Объектно-ориентированное программирование. Обзор среды разработки Visual Studio .NET

ОГУ

Кафедра ВТиЗИ

Галимов Р.Р.

Исходные данные по курсу

Лекций -17

Лабораторные работы -16

Курсовой проект

Итоговый контроль –экзамен.

Среда программирования: Visual Studio C# 12

Язык программирования: C#

Контроль знаний

- Лабораторные работы -4 балла (10 лабораторных работ)=40
- Тесты – 5 баллов(2 раза)=10
- КП – от 0 до 20 баллов = максимум 20
- Посещение занятий -0.5 =16 баллов
- Экзамен -14 баллов
- Допуск к экзамену 70 баллов

Содержание

1. Язык C# и каркас .NET

2Обзор среды разработки Visual Studio .NET

3. Создание первого приложения для Windows

4. Задание на лабораторную работу №1.

5. Контрольные вопросы

Язык C# и каркас .NET

- В 2000 году компания Microsoft представила новый язык программирования C#.
- Своими корнями C# уходит к языкам программирования C, C++ иJava.
- C# обладает примерно такими же возможностями, как и Java; он подходит для самых ответственных задач разработки, включая построение современных крупномасштабных корпоративных приложений, мобильных и «облачных» приложений.

Объектно-ориентированное программирование

- C# относится к категории объектно-ориентированных языков.
- Программы, написанные на C#, могут использовать .NET Framework Class Library - гигантскую библиотеку готовых классов, ускоряющую разработку приложений

Объектно-ориентированное программирование

Ключевые возможности .NET Framework Class Library	
Базы данных	Отладка
Построение веб-приложений	Многопоточное программирование
Графика	Операции с файлами
Ввод-вывод	Безопасность
Сетевые взаимодействия	Веб-взаимодействия
Управление разрешениями	Графический интерфейс
Мобильные приложения	Структуры данных
Обработка строк	

Событийное программирование

- Программирование на C# управляется событиями.
- Разработчик пишет программу, которая реагирует на события, инициированные пользователем: щелчки кнопкой мыши, нажатия клавиш, срабатывания таймера и (новая возможность Visual C# 2012) прикосновения к сенсорному экрану и жесты, широко используемые на смартфонах и планшетах.

Визуальное программирование

- Microsoft Visual C# является *визуальным языком программирования – помимо* написания команд для построения частей приложения, разработчик может использовать графический интерфейс (GUI, Graphic User Interface) Visual Studio для удобного перетаскивания заранее определенных объектов (таких, как кнопки и текстовые поля) *в нужное место экрана, назначения их текста и изменения размеров.*
- Visual Studio сгенерирует большую часть GUI-кода за вас.

СРЕДА ВЫПОЛНЕНИЯ Common Language Runtime

- Библиотека классов — это статическая составляющая каркаса.
- В .NET Framework есть и динамическая составляющая — система, определяющая среду выполнения,— CLR (Common Language Runtime).
- Роль этой среды весьма велика — в ее функции входит управление памятью, потоками, безопасностью, компиляция из промежуточного байт-кода в машинный код и многое другое.
- Важный элемент CLR — это мощный механизм ≪сборки мусора≫ (garbage collector), управляющий работой с памятью.

СРЕДА ВЫПОЛНЕНИЯ Common Language Runtime

- Среда CLR (Common Language Runtime), еще один ключевой компонент .NET Framework, исполняет программы .NET и предоставляет многие полезные средства для упрощения разработки и отладки.
- CLR представляет собой *виртуальную машину (VM, Virtual Machine) - программу, которая управляет выполнением* других программ и изолирует их от базовой операционной системы и оборудования.
- Исходный код программ, выполняемых под управлением CLR, называется *управляемым кодом.*
- *CLR предоставляет управляемому коду различные виды* сервиса - интеграцию программных компонентов, написанных на разных языках .NET, обработку ошибок между такими компонентами, повышенную безопасность, автоматическое управление памятью и т. д.

Платформенная независимость

- Если версия .NET Framework существует и установлена на платформе, эта платформа может выполнить любую программу .NET.
- Способность программы выполняться без изменений на разных платформах называется *платформенной независимостью.*
- Единожды написанный код может использоваться на другом типе компьютеров без изменений, а это экономит время и деньги. Вдобавок платформенная независимость расширяет круг пользователей.
- Прежде компаниям приходилось решать, оправдаются ли затраты по адаптации их программ для других платформ (этот процесс называется *портированием).*
- *С появлением .NET проблемы портирования отпали (по* крайней мере после того, как версии .NET стали доступными на разных платформах).

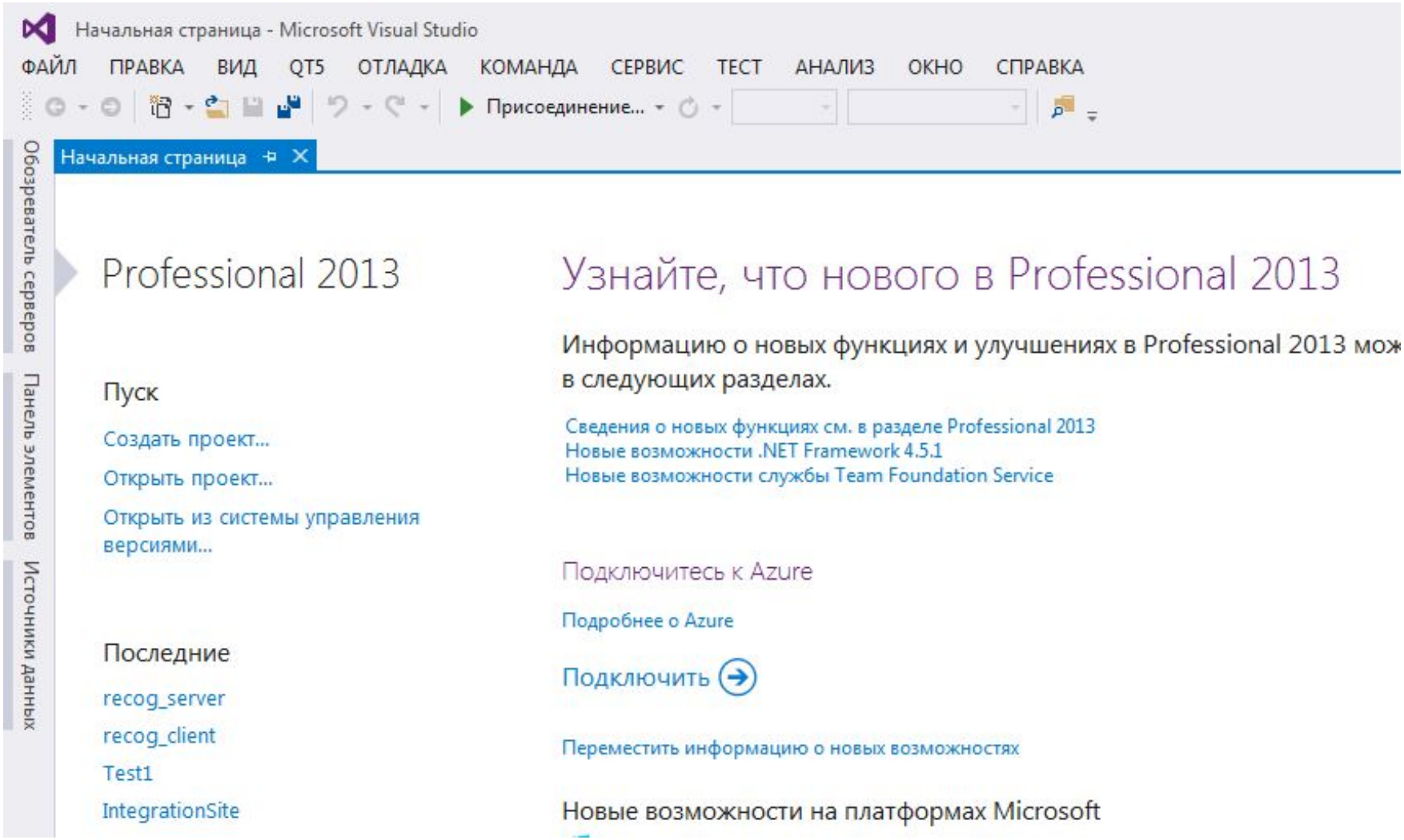
Строгая типизация данных

- Все переменные имеют четко определенный конкретный тип данных.
- Обычно не допускаются никакие действия, в результате которых получаются неопределенные типы данных.
- Общая система типов (Common Type System -CTS) описывает предопределенные типы данных.

Сборка мусора

- Сборщик мусора- это инструмент для восстановления памяти, которую запрашивало приложение.
- При недостатки памяти для приложения исполняющая среда вызывает сборщик мусора для освобождения объектов, на которых нет ссылки в коде.
- Сборщик мусора вызывается исполняющей средой по своему усмотрению

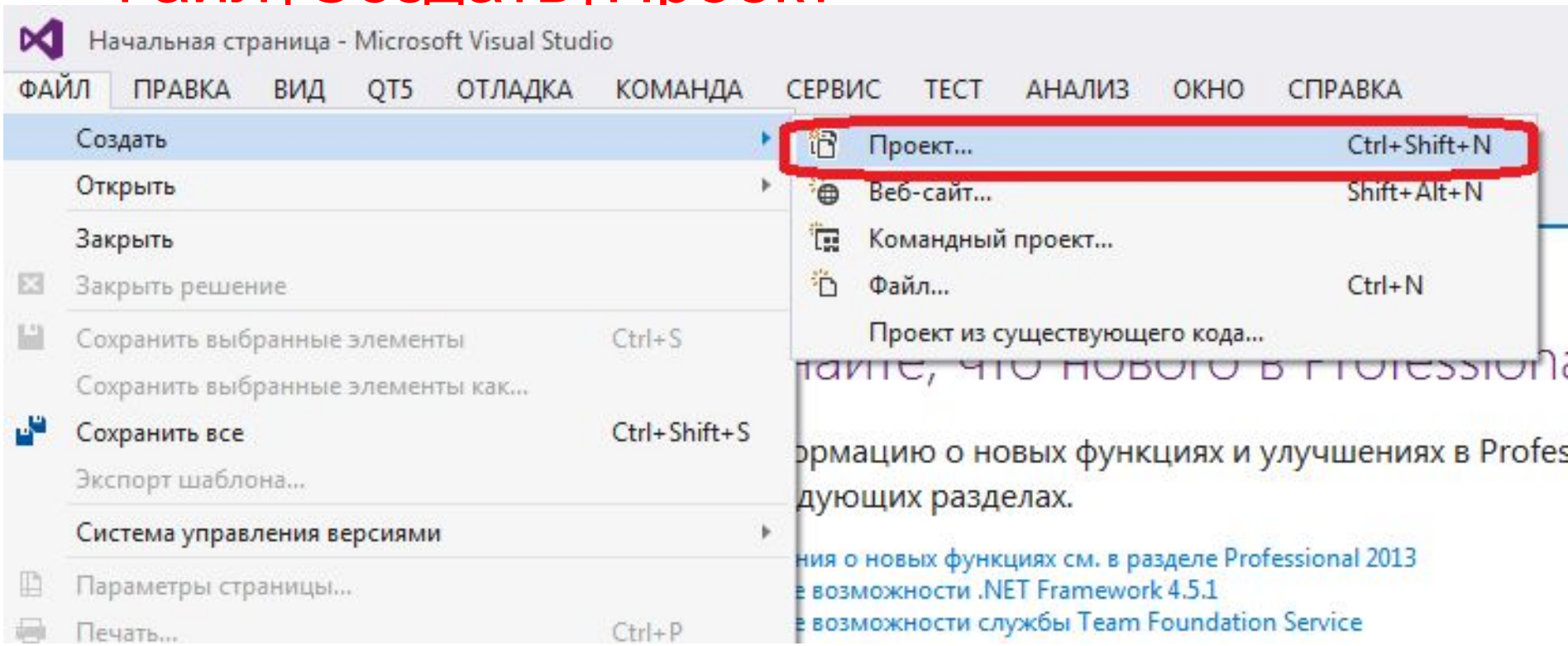
Visual Studio 13 C#



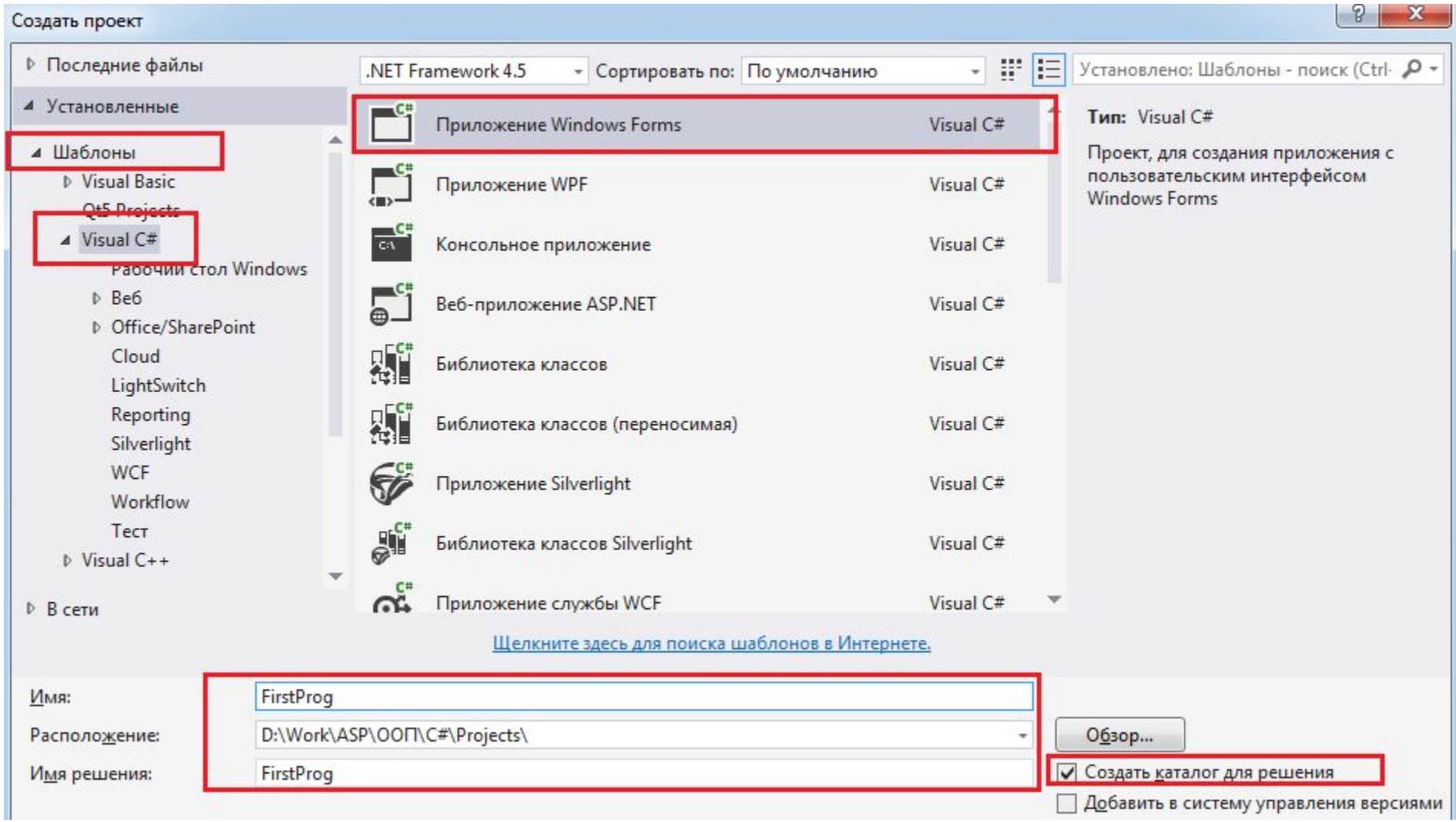
Visual Studio 13 C#

Создать новый проект:

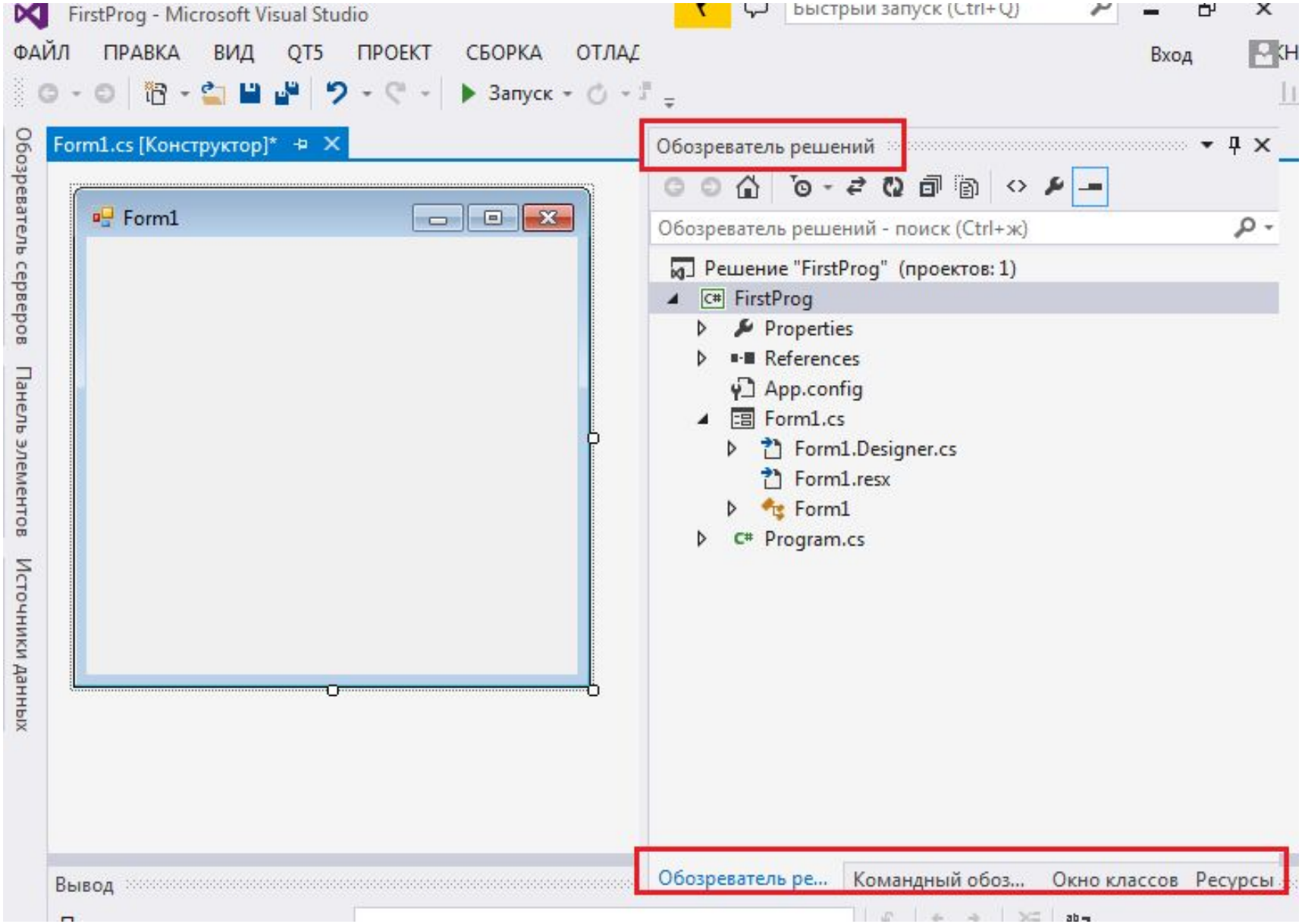
Файл | Создать | Проект



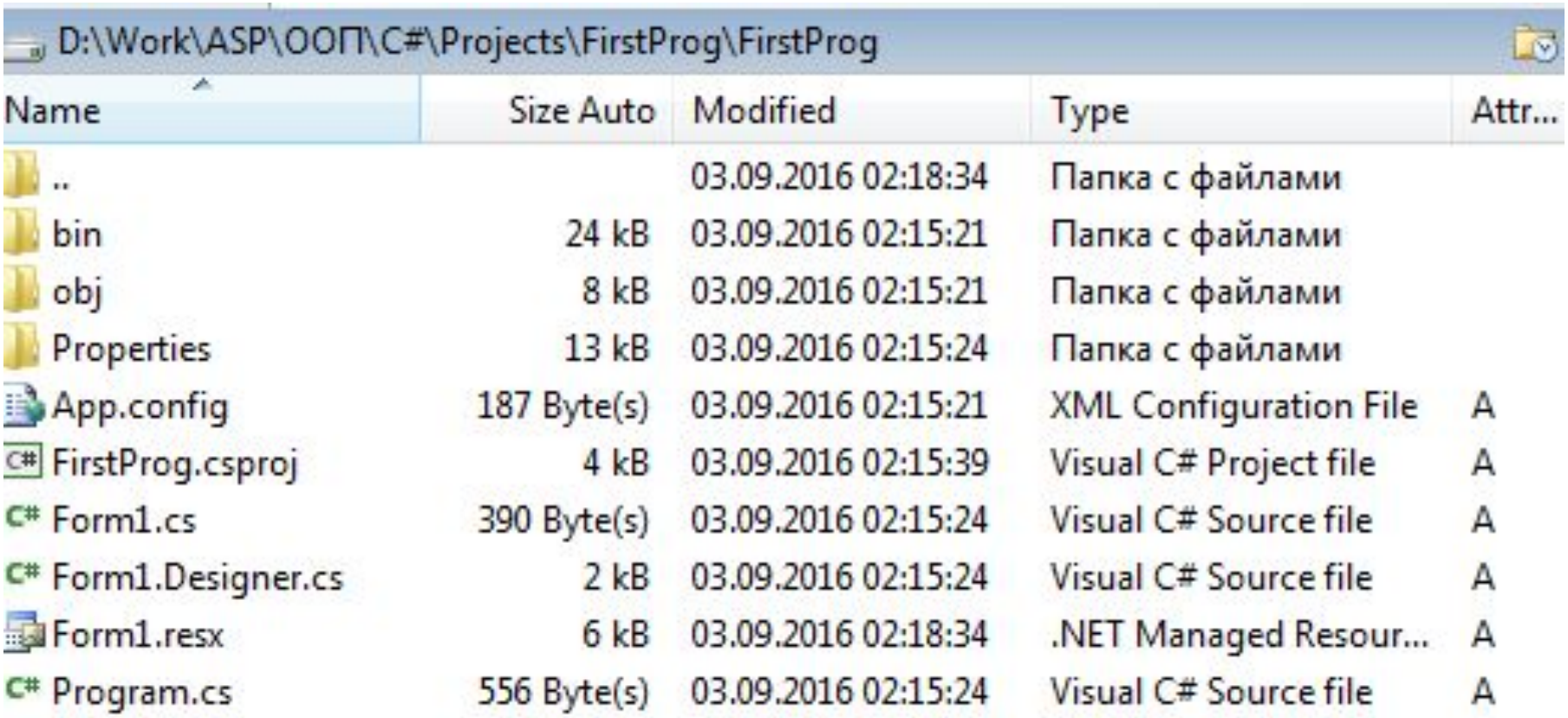
Visual Studio 13 C#



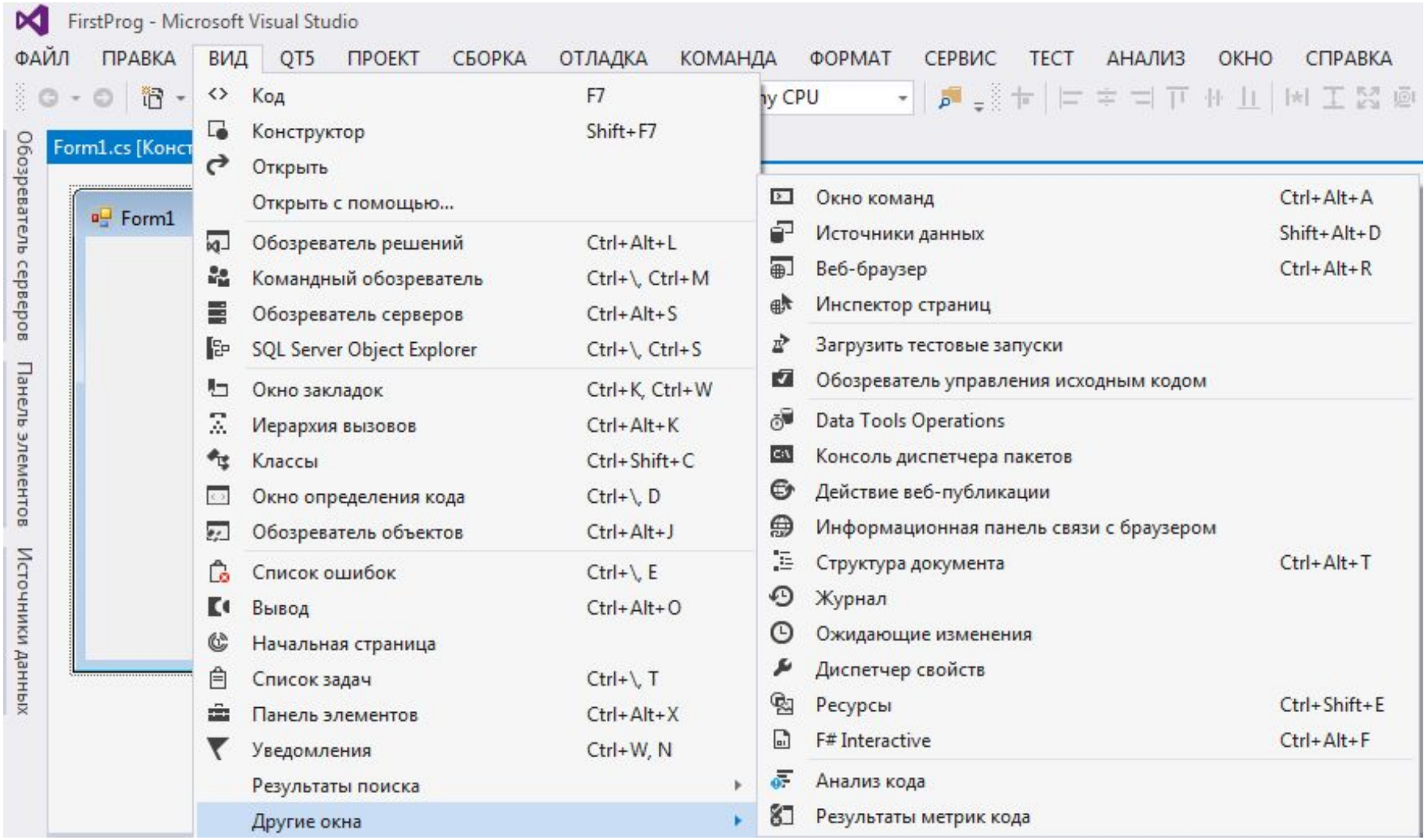
Visual Studio 13 C#



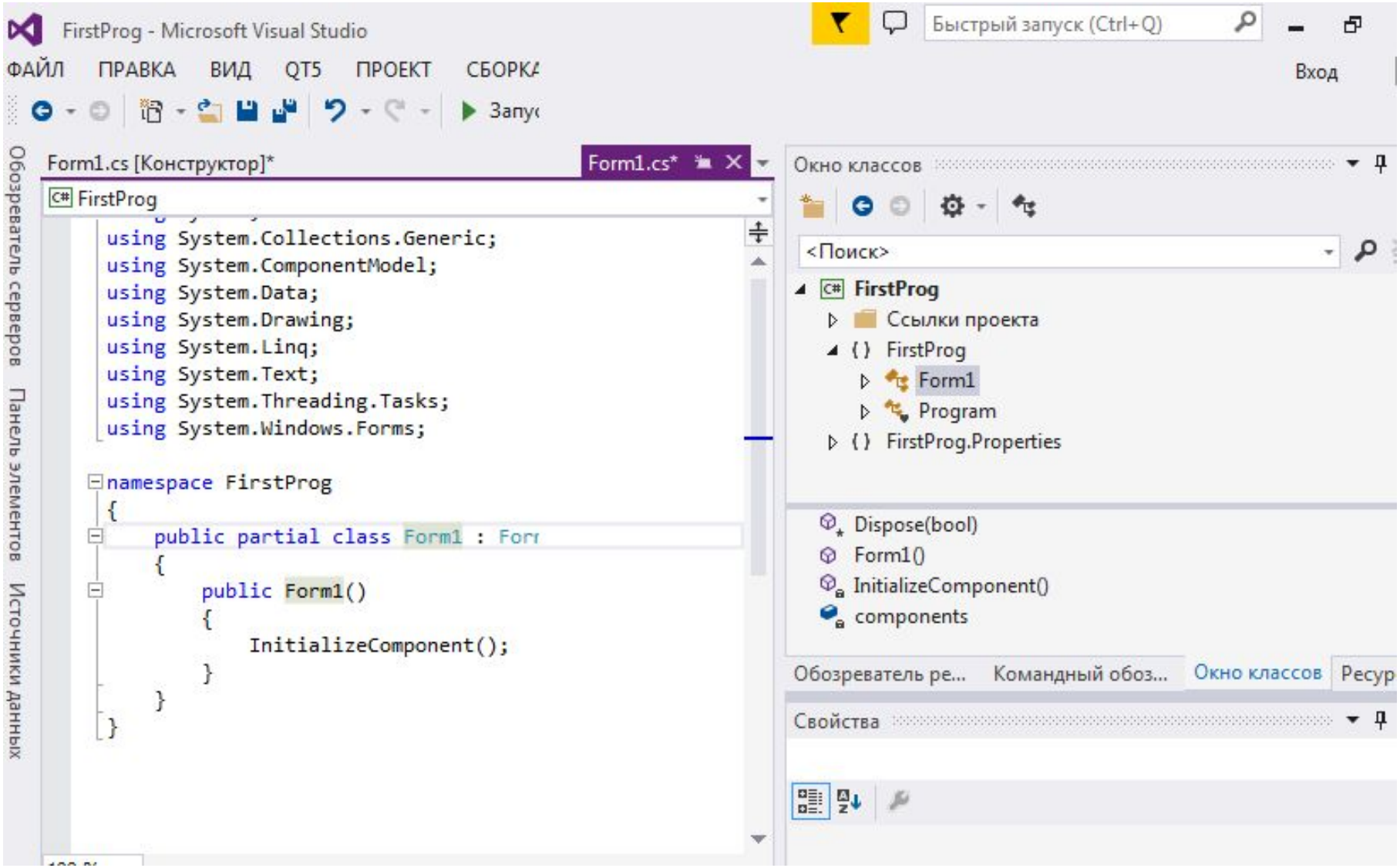
Visual Studio 13 C#



Visual Studio 13 C#



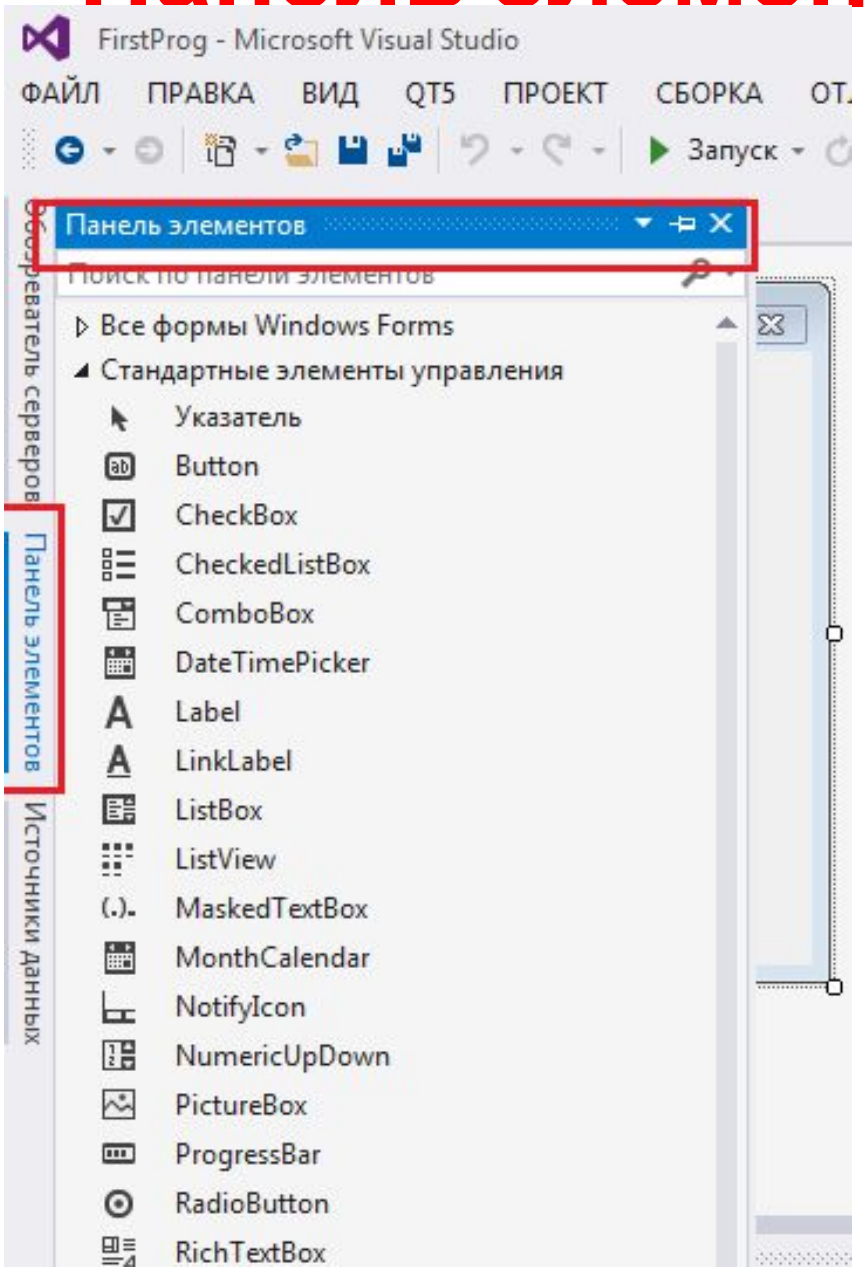
Visual Studio 13 C#



Формы и элементы управления

- Прямоугольник с заголовком Form *(форма) представляет главное окно создаваемого* приложения Windows Forms.
- Приложения Visual C# могут содержать несколько форм (окон);
- Visual Studio содержит много готовых элементов управления и других компонентов, которые могут использоваться для создания и настройки поведения приложений.

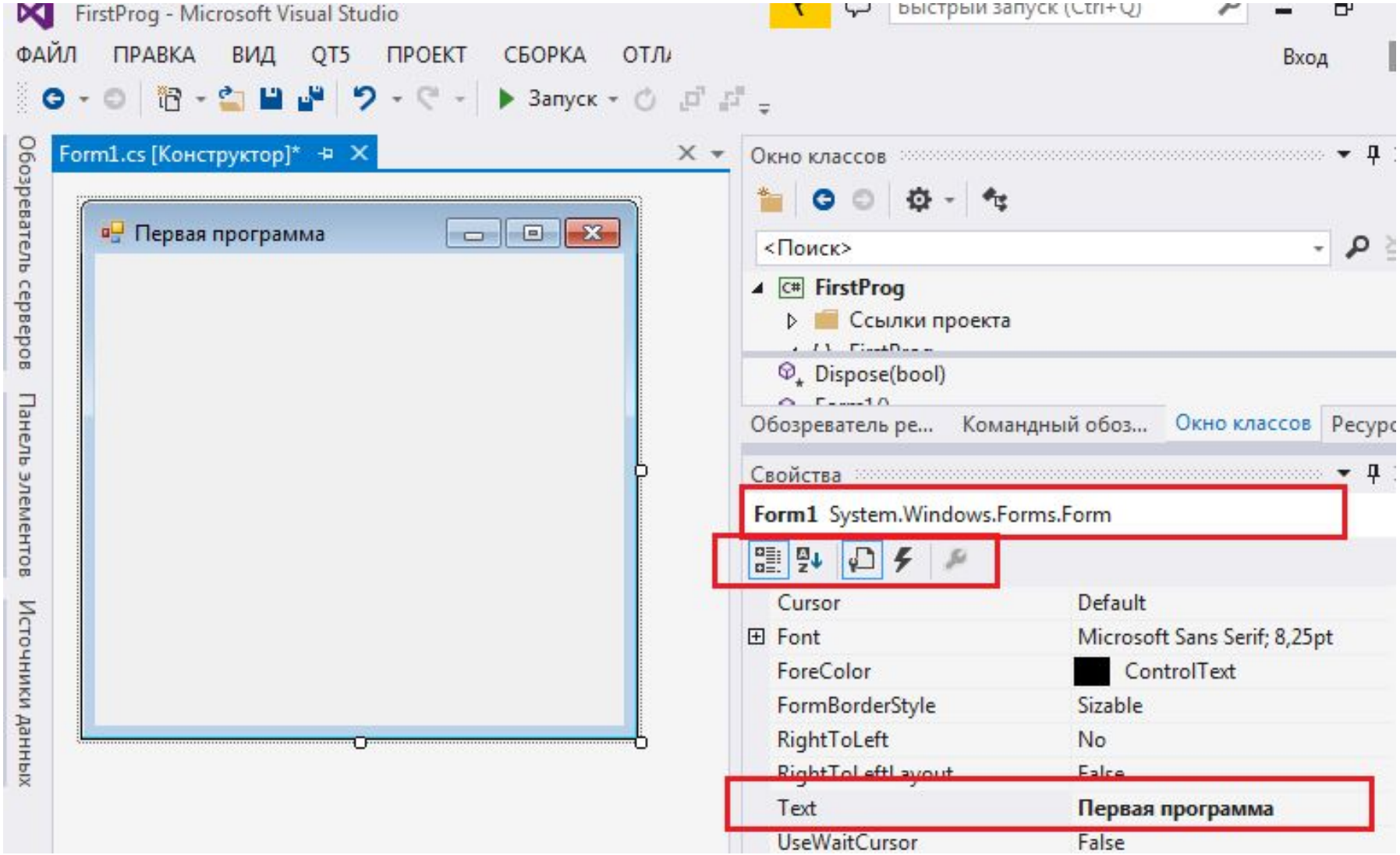
Панель элементов



Панель элементов

- Чтобы вызвать окно панели элементов (Toolbox), выполните команду VIEW | Toolbox.
- В нем содержатся элементы, используемые для модификации форм.
- В методологии визуального программирования разработчик <<перетаскивает» элементы управления на форму, а IDE генерирует код создания этих элементов.
- Так код создается проще и быстрее, чем при самостоятельном написании.
- Чтобы вести машину, не обязательно знать, как устроен ее двигатель; точно так же не обязательно уметь строить элементы, чтобы использовать их.
- Повторное использование готовых элементов экономит время и деньги при разработке.

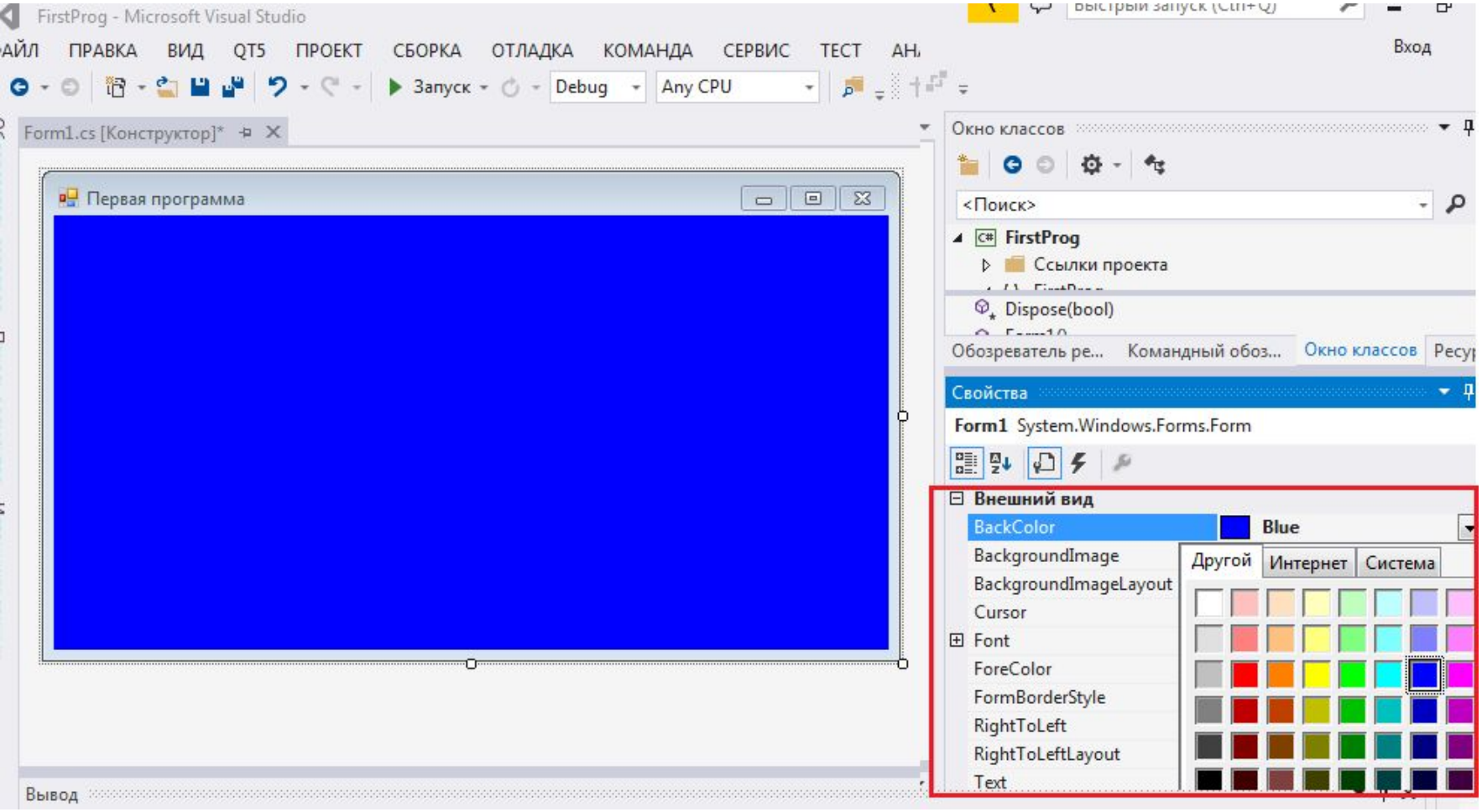
Окно свойств



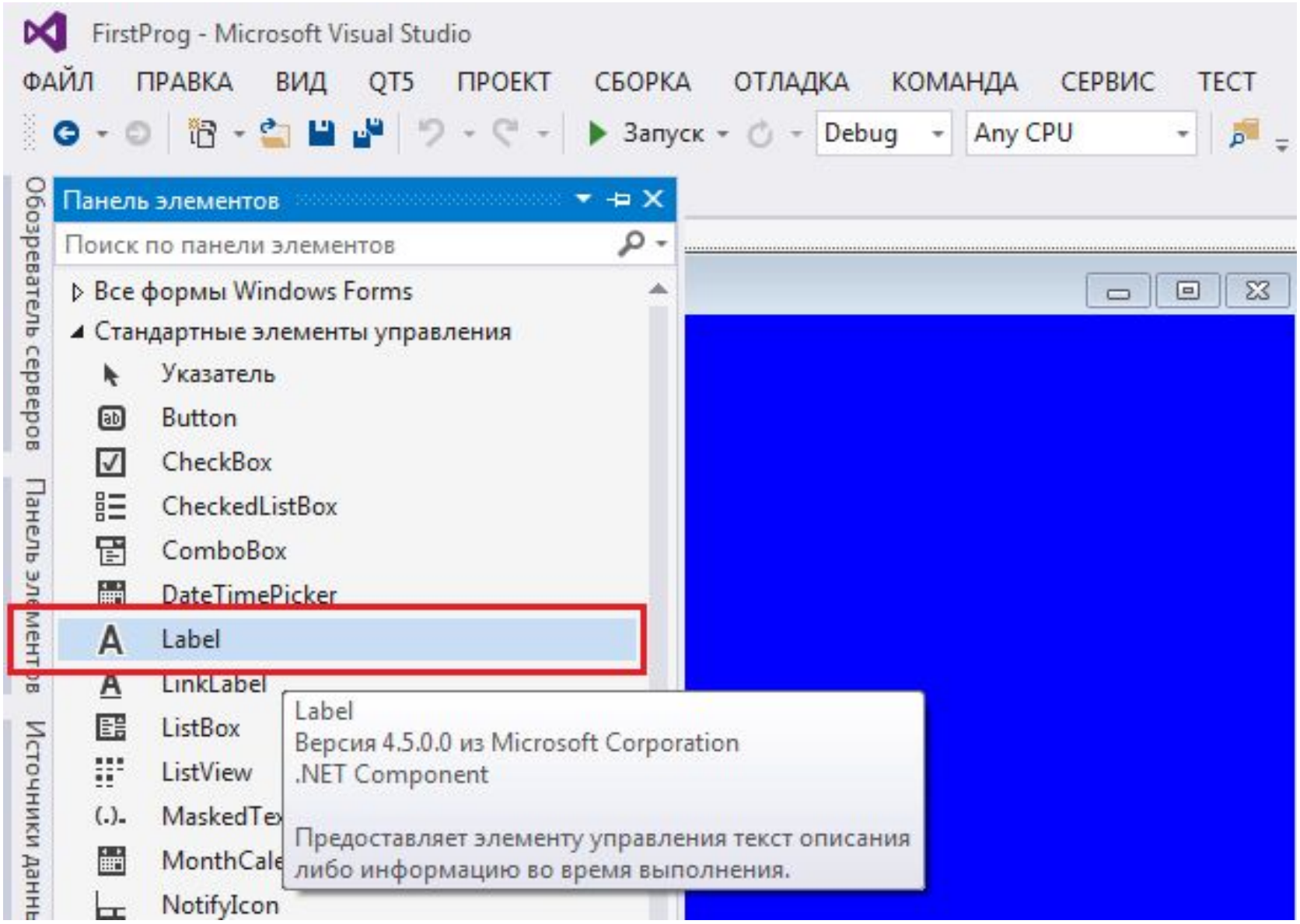
Окно свойств

- Если окно свойств (Properties) не отображается под окном Solution Explorer, выполните команду VIEW | Properties Window.
- В окне свойств выводятся свойства текущей выделенной формы, элемента управления или файла.
- Свойства описывают характеристики формы или элемента управления: размер, цвет, позицию и т. д.
- Разные формы и элементы управления обладают разными наборами свойств – описание свойства выводится в нижней части окна свойств при выделении этого свойства.

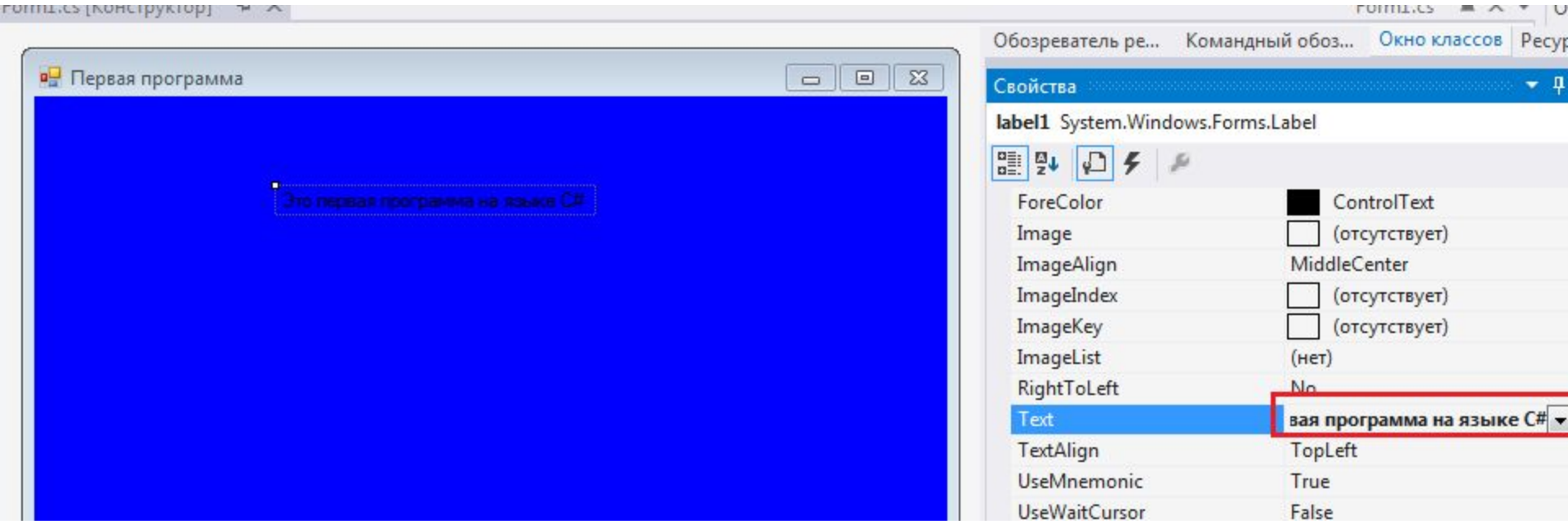
Первая программа



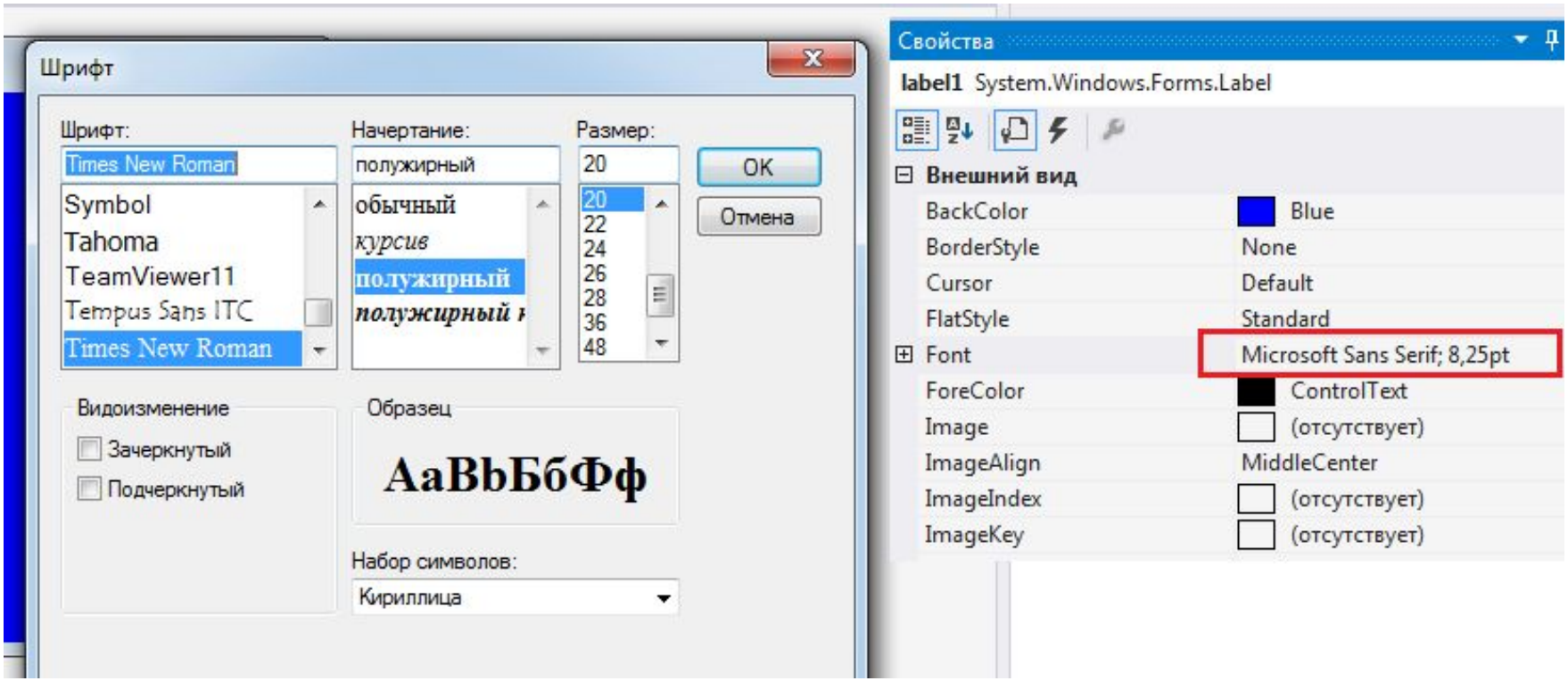
Первая программа



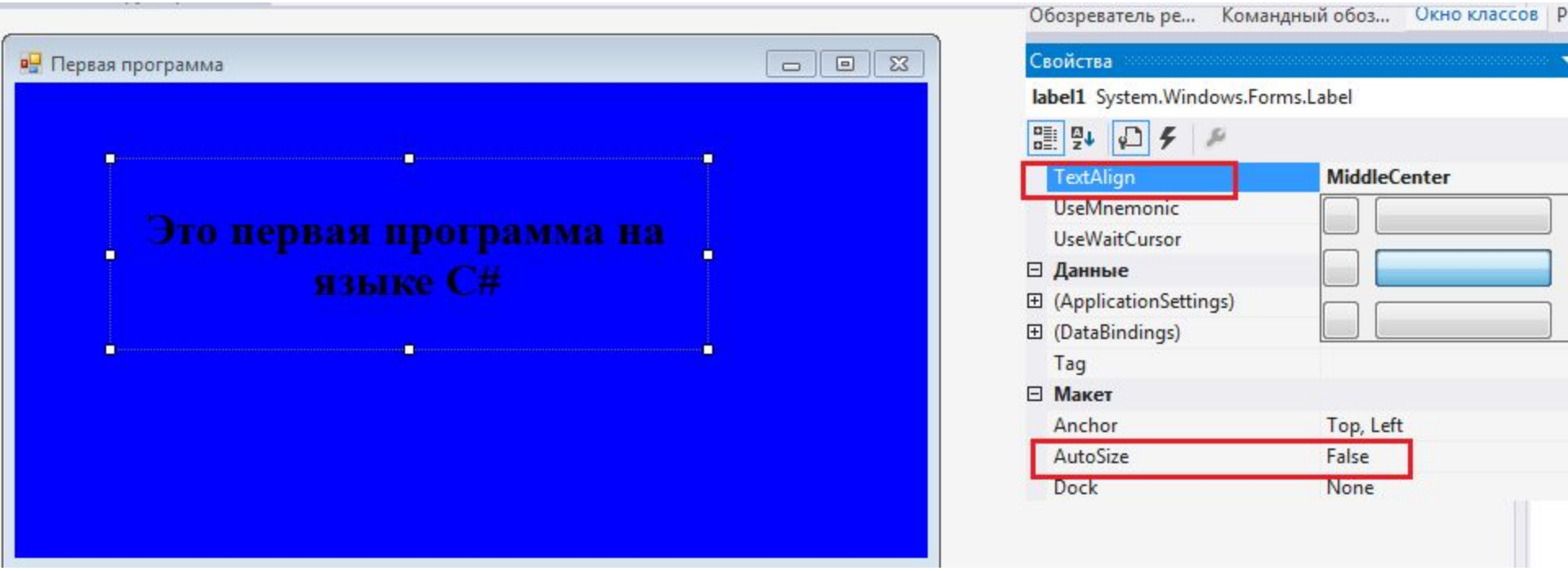
Первая программа



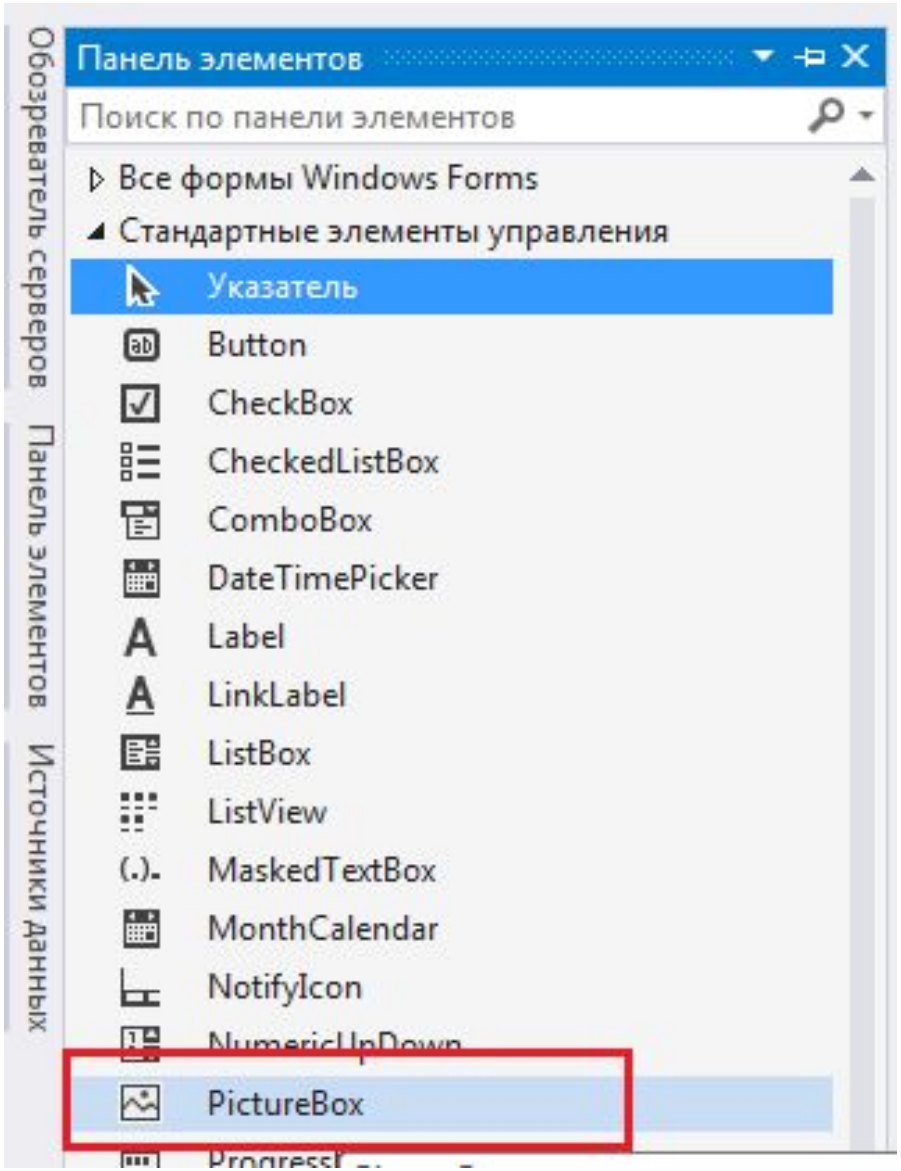
Первая программа



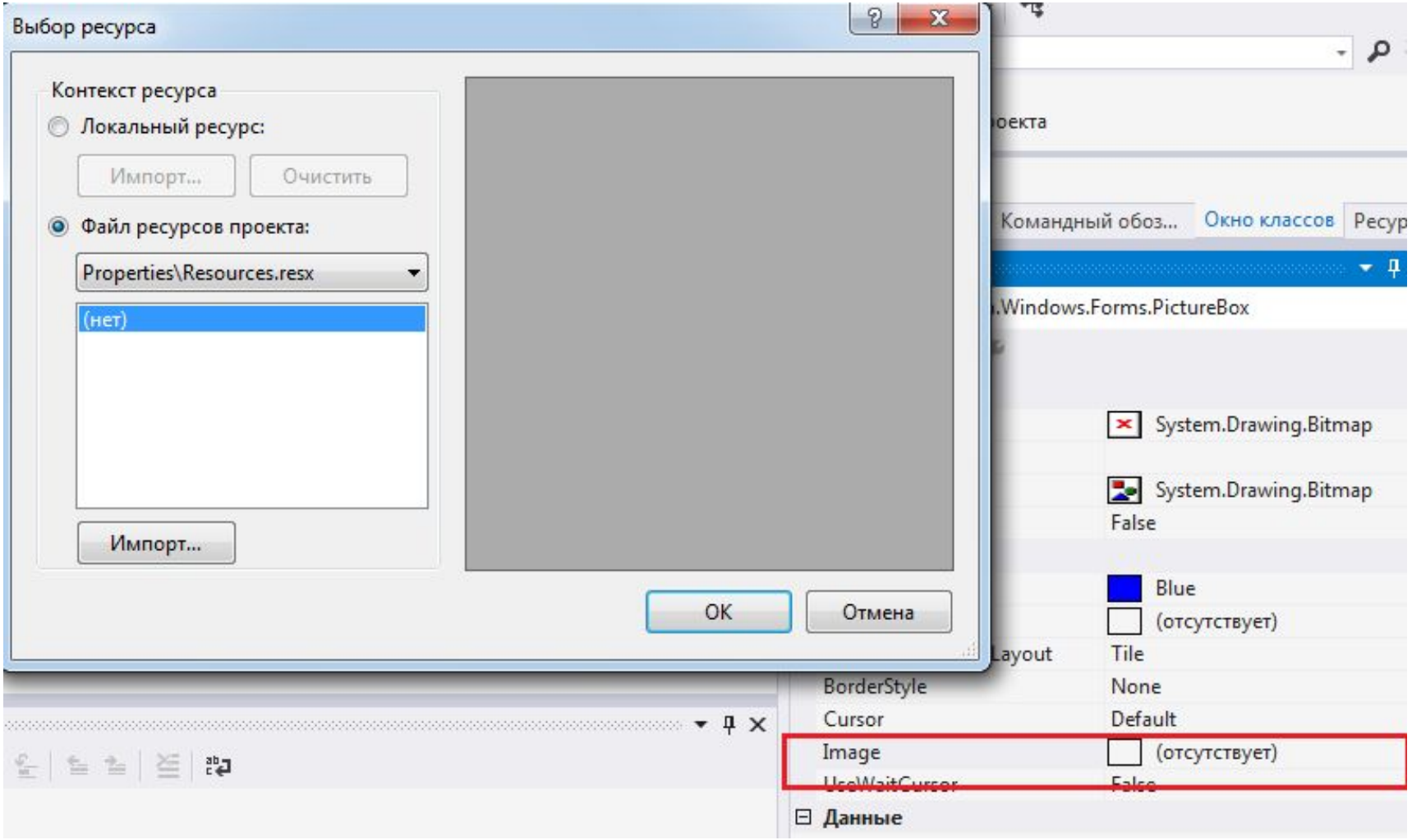
Первая программа



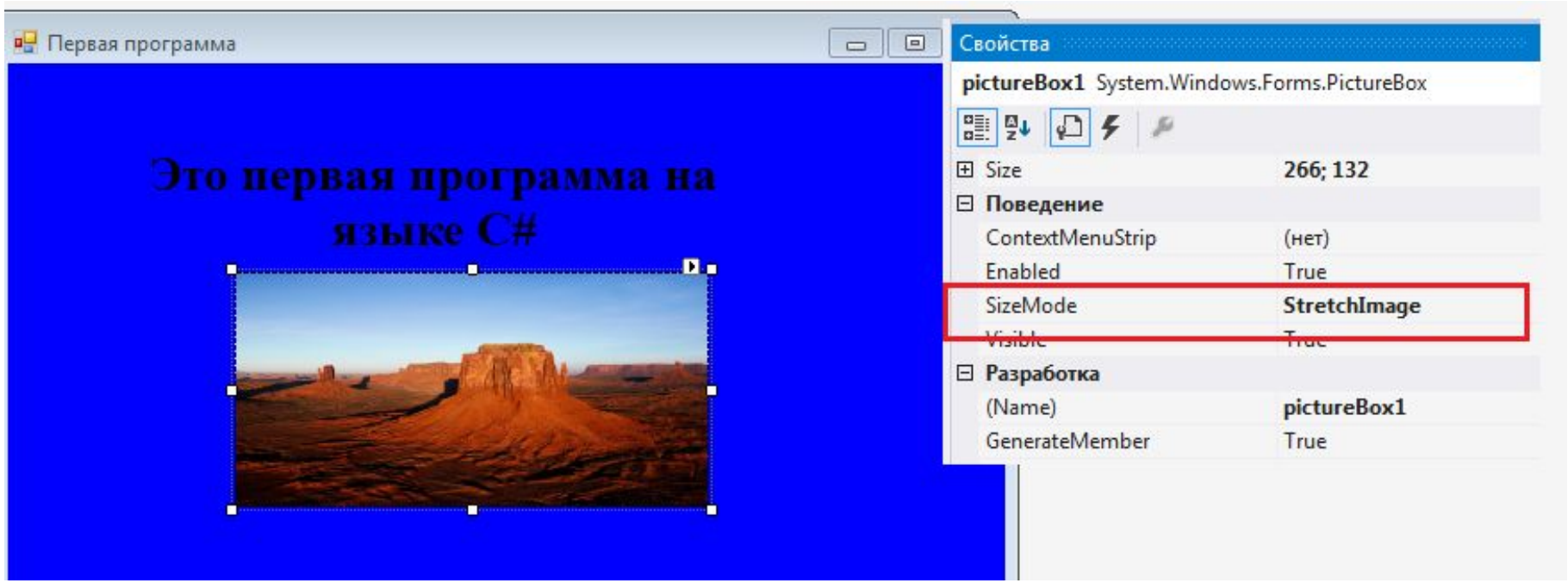
Первая программа



Первая программа



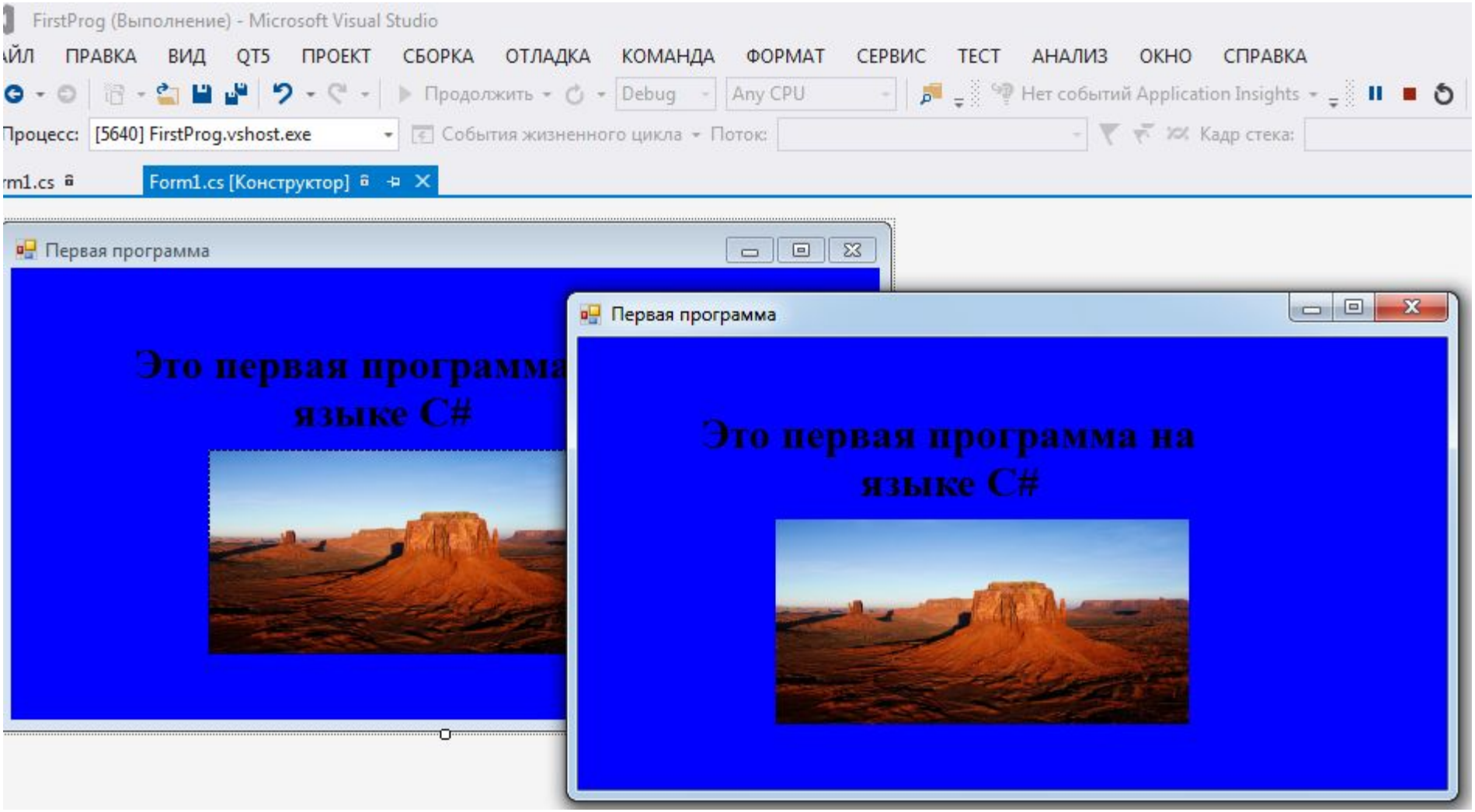
Первая программа



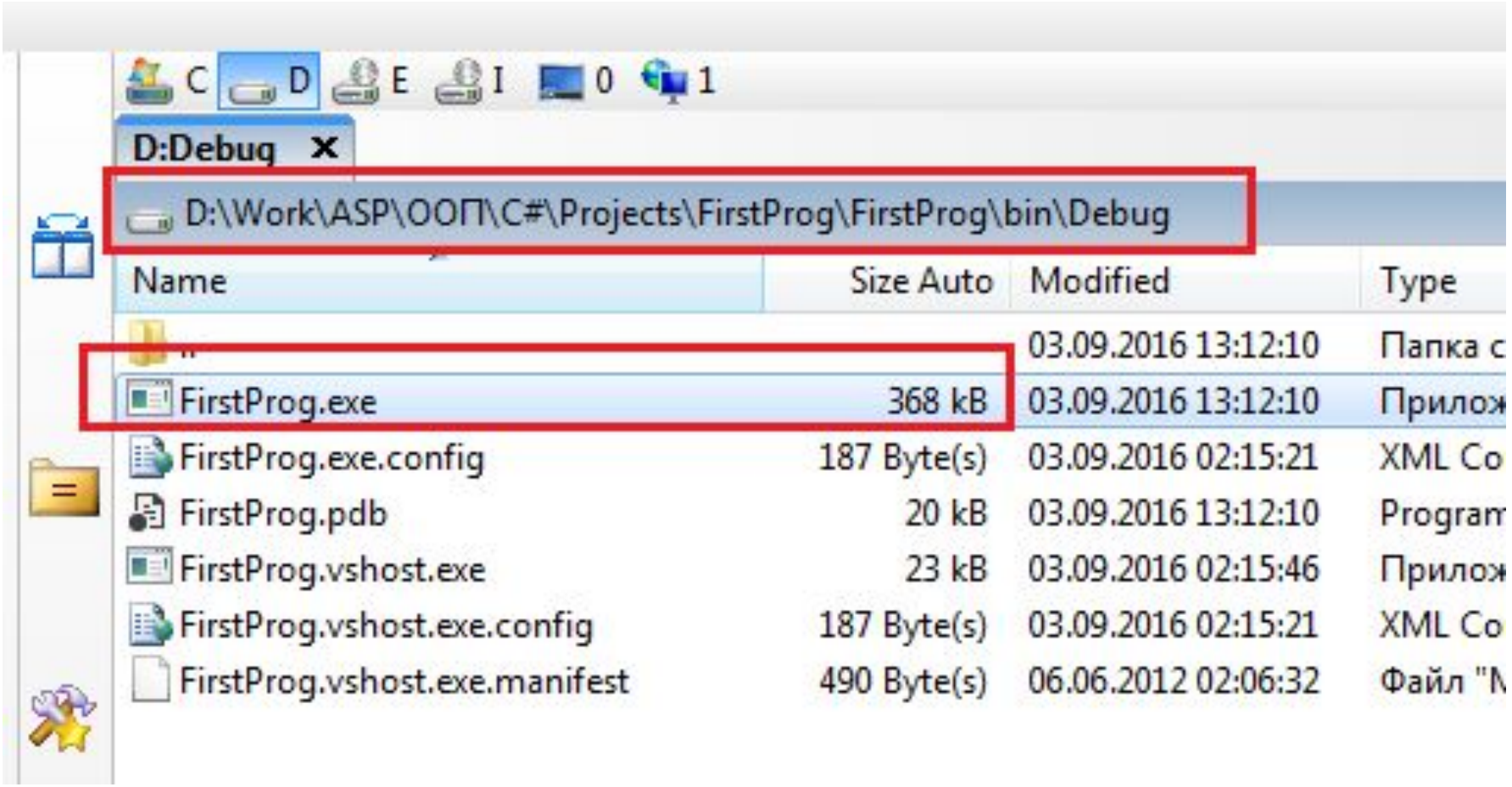
Сохраните проект

- Выполните команду FILE | Save AII, чтобы сохранить всё решение.
- В файле решения (расширение .sln) хранится имя и местонахождение проекта, а файл проекта (расширение .csproject) содержит имена и местонахождения всех файлов проекта.
- Если вы захотите снова открыть свой проект *в будущем,* просто откройте его файл .sln.

Запуск программы



Запуск программы



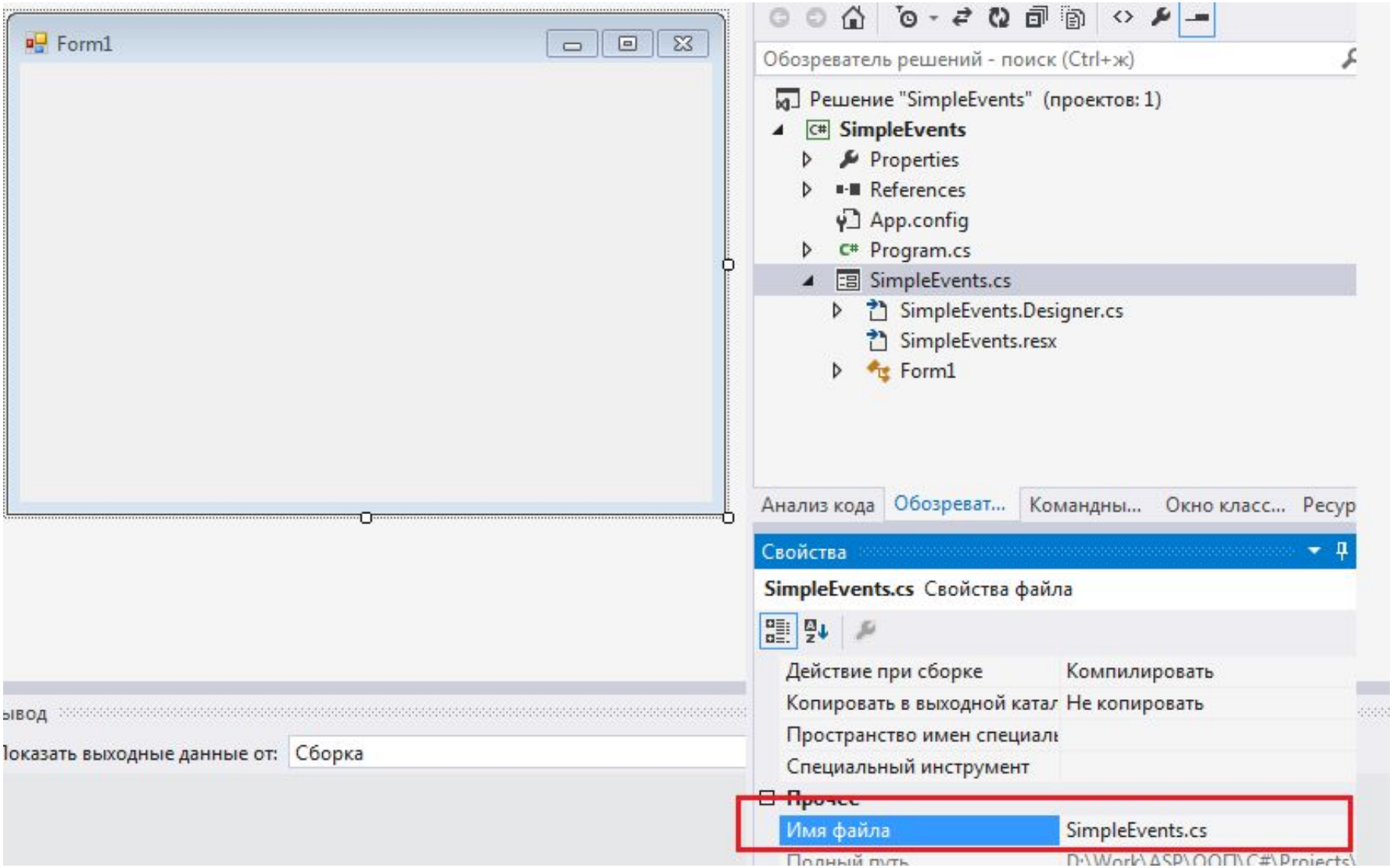
Форма

- Форма (Form) является контейнером для элементов управления и компонентов.
- При перетаскивании значков с панели элементов на форму Visual Studio генерирует код создания объекта и инициализации его основных свойств.
- Этот код обновляется при изменении свойств элемента управления или компонента в IDE.
- Удаление элемента управления или компонента с формы приводит к удалению соответствующего сгенерированного кода.
- IDE хранит сгенерированный код в отдельном файле с использованием *частичных классов - классов, разбитых на* несколько файлов и собираемых в один класс компилятором. Вы можете написать этот код самостоятельно, но гораздо проще поручить <<черную работу>> Visual Studio

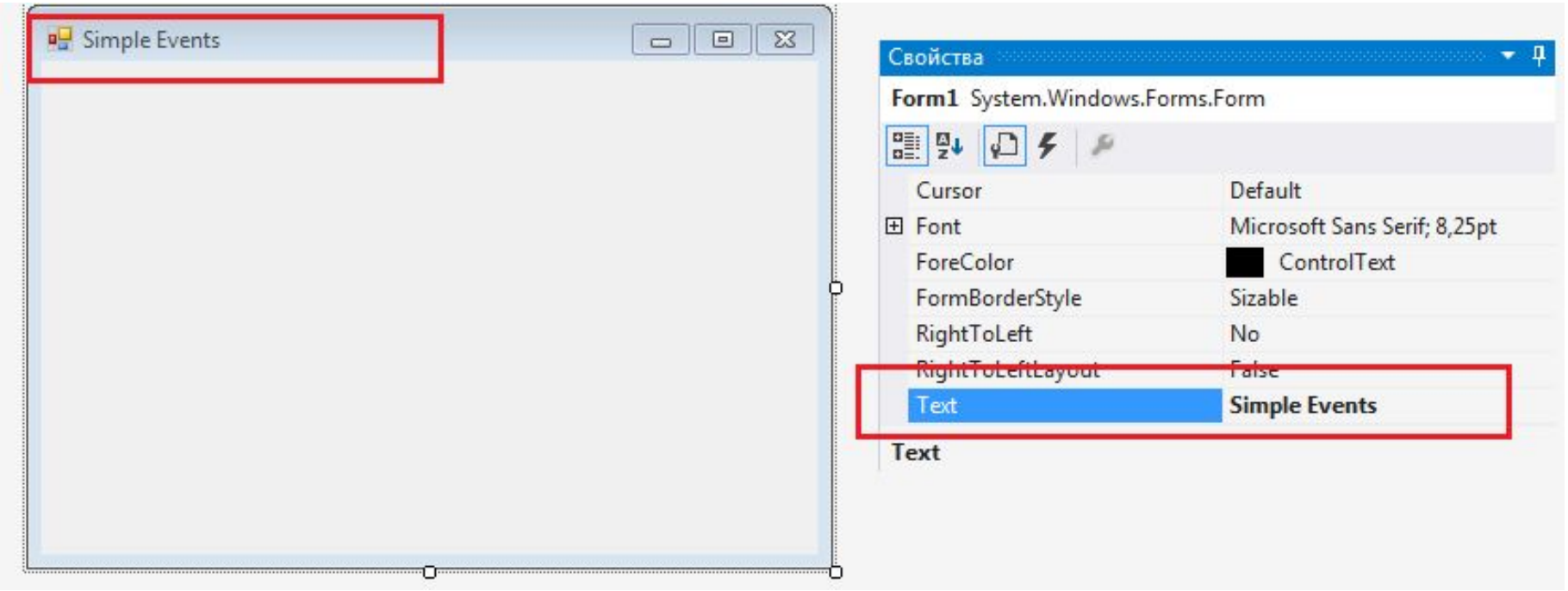
Обработка событий

- Обычно пользователь взаимодействует с графическим интерфейсом приложения для выполнения операций, которые должны выполняться приложением.
- Например, когда вы пишете сообщение в почтовом клиенте, нажатие кнопки Send приказывает приложению отправить сообщение по заданным адресам электронной почты.
- Графический интерфейс *управляется событиями. Когда пользователь взаимодействует* с компонентом, событие заставляет программу выполнить операцию.
- К распространенным событиям, по которым приложение выполняет операцию, относятся нажатия кнопок, ввод текста в текстовых полях, выбор варианта из меню, закрытие окна и перемещение мыши.
- Со всеми элементами управления графического интерфейса связываются события; объекты других типов также могут связываться
- с событиями. Метод, выполняющий операцию в ответ на событие, называется *обработчиком события (event handler), а общий процесс реагирования на события* называется *обработкой событий.*

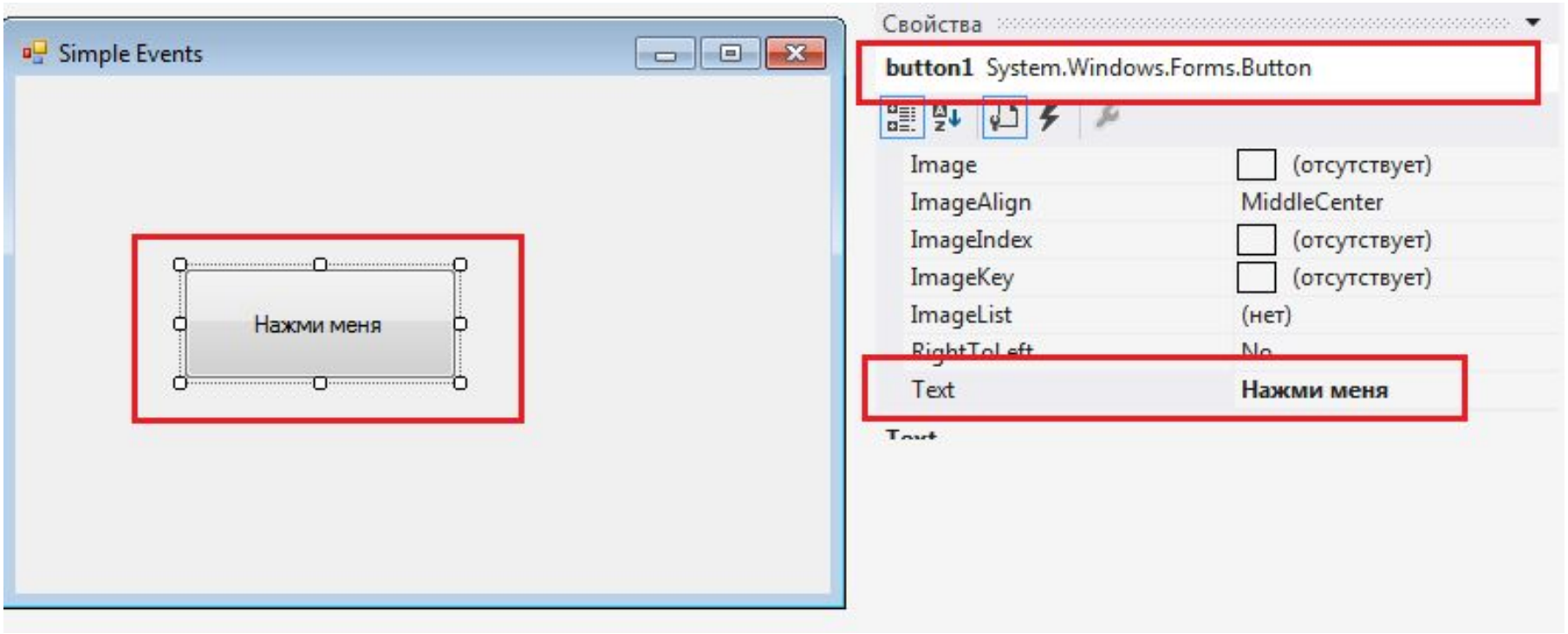
Обработка событий



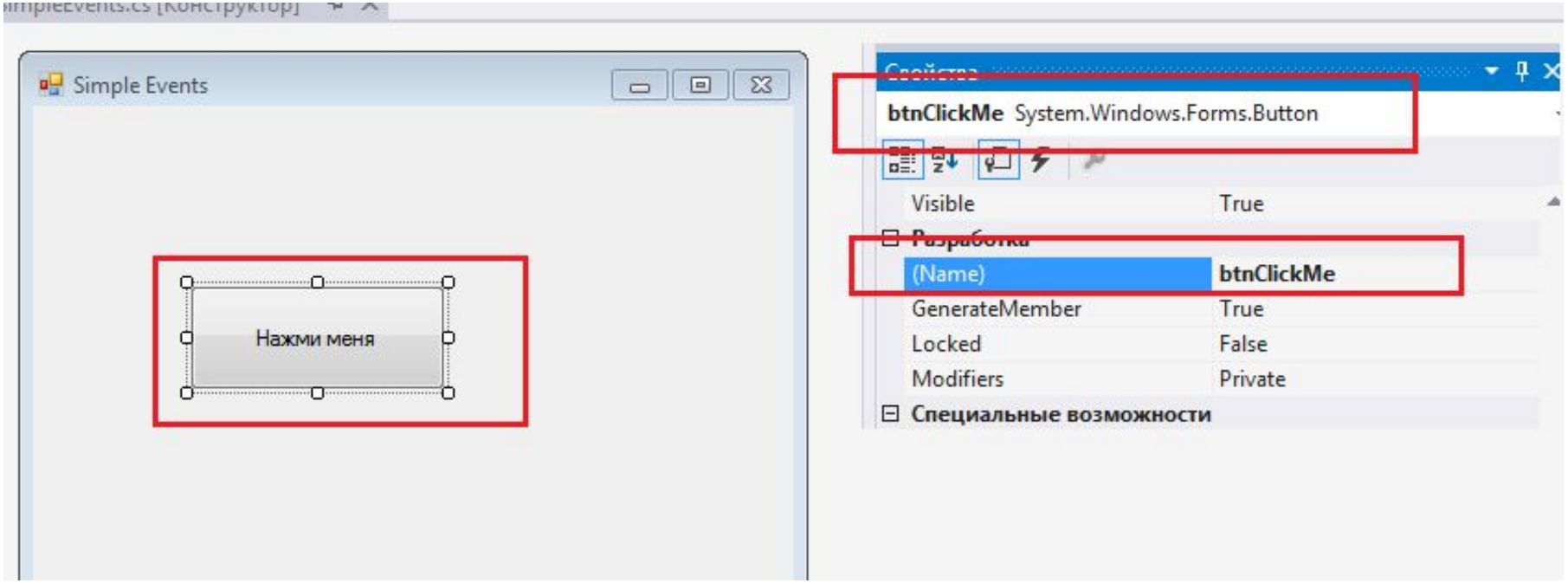
Обработка событий



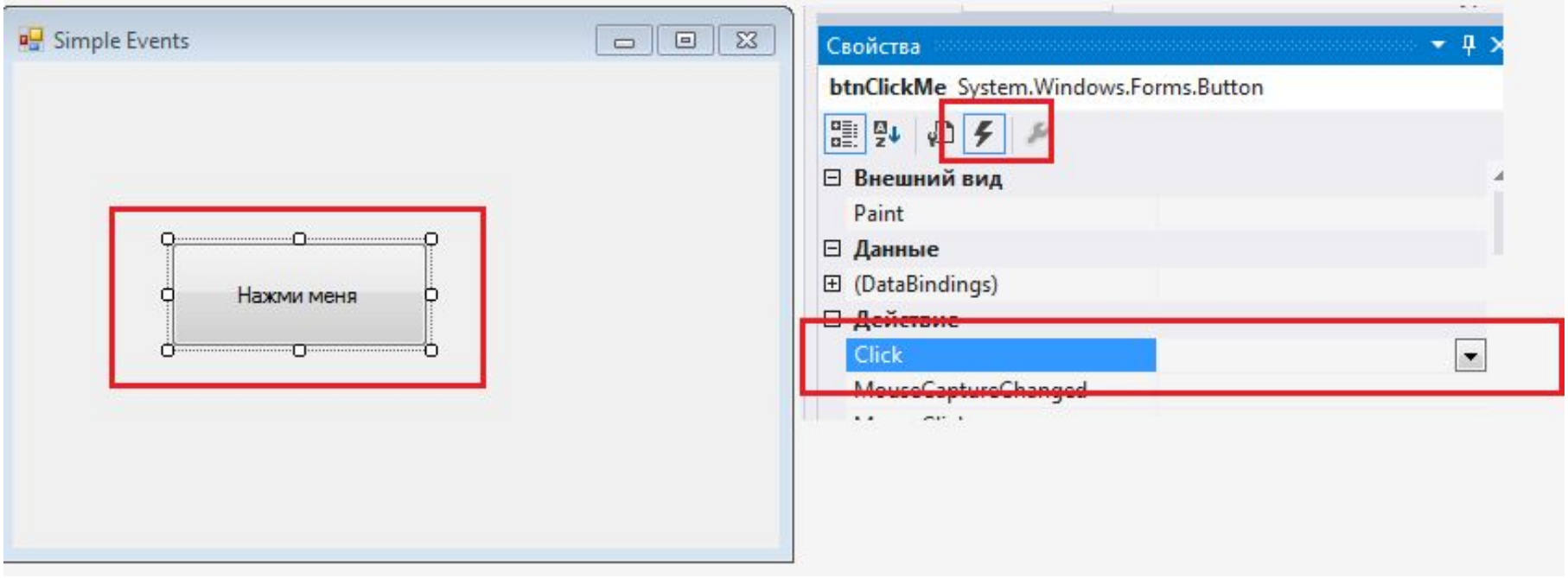
Добавление кнопки на форму



Добавление кнопки на форму



Добавление обработчика события **Click** для кнопки



Добавление обработчика события **Click** для кнопки

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

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

        private void btnClickMe_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Обработано событие нажатия кнопки");
        }
    }
}
```

Добавление обработчика события **Click** для кнопки

- Когда пользователь щелкает на кнопке, наше приложение должно вывести окно сообщения (MessageBox).
- Для этого следует назначить обработчик события Click объекта Button.
- Если сделать двойной щелчок на кнопке, *в программный код включается* следующий пустой обработчик события:
- private void clickButton_Click(object sender, EventArgs e)
- {
- }
- IDE строит имена методов обработчиков событий по схеме имяОбьекта_имяСобытия (например, clickButton_Click). Обработчик clickButton_Click обрабатывает событие щелчка (click) на элементе управления clickButton.

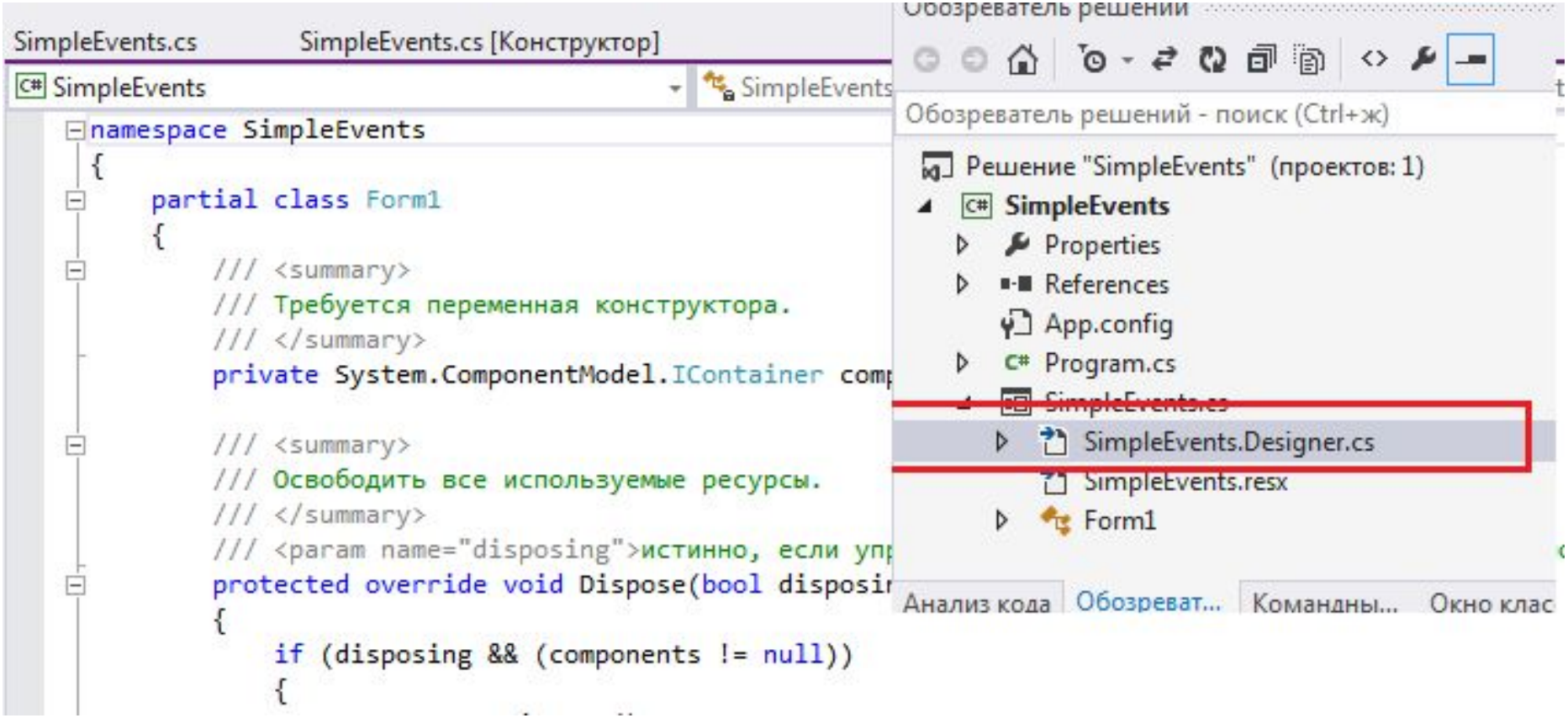
Параметры обработчиков событий

- Каждый обработчик события при вызове получает два параметра.
- Первый – обычно с именем **sender** - содержит ссылку на объект, сгенерировавший событие.
- Второй параметр содержит ссылку на объект **EventArgs** (или объект класса, производного от EventArgs), обычно с именем e.
- В этом объекте содержится дополнительная информация о возникшем событии. EventArgs - базовый класс для

Автоматически генерируемый код графического интерфейса

- Visual Studio помещает автоматически сгенерированный код графического интерфейса *в файл Designer.cs формы (SimpleEventExampleForm.Designer.cs в данном примере).*
- Так как код создается и поддерживается Visual Studio, обычно вам не приходится его просматривать. Более того, для построения приложений GUI понимать большую часть приведенного кода вообще не обязательно.

Автоматически генерируемый код графического интерфейса



Автоматически генерируемый код графического интерфейса

```
#region Код, автоматически созданный конструктором форм Windows

private void InitializeComponent()
        {
            this.btnClickMe = new System.Windows.Forms.Button();
            this.SuspendLayout();
            //
            // btnClickMe
            //
            this.btnClickMe.Location = new System.Drawing.Point(85, 97);
            this.btnClickMe.Name = "btnClickMe";
            this.btnClickMe.Size = new System.Drawing.Size(139, 57);
            this.btnClickMe.TabIndex = 0;
            this.btnClickMe.Text = "Нажми меня";
            this.btnClickMe.UseVisualStyleBackColor = true;
            this.btnClickMe.Click += new System.EventHandler(this.btnClickMe_Click);
            //
            // Form1
            //
}
```

Свойства и макет элемента управления

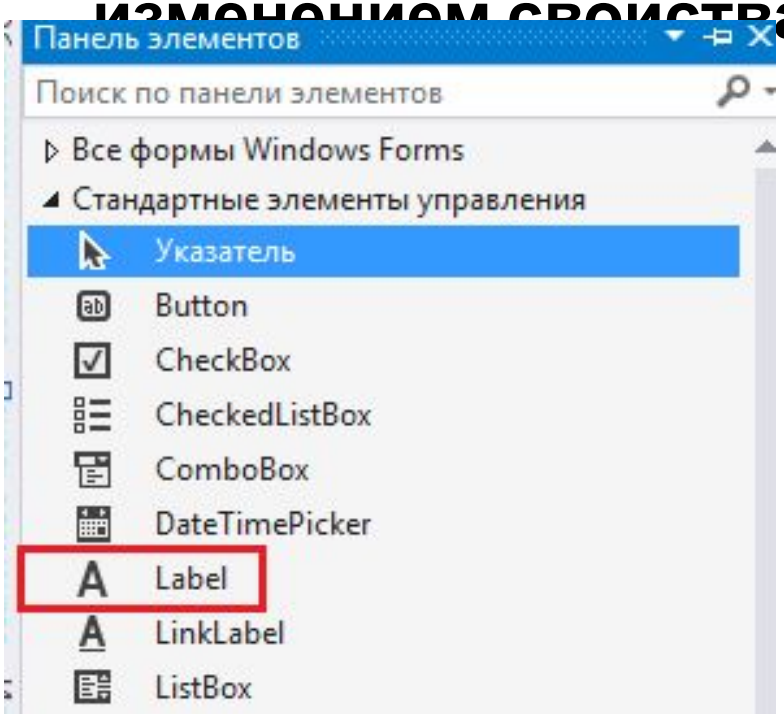
- Элементы управления наследуют от класса Control (пространство имен System. Windows. Forms). На рисунке перечислены некоторые свойства и методы класса Control, которые могут задаваться для многих элементов управления.
- Например, свойство **Text** определяет текст, выводимый на элементе управления. Местонахождение текста зависит от конкретного элемента: для формы текст выводится в заголовке, а для кнопки - непосредственно на поверхности.

Свойства и макет элемента управления

Свойства и методы класса Control	Описание
Часто используемые свойства	
BackColor	Цвет фона
BackgroundImage	Фоновое изображение
Enabled	Признак доступности элемента управления (то есть возможности взаимодействия с ним со стороны пользователя). Недоступный (заблокированный) элемент обычно выделяется серым цветом
Focused	Признак наличия у элемента фокуса ввода (свойство доступно только во время выполнения)
Font	Шрифт, используемый для вывода текста элемента
ForeColor	Основной цвет элемента управления; обычно определяет цвет текста, определяемого свойством Text
TabIndex	Индекс элемента управления в последовательности перебора. При нажатии клавиши Tab фокус ввода передается между элементами в последовательности, которая определяется разработчиком
TabStop	Если свойство равно true, элемент управления может получить фокус при нажатии клавиши Tab
Text	Текст, связанный с элементом. Местонахождение и оформление текста зависит от типа элемента
Visible	Признак видимости элемента

Надписи, текстовые поля и кнопки

- *Надписи используются для вывода текстовой информации (и возможно, графики)* и определяются классом **Label**, производным от Control.
- **Текст, выводимый в надписи, не может *напрямую изменяться пользователем (но может быть изменен на* программном уровне изменением свойства Text объекта Label).**



Надписи, текстовые поля и кнопки

Часто используемые свойства Label	Описание
Font	Шрифт текста надписи
Text	Текст надписи
TextAlign	Режим выравнивания текста в элементе управления — горизонтальный (по левому краю, по центру, по правому краю) и вертикальный (по верхнему краю, по центру, по нижнему краю). По умолчанию используется выравнивание по верхнему и левому краю

Надписи, текстовые поля и кнопки

Часто используемые свойства Label	Описание
Font	Шрифт текста надписи
Text	Текст надписи
TextAlign	Режим выравнивания текста в элементе управления — горизонтальный (по левому краю, по центру, по правому краю) и вертикальный (по верхнему краю, по центру, по нижнему краю). По умолчанию используется выравнивание по верхнему и левому краю