

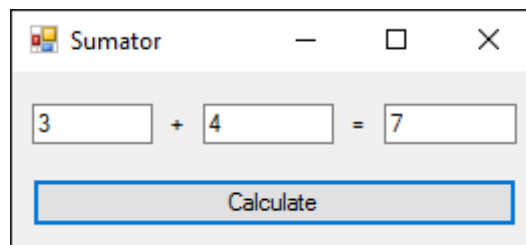
# Упражнения: Графични приложения с Winforms

## 1. Суматор за числа

Да се напише графично (GUI) приложение, което изчислява сумата на две числа. При въвеждане на две числа в първите две текстови полета и натискане на бутона [Calculate] се изчислява тяхната сума и резултатът се показва в третото текстово поле.

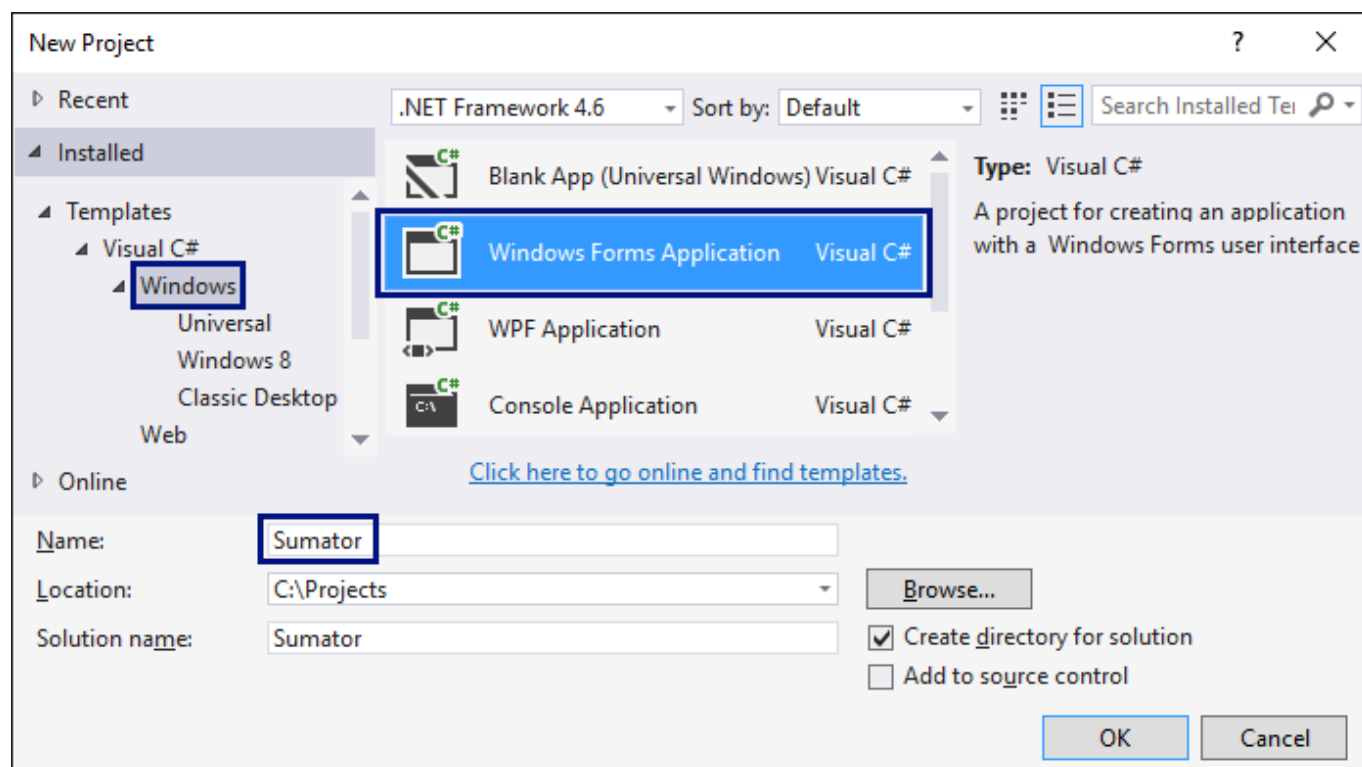
### Решение

В рамките на това упражнение ще направим просто приложение, в което ще използваме Winforms.



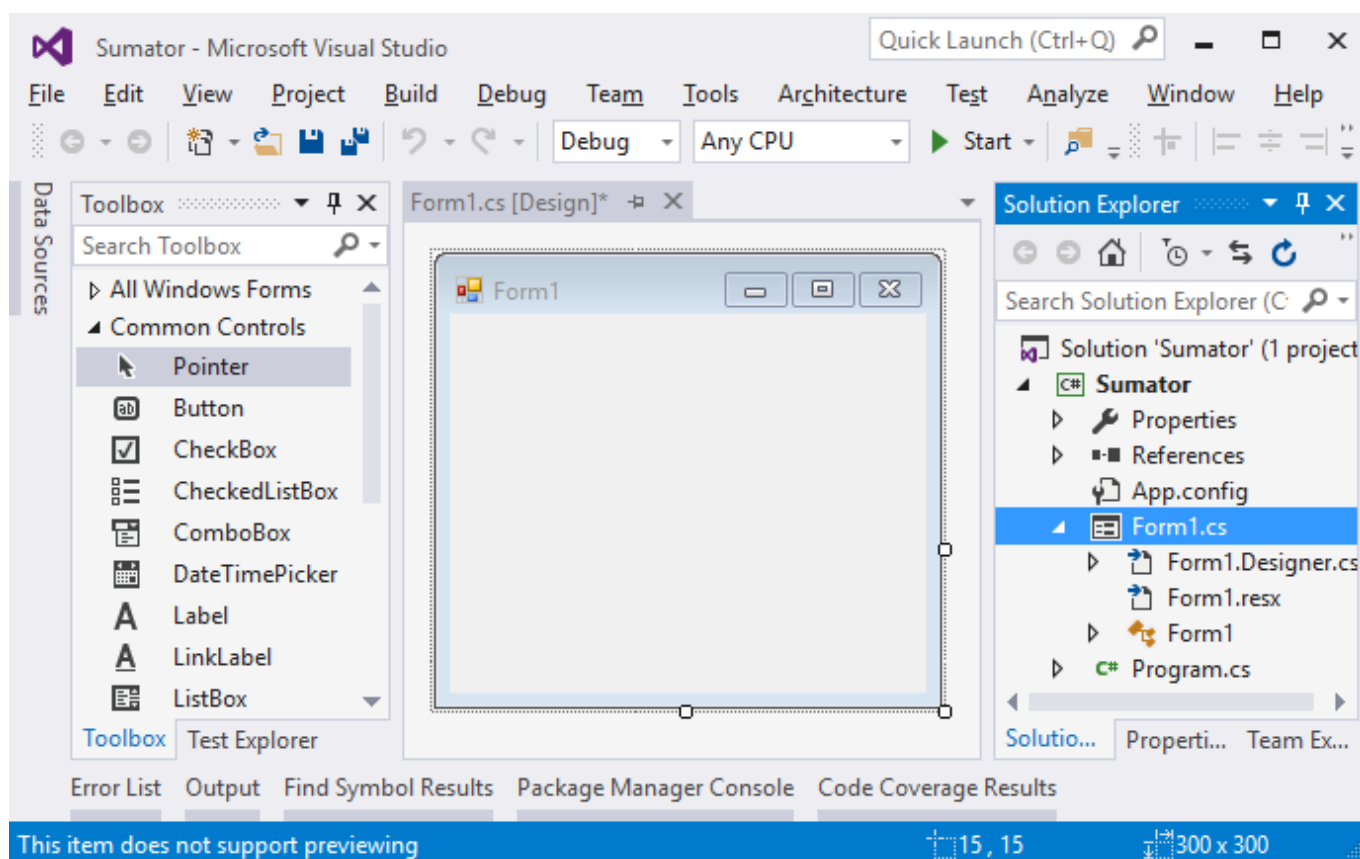
### Стъпка 1. Създаване на проект за графично приложение

Започнете със създаване на **Windows Forms Application** проект **Sumator**.



## Стъпка 2. Добавяне на графичните контроли

При създаването на Windows Forms приложение ще се появи **редактор за потребителски интерфейс**, в който могат да се слагат **различни визуални елементи** (например кутийки с текст и бутони):



Добавянето на компоненти става чрез **Toolbox** лентата вляво (**View->Toolbox**, ако не я виждате). Изтегляме **три текстови полета (TextBox)**, **два надписа (Label)** и **един бутон (Button)** и ги подреждаме в прозореца.

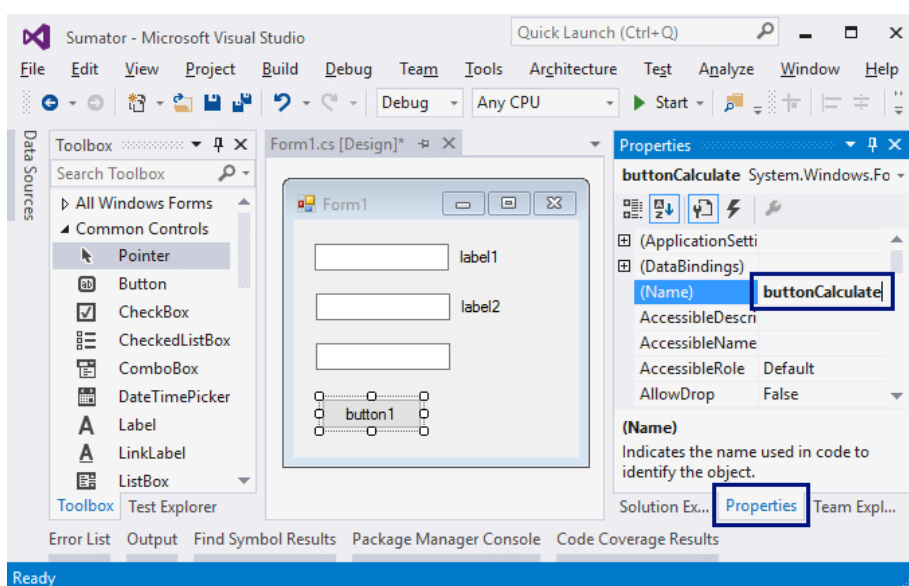
## Стъпка 3. Промяна на свойствата на контролите

След това **променяме имената на всяка от контролите**. Това става от прозорчето **“Properties”** вдясно, чрез промяна на полето **(Name)**:

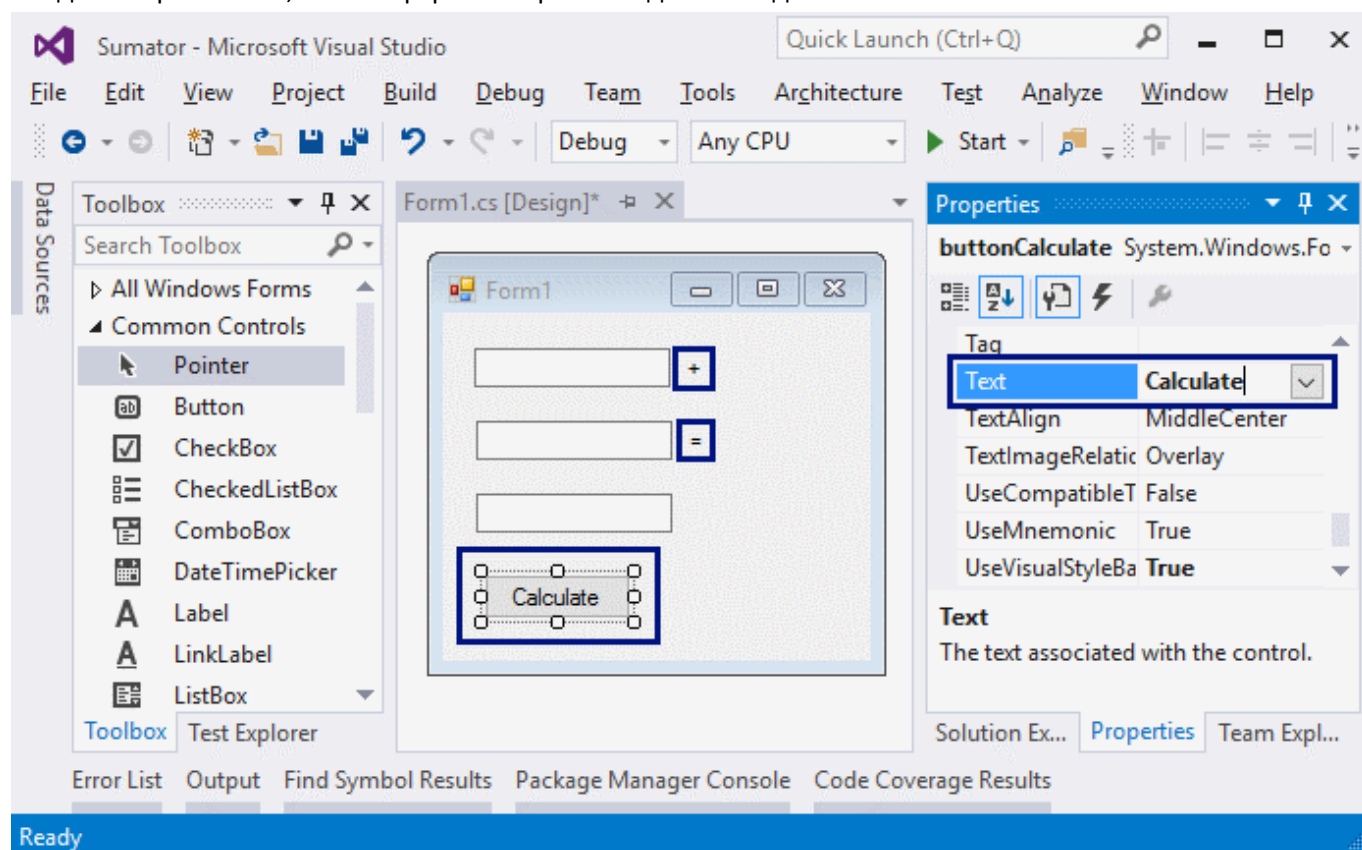
- Имена на текстовите полета: **textBox1**, **textBox2**, **textBoxSum**
- Име на бутона: **buttonCalculate**
- Име на формата: **FormCalculate**

Променяме заглавията (**Text** свойството) на контролите:

- buttonCalculate -> **Calculate**
- label1 -> **+**
- label2 -> **=**
- Form1 -> **Sumator**



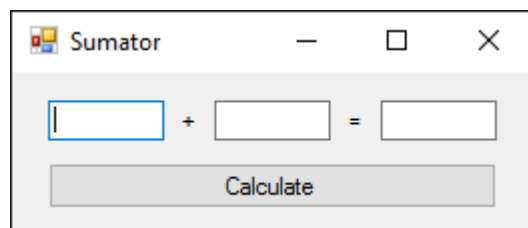
След като приключим, нашата форма би трябвало да изглежда така:



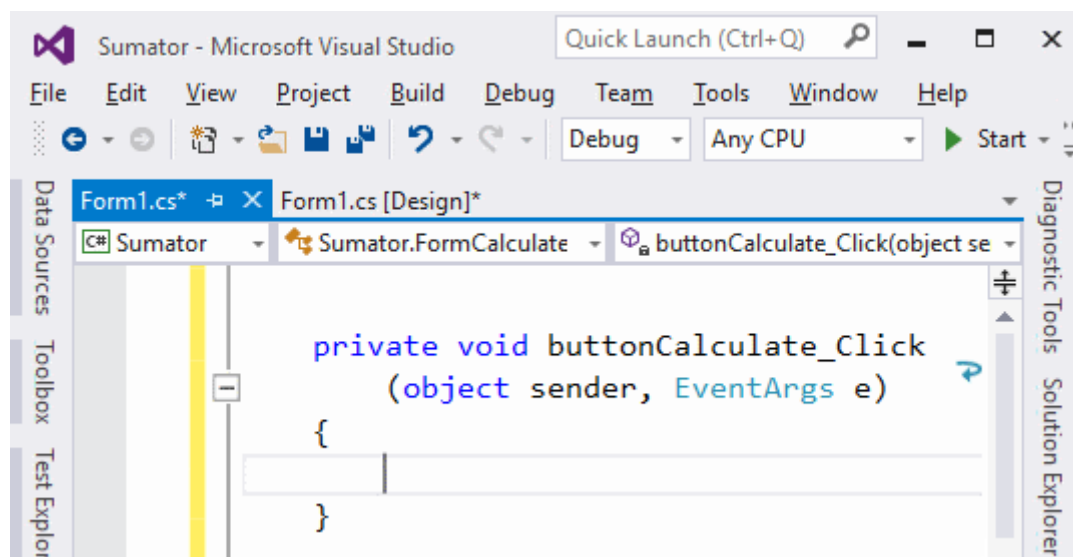
Преоразмеряваме контролите, за да изглеждат по-добре.

#### Стъпка 4. Добавяне на програмен код за събитията

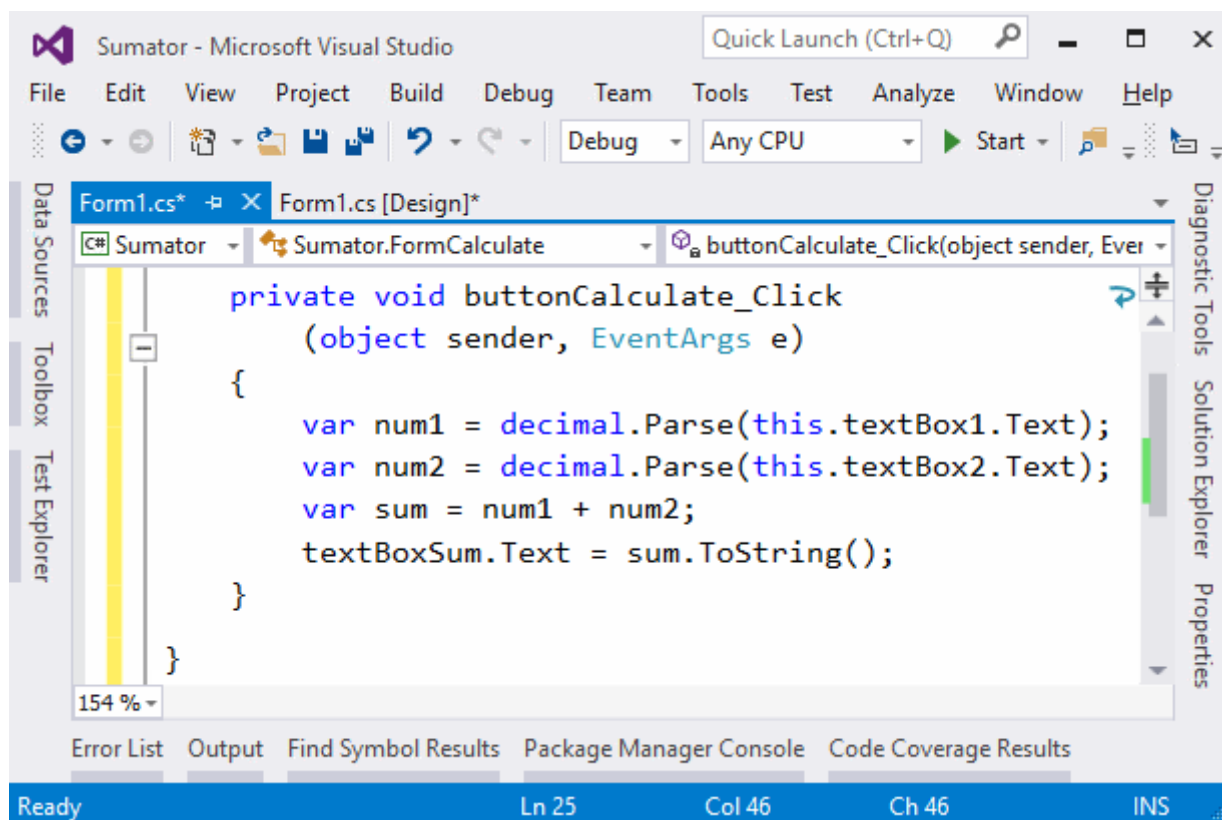
Опитваме да пуснем приложението с [Ctrl+F5]. То би трябвало да стартира, но да **не функционира напълно**, защото не сме написали какво се случва при натискане на бутона.



Сега е време да напишем кода, който **сумира числата** от първите две полета и **показва резултата** в третото поле. За целта кликваме **два пъти върху бутона [Calculate]**. Ще се появи място, в което да напишем какво да се случва при натискане на бутона:



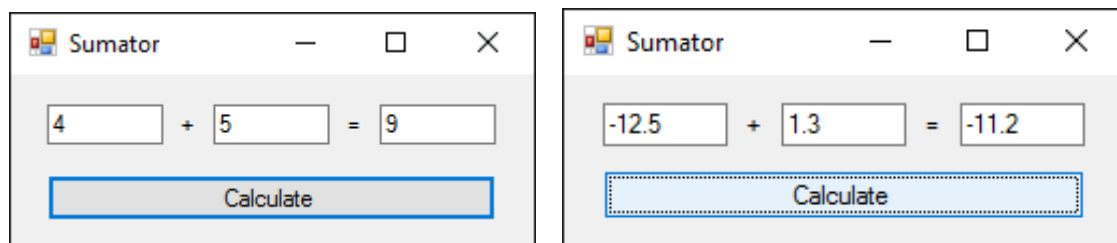
Написваме следния C# код между отварящата и затварящата скоба { }, където е курсорът:



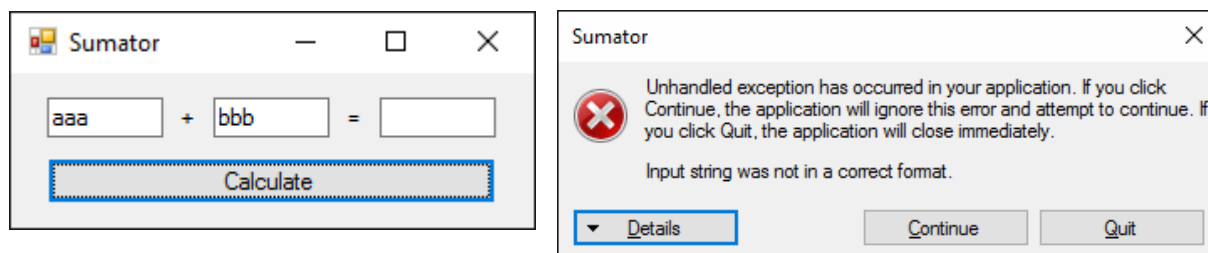
Този код **взима първото число** от полето **textBox1** и го запазва в **променливата num1**, запазва **второто число** от полето **textBox2** в **променливата num2**, след това **сумира num1 и num2** в **променливата sum** и накрая **извежда текстовата стойност на променливата sum** в полето **textBoxSum**.

### Стъпка 5. Стартиране и тестване на програмата

Стартираме отново програмата с [Ctrl+F5] и проверяваме дали работи коректно. Правим опит да сметне **4 + 5**, а след това **-12.5 + 1.3**:



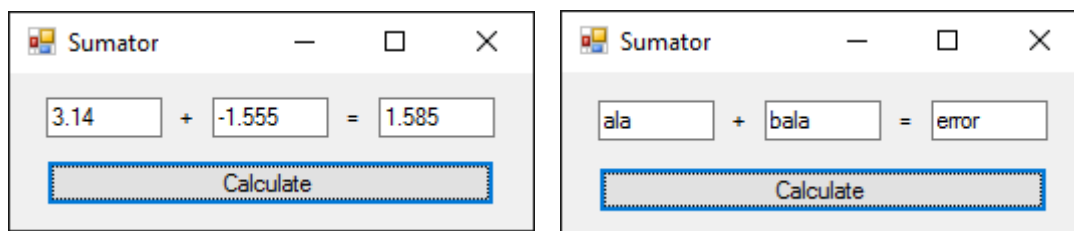
Пробваме и с **невалидни числа**, напр. **“aaa”** и **“bbb”**. Изглежда има проблем:



Проблемът идва от **прехвърлянето на текстово поле в число**. Ако стойността в полето **не е число**, **програмата хвърля изключение** и дава грешка. Можем да поправим кода, за да коригираме този проблем:

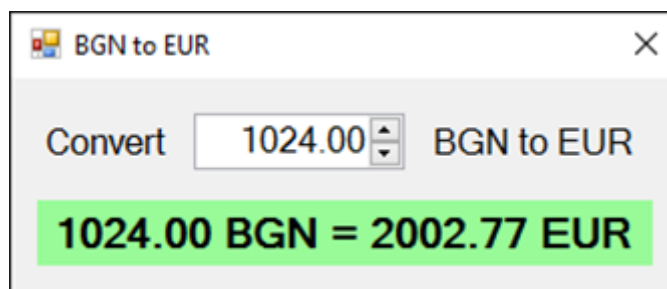
```
private void buttonCalculate_Click
(object sender, EventArgs e)
{
    try
    {
        var num1 = decimal.Parse(this.textBox1.Text);
        var num2 = decimal.Parse(this.textBox2.Text);
        var sum = num1 + num2;
        textBoxSum.Text = sum.ToString();
    }
    catch (Exception)
    {
        textBoxSum.Text = "error";
    }
}
```

Горният код **обработва грешките при работа с числа** (като прихваща изключенията) и в случай на грешка **извежда стойност error** в полето с резултата. Стартираме отново програмата с [Ctrl+F5] и я пробваме дали работи. Този път **при грешно число резултатът е error** и програмата не се чупи:



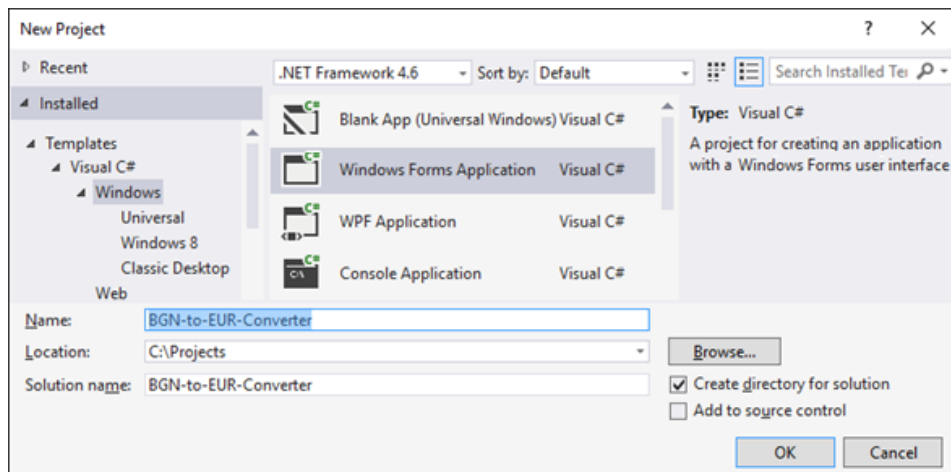
## 2. Конвертор от BGN към EUR

Да се създаде **графично приложение** (GUI application), което пресмята стойността в **евро (EUR)** на парична сума, зададена в **лева (BGN)**. При промяна на стойността в лева, равностойността в евро трябва да се преизчислява автоматично (използваме курс лева / евро: **1.95583**).



### Решение

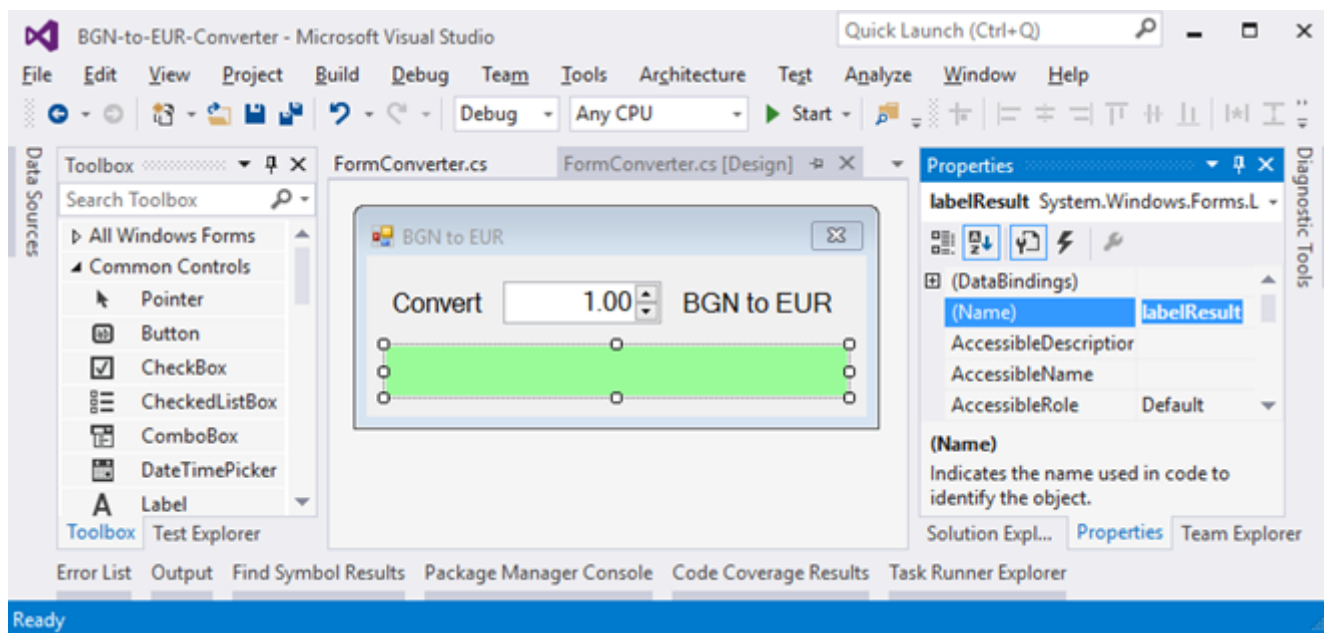
Добавяме към текущото Visual Studio решение (solution) още един проект - **Windows Forms** приложение с име **"BGN-to-EUR-Converter"**:



Поддържащите следните UI контроли във формата:

- **NumericUpDown** с име **numericUpDownAmount** – ще въвежда сумата за конвертиране
- **Label** с име **labelResult** – ще показва резултата след конвертиране
- Още два **Label** компонента, служещи единствено за статично изобразяване на текст

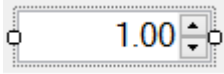

Графичният редактор за потребителски интерфейс може да изглежда по подобен начин:



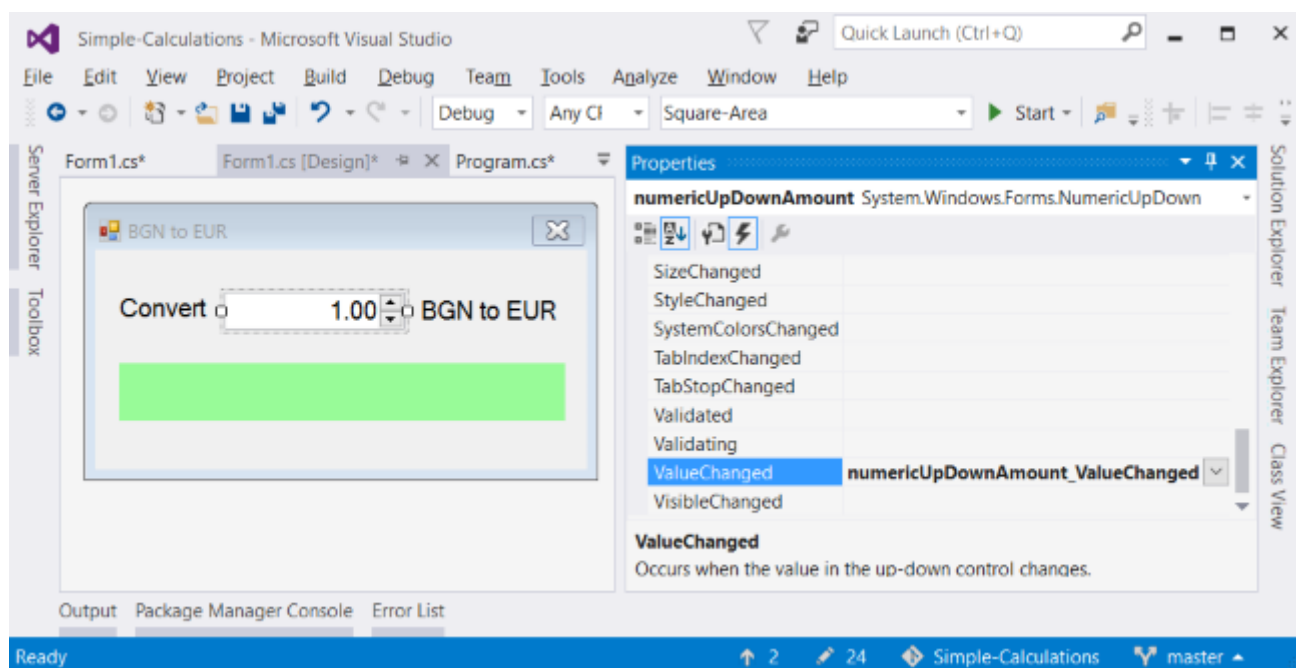
Задаваме следните настройки на формата и на отделните контроли:

Настройка	Снимка
<b>FormConverter:</b> Text = "BGN to EUR", Font.Size = 12, MaximizeBox = False, MinimizeBox = False, FormBorderStyle = FixedSingle	



Настройка	Снимка
<b>numericUpDownAmount:</b> Value = 1, Minimum = 0, Maximum = 10000000, TextAlign = Right, DecimalPlaces = 2	
<b>labelResult:</b> AutoSize = False, BackColor = PaleGreen, TextAlign = MiddleCenter, Font.Size = 14, Font.Bold = True	

Дефинираме следните **обработчици на събития** по контролите:



За **хващане на събитие** ползваме иконката със събитията (Events) в **Properties** прозореца във Visual Studio. Ще прихващаме следните събития:

- **FormConverter.Load** (като кликнем върху формата 2 пъти с мишката)
- **numericUpDownAmount.ValueChanged** (като кликнем върху **NumericUpDown** контролата 2 пъти)
- **numericUpDownAmount.KeyUp** (избираме **Events** от таблото **Properties** и кликаме 2 пъти на **KeyUp**)

Събитието **Form.Load** се изпълнява при стартиране на програмата, преди да се появи прозореца на приложението. Събитието **NumericUpDown.ValueChanged** се изпълнява при промяна на стойността в полето за въвеждане на число. Събитието **NumericUpDown.KeyUp** се изпълнява след натискане на клавиш в полето за въвеждане на число. При всяко от тези събития ще преизчисляваме резултата.

Ще използваме следния **C# код** за обработка на събитията:

```
private void FormConverter_Load(object sender, EventArgs e)
{
    ConvertCurrency();
}

private void numericUpDownAmount_ValueChanged(object sender, EventArgs e)
{
    ConvertCurrency();
}

private void numericUpDownAmount_KeyUp(object sender, KeyEventArgs e)
{
    ConvertCurrency();
}
```

Всички прихванати събития извикват метода **ConvertCurrency()**, който конвертира зададената сума от лева в евро и показва резултата в зелената кутийка.

Трябва да напишем **кода** (програмната логика) за конвертиране от лева към евро:

```
private void ConvertCurrency()
{
    var amountBGN = this.numericUpDownAmount.Value;
    var amountEUR = amountBGN * 1.95583m;
    this.labelResult.Text = amountBGN + " BGN = " + Math.Round(amountEUR, 2) + "EUR";
}
```

Накрая **стартираме проекта** с [Ctrl+F5] и тестваме дали работи коректно.

### 3. Хвани бутона!

Създайте забавно графично приложение „хвани бутона“: една форма съдържаща един бутон. При преместване на курсора на мишката върху бутона той се премества на случайна позиция. Така се създава усещане, че „**бутонът бяга от мишката и е трудно да се хване**“. При „хващане“ на бутона се извежда съобщение-поздрав.

#### Подсказка:

Добавете обработчик за събитието **Button.MouseEnter** и премествайте бутона на случайна позиция. Използвайте генератор за случайни числа **Random**. Позицията на бутона се



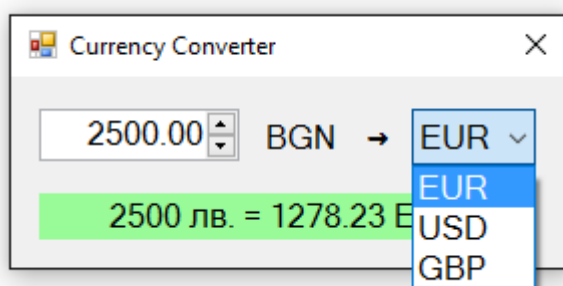


задава от свойството **Location**. За да бъде новата позиция на бутона в рамките на формата, можете да направите изчисления спрямо размера на формата, който е достъпен от свойството **ClientSize**. Можете да ползвате следния примерен код:

```
private void buttonCatchMe_MouseEnter(object sender, EventArgs e)
{
    Random rand = new Random();
    var maxWidth = this.ClientSize.Width - buttonCatchMe.ClientSize.Width;
    var maxHeight = this.ClientSize.Height - buttonCatchMe.ClientSize.Height;
    this.buttonCatchMe.Location = new Point(
        rand.Next(maxWidth), rand.Next(maxHeight));
}
```

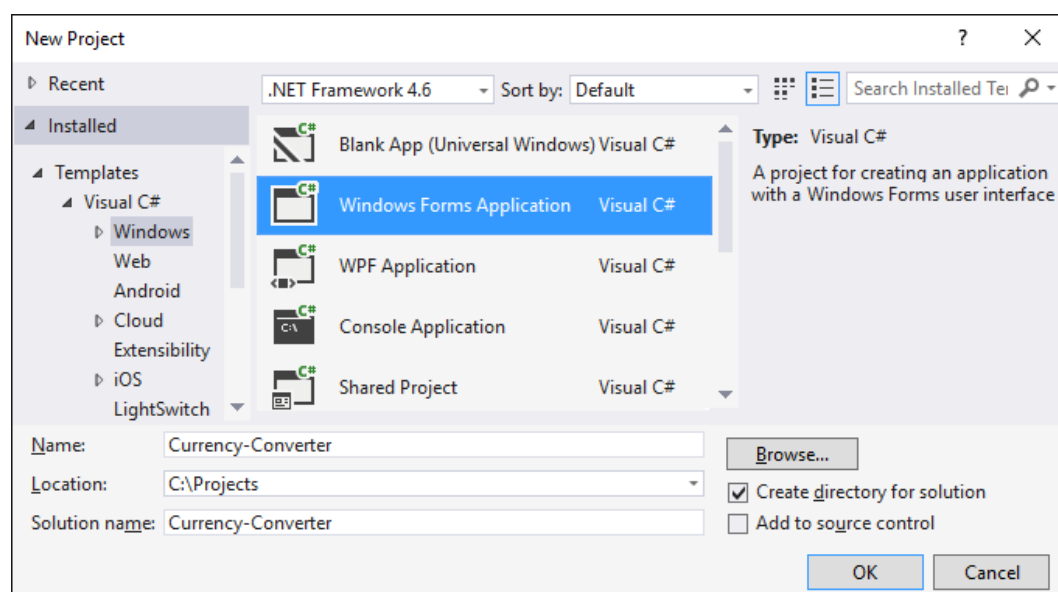
## 4. Конвертор за валути

Нека разгледаме как да създадем графично (GUI) приложение за **конвертиране на валути**. Приложението ще изглежда приблизително като на картинката по-долу:



### Решение

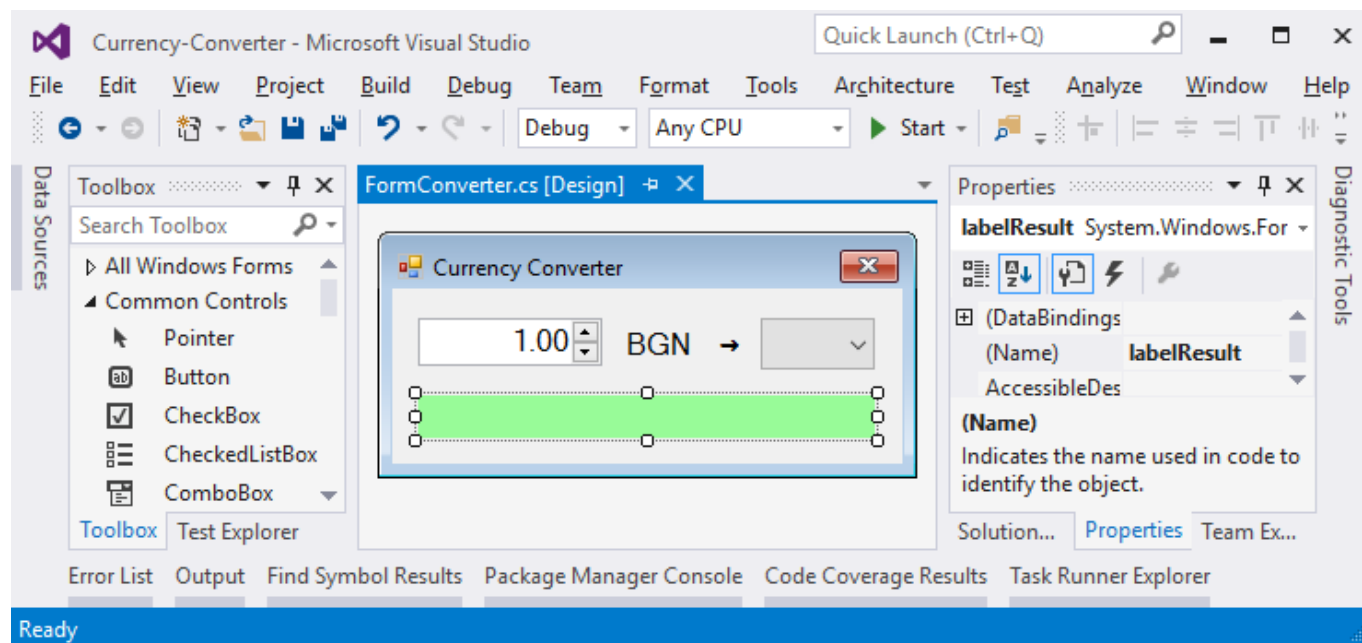
Този път създаваме нов **Windows Forms Application** с име "Currency-Converter":



Нареждаме следните контроли във формата:

- Една кутийка за въвеждане на число (**NumericUpDown**)
- Един падащ списък с валути (**ComboBox**)
- Текстов блок за резултата (**Label**)
- Няколко надписа (**Label**)

Нагласяме **размерите** и свойствата им, за да изглеждат долу-горе като на картинката:



Задаваме следните **настройките на контролите**:

- **За главната форма (Form)**, която съдържа всички контроли:
  - **(name) = FormConverter**
  - **Text = "Currency Converter"**
  - **Font.Size = 12**
  - **MaximizeBox = False**
  - **MinimizeBox = False**
  - **FormBorderStyle = FixedSingle**
- **За полето за въвеждане на число (NumericUpDown):**
  - **(name) = numericUpDownAmount**
  - **Value = 1**
  - **Minimum = 0**
  - **Maximum = 1000000**
  - **TextAlign = Right**
  - **DecimalPlaces = 2**

- За падащия списък с валутите (ComboBox):

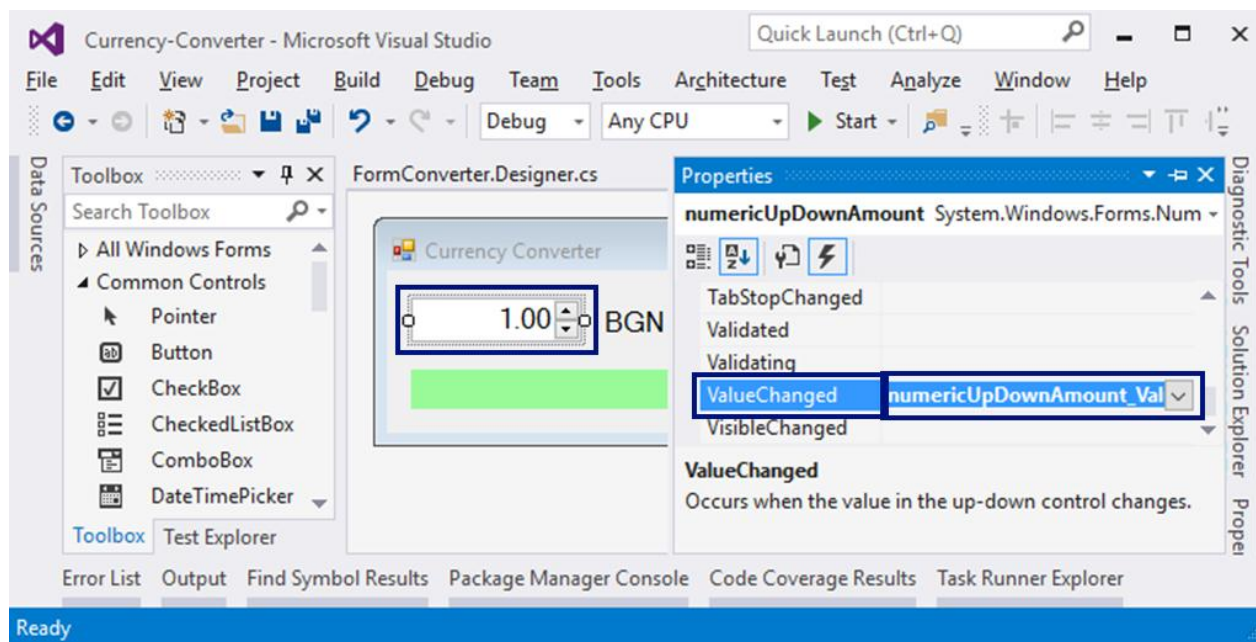
- (name) = comboBoxCurrency
- DropDownStyle = DropDownList
- Items =
  - EUR
  - USD
  - GBP

- За текстовия блок за резултата (Label):

- (name) = labelResult
- AutoSize = False
- BackColor = PaleGreen
- TextAlign = MiddleCenter
- Font.Size = 14
- Font.Bold = True

Трябва да хванем следните **събития**, за да напишем C# кода, който ще се изпълни при настъпването им:

- Събитието **ValueChanged** на контролата за въвеждане на число **numericUpDownAmount**:



- Събитието **Load** на формата **FormConverter**
- Събитието **SelectedIndexChanged** на падащия списък за избор на валута **comboBoxCurrency**

Ще използваме следния **C# код** за обработка на събитията:

```
private void FormConverter_Load(object sender, EventArgs e)
{
    this.comboBoxCurrency.SelectedItem = "EUR";
}
```

```

}

private void numericUpDownAmount_ValueChanged(object sender, EventArgs e)
{
    ConvertCurrency();
}

private void comboBoxCurrency_SelectedIndexChanged(object sender, EventArgs e)
{
    ConvertCurrency();
}

```

Задачата на горния код е да избере при стартиране на програмата валута **“EUR”** и при промяна на стойностите в полето за сума или при смяна на валутата, да изчисли резултата чрез **ConvertCurrency()** метода.

Остава да напишем метода **ConvertCurrency()** за смятане на въведената сума от лева в избраната валута:

```

private void ConvertCurrency()
{
    var originalAmount = this.numericUpDownAmount.Value;
    var convertedAmount = originalAmount;
    if (this.comboBoxCurrency.SelectedItem.ToString() == "EUR")
    {
        convertedAmount = originalAmount / 1.95583m;
    }
    else if (this.comboBoxCurrency.SelectedItem.ToString() == "USD")
    {
        convertedAmount = originalAmount / 1.80810m;
    }
    else if (this.comboBoxCurrency.SelectedItem.ToString() == "GBP")
    {
        convertedAmount = originalAmount / 2.54990m;
    }
    this.labelResult.Text = originalAmount + " лв. = " +
    Math.Round(convertedAmount, 2) + " " + this.comboBoxCurrency.SelectedItem;
}

```

Горният код взима **сумата** за конвертиране от полето **numericUpDownAmount** и **избраната валута** за резултата от полето **comboBoxCurrency**. След това с **условна конструкция**, според избраната валута, сумата се дели на **валутния курс** (който е фиксиран твърдо в сорс кода). Накрая се генерира текстово **съобщение с резултата** (закръглен до 2 цифри след десетичния знак) и се записва в зелената кутийка **labelResult**.