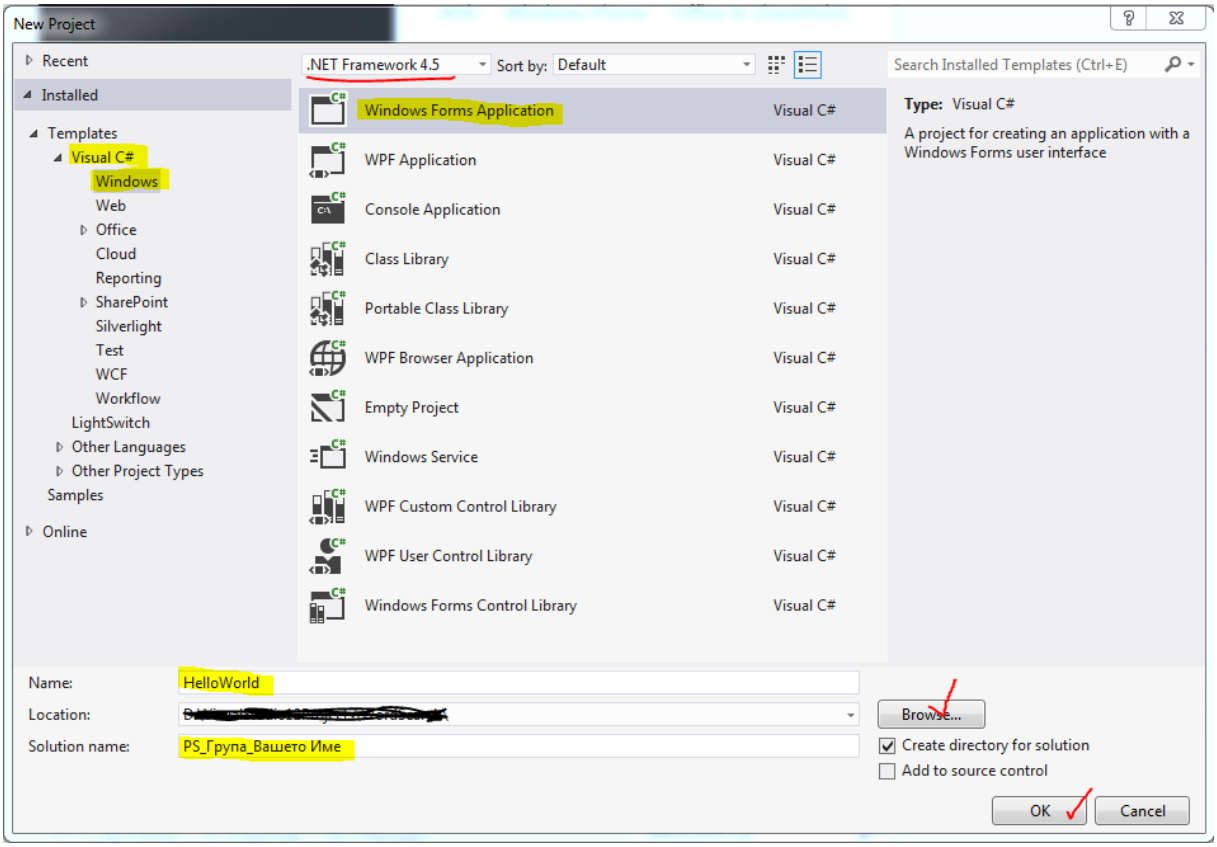
Упражнение №1 по ПС

Въведение в десктоп програмирането. Windows Forms. Контроли и събития.

Целта на това упражнение е студентите да се запознаят технологията Windows Forms, да направят връзките между програмирането за Windows и изучаваното досега по ООП. Разглеждат се основополагащи термини като контроли и събития, прави се начална обработка на информация въведена от потребителя. Разглеждат се начините за отваряне на нови прозорци.

Създаване на нов проект.

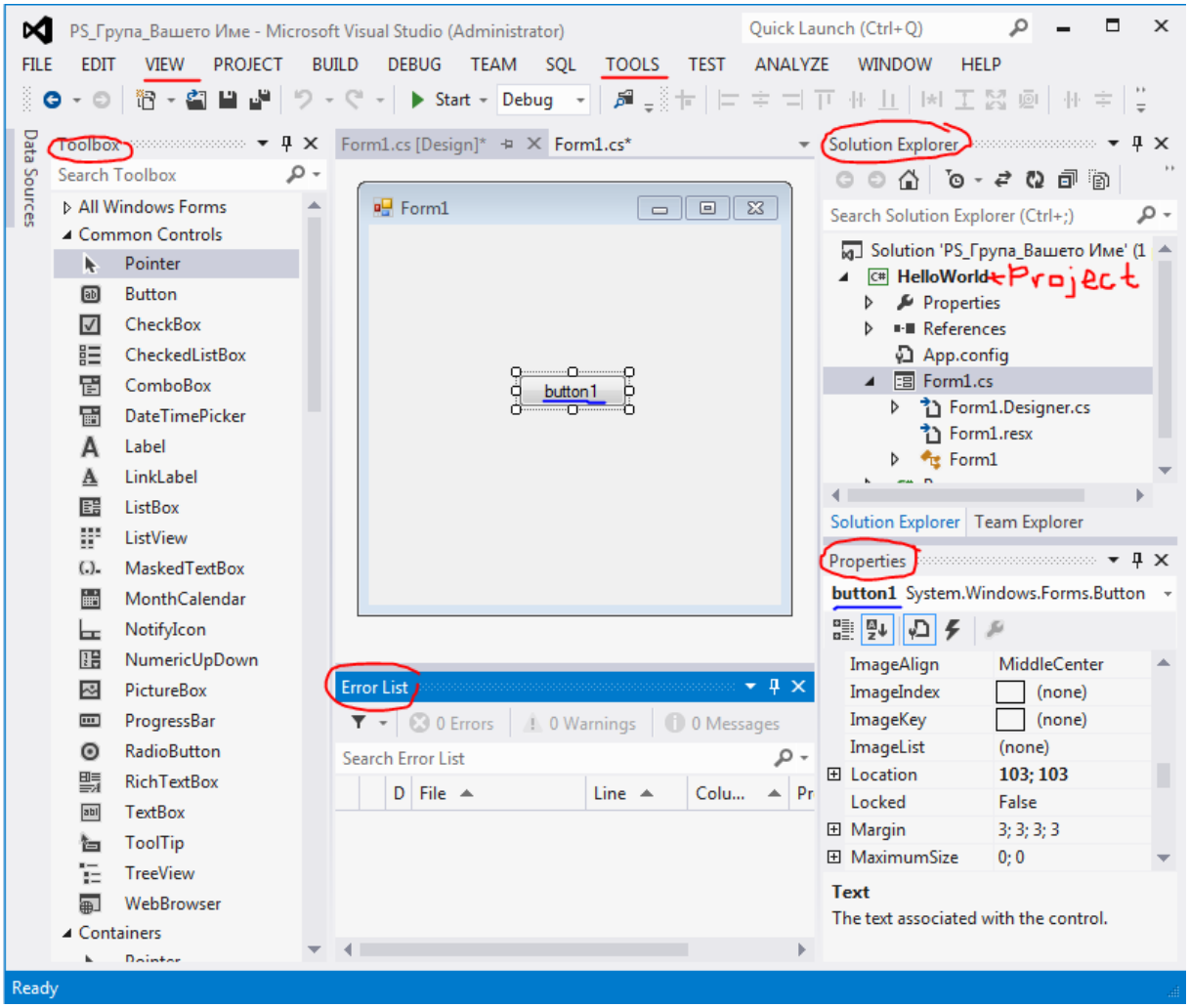
За да създадете нов проект отворете Visual Studio 2012 и от менюто “File” изберете “New” → “Project…”. Отваря се следния диалогов прозорец:



Езикът, с който ще работим, е C#. Ще създаваме приложение с типичните за Windows форми (Windows Forms Application). Кръстете вашия проект HelloWorld, но след това задайте друго име за Solution. Нека името представлява PS_<№ на група>_<име>, например **PS_74_Toni**. В Location изберете D:/PS/<номер на вашата група> и запомнете това място, тъй като от тук нататък всички часове ще работите само с този ваш Solution. Препоръчително е да си носите собствени USB флаш памети, на които да си записвате проектите, за да е сигурно, че можете да си ги доработвате.

Прозорци във Visual Studio 2012

След като е отворен един проект вашето Visual Studio 2012 трябва да има приблизително този вид:



Вашата HelloWorld програмка

Добавете един бутон към прозореца на вашата програма. Кръстете го **btnHello** и променете текста му на „**Здрасти!**”.

Натиснете двойно върху бутона, по този начин създаваме неговия OnClick метод.

В новосъздадения метод добавете следния код:

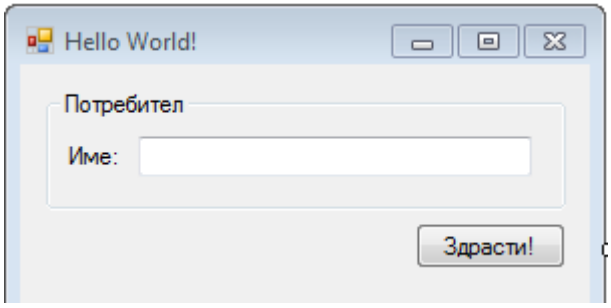
```
MessageBox.Show("Здрасти!!! Това е твоята първа програма
                                    на Visual Studio 2012!");
```

Стартирайте програмата с **F5** или зеления триъгълник.

Добавете GroupBox контрола от групата Containers, която ще обединява всички контроли с въвеждане на данните за един потребител. Дайте ѝ име gbUser и променете текста ѝ.

В горната контрола добавете Label и TextBox контроли. Именувайте ги подходящо (например lblName и txtName).

Прозореца ви трябва да изглежда по подобен начин:



Модифицирайте кода на бутона btnHello по следния начин:

```
MessageBox.Show("Здрасти "+ txtName.Text + "!!! \nТова е твоята първа програма на Visual Studio 2012!");
```

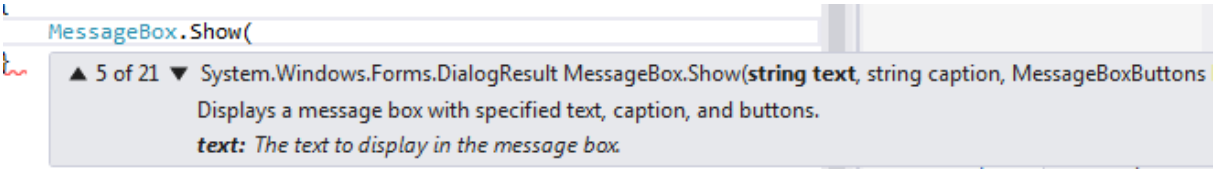
Стартирайте и наблюдавайте разликата в поведението като въведете текст в полето.

Направете проверка за дължината на текста, въведен от потребителя. Поставете ограничение за дължина на името поне 2 символа. Направете, така че само ако въведената информация отговаря на условията, да се извежда съобщението за поздрав, а в противен случай изведете съобщение с подкана за корекция. (Използвайте оператора “If-else”).

Добавите необходимите компоненти и код за пресмятане на n!, където n се въвежда от потребителя.

Добавете метод, който се изпълнява при изход от програмата и проверява дали потребителя е сигурен дали иска да я затвори.

Подсказки:

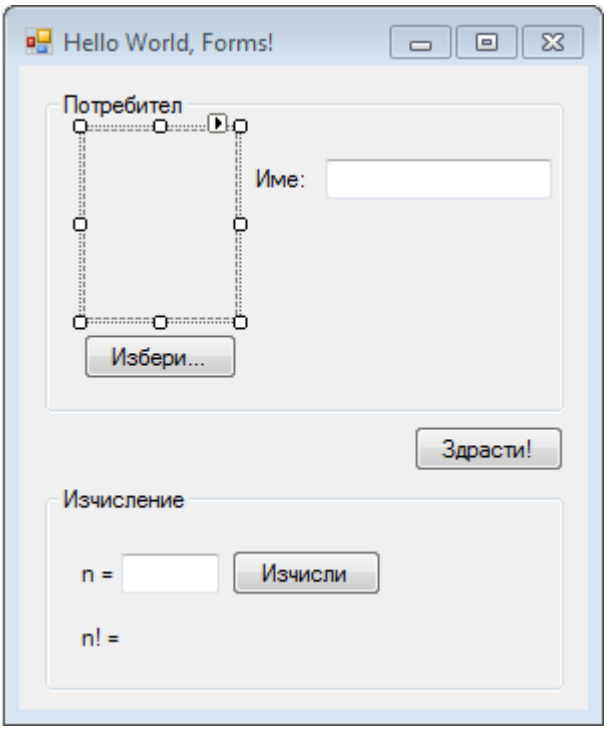


- Намерете правилното събитие на формата, което отговаря за затварянето на програмата.
- Разгледайте възможностите на MessageBox.Show метода. Той има 21 различни имплементации, всяка с различен на брой и вид параметри. Освен текст можете да показвате различни икони, текстове и бутони. Методът НЕ е от тип void, така че вижте какво ви връща за да разберете кой бутон е натиснал вашия потребител.

Стандартни диалогови прозорци.

OpenFileDialog

Добавете необходимите елементи и код, за да позволите на вашия потребител да избере изображение от компютъра си и да го покаже в програмата. Следвайте стъпките по-долу.

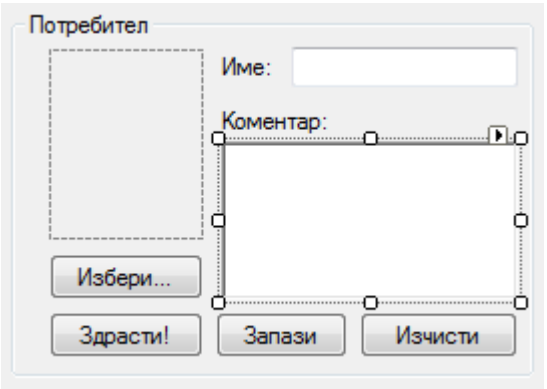


1. Добавете към във формата контрола от тип **PictureBox**, кръстете я **pbAvatar**. Добавете и един бутон с име **btnChooseImage**.
2. Добавете към вашата Form1 един **OpenFileDialog**. Преименувайте го на **openPictureDialog**.
3. Едно от свойствата позволява да направите филтър на файловете. За да можете да избирате само изображения, задайте на свойството Filter стойност **Image Files(*.JPG;*.GIF)|*.JPG;*.GIF| All files (*.*)|*.***
4. Създайте Click метод на **btnChooseImage** и в него добавете следния код:

```
if (openPictureDialog.ShowDialog() == DialogResult.OK)
{
    pbAvatar.ImageLocation = openPictureDialog.FileName;
}
```

5. Тествайте дали работи правилно.
6. Ще забележите, че ако изображението е много голямо, ще се види само част от него. Можете да промените свойството **Size Mode** на **pbAvatar** на **Zoom**, което ще смали изображението без да го разтегля по X или Y.

SaveFileDialog



Добавете необходимите елементи и код, за да позволите на вашия потребител да напише текстов коментар и да го съхрани в компютъра си. Следвайте стъпките по-долу.

1. Добавете към във формата още една контрола от тип **RichTextBox**, кръстете я **richTxtComment**. Добавете и един бутон с име **btnSave**.
2. Добавете към вашата Form1 диалог за запаметяване на файл - **saveFileDialog**.
3. Генерирайте Click метода на бутона **btnSave**, добавете долния код и тествайте.

```
private void btnSave_Click(object sender, EventArgs e)
{
    saveFileDialog.DefaultExt = "*.rtf";
    saveFileDialog.Filter = "RTF Files|*.rtf";
    if (saveFileDialog.ShowDialog() == System.Windows.Forms.DialogResult.OK
                                    && (saveFileDialog.FileName.Length > 0))
    {
       try
       {
          // Save the contents of the RichTextBox into the file.
          richTxtComment.SaveFile(saveFileDialog.FileName);
       }
       catch (Exception)
       {
          MessageBox.Show("Файлът не беше съхранен, поради грешка в записа!",
          "Грешка", MessageBoxButtons.OK, MessageBoxIcon.Error);
       }
    }
}
```

FontDialog и ColorDialog

Задача за самостоятелна работа в къщи:

Разгледайте възможностите на контролата RichTextBox за форматиране на текста в нея. Разгледайте и тествайте останалите стандартни диалогови прозорци, а именно тези за избор на шрифт и цвят.

Пример за смяна на шрифта в RichTextBox

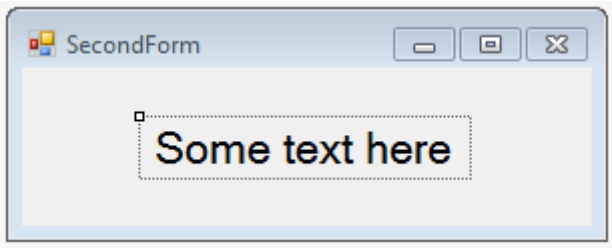
```
private void btnFont_Click(object sender, EventArgs e)
{
    if (fontDialog.ShowDialog() == DialogResult.OK)
    {
        if (richTxtComment.SelectedText == "")
        {
            richTxtComment.Font = fontDialog.Font;
        }
        else
        {
            richTxtComment.SelectionFont = fontDialog.Font;
        }
    }
}
```

Модални и немодални форми.

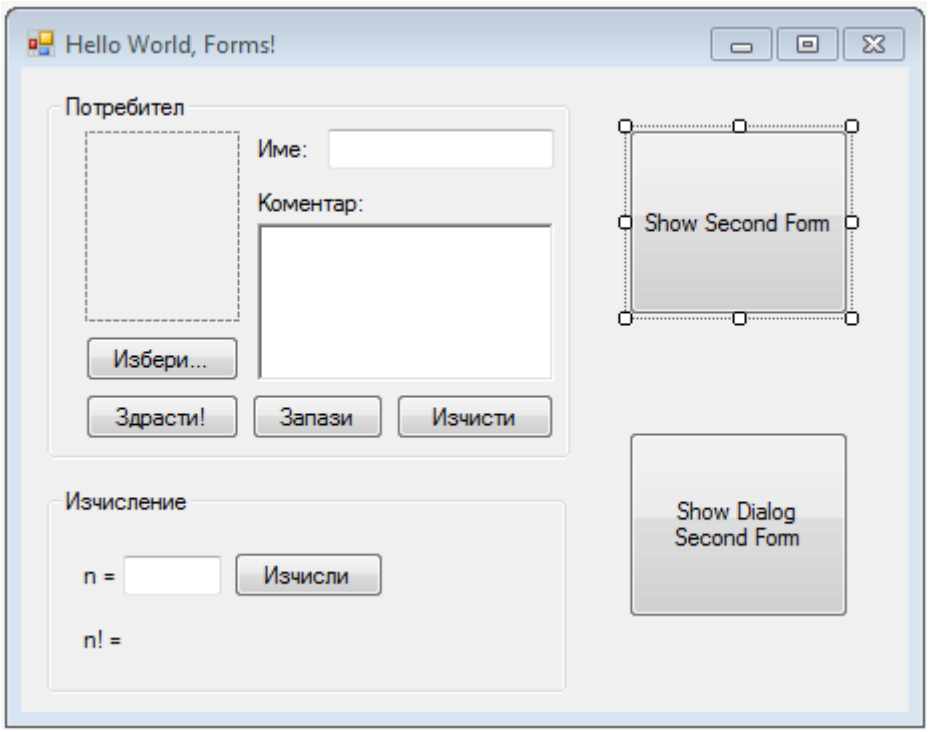
Създаване на собствена форма

Добавете нова форма (прозорец) към проекта и изследвайте начините за нейното отваряне, както и предаването на информация към нея и получаването на резултат от нея. Следвайте стъпките по-долу.

1. В Solution Explorer, кликнете с десен бутон върху името на вашия проект (не бъркайте проекти и Solution!). От контекстното меню изберете Add… → Windows Form… . Задайте име на формата **SecondForm** и натиснете бутона Add.
2. Добавете в нея един label - **lblGreetingText**, и въведете в него някакъв текст.



3. Във първата форма добавете още 2 бутона за целите на теста – **btnShow** и **btnShowDialog.**



4. Генерирайте Click метода на бутона **btnShow**, добавете долния код и тествайте.

```
private void btnShow_Click(object sender, EventArgs e)
{
    SecondForm secondForm = new SecondForm();
    secondForm.Show();
}
```

* *Обърнете внимание, че горния код представлява създаване на обект от класа* ***SecondForm*** *и извикване на метод с този обект. За разлика от статичния метод на класа MessageBox, където нямахме обект, този метод* ***Show***() *не е статичен*!

Ако искаме да предадем към втората форма въведеното от потребителя име, можем да създадем конструктор с параметри на SecondForm, с който още при създаването на обекта, можем да го инициализираме с необходимите данни.

5. В класа SecondForm (файла SecondForm.cs) добавете следния конструктор:

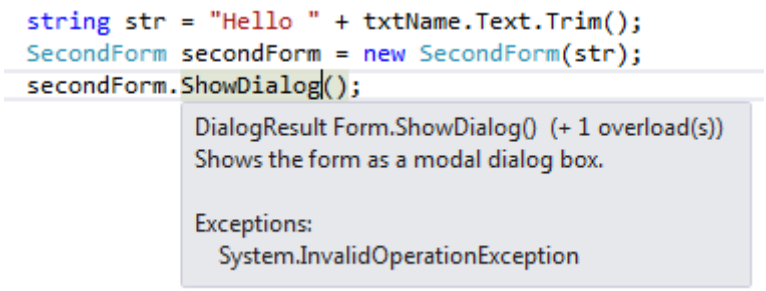
```
public SecondForm(string labelText):this()
{
    lblGreetingText.Text = labelText;
}
```

* *Защо е необходимо да се използва* **:this()** *? Когато правите собствени конструктори на формите, не забравяйте, че трябва да извикате инициализациите в конструктора по подразбиране. Методът* ***InitializeComponent()*** *винаги трябва да е на първо място в конструктора, освен ако не правите това извикване с* ***:this().***

6. Модифицирайте метода btnShow_Click като подадете на конструктора някакъв текст.

```
private void btnShow_Click(object sender, EventArgs e)
{
    //SecondForm secondForm = new SecondForm();
    string str = "Hello " + txtName.Text.Trim();
    SecondForm secondForm = new SecondForm(str);
    secondForm.Show();
}
```

7. Генерирайте Click метода на бутона **btnShowDialog** и копирайте кода за отваряне на втората форма в него. Заменете метода Show с ShowDialog(). Наблюдавайте разликите в поведението на програмата.



Забележете, че докато Show() беше от тип void, ShowDialog() връща стойност от тип DialogResult, подобно на разгледания по-горе MessageBox.Show(). Това означава, че първата форма ще чака резултат от втората и ние можем да го обработим аналогично.

8. В **SecondForm** добавете бутон **btnDialogResultYes**. Настройте свойството му **DialogResult** с **Yes**.
9. В метода **btnShowDialog_Click** на първата форма добавете проверката дали върнатият резултат от втората форма е **Yes**, и изведете подходящи съобщения на потребителя.

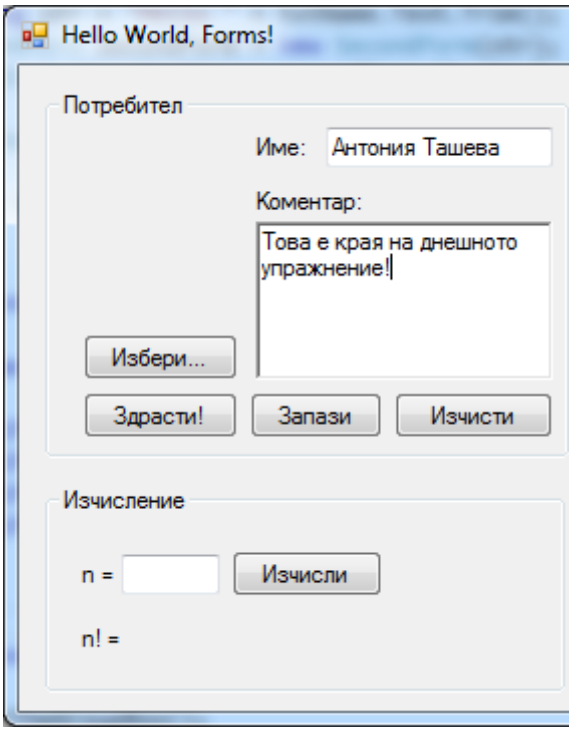
```
private void btnShowDialog_Click(object sender, EventArgs e)
{
    string str = "Hello " + txtName.Text.Trim();
    SecondForm secondForm = new SecondForm(str);
    if (DialogResult.Yes == secondForm.ShowDialog())
    {
        MessageBox.Show("Вие натиснахте бутона Yes");
    }
    else MessageBox.Show("Вие затворихте прозореца без да натиснете бутона Yes");
}
```

Динамично обхождане на елементи в контейнер.

Ако ни се налага многократно да правим едни и същи операции с подобни контроли е добра идея да напишем оптимизиран код, работещ независимо колко на брой са те, а не да пишем повтарящи се аналогични редове за отделните контроли.
По долу е даден пример как да обходим всички контроли и да изчистим/променим тяхното съдържание. Можете да си направите експеримент като добавите още TextBox-и, че те ще се изчистват без да сте добавили нито един ред код.
Тук ключова е идеята, че формата или groupBox-а се разглеждат като контейнери, съдържащи всички елементи в тях. Така можем да ги обходим с 1 цикъл и в зависимост от типа им да направим нещо с тях. Controls е колекцията от всички тези елементи.

```
private void ClearTheGroupBox()
{
    foreach (var item in gbUser.Controls)
    {
        if (item is TextBox) ((TextBox)item).Clear();
        if (item is RichTextBox) ((RichTextBox)item).Clear();
        if (item is Label) ((Label)item).Text = "...";
        if (item is Button) ((Button)item).BackColor = Color.Aqua;
    }
}
```

Преди:



След изпълнение на кода: