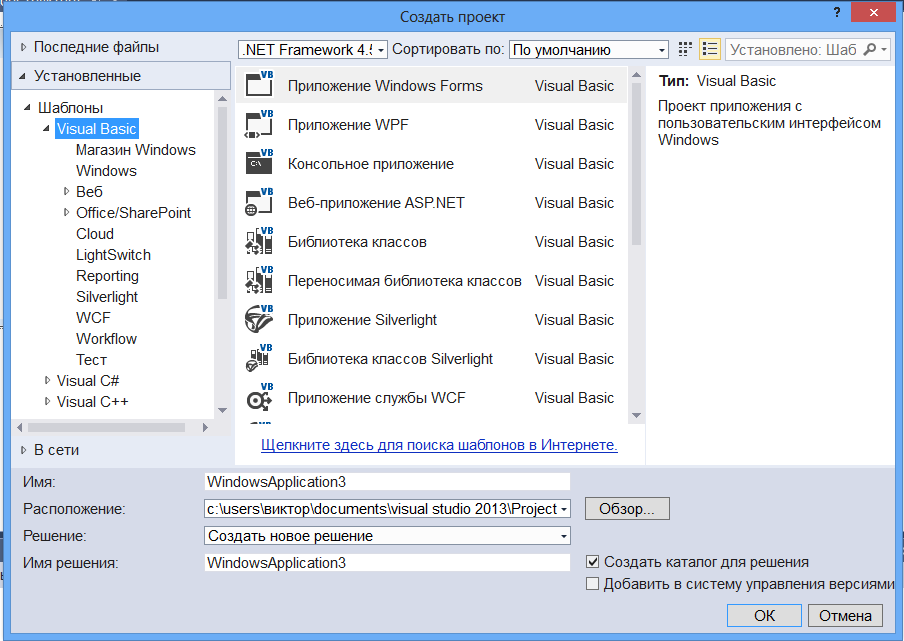


Рис. 4.1.2-2. Диалоговое окно ***Создать проект***

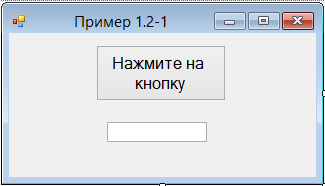
В VBможно создавать приложения по различным шаблонам. В этом пособии будем использовать только шаблон **Приложение Windows Forms.**

Это приложение предназначено для создания традиционных Windows-приложений, пользовательский интерфейс для которых проектируется с помощью форм MS Windows. Причем для формы этого шаблона можно задавать определенные характеристики и располагать на ней различные элементы управления.

Рассмотрим на примереосновные шаги, необходимые для создания проекта **Приложение** **Windows Forms**.

**Пример 4.1.2-1. Создать проект с именем Пример 4.1.2-1и графическим интерфейсом, состоящим из формы, на которой имеется одна кнопка и одно текстовое поле. При нажатии на кнопку должно появиться «Привет, МТУСИ!».**

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**

Для реализации проекта **Пример 4.1.2-1** необходимы следующие шаги:

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

* выбрать элемент Главного меню **Файл**, а затем выполнить команду ***Создать проект;***
* выбрать шаблон проекта **Приложение Windows Forms*.***

В результате откроется окно **Form1.vb[Конструктор]** – **Конструктор форм** (на что указывает соответствующая вкладка), позволяющий реализовать графический интерфейс приложения.

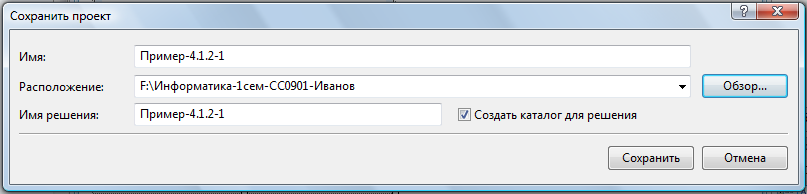
1. **Сохранить созданный проект в «своем» каталоге:**

* выбрать элемент Главного меню **Файл**, а затем выполнить команду ***Сохранить все***;

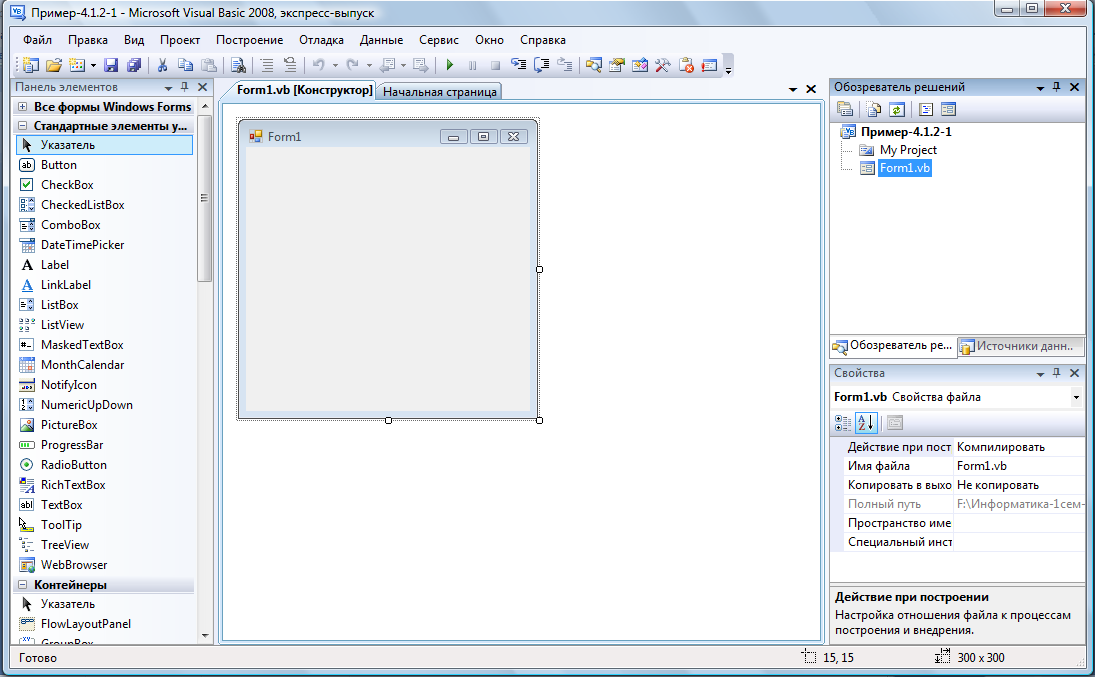
в результате откроется диалоговое окно **Cохранить проект**;

* ввести в поле **Имя** наименование проекта **Пример-4.1.2-1** (по умолчанию среда VB присваивает имя WindowsApplication1), а затем ввести в поле **Расположение имя каталога**, где будет располагаться проект, или воспользоваться кнопкой ***Обзор*** для поиска необходимого каталога (рис. 4.1.2-4);
* щелкнуть по кнопке ***Cохранить***;

в результате среда разработки примет вид как на рис. 4.1.2-5.



*Рис. 4.1.2-4. Диалоговое окно* **Сохранить проект**



*Рис. 4.1.2-5. Среда разработки*

1. **Создать графический интерфейс проекта.**

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

* выбрать элемент Главного меню **Вид**, а затем выполнить команду **Панель элементов**;
* в результате появится окно **Панель элементов** с набором элементов управления, которые можно разместить на форме;
* выбрать необходимый элемент – **Button** (рис. 4.1.2-6) и поместить его на форму – для этого необходимо просто кликнуть в нужном месте формы;
* изменить при необходимости местоположение элемента управления на форме путем перетаскивания или растягивания маркеров;
* изменить свойство Text элемента управления в окне **Свойства** на нужное (рис. 4.1.2-7);
* повторить те же действия для всех необходимых элементов графического интерфейса, в том числе для самой формы.

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | | *Р Р Рис. 4.1.2-6. Фрагмент о окна* **Панель элементов** | *Р*  *Р Рис. 4.1.2-7. Фрагмент окна* **Свойства** | |  |

1. **Дописать необходимый для выполнения задания программный код.**

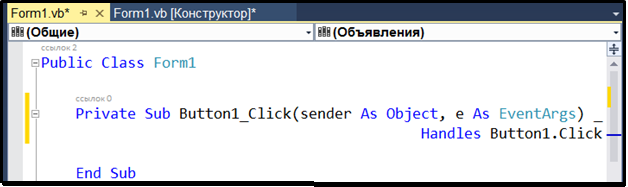
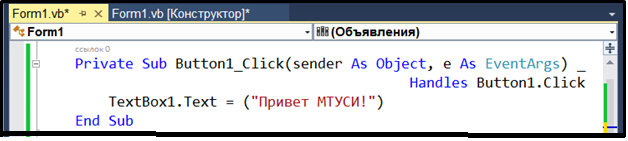


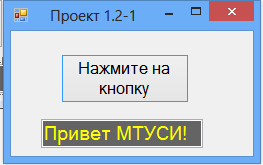
Рис.4.1.2-8. Фрагмент окна ***Редактор кода***

Для того чтобы написать код обработки события необходимо:

* кликнуть дважды по кнопке – в результате откроется окно **Form1.vb\*** - **Редактор программного кода** (рис. 4.1.2-8), (на что указывает соответствующая вкладка);
* ввести необходимый программный код обработки события   
  (рис. 4.1.2-9);
* вернуться в окно **Конструктора форм** - щелкнув по вкладке   
  **Form4.1.vb [Конструктор]**;
* повторить вышеперечисленные действия для остальных кнопок, если они есть.



*Рис. 4.1.2-9.Фрагмент окна* **Редактор кода**



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

1. **Выполнить компиляцию и запуск программы.**

Для этого необходимо:

* нажать на клавишу <F5> клавиатуры или выполнить команду

**Начать отладку** элемента **Отладка** Стандартного меню.

После нажатия на кнопку на экране появится результат работы

проекта (рис. 4.1.2-10).

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

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

В общем виде **Приложения Windows Forms** 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 Modulе Modulе1**  **…**  **End Modulе**  **…**  **Public Modulе Modulе\_1**  **…**  **End Modulе**  **Public Modulе Modulе\_n**  **…**  **End Modulе**  **…**  **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 в VB всегда является процедура с именем **Main()**. Платформа Framework .NET автоматически вызывает процедуру **Main()** при загрузке приложения. Процедура **Main()** содержит программный код, который будет выполняться в первую очередь. В этой процедуре пользователь может определить, какая форма загружается первой при запуске программы, установить, 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 пользователь проекта. Когда форма спроектирована и заполнена элементами управления, то пользователю остается вводить собственные данные в поля формы, нажимать на кнопки, вообще быть инициатором возникновения различных событий 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.

Выполняемый 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()**.

**В Приложении** **Windows Forms** процедура **Main()** создается автоматически. 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 **Основного окна**, которое можно открыть, щелкнув на ***MyProject*** в окне **Обозреватель решений**.

Таким образом, проект VB построен из модулей. 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 конечно, содержит весь программный 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.

Как нам уже известно, при создании проекта и открытии окна   
**Редактора кода**, пользователю предоставляется уже готовая часть кода, расположенная в правильном порядке. Любой программный 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** 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, Public и др., находящиеся внутри модулей или процедур; процедуры, генерируемые средой разработки внутри модулей формы; процедуры, написанные программистом, которые могут находиться внутри любых модулей.

Классы и модули содержат весь программный 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.

**Элементы программирования уровня модуля.** Процедуры, операторы, свойства и события являются единственными элементами программного кода, которые могут содержать исполняемый код (то есть выполняются во время выполнения). Они являются элементами уровня модуля ([операторы **Function**](ms-help://MS.VSCC.v90/MS.MSDNQTR.v90.ru/dv_vbalr/html/a4497077-0f46-4ede-a27f-9e8670df52b9.htm)и[**Sub**](ms-help://MS.VSCC.v90/MS.MSDNQTR.v90.ru/dv_vbalr/html/e347d700-d06c-405b-b302-e9b1edb57dfc.htm)**).**

Элементами данных на уровне модуля являются переменные и константы.

**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**).

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 (класса), до описания процедур пользователей и процедур событий, и тем самым при­дать переменным более широкую область видимости, распространяющу­юся на весь класс (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 **Form2** вызвать процедуру **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.

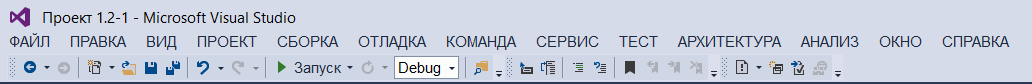
Таким образом, чтобы эффективно управлять элементами, используемыми на этапе разработки, например, ссылками, подключениями данных, папками и файлами, в VS предусмотрены два типа ***контейнеров***. Эти контейнеры называются ***решениями*** и ***проектами***. Также, VS предоставляет папки решений для того, 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.

В **Обозревателе решений** элементы могут быть сгруппированы следующими способами:

* в виде элементов проекта, которые являются элементами, составляющими данный проект, например, формами, исходными файлами и классами в пределах проекта в Обозревателе решений;
* 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.

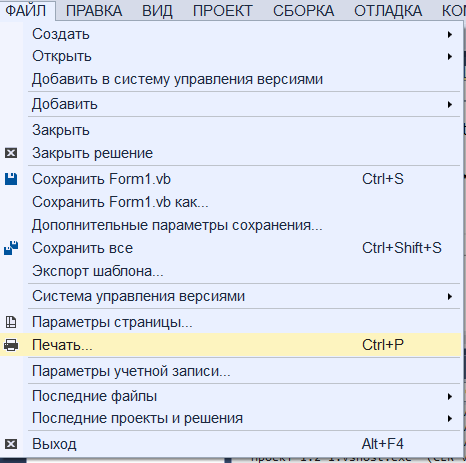
### 4.1.3. Главное меню VS и стандартная панель инструментов

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



*Рис. 4.1.3-1. Главное меню VS и стандартная панель инструментов*

**Главное меню** может содержать следующие элементы: **Файл, Правка, Вид, Проект, Сборка, Отладка, Команда, Сервис, Тест, Архитектура, Анализ, Окно и** **Справка.**



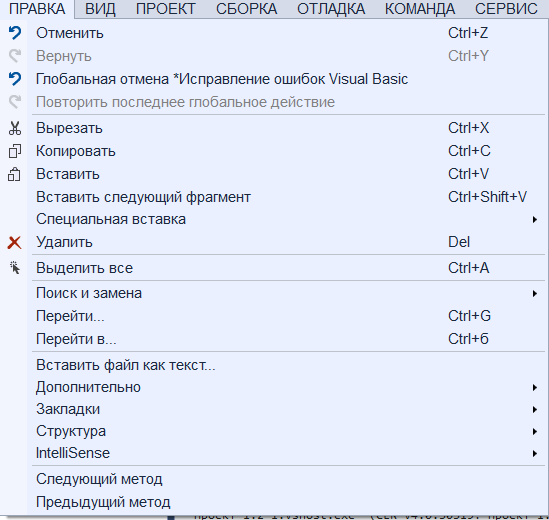
*Рис. 4.1.3-2. Команды элемента Главного меню* **Файл**

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, Построение, Отладка.** Настроить отображение элементов главного меню можно с помощью диалогового окна ***Настроить*,** которое открывается командой ***Настройка*** элемента Главного меню **Сервис.**

Так, например, элемент Главного меню **Файл** содержит команды, связанные с доступом к файлам (рис. 4.1.3-2).

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

Элементы Главного меню **Вид** содержит команды вызова окон среды**.** С помощью этих команд могут открываться окна редактора программного кода, конструктора формы, свойств объектов, обозревателя решений и другие окна.



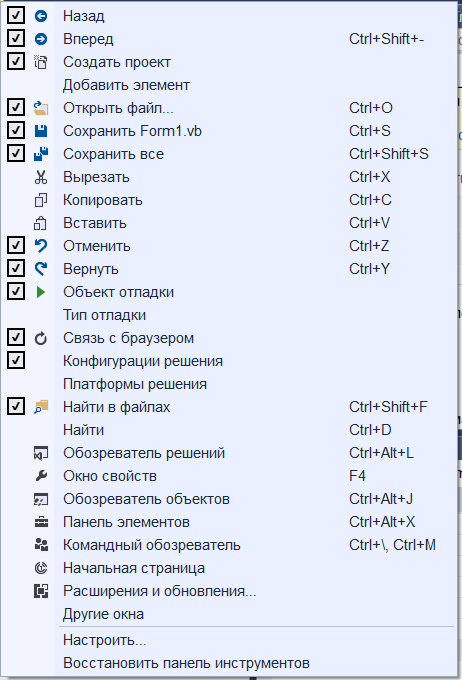
*Рис. 4.1.3-3. Команды элемента главного меню* **Правка**

Элемент главного меню **Проект** 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.

Последней командой элемента меню **Проект** является команда ***Свойства: ...,*** позволяющая открыть окно свойств проекта.

Элемент главного меню Сборкасодержит команды, помогающие скомпоновать решение или проект.

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



*Рис. 4.1.3-4.* ***Стандартной панелью инструментов***

Элемент главного меню **Сервис** содержит средства для настройки среды разработки, создания макросов, а также команды запуска дополнительных утилит.

Элемент главного меню **Окно** содержит команды, которые управляют открытыми на экране окнами. 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.

Элемент главного меню **Окно** содержит команды, которые управляют открытыми на экране окнами. 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.

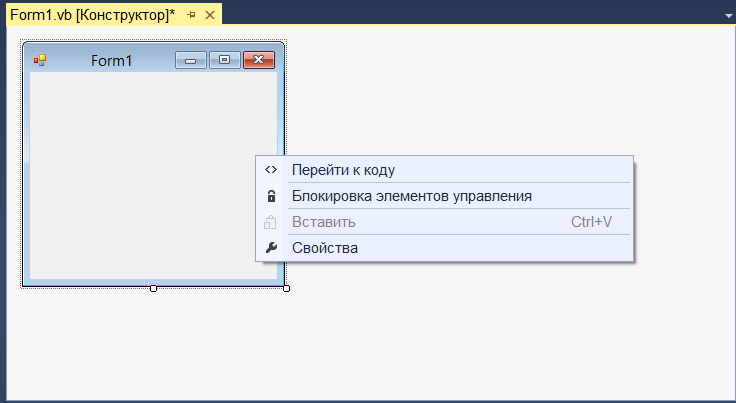
Элемент главного меню **Справка** использует свои команды для вызова справочной системы с различными вариантами представления информации.

В VBсодержится большое количество панелей инструментов для отладки и запуска программ, задания расположения элементов на форме и многого другого. Познакомимся со **Стандартной панелью инструментов** (рис. 4.1.3-1), которая используется во всех режимах работы. Назначение кнопок Стандартной панели инструментов показано на рис. 4.1.3-4.

По умолчанию в Главном окне VS всегда присутствует **Стандартная панель инструментов**. Если для работы необходима стандартная панель инструментов, а ее нет, выберите команду **Стандартная**элемента меню **Панели инструментов** из элемента главного меню **Вид.**

### 4.1.4. Окна панелей Visual Studio

Окно панели **Начальная страница** (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.



*Рис. 4.1.4-1.Окно панели* ***Конструктор форм*** *и ее контекстное меню*

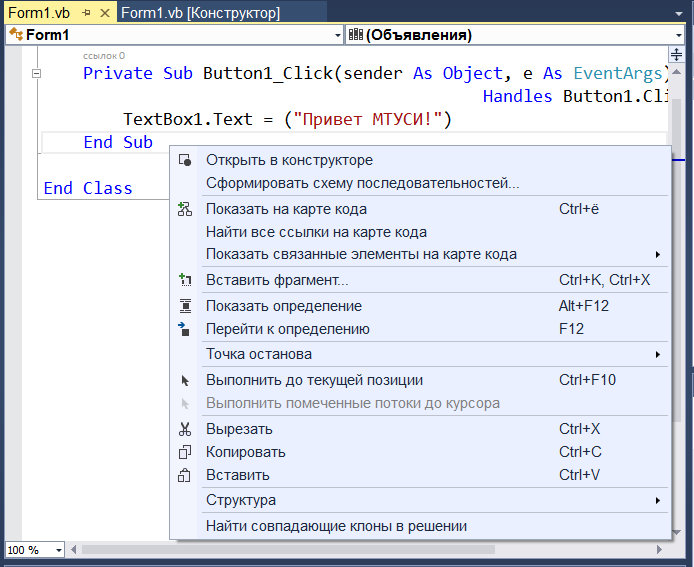
Начальная страница автоматически открывается при запуске VS**.** Если же окно **Начальная страница** не появилось, его можно вызвать с помощью команды **Начальная страница** элемента главного меню **Вид (View).**

Окно панели **Form1.vb**[**Конструктор] (Конструктор форм)   
Основного окна** является ***Основным рабочим окном***, в котором выполняется визуальное проектирование приложения (рис. 4.1.4-1). Отобразить на экране это окно можно щелкнув на вкладке **Form1.vb**[**Конструктор]** или двойным щелчком на названии формы в окне **Обозреватель решений.**

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.

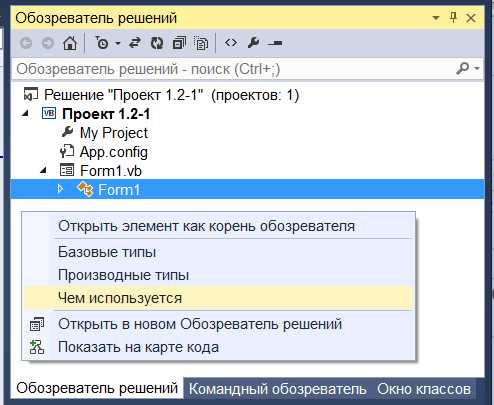
Для работы в окне **Form1.vb**[**Конструктор]** можно использовать контекстное меню (рис. 4.1.4-1).



*Рис. 4.1-2. Окно панели* **Редактора кода***и ее и контекстное меню*

Окно панели **Form1.vb** (**Редактор кода)** – это мощный текстовый редактор с большим количеством возможностей, являющийся основным инструментом программиста для создания и отладки программного кода. Оно также может располагать в **Основном рабочим окне**(рис.4.1-2)***.***

Окно панели **Обозреватель решения и** ее контекстное меню представлено на рис. 4.1.4-3.



*Рис. 4.1.4-3. Окно панели* ***Обозреватель решений*** *и ее контекстное меню*

Панель **Элементов управления** сгруппирована по разделам   
(Рис. 4.1.4-4). Для создании форм пособия необходим в основном раздел ***Стандартных******элементов управления* (**рис. 4.1.4-5).

Окно панели **Свойства** 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 и другие (рис. 4.1.4-6).

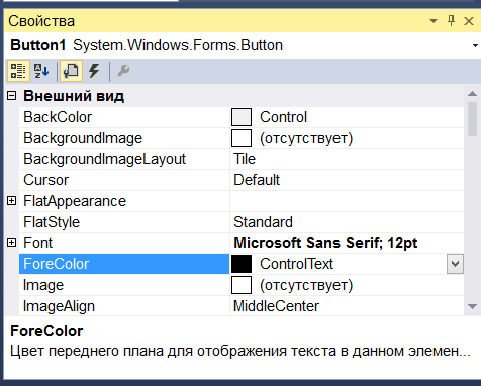
Для открытия диалогового окна **Свойства** следует выполнить одно из следующих действий:

* в элементе главного меню **Вид** выбрать команду **Окно Свойств**;
* нажать кнопку Окно свойств, расположенную на Стандартной   
  панели инструментов;
* выбрать команду **Свойства** контекстного меню выделенного объекта;
* нажать клавишу клавиатуры <F4>.

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

*Рис. 4.1.4-4. Окно панели элементов 4.1.4-5. Панель*

*управления, сгруппированных по разделам (Стандартные элементы)*



*Рис. 4.1.4-6. Окно панели* ***Свойства***

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*** и   
***По категориям*** свойства объекта можно просмотреть в алфавитном порядке или по группам (категориям) соответственно.

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

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

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

### 4.1.5. Настройка параметров интегрированной среды разработки

Окна инструментов, справочная система, параметры компилятора среды разработки VSлегко и очень гибко настраиваются.

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

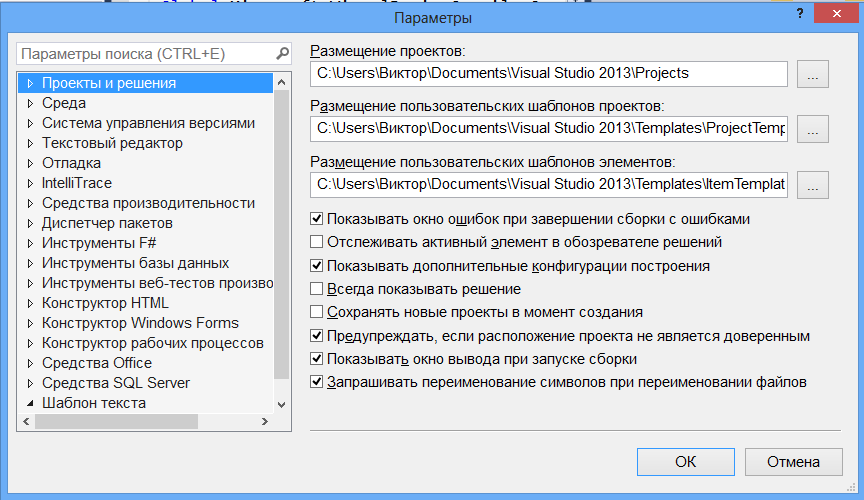
1. В элементе главного меню **Сервис** выполнить команду **Импорт и экспорт параметров**. Для сохранения настроек среды с целью их использования на другом компьютере, загрузки настроек с другого компьютера или сброса настроек – опции, которую требуется выбрать сейчас – можно исполь­зовать появляющийся при этом мастер.
2. Выбрать **Сбросить все настройки**, а затем щелкнуть на кнопку **Далее**. 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.
4. Если необходимо вернуться к этим настройкам, для их восстановления используется этот же мастер и опция **Импорт** выбранных настроек среды.
5. Чтобы увидеть список по умолчанию параметров, используемых для VS, следует щелкнуть на кнопке **Далее**.
6. Щелкнуть на **Параметры для разработки на VB**, а за тем щелкнуть на кнопке **Готово**.
7. Мастер переключит параметры IDE, включая команды меню, панели инструментов и настройки некоторых диалоговых окон, окон **Инструментов** и **Редактор кода**.
8. 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 для использования другого инструмента программирования.
9. Чтобы закрыть мастер, щелкнуть на кнопке **Закрыть**.

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

1. Щелкнуть на команде ***Параметры*** элемента меню **Сервис** и открыть диалоговое окно **Параметры.**
2. Диалоговое окно **Параметры** – это окно, в котором находятся многие из настроечных параметров VS. Чтобы увидеть все параметры, которые можно изменить, надо щелкнуть на флажке **Показать все параметры**, расположенном в нижнем левом углу диалогового окна.
3. В диалоговом окне **Параметры** следует щелкнуть на категории ***Проекты и решения*** и затем на разделе ***Общие*.** Эта группа флажков и опций настраивает параметры проектов и решений VS. Чтобы программное обеспечение соответствовало параметрам, использованным в этом пособии необходимо настроить их так, как показано на   
   рис. 4.1.5-1.

Это «***отложенное***» сохранение позволяет создавать тестовые программы, компилировать и отлаживать программы, и даже запускать их, не сохраняя проект на диске, 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.***

1. Необходимо проверить, что опция **Option Explicit** установлена в значение **On**, a **Option Strict** – в значение **Off** – значения по умолчанию для программирования на VB в VS. **Option Explicit On** – это параметр, который требует явного объявления переменной до ее первого использования в программе – это хороший стиль программирования. **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.
2. Можно изучить другие параметры диалогового окна **Параметры**, связанные со средой программирования и VB. Когда закончится установка параметров необходимо щелкнуть на ОК, чтобы закрыть диалоговое окно **Параметры.**



*Рис. 4.1.5-1. Диалоговое окно* **Параметры**

### 4.1.6. Тестовые задания

1. **Проект - это**
2. разрабатываемое приложение, состоящее из набора файлов, в которых хранится
3. информация обо всех компонентах, используемых в приложении
4. семейство процедур
5. набор модулей
6. множество форм
7. **Проект VB хранится в файле с расширением**
8. **.frm**
9. **.vbproj**
10. **.bas**
11. **.fix**
12. **В окне Обозреватель решений отображаются**
13. форма и элементы управления
14. проекты и файлы текущего решения
15. модули и процедуры
16. программный код
17. **Окно Конструктор форм служит для**
18. создания форм приложения с использованием инструментария среды

разработки кода

1. конструирования модулей
2. написания программного кода
3. **Свойства объекта в среде Visual Basic можно изменить**
4. только на этапе проектирования интерфейса
5. только на этапе кодирования текста программы
6. и на этапе проектирования интерфейса, и на этапе кодирования текста   
   программы
7. **Форма – это**
8. философское понятие, противоположное "содержанию" программы
9. объект, появляющийся на экране при запуске Visual Basic и предназначенный для ввода данных
10. объект Visual Basic, появляющийся на экране при запуске программы, и на котором размещаются элементы управления
11. объект, появляющийся на экране при запуске программы и предназначенный для вывода результатов работы программы
12. **Элемент управления – это**
13. объект, с помощью которого программа запускается на выполнение
14. объект, являющийся элементом графического интерфейса приложения и реагирующий на события
15. инструмент, который используется для создания объектов на форме VB
16. объект, появляющийся на экране при запуске программы и предназначенный для останова программы
17. **Текстовые поля могут быть использованы для**
18. заполнения окна формы
19. вывода информации на экран и для получения данных от пользователя
20. только для вывода информации на экран
21. только для получения данных от пользователя
22. **В файле формы (с расширением. vb) хранится**
23. форма
24. программный код, связанный с формой
25. данные об объектах, расположенных на форме
26. данные о форме, объектах, расположенных на ней, и программный код, связанный с формой
27. **Оператор программы Visual Basic – это**
28. строка кода программы, завершенная конструкция, исполняемая компилятором
29. присваивание, For…Next, If
30. арифметические операции
31. ключевые слова
32. **Свойство в Visual Basic – это**
33. особый элемент управления
34. имя текстового поля
35. окно Свойства (Properties)
36. значение или характеристика, хранимая объектом
37. **Программный код проекта**
38. существует сам по себе
39. привязан к отдельным объектам и не оторван от формы
40. привязан к отдельным объектам и не связан с формой
41. привязан только к форме
42. **Событие в Visual Basic – это**
43. процесс получение данных
44. любые изменения, инициируемые пользователем или систе­мой, на которые может реагировать программа
45. выход из строя монитора
46. ситуация, при которой ничего не происходит
47. **Процедура события – это**
48. блок кода, который исполняется тогда, когда с соответствующим объектом программы производятся какие-либо действия
49. блок кода, который исполняется тогда, когда происходит щелчок на объекте Button
50. блок кода, который исполняется тогда, когда происходит щелчок на объекте TextBox
51. **Именами свойств объ­ектов являются**
52. Text
53. Integer
54. Date
55. Name
56. String
57. Single
58. **В про­грамме при обращении к объекту указывается свойство**
59. Alignment
60. Text
61. Name
62. Enabled.

### 4.1.7. Лабораторная работа по теме «*Интегрированная среда разработки Visual Studio .NET и основные понятия*»

**Цель** выполнения лабораторной работы состоит в изучении интегрированной среды разработки Visual Studio .NET (VS), особенностей технологии визуального программирования, технологии создания интерфейса пользователя, установки свойств объектов, процесса создания программного кода, выполнения приложения, создания исполняемого файла, внесения изменений в приложение.

### Вопросы, подлежащие изучению

1. Инструментальные средства среды разработки VS.
2. Создание нового проекта.
3. Создание интерфейса пользователя.
4. Установка свойств объектов.
5. Создание программного кода.
6. Выполнение приложения.

### Общее задание на выполнение лабораторной работы (на разработку проекта)

Выполнить все шаги из п. 4.1.7.5 «Создание первого приложения (проекта)».

### Варианты индивидуальных заданий

Все студенты выполняют один вариант, соответствующий п. 4.1.7.5.

### Содержание отчета

1. Тема и название лабораторной работы.
2. Фамилия, имя студента, номер группы, номер варианта.
3. Задание на разработку проекта (условие задачи).
4. Формализация и уточнение задания.
5. Элементы разрабатываемого проекта:
   1. процесс создания нового проекта;
   2. графический интерфейс пользователя;
   3. таблица свойств объектов;
   4. схема алгоритма решаемой задачи;
   5. программный код проекта.
6. Результаты выполнения проекта, представленные в виде скриншота.

### Пример выполнения задания

1. **Тема и название лабораторной работы:**

Интегрированная среда разработки *VS* и основные понятия***.***

Создание первого простого проекта (приложение).

1. **Фамилия, имя студента, номер группы, номер варианта:**

Иванов Иван, БИН1404, № 13.

1. **Задание на разработку проекта (условие задачи):**

Создать приложение (проект), которое будет выводить на форму надпись «**Я работаю в среде VS**» на элемент управления **Label** после нажатия на кнопку **Button** с названием **Пуск**.

1. **Формализация и уточнение задания:**

Приложение собирается из многих элементов: форм, элементов управления, программных модулей, процедур, методов, внешних библиотек, картинок, пиктограмм и т.д. Набор всех файлов, необходимых для создания приложения, называется **проектом**. VS обрабатывает все файлы проекта и строит из них выполняемый файл.

1. **Элементы разрабатываемого проекта:**
   1. Процесс создания нового проекта:

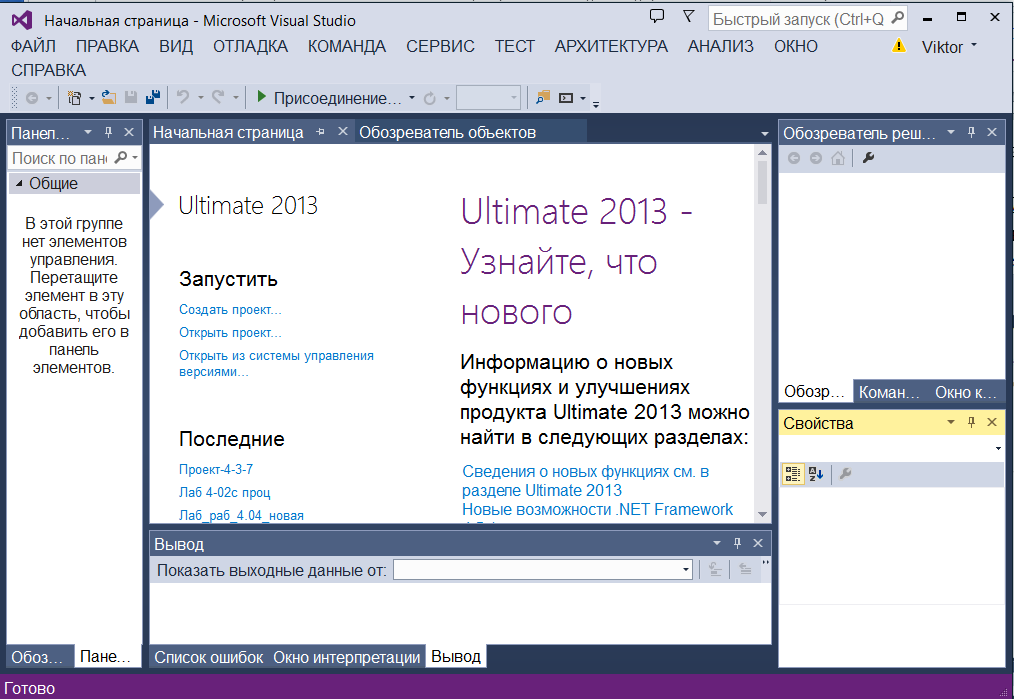
Создайте новый проект в среде VS:

Для этого найдите на рабочем столе Windows иконку VS и   
 щелкните по ней дважды.

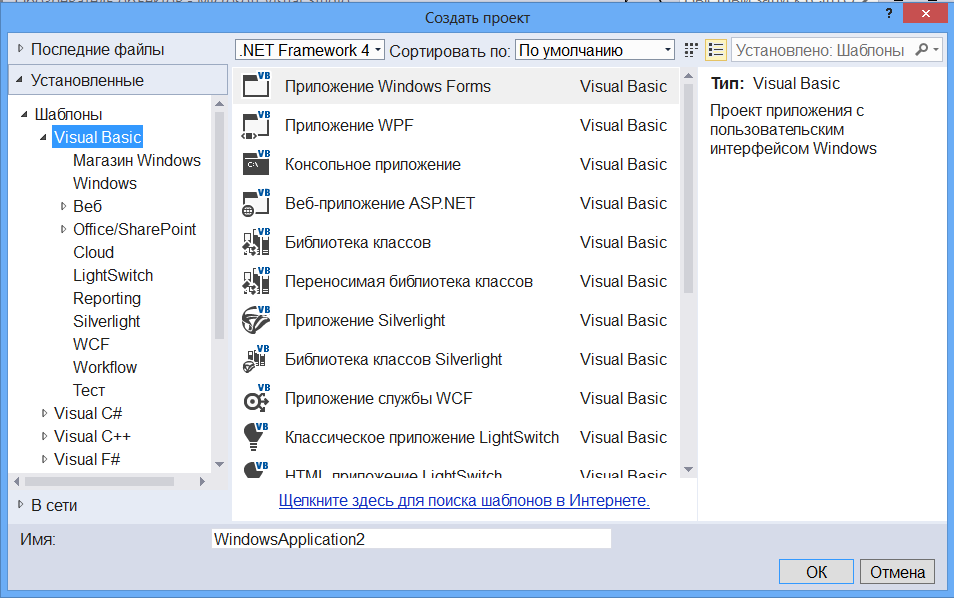
Откроется **Начальная страница** IDE, представленная на рис.

П.4.1.1-1.

Щелкните по ссылке ***Создать проект***…. Откроется окно   
**Создать проект**, представленное на рис. П.4.1.1-2. В этом окне из установленных шаблонов выберите **Visual Basic**, а тип приложения – **Приложение Windows Forms** (Проект приложения с пользовательским интерфейсом).



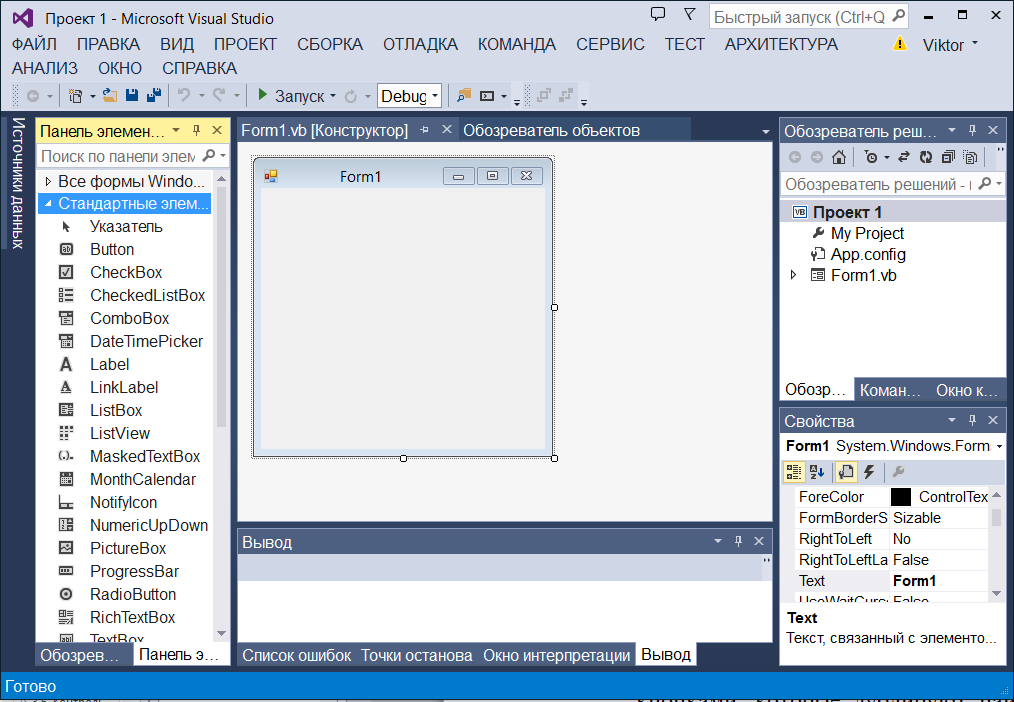
*Рис. 4.1.7-1.* **Начальная страница**VS



*Рис. 4.1.7-2. Окно VS* **Создать проект**

В поле для ввода **Имя** введите имя нашего проекта, например,

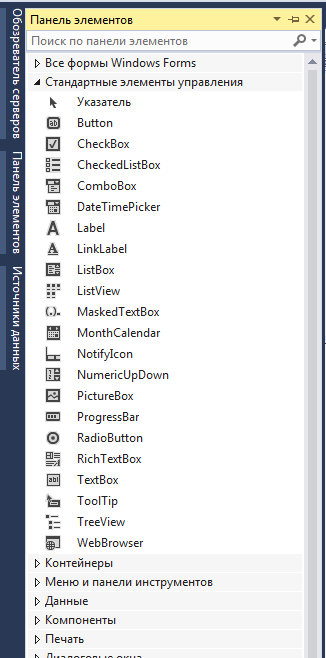
**Проект 1**, а затем нажмите кнопку OК. Откроется **Основное окно среды**, в котором создаются все приложения (см. рис. 4.1.7-3).



*Рис. 4.1.7-3. Основное окно среды VS*

Рассмотрим структуру этого окна. В верхнем левом углу окна располагается имя вашего проекта (**Проект 1**). Ниже находится **Главное меню** среды, а ниже **Стандартная панель** с расположенными на ней кнопками, которые дублируют наиболее часто используемые команды элементов **Основного меню**.

Основное пространство занимает **Главное окно**, в котором осуществляется работа с **Конструктором форм**, **Редактором кода** и другими окнами. После создания проекта в этом окне появится заготовка формы (**Form1**), размеры которой могут изменяться с помощью маркеров. В процессе создания приложения вам будут необходимы несколько дополнительных окон – это окно **Обозреватель решений**, в котором можно видеть структуру разрабатываемого проекта и окно **Свойств**а, предназначенное для установки свойств различных объектов проекта, таких как **Font** (Шрифт), **Text** (Текст), **ForeColor** (Цвет) и т.д.



*Рис. 4.1.7-4. Окно* **Панель Элементов** *управления*

Слева от формы расположена **Панель элементов** управления   
(рис. 4.1.7-4), с помощью, которой можно переносить элементы управления на форму. Используя этот набор и **Редактор форм (*Имя Формы*.vb[Конструктор]**), можно создать любой пользовательский интерфейс, который будет удовлетворять всем требованиям, предъявляемым к интерфейсу в среде Windows.

Напомним, что элементы управления являются объектами, поэтому, как любые объекты, они обладают **свойствами, методами** и **событиями**.

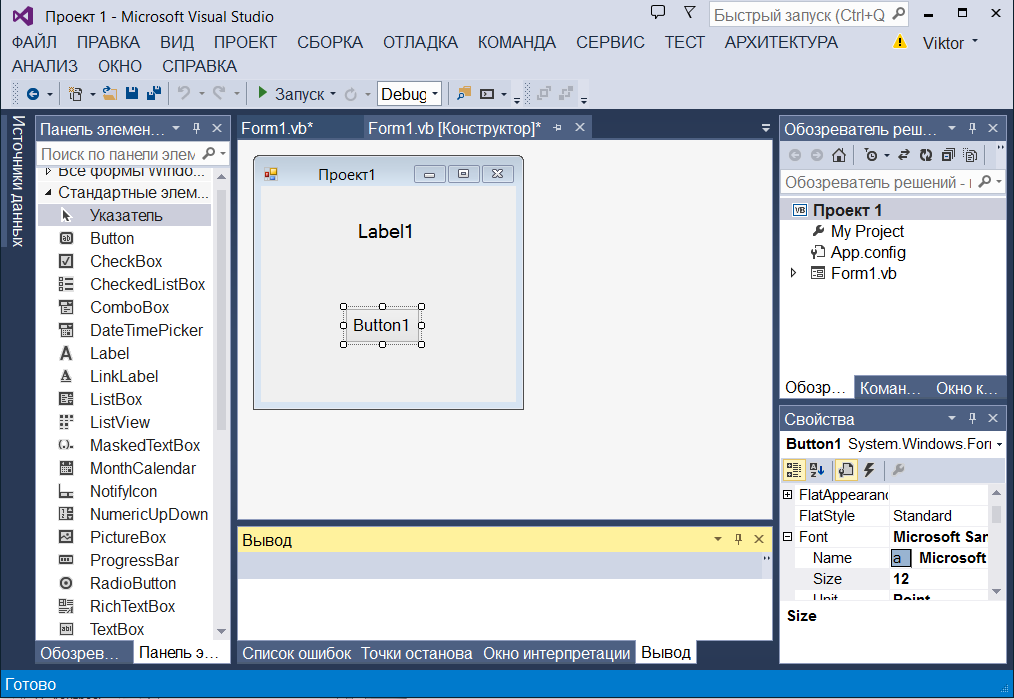
Следует отметить, что если одно из перечисленных выше окон вы случайно закроете, то вновь получить доступ к нему, можно с помощью команд элемента **Главного меню** – **Вид**;

* 1. Графический интерфейс пользователя:

Графические интерфейсы пользователя управляются событиями – они генерируют события в ответ на взаимодействие пользователя программы с графическим интерфейсом. Информация о событиях передается обработчикам событий, которые являются методами, вызываемыми в ответ на возникновение конкретных событий.

Для реализации задания понадобятся два экземпляра элементов управления: **Button1** (Кнопка1) и **Label1** (Надпись1). Стоит заметить, что каждый экземпляр элемента управления имеет свой порядковый номер, отображаемый рядом с именем элемента управления.

Выберите с помощью мышки элемент **Button**, расположенный на **Панели элементов** и перенесите его на форму, а затем измените ее размеры. В произвольное место формы перенесите элемент управления **Label**. Результат этих действий представлен на   
рис.4.1.7-5;



*Рис. 4.1.7-5. Создание формы проекта* **Проект 1**

* 1. Таблица свойств объектов:

Определите и сведите в табл. 4.1.7-1 свойства всех объектов;

Таблица 4.1.7-1

|  |  |  |  |
| --- | --- | --- | --- |
| **Объект** | **Имя объекта** | **Свойство** | **Значение свойства** |
| **Форма** | **Form1** | Name | *Проект1* |
| Text |  |
| **Метка** | **Label1** | Text | *Label1* |
| ForeColor |  |
| ImageAlign | *MiddleCenter* |
| **Кнопка** | **Button1** | Name | *Button1* |
| Text | *Button1* |
| Font | *Arial, жирный, 12* |

* 1. Схема алгоритма решаемой задачи:

Схема алгоритма решения задачи представлена на рис. 4.1.7-6.

|  |
| --- |
|  |

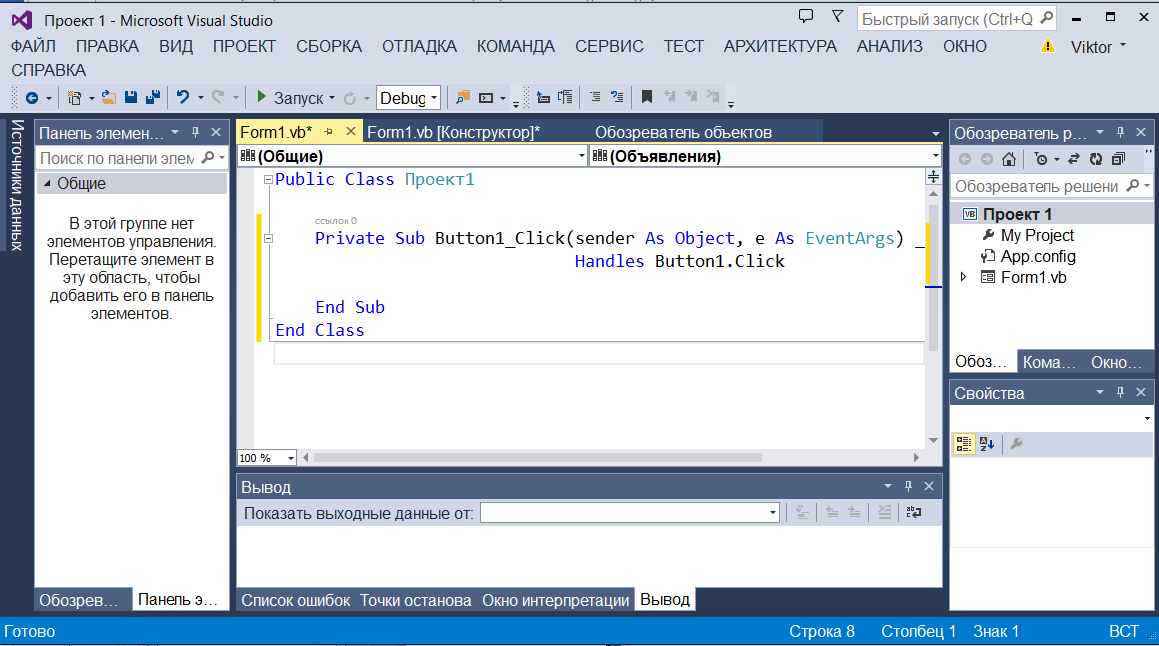
*Рис. 4.1.7-6. Схема алгоритма событийной процедуры*

## Все схемы лабораторных работ должны быть выполнены в графическом редакторе Visio MS Office (правила работы с Visio приведены в Приложение 2. Основы работы с графическим редактором MS Visio

* 1. Программный код проекта:

Выполните двойной щелчок на кнопке **Button1.**

Откроется окно **Редактора программного кода** (рис. 4.1.7-7).



*Рис. 4.1.1-7. Заготовка программного кода процедуры события*

Рассмотрим шаблон процедуры обработчика события **Нажатие кнопки Button1**, который генерируется автоматически при создании этой кнопки на форме (рис. 4.1.7-8).

|  |
| --- |
| **Private Sub Button1\_Click(sender As Object, e As EventArgs)\_**  **Handles Button4.1.Click**  *'программа обработки события Button1\_Click*  **End Sub** |

*Рис. 4.1.7-8. Заготовка программного кода процедуры* **Button1\_Click()**

После щелчка по **Button1** среда VB создала сигнатуру метода обработчика события **Click**, генерируемой элементом **Button1.** Обработчикам события передаются два объекта: ссылка на объект, который генерирует событие **Sender**, и объект аргументов события **e**. Аргумент **e** имеет тип **EventArgs**. Класс **EventArgs** является базовым классом для объектов, которые содержат информацию о событии.

Имя обработчика события по умолчанию совпадает с именем управляющего элемента, за которым следуют символ подчеркивания и имя события. Обработчики (процедуры) события являются методами, которые принимают два аргумента: **Object** (обычно **Sender**) иэкземпляр класса **EventArgs**. **Handles** трактуется как управление чем-либо.

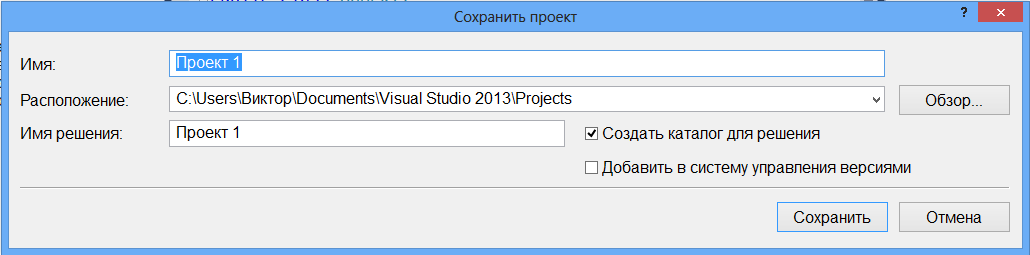
Допишите одну строку программного кода так, как это представлено на рис. 4.1.7-9.

|  |
| --- |
| **Private Sub Button1\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button4.1.Click**  **Label1.Text = "Я работаю в среде Microsoft VS"**  **End Sub** |

*Рис. П. 1.7-9. Полный программного кода процедуры* **Button1\_Click()**

1. **Выполнение приложения.**

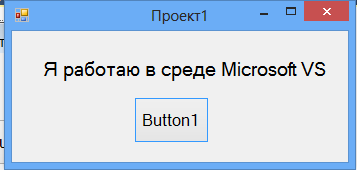
Необходимо заметить, что перед запуском программы на выполнение необходимо сохранить проект. Для этого на **Стандартной панели** нажмите кнопку ***Сохранить все***. После этого на экране всплывет окно **Сохранить проект (**рис. 4.1.7-10), в котором можно задать новое имя проекта, его расположение. При этом сохранятся изменения, сделанные в проекте: в программном коде и интерфейсе проекта.



*Рис. 4.1.7-10. Окно* **Сохранить проект** *VS*

Для запуска программы на выполнение нажмите клавишу <F5> клавиатуры или выполните команду ***Запуск*** на **Стандартной панели**. После отображения на экране формы выполните щелчок на кнопке **Button4*.***

Результат выполненной проекта отображен на рис.4.1.7-11



*Рис. 4.1.7-11. Результат* *выполнения проекта* **Проект 1**

### Контрольные вопросы

1. Что представляет собой технология визуального программирования?
2. Что представляет собой Интегрированная среда разработки?
3. Что такое Объект в среде визуального программирования?
4. Что такое Класс в среде визуального программирования?
5. Что представляет собой коллекция объектов в среде визуального   
   программирования?
6. Что такое событие в Visual Basic?
7. Что такое метод в Visual Basic?
8. Что такое оператор в Visual Basic?
9. Что представляет собой проект в Visual Basic?
10. Что такое расширение файла проекта VB?
11. Что такое окно обозреватель решений?
12. Что такое диалоговое окно Просмотр объектов?
13. Что такое окно Редактора программного кода проекта?
14. Что такое расширение файла программного кода VB?
15. Что такое диалоговое окно Конструктор форм?
16. Что такое окно создания нового проекта?
17. Что такое свойства объекта в среде Visual Basic?
18. Что такое состав пользовательского ин­терфейса?
19. Что такое процедуры обработки событий?
20. Что такое форма в Visual Basic?
21. Что такое текстовые поля?
22. Что такое раскрывающиеся меню?
23. Что такое расширение файла формы?
24. Что представляет собой объектно-ориентированный язык   
    программирования?
25. Что такое элементы управления?
26. Что такое форма и как задавать её свойства?
27. Какие окна используются при разработке приложения?