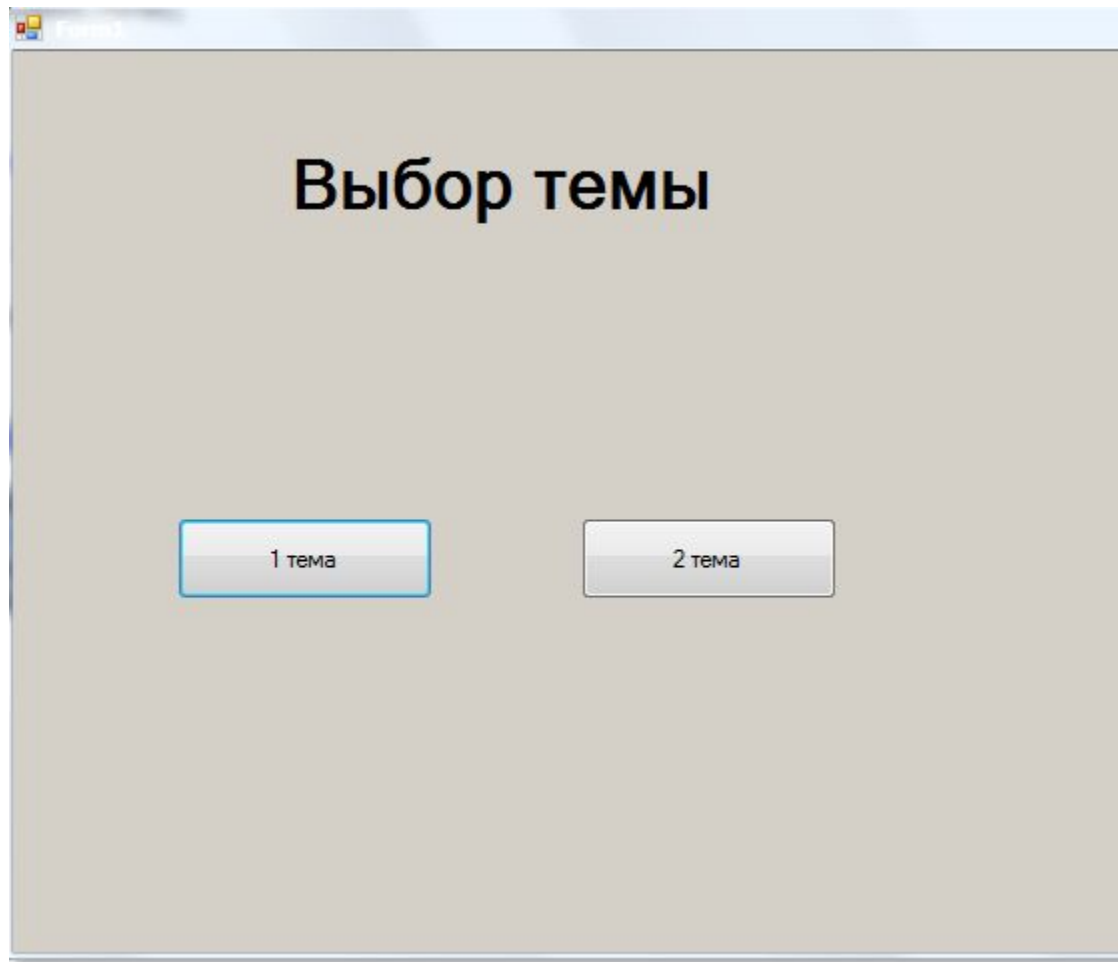


# Программирование на языке Си#

## **Тема 9. ФОРМА**

# Работа с конструктором Формы



# Создать проект

Последние файлы

.NET Framework 4.5

Сортировать по: По умолчанию

Установлено: Шаблоны - поиск (Ctrl-F)

Установленные

Шаблоны

Visual C#

Магазин Windows

Windows

Веб

Office

Cloud

Reporting

SharePoint

Silverlight

WCF

Windows Phone

Workflow

Тест













LightSwitch

Другие языки

Другие типы проектов

Примеры

В Интернете

	Приложение Windows Forms	Visual C#
	Приложение WPF	Visual C#
	Консольное приложение	Visual C#
	Приложение веб-форм ASP.NET	Visual C#
	Библиотека классов	Visual C#
	Переносимая библиотека классов	Visual C#
	Пустое приложение (XAML)	Visual C#
	Веб-приложение ASP.NET MVC 3	Visual C#
	Веб-приложение ASP.NET MVC 4	Visual C#
	Приложение таблицы (XAML)	Visual C#
	Приложение Silverlight	Visual C#
	Разделенное приложение (XAML)	Visual C#

Тип: Visual C#

Проект, для создания приложения с пользовательским интерфейсом Windows Forms

Имя:  WindowsFormsApplication1

Расположение: C:\Users\User\Documents\Visual Studio 2012\Projects

Имя решения: WindowsFormsApplication1

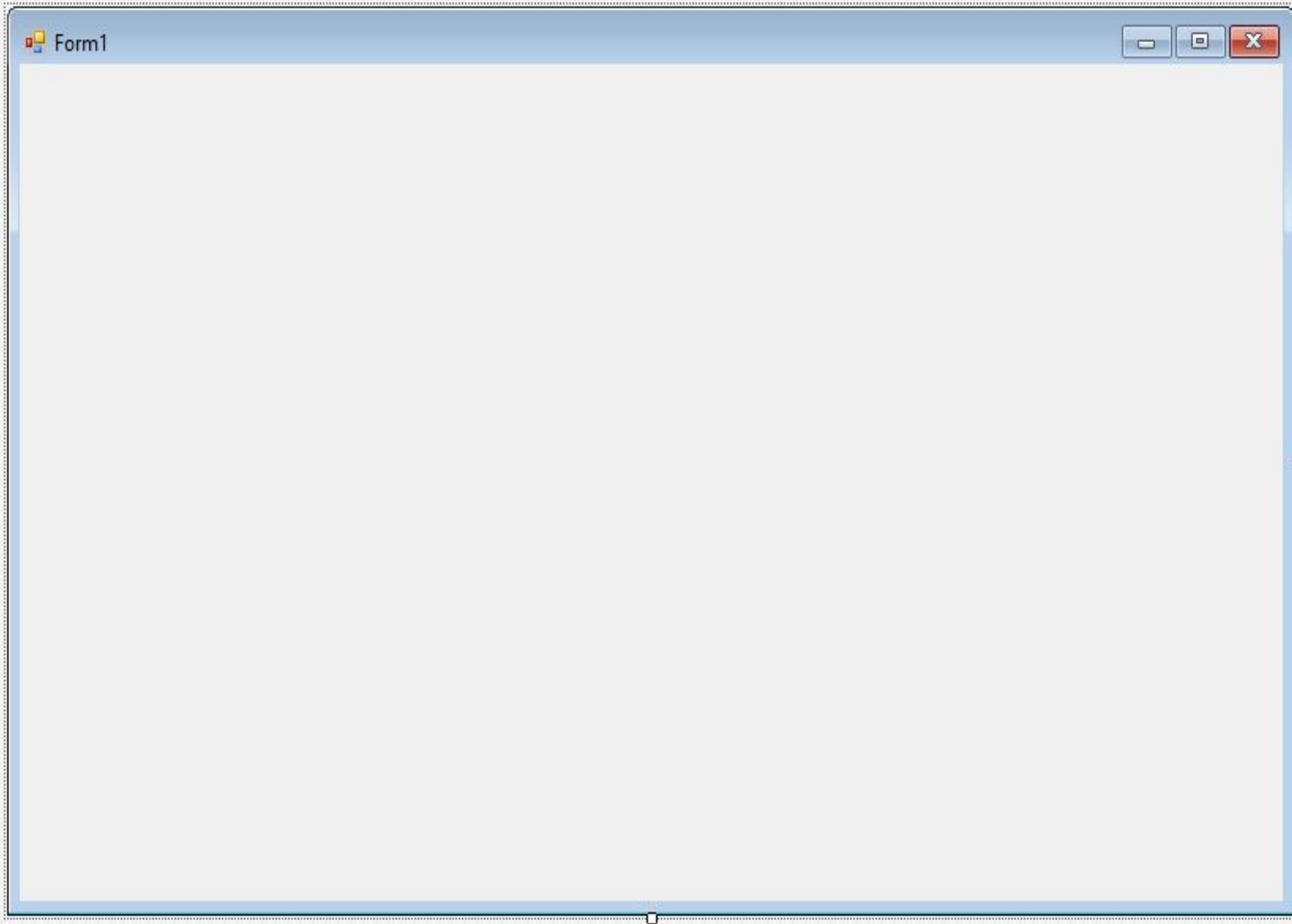
Обзор...

Создать каталог для решения

Добавить в систему управления версиями

OK

Отмена

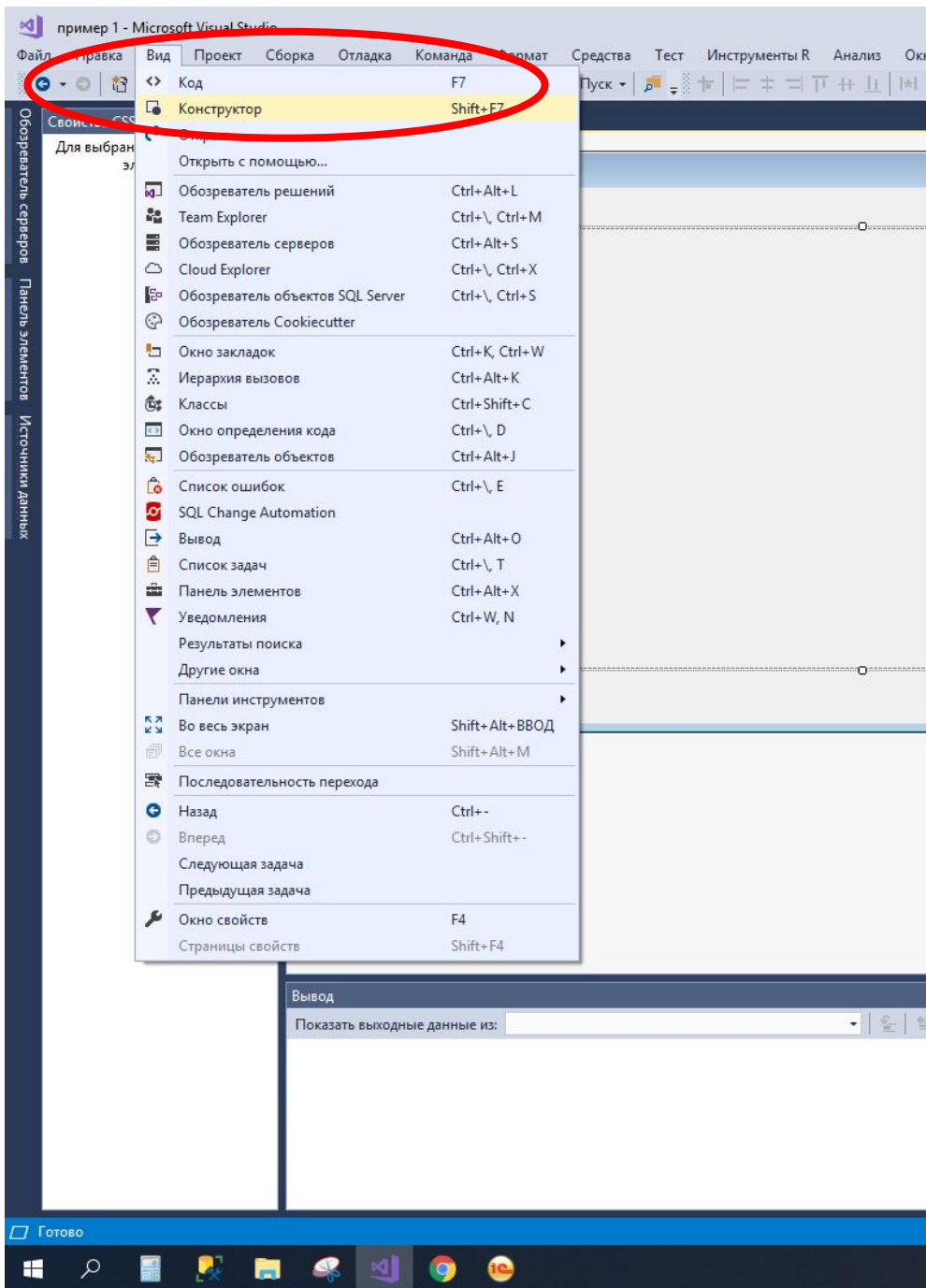


- ▲ **CS** пример 1
  - ▶ Properties
  - ▶ Ссылки
  - ▶ App.config
  - ▶ Form1.cs
  - ▶ Program.cs



## [-] Внешний вид

BackColor	<input type="text"/>	Control
BackgroundImage	<input type="text"/>	(отсутствует)
BackgroundImageLayout		Tile
Cursor		Default
Font		Microsoft Sans Serif;
ForeColor	<input type="text"/>	ControlText
FormBorderStyle		Sizable
RightToLeft		No



HelloApp - Microsoft Visual Studio

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM FORMAT TOOLS TEST ANALYZE WINDOW HELP

Евгений

Form1.cs [Design]

Form1

Структура проекта

Окно свойств

Графический дизайнер формы

Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'HelloApp' (1 project)

- C# HelloApp
  - Properties
  - References
  - App.config
  - Form1.cs
  - Program.cs

Server Explo... Solution Ex... Team Explo...

Properties

Form1 System.Windows.Forms.Form

Size	300; 300
SizeGripStyle	Auto
StartPosition	WindowsDefaultLoc...
Tag	
Text	<b>Form1</b>
TopMost	False

**Text**

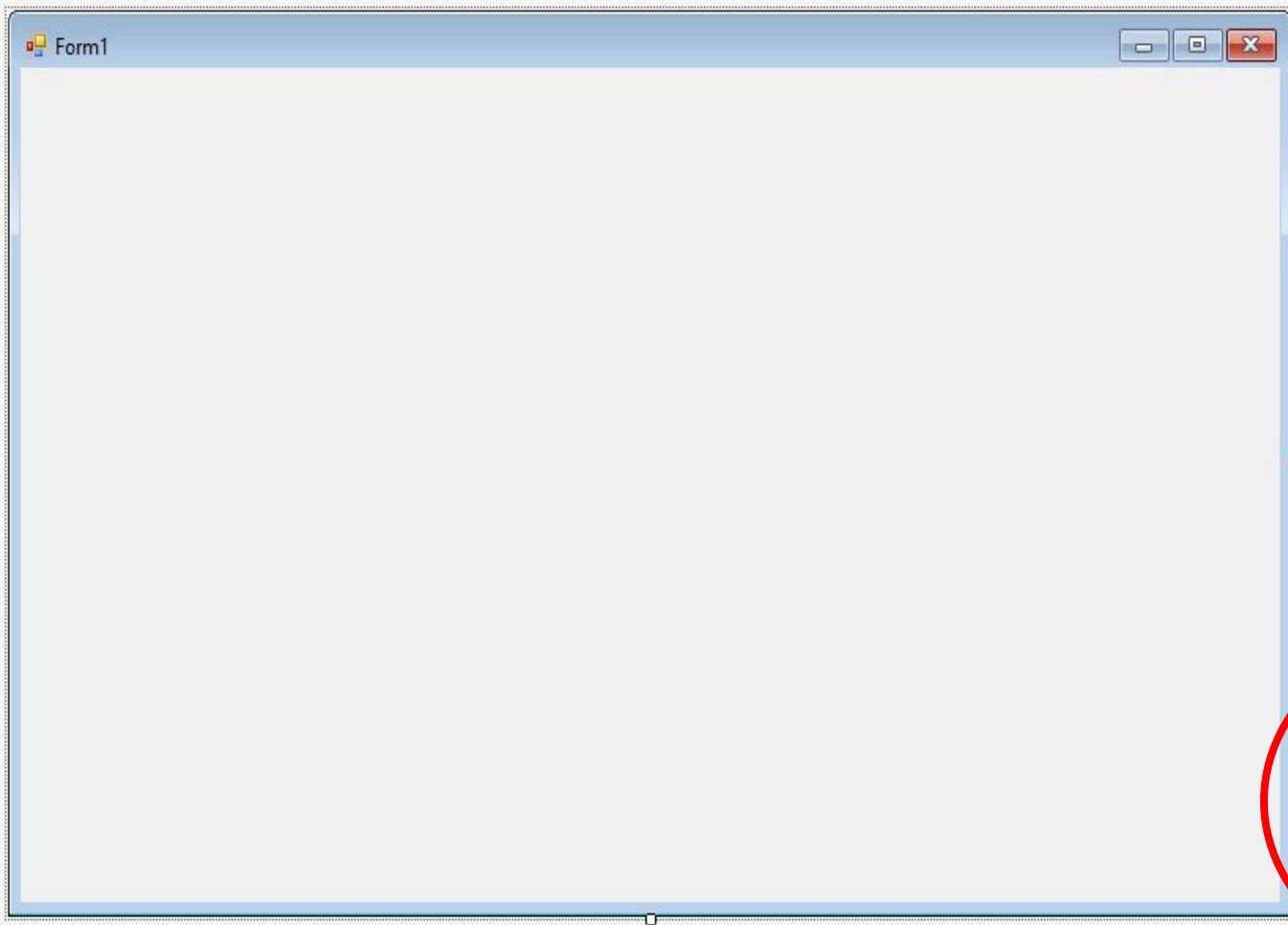
The text associated with the control.

Output

Show output from: Xamarin Log

Package Manager Console Data Tools Operations Error List Output

Ready



Обозреватель решений — поиск (Ctrl+ж)

Решение "пример 1" (проекты: 1 из 1)

- CS: пример 1
  - Properties
  - Ссылки
  - App.config
  - Form1.cs
  - Program.cs

Обозреватель решений Team Explorer

Свойства

Form1 System.Windows.Forms.Form



Внешний вид	
BackColor	Control
BackgroundImage	(отсутствует)
BackgroundImageLayout	Tile
Cursor	Default
Font	
ForeColor	ControlText
FormBorderStyle	Sizable
RightToLeft	No



MainForm.cs [Конструктор]\*

**Свойства**

**Form1** System.Windows.Forms.Form

ContextMenuStrip	(нет)
ControlBox	True
Cursor	Default
DoubleBuffered	False
Enabled	True
Font	Microsoft Sans Serif, 8.25pt
ForeColor	ControlText
<b>FormBorderStyle</b>	<b>None</b>
HelpButton	False
Icon	(Значок)
ImeMode	NoControl
IsMdiContainer	False
KeyPreview	False

Обозреватель решений

Обозреватель решений — поиск (Ctrl+ж)

- Решение "MyFirstForm" (проектов: 1)
  - MyFirstForm
    - Properties
    - Ссылки
    - App.config
    - MainForm.cs
      - MainForm.Designer.cs

Панель элемен... Обозреватель... Team Explorer... Представление...

**Свойства**

**MainForm** System.Windows.Forms.Form

RightToLeft	No
RightToLeftLayout	False
<b>Text</b>	<b>Form 1</b>
UseWaitCursor	False

**Данные**

**Макет**

**Поведение**

**Прочее**

<b>(Name)</b>	<b>MainForm</b>
Language	(По умолчанию)

**Прочее**

Обозреватель серверов  
Источники данных

Уведомления  
Средства диагностики



Свойства

Form1 System.Windows.Forms.Form

**Внешний вид**

BackColor	<input type="checkbox"/> Control
BackgroundImage	<input type="checkbox"/> (отсутствует)
BackgroundImageLayout	Tile
Cursor	Default
Font	Microsoft Sans Serif; 8,25pt
ForeColor	<input type="checkbox"/> ControlText
FormBorderStyle	Sizable
RightToLeft	No
RightToLeftLayout	False
Text	<b>Form1</b>
UseWaitCursor	False

**Данные**

(ApplicationSettings)

(DataBindings)

Tag

**Макет**

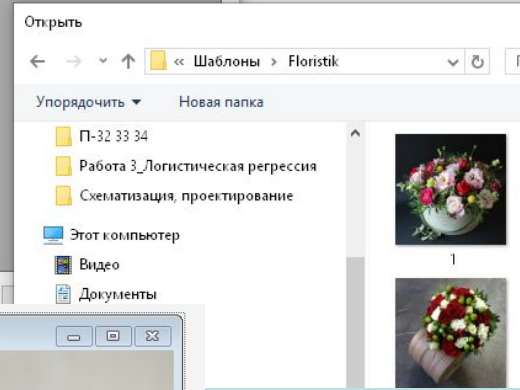
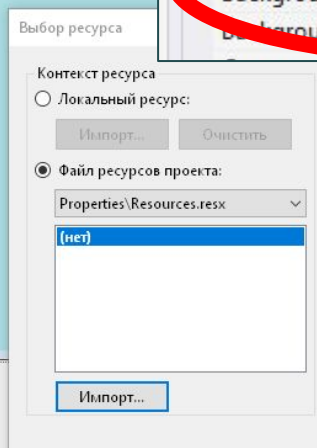
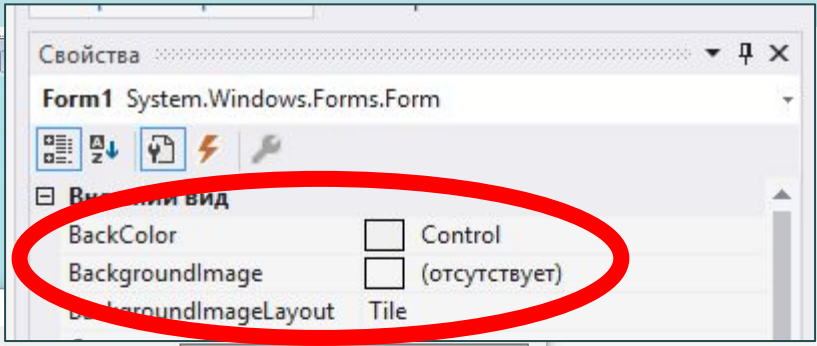
AutoScaleMode	<b>Font</b>
AutoScroll	False
AutoScrollMargin	0; 0
AutoScrollMinSize	0; 0
AutoSize	False

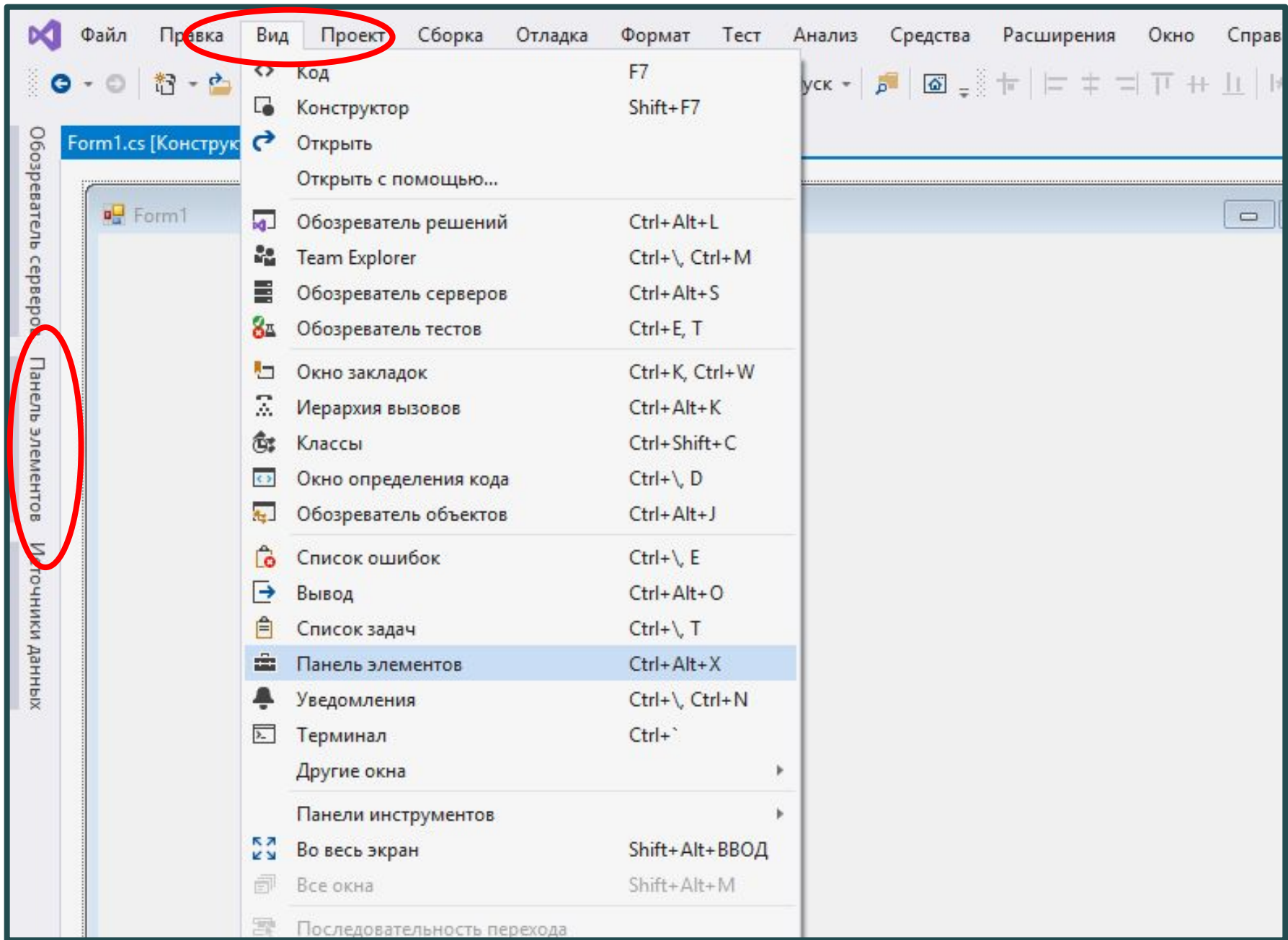
**Text**

Текст, связанный с элементом управления.

AutoSize	False
AutoSizeMode	GrowOnly
<b>Location</b>	0; 0
X	0
Y	0
MaximumSize	0; 0
MinimumSize	0; 0
Padding	0; 0; 0; 0
<b>Size</b>	<b>816; 489</b>
Width	<b>816</b>
Height	<b>489</b>
StartPosition	WindowsDefaultLocation

<b>Font</b>	<b>Microsoft Sans Serif; 8,25pt</b>
ForeColor	<input type="checkbox"/> ControlText
FormBorderStyle	Sizable
RightToLeft	No
RightToLeftLayout	False
Text	<b>Form1</b>
UseWaitCursor	False





## Основные группы элементов управления

Категория	Интерфейсные элементы
Редактирование текста	<b>TextBox, RichTextBox</b>
Отображение текста	<b>Label, LinkLabel, StatusBar</b>
Выбор из списка	<b>CheckedListBox, ComboBox, DomainUpDown, ListBox, ListView, NumericUpDown, TreeView</b>
Отображение графики	<b>PictureBox</b>
Хранение графики	<b>ImageList</b>
Ввод значений	<b>CheckBox, CheckedListBox, RadioButton, TrackBar</b>
Ввод даты	<b>DateTimePicker, MonthCalendar</b>
Диалоговые панели	<b>ColorDialog, FontDialog, OpenFileDialog, PrintDialog, PrintPreviewDialog, SaveFileDialog</b>
Создание меню	<b>MenuStrip, ContextMenuStrip</b>
Команды	<b>Button, LinkLabel, NotifyIcon, ToolBar</b>
Объединение компонентов	<b>Panel, GroupBox, TabControl</b>



Элементы управления представляют собой визуальные классы, которые получают введенные пользователем данные и могут инициировать различные события. Все элементы управления наследуются от класса Control и поэтому имеют ряд общих свойств:

- 1) **Anchor**: Определяет, как элемент будет растягиваться
- 2) **BackColor**: Определяет фоновый цвет элемента
- 3) **BackgroundImage**: Определяет фоновое изображение элемента
- 4) **ContextMenu**: Контекстное меню, которое открывается при нажатии на элемент правой кнопкой мыши. Задается с помощью элемента ContextMenu
- 5) **Cursor**: Представляет, как будет отображаться курсор мыши при наведении на элемент
- 6) **Dock**: Задаёт расположение элемента на форме
- 7) **Enabled**: Определяет, будет ли доступен элемент для использования. Если это свойство имеет значение False, то элемент блокируется.
- 8) **Font**: Устанавливает шрифт текста для элемента
- 9) **ForeColor**: Определяет цвет шрифта
- 10) **Location**: Определяет координаты верхнего левого угла элемента управления
- 11) **Name**: Имя элемента управления
- 12) **Size**: Определяет размер элемента
- 13) **Width**: ширина элемента
- 14) **Height**: высота элемента
- 15) **TabIndex**: Определяет порядок обхода элемента по нажатию на клавишу Tab
- 16) **Tag**: Позволяет сохранять значение, ассоциированное с этим элементом управления

Свойства

button4 System.Windows.Forms.B

UseMnemonic **События**

UseVisualStyleE **True**

UseWaitCursor False

**Данные**

(ApplicationSet)

(DataBindings)

Tag

**Макет**

Anchor Top, Left

AutoSize False

AutoSizeMode GrowOnly

Dock None

Location **80; 306**

Margin 3; 3; 3; 3

MaximumSize 0; 0

MinimumSize 0; 0

Padding 0; 0; 0; 0

Size **153; 62**

**Поведение**

Уведомления

button4 System.Windows.Fc

Click

MouseCapture

MouseClicked

**Ключ**

KeyDown

KeyPress

KeyUp

PreviewKeyDov

**Макет**

Layout

MarginChange

Move

PaddingChang

Resize

**Мышь**

MouseDown

MouseEnter

MouseHover

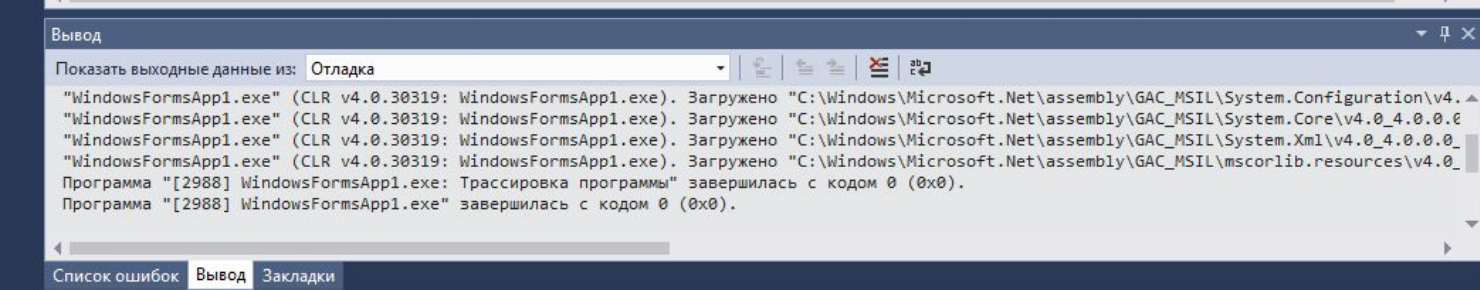
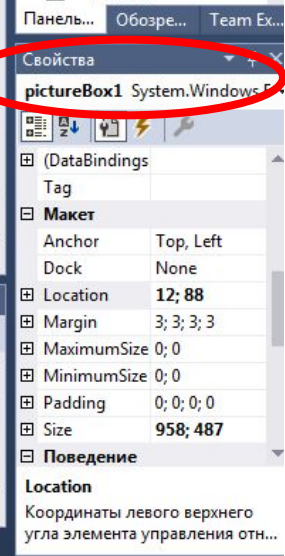
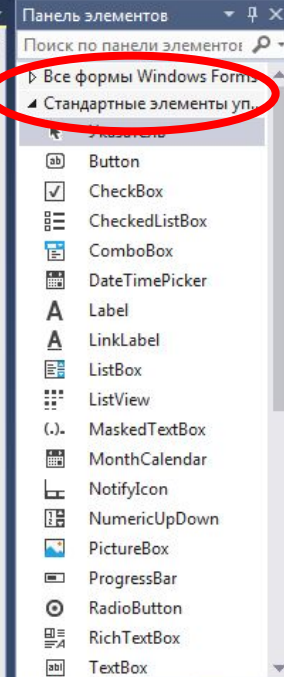
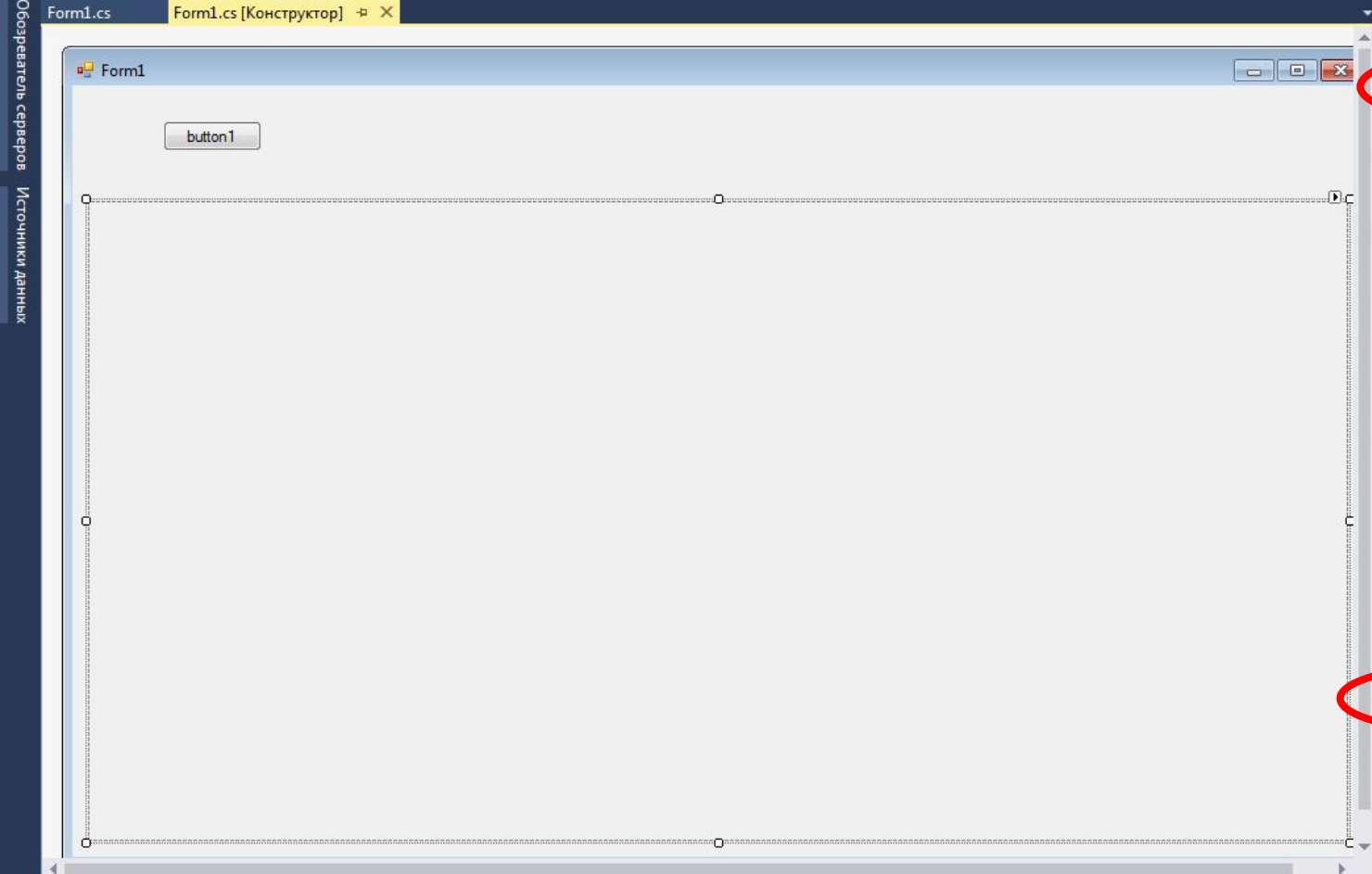
MouseLeave

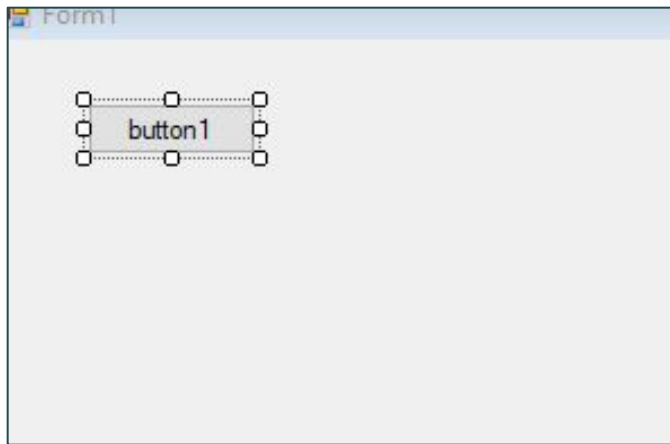
MouseMove



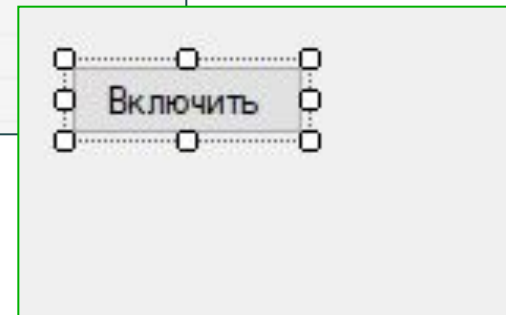
## Задание 1

Поместить на форму кнопку «Включить» и окно pictureBox . Загрузить картинку в окно pictureBox.





Font	Microsoft Sans Serif, 8,25pt
ForeColor	ControlText
Image	(отсутствует)
ImageAlign	MiddleCenter
ImageIndex	(отсутствует)
ImageKey	(отсутствует)
ImageList	(нет)
RightToLeft	No
<b>Text</b>	<b>button1</b>
TextAlign	MiddleCenter
TextImageRelation	Overlay
UseMnemonic	True
UseVisualStyleBackColor	True



PictureBox предназначен для показа изображений. Он позволяет отобразить файлы в формате bmp, jpg, gif, а также метафайлы изображений и иконки.

Для установки изображения в PictureBox можно использовать ряд свойств:

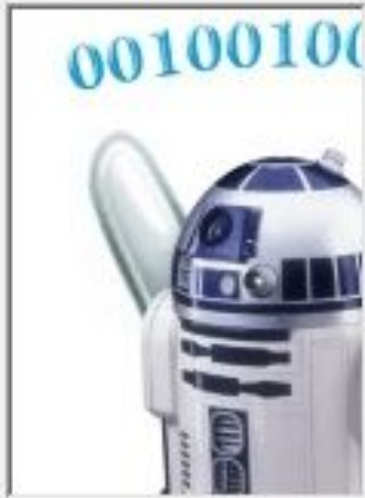
- **Image**: устанавливает объект типа Image
- **ImageLocation**: устанавливает путь к изображению на диске или в интернете
- **InitialImage**: некоторое начальное изображение, которое будет отображаться во время загрузки главного изображения, которое хранится в свойстве Image
- **ErrorImage**: изображение, которое отображается, если основное изображение не удалось загрузить в PictureBox

## 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 образом, чтобы 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**: изображение подстраивается под размеры PictureBox, сохраняя при этом пропорции

Normal



AutoSize



StretchImage



CenterImage

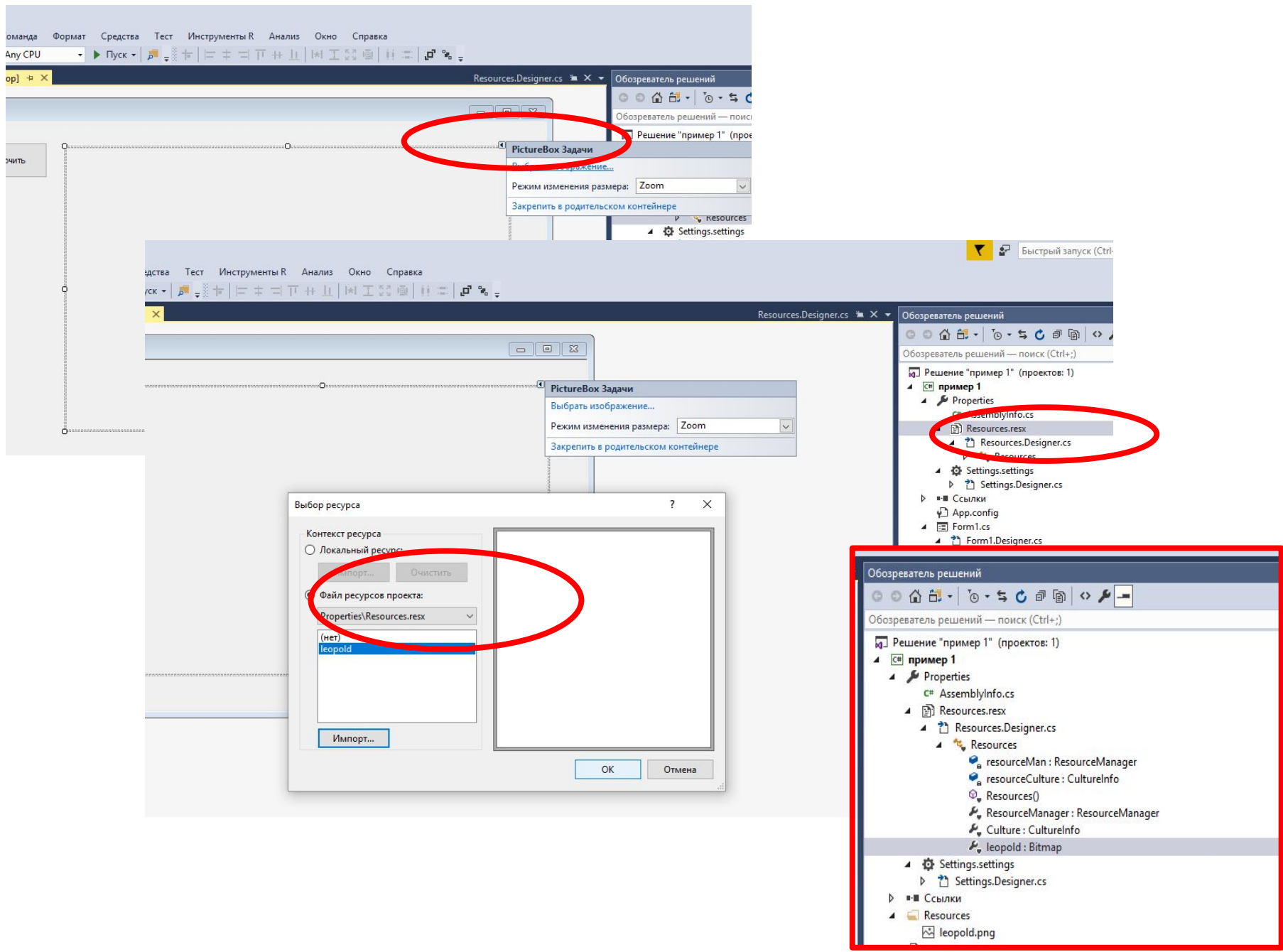


Zoom

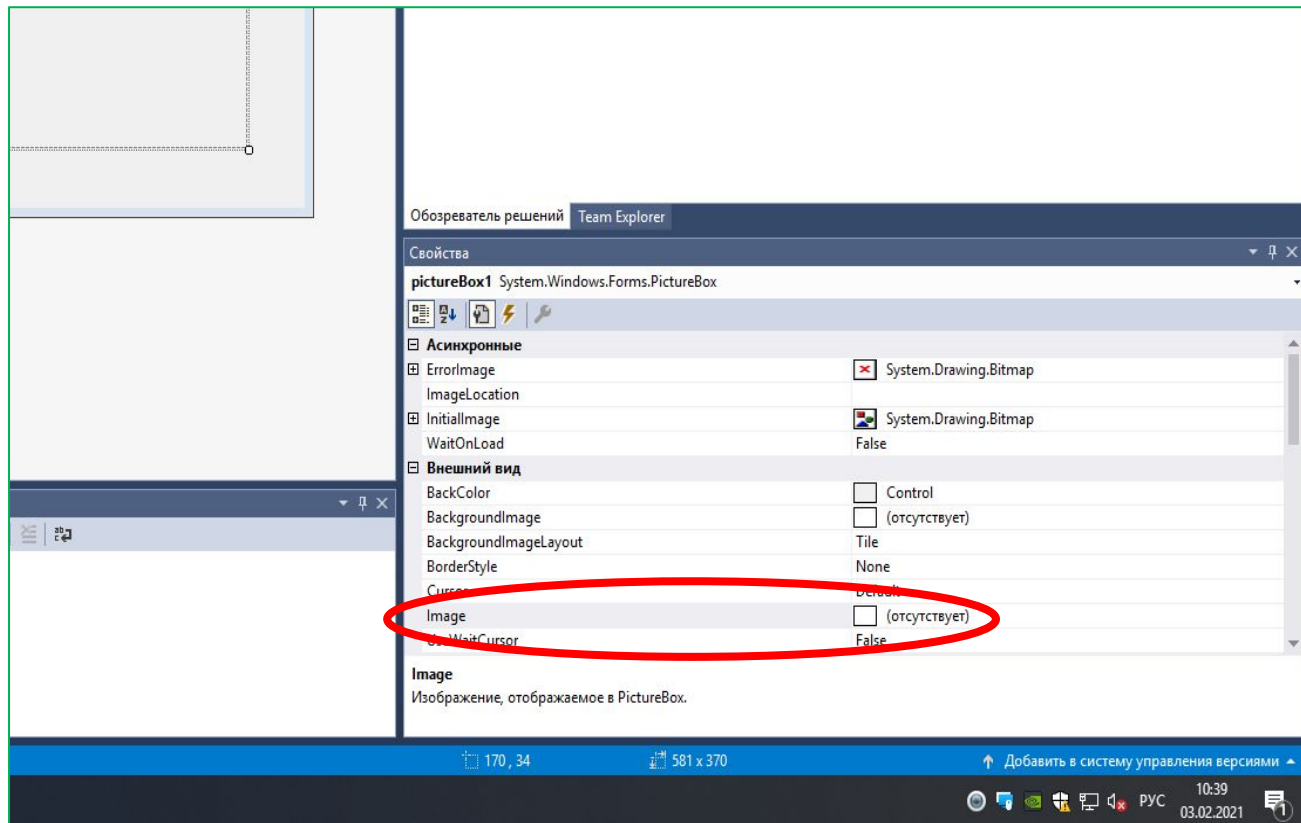




# 1 способ



2 способ



## Задание 2

Поместить на форму кнопку «Включить» и окно pictureBox . При нажатии на кнопку появляется картинка из коллекции.

Microsoft Visual Studio

button4 System.Windows.Forms.Button

Click

MouseDown

MouseEnter

MouseHover

MouseMove

MouseLeave

MouseClick

KeyDown

KeyPress

KeyUp

PreviewKeyDown

Layout

MarginChange

Move

PaddingChange

Resize

Свойства

button4 System.Windows.Forms.Button

События

UseMnemonic

UseVisualStyleE True

UseWaitCursor False

Данные

(ApplicationSet)

(DataBindings)

Tag

Макет

Anchor Top, Left

AutoSize False

AutoSizeMode GrowOnly

Dock None

Location 80; 306

Margin 3; 3; 3; 3

MaximumSize 0; 0

MinimumSize 0; 0

Padding 0; 0; 0; 0

Size 153; 62

Поведение

Свойства

button4 System.Windows.Forms.Button

UseMnemonic

UseVisualStyleE True

UseWaitCursor False

Данные

(ApplicationSet)

(DataBindings)

Tag

Макет

Anchor Top, Left

AutoSize False

AutoSizeMode GrowOnly

Dock None

Location 80; 306

Margin 3; 3; 3; 3

MaximumSize 0; 0

MinimumSize 0; 0

Padding 0; 0; 0; 0

Size 153; 62

Поведение

AllowDrop False

AutoEllipsis False

ContextMenuS (net)

DialogResult None

Enabled True

TabIndex 7

TabStop True

UseCompatible False

Visible True

Разработка

(Name) button4

GenerateMemb True

Locked False

Modifiers Private

Специальные возможности

AccessibleDesc

AccessibleNam

AccessibleRole Default

Фокус

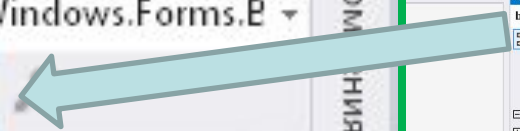
CausesValidatic True

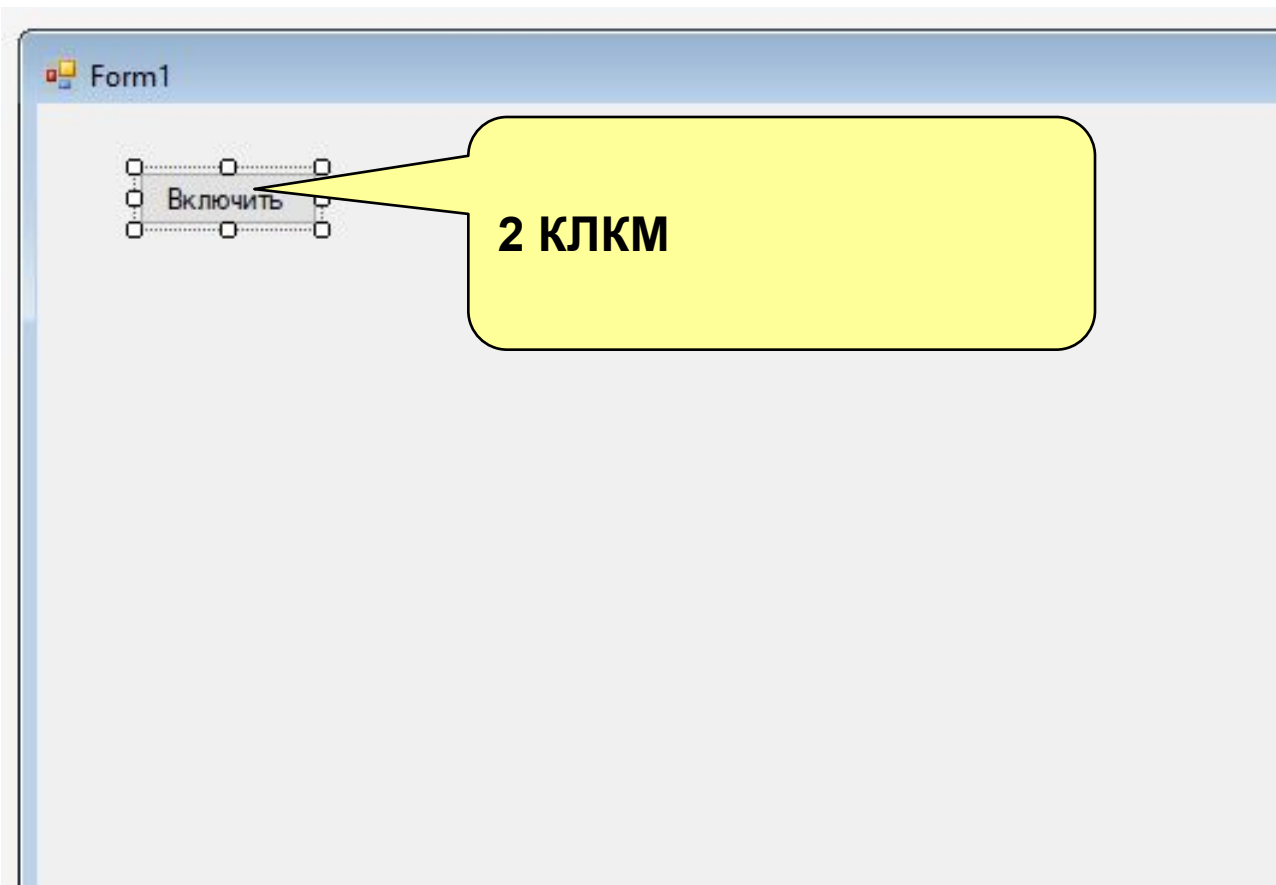
Text

Текст, связанный с элементом управления.

Обзорщик решений

Свойства





```
Form1.cs*  X Form1.cs [Конструктор]*
C# пример 1  пример_1.Form1
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10
11 namespace пример_1
12 {
13     public partial class Form1 : Form
14     {
15         public Form1()
16         {
17             InitializeComponent();
18         }
19
20         private void button1_Click(object sender, EventArgs e)
21         {
22
23         }
24     }
25 }
26
```



Form1.cs\* [Конструктор]\*

C# пример 1

пример\_1.Form1

```
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10
11 namespace пример_1
12 {
13     public partial class Form1 : Form
14     {
15         public Form1()
16         {
17             InitializeComponent();
18         }
19
20         private void button1_Click(object sender, EventArgs e)
```

```
pictureBox1.Image = Image.FromFile("Z:\\Прилепина Е.В\\Основы программирования\\Картинки\\foto.jpg");
```

Вместо одного \ пишем два \\ !!!!!

# Кнопка

Наиболее часто используемым элементом управления является кнопка. Обработывая событие нажатия кнопки, может производить те или иные действия.

## Оформление кнопки

Чтобы управлять внешним отображением кнопки, можно использовать свойство FlatStyle.

Оно может принимать следующие значения:

1. Flat - Кнопка имеет плоский вид
2. PopUp - Кнопка приобретает объемный вид при наведении на нее указателя, в иных случаях она имеет плоский вид
3. Standard - Кнопка имеет объемный вид (используется по умолчанию)
4. System - Вид кнопки зависит от операционной системы

## Изображение на кнопке

Как и для многих элементов управления, для кнопки можно задавать изображение с помощью свойства BackgroundImage. Однако мы можем также управлять размещением текста и изображения на кнопки. Для этого надо использовать свойство TextImageRelation. Оно приобретает следующие значения:

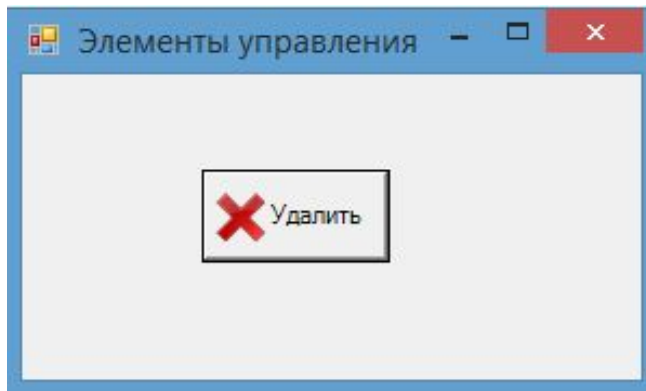
1. Overlay: текст накладывается на изображение
2. ImageAboveText: изображение располагается над текстом
3. TextAboveImage: текст располагается над изображением
4. ImageBeforeText: изображение располагается перед текстом
5. TextBeforeImage: текст располагается перед изображением

Компоненту Button (командная кнопка) можно ставить в соответствие функцию, которая будет выполнена при нажатии на кнопку.

Свойство `TextImageRelation`

установить для него значение `ImageBeforeText`.

В итоге мы получим кнопку, где сразу после изображения идет надпись на кнопке:

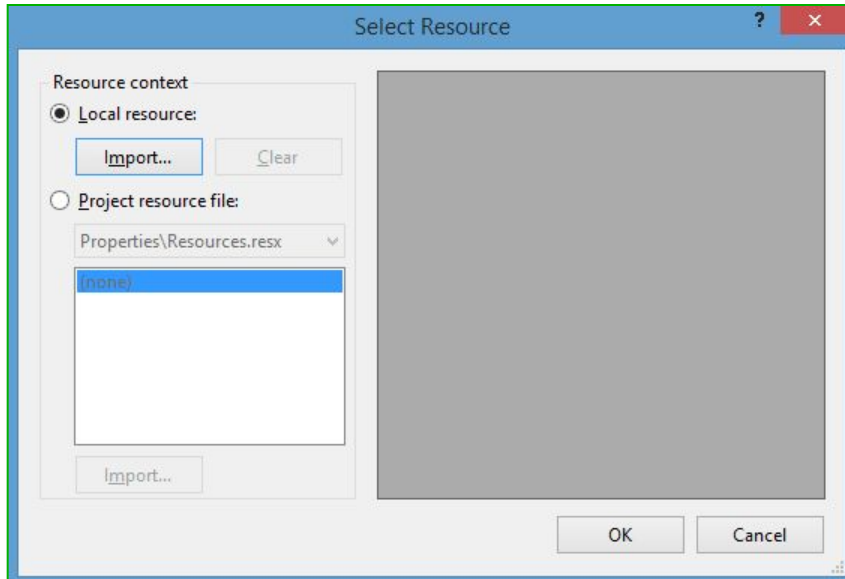


## Задание 3

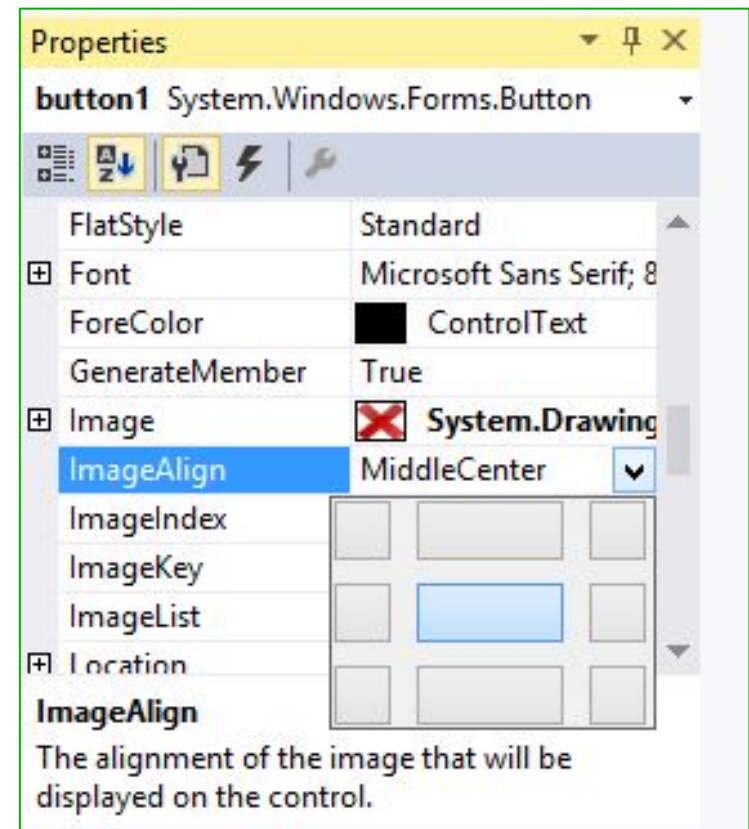
Поместить на форму кнопку . Настроить интерфейс кнопки, добавив картинку и надпись.



Установить для кнопки изображение. Для этого выберем кнопку и в окне Свойств нажмем на поле Image (не путать с BackgroundImage). Откроется диалоговое окно установки изображения:



После выбора изображения установить свойство ImageAlign, которое управляет позиционированием изображения на кнопке:



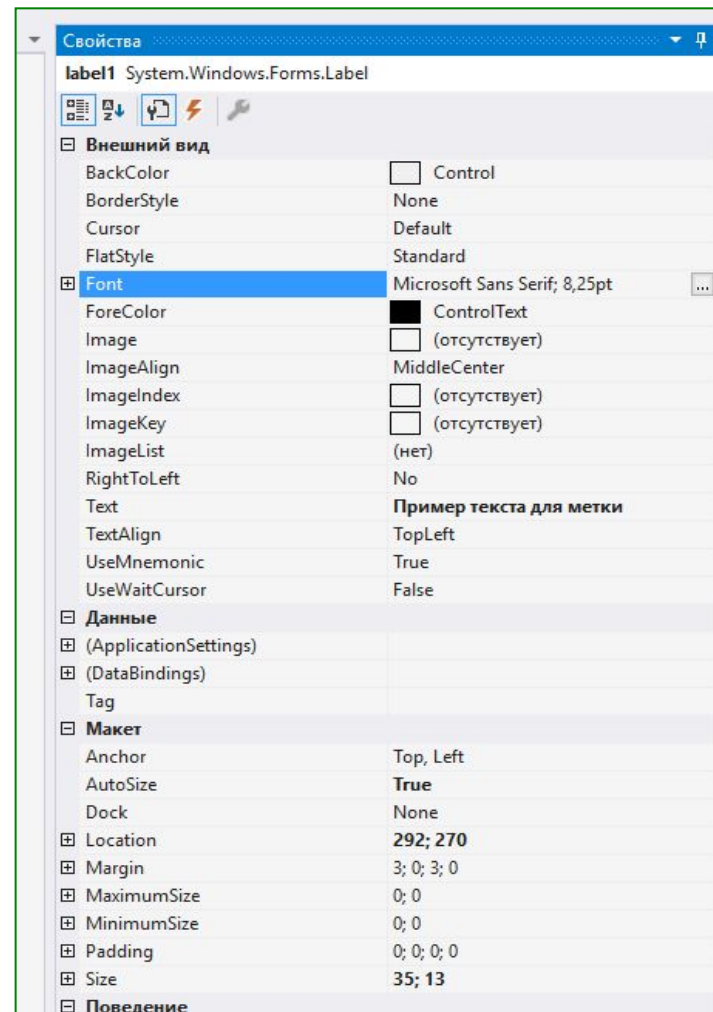
# Метки и ссылки

## Label

Для отображения простого текста на форме, доступного только для чтения, служит элемент Label. Чтобы задать отображаемый текст метки, надо установить свойство Text элемента.

## LinkLabel

Особый тип меток представляют элементы LinkLabel, которые предназначены для вывода ссылок, которые аналогичны ссылкам, размещенным на стандартных веб-станциях.



Во время работы приложения невозможно редактировать содержимое меток, поэтому их рекомендуют использовать для вывода пояснительного текста и результатов.

# Текстовое поле TextBox

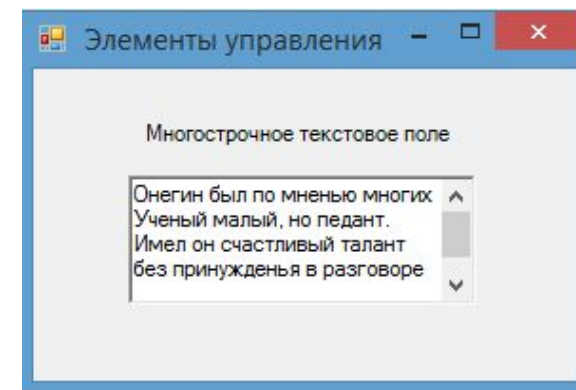
Для ввода и редактирования текста предназначены текстовые поля - элемент TextBox. Текст элемента TextBox можно установить или получить с помощью свойства Text.

По умолчанию создается однострочное текстовое поле. Для отображения больших объемов информации в текстовом поле нужно использовать его свойства Multiline и ScrollBars. При установке для свойства Multiline значения true, все избыточные символы, которые выходят за границы поля, будут переноситься на новую строку.

Кроме того, можно сделать прокрутку текстового поля, установив для его свойства ScrollBars одно из значений:

- **None:** без прокруток (по умолчанию)
- **Horizontal:** создает горизонтальную прокрутку при длине строки, превышающей ширину текстового поля
- **Vertical:** создает вертикальную прокрутку, если строки не помещаются в текстовом поле
- **Both:** создает вертикальную и горизонтальную прокрутку

TextBox (строка редактирования) предназначена для ввода/вывода, тип данных в нем – всегда String и все преобразования должны выполняться программистом.

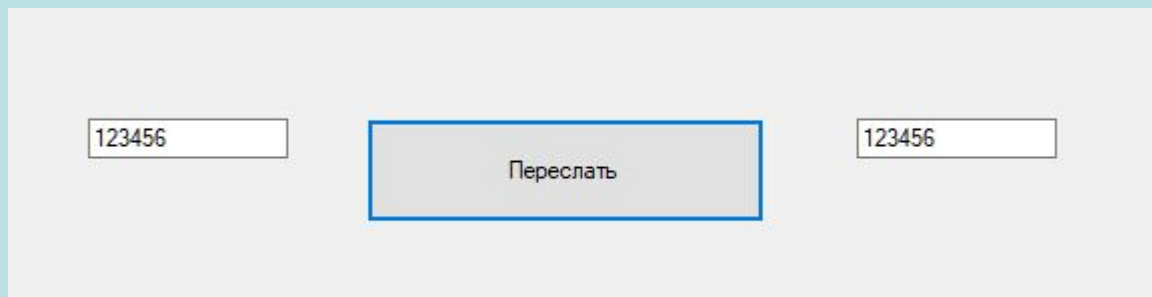




#### Задание 4.

Поместить на форму два поля ввода и кнопку "Переслать".

При нажатии на кнопку текст из первого поля ввода копируется во второе.

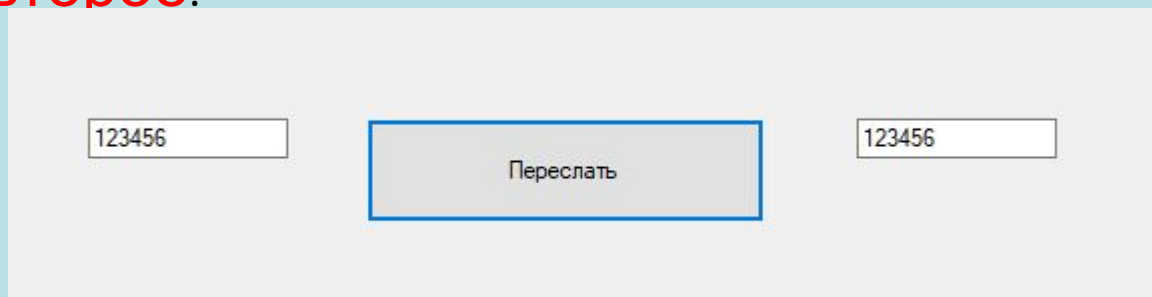


```
ссылка: 1
private void button2_Click(object sender, EventArgs e)
{
    textBox1.Text = textBox2.Text;
}
```

#### Задание 4.

Поместить на форму два поля ввода и кнопку "Переслать".

При нажатии на кнопку текст **из первого** поля ввода копируется **во второе**.

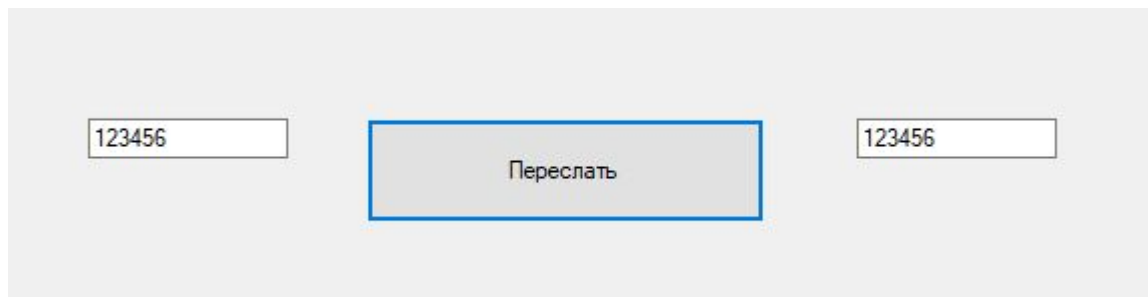


```
сылка: 1
private void button2_Click(object sender, EventArgs e)
{
    textBox1.Text = textBox2.Text;
}
```

#### Задание 4.

Поместить на форму два поля ввода и кнопку "Переслать".

При нажатии на кнопку текст из первого поля ввода копируется во второе.



```
ссылка: 1
private void button2_Click(object sender, EventArgs e)
{
    textBox1.Text = textBox2.Text;
}
```

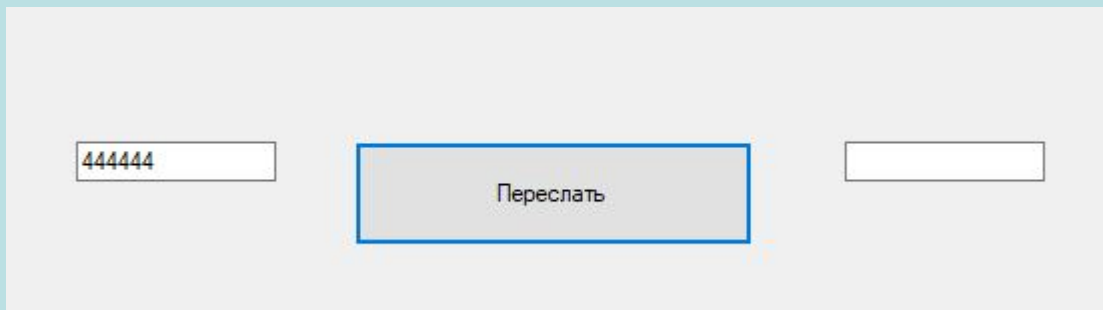
А если написать:

```
textBox2.Text = textBox2.Text + textBox1.Text
```

Что будет?

## Задание 5.

При нажатии на кнопку текст из первого поля ввода переписывается во второе, то есть исчезает в первом поле и появляется во втором.



```
ссылка: 1  
private void button2_Click(object sender, EventArgs e)  
{  
    textBox1.Text = textBox2.Text;  
    textBox2.Text = "";  
}
```

или

```
textBox1.Clear();
```

## Задание 6.

Создадим шуточную программу, представляющую собой диалоговое окно с двумя кнопками.

Назовем его SocOpros. Из окна Toolbox перетаскиваем на форму две кнопки Button и надпись Label и устанавливаем следующие свойства элементов управления и формы:

Объект, свойства	Значение
Text	Социологический опрос
<b>Label1, свойство</b>	<b>Значение</b>
Bold	True
Text	Вы довольны своей зарплатой?
<b>Button1, свойство</b>	<b>Значение</b>
Name	Btnyes
Text	Да
<b>Button2, свойство</b>	<b>Значение</b>
Name	Btnno
Text	Нет

Щелкаем дважды по кнопке "Да".

В обработчике этой кнопки вставляем следующий код:

```
void btnyes_Click(object sender, EventArgs e)
{
    MessageBox.Show("Мы и не сомневались, что Вы так думаете!");
}
```

Выделяем кнопку "Нет".

Открываем окно Properties (События).

Переключаемся в окно событий и дважды щелкаем в поле MouseMove.

В обработчике связываем движение мыши с координатами кнопки и устанавливаем координаты, куда она будет возвращаться, если во время своего движения выйдет за указанную область:

```
private void Btnno_MouseMove(object sender, MouseEventArgs e)
{
    Btnno.Top -= e.Y;
    Btnno.Left += e.X;
    if (Btnno.Top < -10 || Btnno.Top > 100)
        Btnno.Top = 60;
    if (Btnno.Left < -80 || Btnno.Left > 250)
        Btnno.Left = 120;
}
```

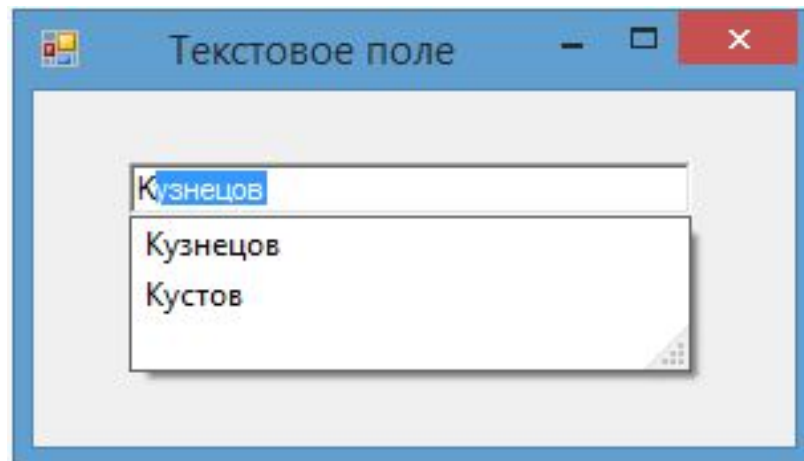


## Автозаполнение текстового поля

Элемент TextBox обладает достаточными возможностями для создания автозаполняемого поля. Для этого нам надо привязать свойство `AutoCompleteCustomSource` элемента `TextBox` к некоторой коллекции, из которой берутся данные для заполнения поля.

```
AutoCompleteStringCollection source = new AutoCompleteStringCollection()  
{  
    "Кузнецов",  
    "Иванов",  
    "Петров",  
    "Кустов"  
};  
textBox1.AutoCompleteCustomSource = source;  
textBox1.AutoCompleteMode = AutoCompleteMode.SuggestAppend;  
textBox1.AutoCompleteSource = AutoCompleteSource.CustomSource;
```

Элементы в список могут добавляться как во время разработки, так и программным способом!



## Перенос по словам

Чтобы текст в элементе `TextBox` переносился по словам, надо установить свойство `WordWrap` равным `true`. То есть если одно слово не помещается на строке, то оно переносится на следующую. Данное свойство будет работать только для многострочных текстовых полей.

## Ввод пароля

Также данный элемент имеет свойства, которые позволяют сделать из него поле для ввода пароля. Так, для этого надо использовать `PasswordChar` и `UseSystemPasswordChar`.

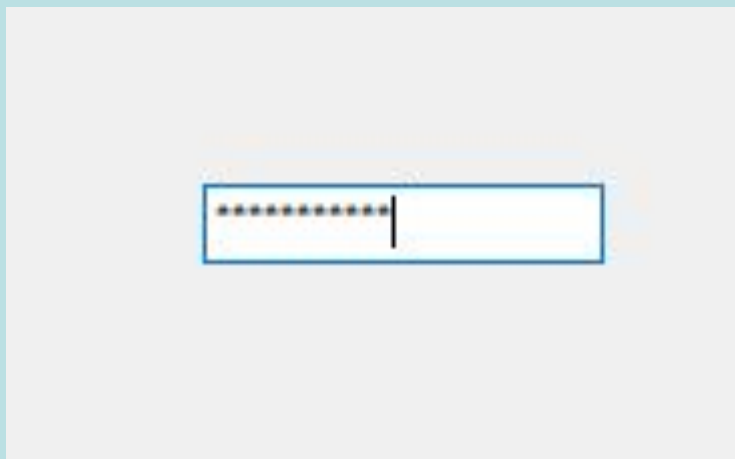
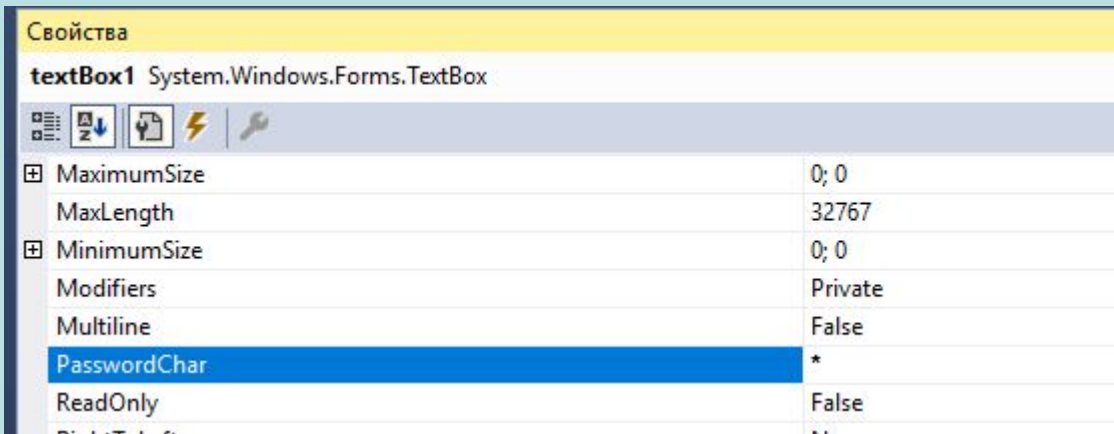
Свойство `PasswordChar` по умолчанию не имеет значение, если мы установим в качестве него какой-нибудь символ, то этот символ будет отображаться при вводе любых символов в текстовое поле.

Свойство `UseSystemPasswordChar` имеет похожее действие. Если мы установим его значение в `true`, то вместо введенных символов в текстовом поле будет отображаться знак пароля, принятый в системе, например, точка.

## Задание 7.

### Свойство PasswordChar

У элемента TextBox установить в Свойство PasswordChar какой-нибудь символ, то этот символ будут отображаться при вводе любых символов в текстовое поле.



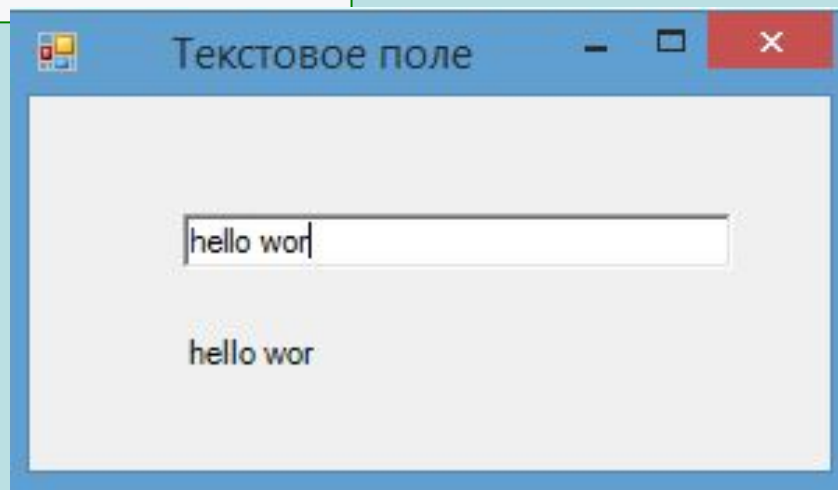
## Задание 8.

Поместить на форму кроме текстового поля метку. Настроить **Событие TextChanged** и сделать так, чтобы при изменении текста в текстовом поле также менялся текст на метке:

```
public Form1()
{
    InitializeComponent();

    textBox1.TextChanged += textBox1_TextChanged;
}

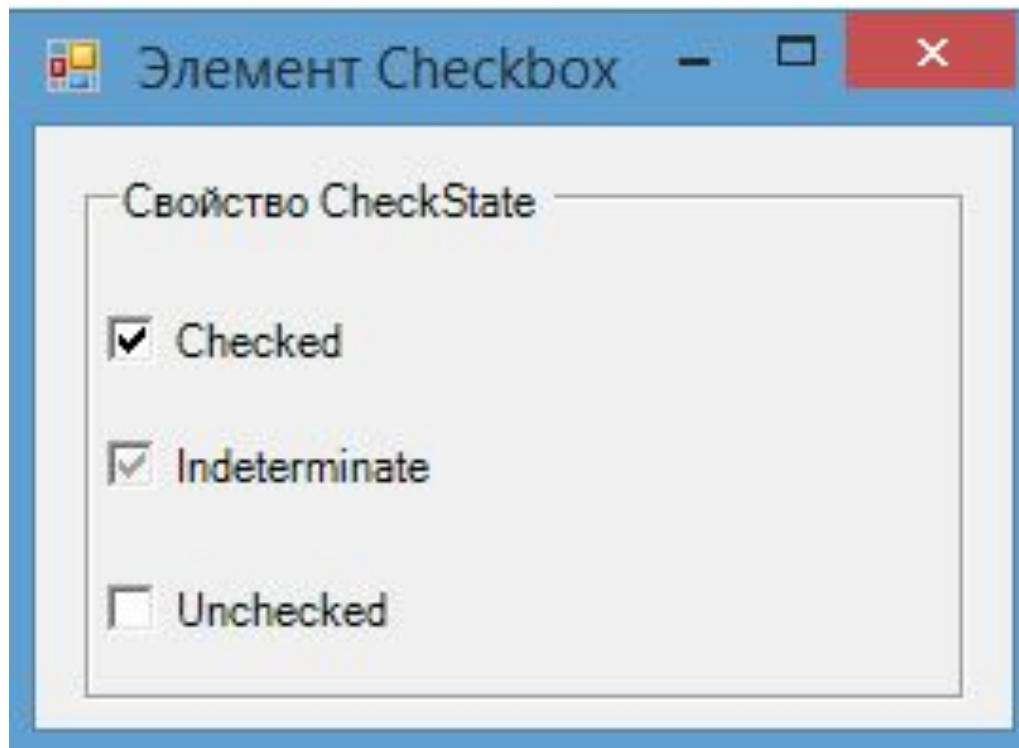
private void textBox1_TextChanged(object sender, EventArgs e)
{
    label1.Text = textBox1.Text;
}
```



## CheckBox

Элемент CheckBox или флажок предназначен для установки одного из двух значений: отмечен или не отмечен. Чтобы отметить флажок, надо установить у его свойства **Checked** значение true.

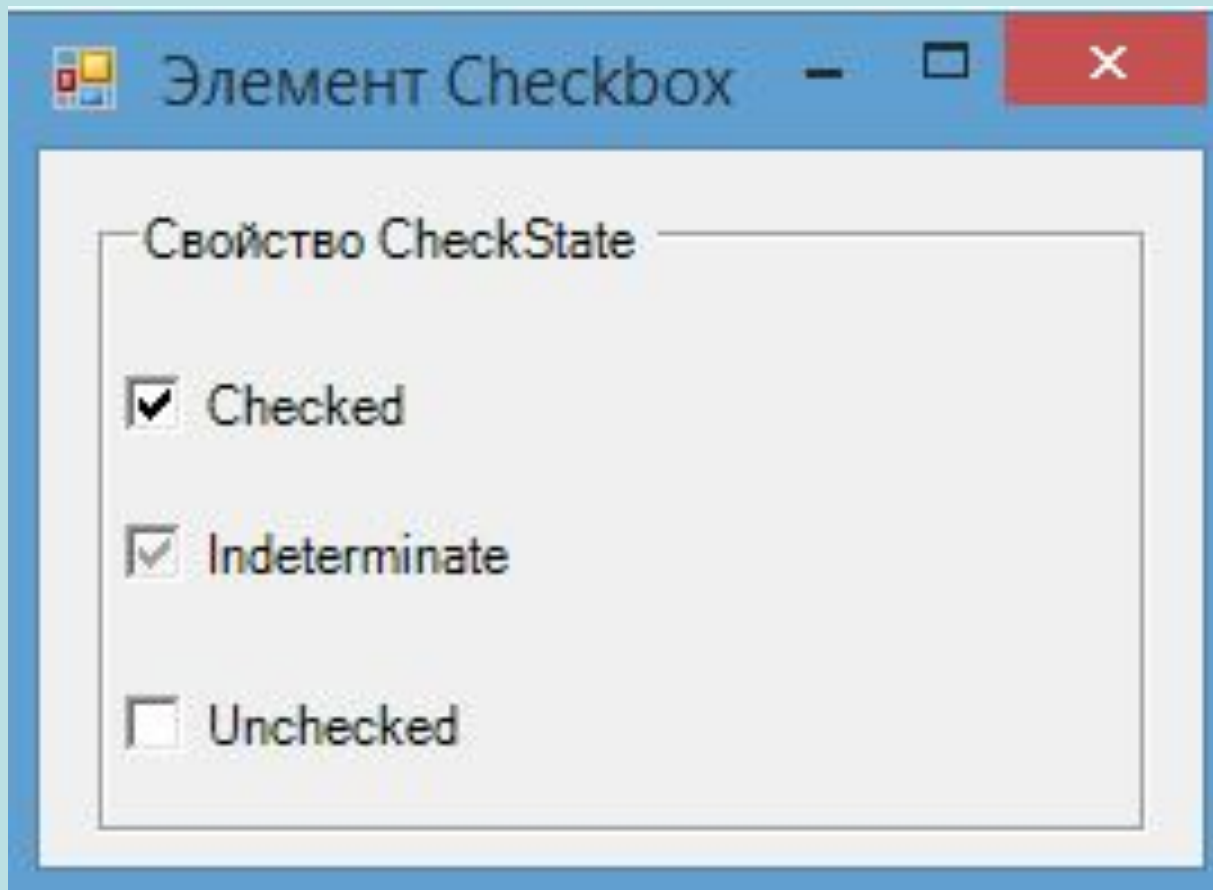
Кроме свойства Checked у элемента CheckBox имеется свойство **CheckState**, которое позволяет задать для флажка одно из трех состояний - Checked (отмечен), Indeterminate (флажок не определен - отмечен, но находится в неактивном состоянии) и Unchecked (не отмечен)



## Задание 9.

Поместить на форму 3 флажка.

Проверить работу со свойствами - **Checked** , **CheckState**



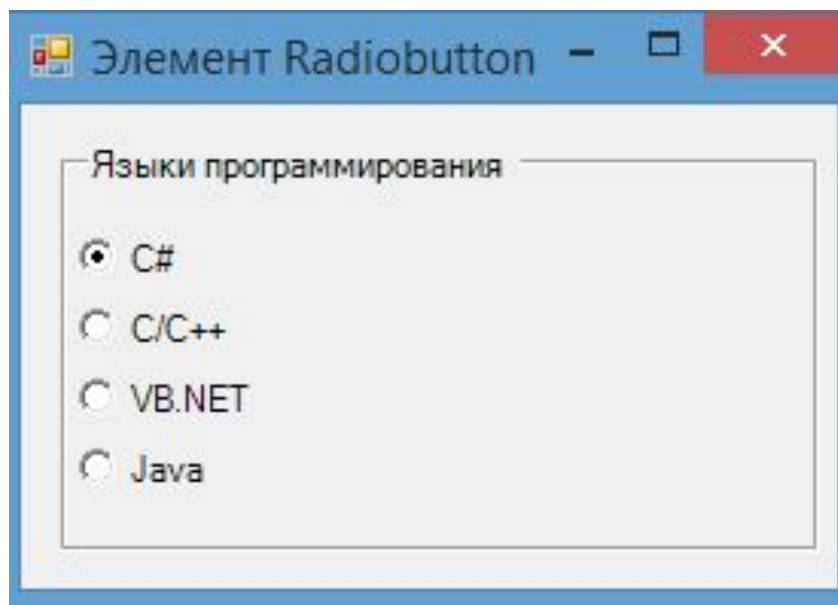


## Radiobutton

На элемент CheckBox похож элемент RadioButton или переключатель. Переключатели располагаются группами, и включение одного переключателя означает отключение всех остальных.

Чтобы установить у переключателя включенное состояние, надо присвоить его свойству Checked значение true.

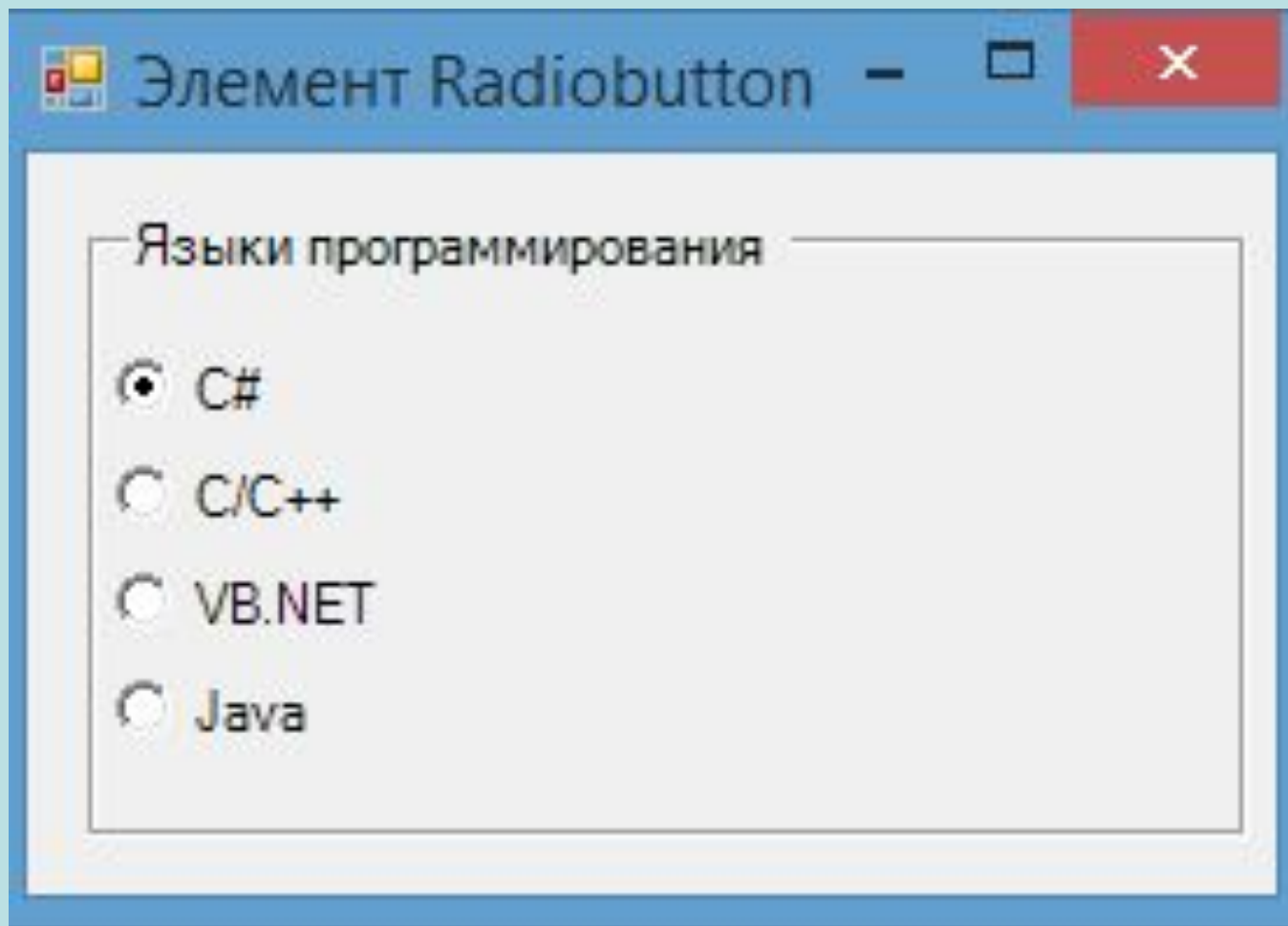
Для создания группы переключателей, из которых можно бы было выбирать, надо поместить несколько переключателей в какой-нибудь контейнер, например, в элементы **GroupBox** или **Panel**. Переключатели, находящиеся в разных контейнерах, будут относиться к разным группам:



## Задание 10.

Поместить на форму элементы **GroupBox** и **Panel**. Поместить по 3 радиокнопки в каждый.

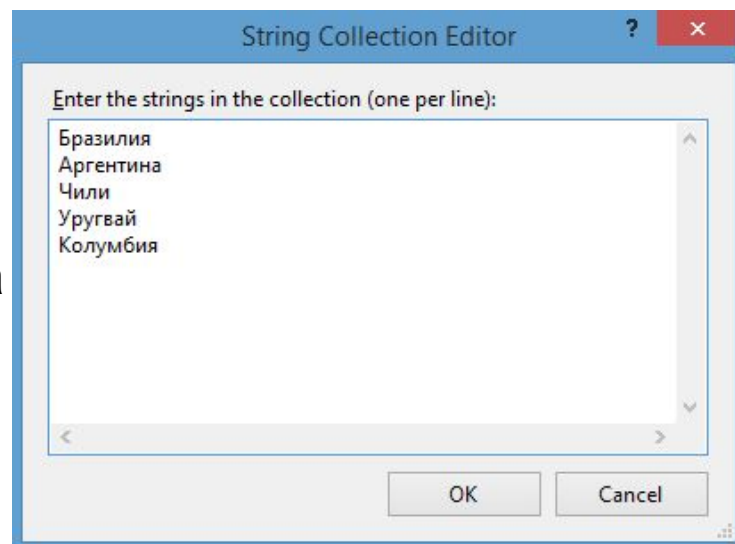
Проверить работу со свойством - **Checked**.



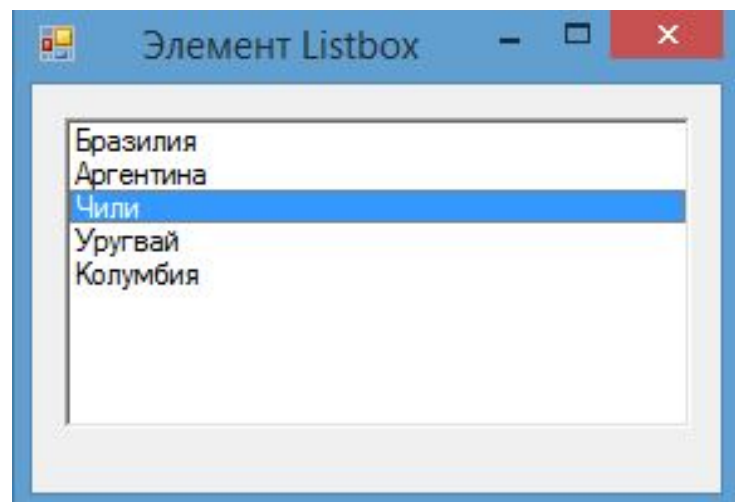
## ListBox

Элемент ListBox представляет собой простой список. Ключевым свойством этого элемента является свойство `Items`, которое как раз и хранит набор всех элементов списка.

Элементы в список могут добавляться как во время разработки, так и программным способом. В Visual Studio в окне Properties (Свойства) для элемента ListBox мы можем найти свойство `Items`. После двойного щелчка на свойство нам отобразится окно для добавления элементов в список:

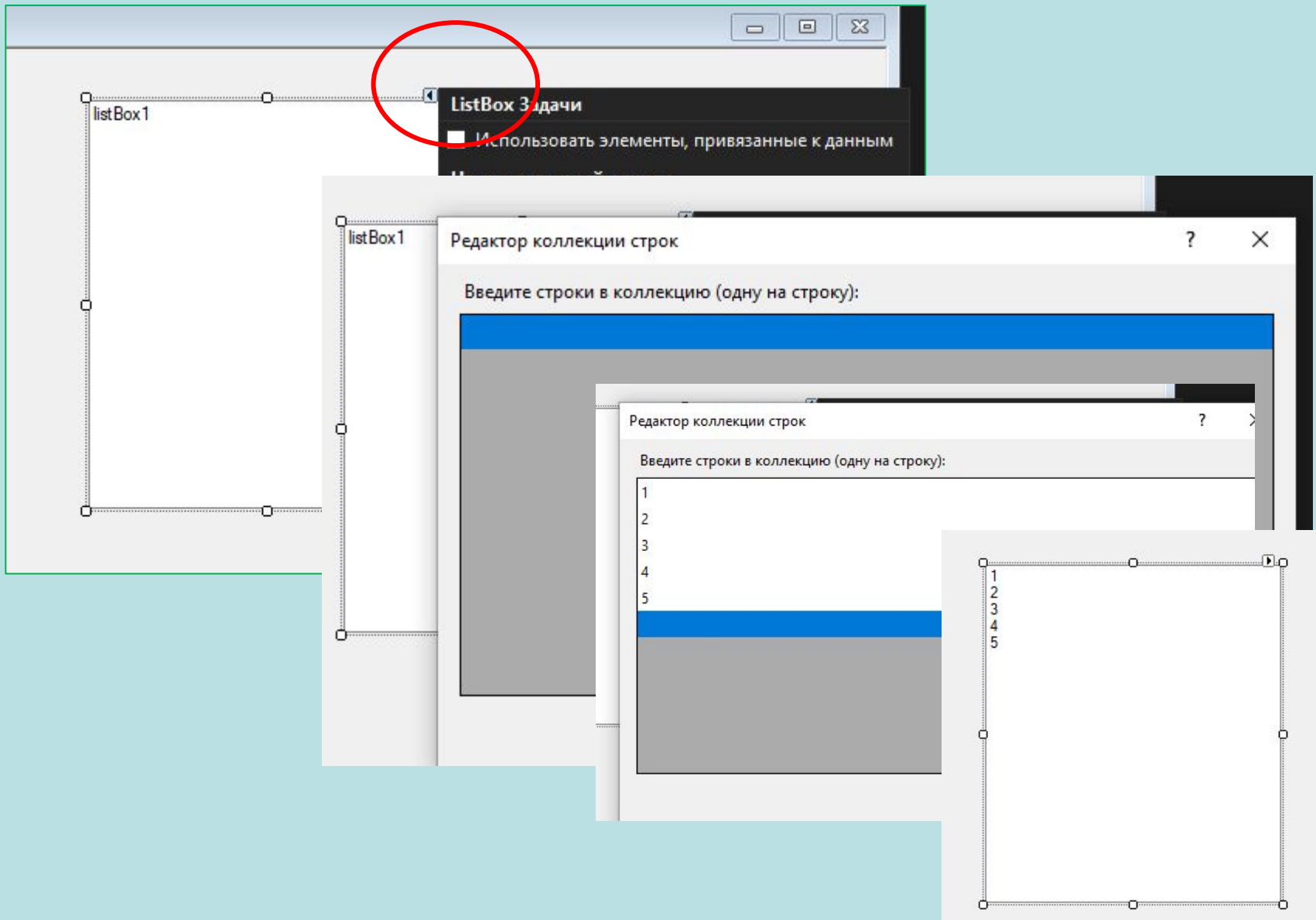


В пустое поле мы вводим по одному элементу списка - по одному на каждой строке. После этого все добавленные нами элементы окажутся в списке, и мы сможем ими управлять:



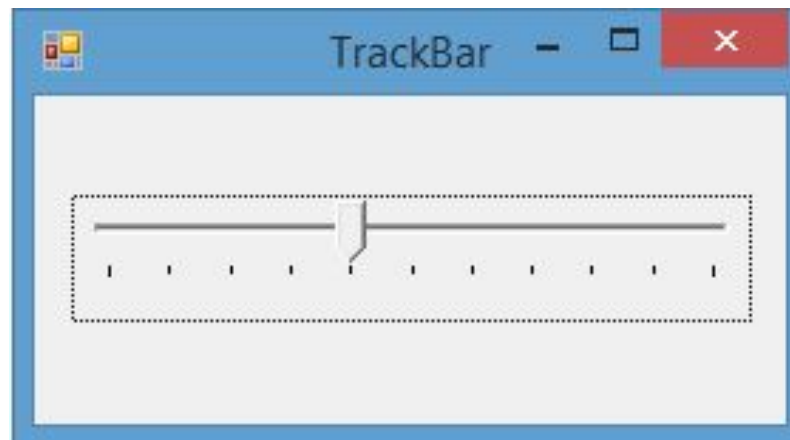
## Задание 11.

Поместить на форму элемент **ListBox**. Добавьте элементы в список.



## TrackBar

TrackBar представляет собой элемент, который с помощью перемещения ползунка позволяет вводить числовые значения.



Некоторые важные свойства TrackBar:

- Orientation: задает ориентацию ползунка - расположение по горизонтали или по вертикали
- TickStyle: задает расположение делений на ползунке
- TickFrequency: задает частоту делений на ползунке
- Minimum: минимальное возможное значение на ползунке (по умолчанию 0)
- Maximum: максимальное возможное значение на ползунке (по умолчанию 10)
- Value: текущее значение ползунка. Должно находиться между Minimum и Maximum

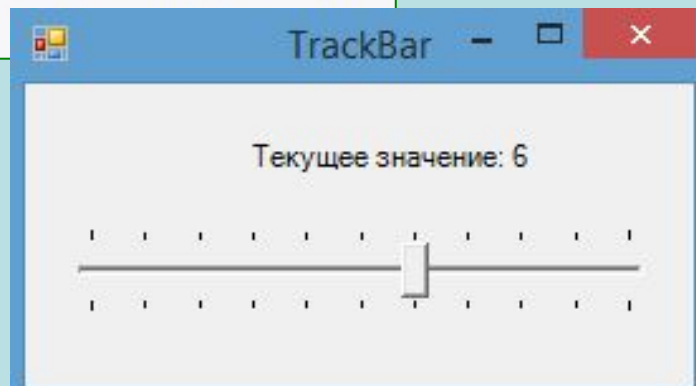
## Задание 12.

Поместите на форму TrackBar. Настройте его событие.

К наиболее важным событиям элемента следует отнести событие Scroll, которое позволяет обработать перемещение ползунка от одного деления к другому. Что может быть полезно, если нам надо, например, устанавливать соответствующую громкость звука в зависимости от значения ползунка, либо какие-нибудь другие настройки.

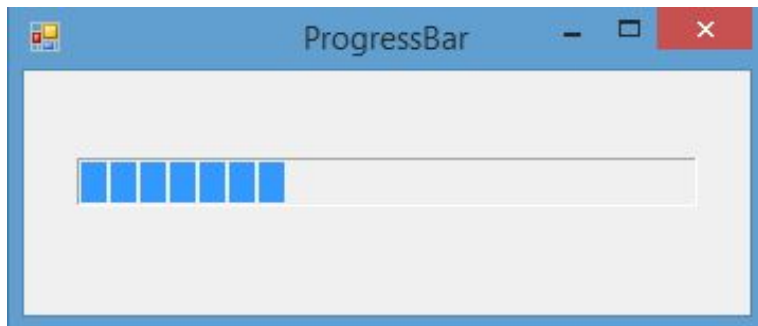
```
public Form1()
{
    InitializeComponent();
    // установка обработчика события Scroll
    trackBar1.Scroll+=trackBar1_Scroll;
}

private void trackBar1_Scroll(object sender, EventArgs e)
{
    label1.Text = String.Format("Текущее значение: {0}", trackBar1.Value);
}
```



## ProgressBar

Элемент ProgressBar служит для того, чтобы дать пользователю информацию о ходе выполнения какой-либо задачи.



Наиболее важные свойства ProgressBar:

- `Minimum` : минимальное возможное значение
- `Maximum` : максимальное возможное значение
- `Value` : текущее значение элемента
- `Step` : шаг, на который изменится значение `Value` при вызове метода `PerformStep`

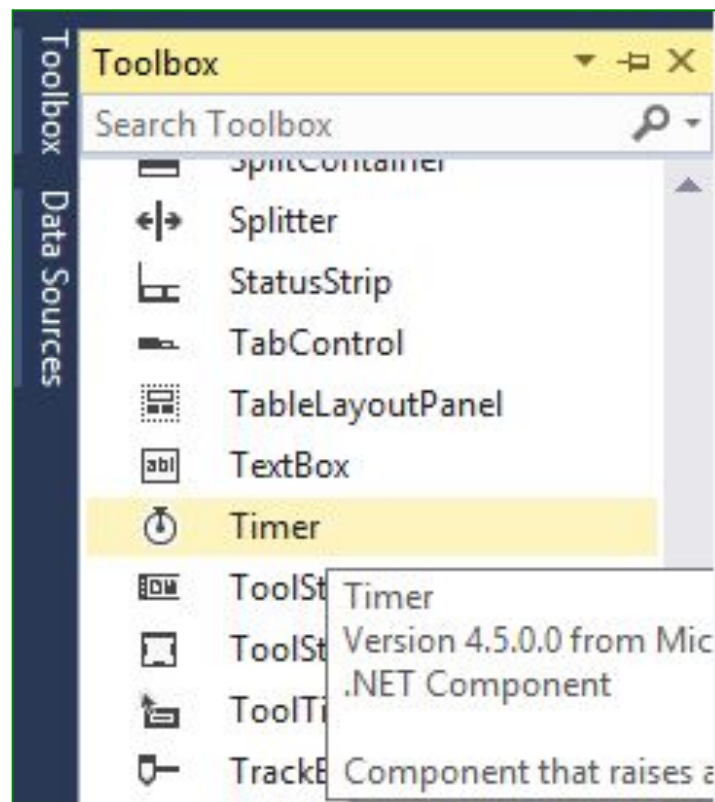


## Timer

Timer является компонентом для запуска действий, повторяющихся через определенный промежуток времени. Хотя он не является визуальным элементом, но его а также можно перетащить с Панели Инструментов на форму:

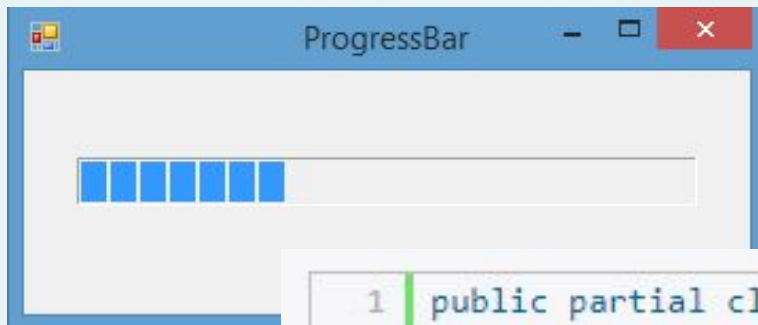
Наиболее важные свойства и методы таймера:

- Свойство Enabled: при значении true указывает, что таймер будет запускаться вместе с запуском формы
- Свойство Interval: указывает интервал в миллисекундах, через который будет срабатывать обработчик события Tick, которое есть у таймера
- Метод Start(): запускает таймер
- Метод Stop(): останавливает таймер



### Задание 13.

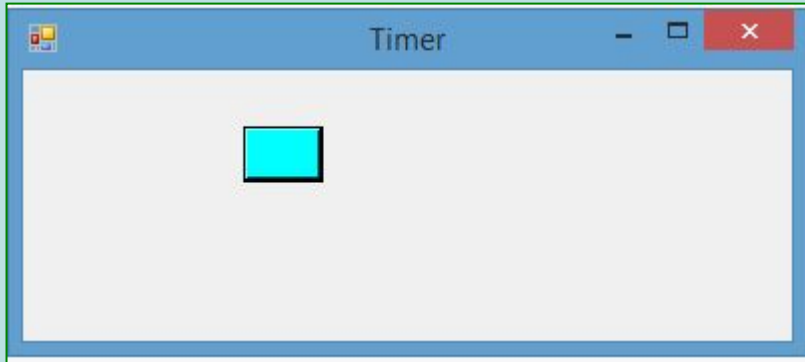
Для имитации работы прогрессбара поместите на форму таймер и в коде формы определите следующий код:



```
1 public partial class Form1 : Form
2 {
3     public Form1()
4     {
5         InitializeComponent();
6
7         timer1.Interval = 500; // 500 миллисекунд
8         timer1.Enabled = true;
9         timer1.Tick += timer1_Tick;
10    }
11    // обработчик события Tick таймера
12    void timer1_Tick(object sender, EventArgs e)
13    {
14        progressBar1.PerformStep();
15    }
16 }
```

## Задание 14 .

Создайте форму, на которую добавьте кнопку и таймер.



Условие:

1. В конструкторе формы устанавливаются начальные значения для таймера, кнопки и формы.
2. Через каждый интервал таймера будет срабатывать обработчик `timer1_Tick`, в котором изменяется положение кнопки по горизонтали с помощью свойства `button1.Left`.
3. А с помощью дополнительной переменной `coef` можно управлять направлением движения.
4. Кроме того, с помощью обработчика нажатия кнопки `button1_Click` можно либо остановить таймер (и вместе с ним движение кнопки), либо опять его запустить.

## Пример кода:

```

1 public partial class Form1 : Form
2 {
3     int koef = 1;
4     public Form1()
5     {
6         InitializeComponent();
7
8         this.Width = 400;
9         button1.Width = 40;
10        button1.Left = 40;
11        button1.Text = "";
12        button1.BackColor = Color.Aqua;
13
14        timer1.Interval = 500; // 500 миллисекунд
15        timer1.Enabled = true;
16        button1.Click += button1_Click;
17        timer1.Tick += timer1_Tick;
18    }

```

```

// обработчик события Tick таймера
void timer1_Tick(object sender, EventArgs e)
{
    if (button1.Left == (this.Width-button1.Width-10))
    {
        koef=-1;
    }
    else if (button1.Left == 0)
    {
        koef = 1;
    }
    button1.Left += 10 *koef;
}

```

```

// обработчик нажатия на кнопку
void button1_Click(object sender, EventArgs e)
{
    if(timer1.Enabled==true)
    {
        timer1.Stop();
    }
    else
    {
        timer1.Start();
    }
}

```