

ЛЕКЦИЯ 5

5

СЪДЪРЖАНИЕ

I. Основни контроли.

II. Разполагане на контроли във формата.

1. Етикети

Етикетите са едни от най-често използваните C# контроли. Можем да използваме контролата **Label**, за да покажем текст на зададено място на страницата. Контролите за етикети могат също да се използват за добавяне на описателен текст към формуляр за попълване, за да се предостави на потребителя полезна информация. Класът **Label** е дефиниран в пространството на имената `System.Windows.Forms`.

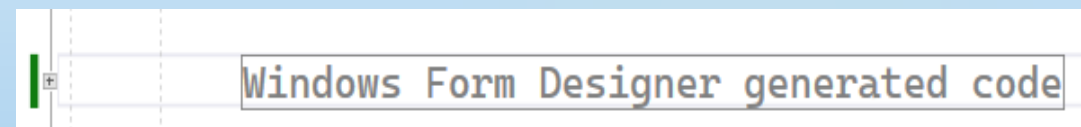
The diagram illustrates the process of creating a Label control in a Windows Forms application. It consists of three main parts:

- Toolbox:** A screenshot of the Visual Studio Toolbox showing the 'Common Controls' section. The 'Label' control is highlighted, and a red arrow points from it to the visual representation of the label.
- Visual Representation:** A visual representation of a Label control, showing a rectangular box with the text 'This is a Label' inside. A red arrow points from this visual representation to the C# code snippet.
- C# Code Snippet:** A code snippet showing the instantiation and configuration of a Label control in C#. The code is enclosed in a dashed red box, and a red arrow points from it to a text box. The code is as follows:

```
private Label label1;  
  
this.label1.AutoSize = true;  
this.label1.Font = new System.Drawing.Font("Segoe UI", 16.2F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point);  
  
this.label1.Location = new System.Drawing.Point(432, 314);  
this.label1.Name = "label1";  
this.label1.Size = new System.Drawing.Size(189, 38);  
this.label1.TabIndex = 7;  
this.label1.Text = "This is a Label";  
this.label1.Click += new System.EventHandler(this.label1_Click);
```
- Text Box:** A text box containing the text 'Инстанция от класа Label', which translates to 'Instance of the Label class'. A red arrow points from the C# code snippet to this text box.

Динамично управление на етикетите чрез контрол на някои техни свойства:

В допълнение към показването на текст контролата Label може също да показва изображение чрез свойството Image или комбинация от свойствата ImageIndex и ImageList.



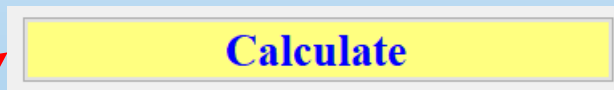
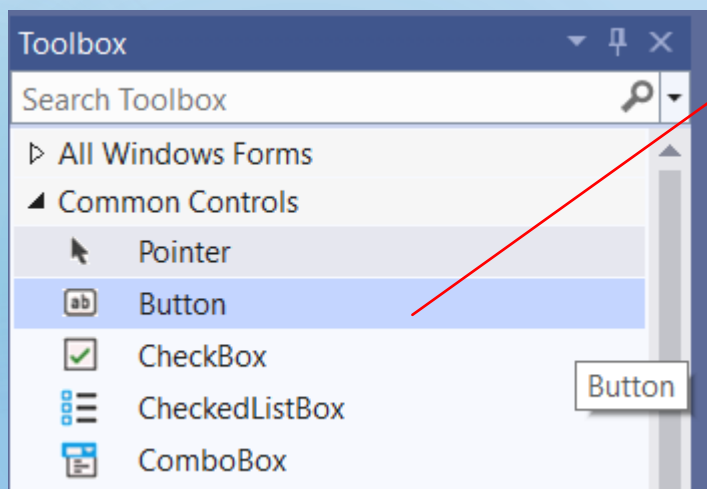
```
//  
// label1  
//
```

```
this.label1.Image = Image.FromFile("C:\\testimage.jpg");
```

```
this.label1.BorderStyle = BorderStyle.FixedSingle;  
this.label1.TextAlign = ContentAlignment.MiddleCenter;
```

2. Бутони

Бутонът е контрола, която е интерактивен компонент, позволяващ на потребителите да комуникират с дадено приложение. Класът Button наследява директно от класа ButtonBase. Върху бутон може да се щракне с помощта на мишката, клавиша ENTER или ИНТЕРВАЛ, ако бутонът има фокус.



```
private Button buttonCalculate;
```

Инстанция от
класа Button

Ако искате да заредите изображение в контрола на бутон, можете да използвате следния код:

```
this.buttonCalculate.Image = Image.FromFile("C:\\testimage.jpg");
```

Form1.Designer.cs

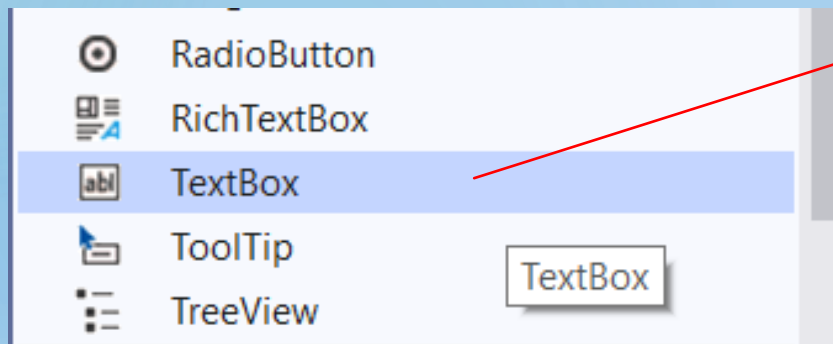
Form1.cs [Design]*

Windows Form Designer generated code

```
//  
// buttonCalculate  
//  
this.buttonCalculate.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(255)))),  
                                                                    ((int)(((byte)(255)))),  
                                                                    ((int)(((byte)(128)))));  
this.buttonCalculate.Font = new System.Drawing.Font("Times New Roman", 16.2F,  
                                                    System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);  
this.buttonCalculate.ForeColor = System.Drawing.Color.Blue;  
this.buttonCalculate.Location = new System.Drawing.Point(14, 85);  
this.buttonCalculate.Name = "buttonCalculate";  
this.buttonCalculate.Size = new System.Drawing.Size(370, 42);  
this.buttonCalculate.TabIndex = 5;  
this.buttonCalculate.Text = "Calculate";  
this.buttonCalculate.UseVisualStyleBackColor = false;  
this.buttonCalculate.Click += new System.EventHandler(this.buttonCalculate_Click);
```

3. Текстови полета

Контрола TextBox се използва за показване или приемане като вход на един ред текст. Тази контрола има допълнителна функционалност, която не се намира в стандартната контрола на текстовото поле на Windows, включително многоредово редактиране и маскиране на знаци за парола. Обект на текстово поле се използва за показване на текст във формуляр или за получаване на въведени от потребителя данни. В текстово поле потребителят може да въведе данни или да ги постави в контролата от клипборда.



```
private TextBox textBoxNumLeft;  
private TextBox textBoxNumRight;
```

Можете също така да прочетете въведената от потребителя стойност в контролата TextBox и да я запишете в променлива по този начин:

```
string var;  
var = textBoxNumLeft.Text;
```


I. ОСНОВНИ КОНТРОЛИ

5

Form1.Designer.cs

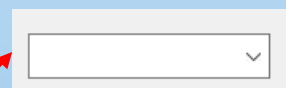
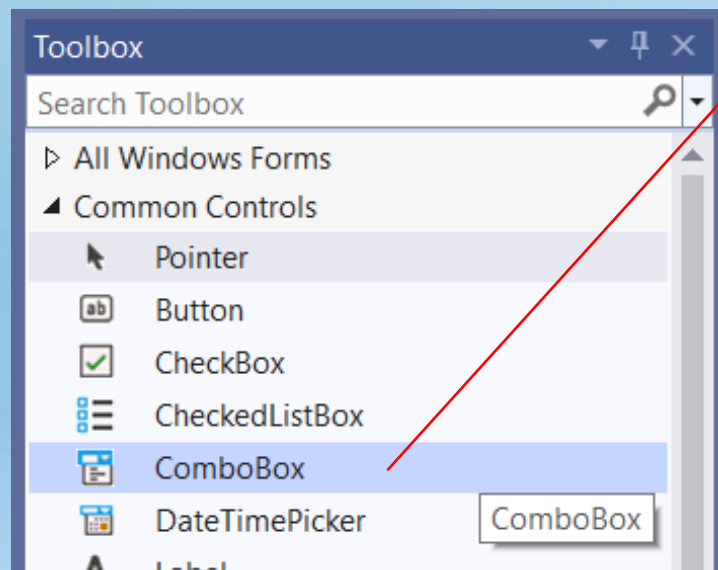
Form1.cs [Design]*

Windows Form Designer generated code

```
//  
// textBoxResult  
//  
this.textBoxResult.BackColor = System.Drawing.SystemColors.ButtonHighlight;  
this.textBoxResult.Font = new System.Drawing.Font("Times New Roman", 16.2F,  
                                                    System.Drawing.FontStyle.Bold,  
                                                    System.Drawing.GraphicsUnit.Point);  
this.textBoxResult.ForeColor = System.Drawing.Color.Lime;  
this.textBoxResult.Location = new System.Drawing.Point(291, 21);  
this.textBoxResult.Name = "textBoxResult";  
this.textBoxResult.ReadOnly = true;  
this.textBoxResult.Size = new System.Drawing.Size(93, 39);  
this.textBoxResult.TabIndex = 4;
```

4. Контрола от типа ComboBox

Контролата от типа ComboBox показва текстово поле, комбинирано със ListBox, което позволява на потребителя да избира елементи от списък или да въвежда нова стойност. Потребителят може да въведе стойност в текстовото поле или да щракне върху бутона, за да покаже падащ списък. Можете да добавяте отделни обекти с метода **Add**. Можете да изтриете елементи с метода **Remove** или да изчистите целия списък с метода **Clear**.



```
private ComboBox comboBox1;
```

```
//  
// comboBox1  
//  
this.comboBox1.FormattingEnabled = true;  
this.comboBox1.Items.AddRange(new object[] {  
    "Опция 1",  
    "Опция 2",  
    "Опция 3"});  
this.comboBox1.Location = new System.Drawing.Point(85, 276);  
this.comboBox1.Name = "comboBox1";  
this.comboBox1.Size = new System.Drawing.Size(151, 28);  
this.comboBox1.TabIndex = 8;
```


I. ОСНОВНИ КОНТРОЛИ



Динамично управление на контролата ComboBox

- Как да добавите елемент към ComboBox контрола:

```
this.comboBox1.Items.Add("Sunday");  
this.comboBox1.Items.Add("Monday");  
this.comboBox1.Items.Add("Tuesday");
```

- Как да извлечете стойност от ComboBox контрола:

Начин 1:

```
string var;  
var = comboBox1.Text;
```

Начин 2:

```
var item = this.comboBox1.GetItemText(this.comboBox1.SelectedItem);  
MessageBox.Show(item);
```

- Как да премахнете елемент от ComboBox контрола:

```
comboBox1.Items.RemoveAt(1); или comboBox1.Items.Remove("Friday");
```

I. ОСНОВНИ КОНТРОЛИ

5

- Как да зададете конкретен елемент в ComboBox контрола:

Начин 1:

```
comboBox1.Items.Add("test1");  
comboBox1.Items.Add("test2");  
comboBox1.Items.Add("test3");  
comboBox1.SelectedItem = "test3";
```



Начин 2:

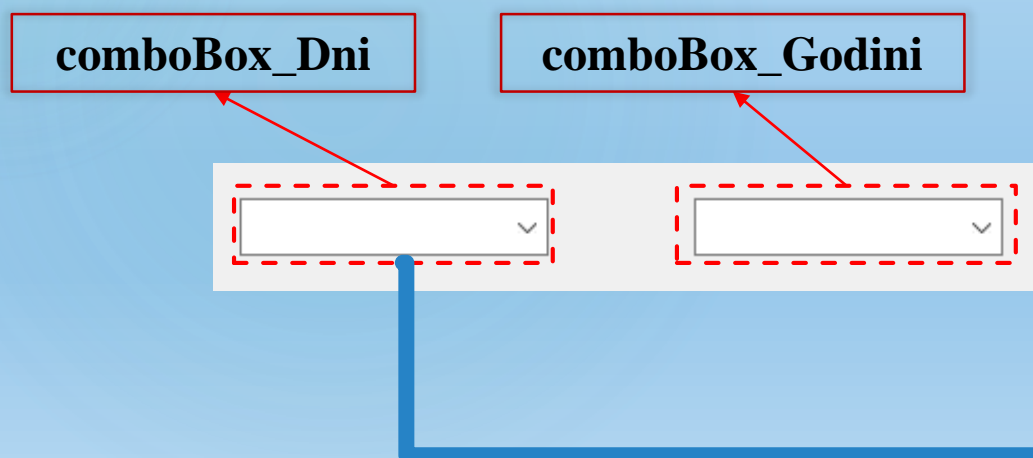
```
comboBox1.SelectedIndex = comboBox1.FindStringExact("test2");
```



- Използване на събитието **SelectedIndexChanged** за ComboBox контроли:

Събитието **SelectedIndexChanged** на ComboBox контрола се активира, когато промените избрания елемент в даден ComboBox. Ако искате да направите нещо, когато промените селекцията, можете да използвате събитието **SelectedIndexChanged**.

От следващия код можете да разберете как да задавате стойности в събитието **SelectedIndexChanged** на поле със списък.



```
//  
// comboBox1  
//  
this.comboBox1.FormattingEnabled = true;  
this.comboBox1.Items.AddRange(new object[] {  
    "Дни от седмицата",  
    "Години"});
```

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    comboBox2.Items.Clear();
    if (comboBox1.SelectedItem == "Дни от седмицата")
    {
        comboBox2.Items.Add("Sunday");
        comboBox2.Items.Add("Monday");
        comboBox2.Items.Add("Tuesday");
    }
    else if (comboBox1.SelectedItem == "Години")
    {
        comboBox2.Items.Add("2012");
        comboBox2.Items.Add("2013");
        comboBox2.Items.Add("2014");
    }
}
```

Автоматично попълване на
опции за избор за дните от
седмицата

Автоматично попълване на
опции за избор години

I. ОСНОВНИ КОНТРОЛИ



- Задаване на стойност по подразбиране за ComboBox контроли:

Можете да зададете стойност по подразбиране на ComboBox контрола, като използвате **свойството SelectedIndex** или динамично чрез използване на следния КОД:

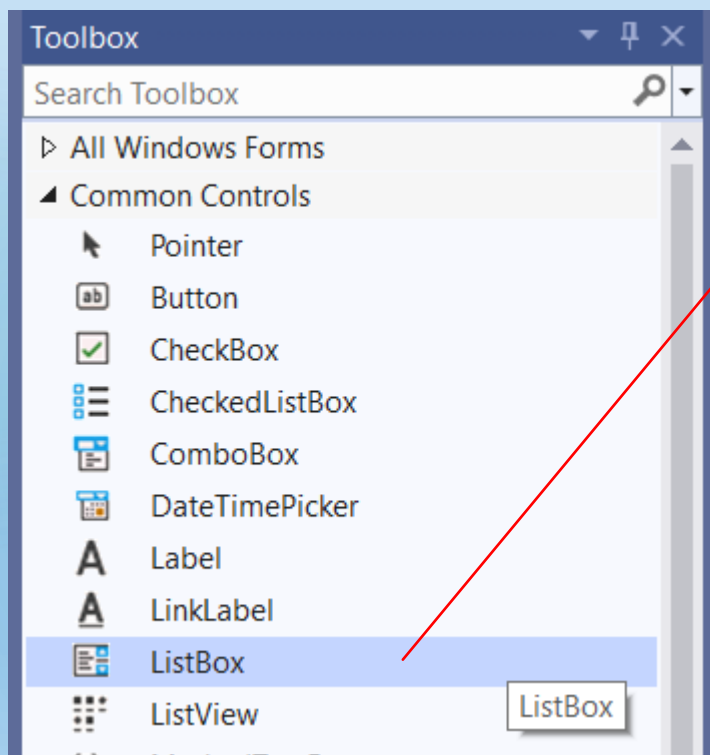
```
comboBox1.SelectedIndex = 6;
```

- ComboBox контроли само за четене:

Можете да направите ComboBox само за четене, което означава, че потребителят не може да пише в бялото поле, но може да избира дадените елементи по два начина. По подразбиране свойството DropDownStyle на Combobox е DropDown. В този случай потребителят може да въвежда стойности в ComboBox-а. Когато промените свойството DropDownStyle на DropDownList, Combobox контролата ще стане само за четене и потребителят не може да въвежда стойности в combobox-а. Втори метод, ако искате комбинираният списък изцяло само за четене, можете да зададете comboBox1.Enabled = false.

5. Контрола от типа ListBox

Контролата ListBox ви позволява да покажете на потребителя списък с елементи, които потребителят може да избере чрез щракване.



Елемент 1 от списъка
Елемент 2 от списъка
Елемент 3 от списъка
Елемент 4 от списъка
Елемент 5 от списъка

```
private ListBox listBox1;
```

```
//  
// listBox1  
//  
this.listBox1.FormattingEnabled = true;  
this.listBox1.ItemHeight = 20;  
this.listBox1.Items.AddRange(new object[] {  
    "Елемент 1 от списъка ",  
    "Елемент 2 от списъка ",  
    "Елемент 3 от списъка ",  
    "Елемент 4 от списъка ",  
    "Елемент 5 от списъка "});  
this.listBox1.Location = new System.Drawing.Point(624, 233);  
this.listBox1.Name = "listBox1";  
this.listBox1.Size = new System.Drawing.Size(190, 164);  
this.listBox1.TabIndex = 11;
```


I. ОСНОВНИ КОНТРОЛИ



- Добавяне на елементи в контрола от типа ListBox:

В допълнение към функционалността за показване и избор, ListBox предоставя и функции, които ви позволяват ефективно да добавяте елементи към ListBox и да намирате текст в елементите на списъка. Можете да използвате **метода Add** или **Insert**, за да добавите елементи към списъчно поле. **Методът Add добавя нови елементи в края на несортирано списъчно поле.**

Начин 1:

```
this.listBox1.Items.AddRange(new object[] {  
    "Елемент 1 от списъка ",  
    "Елемент 2 от списъка ",  
    "Елемент 3 от списъка ",  
    "Елемент 4 от списъка ",  
    "Елемент 5 от списъка "});
```

Начин 2:

```
listBox1.Items.Add("Sunday");  
listBox1.Items.Add("Monday");  
listBox1.Items.Insert(2, "Inserted item is Here");
```

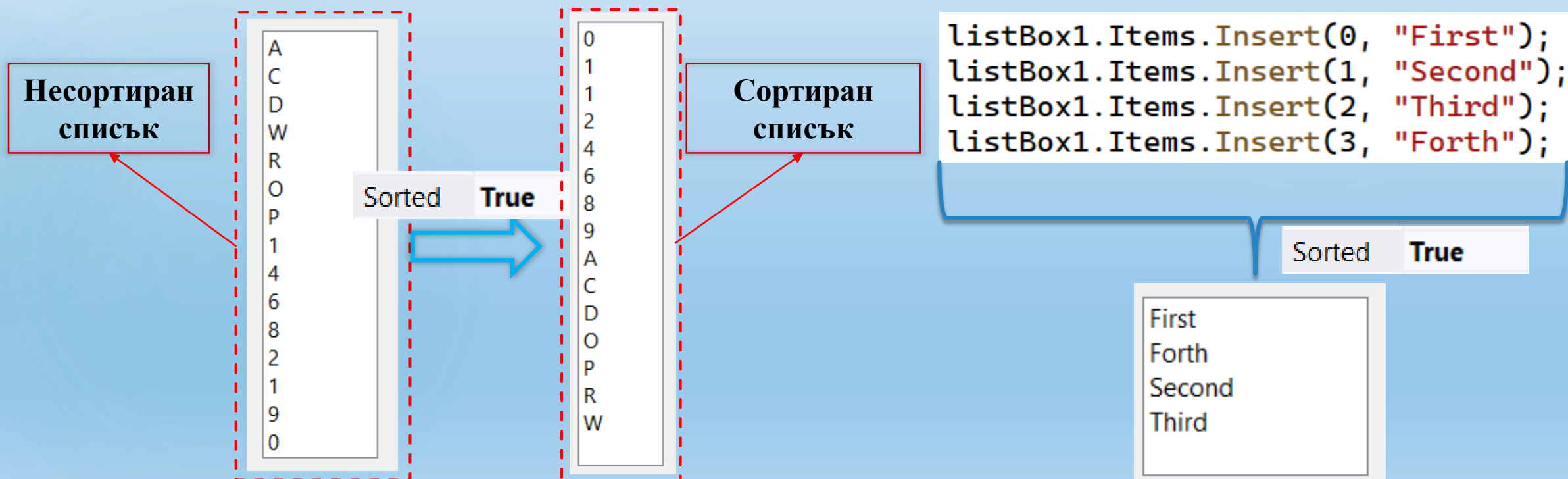
Елемент 1 от списъка
Елемент 2 от списъка
Inserted item is Here
Елемент 3 от списъка
Елемент 4 от списъка
Елемент 5 от списъка
Monday
Sunday

I. ОСНОВНИ КОНТРОЛИ



- Сортиране на елементите в контрола от типа ListBox:

Ако свойството **Sorted** на ListBox е зададено на **true**, елементът се вмъква в списъка по **азбучен ред**. В противен случай елементът се вмъква в **края на ListBox**.



I. ОСНОВНИ КОНТРОЛИ



- Извличане на единичен избран елемент от ListBox и присвояване на прочетената стойност в променлива:

```
string item = listBox1.GetItemText(listBox1.SelectedItem);
```

ИЛИ

```
string item2 = listBox1.SelectedItem.ToString();
```

ИЛИ

```
string item3 = listBox1.Text;
```

- Избиране на множество елементи от Listbox - Свойството **SelectionMode** определя колко елемента в списъка могат да бъдат избрани наведнъж. Контрола ListBox може да предостави единични или множество селекции с помощта на свойството SelectionMode. Ако промените свойството за режим на избор на множествено избиране, тогава ще извлечете колекция от елементи от свойството **ListBox1.SelectedItems**.

```
listBox1.SelectionMode = SelectionMode.MultiSimple;
```

Класът **ListBox** има два **SelectionMode** режима на работа. Множество или Разширен. В режим Множество можете да изберете или премахнете избора на всеки елемент в **ListBox**, като щракнете върху него. В Разширен режим трябва да задържите клавиша **Ctrl**, за да изберете допълнителни елементи, или клавиша **Shift**, за да изберете набор от елементи.

Следващият пример на **C#** програма първоначално запълва седем дни в седмицата, докато е в събитието за зареждане на формуляра и задава свойството за режим на избор на **MultiSimple**. При събитието с щракване върху бутон ще се покажат избраните елементи.

```
listBox1.Items.Add("Sunday");  
listBox1.Items.Add("Monday");  
listBox1.Items.Add("Tuesday");  
listBox1.Items.Add("Wednesday");  
listBox1.Items.Add("Thursday");  
listBox1.Items.Add("Friday");  
listBox1.Items.Add("Saturday");  
listBox1.SelectionMode = SelectionMode.MultiSimple;
```

```
0 references  
private void button1_Click(object sender, EventArgs e)  
{  
    foreach (Object obj in listBox1.SelectedItems)  
    {  
        MessageBox.Show(obj.ToString());  
    }  
}
```

I. ОСНОВНИ КОНТРОЛИ

5

- Извличане на всички елементи от ListBox:

Колекцията Items на C# Winforms Listbox връща тип колекция от типа Object, така че можете да използвате ToString() за всеки елемент, за да покажете неговата текстова стойност, както е показано по-долу:

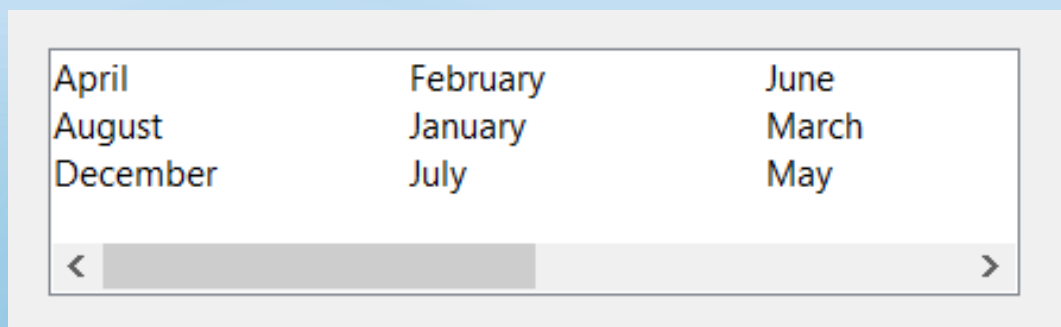
```
private void button1_Click_1(object sender, EventArgs e)
{
    string items = "";
    foreach (var item in listBox1.Items)
    {
        items += item.ToString() + ", ";
    }
    MessageBox.Show(items);
}
```

I. ОСНОВНИ КОНТРОЛИ



- ListBox в многоколонен режим:

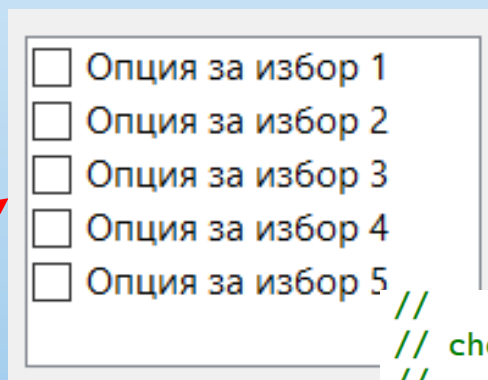
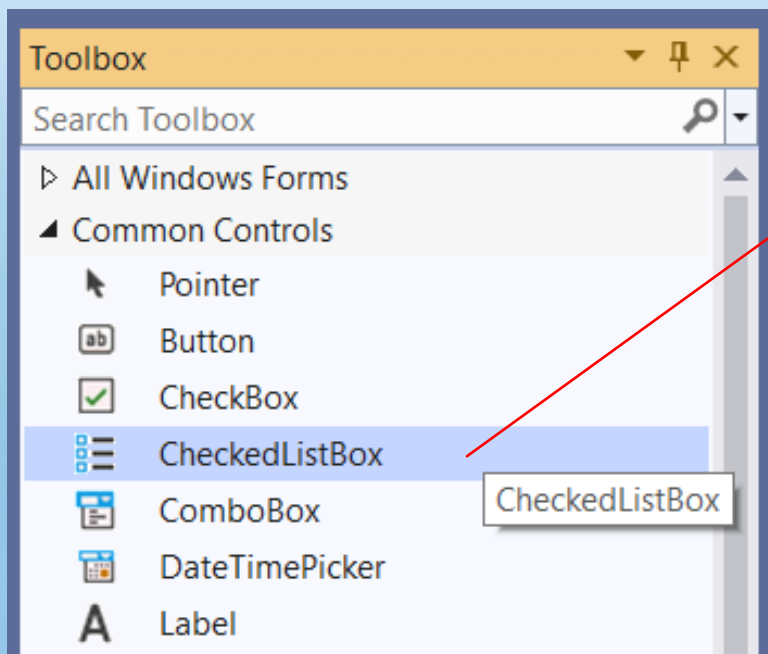
Списъчно поле с няколко колони поставя елементи в толкова колони, колкото са необходими, за да направи вертикалното превъртане ненужно. Потребителят може да използва клавиатурата, за да навигира до колони, които в момента не са видими.



```
this.listBox1.FormattingEnabled = true;  
this.listBox1.HorizontalScrollbar = true;  
this.listBox1.ItemHeight = 20;  
this.listBox1.Items.AddRange(new object[] {  
    "April",  
    "August",  
    "December",  
    "February",  
    "January",  
    "July",  
    "June",  
    "March",  
    "May",  
    "November",  
    "October",  
    "September"});
```


6. Контрола от типа **CheckedListBox**

Потребителят може да постави отметка до един или повече елементи и отметнатите елементи могат да бъдат навигирани с помощта на класовете `CheckedListBox.CheckedItemCollection` и `CheckedListBox.CheckedIndexCollection`.



```
private CheckedListBox checkedListBox1;
```

```
//  
// checkedListBox1  
//  
this.checkedListBox1.FormattingEnabled = true;  
this.checkedListBox1.Items.AddRange(new object[] {  
    "Опция за избор 1",  
    "Опция за избор 2",  
    "Опция за избор 3",  
    "Опция за избор 4",  
    "Опция за избор 5"});  
this.checkedListBox1.Location = new System.Drawing.Point(479, 383);  
this.checkedListBox1.Name = "checkedListBox1";  
this.checkedListBox1.Size = new System.Drawing.Size(187, 136);  
this.checkedListBox1.TabIndex = 12;
```

I. ОСНОВНИ КОНТРОЛИ



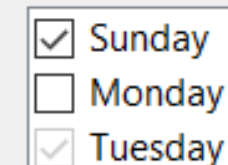
- Динамично добавяне на елементи в контролата `CheckedListBox`:

Списъчно поле с няколко колони поставя елементи в толкова колони, колкото са необходими, за да направи вертикалното превъртане ненужно. Потребителят може да използва клавиатурата, за да навигира до колони, които в момента не са видими.

```
public int Add(object item, bool isChecked);
```

Можете да добавяте отделни елементи към списъка с **метода Add**. Обектът `CheckedListBox` поддържа три състояния чрез изброяването на `CheckState`: `Checked`, `Indeterminate`, и `Unchecked`.

```
checkedListBox1.Items.Add("Sunday", CheckState.Checked);  
checkedListBox1.Items.Add("Monday", CheckState.Unchecked);  
checkedListBox1.Items.Add("Tuesday", CheckState.Indeterminate);
```



I. ОСНОВНИ КОНТРОЛИ

5

Ако искате да добавите обекти към списъка по време на изпълнение, задайте масив от препратки към обекти с метода `AddRange`. След това списъкът показва стойността на низа по подразбиране за всеки обект.

```
string[] days = new[] { "Sunday", "Monday", "Tuesday" };  
checkedListBox1.Items.AddRange(days);
```

По подразбиране елементите в списъка с отметки не са отменати (маркирани).

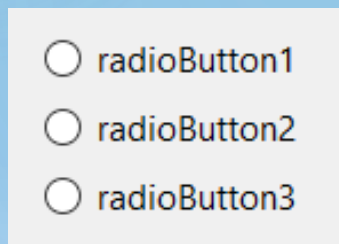
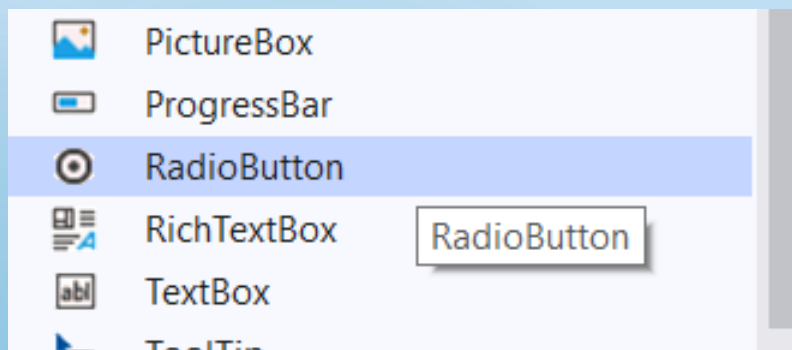
Ако искате в хода на работа на програмата да поставите отметка за **всички елементи** в `CheckedListBox`, променете стойността на метода **`SetItemChecked`** на **`true`** чрез използване на цикъл.

```
for (int i = 0; i < checkedListBox1.Items.Count; i++)  
{  
    checkedListBox1.SetItemChecked(i, true);  
}
```

Или задайте `false`, за
демаркиране

7. Контрола от типа RadioButton

Бутон за избор или RadioButton позволява на потребителя да избере една опция от група възможности за избор, когато е сдвоена с други контроли на RadioButton. Когато потребител щракне върху радио бутон, той става отметнат, а всички други радио бутони от същата група стават немаркирани.



```
private RadioButton radioButton1;  
private RadioButton radioButton2;  
private RadioButton radioButton3;
```

```
//  
// radioButton2  
//  
this.radioButton2.AutoSize = true;  
this.radioButton2.Location = new System.Drawing.Point(732, 451);  
this.radioButton2.Name = "radioButton2";  
this.radioButton2.Size = new System.Drawing.Size(117, 24);  
this.radioButton2.TabIndex = 14;  
this.radioButton2.TabStop = true;  
this.radioButton2.Text = "radioButton2";  
this.radioButton2.UseVisualStyleBackColor = true;  
  
radioButton1.Checked = true;
```

I. ОСНОВНИ КОНТРОЛИ

5

Контролата RadioButton може да показва текст, изображение или и двете. Използвайте свойството Checked, за да получите или зададете състоянието на RadioButton.

Пример:

Ако искаме даден радио бутон да бъде маркиран по подразбиране използваме следния код:

```
radioButton1.Checked = true;
```

I начин:

```
private void button1_Click(object sender, EventArgs e)
{
    if (radioButton1.Checked == true)
    { MessageBox.Show("Избрахме Червено !! "); return; }
    else if (radioButton2.Checked == true)
    { MessageBox.Show("Избрахме Синьо !! "); return; }
    else if (radioButton3.Checked == true)
    { MessageBox.Show("Избрахме Зелено !! "); return; }
    else if (radioButton4.Checked == true)
    { MessageBox.Show("Избрахме Жълто !! "); return; }
    else if (radioButton5.Checked == true)
    { MessageBox.Show("Избрахме Магента !! "); return; }
    else if (radioButton6.Checked == true)
    { MessageBox.Show("Избрахме Кафяво !! "); return; }
}
```

С натискане на бутона се прочита кой радио бутон е избран

☒ Червено

☐ Синьо

☐ Зелено

☐ Жълто

☐ Магента

☐ Кафяво

Read RadioButtons

I. ОСНОВНИ КОНТРОЛИ

5

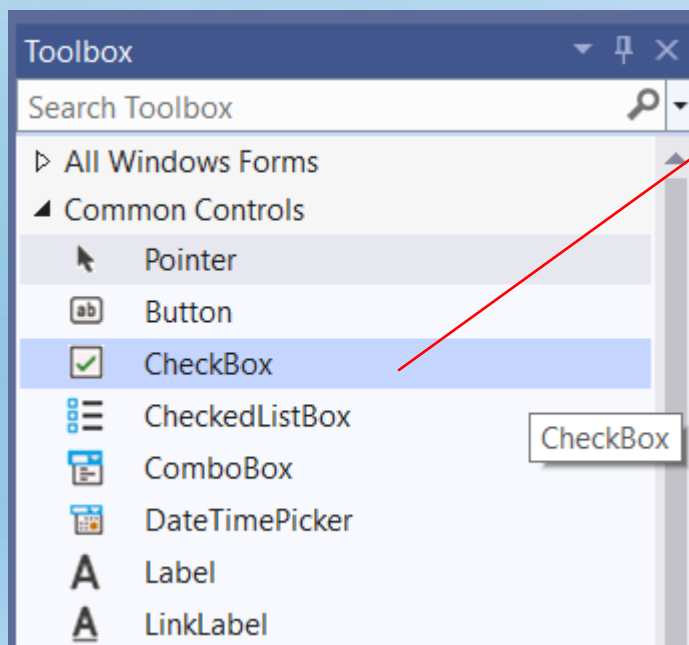
```
RadioButton radioButton = this.panel1.Controls.OfType<RadioButton>()
    .Where(x => x.Checked).FirstOrDefault();
if (radioBtn != null)
{
    switch (radioBtn.Name)
    {
        case "radioButton1":
            MessageBox.Show("Избрахте Червено !! ");
            break;
        case "radioButton2":
            MessageBox.Show("Избрахте Синьо !! ");
            break;
        case "radioButton3":
            MessageBox.Show("Избрахте Зелено !! ");
            break;
        case "radioButton4":
            MessageBox.Show("Избрахте Жълто !! ");
            break;
        case "radioButton5":
            MessageBox.Show("Избрахте Магента !! ");
            break;
        case "radioButton6":
            MessageBox.Show("Избрахте Кафяво !! ");
            break;
    }
}
```

Всеки път при натискане на бутона се създава обект от класа **RadioButton**, на който се присвоява онази контрола (от всички, намиращи се в **panel1**), чието състояние е **Checked** и е зададена като първа или по подразбиране.

II начин:

8. Контрола от типа CheckBox

Квадратчетата за отметка позволяват на потребителя да направи множество селекции от няколко опции. Маркирано квадратче с отметка, дава на потребителя опция, като вярно/невярно или да/не. Можете да щракнете върху квадратче за да поставите отметка, т.е. за да го изберете, или да щракнете отново върху него, за да премахнете отметката.



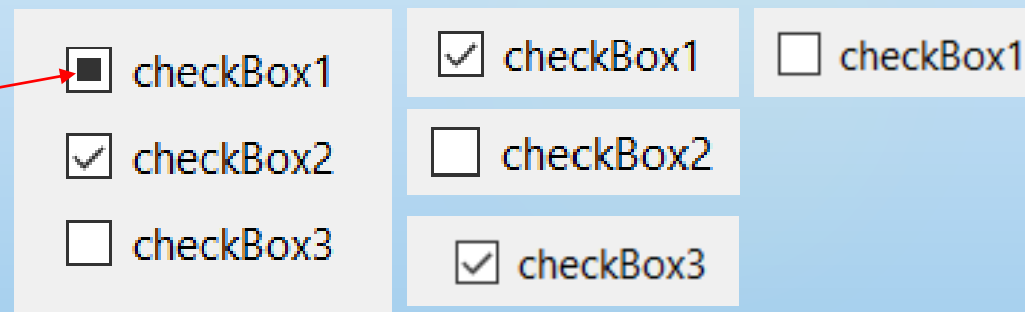
☐ checkBox1
☐ checkBox2

```
private CheckBox checkBox1;  
private CheckBox checkBox2;
```

```
//  
// checkBox1  
//  
this.checkBox1.AutoSize = true;  
this.checkBox1.Location = new System.Drawing.Point(426, 344);  
this.checkBox1.Name = "checkBox1";  
this.checkBox1.Size = new System.Drawing.Size(101, 24);  
this.checkBox1.TabIndex = 33;  
this.checkBox1.Text = "checkBox1";  
this.checkBox1.UseVisualStyleBackColor = true;
```

Можете да използвате свойството `ThreeState` на контролата `CheckBox`, за да насочите контролата да върне стойностите `Checked`, `Unchecked` и `Indeterminate`. Трябва да зададете свойството **`ThreeState`** на **`True`**, за да посочите, че искате да поддържа три състояния.

```
this.checkBox1.ThreeState = true;
```



Радио бутоните и `CheckBox` бутоните се използват за различни функции. Използвайте `RadioButton`, когато искате потребителят да избере само една опция. Когато искате потребителят да избере всички подходящи опции, използвайте `CheckBox`. Следващата програма на C# показва как да откриете дали квадратчето за отметка е избрано или не.

I. ОСНОВНИ КОНТРОЛИ

5

```
private void button2_Click(object sender, EventArgs e)
{
    string msg = "Вие избрахте";

    if (checkBox1.Checked == true)
    {
        msg = "  CheckBox 1";
    }

    if (checkBox2.Checked == true)
    {
        msg = msg + "  CheckBox 2";
    }

    if (checkBox3.Checked == true)
    {
        msg = msg + "  CheckBox 3";
    }

    if (msg.Length > 0)
    {
        MessageBox.Show(msg + " selected ");
    }
    else
    {
        MessageBox.Show("No checkbox selected");
    }

    checkBox1.ThreeState = true;
}
```

Променлива за главно съобщение

Проверки за избран СheckBox

Второстепенни съобщения

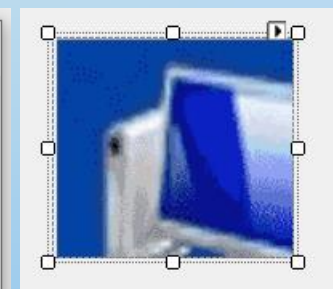
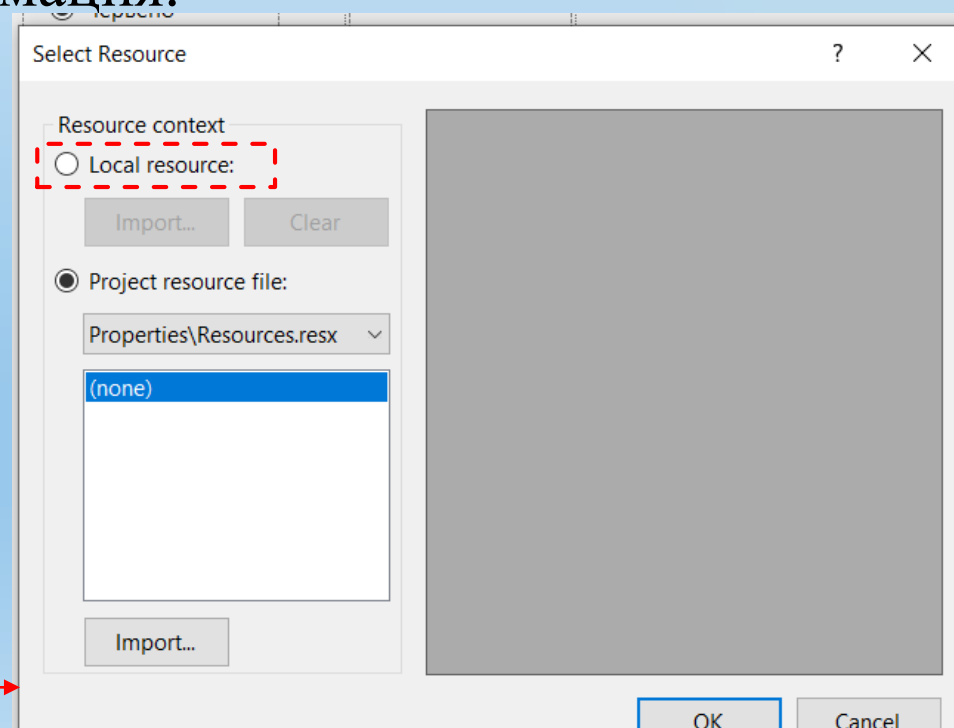
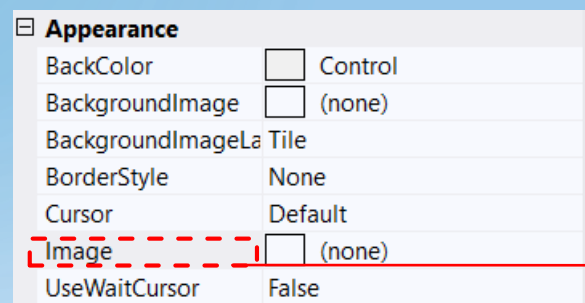
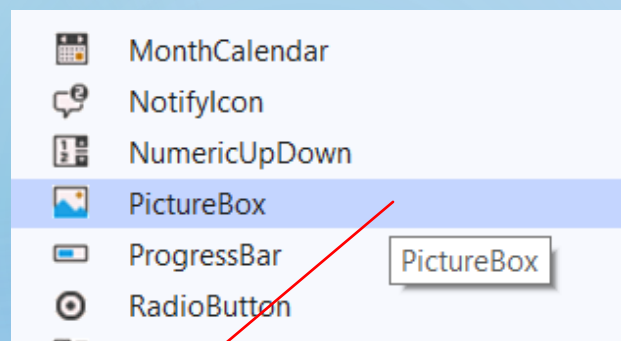
Проверка дали въобще е избран
някой от CheckBox-овете

I. ОСНОВНИ КОНТРОЛИ



9. Контрола от типа PictureBox

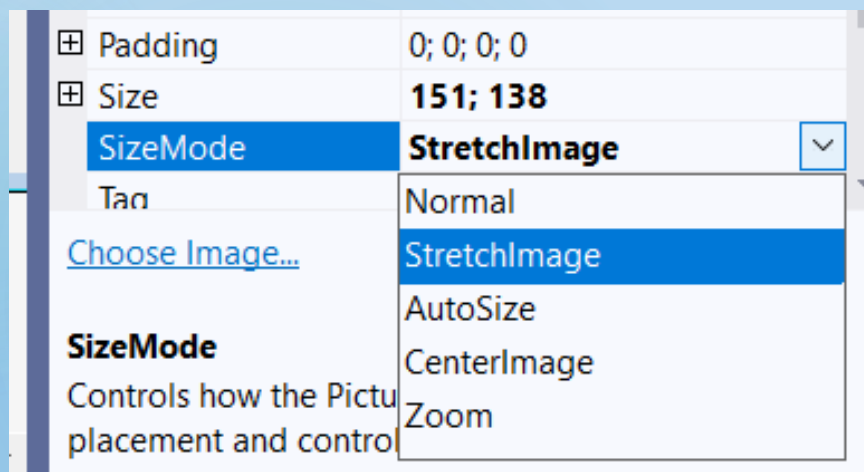
Контролата Windows Forms PictureBox се използва за показване на изображения в растерни, GIF, PNG, икони или JPEG формати. Можете да зададете свойството Image на изображението, което искате да покажете, или по време на проектиране, или по време на изпълнение. Можете програмно да промените изображението, показано в полето за картина, което е особено полезно, когато използвате един формуляр за показване на различни части от информация.



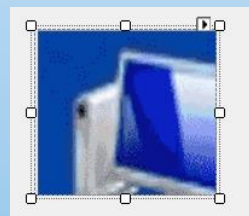
I. ОСНОВНИ КОНТРОЛИ

5

Когато работите с изображения, винаги обръщайте внимание на размера на контролата PictureBox и размера на изображението, което искате да използвате !!! Предварително преоразмерявайте използваното изображение, за да се побира в PictureBox-а или променяйте размера на изображение в хода на програмата т.е. динамично при самото зареждане на програмата.



SizeMode Normal



SizeMode **StretchImage**



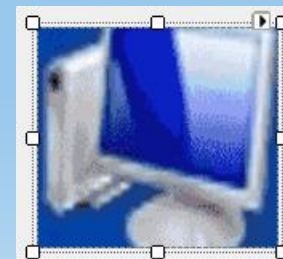
SizeMode **Zoom**



SizeMode **AutoSize**



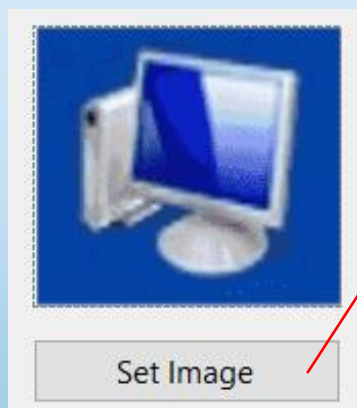
SizeMode **CenterImage**



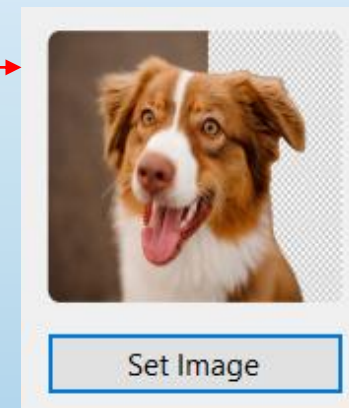
I. ОСНОВНИ КОНТРОЛИ



Динамично зареждане на изображение:



```
private void button3_Click(object sender, EventArgs e)
{
    pictureBox1.Image = Image.FromFile("C:\\Users\\user\\Pictures\\img2.png");
}
```



Динамично може да се сменя стойността на свойството **SizeMode**:

```
pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
```

AutoSize - Оразмерява полето с картина спрямо изображението;

CenterImage – Центрира изображението в полето за картина;

Normal – Постава горния ляв ъгъл на изображението в горния ляв ъгъл в полето за картина;

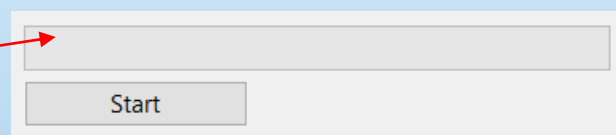
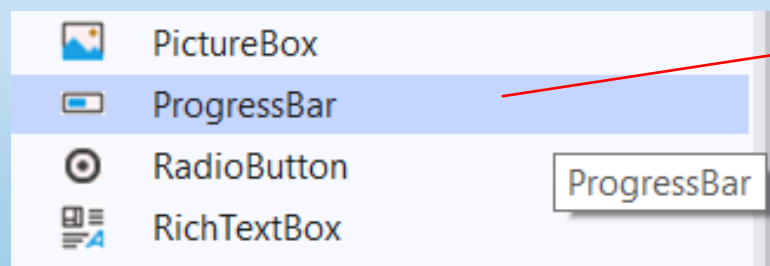
StretchImage - Позволява ви да разтегнете изображението спрямо размера на контейнера, в който се намира

I. ОСНОВНИ КОНТРОЛИ



10. Контрола от типа ProgressBar

Лентата за напредъка (ProgressBar) е контрола, която приложението може да използва, за да покаже напредъка на дълга операция, като например изчисляване на сложен резултат, изтегляне на голям файл от мрежата и т.н.



```
private ProgressBar progressBar1;
```

Контролите ProgressBar се използват винаги, когато дадена операция отнема повече от кратък период от време. Свойствата "Maximum" и "Minimum" определят диапазона от стойности за представяне на напредъка на дадена задача.

- **Минимум:** Задава по-ниската стойност за диапазона от валидни стойности за напредък.
- **Максимум :** Задава горната стойност за диапазона от валидни стойности за напредък.
- **Стойност:** Това свойство получава или задава текущото ниво на напредък. По подразбиране Минимумът и Максимумът са зададени на 0 и 100. Докато задачата продължава, лентата за напредъка се запълва отляво надясно.

I. ОСНОВНИ КОНТРОЛИ



Пример за използване на ProgressBar:

```
1 reference
private void button4_Click(object sender, EventArgs e)
{
    int i;

    progressBar1.Minimum = 0;
    progressBar1.Maximum = 200;

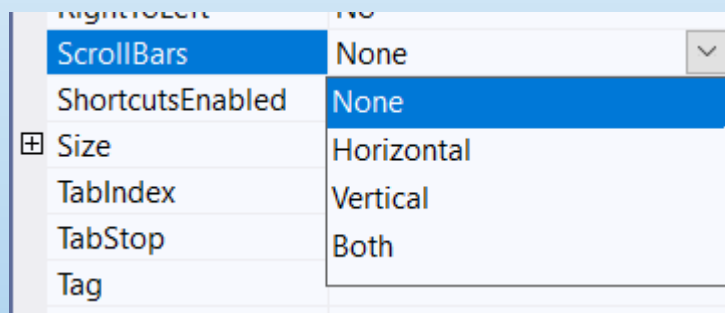
    for (i = 0; i <= 200; i++)
    {
        progressBar1.Value = i;
    }
}
```

I. ОСНОВНИ КОНТРОЛИ



11. Контрола от типа ScrollBar

Лентата за превъртане ви позволява да преглеждате съдържание, което е извън текущата зона за гледане, като плъзнете плъзгача, за да направите съдържанието ВИДИМО.

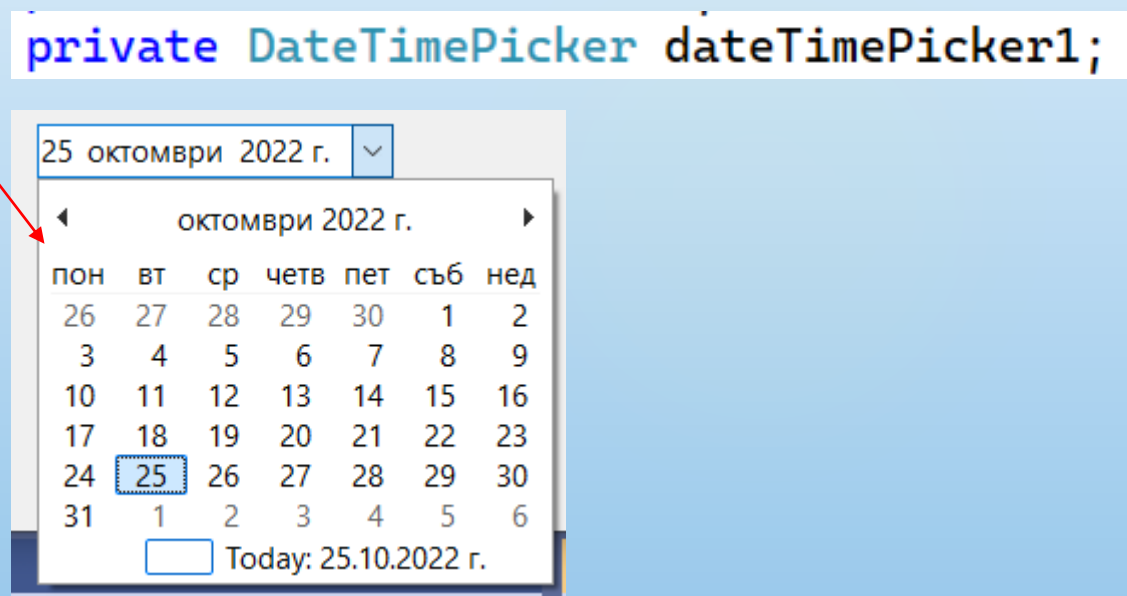
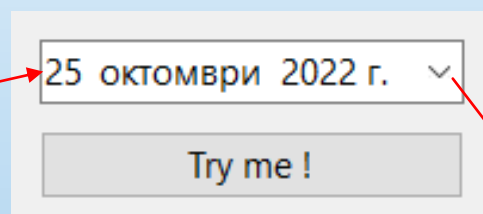
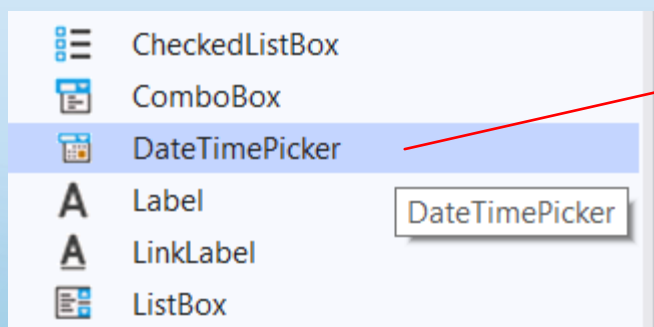


Контролата ScrollBar съдържа контрола Track. Контролът Track се състои от контрол Thumb и два бутона RepeatButton. Можете да увеличите и намалите свойството Value на контролата ScrollBar, като натиснете контролите RepeatButton или като преместите Thumb. Можете сами да зададете свойството Value в код, който премества полето за превъртане,. Свойствата Минимум и Максимум определят диапазона от стойности, които контролата може да покаже. Диапазонът от стойности по подразбиране за свойството Стойност е от 0 до 1.

I. ОСНОВНИ КОНТРОЛИ

12. Контрола от типа DateTimePicker

Контролата DateTimePicker ви позволява да показвате и събирате информация за дата и час от потребителя с определен формат.



```
//  
// dateTimePicker1  
//  
this.dateTimePicker1.Location = new System.Drawing.Point(618, 413);  
this.dateTimePicker1.Name = "dateTimePicker1";  
this.dateTimePicker1.Size = new System.Drawing.Size(178, 27);  
this.dateTimePicker1.TabIndex = 43;
```

I. ОСНОВНИ КОНТРОЛИ



Контролата DateTimePicker има две части, етикет, който показва избраната дата, и изскачащ календар, който позволява на потребителите да избират нова дата. Най-важното свойство на DateTimePicker е свойството **Value**, което съдържа избраните дата и час.

```
dateTimePicker1.Value = DateTime.Today;
```

Свойството Value съдържа текущата дата и час, на които е зададена контролата. Можете да използвате свойството Text или съответния член на Value, за да получите стойността за дата и час.

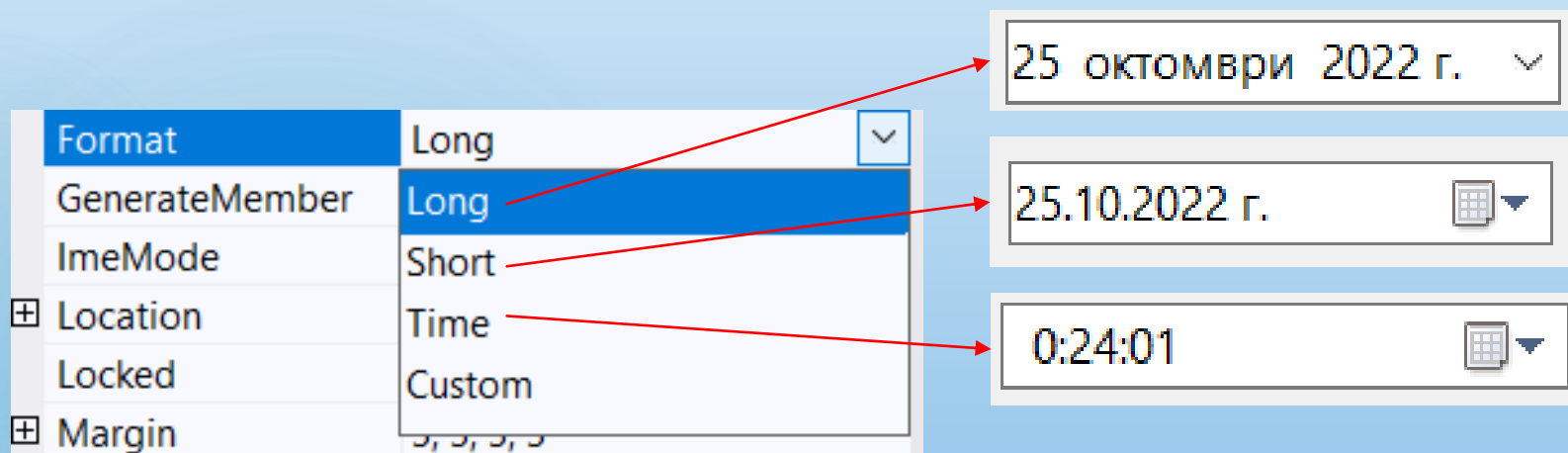
```
DateTime iDate;  
iDate = dateTimePicker1.Value;
```

```
1 reference  
private void button5_Click(object sender, EventArgs e)  
{  
    DateTime iDate;  
    iDate = dateTimePicker1.Value;  
    MessageBox.Show("Selected date is " + iDate);  
}
```

I. ОСНОВНИ КОНТРОЛИ



Контролата може да показва един от няколко стила в зависимост от стойностите на нейните свойства. Стойностите могат да бъдат показани в четири формата, които се задават от свойството Format: Long, Short, Time или Custom.



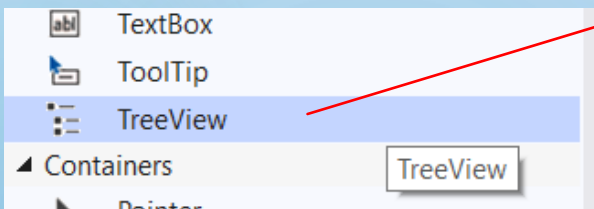
```
dateTimePicker1.Format = DateTimePickerFormat.Short;
```


I. ОСНОВНИ КОНТРОЛИ



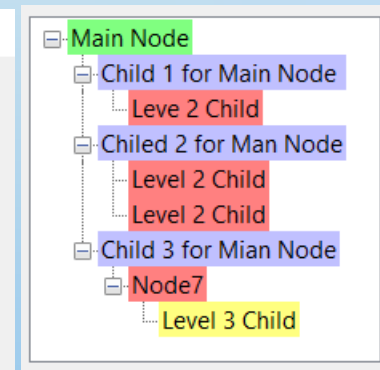
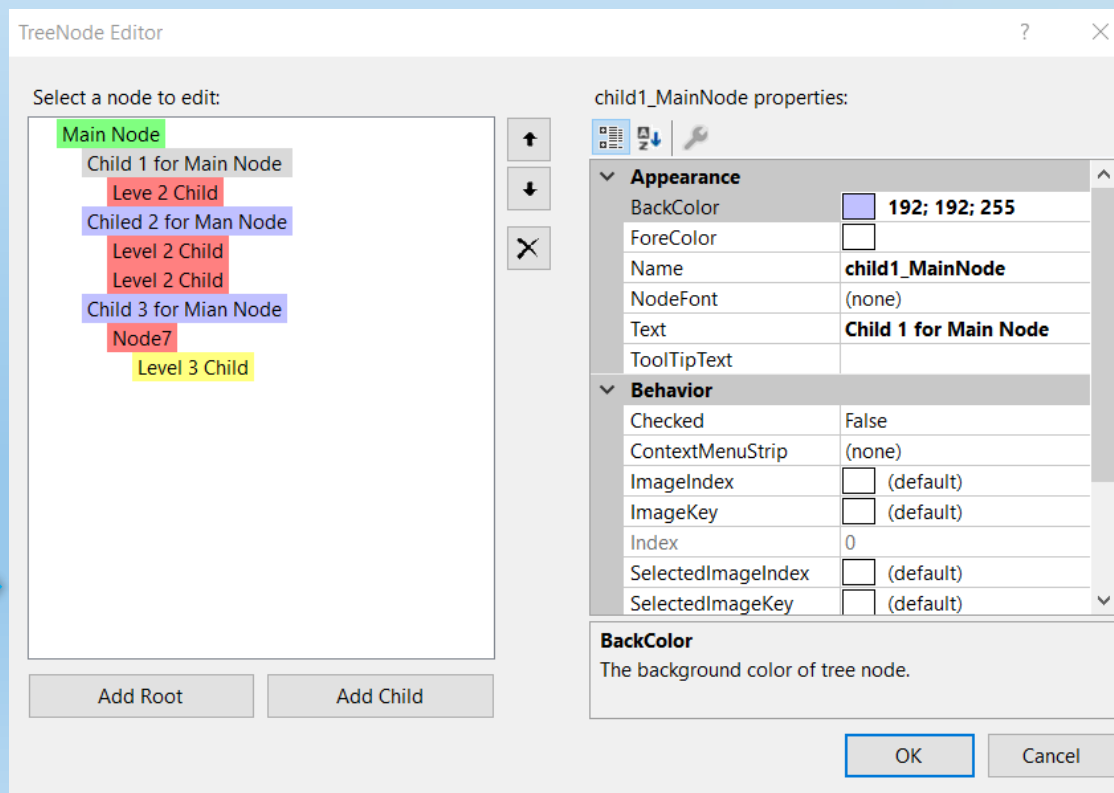
13. Контрола от типа Treeview

Контролата TreeView съдържа йерархия от контроли TreeViewItem. Той предоставя начин за показване на информация в йерархична структура чрез използване на “сгъваеми” възли. Най-горното ниво в дървовидния изглед са кореновите възли, които могат да бъдат разширени или свити, ако възлите имат дъщерни възли.



```
private TreeView treeView1;
```

Nodes (Collection)



I. ОСНОВНИ КОНТРОЛИ

5

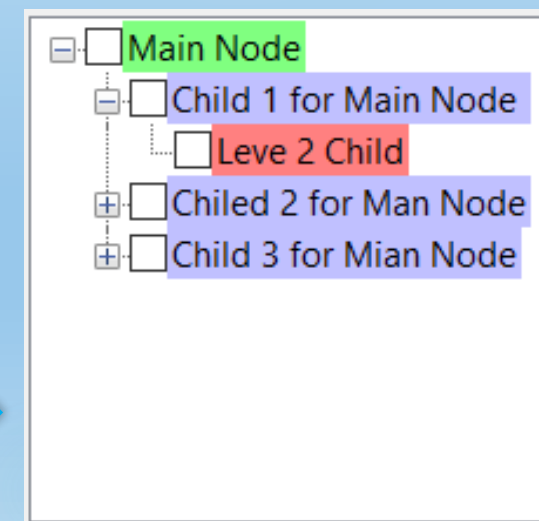
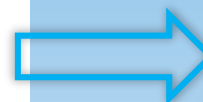
Можете изрично да дефинирате съдържанието на `TreeView` или някакъв източник на данни може да предостави съдържанието. Потребителят може да разшири `TreeNode`, като щракне върху бутона със знак плюс (+), ако такъв се показва до `TreeNode`, или можете да разширите `TreeNode`, като извикате метода `TreeNode.Expand`. Можете също така да навигирате през дървовидни изгледи с различни **свойства**: **`FirstNode`**, **`LastNode`**, **`NextNode`**, **`PrevNode`**, **`NextVisibleNode`**, **`PrevVisibleNode`**.

Методът **`fullpath`** на контрола на дървовидния изглед осигурява пътя от основния възел до избрания възел:

```
treeView1.SelectedNode.FullPath.ToString();
```

Възлите на дървото могат по избор да показват квадратчета за отметка. За да покажете квадратчетата за отметка, задайте свойството **`CheckBoxes`** на **`TreeView`** на `true`.

```
treeView1.CheckBoxes = true;
```



I. ОСНОВНИ КОНТРОЛИ



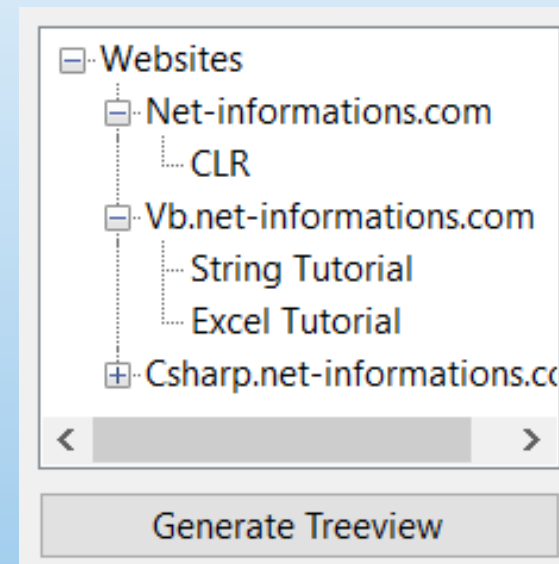
Динамично създаване на дървовидна структура:

```
0 references
private void Form1_Load(object sender, EventArgs e)
{
    TreeNode tNode;
    tNode = treeView1.Nodes.Add("Websites");

    treeView1.Nodes[0].Nodes.Add("Net-informations.com");
    treeView1.Nodes[0].Nodes[0].Nodes.Add("CLR");

    treeView1.Nodes[0].Nodes.Add("Vb.net-informations.com");
    treeView1.Nodes[0].Nodes[1].Nodes.Add("String Tutorial");
    treeView1.Nodes[0].Nodes[1].Nodes.Add("Excel Tutorial");

    treeView1.Nodes[0].Nodes.Add("Csharp.net-informations.com");
    treeView1.Nodes[0].Nodes[2].Nodes.Add("ADO.NET");
    treeView1.Nodes[0].Nodes[2].Nodes[0].Nodes.Add("Dataset");
}
```

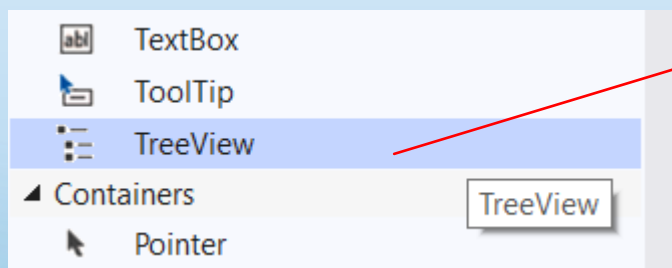


I. ОСНОВНИ КОНТРОЛИ



15. Контрола от типа ListView

Контролата ListView е ItemsControl, която е извлечена от ListBox.



```
private ListView listView1;
```

```
listView1  
  
this.listView1.Columns.AddRange(new System.Windows.Forms.ColumnHeader[] {  
    this.name,  
    this.address,  
    this.id_number,  
    this.telephone});  
this.listView1.Items.AddRange(new System.Windows.Forms.ListViewItem[] {  
    listViewItem1,  
    listViewItem2,  
    listViewItem3,  
    listViewItem4});  
this.listView1.Location = new System.Drawing.Point(924, 11);  
this.listView1.Margin = new System.Windows.Forms.Padding(3, 2, 3, 2);  
this.listView1.Name = "listView1";  
this.listView1.Size = new System.Drawing.Size(333, 95);  
this.listView1.TabIndex = 49;  
this.listView1.UseCompatibleStateImageBehavior = false;  
this.listView1.View = System.Windows.Forms.View.Details;
```

I. ОСНОВНИ КОНТРОЛИ





По-важни свойства на контролата ListView

HeaderStyle	Clickable
HideSelection	None
HotTracking	Nonclickable
HoverSelection	Clickable
ImeMode	NoControl

Columns	(Collection)
---------	--------------

Dock	None
------	------

BackColor	 128; 255; 128
BackgroundImage	 (none)

Font	Segoe UI; 9pt
Name	ab Segoe UI
Size	9
Unit	Point
Bold	False
GdiCharSet	1
GdiVerticalFont	False
Italic	False
Strikeout	False
Underline	False
ForeColor	 WindowText

Name	Address	ID_Number	Telephone

CheckBoxes	True
------------	------

Name	Address	ID_Number	Telephone
<input type="checkbox"/>			
<input type="checkbox"/>			
<input type="checkbox"/>			
<input type="checkbox"/>			

GridLines	True
-----------	------



Name	Address	ID_Number	Telephone
<input type="checkbox"/>			
<input type="checkbox"/>			
<input type="checkbox"/>			
<input type="checkbox"/>			

MultiSelect	True
-------------	------

Scrollable	True
------------	------

Size	333; 127
Width	333
Height	127

Sorting	None
---------	------

I. ОСНОВНИ КОНТРОЛИ



Статично добавяне на колони и записи в ListView

Columns (Collection)



ColumnHeader Collection Editor

Members:

0	name
1	address
2	id_number
3	telephone

↑

↓

Add Remove

name properties:

Behavior

DisplayIndex 0

Data

Tag

Design

(Name) name

GenerateMember True

Modifiers Private

Misc

ImageIndex ☐ (none)

ImageKey ☐ (none)

Text Name

TextAlign Left

Width 60

OK Cancel

View	Details
------	---------

Тази настройка позволява да се виждат заглавията на колоните

Name	Address	ID_Number	Telephone
------	---------	-----------	-----------

I. ОСНОВНИ КОНТРОЛИ



Четене на избрания елемент (ред/запис) от ListView

```
1 reference
private void button8_Click(object sender, EventArgs e)
{
    string name = null;
    string address = null;
    string id_number = null;
    string telephone = null;
    string productName = null;
    string product_id = null;
    string quantity = null;

    name = listView1.SelectedItems[0].SubItems[1].Text;
    address = listView1.SelectedItems[0].SubItems[1].Text;
    id_number = listView1.SelectedItems[0].SubItems[2].Text;
    telephone = listView1.SelectedItems[0].SubItems[3].Text;
    productName = listView1.SelectedItems[0].SubItems[4].Text;
    product_id = listView1.SelectedItems[0].SubItems[5].Text;
    quantity = listView1.SelectedItems[0].SubItems[6].Text;

    MessageBox.Show(name + " , " , " , " + address + " , " + id_number+ " , " +
        telephone+ " , " + productName+ " , " + product_id+ " , " + quantity);
}
```