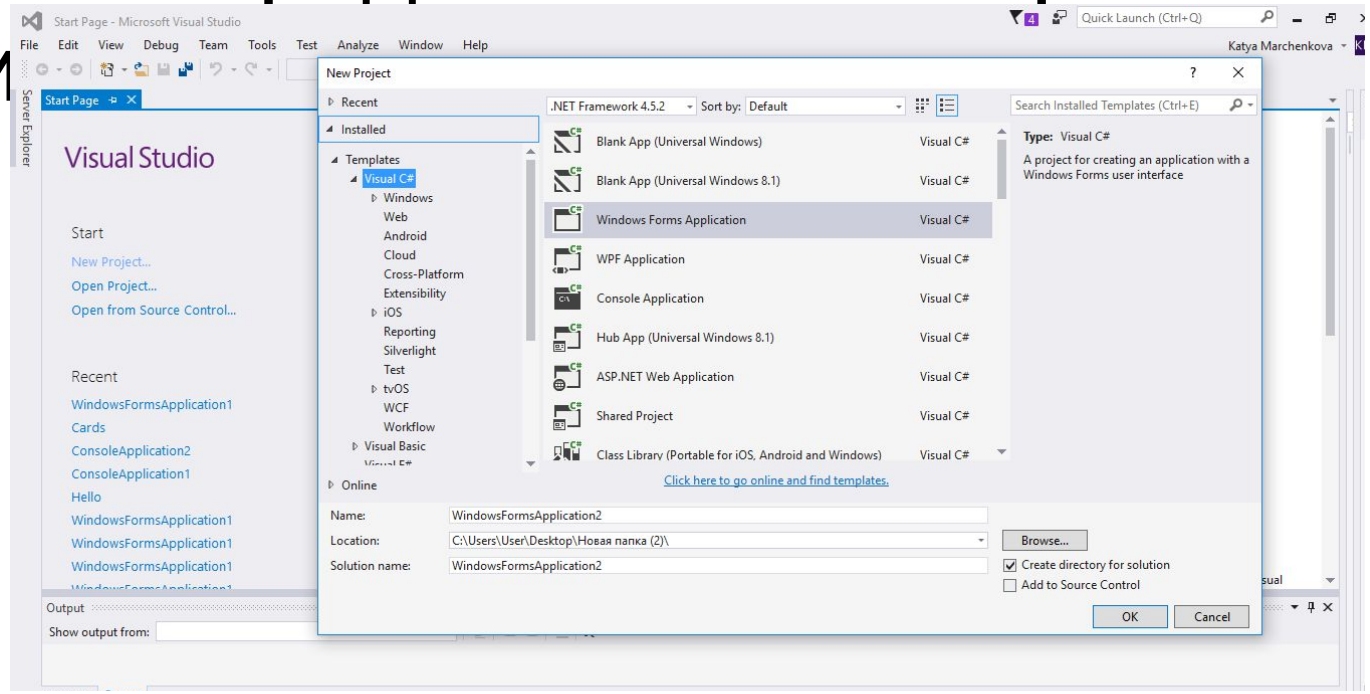
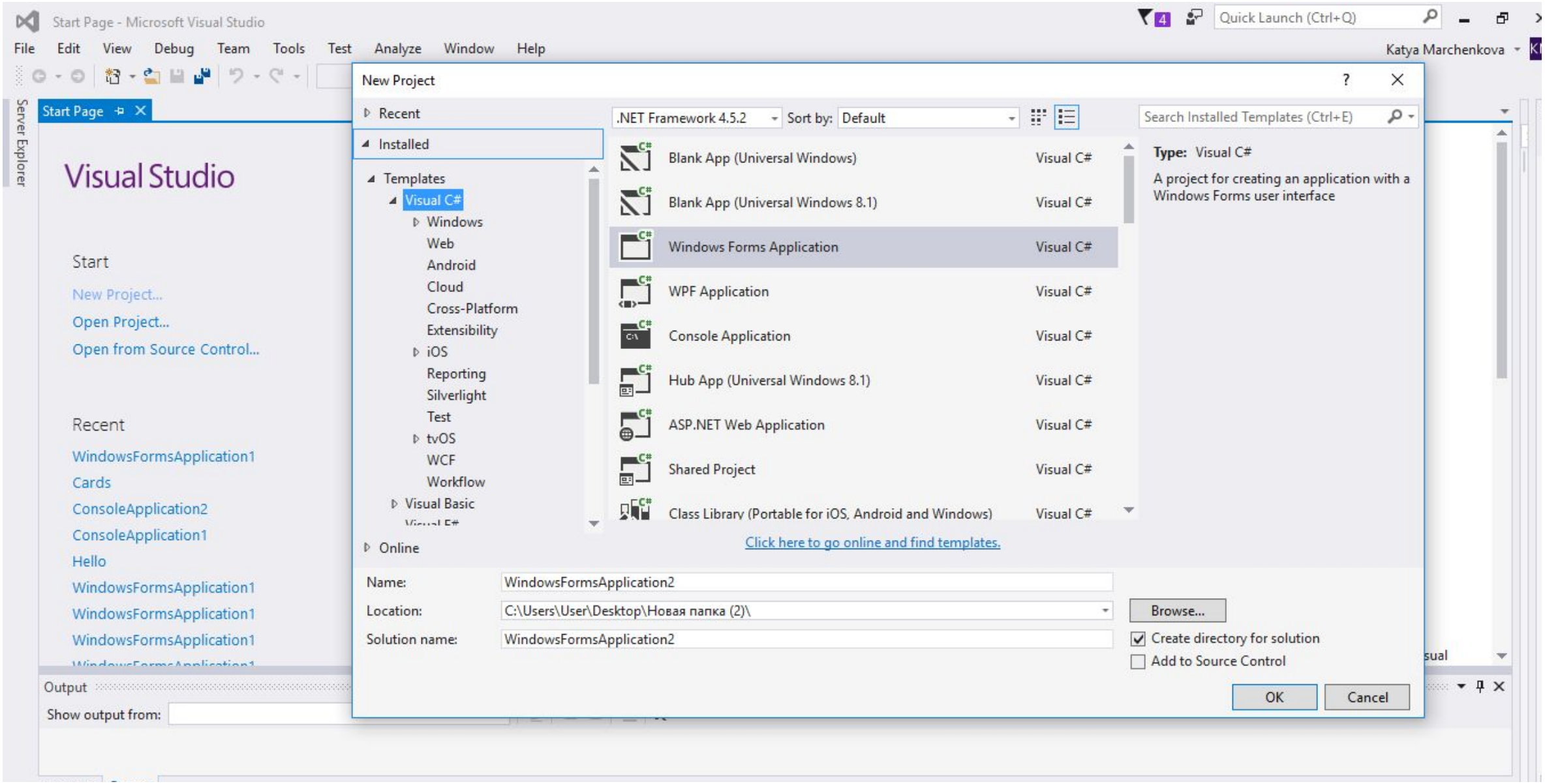
Windows Forms

Преподаватель Марченкова Е. А.

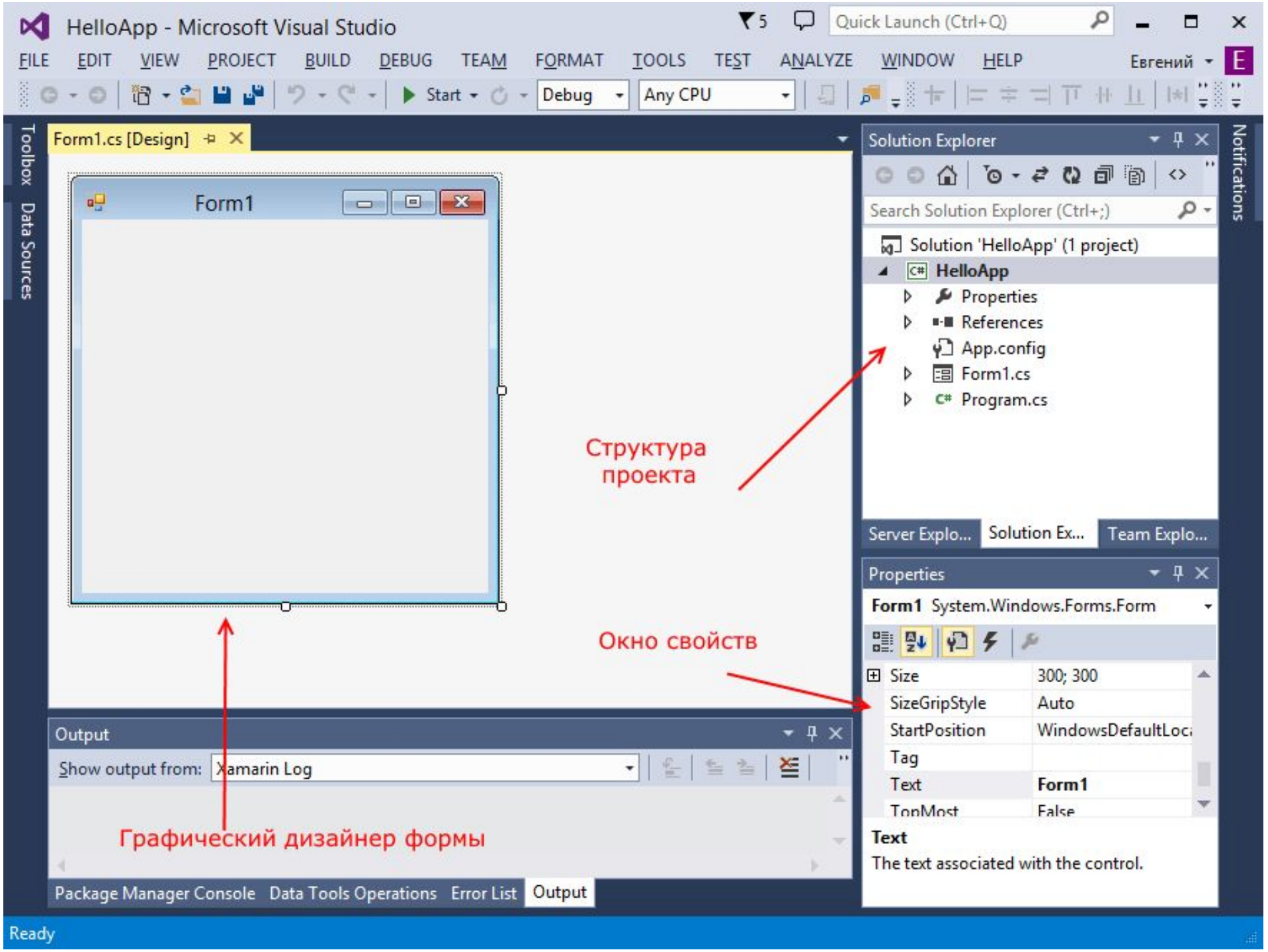
Создание графического приложения

После установки среды и всех ее компонентов, запустим Visual Studio и создадим проект графического приложения. Для этого в меню выберем пункт File (Файл) и в подменю выберем New - > Project (Создать - > Проект). После этого перед нами откроется диалоговое окно создани

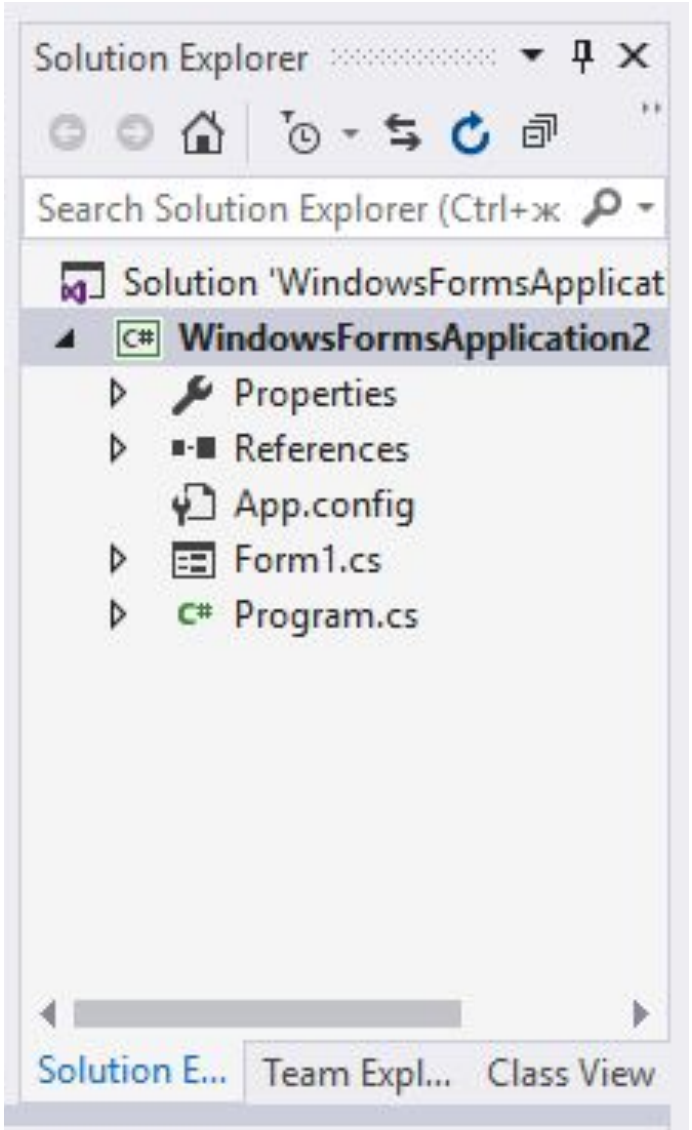



Start Page - Microsoft Visual Studio
File Edit View Debug Team Tools Test Analyze Window Help
Katya Marchenkova
Quick Launch (Ctrl+Q)
Server Explorer
Start Page
Visual Studio
Start
New Project...
Open Project...
Open from Source Control...
Recent
WindowsFormsApplication1
Cards
ConsoleApplication2
ConsoleApplication1
Hello
WindowsFormsApplication1
WindowsFormsApplication1
WindowsFormsApplication1
Output
Show output from:
New Project
Recent
Installed
Templates
Visual C#
Windows
Web
Android
Cloud
Cross-Platform
Extensibility
iOS
Reporting
Silverlight
Test
tvOS
WCF
Workflow
Visual Basic
Online
.NET Framework 4.5.2
Sort by: Default
Search Installed Templates (Ctrl+E)
Blank App (Universal Windows) Visual C#
Blank App (Universal Windows 8.1) Visual C#
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
Hub App (Universal Windows 8.1) Visual C#
ASP.NET Web Application Visual C#
Shared Project Visual C#
Class Library (Portable for iOS, Android and Windows) Visual C#
Type: Visual C#
A project for creating an application with a Windows Forms user interface
Click here to go online and find templates.
Name: WindowsFormsApplication2
Location: C:\Users\User\Desktop\Новая папка (2)\
Solution name: WindowsFormsApplication2
Browse...
Create directory for solution
Add to Source Control
OK
Cancel

После этого Visual Studio откроет наш проект с созданными по умолчанию файлами:

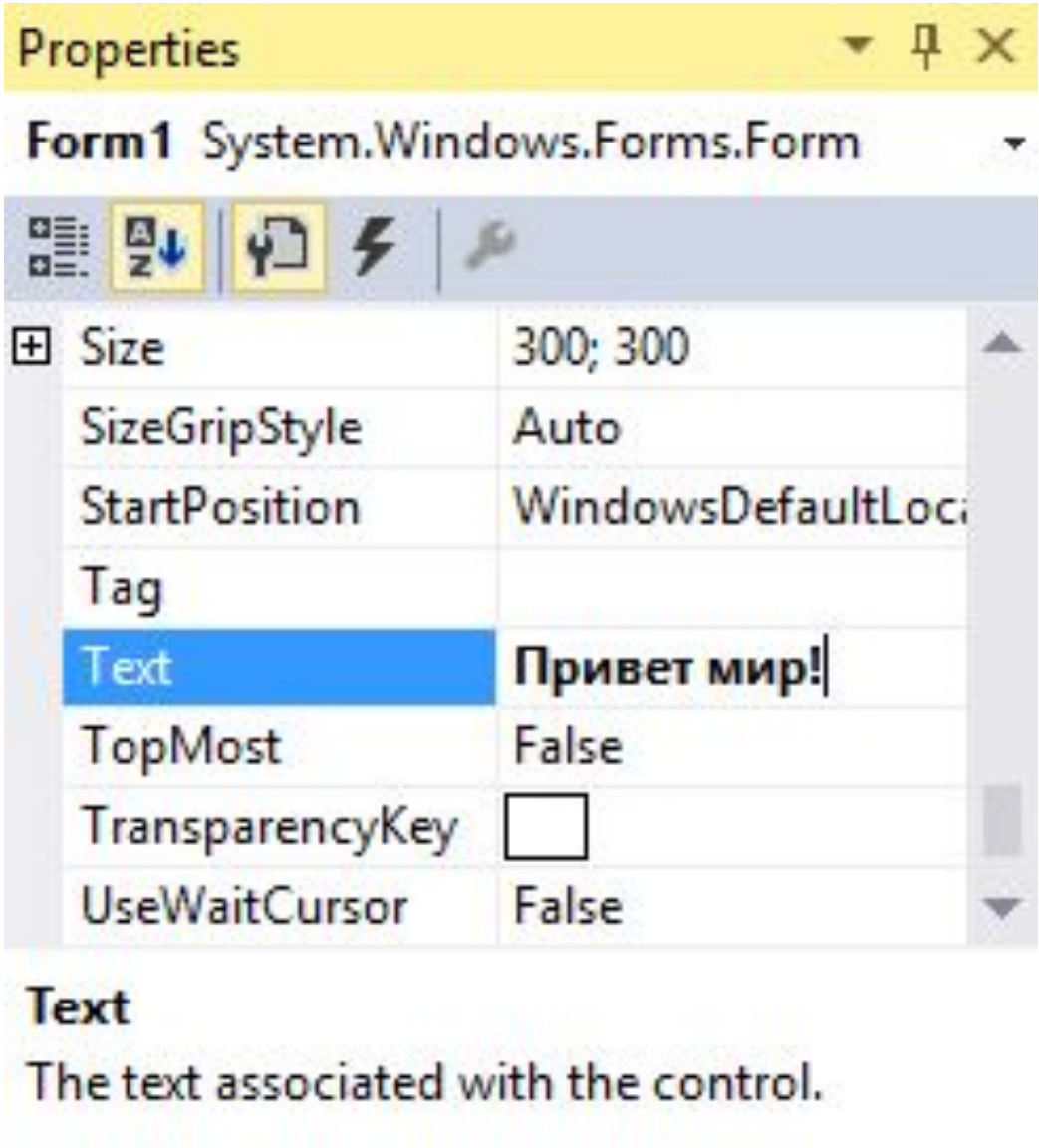


Большую часть пространства Visual Studio занимает графический дизайнер, который содержит форму будущего приложения. Пока она пуста и имеет только заголовок Form1. Справа находится окно файлов решения/проекта - Solution Explorer (Обозреватель решений). Там и находятся все связанные с нашим приложением файлы, в том числе файлы формы Form1.cs.

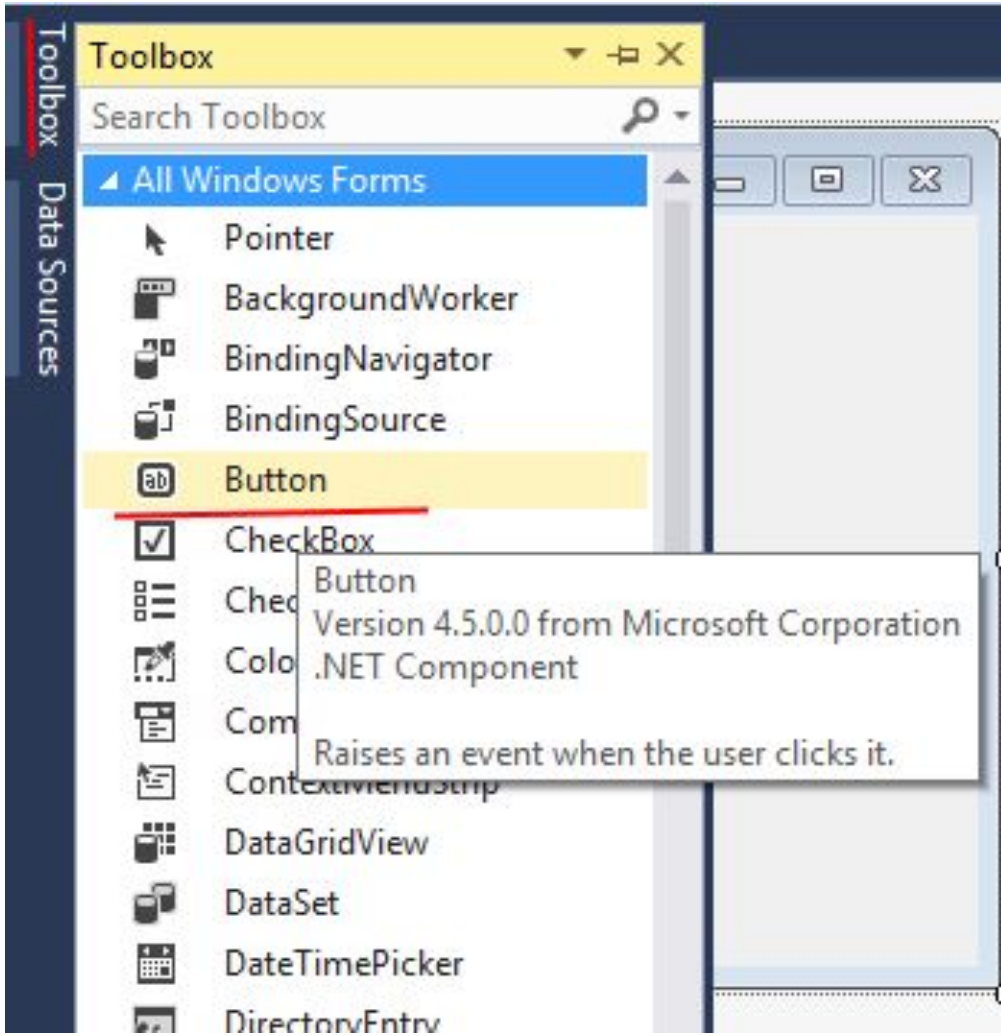


Внизу справа находится окно свойств - Properties. Так как у меня в данный момент выбрана форма как элемент управления, то в этом поле отображаются свойства, связанные с формой.

Теперь найдем в этом окне свойство формы Text и изменим его значение на любое другое:



Таким образом мы поменяли заголовок формы. Теперь перенесем на поле какой-нибудь элемент управления, например, кнопку. Для этого найдем в левой части Visual Studio вкладку Toolbox (Панель инструментов). Нажмем на эту вкладку, и у нас откроется панель с элементами, откуда мы можем с помощью мыши перенести на форму любой



- Добавим вывод сообщения по нажатию кнопки, изменив код следующим образом:

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

        private void button1_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Привет");
        }
    }
}
```

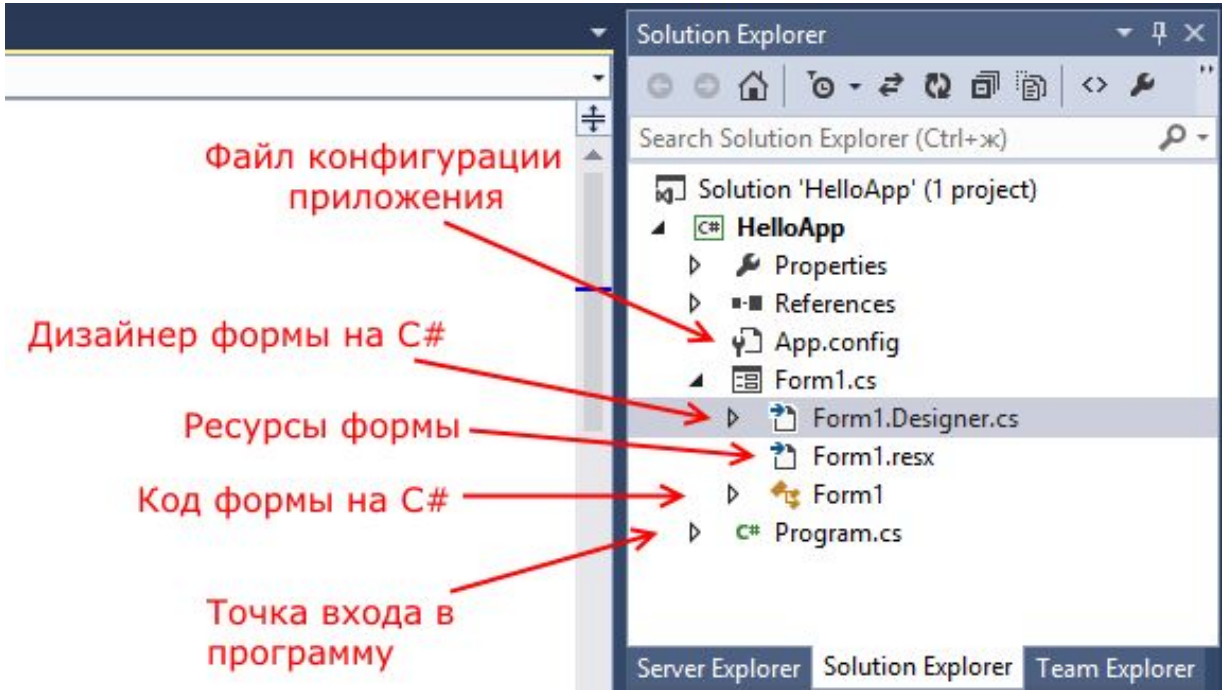
Запуск приложения

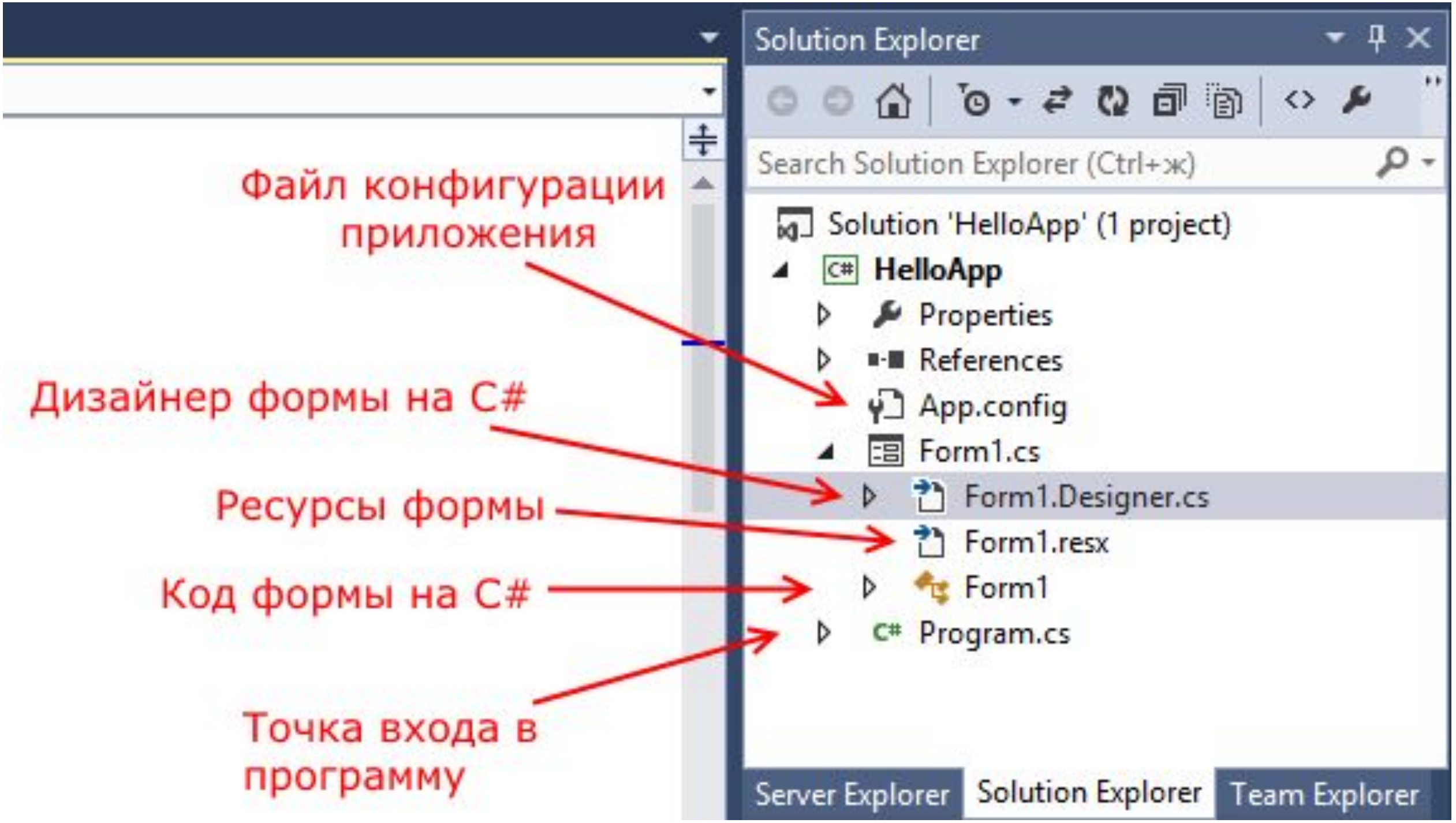
- Чтобы запустить приложение в режиме отладки, нажмем на клавишу F5 или на зеленую стрелочку на панели Visual Studio. После этого запустится наша форма с одинокой кнопкой. И если мы нажмем на кнопку на форме, то нам будет отображено сообщение с приветствием.

- После запуска приложения студия компилирует его в файл с расширением exe. Найти данный файл можно, зайдя в папку проекта и далее в каталог bin/Debug или bin/Release

Работа с формами

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




Файл конфигурации приложения
Дизайнер формы на C#
Ресурсы формы
Код формы на C#
Точка входа в программу
Solution Explorer
Search Solution Explorer (Ctrl+ж)
Solution 'HelloApp' (1 project)
HelloApp
Properties
References
App.config
Form1.cs
Form1.Designer.cs
Form1.resx
Form1
Program.cs
Server Explorer
Solution Explorer
Team Explorer

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

```
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:

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.

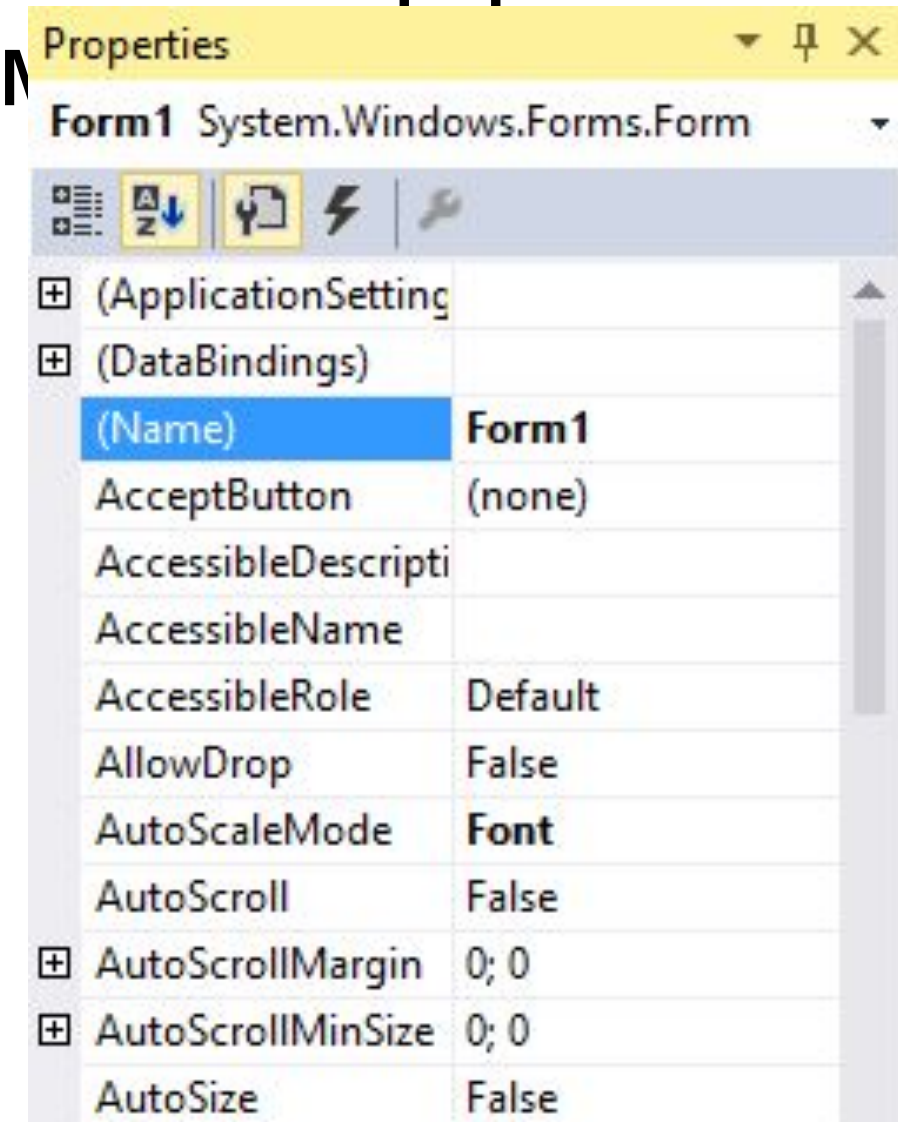
Form1.resx - хранит ресурсы формы. Как правило, ресурсы используются для создания однообразных форм сразу для нескольких языковых культур.

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

```
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.

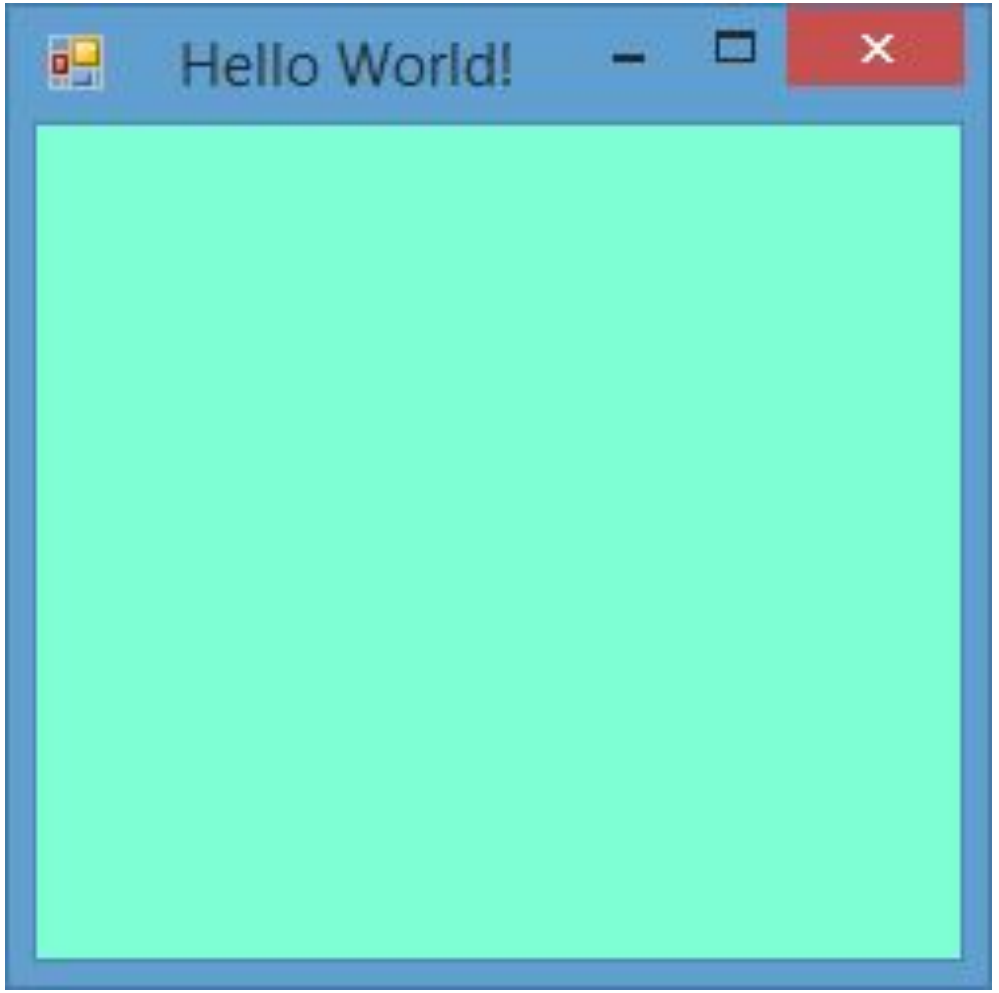
С помощью специального окна Properties (Свойства) справа Visual Studio предоставляет нам удобный интерфейс для управления свойствам



Программная настройка свойств

Внешний вид формы

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



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

Установить форму в центре экрана

```
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;
```

Пример:
Допустим, первая форма по нажатию на кнопку будет вызывать вторую форму.

- Во-первых, добавим на первую форму Form1 кнопку и двойным щелчком по кнопке перейдем в файл кода.
- Во-вторых, добавим в него код вызова второй формы. У нас вторая форма называется Form2, поэтому сначала мы создаем объект данного класса, а потом для его отображения на экране вызываем метод Show:

```
private void button1_Click(object sender, EventArgs e)
{
   Form2 newForm = new Form2();
    newForm.Show();
}
```

Теперь сделаем наоборот - чтобы вторая форма воздействовала на первую.

```
namespace HelloApp
{ public partial class Form2 : Form
  {     public Form2()
   {        InitializeComponent();    }
   public Form2(Form1 f)
    {        InitializeComponent();
      f.BackColor = Color.Yellow;    } }}
```

Фактически мы только добавили здесь новый конструктор public Form2(Form1 f), в котором мы получаем первую форму и устанавливаем ее фон в желтый цвет.

Теперь перейдем к коду первой формы, где мы вызывали вторую форму и изменим его на следующий:

```
private void button1_Click(object sender, EventArgs e)
{
    Form2 newForm = new Form2(this);
   newForm.Show();
}
```

Теперь после нажатия на кнопку у нас будет создана вторая форма, которая сразу изменит цвет первой формы.

Поскольку в данном случае ключевое слово this представляет ссылку на текущий объект - объект Form1, то при создании второй формы она будет получать ее (ссылку) и через нее управлять первой формой.

Мы можем также создавать объекты и текущей формы:

```
private void button1_Click(object sender, EventArgs e)
{Form1 newForm1 = new Form1();
   newForm1.Show();

   Form2 newForm2 = new Form2(newForm1);
   newForm2.Show();}
```

При работе с несколькими формами надо учитывать, что одна из них является главной - которая запускается первой в файле Program.cs. Если у нас одновременно открыта куча форм, то при закрытии главной закрывается все приложение и вместе с ним все остальные формы.