

## **Wykład 4**

# **Programowanie wizualne z wykorzystaniem Windows Forms**



**GUI**

**Graficzny Interfejs Użytkownika**



## Metody

---

- ✓ W Visual Studio aplikacje z graficznym interfejsem użytkownika (ang. graphical user interface — GUI) przeznaczone na pulpit systemu Windows można tworzyć korzystając z dwóch bibliotek kontrolek:
  - ✓ tradycyjnej **Windows Forms**,
  - ✓ nowszej **Windows Presentation Foundation (WPF)**.

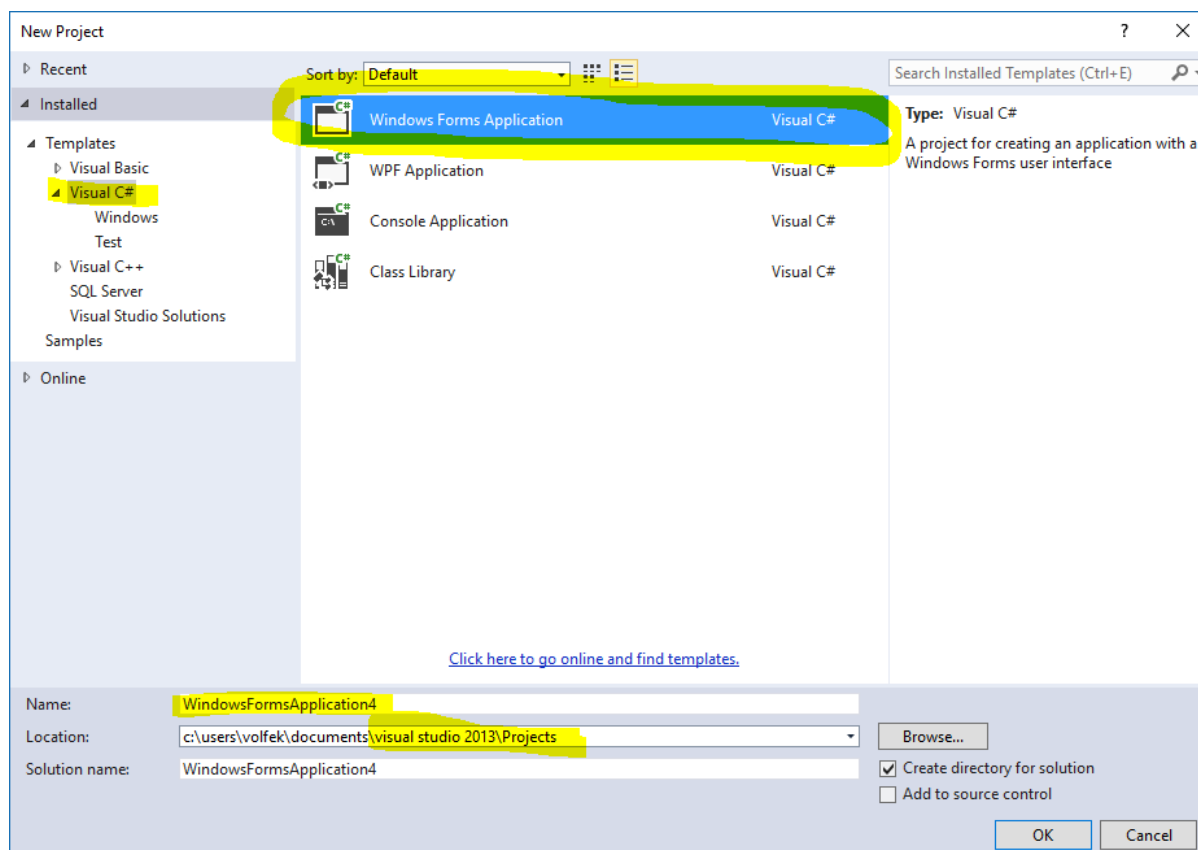


## Tworzenie projektu

## Tworzenie projektu

Projekt aplikacji okienkowej (typu Windows Forms Application):

- ✓ W menu File środowiska Visual Studio wybierz podmenu New, a następnie Project....



# Tworzenie projektu

**Zbiór dostępnych kontrolerek**

**Podgląd formy (okna)**

**Lista plików Rozwiązania (Solution) i projektu**

**okno Properties Właściwości aktualnie wybranego obiektu**

Form1.cs [Design]

Form1

Properties

Form1 System.Windows.Forms.Form

**Accessibility**

AccessibleDescription	
AccessibleName	
AccessibleRole	Default

**Appearance**

BackColor	<input type="checkbox"/> Control
BackgroundImage	<input type="checkbox"/> (none)
BackgroundImageLayc	Tile
Cursor	Default

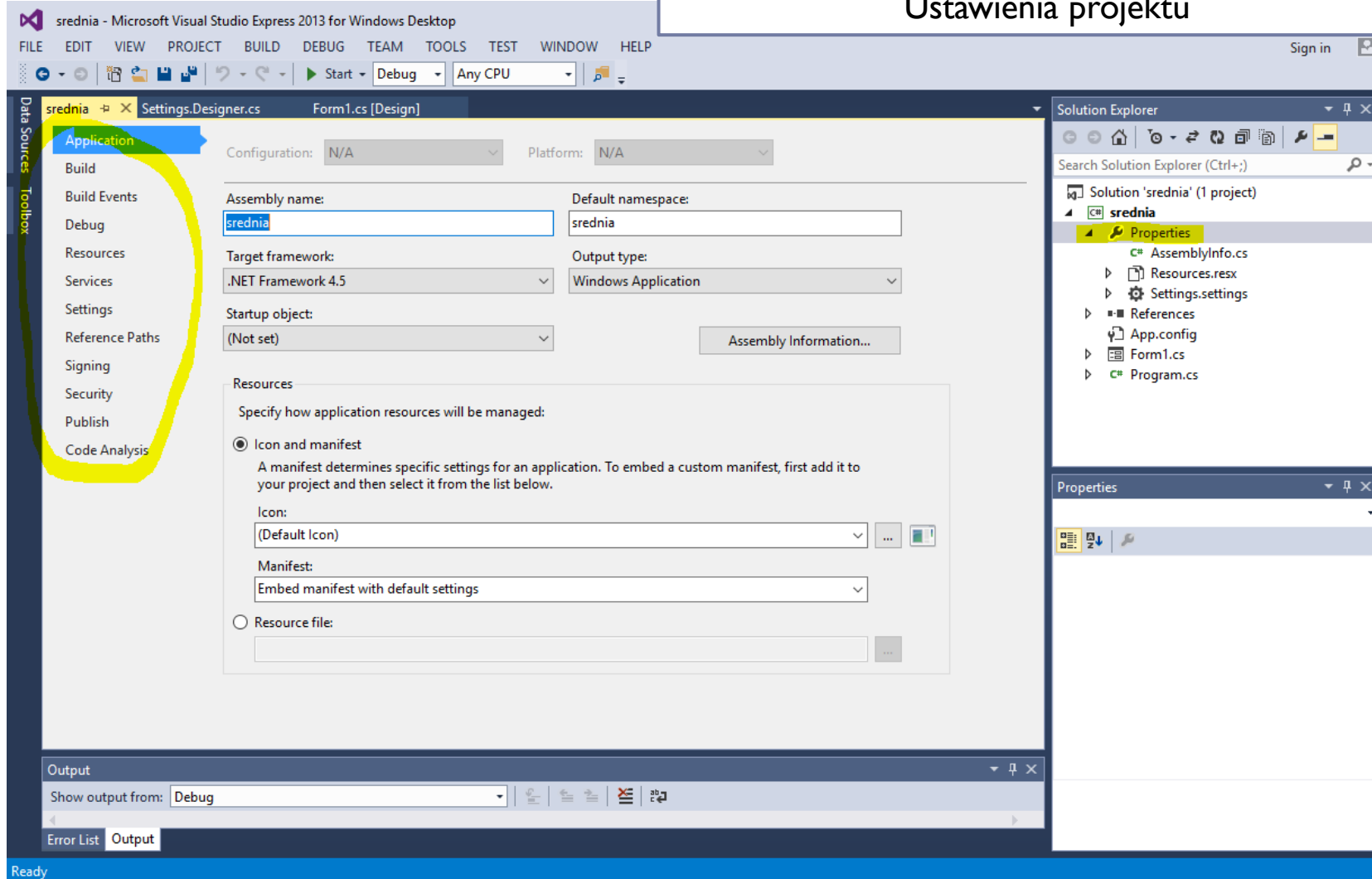
**Text**

The text associated with the control.

# Tworzenie projektu



## Ustawienia projektu



# Podstawowe kontrolki

The screenshot displays the Windows Forms Designer interface. On the left is the **Toolbox** with a search bar and a list of controls: All Windows Forms, Common Controls, Pointer, Button, CheckBox, CheckedListBox, ComboBox, DateTimePicker, Label, LinkLabel, ListBox, ListView, MaskedTextBox, MonthCalendar, NotifyIcon, NumericUpDown, PictureBox, ProgressBar, RadioButton, RichTextBox, TextBox, ToolTip, TreeView, and WebBrowser. The main area shows a form named **Form1** containing several controls: a button labeled **button1**, a checkbox labeled **checkBox1**, a label **label1**, a radio button labeled **radioButton1**, a text box, a dropdown menu, a numeric up-down control showing **0**, and a progress bar. On the right is the **Properties** window for **button1** (System.Windows.Forms.Button). It shows various property categories: Accessibility, Appearance, FlatAppearance, Font, Image, ImageList, RightToLeft, Text, Behavior, and Data. The **Text** property is highlighted and set to **button1**. A text box at the bottom right states: "Każda z kontrolki posiada swój własny zestaw właściwości (Choć oczywiście większość z nich się powtarza)".

Każda z kontrolki posiada swój własny zestaw właściwości (Choć oczywiście większość z nich się powtarza)





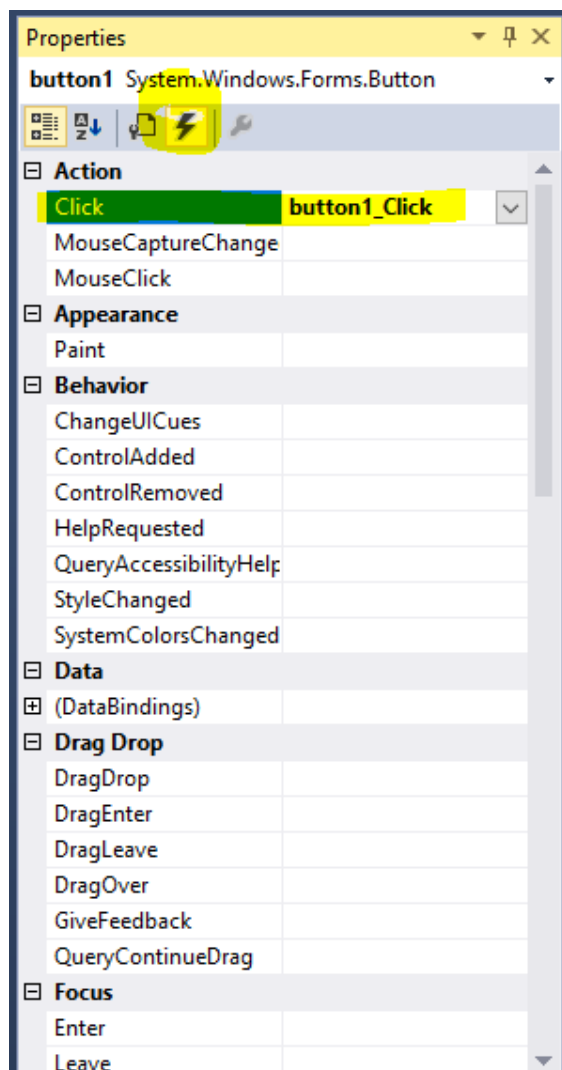
## Konstruktor okna

---

Konstruktor klasy `Form1` (okna programu) tworzony jest automatycznie. Można uzupełniać go o akcje, które mają być wykonane na starcie programu.

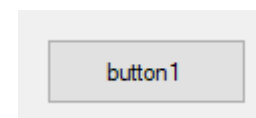
```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }
}
```

# Zdarzenia



Każdy z obiektów (w szczególności kontrolki) ma przypisany do nich zestaw zdarzeń na które mogą zareagować.

Standardowym (choć jak widać na rysunku po lewej nie jednym) zdarzeniem kontrolki button jest zdarzenie Click – reakcja na kliknięcie

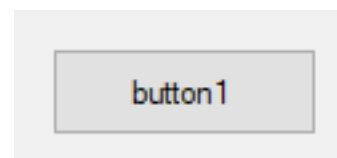


## Zdarzenia – kontrolka Button

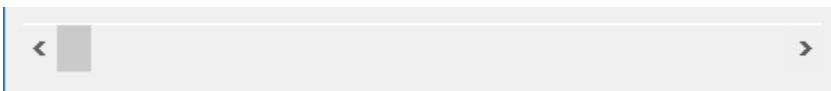
Zdarzenia obsługujemy implementując odpowiadające im metody (generowanie automatycznie po wybraniu odpowiedniej metody z listy)

```
namespace srednia
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
        }
    }
}
```

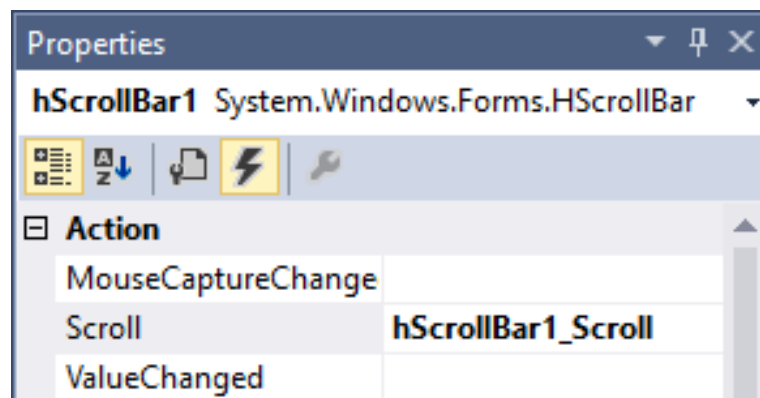


## Zdarzenia – kontrolka ScrollBar



Zdarzenia Kontrolki ScrollBar obsługujemy najczęściej implementując metody:

- **Scroll** – wykonywa podczas przesuwania kontrolki (Uwaga na obliczeniochłonne akcje)
- **ValueChanged** – wykonywana po zmianie wartości
- **MouseCaptureChange** – wykonywana po „puszczeniu” klawisza myszy – dzięki temu nadaje się do zaimplementowania akcji, które nie wykonują się „w czasie rzeczywistym”



# TextBox

## Odczytanie danych z kontrolki TextBox



```
String s = textBox1.Text;
```



```
int liczba = int.Parse(textBox1.Text);
```

Metoda **TryParse** pozwala zabezpieczyć się przed zawieszeniem programu przy próbie zamiany na liczbę ciągu znaków, który nią nie jest.

```
int liczba;  
int number;  
if (int.TryParse(textBox1.Text, out number))  
    liczba = number;  
else liczba = 0;
```

## Zapis danych do kontrolki TextBox

```
textBox1.Text="Napis";
```



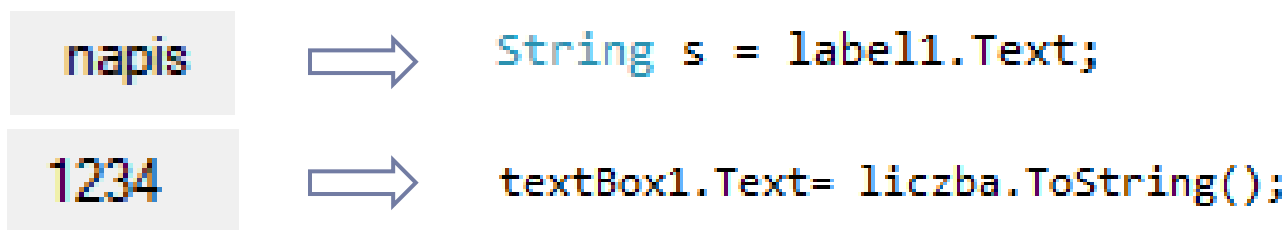
```
textBox1.Text= liczba.ToString();
```



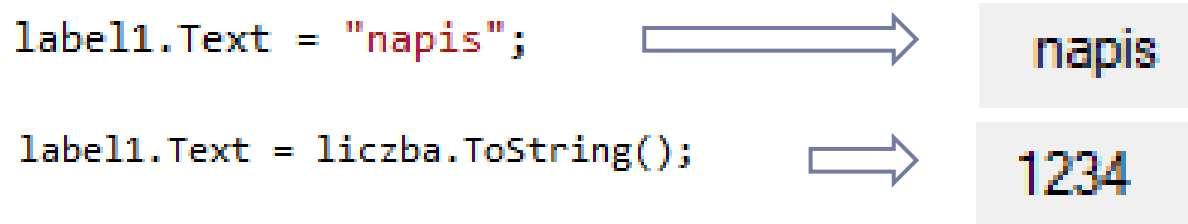
# Label

Kontrolkę Label możemy traktować analogicznie jak TextBox – ale tylko z poziomu programu – użytkownik nie może wpisać tekstu do kontrolki Label

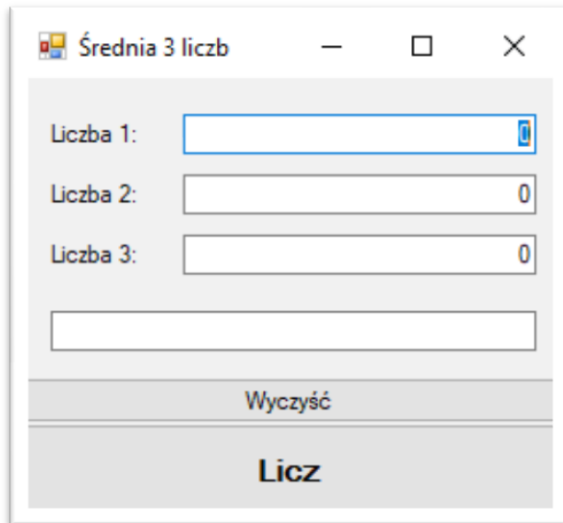
## Odczytanie danych z kontrolki Label



## Zapis danych do kontrolki TextBox



## Przykład



Średnia 3 liczb

Liczba 1:

Liczba 2:  0

Liczba 3:  0

Wyczyść

Licz

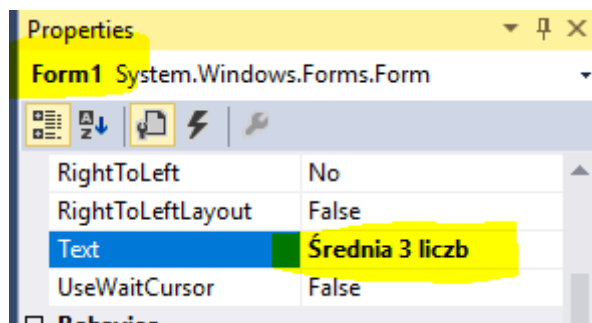
## PRZYKŁAD: ŚREDNIA TRZECH LICZB

Program wykorzystuje kontrolki:

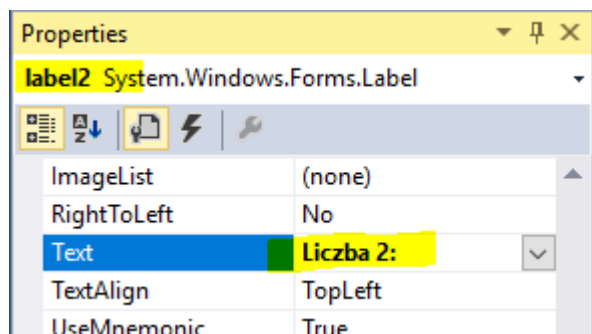
- ✓ TextBox,
- ✓ Label
- ✓ Button.

## Przykład

### Kilka przydatnych właściwości kontroltek



Tytuł okna programu  
(pojawi się na belce)

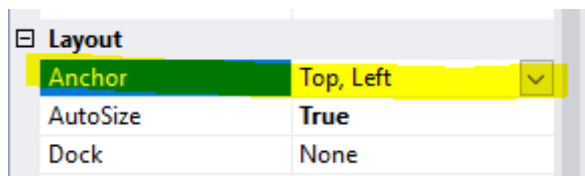


Zawartość wyświetlana  
przez kontrolki Label

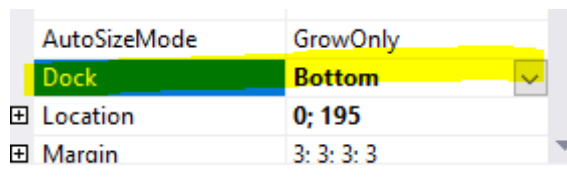
Dla kontrolki TextBox działa  
to analogicznie



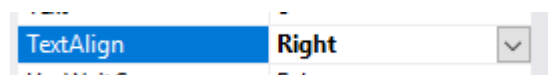
## Przykład



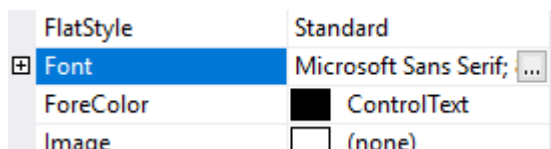
Anchor – pilnuje zakotwiczenia kontrolki – tzn. podczas skalowania okna odległości od sąsiadujących elementów będą zachowywane



Dock – dokuje element do wskazanej krawędzi wolnej przestrzeni – tzn. przykleja element i rozciąga.



Wyrównanie tekstu wewnątrz kontrolki



Zestaw właściwości Fontu



## Przykład

### Zawartość metody Click przycisku „Licz”

```
private void button1_Click(object sender, EventArgs e)
{
    int a, b, c;
    double srednia;
    // a = int.Parse(textBox1.Text);
    // b = int.Parse(textBox1.Text);
    // c = int.Parse(textBox1.Text);
    int number;
    if (int.TryParse(textBox1.Text, out number))
        a=number; else a=0;
    if (int.TryParse(textBox2.Text, out number))
        b=number; else b=0;
    if (int.TryParse(textBox3.Text, out number))
        c=number; else c=0;
    srednia = (a + b + c) / 3.0;
    textBox4.Text = srednia.ToString();
}
```

Wersja wczytania danych z TextBox bez sprawdzenia poprawności (tu w komentarzu)

Wersja wczytania danych z TextBox ze sprawdzeniem poprawności



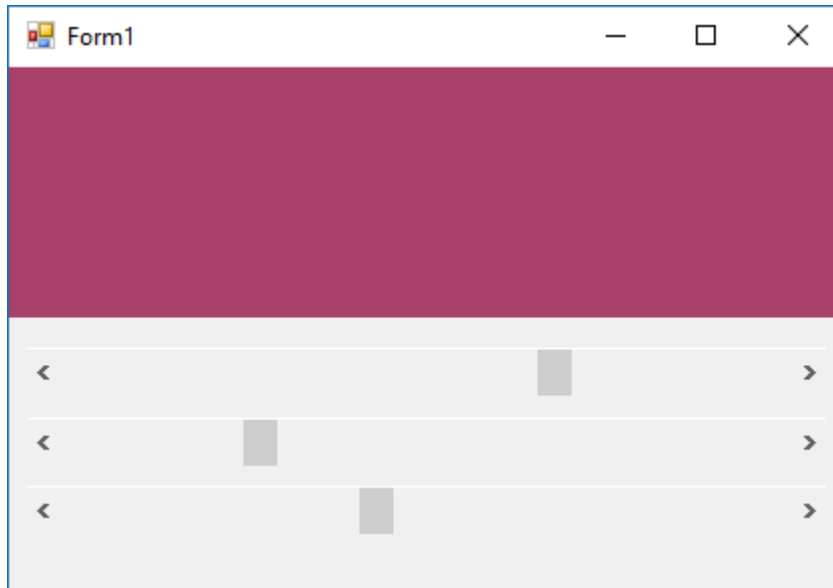
## Przykład

---

### Zawartość metody Click przycisku „Wyczyść”

```
private void button2_Click(object sender, EventArgs e)
{
    textBox1.Text = "0";
    textBox2.Text = "0";
    textBox3.Text = "0";
    textBox4.Text = "";
}
```

## Przykład 2

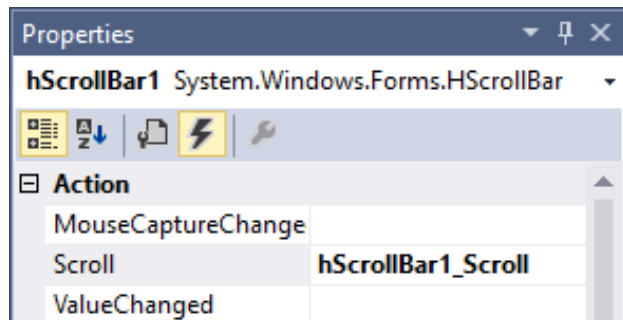


## PRZYKŁAD: TESTER KOLORÓW RGB

Okno programu składa się z kontrolki **Panel**, której kolor będziemy zmieniać oraz z trzech kontrolek **hScrollBar** (pozioma wersja suwaka)

## Przykład 2

Kolor ustawiamy za pomocą metody  
`Color.FromArgb( )`



```
private void hScrollBar1_Scroll(object sender, ScrollEventArgs e)
{
    panel1.BackColor = Color.FromArgb(hScrollBar1.Value,
                                      hScrollBar2.Value,
                                      hScrollBar3.Value);
}
```

Analogicznie dla pozostałych suwaków...

## Przykład 2

Behavior	
AllowDrop	False
ContextMenuStrip	(none)
Enabled	True
LargeChange	10
Maximum	255
Minimum	0
SmallChange	1
TabIndex	1
TabStop	False
Value	0

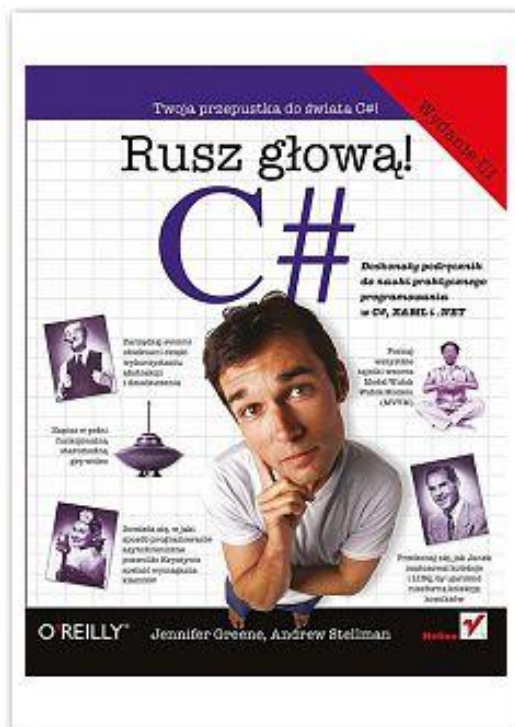
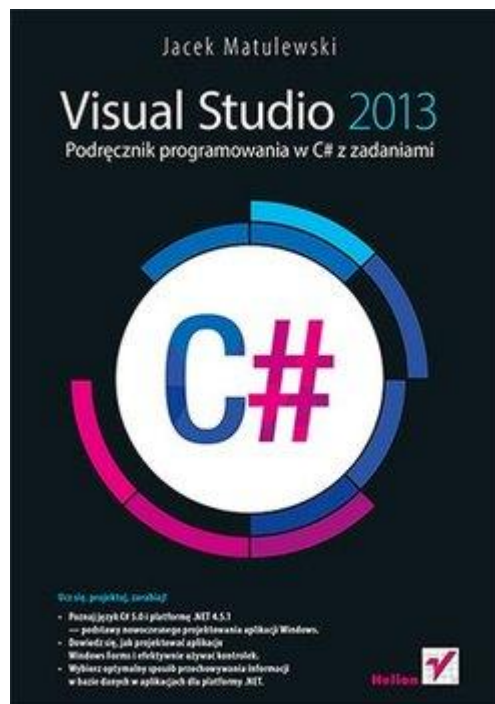
Aby program działał prawidłowo ustawić należy zakres wartości suwaków <0 ; 255>

**Zdarzenie można wywołać „sztucznie” w kodzie programu**

```
public Form1()
{
    InitializeComponent();
    this.hScrollBar1_Scroll(this, null);
}
```

W tym przypadku w konstruktorze okna wywołujemy zdarzenie suwaka, czyli ustawiamy kolor panelu

# Literatura:



Użyte w tej prezentacji tabelki pochodzą z książki: Visual Studio 2013. Podręcznik programowania w C# z zadaniami  
Autor: Matulewski Jęko, Helion