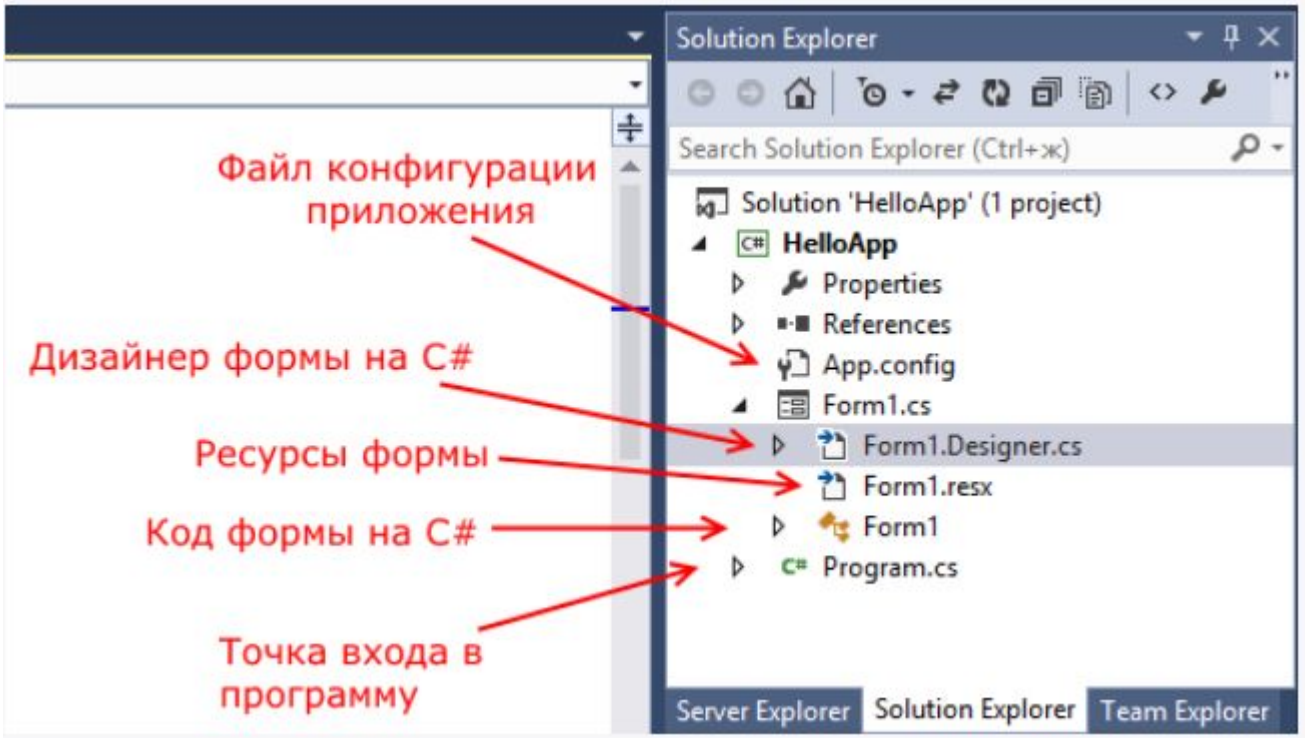
Тема 4.2. Элементы управления

Введение в Windows Forms

- Для создания графических интерфейсов с помощью платформы .NET применяются разные технологии - Window Forms, WPF, приложения для магазина Windows Store (для ОС Windows 8/8.1/10).
- Однако наиболее простой и удобной платформой до сих пор остается Window Forms или формы.

- 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.

- Если запустить приложение, то отобразится одна пустая форма.
- Проект с пустой формой имеет несколько компонентов:

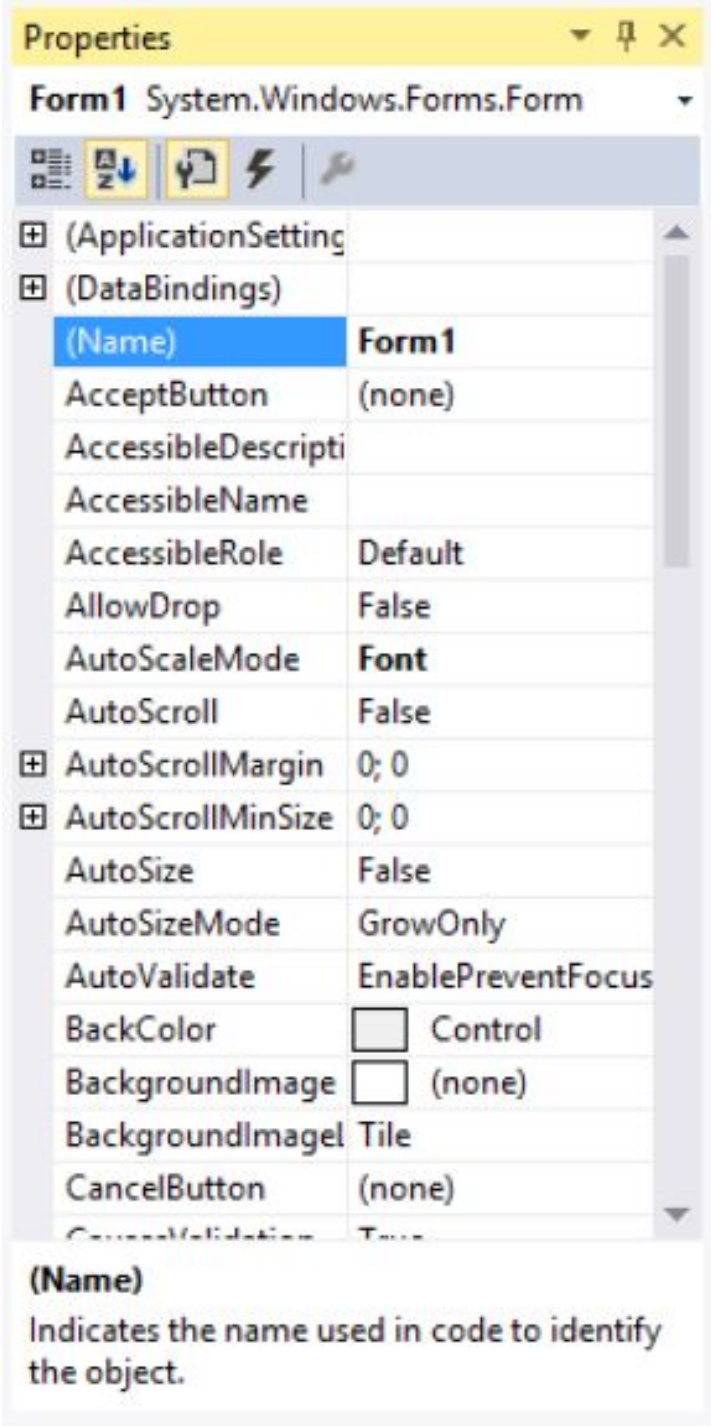


- Стартовой точкой входа в графическое приложение является класс Program, расположенный в файле *Program.cs*:

```
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());
        }
    }
}
```

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



Основные свойства:

- **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.

Основные свойства:

- **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

Основные свойства:

- **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

Основные свойства:

- **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

Основные свойства:

- **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

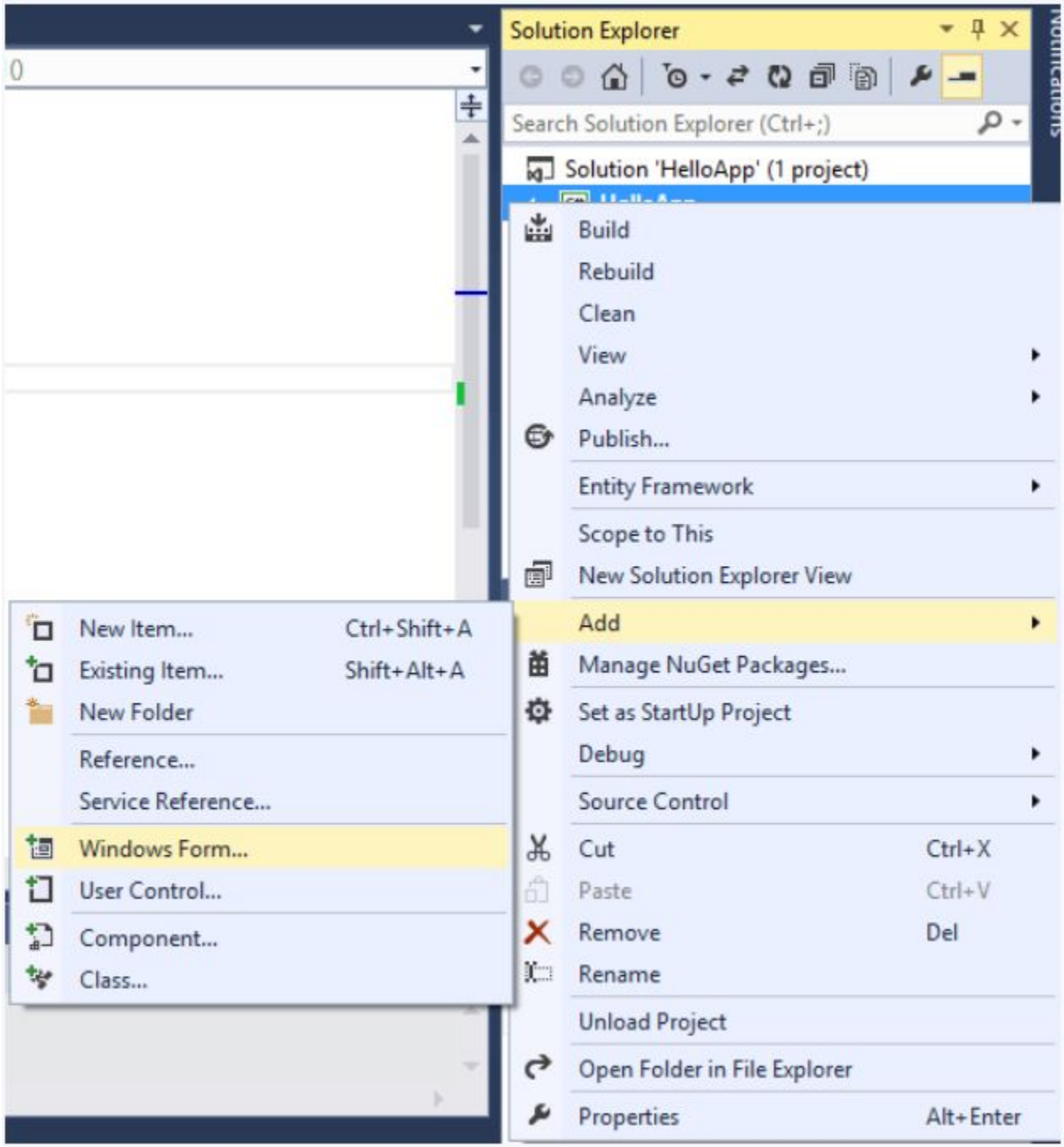
Основные свойства:

- **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

Основные свойства:

- **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

- Чтобы добавить еще одну форму в проект, нажмем на имя проекта в окне Solution Explorer (Обозреватель решений) правой кнопкой мыши и выберем Add(Добавить) ->Windows Form...

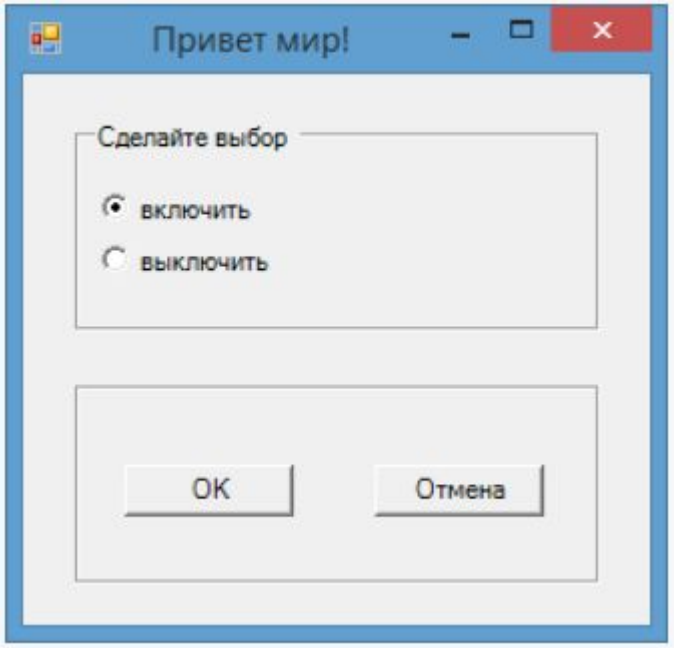


Элемент GroupBox

- GroupBox представляет собой специальный контейнер, который ограничен от остальной формы границей.
- Он имеет заголовок, который устанавливается через свойство Text.
- Чтобы сделать GroupBox без заголовка, в качестве значения свойства Text просто устанавливается пустая строка.

Элемент GroupBox

- Нередко этот элемент используется для группирования переключателей - элементов RadioButton, так как позволяет разграничить их группы.

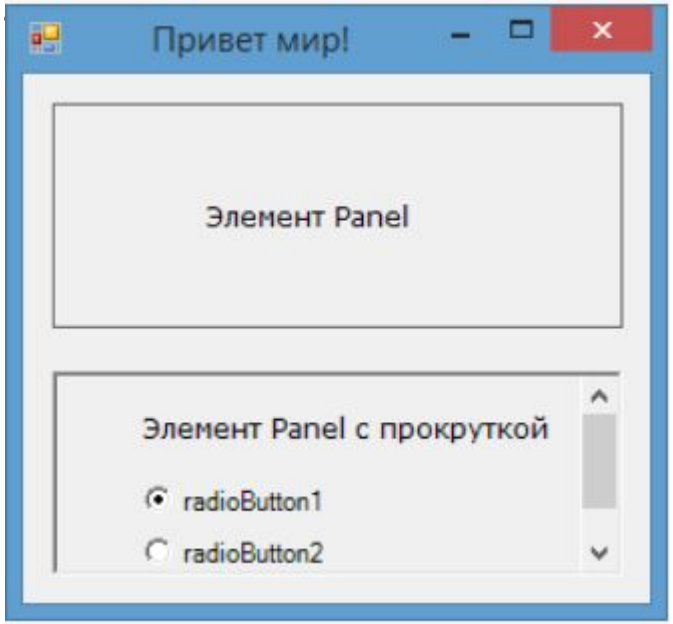


Элемент Panel

- Элемент Panel представляет панель и также, как и GroupBox, объединяет элементы в группы.
- Она может визуально сливаться с остальной формой, если она имеет то же значение цвета фона в свойстве BackColor, что и форма.
- Чтобы ее выделить можно кроме цвета указать для элемента границы с помощью свойства BorderStyle, которое по умолчанию имеет значение None, то есть отсутствие границ.

Элемент Panel

- Также если панель имеет много элементов, которые выходят за ее границы, мы можем сделать прокручиваемую панель, установив ее свойство Au



Элемент FlowLayoutPanel

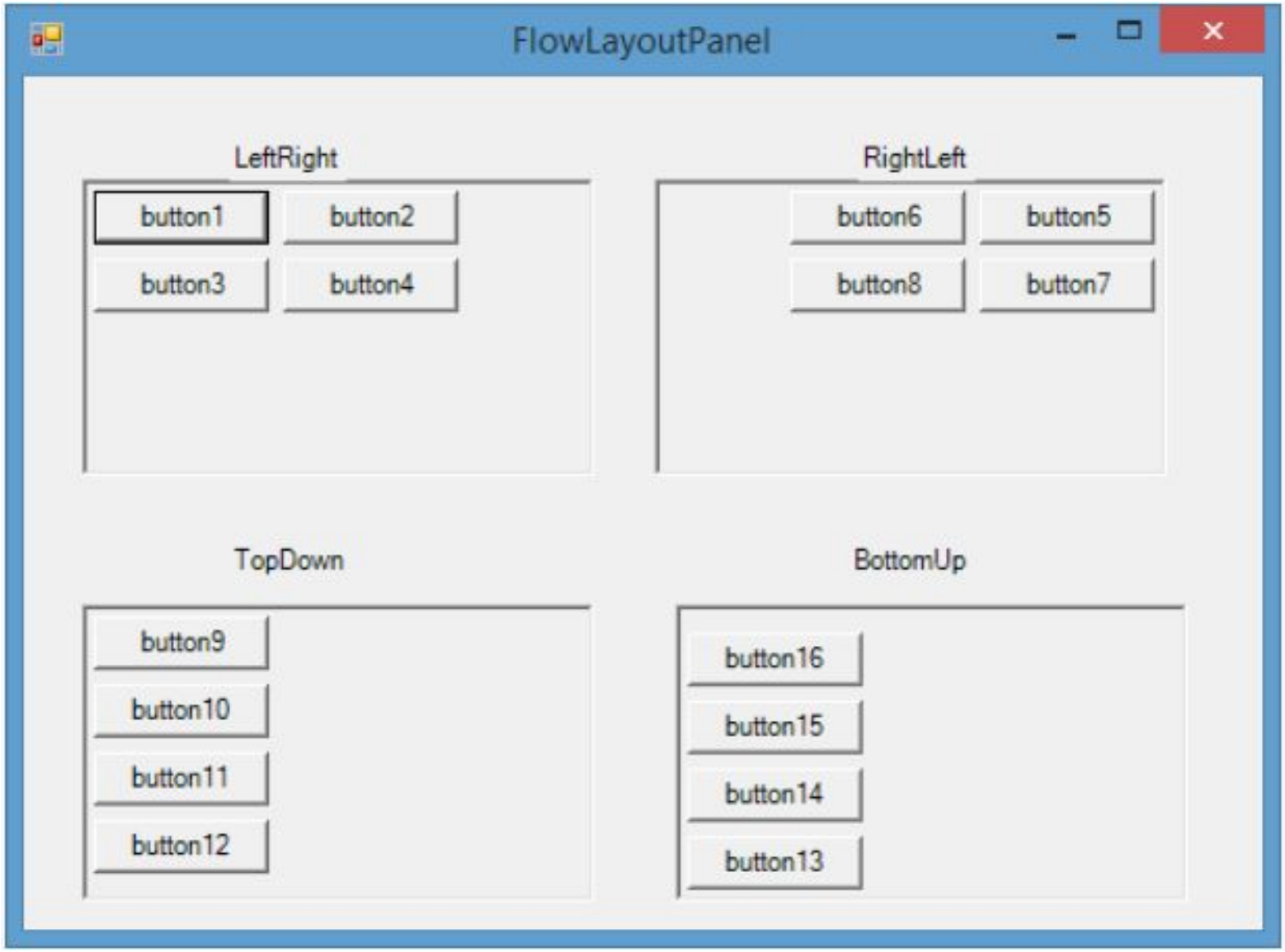
- Элемент FlowLayoutPanel является унаследован от класса Panel, и поэтому наследует все его свойства.
- Однако при этом добавляя дополнительную функциональность.
- Так, этот элемент позволяет изменять позиционирование и компоновку дочерних элементов при изменении размеров формы во время выполнения программы.

Элемент FlowLayoutPanel

- Свойство элемента **FlowDirection** позволяет задать направление, в котором направлены дочерние элементы.
- По умолчанию имеет значение LeftToRight - то есть элементы будут располагаться начиная от левого верхнего края.
- Следующие элементы будут идти вправо.

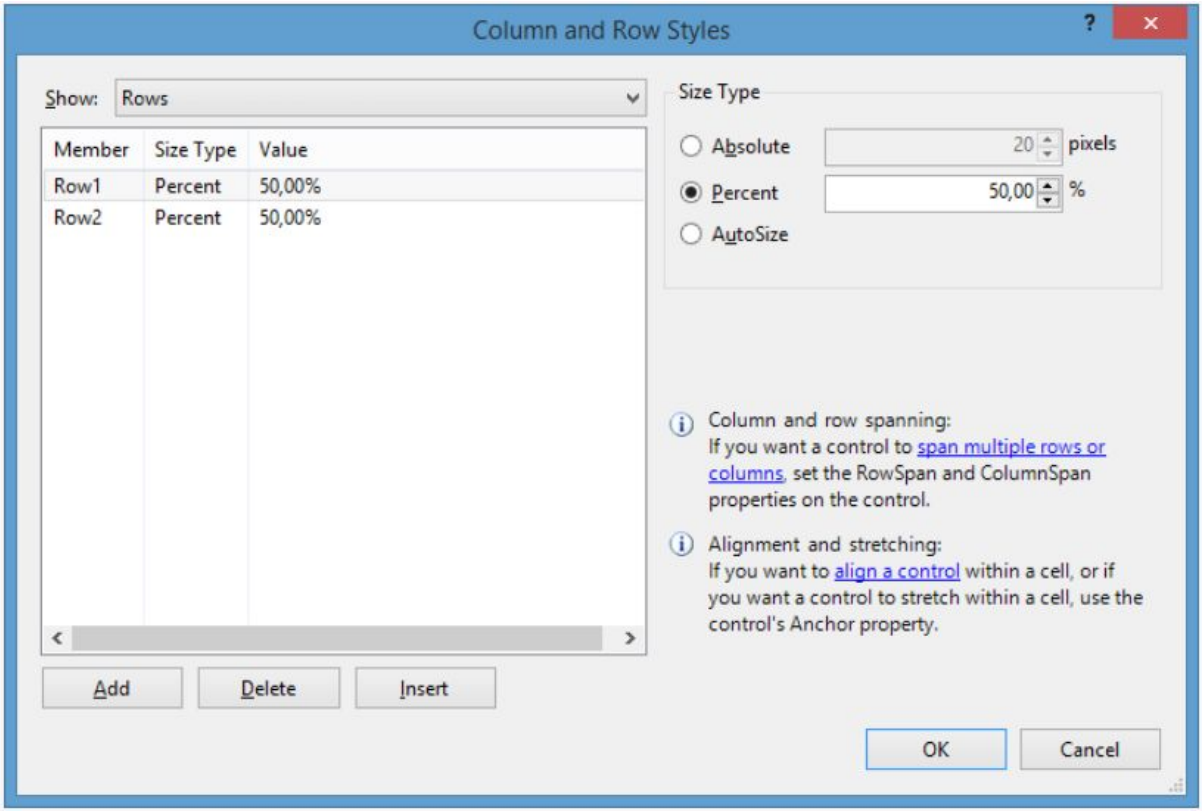
Элемент FlowLayoutPanel

- Это свойство также может принимать следующие значения:
  - **RightToLeft** - элементы располагаются от правого верхнего угла в левую сторону
  - **TopDown** - элементы располагаются от левого верхнего угла и идут вниз
  - **BottomUp** - элементы располагаются от левого нижнего угла и идут вверх


FlowLayoutPanel
LeftRight
button1
button2
button3
button4
RightLeft
button6
button5
button8
button7
TopDown
button9
button10
button11
button12
BottomUp
button16
button15
button14
button13

- Элемент TableLayoutPanel также переопределяет панель и располагает дочерние элементы управления в виде таблицы, где для каждого элемента имеется своя ячейка.
- Если необходимо поместить в ячейку более одного элемента, то в эту ячейку добавляется другой компонент TableLayoutPanel, в который затем вкладываются другие элементы.

- Чтобы установить нужное число строки столбцов таблицы, мы можем использовать свойства Rows и Columns соответственно.
- Выбрав один из этих пунктов в окне Properties (Свойства), отобразится следующее окно для настройки столбцов и строк



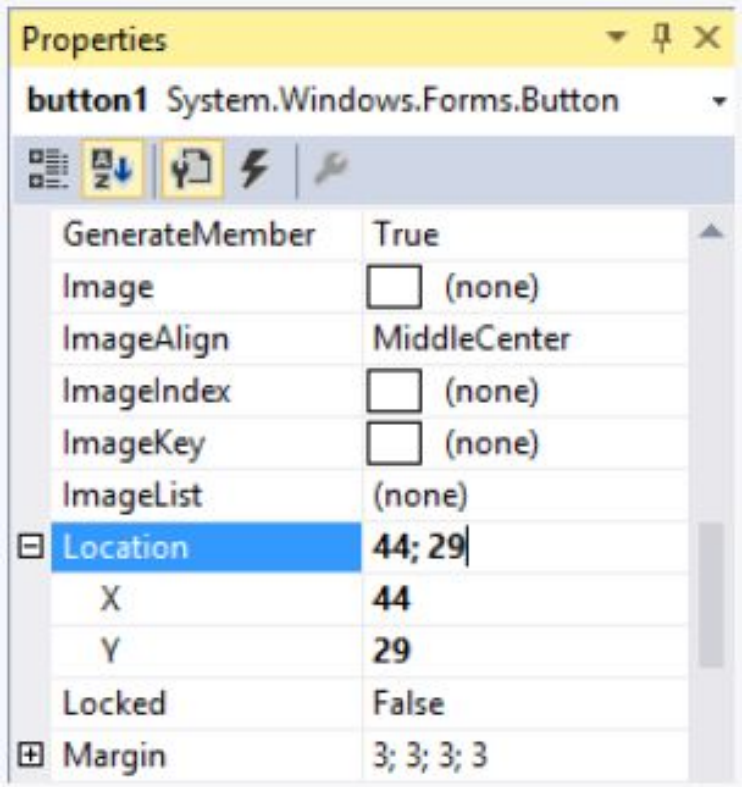
- В поле Size Type можно указать размер столбцов / строк. Нам доступны три возможные варианта:
  - Absolute: задается абсолютный размер для строк или столбцов в пикселях
  - Percent: задается относительный размер в процентах. Если нам надо создать резиновый дизайн формы, чтобы ее строки и столбцы, а также элементы управления в ячейках таблицы автоматически масштабировались при изменении размеров формы, то нам нужно использовать именно эту опцию
  - AutoSize: высота строк и ширина столбцов задается автоматически в зависимости от размера самой большой в строке или столбце ячейки

- Также мы можем комбинировать эти значения, например, один столбец может быть фиксированным с абсолютной шириной, а остальные столбцы могут иметь ширину в процентах.

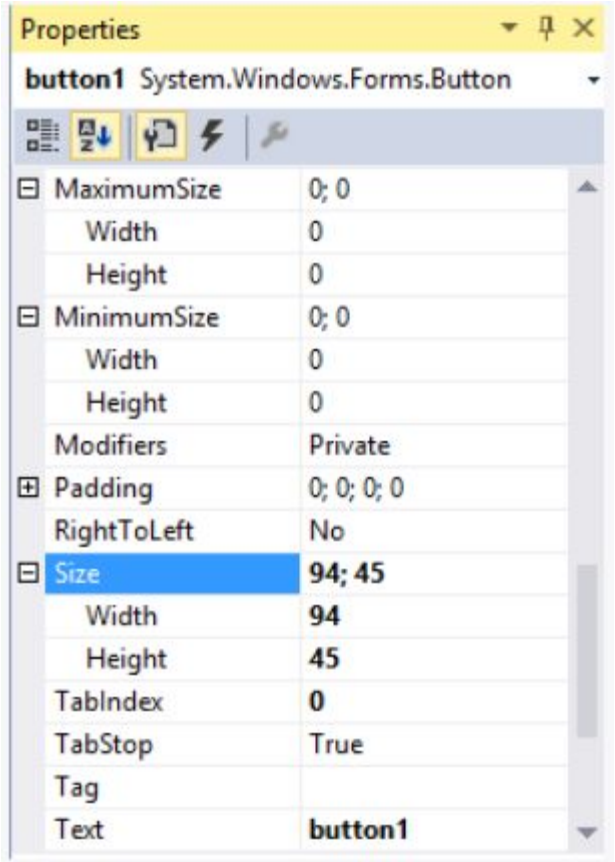
- В этом диалоговом окне мы также можем добавить или удалить строки и столбцы.
- В тоже время графический дизайнер в Visual Studio не всегда сразу отображает изменения в таблице - добавление или удаление строк и столбцов, изменение их размеров, поэтому, если изменений на форме никаких не происходит, надо ее закрыть и потом открыть заново в графическом дизайнере.

- Для каждого элемента управления мы можем определить свойство Location, которое задает координаты верхнего левого угла элемента относительно контейнера.
- При переносе элемента с панели инструментов на форму это свойство устанавливается автоматически.

- В окне Свойств можно вручную поправить координаты положения элемента:

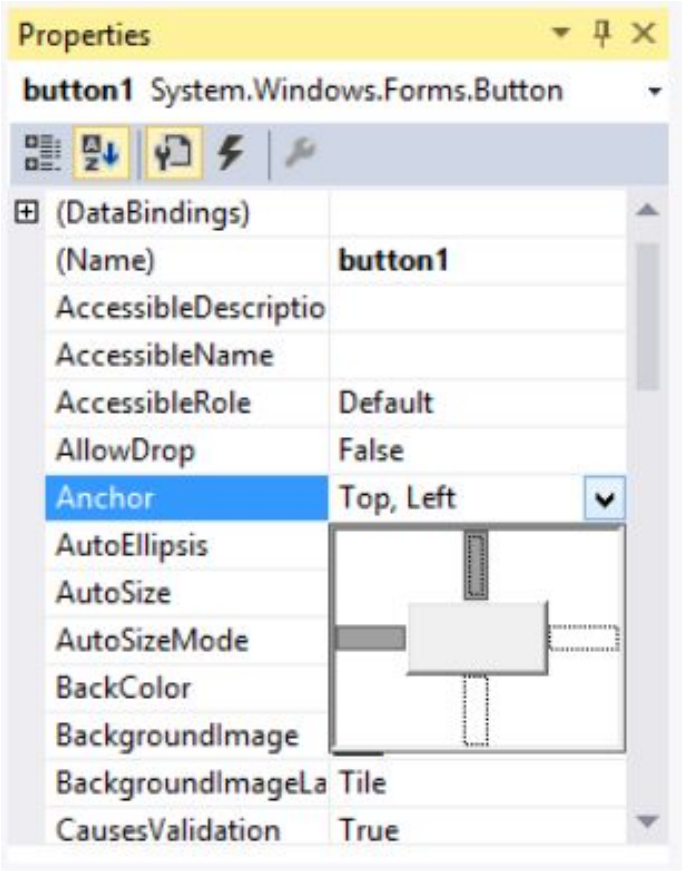


- С помощью свойства **Size** можно задать размеры элемента:



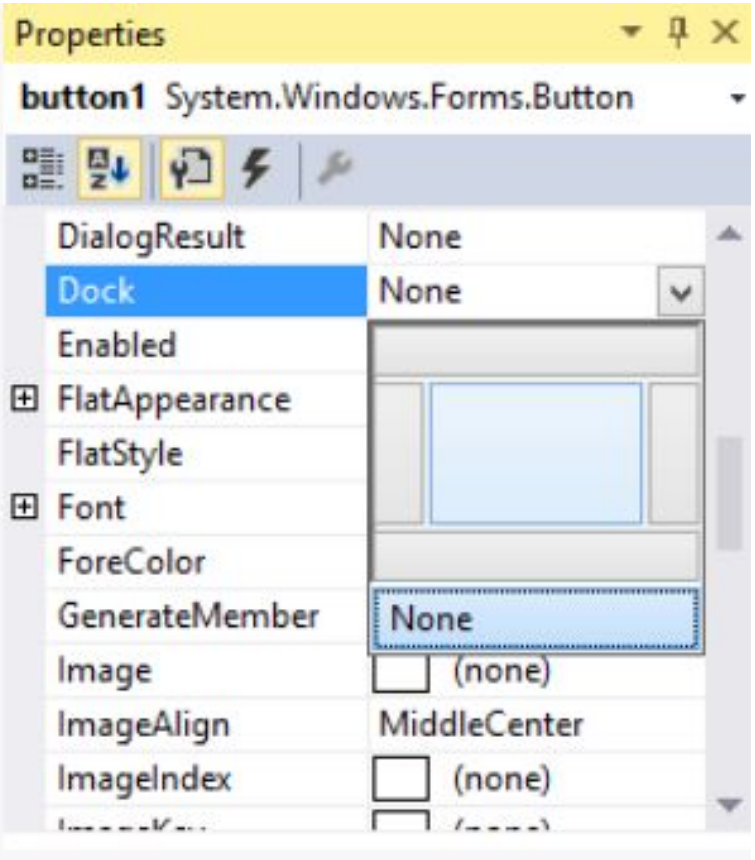
- Дополнительные возможности по позиционировании элемента позволяет определить свойство Anchor.
- Это свойство определяет расстояние между одной из сторон элемента и стороной контейнера.
- И если при работе с контейнером мы будем его растягивать, то вместе с ним будет растягиваться и вложенный элемент.

- По умолчанию у каждого добавляемого элемента это свойство равно Top, Left:



- Можно задать четыре возможных значения для этого свойства или их комбинацию:
  - Top
  - Bottom
  - Left
  - Right

- Свойство Dock позволяет прикрепить элемент к определенной стороне контейнера.
- По умолчанию оно имеет значение None, но также позволяет задать еще пять значений:

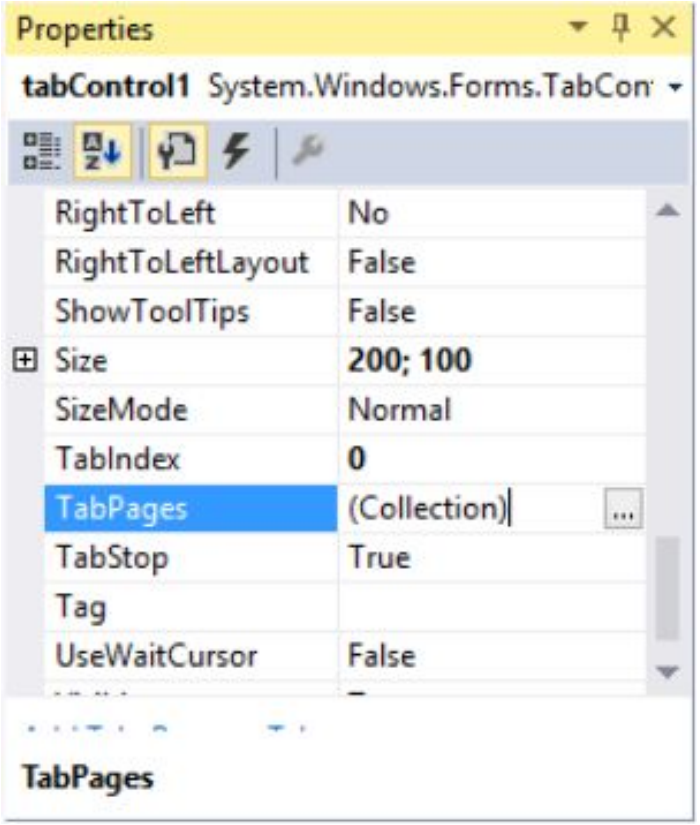


- Top: элемент прижимается к верхней границе контейнера
- Bottom: элемент прижимается к нижней границе контейнера
- Left: элемент прижимается к левой стороне контейнера
- Right: элемент прикрепляется к правой стороне контейнера
- Fill: элемент заполняет все пространство контейнера

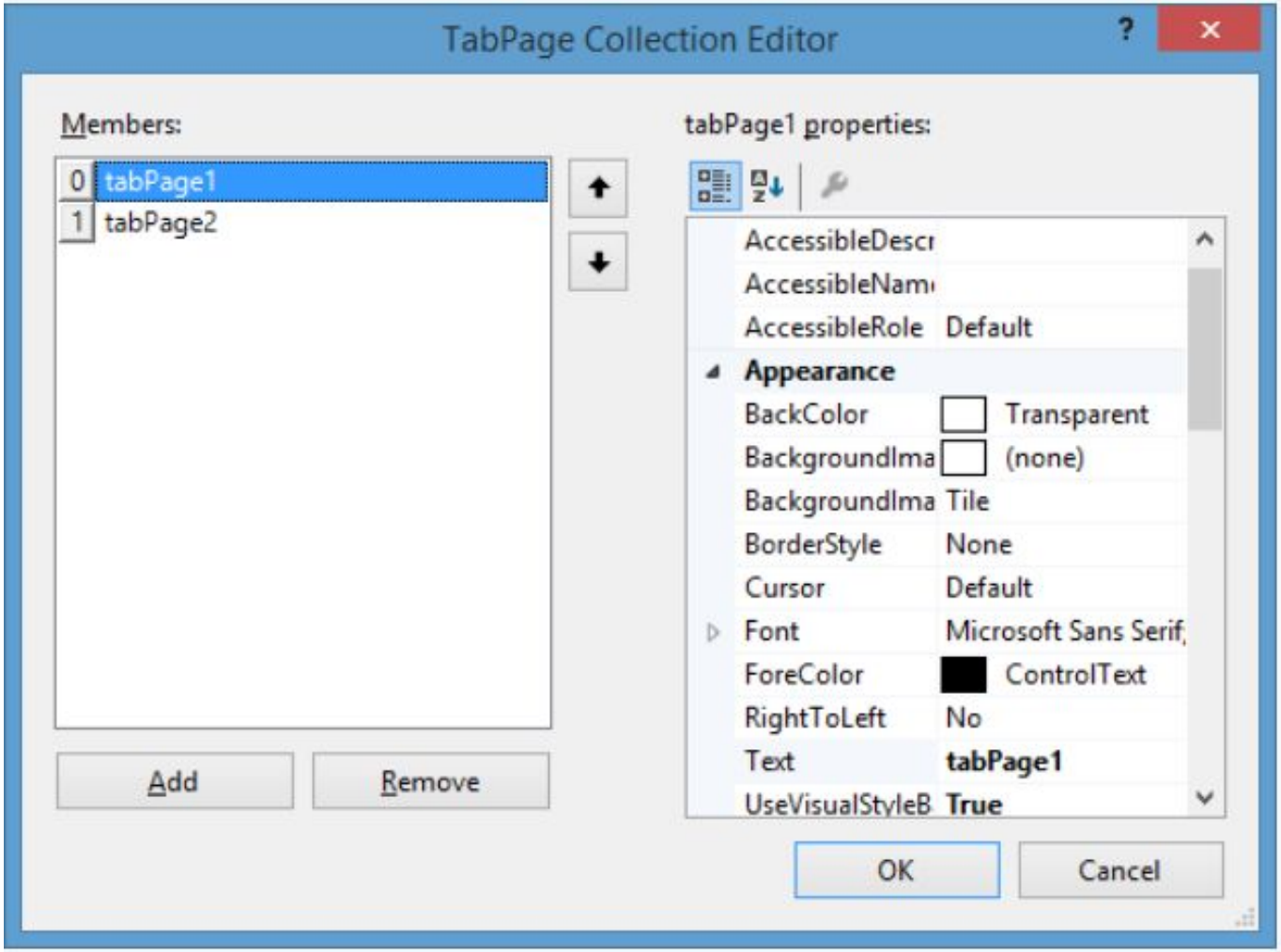
- Элемент TabControl позволяет создать элемент управления с несколькими вкладками.
- И каждая вкладка будет хранить некоторый набор других элементов управления, как кнопки, текстовые поля и др.
- Каждая вкладка представлена классом **TabPage**.

- Чтобы настроить вкладки элемента TabControl используем свойство **TabPages**.
- При переносе элемента TabControl с панели инструментов на форму по умолчанию создаются две вкладки - tabPage1 и tabPage2.

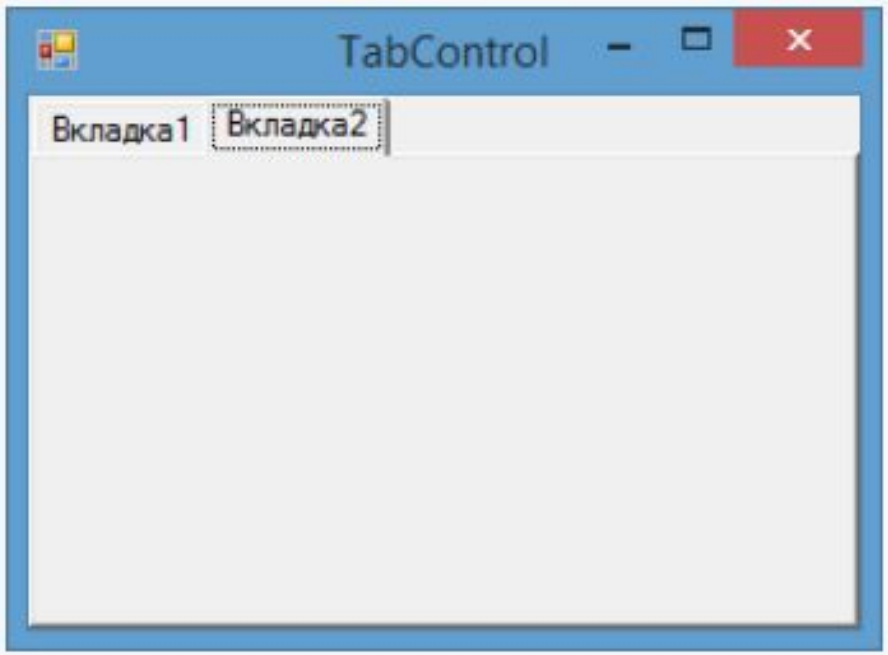
- Изменим их отображение с помощью свойства TabPages:



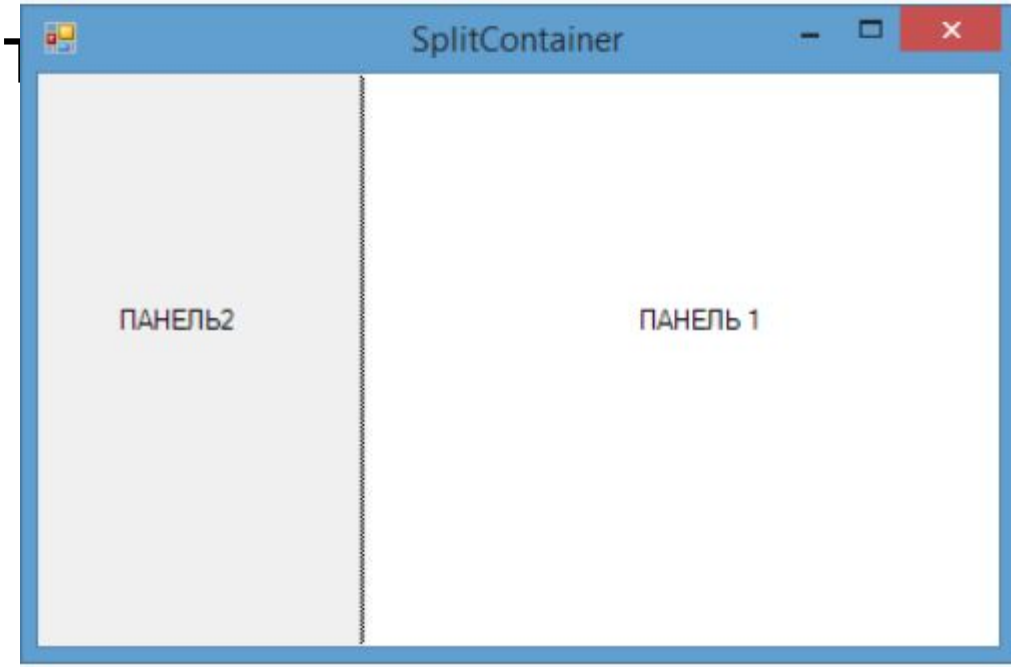
Окно редактирования/добавления и удаления вкладок



- Каждая вкладка представляет своего рода панель, на которую мы можем добавить другие элементы управления, а также заголовок, с помощью которого мы можем переключаться по вкладкам.
- Текст заголовка задается с помощью свойства Text
-



- Элемент SplitContainer позволяет создавать две разделенные сплитером панели.
- Изменяя положение сплитера, можно изменит



- 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

Свойства

- **Anchor**: Определяет, как элемент будет растягиваться
- **BackColor**: Определяет фоновый цвет элемента
- **BackgroundImage**: Определяет фоновое изображение элемента
- **ContextMenu**: Контекстное меню, которое открывается при нажатии на элемент правой кнопкой мыши. Задается с помощью элемента ContextMenu

Свойства

- **Cursor**: Представляет, как будет отображаться курсор мыши при наведении на элемент
- **Dock**: Задает расположение элемента на форме
- **Enabled**: Определяет, будет ли доступен элемент для использования. Если это свойство имеет значение False, то элемент блокируется.
- **Font**: Устанавливает шрифт текста для элемента
- **ForeColor**: Определяет цвет шрифта

Свойства

- **Location**: Определяет координаты верхнего левого угла элемента управления
- **Name**: Имя элемента управления
- **Size**: Определяет размер элемента
- **Width**: ширина элемента
- **Height**: высота элемента
- **TabIndex**: Определяет порядок обхода элемента по нажатию на клавишу Tab
- **Tag**: Позволяет сохранять значение, ассоциированное с этим элементом управления

- Наиболее часто используемым элементом управления является кнопка.
- Обрабатывая событие нажатия кнопки, мы может производить те или иные действия.
- При нажатии на кнопку на форме в редакторе Visual Studio мы по умолчанию попадаем в код обработчика события Click, который будет выполняться при нажатии

- Чтобы управлять внешним отображением кнопки, можно использовать свойство **FlatStyle**.
- Оно может принимать следующие значения:
  - **Flat** - Кнопка имеет плоский вид
  - **Popup** - Кнопка приобретает объемный вид при наведении на нее указателя, в иных случаях она имеет плоский вид
  - **Standard** - Кнопка имеет объемный вид (используется по умолчанию)
  - **System** - Вид кнопки зависит от операционной системы

- Как и для многих элементов управления, для кнопки можно задавать изображение с помощью свойства BackgroundImage.
- Можно также управлять размещением текста и изображения на кнопки.
- Для этого надо использовать свойство **TextImageRelation**.
- Оно приобретает следующие значения:
  - **Overlay**: текст накладывается на изображение
  - **ImageAboveText**: изображение располагается над текстом
  - **TextAboveImage**: текст располагается над изображением
  - **ImageBeforeText**: изображение располагается перед текстом
  - **TextBeforeImage**: текст располагается перед изображением

Label

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

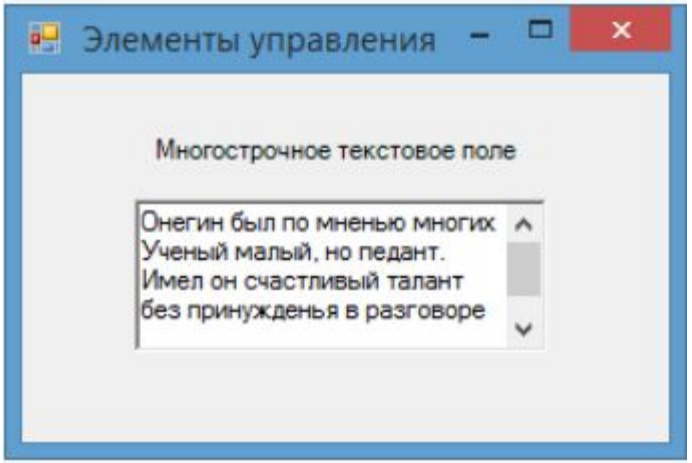
LinkLabel

- Особый тип меток представляют элементы LinkLabel, которые предназначены для вывода ссылок, которые аналогичны ссылкам, размещенным на стандартных веб-станиц.
- Также, как и с обычными ссылками на веб-страницах, мы можем по отношению к данному элементу определить три цвета:
  - Свойство **ActiveLinkColor** задает цвет ссылки при нажатии
  - Свойство **LinkColor** задает цвет ссылки до нажатия, по которой еще не было переходов
  - Свойство **VisitedLinkColor** задает цвет ссылки, по которой уже были переходы

- Кроме цвета ссылки для данного элемента мы можем задать свойство **LinkBehavior**, которое управляет поведением ссылки.
- Это свойство принимает четыре возможных значения:
  - **SystemDefault**: для ссылки устанавливаются системные настройки
  - **AlwaysUnderline**: ссылка всегда подчеркивается
  - **HoverUnderline**: ссылка подчеркивается только при наведении на нее курсора мыши
  - **NeverUnderline**: ссылка никогда не подчеркивается

Текстовое поле TextBox

- Для ввода и редактирования текста предназначены текстовые поля - элемент TextBox.
- Так же как и у элемента Label текст элемента TextBox можно установить или получить с помощью свойства Text.



Текстовое поле TextBox

- По умолчанию при переносе элемента с панели инструментов 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.

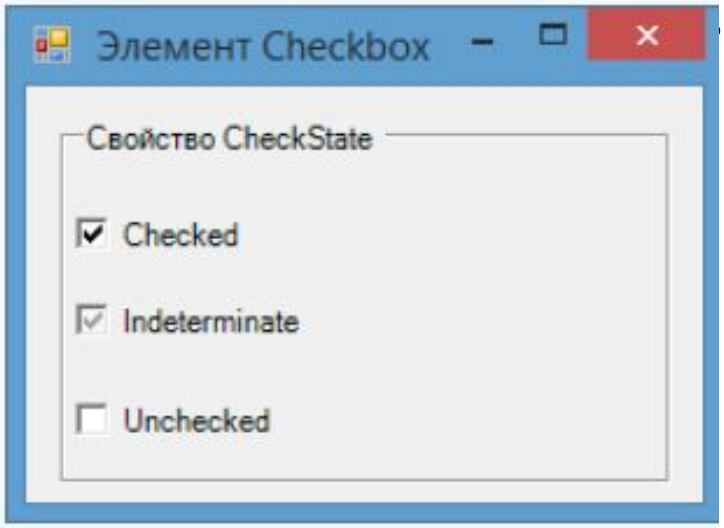
- Кроме того, можно сделать прокрутку текстового поля, установив для его свойства ScrollBars одно из значений:
  - **None**: без прокруток (по умолчанию)
  - **Horizontal**: создает горизонтальную прокрутку при длине строки, превышающей ширину текстового поля
  - **Vertical**: создает вертикальную прокрутку, если строки не помещаются в текстовом поле
  - **Both**: создает вертикальную и горизонтальную прокрутку

Элемент MaskedTextBox

- Элемент MaskedTextBox по сути представляет обычное текстовое поле.
- Однако данные элемент позволяет контролировать ввод пользователя и проверять его автоматически на наличие ошибок.

CheckBox

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



CheckBox

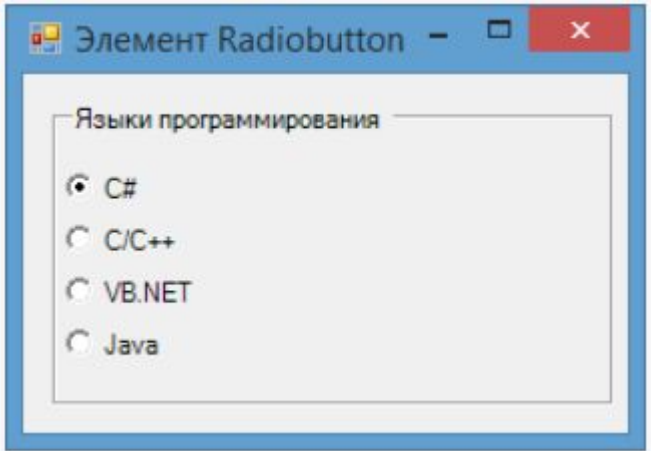
- Кроме свойства Checked у элемента CheckBox имеется свойство **CheckState**, которое позволяет задать для флажка одно из трех состояний - Checked (отмечен), Indeterminate (флажок не определен - отмечен, но находится в неактивном состоянии) и Unchecked (не отмечен)

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.

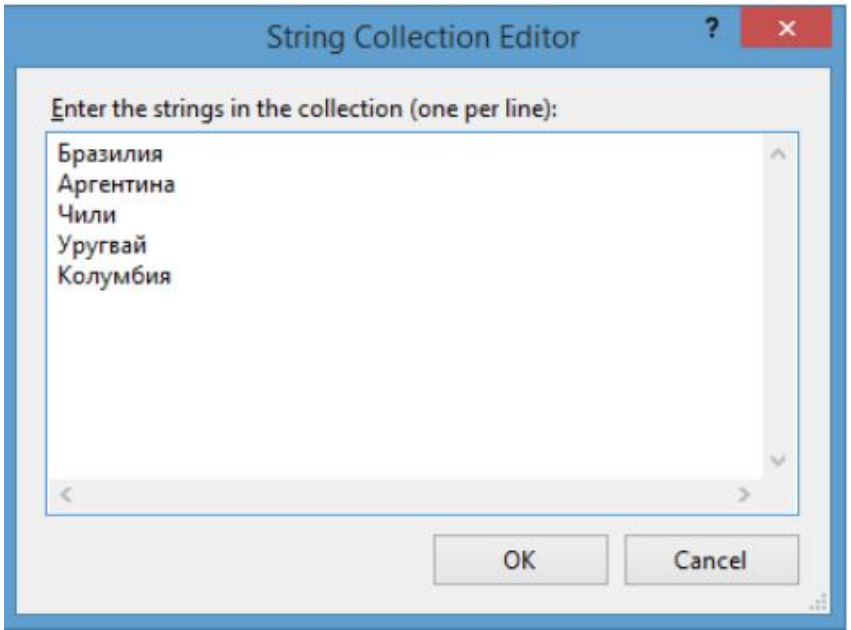
Radiobutton

- 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



ListBox

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



ListBox

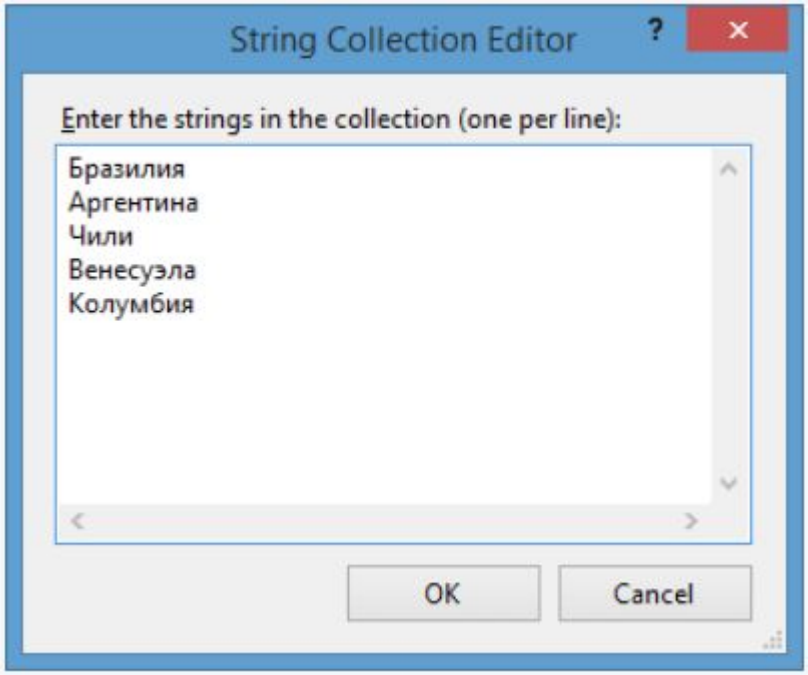
- Элемент ListBox представляет собой простой список.
- Ключевым свойством этого элемента является свойство **Items**, которое как раз и хранит набор всех элементов списка.
- Элементы в список могут добавляться как во время разработки, так и программным способом. В Visual Studio в окне Properties (Свойства) для элемента ListBox мы можем найти свойство Items. После двойного щелчка на свойство нам отобразится окно для добавления элементов в список:

Элемент ComboBox

- Элемент ComboBox образует выпадающий список и совмещает функциональность компонентов ListBox и TextBox.
- Для хранения элементов списка в ComboBox также предназначено свойство **Items**.

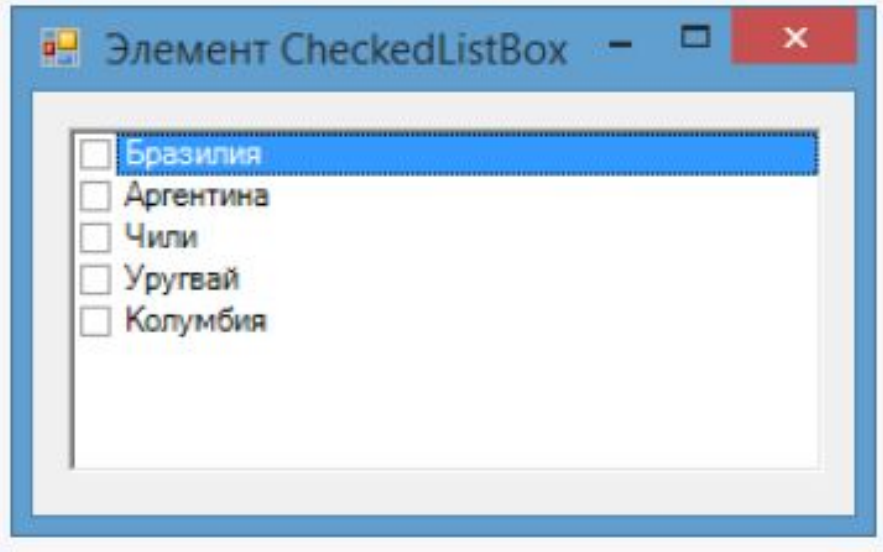
Элемент ComboBox

- Подобным образом, как и с ListBox, мы можем в окне свойств на свойство Items и нам отобразится окно для добавления элементов ComboBox:



Элемент CheckedListBox

- Элемент CheckedListBox представляет симбиоз компонентов ListBox и CheckBox.
- Для каждого элемента такого списка определено специальное поле CheckBox, которое можно отметить.



Элемент CheckedListBox

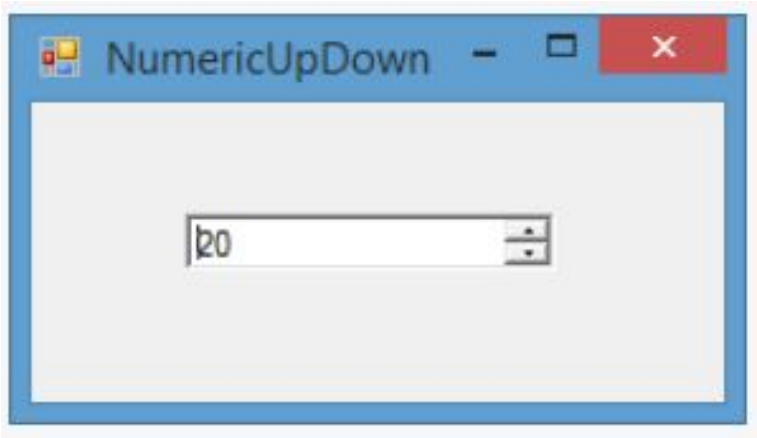
- Все элементы задаются в CheckedListBox задаются в свойстве **Items**.
- Также, как и для элементов ListBox и ComboBox, мы можем задать набор элементов.
- По умолчанию для каждого добавляемого нового элемента флажок не отмечен:

- Чтобы поставить отметку в checkBox рядом с элементом в списке, нам надо сначала выделить элемент и дополнительным щелчком уже установить флажок.
- Однако это не всегда удобно, и с помощью свойства **CheckOnClick** и установке для него значения true мы можем определить сразу выбор элемента и установку для него флажка в один клик.

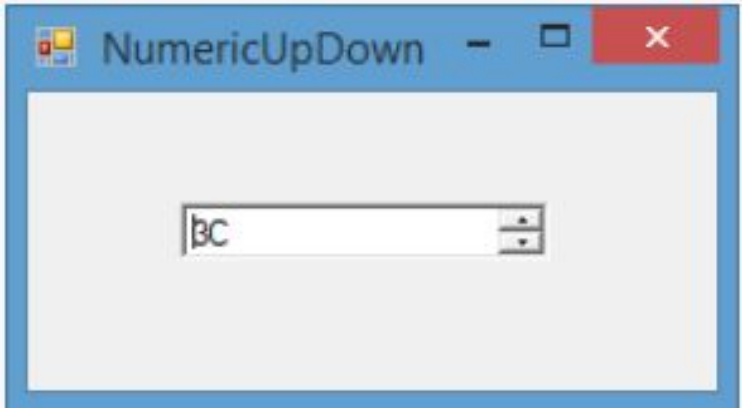
NumericUpDown

- Элемент NumericUpDown представляет пользователю выбор числа из определенного диапазона.
- Для определения диапазона чисел для выбора NumericUpDown имеет два свойства: **Minimum** (задает минимальное число) и **Maximum** (задает максимальное число).

- Само значение элемента хранится в свойстве **Value**:

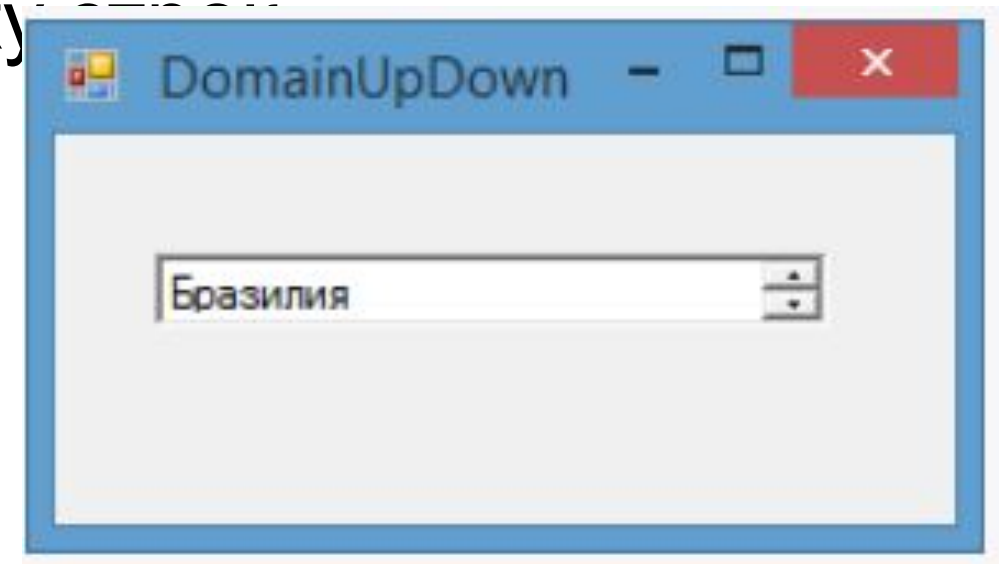


- По умолчанию элемент отображает десятичные числа.
- Однако если мы установим его свойство **Hexadecimal** равным true, то элемент будет отображать все числа в шестнадцатеричной системе.



DomainUpDown

- Элемент DomainUpDown предназначен для ввода текстовой информации.
- Он имеет текстовое поле для ввода строки и две стрелки для перемещения по списку строк.



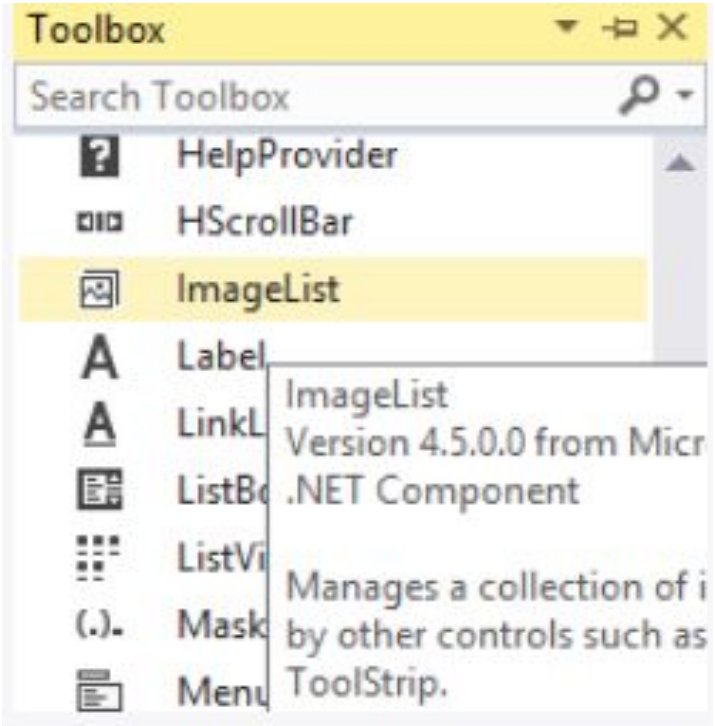
- Список для DomainUpDown задается с помощью свойства **Items**. Список можно сразу упорядочить по алфавиту. Для этого надо свойству **Sorted** присвоить значение true.
- Чтобы можно было циклично перемещаться по списку, то есть при достижении конца или начала списка его просмотр начинался с первого или последнего элемента, надо установить для свойства **Wrap** значение true.

ImageList

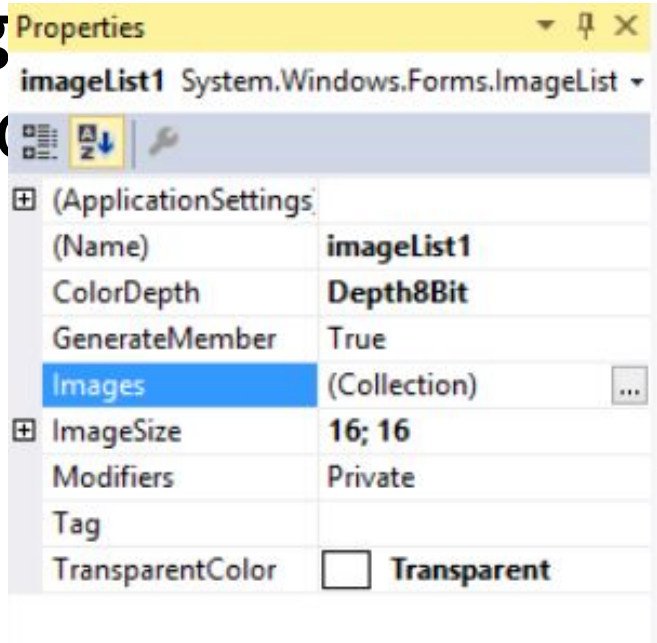
- ImageList не является визуальным элементом управления, однако он представляет компонент, который используется элементами управления.
- Он определяет набор изображений, который могут использовать такие элементы, как ListView или TreeView.

ImageList

- Чтобы его добавить в проект, его также можно перенести на форму с Панели Инструментов:



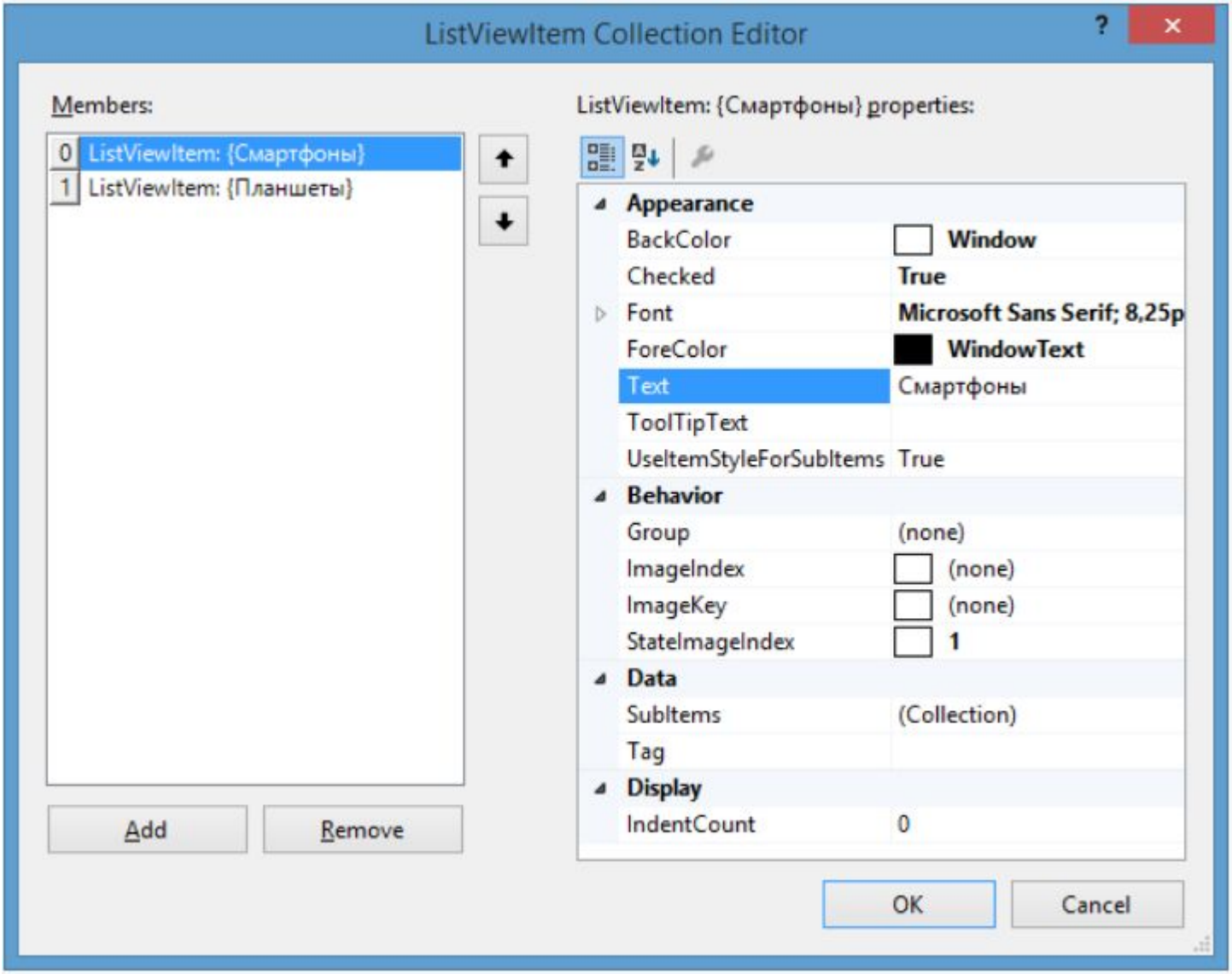
- Так как компонент не является визуальным элементом, то мы увидим его под формой.
- Ключевым свойством ImageList является свойство **Imag** ет коллекцию из



ListView

- Элемент ListView представляет список, но с более расширенными возможностями, чем ListBox.
- В ListView можно отображать сложные данные в различных стобцах, можно задавать данным изображения и пиктограммы.

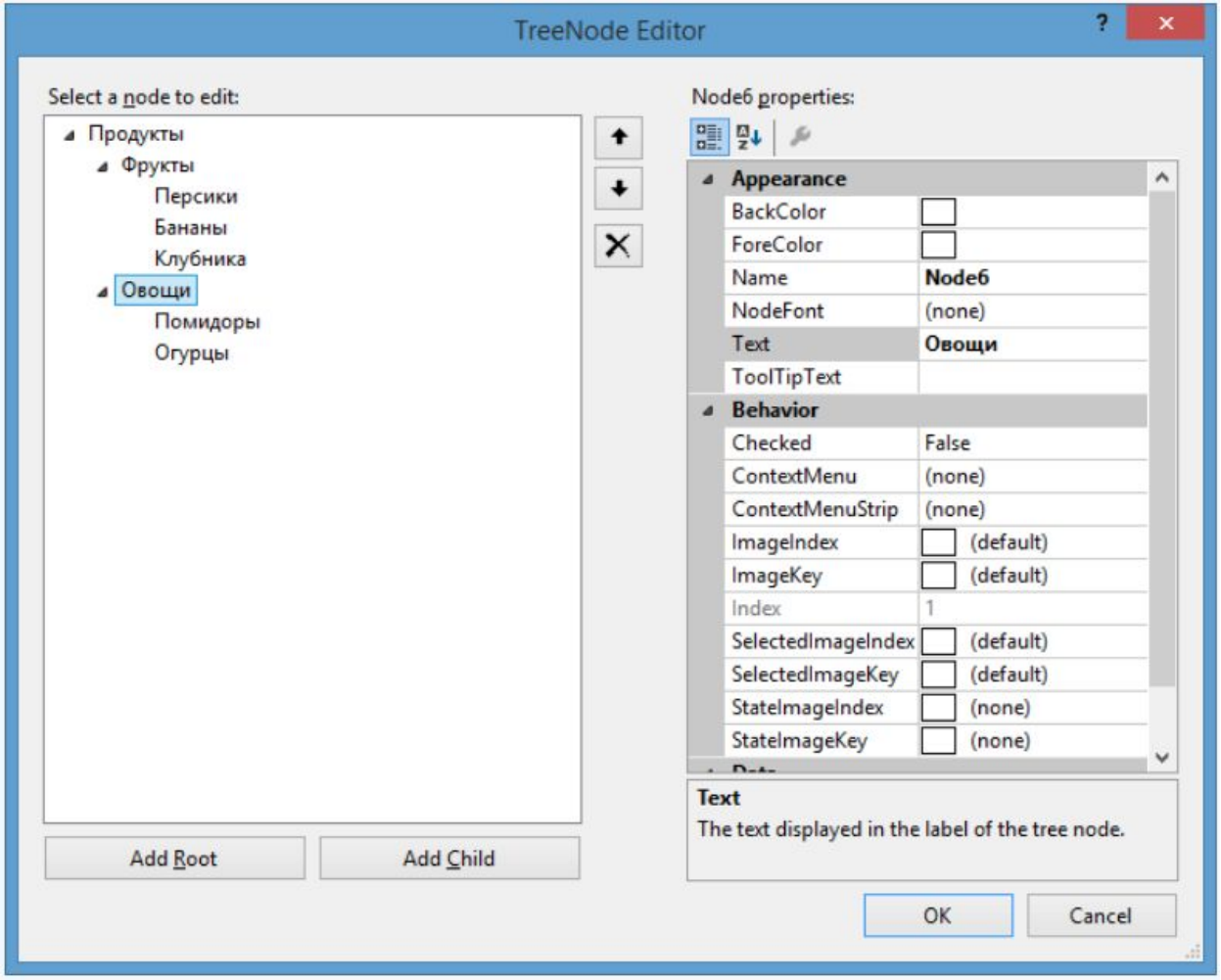
- Все элементы, как и в других списковых визуальных компонентах, задаются с помощью свойства **Items**.
- Но в отличие от ListBox или ComboBox, если мы через панель Свойств откроем окно редактирования элементов ListView:


ListViewItem Collection Editor
Members:
0 ListViewItem: {Смартфоны}
1 ListViewItem: {Планшеты}
Add
Remove
ListViewItem: {Смартфоны} properties:
Appearance
BackColor Window
Checked True
Font Microsoft Sans Serif; 8,25p
ForeColor WindowText
Text Смартфоны
ToolTipText
UseItemStyleForSubItems True
Behavior
Group (none)
ImageIndex (none)
ImageKey (none)
StateImageIndex 1
Data
SubItems (Collection)
Tag
Display
IndentCount 0
OK
Cancel

TreeView

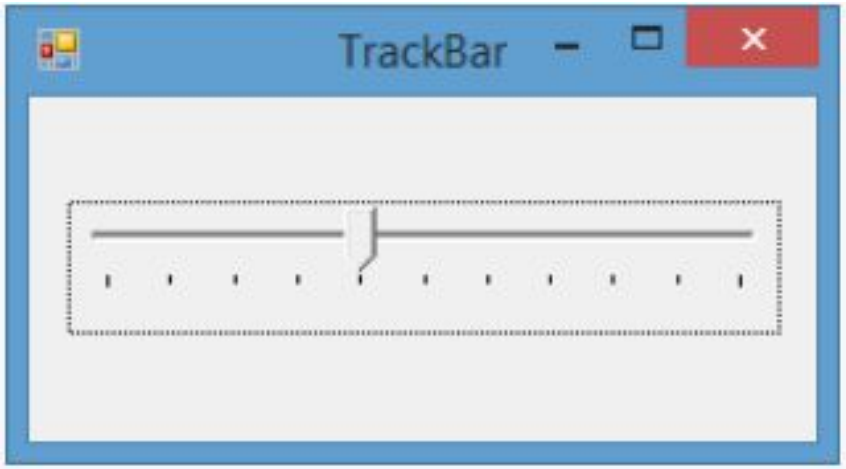
- TreeView представляет визуальный элемент в виде дерева.
- Дерево содержит узлы, которые представляют объекты **TreeNode**. Узлы могут содержать другие подузлы и могут находиться как скрытом, так и в раскрытом состоянии.
- Все узлы содержатся в свойстве **Nodes**.

Если мы нажем в панели Свойств на свойство Nodes, то нам откроется окно редактирования узлов TreeView:



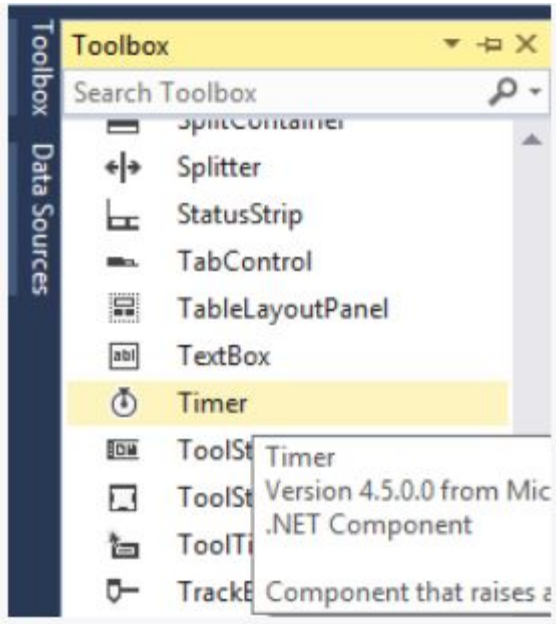
TrackBar

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



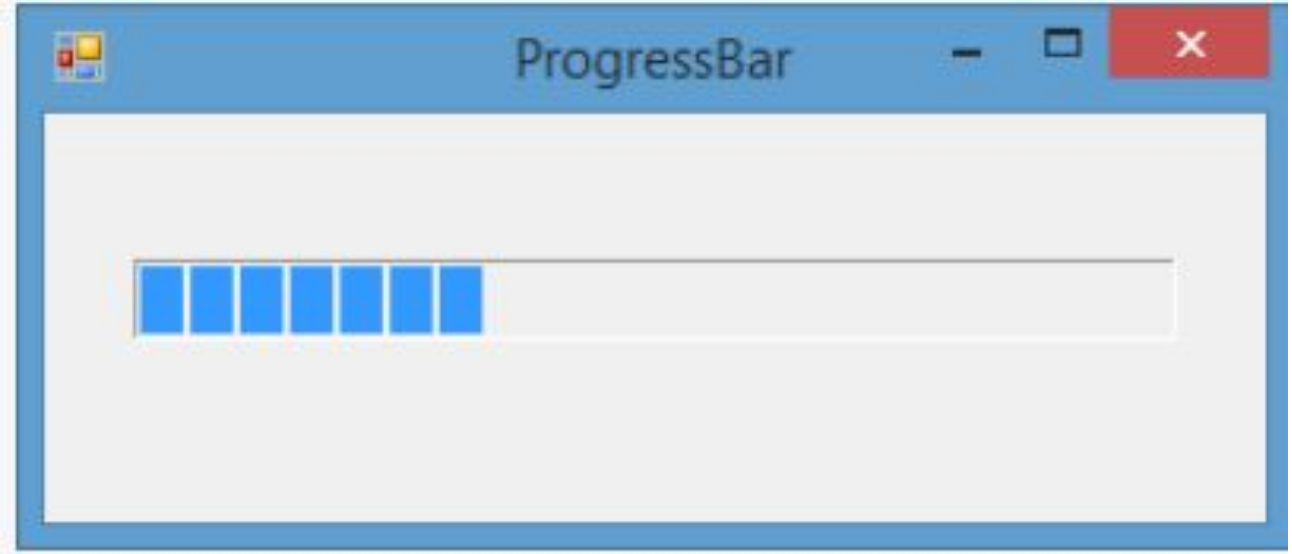
Timer

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



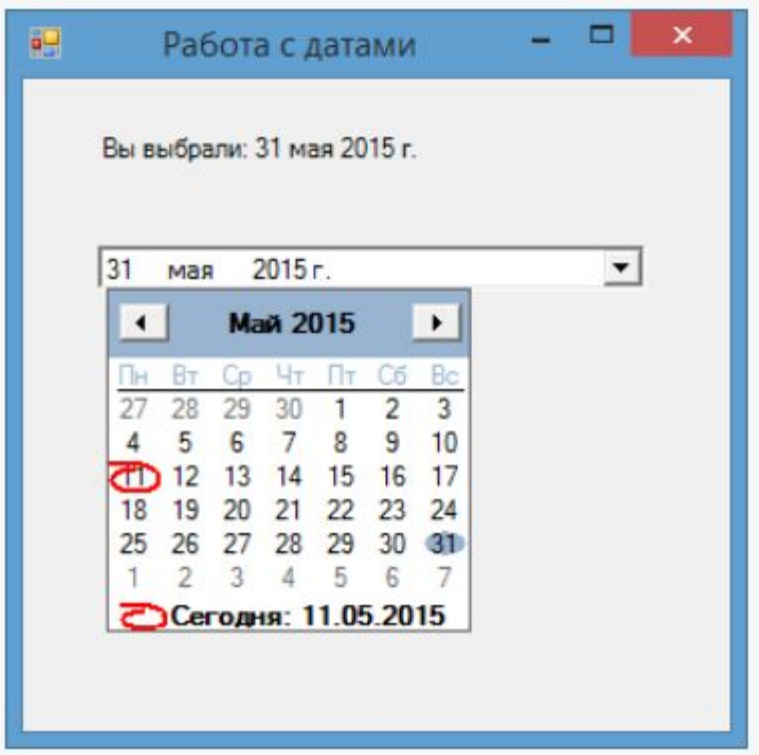
Индикатор прогресса ProgressBar

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



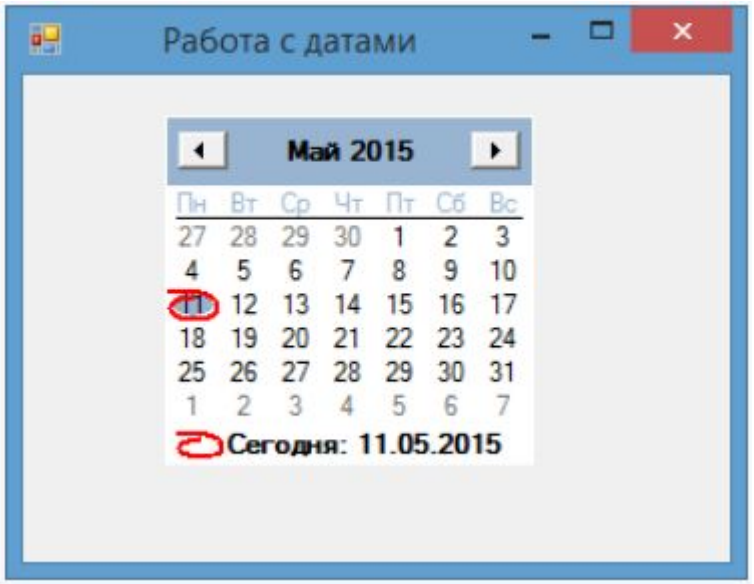
DateTimePicker

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



MonthCalendar

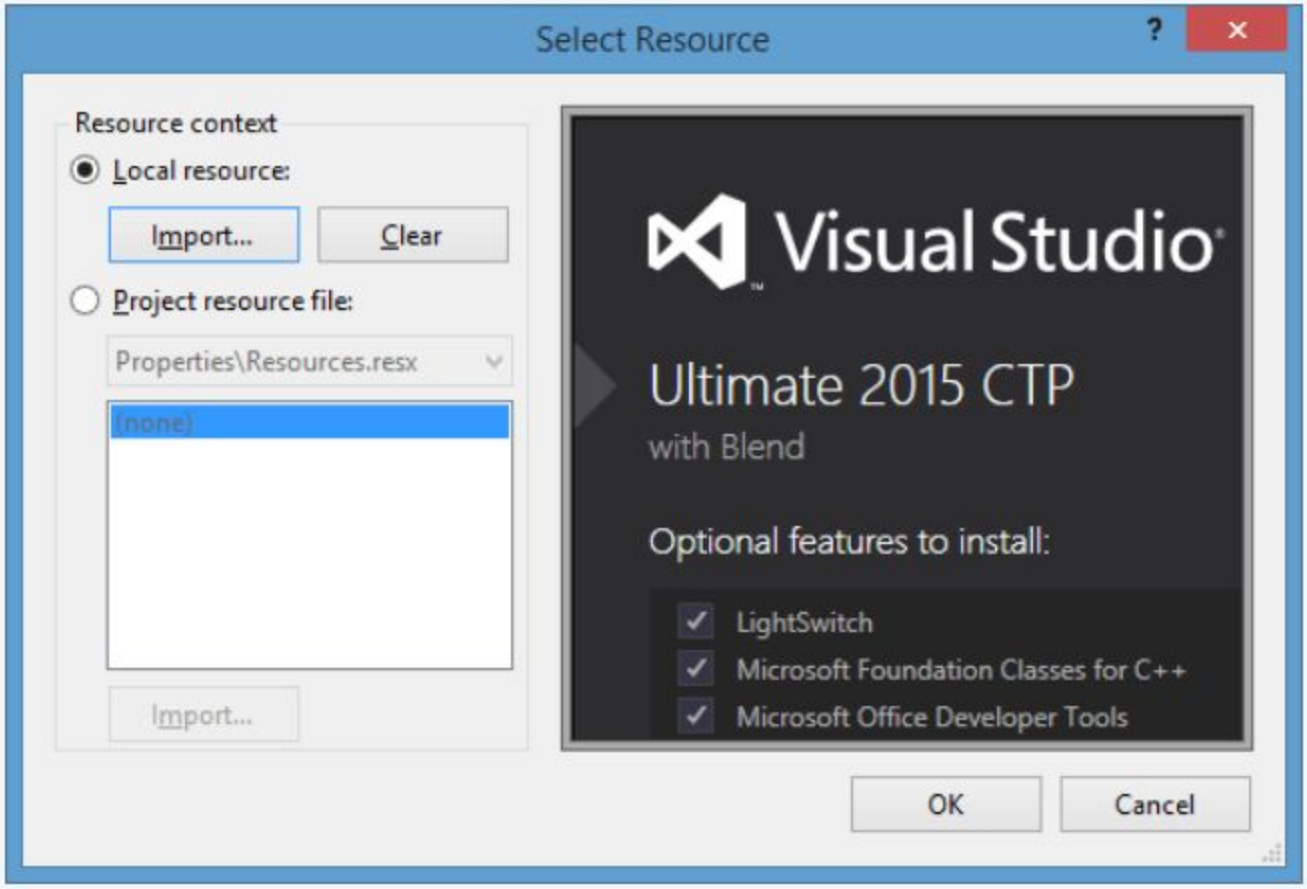
- С помощью MonthCalendar также можно выбрать дату, только в данном случае этот элемент представляет сам календарь, который не надо раскрывать:



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:


Select Resource
Resource context
Local resource:
Import...
Clear
Project resource file:
Properties\Resources.resx
(none)
Import...
Visual Studio
Ultimate 2015 CTP
with Blend
Optional features to install:
LightSwitch
Microsoft Foundation Classes for C++
Microsoft Office Developer Tools
OK
Cancel

Элемент WebBrowser

- WebBrowser предоставляет функции интернет-браузера, позволяя загружать и отображать контент из сети интернет.
- В то же время важно понимать, что данный элемент не является полноценным веб-браузером, и возможности по его настройки и изменению довольно ограничены.

Элемент NotifyIcon

- Элемент NotifyIcon позволяет задать значок, который будет отображаться при запуске приложения в панели задач.