

## **Wykład 1**

### **Programowanie wizualne z wykorzystaniem Windows Forms**



**GUI**

**Graficzny Interfejs Użytkownika**



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

Windows Forms – to interfejs programowania graficznych aplikacji (API) w ramach Microsoft .NET Framework, umożliwiający natywny dostęp do elementów interfejsu graficznego Microsoft Windows.

Obecnie miejsce aplikacji Windows Forms jest zajmowane przez aplikacje WPF, które są tworzone przy użyciu języka XAML.



## Tworzenie projektu

## Tworzenie projektu







### Instalacja MS Visual Studio





Modyfikowanie — Visual Studio Community 2019 — 16.7.5

**Pakiety robocze**    Poszczególne składniki    Pakiety językowe    Lokalizacje instalacji

Sieć Web i chmura (4)

-  Opracowywanie zawartości dla platformy ASP.NET i sieci Web  
Twórz aplikacje internetowe dla wielu platform przy użyciu...
-  Programowanie na platformie Azure  
Zestawy SDK, narzędzia i projekty platformy Azure do tworzenia aplikacji w chmurze i zasobów za pomocą...
-  Opracowywanie zawartości w języku Python  
Edytowanie, debugowanie, opracowywanie interakcyjne oraz kontrola źródła dla języka Python.
-  Programowanie za pomocą oprogramowania Node.js  
Twórz skalowalne aplikacje sieciowe przy użyciu środowiska Node.js — asynchronicznego środowiska...

Komputery i urządzenia przenośne (5)

-  Programowanie aplikacji klasycznych dla platformy .NET  
Twórz aplikacje WPF i Windows Forms oraz aplikacje konsolowe przy użyciu języków C#, Visual Basic i F# za... ☒
-  Programowanie aplikacji klasycznych w języku C++  
Kompiluj nowoczesne aplikacje C++ dla systemu Windows przy użyciu wybranych przez siebie narzędzi, takich jak...
-  Opracowywanie zawartości dla platformy uniwersalnej systemu Windows  
Twórz aplikacje dla platformy uniwersalnej systemu...
-  Opracowywanie aplikacji mobilnych za pomocą środowiska .NET  
Twórz aplikacje dla wielu platform (iOS, Android lub...

Lokalizacja  
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community

Kontynuując, wyrażasz zgodę na warunki [licencji](#) wybranej wersji programu Visual Studio. Oferujemy również możliwość pobrania innego oprogramowania z programem Visual Studio. To oprogramowanie jest licencjonowane oddzielnie, zgodnie z informacjami podanymi w temacie [3rd Party Notices](#) (Materiały innych producentów) lub w dołączonej licencji. Kontynuując, wyrażasz również zgodę na warunki tych licencji.

**Szczegóły instalacji**

> Visual Studio Core Editor

✓ Programowanie aplikacji klasycznych dla plat...  
Zawarte

- ✓ Narzędzia do programowania aplikacji klasyczn...
- ✓ .NET Framework 4.7.2 Developer Tools
- ✓ C# i Visual Basic
- ✓ IntelliCode

Opcjonalnie

- ☒ Narzędzia programistyczne platformy .NET Core
- ☒ Środowisko uruchomieniowe .NET Core 2.1 (LTS)
- ☒ Narzędzia programistyczne programu .NET Fra...
- ☒ Blend for Visual Studio
- ☒ Entity Framework 6 Tools
- ☒ Narzędzia profilowania dla programu .NET
- ☒ Debugger just in time
- ☒ Live Share
- ☒ ML.NET Model Builder (wersja zapoznawcza)
- ☐ Obsługa języka F# dla komputerów
- ☐ PreEmptive Protection - Dotfuscator
- ☐ Narzędzia deweloperskie dla platformy .NET Fra...

Całkowite wymagane miejsce to -6,33 GB

Zainstaluj podczas pobierania

# Tworzenie projektu



## Tworzenie nowego projektu

## Ekran powitalny

Co chcesz zrobić?

Otwórz ostatnio używane

Przeszukaj ostatnie (Alt+S)

- Dzisiaj**
  - Test.sln  
C:\Users\AB\source\repos\Test 13.10.2020 16:49
- Wczoraj**
  - tablice-konsola.sln  
C:\Users\AB\source\repos\tablice-konsola 12.10.2020 19:11
  - kalk\_konsola.sln  
C:\Users\AB\source\repos\kalk\_konsola 12.10.2020 18:38
- Ten tydzień**
  - w01p01.sln  
C:\Users\AB\source\repos\w01p01 07.10.2020 12:31
  - L1p01-gra.sln  
C:\Users\AB\source\repos\L1p01-gra 06.10.2020 14:32
  - L3p01-gra.sln  
C:\Users\AB\source\repos\L3p01-gra 06.10.2020 12:52

Rozpocznij

- Klonuj repozytorium  
Pobierz kod z repozytorium online, takiego jak GitHub lub Azure DevOps
- Otwórz projekt lub rozwiązanie  
Otwórz lokalny plik projektu lub sln programu Visual Studio
- Otwórz folder lokalny  
Nawiguj i edytuj kod w dowolnym folderze
- Utwórz nowy projekt**  
Wybierz szablon projektu z tworzeniem szkieletu kodu, aby rozpocząć pracę

Utwórz nowy projekt

Wyszukaj szablony (Alt+S)

Ostatnie szablony projektów

- Aplikacja Windows Forms (.NET Framework) C#
- Aplikacja konsoli (.NET Core) C#

C# Windows Klasyczny

- Projekt testowy NUnit (.NET Core)  
Projekt zawierający testy NUnit, które mogą być uruchamiane na platformie .NET Core w systemach Windows, Linux i MacOS.
- Aplikacja Windows Forms (.NET Framework)**  
Projekt do tworzenia aplikacji z interfejsem użytkownika w modelu Windows Forms (WinForms)  
C# Windows Klasyczny
- Aplikacja WPF (.NET Framework)  
Aplikacja kliencka Windows Presentation Foundation  
C# XAML Windows Klasyczny
- WPF App (.NET Core)  
Aplikacja kliencka Windows Presentation Foundation  
C# XAML Windows Klasyczny
- WPF Custom Control Library (.NET Core)  
Biblioteka kontrolki niestandardowych Windows Presentation Foundation  
C# XAML Windows Klasyczny Biblioteka

Wstecz Dalej

# Tworzenie projektu



Środowisko:

Zrzut ekranu środowiska deweloperskiego Visual Studio. W centralnym obszarze widoczny jest podgląd formularza 'Form1'. Po lewej stronie znajduje się 'Przybornik' (Toolbox) z listą dostępnych kontrolek, takich jak Wskaźnik, Button, CheckBox, czy TextBox. Po prawej stronie widoczny jest 'Eksplorator rozwiązań' (Solution Explorer) z listą plików projektu, w tym 'Form1.cs' i 'Program.cs'. W dolnej części po prawej stronie znajduje się 'okno Properties' (Właściwości), które wyświetla ustawienia dla wybranego obiektu, takie jak 'AutoScroll', 'Location', 'Size' (816; 489) oraz 'BackColor'. W dolnej części ekranu widoczny jest pasek zadań z przyciskami 'Dane wyjściowe' i 'Lista błędów'. W prawym dolnym rogu znajduje się przycisk 'Dodaj do kontroli źródła'.

Podgląd formy (okna)

Lista plików Rozwiązania (Solution) i projektu

okno *Properties* Właściwości aktualnie wybranego obiektu

Zbiór dostępnych kontrolek

# Tworzenie projektu



## Ustawienia projektu

Test\* -> Form1.cs [Projekt]\*

**Aplikacja**

Konfiguracja: ND Platforma: ND

Nazwa zestawu: Test Domyślna przestrzeń nazw: Test

Docelowa struktura: .NET Framework 4.7.2 Typ wyjściowy: Aplikacja systemu Windows

☒ Automatycznie generuj przekierowania powiązań

Obiekt uruchomieniowy: (Nie ustawiono) Informacje o zestawie...

**Zasoby**

Określa, jak będą zarządzane zasoby aplikacji:

☒ Ikona i manifest

Manifest określa specyficzne ustawienia dla aplikacji. Aby osadzić niestandardowy manifest, najpierw dodaj go do projektu, a następnie wybierz go z poniższej listy.

Ikona: (Ikona domyślna) Przeglądaj...

Manifest: Osadź manifest w ustawieniach domyślnych

☐ Plik zasobów: Przeglądaj...

**Eksplorator rozwiązań**

Przeszukaj: Eksplorator rozwiązań (Ctrl+;) 🔍

Rozwiązanie „Test” (liczba projektów: 1 z 1)

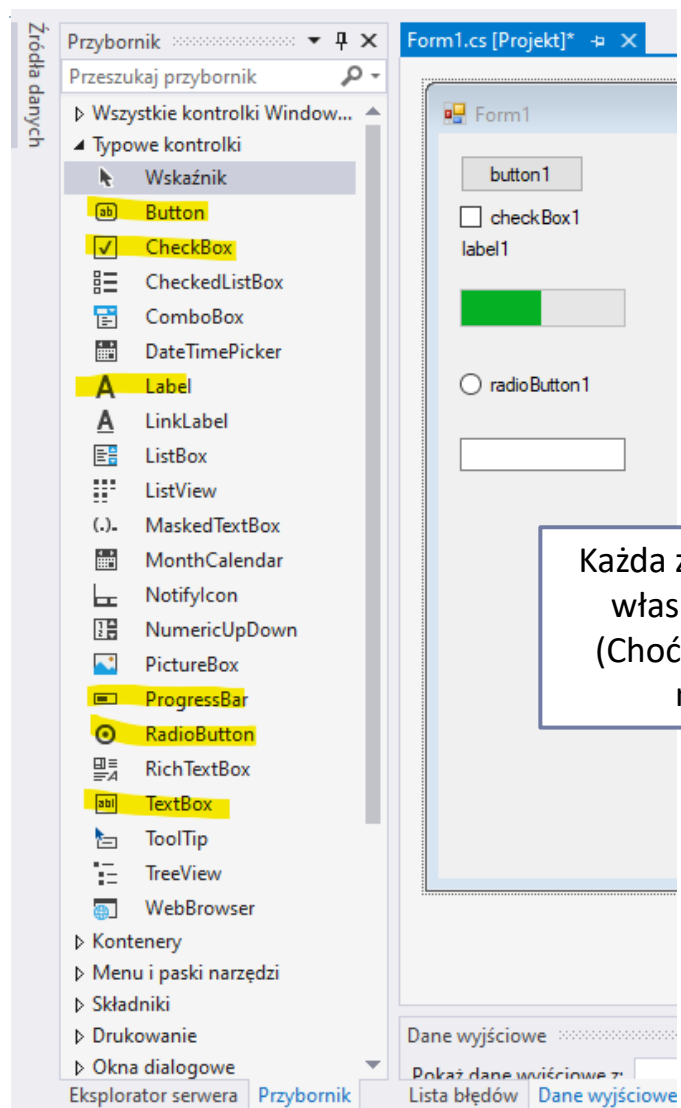
- Test
  - Properties
  - Odwołania
  - App.config
  - Form1.cs
    - Form1.Designer.cs
    - Form1.resx
  - Program.cs

Live Share Eksplorator rozwiązań... Team Explorer

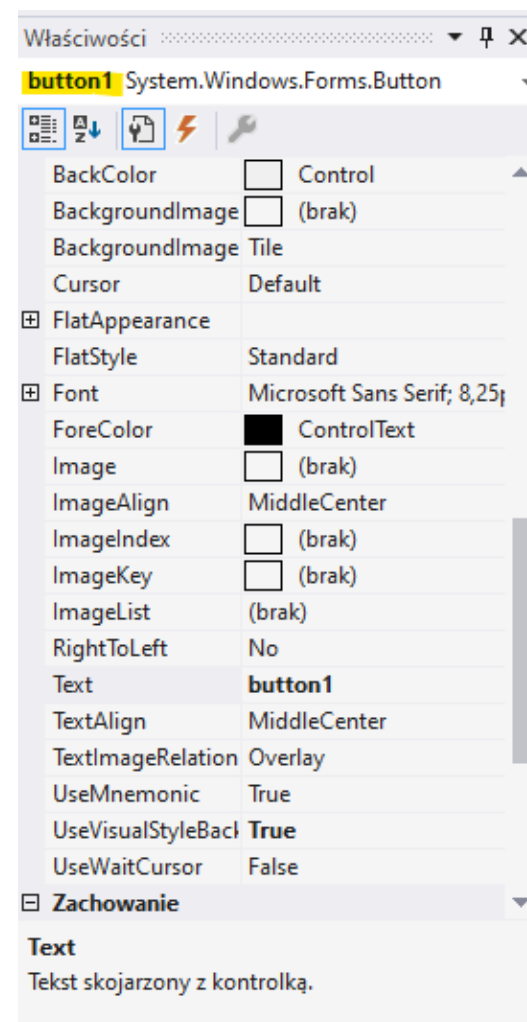
**Właściwości**



# Podstawowe kontrolki



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





## Konstruktor okna

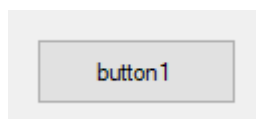
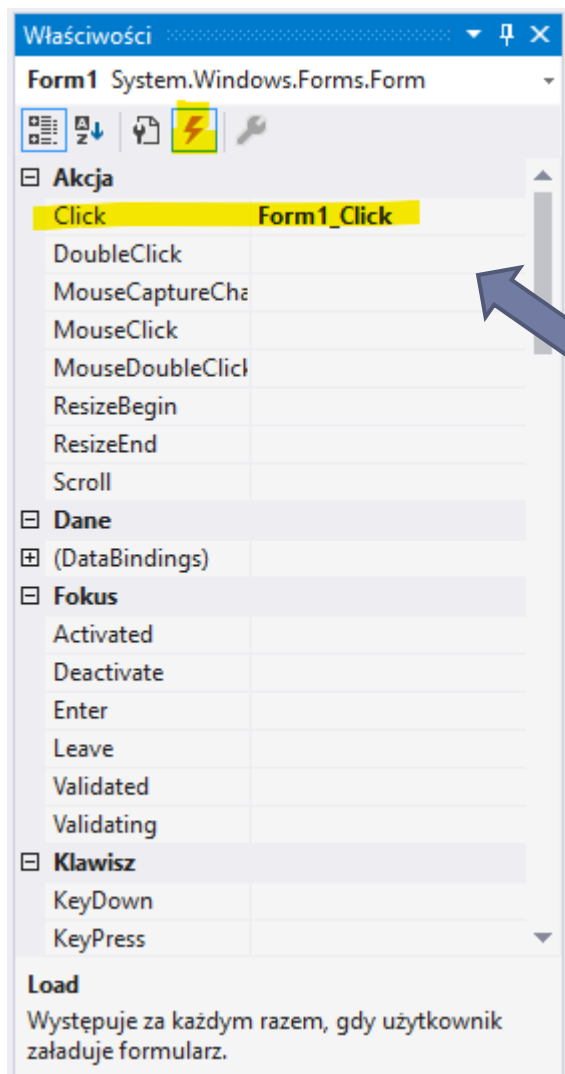
---

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

Aplikacje Windows Forms bazują na zdarzeniach wspieranych przez Microsoft .NET Framework. Oznacza to, że w trakcie działania aplikacji czeka ona na wykonanie przez użytkownika czynności, np. pisanie tekstu do pola tekstowego lub kliknięcie przycisku.



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

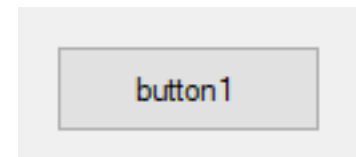
Standardowym (choć jak widać na rysunku po lewej nie jednym) zdarzeniem kontrolki button jest zdarzenie Click – reakcja na kliknięcie

## 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)
        {
        }
    }
}
```

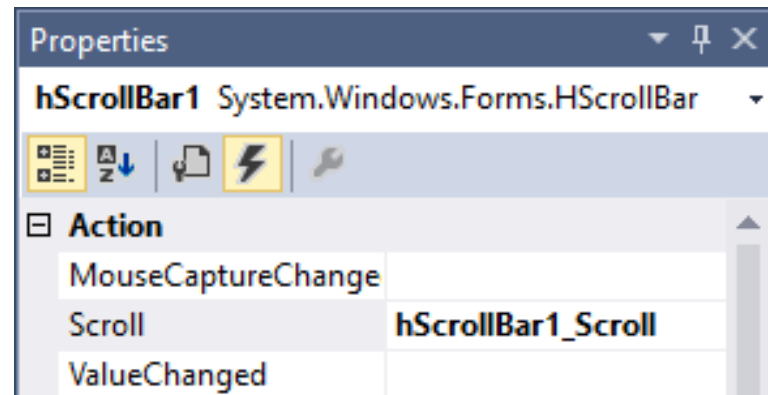


## Zdarzenia – kontrolka ScrollBar



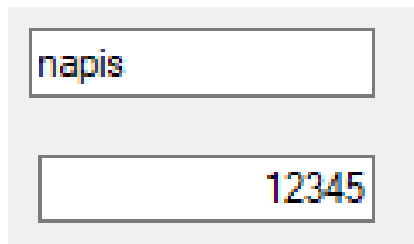
Zdarzenia Kontrolki ScrollBar obsługujemy najczęściej implementując metody:

- **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";
```



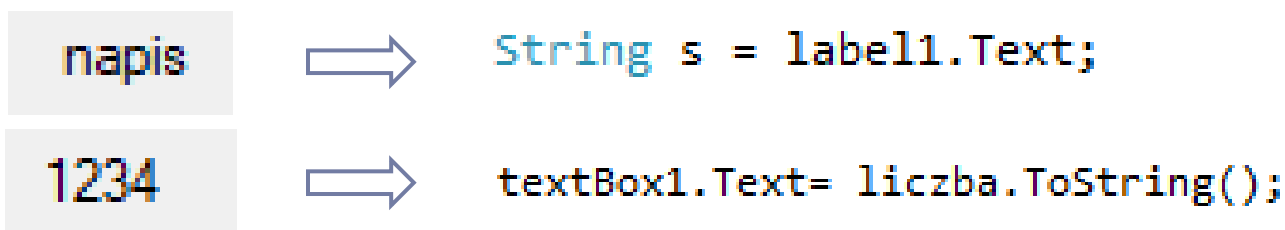
```
textBox1.Text= liczba.ToString();
```



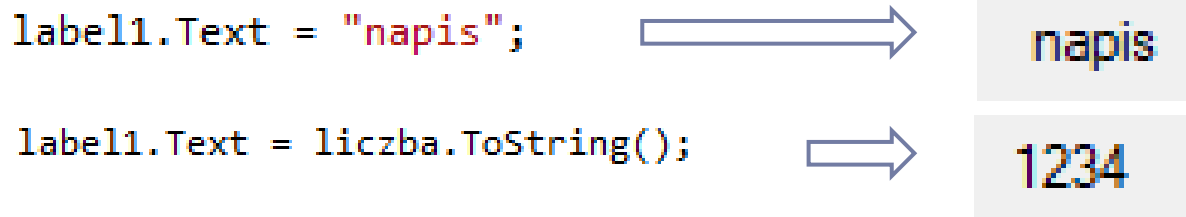
# Label

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

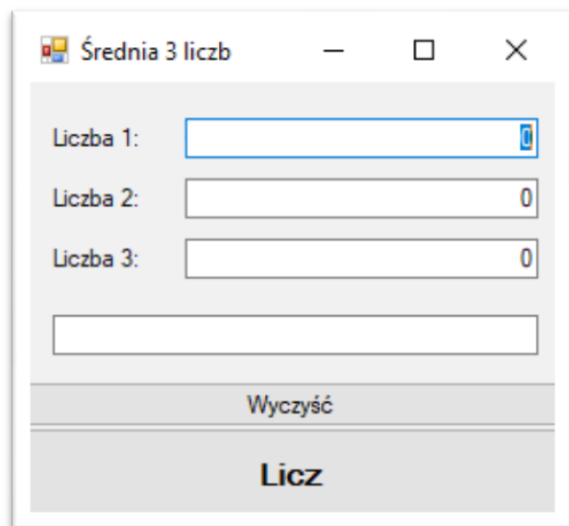
## Odczytanie danych z kontrolki Label



## Zapis danych do kontrolki TextBox



## Przykład



Średnia 3 liczb

Liczba 1:

Liczba 2:  0

Liczba 3:  0

Wyczyść

Licz

## PRZYKŁAD: ŚREDNIA TRZECH LICZB

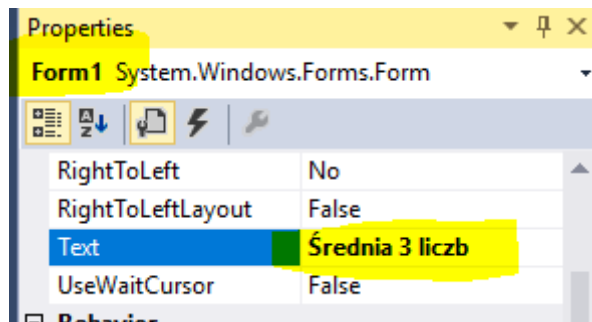
Program wykorzystuje kontrolki:

- ✓ TextBox,
- ✓ Label
- ✓ Button.

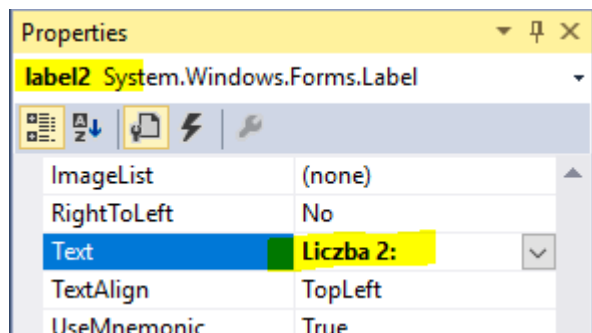


## Przykład

### Kilka przydatnych właściwości kontroltek



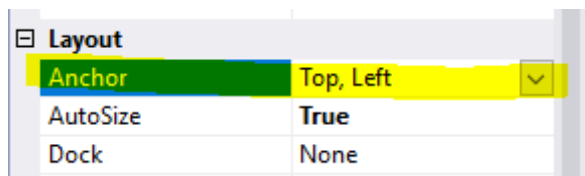
Tytuł okna programu  
(pojawi się na belce)



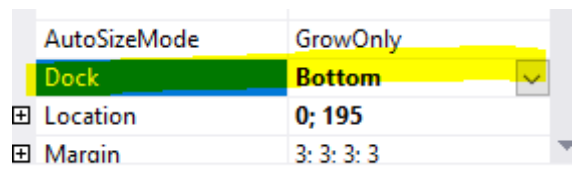
Zawartość wyświetlana  
przez kontrolki Label

Dla kontrolki TextBox działa  
to analogicznie

## Przykład



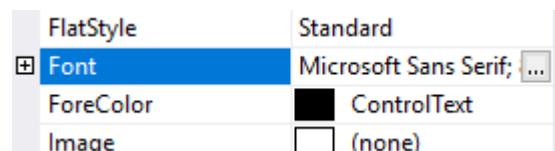
Anchor – pilnuje zakotwiczenia kontrolki – tzn. podczas skalowania okna odległości od sąsiadujących elementów będą zachowywane



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 = "";
}
```



**GUI**

**Przydatne kontrolki**

## Przykład 1

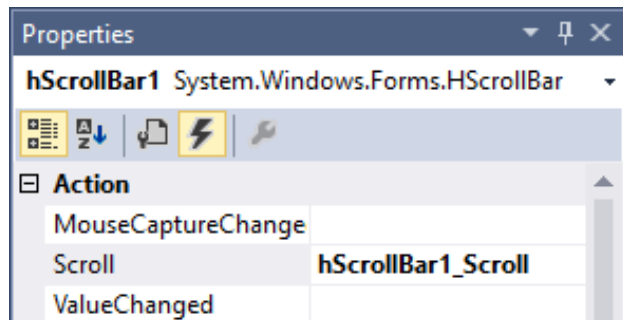


## PRZYKŁAD: TESTER KOLORÓW RGB

Okno programu składa się z kontrolki **Panel**, której kolor będziemy zmieniać oraz z trzech kontroltek **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:

