Практическая работа 1
Изучение основ применения VisualStudio

Цель занятия

Получить практические навыки создания проектов с использование VisualStudio

Перечень оборудования и программного обеспечения

Персональный компьютер
Microsoft Office (Word, Visio)
Microsoft Visual Studio 2015

Краткие теоретические сведения

VisualStudio

Интегрированная среда разработки (IntegratedDevelopmentEnvironment — IDE) VisualStudio 2010 предлагает ряд высокоуровневых функциональных возможностей, которые выходят за рамки базового управления кодом.

Встроенный Web-сервер. Для обслуживания Web-приложения ASP.NET необходим Web-сервер, подобный IIS, который будет ожидать Web-запросы и обрабатывать соответствующие страницы. Установить свойWeb-сервер несложно, но может быть не совсем удобно. Наличие в VisualStudio интегрированного Web-сервера позволяет запускать Web-сайт прямо из среды проектирования, а также повышает безопасность.

Поддержка множества языков при разработке. VisualStudio позволяет писать код на своем языке или любых других предпочитаемых языках, используя все время один и тот же интерфейс (IDE).

Меньше кода для написания. Для создания большинства приложений требуется достаточное количество стандартного стереотипного кода. В VisualStudio такой код генерируется автоматически.

Интуитивный стиль кодирования. По умолчанию VisualStudio форматирует код по мере его ввода, автоматически вставляя необходимые отступы и применяя цветовое кодирование для выделения элементов типа комментариев. Такие незначительные отличия делают код более удобным для чтения и менее подверженным ошибкам. Многие из функциональных возможностей VisualStudio направлены на то, чтобы помогать разработчику делать свою работу как можно быстрее. Удобные функции, вроде функции IntelliSense (которая умеет перехватывать ошибки и предлагать правильные варианты), функции поиска и замены (которая позволяет отыскивать ключевые слова как в одном файле, так и во всем проекте) и функции автоматического добавления и удаления комментариев (которая может

временно скрывать блоки кода), позволяют разработчику работать быстро и эффективно.

Возможности отладки. Предлагаемые в VisualStudio инструменты отладки являются наилучшим средством для отслеживания загадочных ошибок и диагностирования странного поведения. Разработчик может выполнять свой код по строке за раз, устанавливать интеллектуальные точки прерывания, при желании сохраняя их для использования в будущем, и в любое время просматривать текущую информацию из памяти.

Microsoft .NET Framework

.NET Framework – среда для создания и запуска приложений. Все приложения не просто используют библиотеки .NET Framework, но и выполняются под управлением ее компонент.

Её основные компоненты:

- общеязыковая исполняющая среда (CommonLanguageRuntime) – динамический компонент;
- библиотека классов .NET FrameworkClassLibrary – статический компонент.

Пространство имен

.NET Framework располагает большим набором полезных функций. Каждая из них является членом какого-либо класса. Классы группируются по пространствам имен. Это означает, что в общем случае имя класса может иметь сложную структуру — состоять из последовательности имен, разделенных между собой точками. Последнее имя в этой последовательности собственно и является именем класса. Классы, имена которых различаются лишь последними членами (собственно именами классов) последовательностей, считаются принадлежащими одному пространству имен.

Помещение типа в пространство имен присваивает этому типу длинное имя, состоящее из всех пространств, как серии их имен, разделенных точками (.) и заканчивающееся именем класса.

Если не использовать оператор using, для корректного обращения к функциям необходимо писать полный путь класса, что является достаточно емкой работой.

Средством "навигации" по пространствам имен, а точнее, средством, которое позволяет сокращать имена классов, является оператор

using<ИмяПространстваИмен>;

В приложении может объявляться собственное пространство имен, а также могут использоваться ранее объявленные пространства.

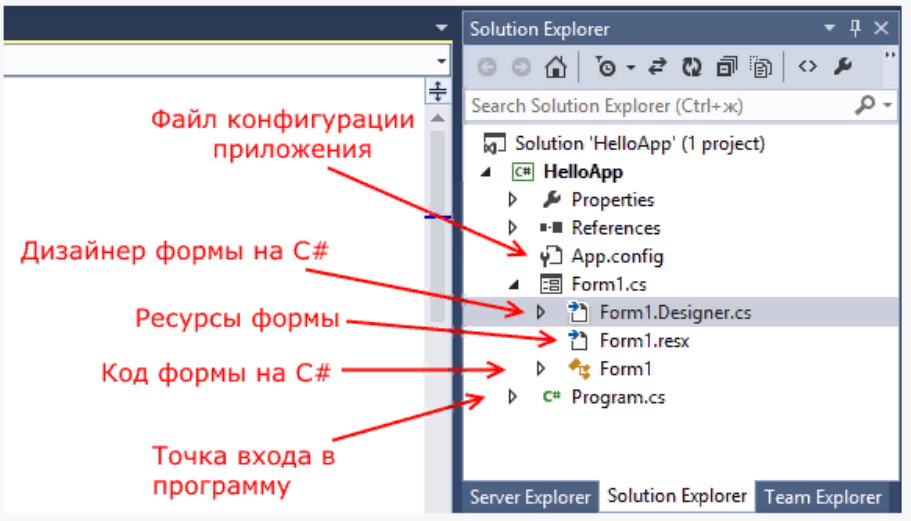
Оператор using сам по себе не обеспечивает доступа к именам, находящимся в других пространствах имен. До тех пор, пока код из пространства имен не будет каким-либо способом привязан к нашему проекту (например, описан в исходном файле проекта или описан в каком-либо коде), привязанному к этому проекту, мы не получим доступа к содержащимся в нем именам.

Основы форм

Внешний вид приложения является нам преимущественно через формы. Формы являются основными строительными блоками. Они предоставляют контейнер для различных элементов управления. А механизм событий позволяет элементам формы отзываться на ввод пользователя, и, таким образом, взаимодействовать с пользователем.

При открытии проекта в Visual Studio в графическом редакторе мы можем увидеть визуальную часть формы - ту часть, которую мы видим после запуска приложения и куда мы переносим элементы с панели управления. Но на самом деле форма скрывает мощный функционал, состоящий из методов, свойств, событий и прочее. Рассмотрим основные свойства форм.

Если мы запустим приложение, то нам отобразится одна пустая форма. Однако даже такой простой проект с пустой формой имеет несколько компонентов:



Несмотря на то, что мы видим только форму, но стартовой точкой входа в графическое приложение является класс Program, расположенный в файле Program.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace HelloApp
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
```

```
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Сначала программой запускается данный класс, затем с помощью выражения Application.Run(new Form1()) он запускает форму Form1. Если вдруг мы захотим изменить стартовую форму в приложении на какую-нибудь другую, то нам надо изменить в этом выражении Form1 на соответствующий класс формы.

Сама форма сложна по содержанию. Она делится на ряд компонентов. Так, в структуре проекта есть файл Form1.Designer.cs, который выглядит примерно так:

```
namespace HelloApp
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
```

```
        {
            this.SuspendLayout();
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(284, 261);
            this.Name = "Form1";
            this.Text = "Привет мир!";
            this.ResumeLayout(false);

        }

        #endregion

    }
}
```

Здесь объявляется частичный класс формы Form1, которая имеет два метода: Dispose(), который выполняет роль деструктора объекта, и InitializeComponent(), который устанавливает начальные значения свойств формы.

При добавлении элементов управления, например, кнопок, их описание также добавляется в этот файл.

Но на практике мы редко будем сталкиваться с этим классом, так как они выполняет в основном дизайнерские функции - установка свойств объектов, установка переменных.

Еще один файл - Form1.resx - хранит ресурсы формы. Как правило, ресурсы используются для создания однообразных форм сразу для нескольких языковых культур.

И более важный файл - Form1.cs, который в структуре проекта называется просто Form1, содержит код или программную логику формы:

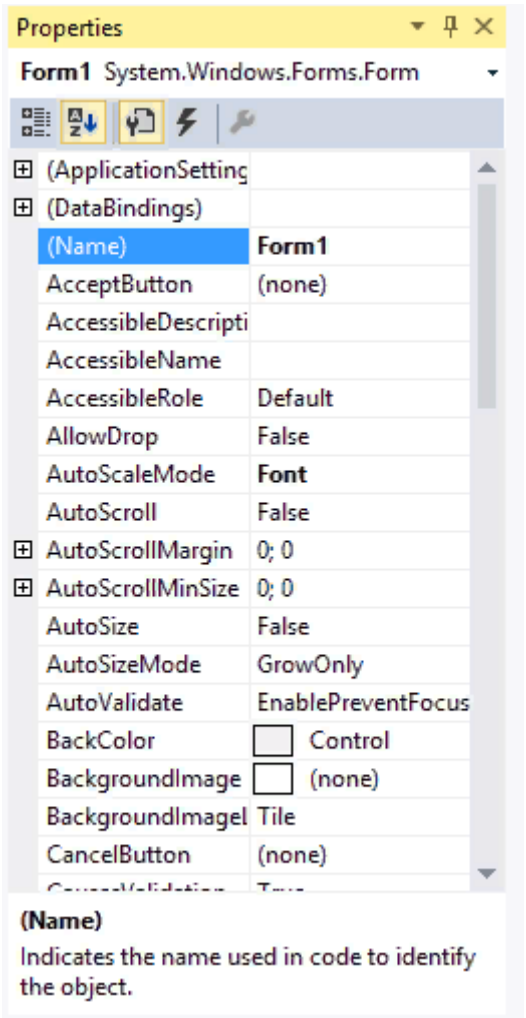
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace HelloApp
```

```
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
   }
}
```

По умолчанию здесь есть только конструктор формы, в котором просто вызывается метод InitializeComponent(), объявленный в файле дизайнера Form1.Designer.cs. Именно с этим файлом мы и будем больше работать.

С помощью специального окна Properties (Свойства) Visual Studio предоставляет нам удобный интерфейс для управления свойствами элемента:



Большинство этих свойств оказывает влияние на визуальное отображение формы. Пробежимся по основным свойствам:

Name: устанавливает имя формы - точнее имя класса, который наследуется от класса Form

BackColor: указывает на фоновый цвет формы. Щелкнув на это свойство, мы сможем выбрать тот цвет, который нам подходит из списка предложенных цветов или цветовой палитры

BackgroundImage: указывает на фоновое изображение формы

BackgroundImageLayout: определяет, как изображение, заданное в свойстве BackgroundImage, будет располагаться на форме.

ControlBox: указывает, отображается ли меню формы. В данном случае под меню понимается меню самого верхнего уровня, где находятся иконка приложения, заголовок формы, а также кнопки минимизации формы и крестик. Если данное свойство имеет значение false, то мы не увидим ни иконку, ни крестика, с помощью которого обычно закрывается форма

Cursor: определяет тип курсора, который используется на форме

Enabled: если данное свойство имеет значение false, то она не сможет получать ввод от пользователя, то есть мы не сможем нажать на кнопки, ввести текст в текстовые поля и т.д.

Font: задает шрифт для всей формы и всех помещенных на нее элементов управления. Однако, задав у элементов формы свой шрифт, мы можем тем самым переопределить его

ForeColor: цвет шрифта на форме

FormBorderStyle: указывает, как будет отображаться граница формы и строка заголовка. Устанавливая данное свойство в None можно создавать внешний вид приложения произвольной формы

HelpButton: указывает, отображается ли кнопка справки формы

Icon: задает иконку формы

Location: определяет положение по отношению к верхнему левому углу экрана, если для свойства StartPosition установлено значение Manual

MaximizeBox: указывает, будет ли доступна кнопка максимизации окна в заголовке формы

MinimizeBox: указывает, будет ли доступна кнопка минимизации окна

MaximumSize: задает максимальный размер формы

MinimumSize: задает минимальный размер формы

Opacity: задает прозрачность формы

Size: определяет начальный размер формы

StartPosition: указывает на начальную позицию, с которой форма появляется на экране

Text: определяет заголовок формы

TopMost: если данное свойство имеет значение true, то форма всегда будет находиться поверх других окон

Visible: видима ли форма, если мы хотим скрыть форму от пользователя, то можем задать данному свойству значение false

WindowState: указывает, в каком состоянии форма будет находиться при запуске: в нормальном, максимизированном или минимизированном

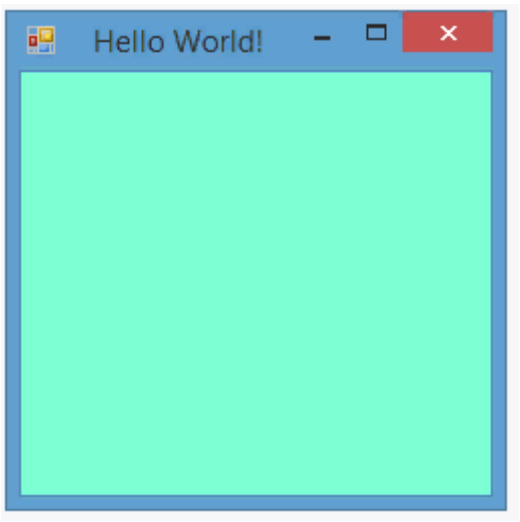
Программная настройка свойств

С помощью значений свойств в окне Свойства мы можем изменить по своему усмотрению внешний вид формы, но все то же самое мы можем сделать динамически в коде. Перейдем к коду, для этого нажмем правой кнопкой мыши на форме и выберем в появившемся контекстном меню View Code (Просмотр кода). Перед нами открывается файл кода Form1.cs.

Изменим его следующим образом:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace HelloApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            Text = "Hello World!";
            this.BackColor = Color.Aquamarine;
            this.Width = 250;
            this.Height = 250;
        }
    }
}
```



В данном случае мы настроили несколько свойств отображения формы: заголовок, фоновый цвет, ширину и высоту. При использовании конструктора формы надо учитывать, что весь остальной код должен идти после вызова метода InitializeComponent(), поэтому все установки свойств здесь расположены после этого метода.

Установка размеров формы

Для установки размеров формы можно использовать такие свойства как Width/Height или Size. Width/Height принимают числовые значения, как в вышеприведенном примере. При установке размеров через свойство Size, нам надо присвоить свойству объект типа Size:

```
this.Size = new Size(200,150);
```

Объект Size в свою очередь принимает в конструкторе числовые значения для установки ширины и высоты.

Начальное расположение формы

Начальное расположение формы устанавливается с помощью свойства StartPosition, которое может принимать одно из следующих значений:

Manual: Положение формы определяется свойством Location

CenterScreen: Положение формы в центре экрана

WindowsDefaultLocation: Позиция формы на экране задается системой Windows, а размер определяется свойством Size

WindowsDefaultBounds: Начальная позиция и размер формы на экране задается системой Windows

CenterParent: Положение формы устанавливается в центре родительского окна

Все эти значения содержатся в перечислении FormStartPosition, поэтому, чтобы, например, установить форму в центре экрана, нам надо прописать так:

```
this.StartPosition = FormStartPosition.CenterScreen;
```

Фон и цвета формы

Чтобы установить цвет как фона формы, так и шрифта, нам надо использовать цветовое значение, хранящееся в структуре Color:

```
this.BackColor = Color.Aquamarine;
this.ForeColor = Color.Red;
```

Кроме того, мы можем в качестве фона задать изображение в свойстве BackgroundImage, выбрав его в окне свойств или в коде, указав путь к изображению:

```
this.BackgroundImage = Image.FromFile("C:\\Users\\Eugene\\Pictures\\3332.jpg");
```

Чтобы должным образом настроить нужное нам отображение фоновой картинки, надо использовать свойство BackgroundImageLayout, которое может принимать одно из следующих значений:

None: Изображение помещается в верхнем левом углу формы и сохраняет свои первоначальные значения

Tile: Изображение располагается на форме в виде мозаики

Center: Изображение располагается по центру формы

Stretch: Изображение растягивается до размеров формы без сохранения пропорций

Zoom: Изображение растягивается до размеров формы с сохранением пропорций

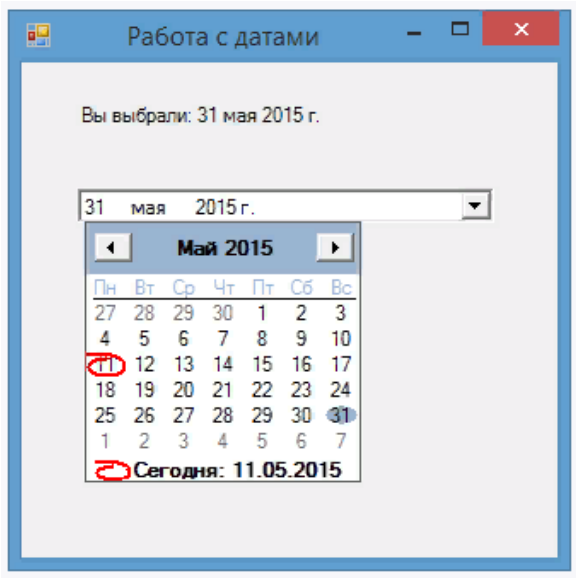
Например, расположим форму по центру экрана:

```
this.StartPosition = FormStartPosition.CenterScreen;
```

Для работы с датами в Windows Forms имеются элементы DateTimePicker и MonthCalendar.

DateTimePicker

DateTimePicker представляет раскрывающийся по нажатию календарь, в котором можно выбрать дату. собой элемент, который с помощью перемещения ползунка позволяет вводить числовые значения.



Наиболее важные свойства DateTimePicker:

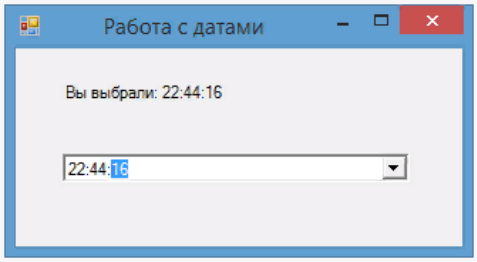
Format: определяет формат отображения даты в элементе управления. Может принимать следующие значения:

Custom: формат задается разработчиком

Long: полная дата

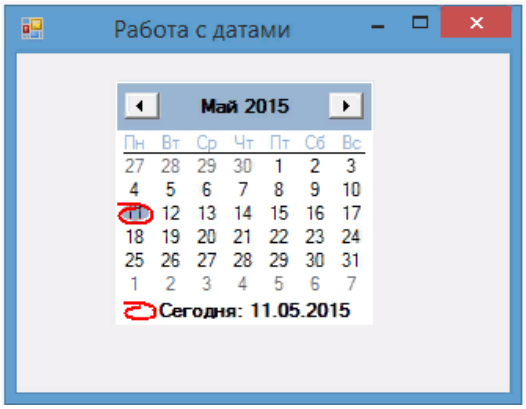
Short: дата в сокращенном формате
Time: формат для работы с временем
CustomFormat: задает формат отображения даты, если для свойства Format установлено значение Custom
MinDate: минимальная дата, которую можно выбрать
MaxDate: наибольшая дата, которую можно выбрать
Value: определеяте текущее выбранное значение в DateTimePicker
Text: представляет тот текст, который отображается в элементе

Свойство Value хранит объект DateTime, поэтому с ним можно работать как и с любой другой датой. В данном случае выбранная дата преобразуется в строку времени.



MonthCalendar

С помощью MonthCalendar также можно выбрать дату, только в данном случае этот элемент представляет сам календарь, который не надо раскрывать:



Рассмотрим некоторые основные свойства элемента.

Свойства выделения дат:

AnnuallyBoldedDates: содержит набор дат, которые будут отмечены жирным в календаре для каждого года

BoldedDates: содержит набор дат, которые будут отмечены жирным (только для текущего года)

MonthlyBoldedDates: содержит набор дат, которые будут отмечены жирным для каждого месяца

Свойства для определения дат в календаре:

MinDate: определяет минимальную дату для выбора в календаре

MaxDate: задает наибольшую дату для выбора в календаре

FirstDayOfWeek: определяет день недели, с которого должна начинаться неделя в календаре

SelectionRange: определяет диапазон выделенных дат

SelectionEnd: задает конечную дату выделения

SelectionStart: определяет начальную дату выделения

ShowToday: при значении true отображает внизу календаря текущую дату

ShowTodayCircle: при значении true текущая дата будет обведена кружочком

TodayDate: определяет текущую дату. По умолчанию используется системная дата на компьютере, но с помощью данного свойства мы можем ее изменить.

Элемент PictureBox

PictureBox предназначен для показа изображений. Он позволяет отобразить файлы в формате bmp, jpg, gif, а также метафайлы ищображений и иконки. Для установки изображения в PictureBox можно использовать ряд свойств:

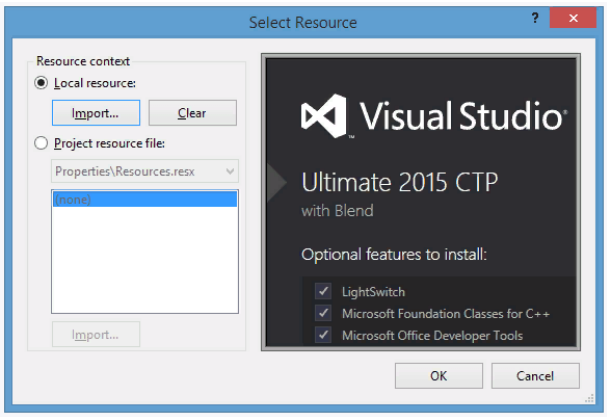
Image: устанавливает объект типа Image

ImageLocation: устанавливает путь к изображению на диске или в интернете

InitialImage: некоторое начальное изображение, которое будет отображаться во время загрузки главного изображения, которое хранится в свойстве Image

ErrorImage: изображение, которое отображается, если основное изображение не удалось загрузить в PictureBox

Чтобы установить изображение в Visual Studio, надо в панели Свойств PictureBox выбрать свойство Image. В этом случае нам откроется окно импорта изображения в проект, где мы собственно и сможем выбрать нужное изображение на компьютере и установить его для PictureBox:



И затем мы сможем увидеть данное изображение в PictureBox:

Либо можно загрузить изображение в коде:

```
pictureBox1.Image = Image.FromFile("C:\Users\Eugene\Pictures\12.jpg");
```

Размер изображения

Для установки изображения в PictureBox используется свойство SizeMode, которое принимает следующие значения:

Normal: изображение позиционируется в левом верхнем углу PictureBox, и размер изображения не изменяется. Если PictureBox больше размеров изображения, то по справа и снизу появляются пустоты, если меньше - то изображение обрезается

StretchImage: изображение растягивается или сжимается таким обраом, чобы вместиться по всей ширине и высоте элемента PictureBox

AutoSize: элемент PictureBox автоматически растягивается, подстраиваясь под размеры изображения

CenterImage: если PictureBox меньше изображения, то изображение обрезается по краям и выводится только его центральная часть. Если же PictureBox больше изображения, то оно позиционируется по центру.

Zoom: изоражение подстраивается под размеры PictureBox, сохраняя при этом пропорции

Задания

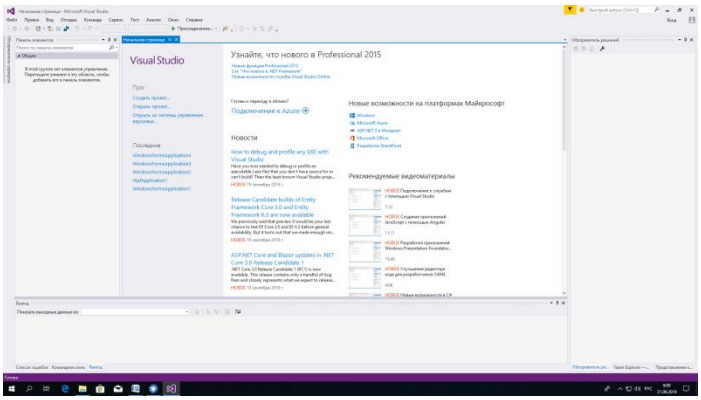
1. Изучить теоретические сведения.
2. Создать сообщение в консольном приложении.
3. Создать приложение WindowsForms по варианту, в которое добавить кнопку ВЫХОД.

Порядок выполнения работы

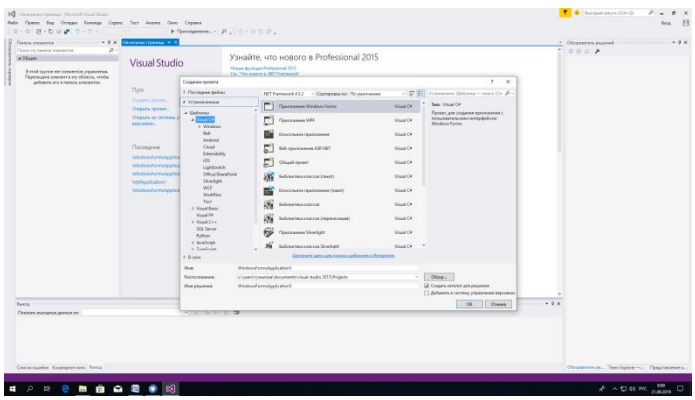
Задание 1 Изучение окон и служб среды разработки VisualStudio

После вызова **VisualStudio** на экране появляется главное окно. Основными элементами являются:

- строка заголовка, содержащая название программы и имя открытого файла, а также кнопки свертывания, восстановления и закрытия;
- строка меню, которая представляет доступ ко всем функциям и командам;
- панель элементов, состоящая из нескольких закладок, на которых располагаются визуальные компоненты, используемые при создании программ;
- окно свойств, в котором производится настройка основных свойств визуальных компонентов;
- список ошибок, содержащий ошибки, найденные транслятором на всех этапах построения проекта:
- обозреватель решений обеспечивает выбор решения, проекта, исходного файла, ссылок на внешние сборки и прочих ресурсов, которые и образуют проект.



В меню Файл последовательно выберите пункты Создать и Проект.



В области Категории шаблонов откройте Visual C#, а затем щелкните Windows.

В области Шаблоны щелкните Консольное приложение.

Введите имя проекта в поле Имя. Нажмите кнопку ОК.

В Обозревателе решений появится новый проект.

Если файл Program.cs не открыт в редакторе кода, щелкните правой кнопкой мыши Program.cs в обозревателе решений, а затем нажмите кнопку Просмотреть код.

Любая программа на языке C# - это набор классов, которые взаимодействуют друг с другом.

В одном из классов программы должна находиться, так называя «точка входа» - статический метод Main. Наличие или отсутствие этого метода определяет тип получаемого результата компиляции – сборки.

Если метод присутствует – получаем исполняемую программу EXE, в противном случае – библиотеку DLL. Классы могут быть вложены друг в друга. Но точка входа должна быть только в одном.

Класс определяется с помощью ключевого слова class, после которого идет имя класса. Тело класса заключается в фигурные скобки.

Язык программирования C# чувствителен к регистру символов, поэтому метод Main должен начинаться с большой буквы. Кроме того, этот метод должен быть определен как static, что позволит вызвать метод без создания объекта класса. Заголовок можно безболезненно упростить, удалив аргументы, которые, как правило, не задаются. Они имеют смысл, когда проект вызывается из командной строки, позволяя с помощью параметров задать нужную стратегию выполнения проекта.

Вставьте следующий код в метод Main между фигурными скобками:

```
Console.WriteLine("Всем привет! Я, <Фамилия, Имя>, здесь и рад встрече!");
Console.ReadKey();
```

Нажмите клавишу F5 для запуска проекта. Появляется окно командной строки, содержащее строку.

2 Создание приложения WindowsForms

Ядром Windows-программ, написанных на C#, является форма. Форма инкапсулирует основные функции, необходимые для: 1) создания окна, 2) его отображения на экране и 3) получения сообщений. Форма может представлять собой окно любого типа, включая основное окно приложения, дочернее или даже диалоговое окно.

Первоначально окно создается пустым. Затем в него добавляются меню и элементы управления, например экранные кнопки, списки и флажки. Таким образом, форму можно представить в виде контейнера для других Windows-объектов.

Событие — это нечто, инициируемое действиями пользователя при работе с оконным интерфейсом.

В Windows определены различные типы управляющих элементов: экранные кнопки, флажки, переключатели, окна списков.

Несмотря на различия между ними, способы их создания и обработки примерно одинаковы.

В меню Файл выберите команду Создать и щелкните Проект.

Выберите шаблон Приложение WindowsForms, в поле Имя введите MyProject и нажмите кнопку ОК.

Откроется конструктор WindowsForms с формой Windows. Это пользовательский интерфейс для создаваемого приложения.

Несмотря на то, что мы видим только форму, но стартовой точкой входа в графическое приложение является класс Program, расположенный в файле Program.cs:

В меню Вид щелкните Панель элементов, чтобы открыть список элементов управления.

Разверните список Стандартные элементы управления и перетащите элемент управления Label.

Также, из списка панели элементов Стандартные элементы управления перетащите в форму кнопку Button и поместите ее рядом с подписью.

Дважды щелкните новую кнопку, чтобы открыть редактор кода. Visual C# вставил метод с именем button1_Click, который выполняется при нажатии кнопки.

Изменитеметодследующимобразом.

```
private void button1_Click(object sender, EventArgs e)
{
label1.Text = "Hello, World!";
}
```

Нажмите клавишу F5 чтобы скомпилировать и запустить приложение.

При нажатии на кнопку теперь будет выводиться текстовое сообщение.

Элемент управления TextBox позволяет осуществлять ввод текстовой информации. Введенное значение помещается в textBox.Text

Для преобразования переменных в другой тип воспользуйтесь методом Convert.

Для выдачи предупреждающих сообщений используйте метод MessageBox.Show(“Сообщение”)

Для выхода из проекта используйте метод Application.Exit().

Значения progressBar и trackBar хранятся в свойстве Value.

Чтобы выполнять действия с элементами формы, содержащими числовые значения, необходимо преобразовать их в числовой формат, т.к. на форме все отображается в текстовом виде. Для этого необходимо задать команду преобразования Convert.ToInt32(<текстовое значение>).

Например,

```
Convert.ToInt32(label1.Text).
```

Чтобы преобразовать значения вычислений в текстовый формат, используем Convert.ToSnring(<значение>).

Например,

```
textBox1.Text = Convert.ToString(2 + 6 - 1);
```

или

```
textBox1.Text = (2 + 6 - 1).ToString();
```

Содержание отчета

1 Название работы
2 Цель работы
3 Технические средства обучения
4 Задания (условия задач)
5 Порядок выполнения работы
6 Ответы на контрольные вопросы
7 Вывод

Варианты заданий

Задание 3

1 Разработка проекта, содержащего форму, на которую помещены две кнопки и три элемента управления label. При нажатии на первую кнопку в элементе управления label1 выдается ваша фамилия, в label2 – имя, в label3 – отчество, при нажатии на вторую – надписи меняются местами.

2 Разработка проекта, содержащего форму, на которую помещена кнопка и элементы управления label и textBox. При нажатии на кнопку содержимое элемента управления textBox1 выдается в элементе управления label1.

3 Разработка проекта, содержащего форму, на которую помещена кнопка и элемент управления label, содержащий числовое значение. При каждом нажатии на кнопку к содержимому элемента управления label1 добавляется еще единица.

4 Разработка проекта, содержащего форму, на которую помещена кнопка управления, label и 2 элемента textBox. При нажатии на кнопку содержимое элементов управления textBox1 и textBox2 выдается в элементе управления label1.

5 Разработка проекта, содержащего форму, на которую помещены элементы управления textBox и label с текстом «Было набрано ». При каждом изменении значения textBox его содержимое добавляется в label после слов «, а затем ».

6 Разработка проекта, содержащего форму, на которую помещены кнопка, элементы управления textBox и label. При нажатии на кнопку содержимое textBox выдается в label.

7 Разработка проекта, содержащего форму, на которую помещены кнопка, два элемента управления textBox, содержащих числа, и label. При нажатии на кнопку содержимое textBox складывается и выдается после содержимого label в виде сообщения в MessageBox.

8 Разработка проекта, содержащего форму, на которую помещена кнопка и элемент управления label с произвольным текстом. При нажатии на кнопку содержимое элемента управления label изменяется на название вашей группы.

9 Разработка проекта, содержащего форму, на которую помещена кнопка и элемент управления label, в котором содержится слово «сегодня» и dateTimePicker. При нажатии на кнопку содержимое элемента управления dateTimePicker добавляется в label.

10 Разработка проекта, содержащего форму, на которую помещена кнопка и элемент управления textBox. При нажатии на кнопку содержимое элемента управления textBox выдается в виде сообщения в MessageBox.

11 Разработка проекта, содержащего форму, на которую помещена кнопка и два элемента управления label. При нажатии на кнопку в MessageBox выдается содержимое элементов управления label1 и label2.

12 Разработка проекта, содержащего форму, на которую помещена кнопка и элементы управления textBox и dateTimePicker. При нажатии на кнопку содержимое элемента управления dateTimePicker выдается в textBox.

13 Разработка проекта, содержащего форму, на которую помещена кнопка и элементы управления textBox и label, содержащие числовые значение. При каждом нажатии на кнопку к содержимому элемента управления label1 добавляется содержимое элемента управления textBox.

14 Разработка проекта, содержащего форму, на которую помещена кнопка и элемент управления label, содержащий числовое значение. При каждом нажатии на кнопку к содержимому элемента управления label1 добавляется еще единица.

15 Разработка проекта, содержащего форму, на которую помещена кнопка, элементы управления label и monthCalendar. При нажатии на кнопку в элементе управления label1 выдается выбранная дата, а в label2 день недели.

16 Разработка проекта, содержащего форму, на которую помещена кнопка, элементы управления label, в которых помещен текст, и monthCalendar. При нажатии на кнопку в элементе управления label1 добавляется текущая дата, а в label2 день недели.

17 Разработка проекта, содержащего форму, на которую помещена кнопка, элемент управления pictureBox. При нажатии на кнопку в элементе управления pictureBox добавляется изображение из файла, ссылка на который начинается с @, а затем следует полный путь к файлу в кавычках, например @"C:\Users\...\круг.png"

18 Разработка проекта, содержащего форму, на которую помещена кнопка, элементы управления label и dateTimePicker1. При нажатии на кнопку в элементе управления label1 выдается выбранная дата, а в label2 день недели.

19 Разработка проекта, содержащего форму, на которую помещена кнопка, элементы управления label, в которых помещен текст, и dateTimePicker1. При нажатии на кнопку в элементе управления label1 добавляется текущая дата, а в label2 день недели.

20 Разработка проекта, содержащего форму, на которую помещена кнопка и элемент управления textBox. При нажатии на кнопку меняется цвет фона, цвет фона в textBox и цвет текста.

21 Разработка проекта, содержащего форму, на которую помещена кнопка. При нажатии на кнопку в качестве фона добавляется изображение из файла, ссылка на который начинается с @, а затем следует полный путь к файлу в кавычках, например @"C:\Users\...\круг.png"

22 Разработка проекта, содержащего форму, на которую помещена кнопки и элементы управления label. При нажатии на одну кнопку меняется цвет фона, а на другую – цвет текста в label.

23 Разработка проекта, содержащего форму, на которую помещена кнопка и элементы управления label, в котором содержится слово «сегодня» и dateTimePicker. При нажатии на кнопку по выбранной дате в dateTimePicker в label1 добавляется день недели, а в label2 текущее время.

Контрольные вопросы

1 Что такое Visual Studio?
2 Основные возможности Visual Studio.
3 Что такое Microsoft.NET Framework и для чего используется?
4 Для чего служит консольное приложение?
5 Что такое форма и для чего предназначена?
6 Как задать название, цвет и другие параметры формы?
7 Для чего используется элементы управления label, textBox, кнопка?
8 Для чего используется элементы управления DateTimePicker и MonthCalendar?
9 Чем отличаются DateTimePicker и MonthCalendar?

Используемая литература

- Шарп Джон Ш26 Microsoft Visual C#. Подробное руководство. 8-е изд. — СПб.: Питер, 2017.
- Васильев А.Н. Программирование на C# для начинающих. Основные сведения. – Москва: Эксмо, 2018.
- Васильев А.Н. Программирование на C# для начинающих. Особенности языка. – Москва: Эксмо, 2019.
- http://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx.
- https://metanit.com/sharp/windowsforms