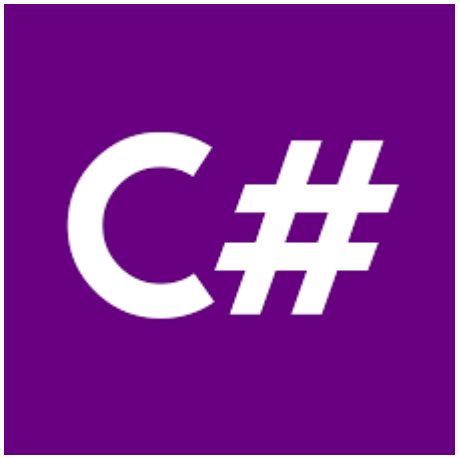
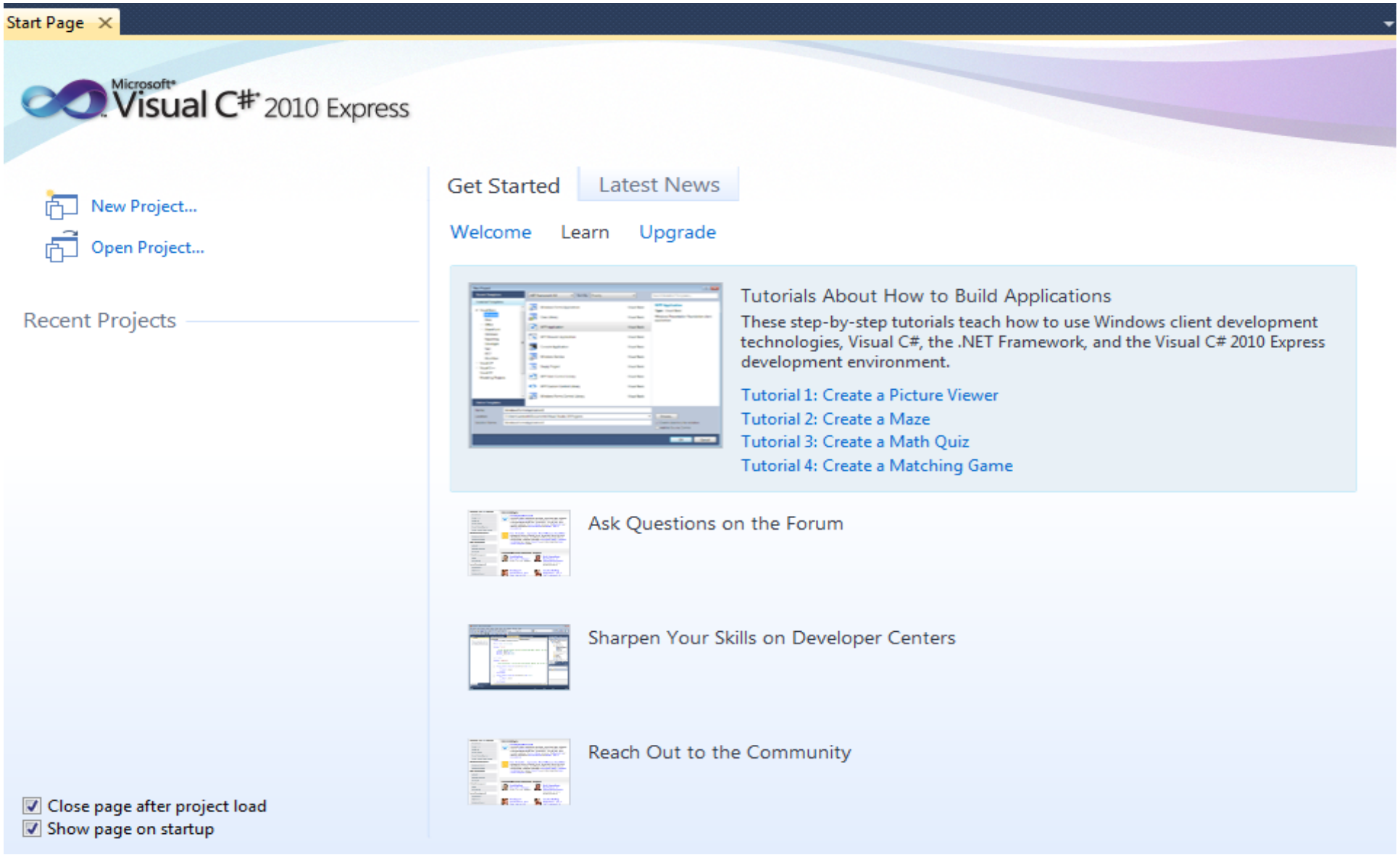


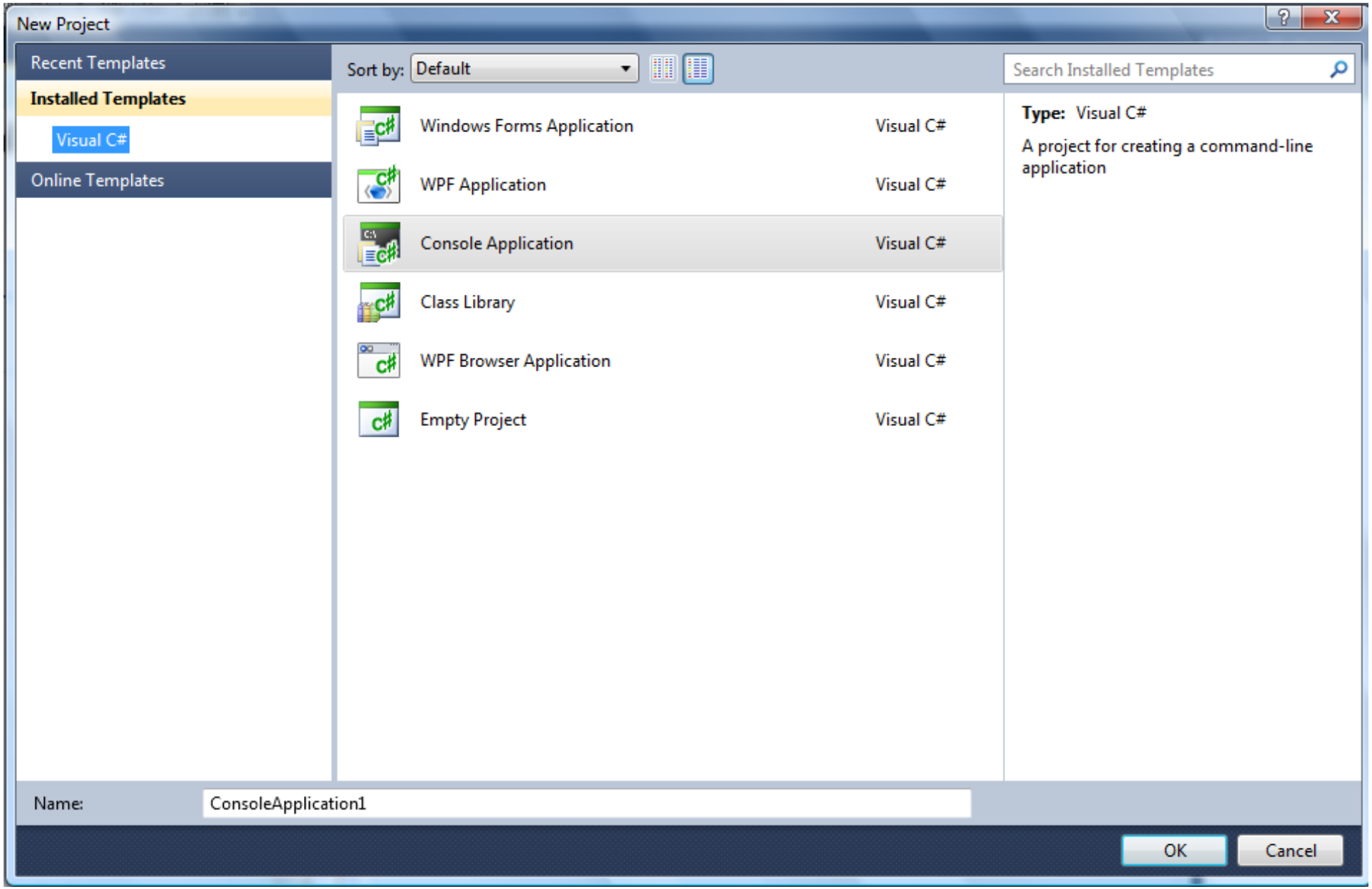
Programowanie Strukturalne i Obiektowe

Paweł Chwietczuk

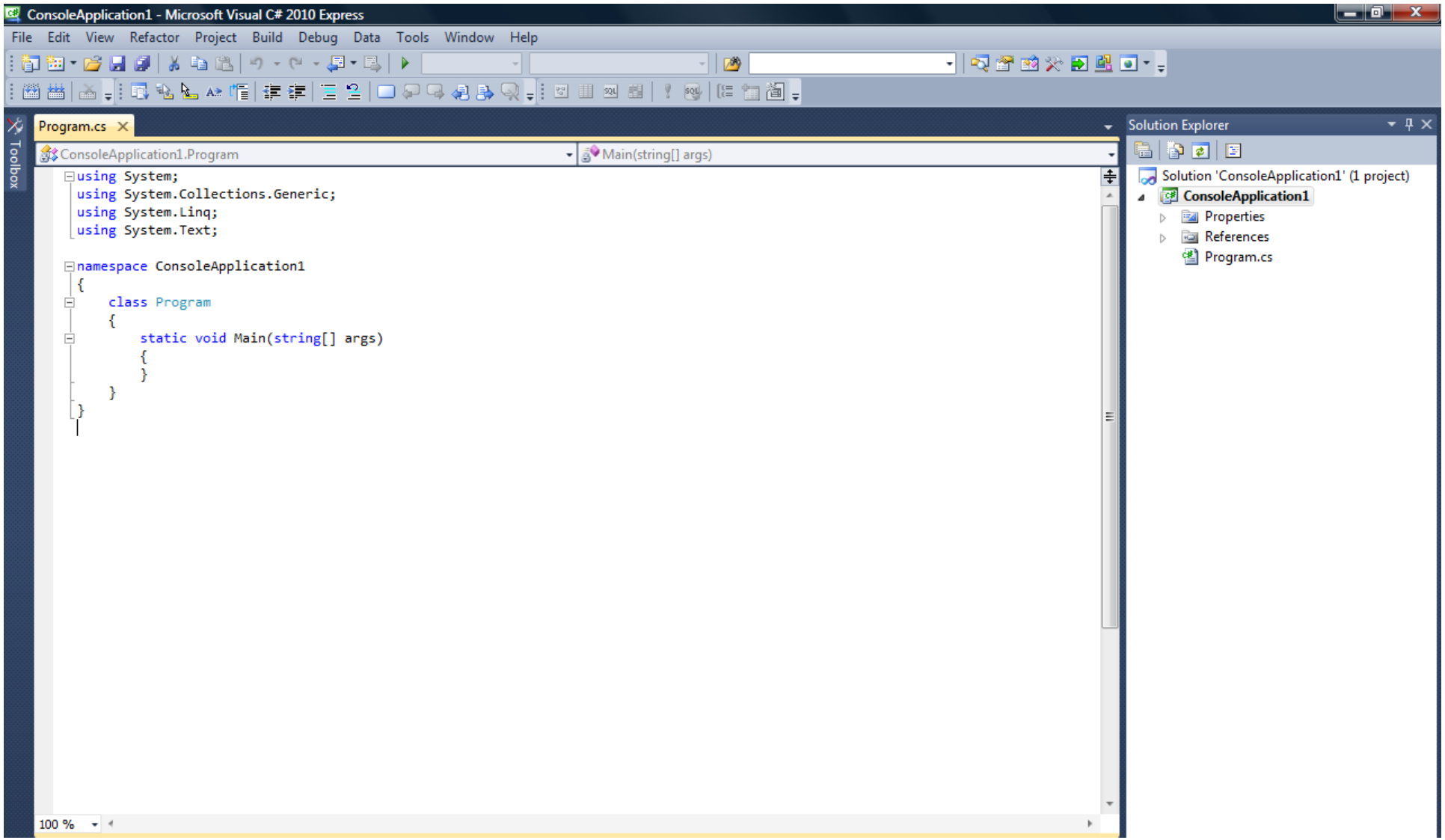
Wprowadzenie do środowiska



Zakładamy nowy projekt



Wygląd nowego pustego projektu



Kod nowego pustego projektu

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            W metodzie Main będziemy pisać instrukcje wykonywane
            przez program
        }
    }
}
```

Komentarze i ich umieszczanie

```
class Program
   {
      static void Main(string[] args)
      {
         // linia pojedynczego komentarza
         /* Komentarz
          * blokowy
          */
      }
   }
```

Instrukcja wypisywania na ekranie

Instrukcja: Console.WriteLine("");

```
class Program
   {
      static void Main(string[] args)
      {
         Console.WriteLine("Witaj jak się masz");
         // instrukcja wyświetla napis: Witaj jak się masz
      }
   }
```

Oczekiwanie na przycisk lub napis

Instrukcja: Console.ReadKey();
Console.ReadLine();
Console.Read();

```
class Program
   {
      static void Main(string[] args)
      {
         Console.WriteLine("Witaj jak się masz");
         // instrukcja wyświetla napis: Witaj jak się masz"
         Console.ReadKey();
         // Oczekiwanie na przycisk
      }
   }
```

Różne sposoby wypisywania

```
 static void Main(string[] args)
{
Console.WriteLine("Linia1 \n Linia2"); // przejście do nowej linii
Console.WriteLine("Wzrost 170 \t Waga 65"); // \t - tabulacja
Console.WriteLine(@"Wzrost 170\tWaga 65"); // @ - dosłowna
    interpretacja
Console.WriteLine("C:\\Windows\\Temp"); // podwójny "\\" wstawia
    "\"
Console.WriteLine("Tytuł filmu: \"Rój\"""); // znak \" wstawia znak "
Console.ReadKey();
}
```

Zadanie

Zbuduj program konsolowy który w jednym wierszu wyświetli twoje Imię a w drugim Nazwisko

Instrukcja wczytywania z klawiatury

```
namespace ConsoleApplication1
{
   class Program
   {
      static void Main(string[] args)
      {
         string a;
         a=Console.ReadLine();
         Console.WriteLine(a);
         Console.ReadKey();
      }
   }
}
```

Typy danych

Typ danych	Zakres
byte	od 0 do 255
sbyte	od –128 do 127
short	od –32,768 do 32,767
int	od –2,147,483,648 do 2,147,483,647
uint	od 0 do 4,294,967,295
long	od –9,223,372,036,854,775,808 do 9,223,372,036,854,775,807
ulong	od 0 do 18,446,744,073,709,551,615
float	od –3.402823e38 do 3.402823e38
double	od –1.79769313486232e308 do 1.79769313486232e308
decimal	od –79228162514264337593543950335 do 79228162514264337593543950335
char	Pojedynczy znak
string	Łańcuch znaków typu char
bool	true lub false

Zmienna typu int

```
class Program
   {
      static void Main(string[] args)
      {
         int i;
         i = 5;
         Console.WriteLine(i);
         Console.ReadKey();

      }
   }
```

Zmienna typu string

ssemblyInfo.cs | Program.cs

ConsoleApplication1.Program | Main(string[] args)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Cześć, jak masz na imię?");

            string name; // deklaracja zmiennej

            name = Console.ReadLine(); // pobranie tekstu wpisanego

            Console.WriteLine("Miło mi " + name );
            Console.ReadLine();
        }
    }
}
```

Operatory arytmetyczne

Operator	Język C#
Dodawanie	+
Odejmowanie	-
Mnożenie	*
Dzielenie rzeczywiste	/
Dzielenie całkowite	/
Reszta z dzielenia	%

Przykłady

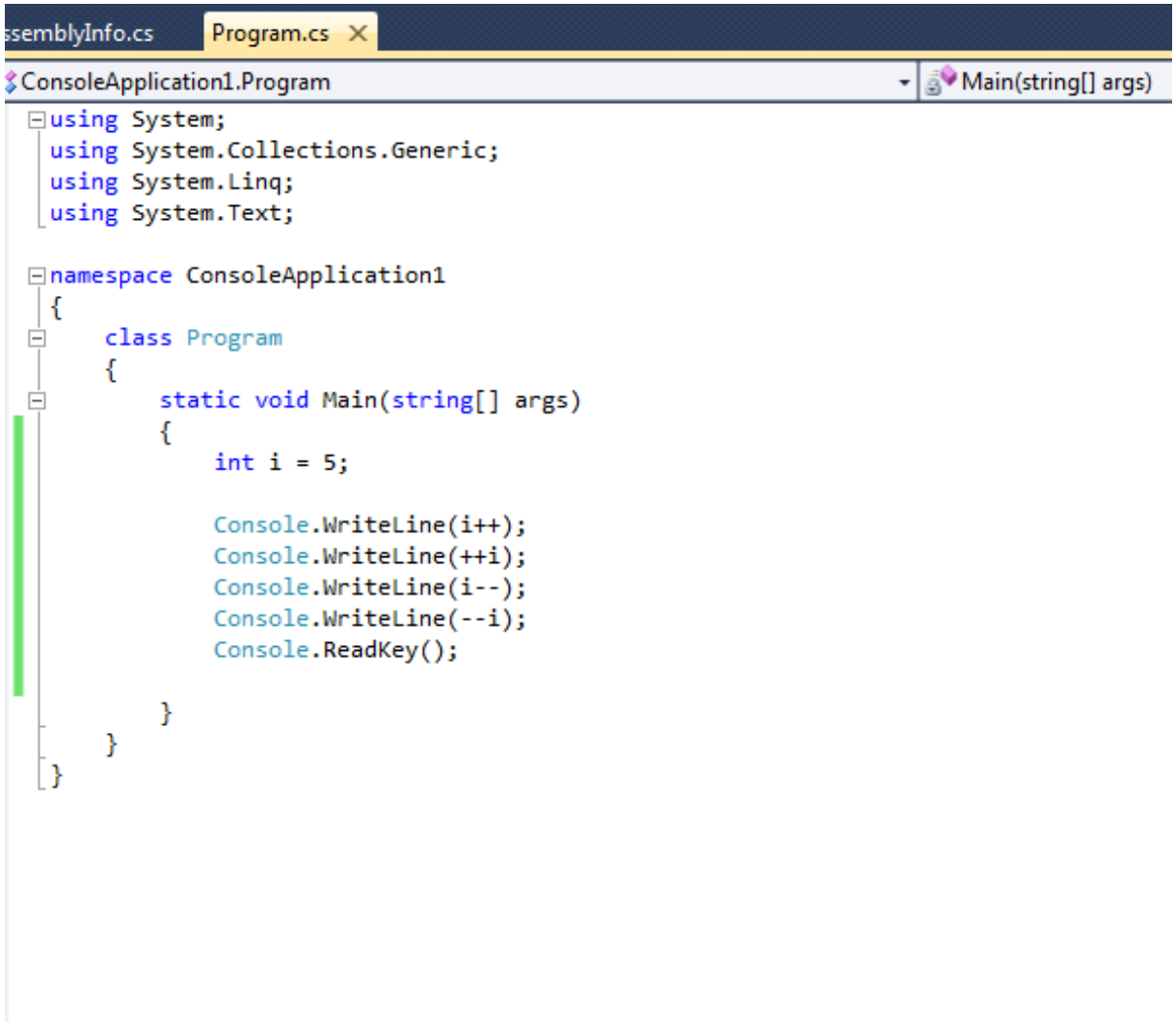
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            double d = 5.0;
            int i = 5;
            Console.WriteLine(5.0 / 5);      // 1
            Console.WriteLine(-i / 2);      // -2
            Console.WriteLine(-d / 2);      // -2.5

            int a, b, wynik;
            a = 3;
            b = 2;
            wynik = 5 * a - 4 * b;
            Console.WriteLine(wynik);
            Console.ReadKey();

        }
    }
}
```

Inkrementacja oraz dekrementacja



Sposoby prezentowania danych

```
static void Main(string[] args)
    {
        int x = 10, y = 25;
        Console.WriteLine("Suma=" + x + " Iloczyn="
+ y);
        Console.WriteLine("Suma={0} Iloczyn={1}", x,
y);
        Console.ReadKey();
    }
```

Sposoby wyświetlania wartości

- *C – formatowanie waluty,*
  D – liczby dziesiętne, określa minimalną ilość cyfr (brakujące wypełnia zerami),
  E – notacja wykładnicza,
  F – formatowanie z ustaloną liczbą miejsc po przecinku,
  G – ogólne formatowanie,
  N – podstawowy format liczbowy,
  X – format heksadecymalny.

Przykład

```
static void Main(string[] args)
{
int liczba = 2014;

Console.WriteLine("Produkt kosztuje {0:C}!", liczba);   //C
Console.WriteLine("Liczba dziesiętna {0:D6}.", liczba); //D
Console.WriteLine("Miejsca po przecinku: {0:F3}.", liczba); //F
Console.WriteLine("System heksadecymalny : {0:X}!",
    liczba);   //X
Console.ReadLine();
}
```

Konwersje typów zmiennych

- x =Convert.ToDouble();
- y= double.Parse();
- z= int.Parse()

- Konwersja dla danych wczytywanych z klawiatury
- x =Convert.ToDouble(Console.ReadLine());
- y= double.Parse(Console.ReadLine());

Operatory porównania

Operator	Język C#
Nierówności	!=
Równości	==
Większości	>
Mniejszości	<
Większe lub równe	>=
Mniejsze lub równe	<=

Operatory logiczne

Operator	Język C#
Logiczne i	&&
Logiczne lub	||
Zaprzeczenie	!

Funkcje logiczne

p	q	Koniunkcja (p && q)	Alternatywa (p || q)	Negacja (!p)
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Rozważmy przykłady

Int a=5

Int b=7

- (a>5&&b>7)
- (a>5||b>7)
- (a>4&&b>7)
- (a>4&&b>6)
- (a>4||b>7)

Przykład

```
static void Main(string[] args)
      {
          int x = 3, y = 10;
          bool wynik;
          wynik = (x >= 2 || y++ >= 2);
          Console.WriteLine(wynik);
          Console.WriteLine(x);
          Console.WriteLine(y);
          Console.ReadKey();
      }
```

Instrukcje warunkowe If i else

If (wyrażenie)

Instrukcja1

else [

Instrukcja 2]

Przykład

```
static void Main(string[] args)
      {
         int a = 12;
         if (a >= 12)
         {
            Console.WriteLine("Wykonano kod z if()!");
         }
         else
         {
            Console.WriteLine("Wykonano kod z else!");
         }
          Console.ReadKey();
      }
```

Uzupełnij znaki zapytania liczbami aby wyświetlony został napis Tak

ConsoleApplication1.Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int x, y, z;

            x = ??;
            y = ??;
            z = ??;

            if ((x == 2 || y == 4) && z > 20)
            {
                Console.WriteLine("Tak");

            }
            Console.ReadKey();
        }
    }
}
```

Sposoby zagnieżdżania

```
static void Main(string[] args)
        {
           int a = 12;
            if (a == 12)
                Console.WriteLine("Liczba jest równa 12");
            if (a > 12)
                Console.WriteLine("Liczba jest większa od 12");
            if (a < 12)
                Console.WriteLine("Liczba jest mniejsza od 12");

          Console.ReadKey();
        }
```

Sposoby zagnieżdżania cd.

```
static void Main(string[] args)
        {
Int a=12;
if (a == 12)
            Console.WriteLine("Liczba jest równa 12");
        else if (a > 12)
            Console.WriteLine("Liczba jest większa od 12");
        else if (a < 12)
            Console.WritaeLine("Liczba jest mniejsza od 12");
    Console.ReadKey();
        }
```

Sposoby zagnieżdżania cd.

```
static void Main(string[] args)
      {
          int a = 12;
          if (a == 12)
             Console.WriteLine("Liczba jest równa 12");
          else
          {
             if (a > 12)
             {
                Console.WriteLine("Liczba jest większa od 12");
             }
             else
             {
                if (a < 12) Console.WriteLine("Liczba jest mniejsza od 12");
             }
          }
Console.ReadKey();
      }
```

```
static void Main(string[] args)
{
Console.WriteLine("Wpisz nr dnia tygodnia");
string numer = Console.ReadLine();
if (numer == "1")
Console.WriteLine("Poniedziałek");
else if (numer == "2")
Console.WriteLine("Wtorek");
else if (numer == "3")
Console.WriteLine("Środa");
else if (numer == "4")
Console.WriteLine("Czwartek");
else if (numer == "5")
Console.WriteLine("Piątek");
else if (numer == "6")
Console.WriteLine("Sobota");
else if (numer == "7")
Console.WriteLine("Niedziela");
else
Console.WriteLine("Nie ma takiego dnia tygodnia");}
```

Zadanie

- Program prosi o podanie temperatury z klawiatury
- Jeżeli temperatura wynosi poniżej 0, program ostrzega użytkownika o występowaniu lodu na drodze.
- Jeżeli temperatura wynosi dokładnie 0, program informuje nas o możliwości występowania oblodzonych fragmentów jezdni.
- W przeciwnym razie wyświetla komunikat o temperaturze dodatniej

Instrukcja switch

Swich (wyrazenie)

{

Case stale_wyrazenie:

Instrukcja

Instrukcja skoku

[default: instrukcja]

}

Przykład

```
static void Main(string[] args)       {
        int lb;
        Console.Write("Podaj liczbę: ");
        lb =int.Parse( Console.ReadLine());
        switch (lb)
        {
           case 0:
              Console.WriteLine("Twoja liczba to 0!");
              break;
           case 1:
              Console.WriteLine("Twoja liczba to 1!");
              break;
           case 2:
              Console.WriteLine("Twoja liczba to 2!");
              break;
           case 3:
              Console.WriteLine("Twoja liczba to 3!");
              break;
           default:
              Console.WriteLine("Podałeś za dużą liczbę!");
              break;          }
        Console.ReadKey();       }
```

```
static void Main(string[] args)
        {
            double cena = 0.0;
            Console.WriteLine("Podaj porcję (S / M / L)");
            string porcja = Console.ReadLine();
            switch (porcja)
            {
                case "S":
                    cena += 4.5;
                    break;
                case "M":
                    cena += 2.0;
                    goto case "S";
                case "L":
                    cena += 3.0;
                    goto case "M";
                default:
                    Console.WriteLine("Podano zły symbol");
                    break;
            }
            Console.WriteLine(cena);
            Console.ReadKey();
        }
```

Zadanie

Napisz prosty kalkulator na case-ach, który będzie potrafił dodawać, odejmować, mnożyć i dzielić. Program ten ma działać następująco:

Pobiera symbol działania.

Pobiera dwie cyfry do działania

Jeżeli wprowadzimy zły symbol program zwraca komunikat

Pętla while

while (wyrażenie logiczne)
{(instrukcje)}
Przykład

```
static void Main(string[] args)
{
int i = 0;
while (i < 20)
{
Console.WriteLine(i);
i++;
}
Console.ReadKey();
}
```

Zadanie

Stosując tylko jdną pętle while i if wypisz na ekranie wszystkie liczby parzyste w przedziale od 3 do 25

Program ma być odporny na zmianę wartości początkowych

Pętla do..while

do {instrukcje} while(warunek)

Przykład

```
static void Main(string[] args)
    {
        int i = 0;

        do
        {
            Console.WriteLine(i);
            i++;

        } while (i < 20);
        Console.ReadKey();
    }
```

Przykład

```
Satic void Main(string[] args)
      {
         string odp = "";
         while (true)
         {
            Console.WriteLine("Wyjść z pętli? [Y/N]");
            odp = Console.ReadLine();
            if (odp == "y" || odp == "Y")
               break;
         }
         Console.ReadKey();
               }
```

Przykład

```
static void Main(string[] args)
      {
         bool warunek = false;
         string zapetlanie = "";
         do
         {
            Console.WriteLine("WYŚWIETLIĆ TEN NAPIS JESZCZE RAZ? [Y/N]");
            zapetlanie = Console.ReadLine();
            if (zapetlanie == "Y" || zapetlanie == "y")
               warunek = true;
            else
               warunek = false;
         } while (warunek);
         Console.ReadKey();
               }
```

Zadanie

Napisz program który będzie prosił o kod dostępu dopóki nie dostanie poprawnego. Gdy go otrzyma wyświetli napis "Dostęp przyznany!".

Pętla for

For

([inicjalizacja];

[wyrażenie];

[iteracja])

instrukcja

Przykład

```
static void Main(string[] args)
    {
        for (int a = 0; a < 10; a++)
       {
         Console.WriteLine("Napis numer " + (a + 1));
       }
         Console.ReadKey();
     }
```

Przykład z instrukcja Write

```
static void Main(string[] args)
      {
         int i ;

         for (i=0; i <= 100; i++)
         {

              Console.Write("{0} ", i);
              if (i % 10 == 0&& i!=0)
              {
                 Console.WriteLine();
              }
         } Console.ReadKey();
         }
```

Ilości powtórzeń

- for (int i = 1 ; i <= 100 ; i++)
- for (int i = 0 ; i < 100 ; i++)
- for (int i = 20 ; i < 120 ; i++)

Zadanie

Napisz program obliczający silnie n!

Program prosi o podanie n z klawiatury

Zadanie

Napisz program, który wyświetla wszystkie naturalne liczby trzycyfrowe, które są podzielne przez 3 i 5

Program wyświetli nam ilość takich cyfr

Pętla foreach

```
static void Main(string[] args)
        {
            int[] tablica = { 1, 2, 3, 4 };
foreach (int x in tablica)
    Console.WriteLine("W tablicy znaleziono
    element {0}", x);
            Console.ReadKey();
            }
```

Tablice jednowymiarowe

```
static void Main(string[] args)
{
int[] uczestnicy = { 19, 34, 23, 54, 31 };
Console.WriteLine(uczestnicy[0]);
Console.WriteLine(uczestnicy[1]);
Console.WriteLine(uczestnicy[2]);
Console.WriteLine(uczestnicy[3]);
Console.WriteLine(uczestnicy[4]);
Console.ReadKey();
}
```

```
static void Main(string[] args)
        {
            int[] tablica;
            int rozmiar;
            Console.WriteLine("Podaj rozmiar tablicy:");
            rozmiar = int.Parse(Console.ReadLine());
            tablica = new int[rozmiar];

            for (int i = 0; i < rozmiar; i++)
            {
                Console.WriteLine("Podaj {0} element tablicy: ", i);
                tablica[i] = int.Parse(Console.ReadLine());
            }
            Console.WriteLine("Zawartość Twojej tablicy to: ");
            foreach (int x in tablica)
                Console.Write(x + ", ");
            Console.ReadLine();
        }
```

Lenght

```
static void Main(string[] args)
        {
            Console.WriteLine("Ile chcesz wpisać imion?");
            int rozmiar = Convert.ToInt32(Console.ReadLine());
            string[] imiona = new string[rozmiar];
            for (int i = 0; i < imiona.Length; i++)
            {
                Console.WriteLine("Podaj {0} imię", i + 1);
                imiona[i] = Console.ReadLine();
            }
            for (int i = 0; i < imiona.Length; i++)
            {
                Console.Write(imiona[i] + ", ");
            }
            Console.ReadKey();
        }
```

Tworzenie tablicy odwrotnej

```
static void Main(string[] args)
{
int[] uczestnicy = new int[] { 19, 34, 23, 54, 31 };
int[] odwrotnie = new int[uczestnicy.Length];
// Wpisywanie elementów do tablicy odwrotnie
for (int i = uczestnicy.Length - 1; i >= 0; i--)
odwrotnie[uczestnicy.Length - i - 1] = uczestnicy[i];
// Wyświetlenie elementów tablicy odwrotnie
for (int i = 0; i < odwrotnie.Length; i++)
Console.WriteLine(odwrotnie[i]);
Console.ReadKey();
}
```

Zadanie

Napisz program, który nada następujące wartości początkowe tablicy 8-elementowej: 1, 2, 5, 8, 9,12,15,21 a następnie wyświetli najpierw wartości parzyste tej tablicy, a następnie nieparzyste.

Tablice wielowymiarowe

```
int tablica0 = new int[5];
```

```
int tablica1 = new int[5];
```

```
int tablica2 = new int[5];
```

```
int tablica3 = new int[5];
```

```
int[,] tablica = new int[4,5];
```

Sposoby deklarowania

```
int[,] tablica = {
                    {12, 32, 24, 245, 32},
                    {35, 36, 36, 87, 62},
                    {92, 57, 95, 45, 38},
                    {0, 100, 99, 42, 38}
                  };
```

```
static void Main(string[] args)
        {
            int[,] tablica;
            int wiersze, kolumny;
            Console.WriteLine("Podaj ilość wierszy: ");
            wiersze = int.Parse(Console.ReadLine());
            Console.WriteLine("Podaj ilość kolumn: ");
            kolumny = int.Parse(Console.ReadLine());
            tablica = new int[wiersze, kolumny];
            for (int x = 0; x < wiersze; x++)           {
                for (int y = 0; y < kolumny; y++)               {
                    Console.WriteLine("Podaj element o wspolrzednych
    [{0}][{1}]:", x, y);
                    tablica[x, y] = int.Parse(Console.ReadLine());
                }           }
            Console.Write("\nZawartość Twojej tablicy: ");
            foreach (int z in tablica)
                Console.Write("{0}, ", z);
            Console.ReadLine();       }
```

Zadanie

Przykład z poprzedniej strony przekształć tak aby zbudować tablice trójwymiarową

Metoda Array.Copy

```
 static void Main(string[] args)
{
int[] a = { 11, 22, 33, 44, 55, 66, 77, 88, 99 };
int[] b = new int[10];
Array.Copy(a, 0, b, 3, 5);
 // kopjuje z a/ zacznij od pozycji w tablicy a=0 skopjuj do tablicy b od
    //pozycji 3 , 5 elementów z a
foreach (int x in b)
{
Console.Write("{0}, ", x);
}
Console.ReadKey();
}
```

Metoda Array.Reverse

```
 static void Main(string[] args)
{int[] tab = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
Array.Reverse(tab);
foreach (int x in tab)
Console.Write("{0,3}", x);
Console.ReadKey();
}
```

Metoda Array.Sort

```
static void Main(string[] args)
{
int[] tab = { 4, 2, 6, 23, 1, 3, 7, 0 };
Array.Sort(tab); // sortowanie tablicy
for (int i = 0; i < tab.Length; i++)
Console.WriteLine(tab[i]);
Console.ReadKey();
}
```

Zadanie

- Mając tablicę 1,8,658,56,52,7,9,15,13,5
- Korzystając z odpowiedniej metody skopuj do nowej tablicy 5 elementów od drugiego włącznie.
- Następnie należy przesortować nową tablicę

Tekst jako tablica znaków

```
static void Main(string[] args)
{
string tekst = "Ala ma kota";
for (int i = tekst.Length - 1; i >= 0; i--)
Console.Write(tekst[i]);
Console.ReadKey();
}
```

Metoda Substring

```
static void Main(string[] args)
{
string tekst = "Ala ma kota";
string fragment;
fragment = tekst.Substring(2, 7);
Console.WriteLine(fragment);
Console.ReadKey();
}
```

Metoda Compare

```
static void Main(string[] args)
{
int wynik;
string tekst1 = "Kowalski";
string tekst2 = "Nowak";
wynik = String.Compare(tekst1, tekst2);
Console.WriteLine(wynik);
Console.ReadKey();
}
```

Metoda Concat

```
static void Main(string[] args)
    {
        string tekst_zlaczony;
        string tekst1 = "Ala ma psa";
        string tekst2 = " i chomika";
        tekst_zlaczony = string.Concat(tekst1,
  tekst2);
        Console.WriteLine(tekst_zlaczony);
        Console.ReadKey();
    }
```

Metody statyczne

```
class Program
   {
      static void Main(string[] args)
      {

      }
      static void nasza_metoda(int x)
      {
      }
   }
```

Definicja metody

[Modyfikatory] Typ Nazwa ([Lista argumentów]) {
[Ciało metody]}

- *Modyfikatory - określają zachowanie i dostępność metody,*
- *Typ – typ danej zwracanej przez metodę,*
- *Nazwa – nazwa metody, różna od nazwy klasy, w której została zdefiniowana*
- *Lista argumentów – lista argumentów przekazywanych do metody*
- *Ciało metody – inaczej treść metody, kod (zbiór instrukcji) realizujący działanie metody.*

Definiowanie własnej metody

```
class Program
   {
      static void dodaj(int x, int y)
      {
         Console.WriteLine("wynik " + (x + y));
      }
      static void Main(string[] args)
      {
         dodaj(4, 3);
         Console.ReadKey();
      }
   }
```

Zadanie

Napisz program dzielący dwie liczby

Dzielenie wykonywane jest w metodzie dziel

Zwracanie wartości przez metodę

```
class Program
{
static double Dziel(double x, int y)
{
return (x / y); // zakładamy, że y jest różne od zera
}
static void Main(string[] args)
{
Console.WriteLine(Dziel(1.5, 3)); // wywołanie metody
Console.ReadKey();
}
}
```

Zadanie

Do programu z poprzedniego slajdu opisz zabezpieczenie uniemożliwiające dzielenie przez 0

Rozwiązanie

```
static double Dziel(double x, int y)
{
double wynik = 0;
if (y != 0)
{
wynik = x / y;
}
return (wynik);
}
```

Przekazywanie argumentów przez wartość

```
class Program
{
static void Dodaj(int a)
{
a++;
Console.WriteLine("Argument z wnętrza metody: " + a);
}
static void Main(string[] args)
{
int x = 5;
Console.WriteLine("Przed wywołaniem metody: " + x);
Dodaj(x);
Console.WriteLine("Po wywołaniu metody: " + x);
Console.ReadKey();
}}
```

Przekazywanie argumentów przez referencje

```
class Program
{
static void Dodaj(ref int a)
{
a++;
Console.WriteLine("Argument z wnętrza metody: " + a);
}
static void Main(string[] args)
{
int x = 5;
Console.WriteLine("Przed wywołaniem metody: " + x);
Dodaj(ref x);
Console.WriteLine("Po wywołaniu metody: " + x);
Console.ReadKey();
}}
```

Przekazywanie i zwracanie tablic

```
class Program    {
      static void Wielkie(string[] tab)       {
         for (int i = 0; i < tab.Length; i++)          {
            tab[i] = tab[i].ToUpper();
         }       }
      static void Main(string[] args)       {
         string[] tab1 = { "jeden", "dwa", "trzy" };
         Wielkie(tab1); // wywołanie metody (tablica
   argumentem)
         for (int i = 0; i < tab1.Length; i++)          {
            Console.Write(tab1[i] + " ");
         }
         Console.ReadKey();
      }   }}
```

Przeciążanie Metod cz.1

```
class Program{
static string OpiszTyp(){
return "Metoda bez argumentów”;}
static string OpiszTyp(int x){
return "Liczba całkowita”;}
static string OpiszTyp(string x){
return "Łańcuch znaków”;}
static string OpiszTyp(double x, int y){
return "Liczba double i liczba całkowita”;}
```

Przeciążanie Metod cz.2

```
static void Main(string[] args)
{
Console.WriteLine(OpiszTyp());
Console.WriteLine(OpiszTyp(20));
Console.WriteLine(OpiszTyp(“aaa 20"));
Console.WriteLine(OpiszTyp(20.45, 20));
Console.ReadKey();
}
}
```

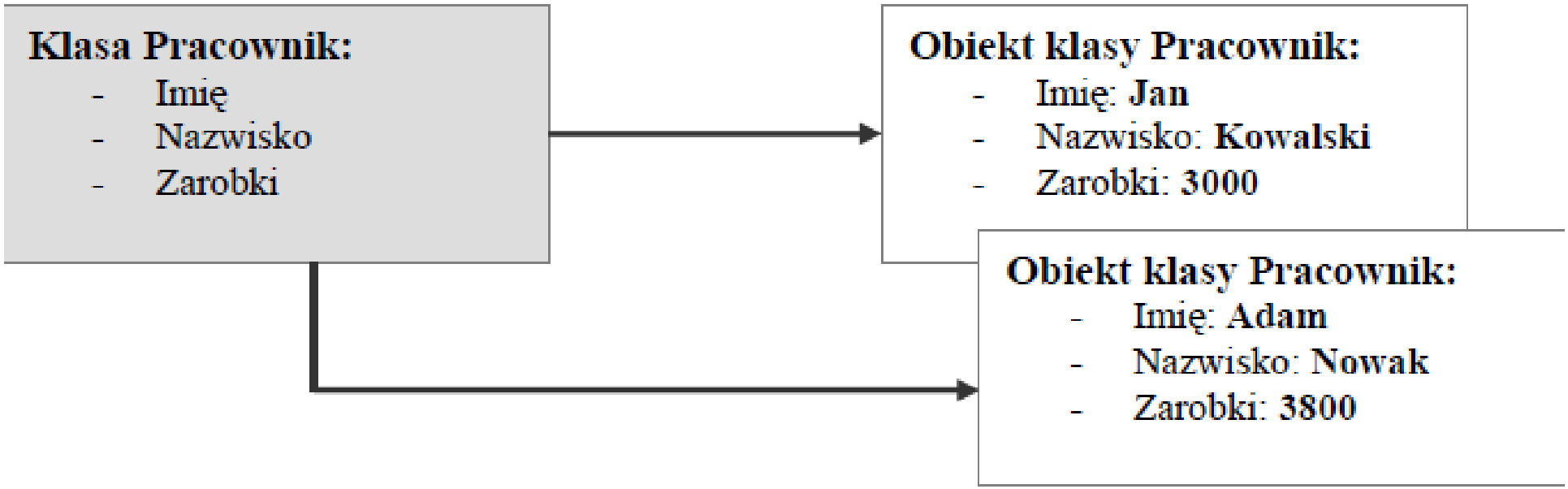
Przeciągnie metody cd.

```
class Program    {
        static int Daj(int x = 0) // argument domyślny
        {
            return x;          }
        static int Daj()
        {
            return -1;
        }
        static void Main(string[] args)
        {
            Console.WriteLine(Daj(2));
            Console.ReadKey();
        }   }
```

Rekurencja

```
static int Silnia(int n)
{
if (n <= 1) return 1;
else return n * Silnia(n - 1);
}
static void Main(string[] args)
{
Console.WriteLine(Silnia(5));
Console.ReadKey();
}
```

Klasa a obiekt



Budowa klasy (*definicja klasy*)

Opis klasy

- *atrybuty,*
- *dostęp do klasy,*
- *rodzaj klasy,*
- *nazwa klasy,*
- *klasa bazowa (interfejsy bazowe).*

Elementy klasy (*ciało klasy*)

Elementy danych

Elementy funkcyjne

Modyfikatory dostępu do klasy

public

- private

protected

- internal

Metoda tworzenia własnej klasy

```
public static class Osoba
{
// ciało klasy
}
```

Przykładowa klasa

```
public class Pracownik{
public string nazwisko;
private double zarobki;
public static int liczbaPracownikow;
public const double etat = 1.0;}
```

- *nazwisko – typu string, do którego jest dostęp publiczny,*
- *zarobki – typu double, do którego jest dostęp prywatny,*
- *liczbaPracownikow – statyczne typu int, do którego jest dostęp publiczny,*
- *etat – stałe typu double, do którego jest dostęp publiczny.*

Definicja klasy *Pracownik*

Opis klasy

```
public class Pracownik
```

Elementy klasy (*ciało klasy*)

Elementy danych:

```
public string nazwisko
private double zarobki
```

Elementy funkcyjne:

```
public Pracownik(string n, double z)
public void PokazPracownika()
```

```
public class Pracownik {
public string nazwisko;
private double zarobki;
public Pracownik(string n, double z) // konstruktor
{
 nazwisko = n;
zarobki = z; }
public void PokazPracownika() // metoda
{
Console.WriteLine("{0,-15} {1,10}", nazwisko, zarobki); } }
class Program {
static void Main(string[] args) {
Pracownik p1 = new Pracownik("Kowalski", 1000);
p1.PokazPracownika();
Console.ReadKey(); }
}
```

```
public class Pracownik {
public string nazwisko;
private double zarobki;
public double Zarobki // właściwość dla pola zarobki {
get { return zarobki; }
set { zarobki = value; } }
public void PokazPracownika() // metoda  {
Console.WriteLine("{0,-15} {1,10}", nazwisko, zarobki); } }
class Program {
static void Main(string[] args) {
Pracownik p1 = new Pracownik();
p1.nazwisko = "Kowalski";
p1.Zarobki = 1250.0; // użycie akcesora set
p1.PokazPracownika();
Console.WriteLine(p1.Zarobki); // użycie akcesora get
Console.ReadKey();  } }
```

Składniki statyczne cz.1

```
public class Pracownik {
public static int liczbaPrac; // pole statyczne
public string Nazwisko { get; set; } // właściwość
public double Zarobki { get; set; } // właściwość
public Pracownik(string naz, double zar) // konstruktor
{
liczbaPrac++;
Nazwisko = naz;
Zarobki = zar; }
static Pracownik() //konstruktor statyczny
{
liczbaPrac = 0; }
public void PokazPracownika() // metoda
{
Console.WriteLine("{0,-15} {1,10}", Nazwisko, Zarobki); } }
```

Składniki statyczne cz.2

```
class Program{
static void Main(string[] args)
{
Console.WriteLine("Liczba {0}",
   Pracownik.liczbaPrac);
Pracownik p1 = new Pracownik("Kowalski", 1250.0);
p1.PokazPracownika();
Pracownik p2 = new Pracownik("Nowak", 1340.0);
p2.PokazPracownika();
Console.WriteLine("Liczba {0}",
   Pracownik.liczbaPrac);
Console.ReadKey();
}}
```

Tablice obiektów cz.1

```
public class Pracownik
{
public string Nazwisko { get; set; } // właściwość
public double Zarobki { get; set; } // właściwość
public Pracownik(string naz, double zar) //
   konstruktor
{
Nazwisko = naz;
Zarobki = zar;
}
```

Tablice obiektów cz.2

```
public void PokazPracownika() // metoda
{
Console.WriteLine("{0,-15} {1,10}", Nazwisko, Zarobki);
}
public static double Sumuj(Pracownik[] tab) // metoda
    statyczna
{
double suma = 0;
for (int i = 0; i < tab.Length; i++)
{
suma += tab[i].Zarobki;
}
return suma;
}}
```

Tablice obiektów cz.3

```
class Program{
static void Main(string[] args)
{
Pracownik[] tab = new Pracownik[3];
tab[0] = new Pracownik("Kowalski", 1250.0);
tab[1] = new Pracownik("Nowak", 1340.0);
tab[2] = new Pracownik("Abacki", 2210.0);
foreach (Pracownik p in tab){
p.PokazPracownika();
}
Console.WriteLine("Suma {0}", Pracownik.Sumuj(tab));
Console.ReadKey();
}}
```

Typ referencyjny cz.1

```
public class MojaKlasa
{
public int Dana { get; set; }
}
class Program
{
static void Main(string[] args)
{
MojaKlasa p1 = new MojaKlasa();
p1.Dana = 5;
MojaKlasa p2 = p1;
Console.WriteLine("p1.Dana = {0}", p1.Dana);
Console.WriteLine("p2.Dana = {0}", p2.Dana);
```

Typ referencyjny cz.2

```
p1.Dana = 8;
Console.WriteLine();
Console.WriteLine("Wartości po zmianie obiektu
   p1");
Console.WriteLine("p1.Dana = {0}", p1.Dana);
Console.WriteLine("p2.Dana = {0}", p2.Dana);
Console.ReadKey();
}
}
```

Struktury

```
static void Main(string[] args){
DateTime t1 = DateTime.Now;
Console.WriteLine("Czas początkowy t1: {0}", t1);
int licznik = 0;
for (int i = 0; i < int.MaxValue; i++)
licznik++;
Console.WriteLine("Wartość zmiennej licznik {0}", licznik);
DateTime t2 = DateTime.Now;
Console.WriteLine("Czas końcowy t2: {0}", t2);
Console.WriteLine("Różnica t2-t1: {0}", t2 - t1);
Console.WriteLine("Dziś jest {0} dzień roku", t1.DayOfYear);
Console.ReadKey();
}
```

Struktury cd.

```
struct Kwadrat{
int bok;
ConsoleColor kolor;
public Kwadrat(int bok1, ConsoleColor kolor1){
bok = bok1;
kolor = kolor1;}
public void RysujKwadrat(){
Console.ForegroundColor = kolor;
for(int i = 1; i <= bok; i++){
for(int j = 1; j <= bok; j++)
Console.Write("*");
Console.WriteLine();}}}
```

Struktury cd..

```
class Program
{
static void Main(string[] args)
{
Kwadrat k1 = new Kwadrat(5, ConsoleColor.Blue);
k1.RysujKwadrat();
Console.ReadKey();
}
}
```

Interfejsy cz.1

```
using System;
namespace InterfaceTest
{
   interface IKucharz    {
      void Gotowanie();
   }
   class TypOsoby1 : IKucharz    {
      int Id { get; set; }
      string Imie { get; set; }
      public TypOsoby1(int id, string imie)       {
         this.Id = id;
         this.Imie = imie;
      }
```

Interfejsy cz.2

```
public void Gotowanie()       {
        Console.WriteLine(this.ToString() + " gotuje...");
    }
    public override string ToString()       {
        return string.Format("{1}", this.Id, this.Imie);
    }   }
  class MainClass    {
    public static void Main(string[] args)       {
        TypOsoby1 adam = new TypOsoby1(1, "Adam");
        adam.Gotowanie();
        Console.ReadKey();
    }   }}
```

Obsługa wyjątków

```
int X, Y, Z;
 X = 10; Y = 0;
 try { Z = X / Y; }
 catch {
 Console.WriteLine("Prosimy nie dzielić przez
    zero!"); }
```

Obsługa wyjątków cz1

```
static void Main(string[] args)
        {
            Console.WriteLine("Podaj dwie liczby");
            try
            {
                string x = Console.ReadLine();
                int xx = int.Parse(x);
                string y = Console.ReadLine();
                int yy = int.Parse(y);
                Console.WriteLine(xx / yy);

            }
```

Obsługa wyjątków cz2

```
catch (FormatException fEx)          {
            Console.WriteLine(fEx.Message);
        }
        catch (OverflowException OverEx)          {
            Console.WriteLine(OverEx.Message);
        }
        catch (ArithmeticException ArgEx)          {
            Console.WriteLine(ArgEx.Message);
        }
        catch (Exception Ex)          {
            Console.WriteLine("Coś poszło nie tak");
        }
        Console.ReadKey();
    }   }}
```

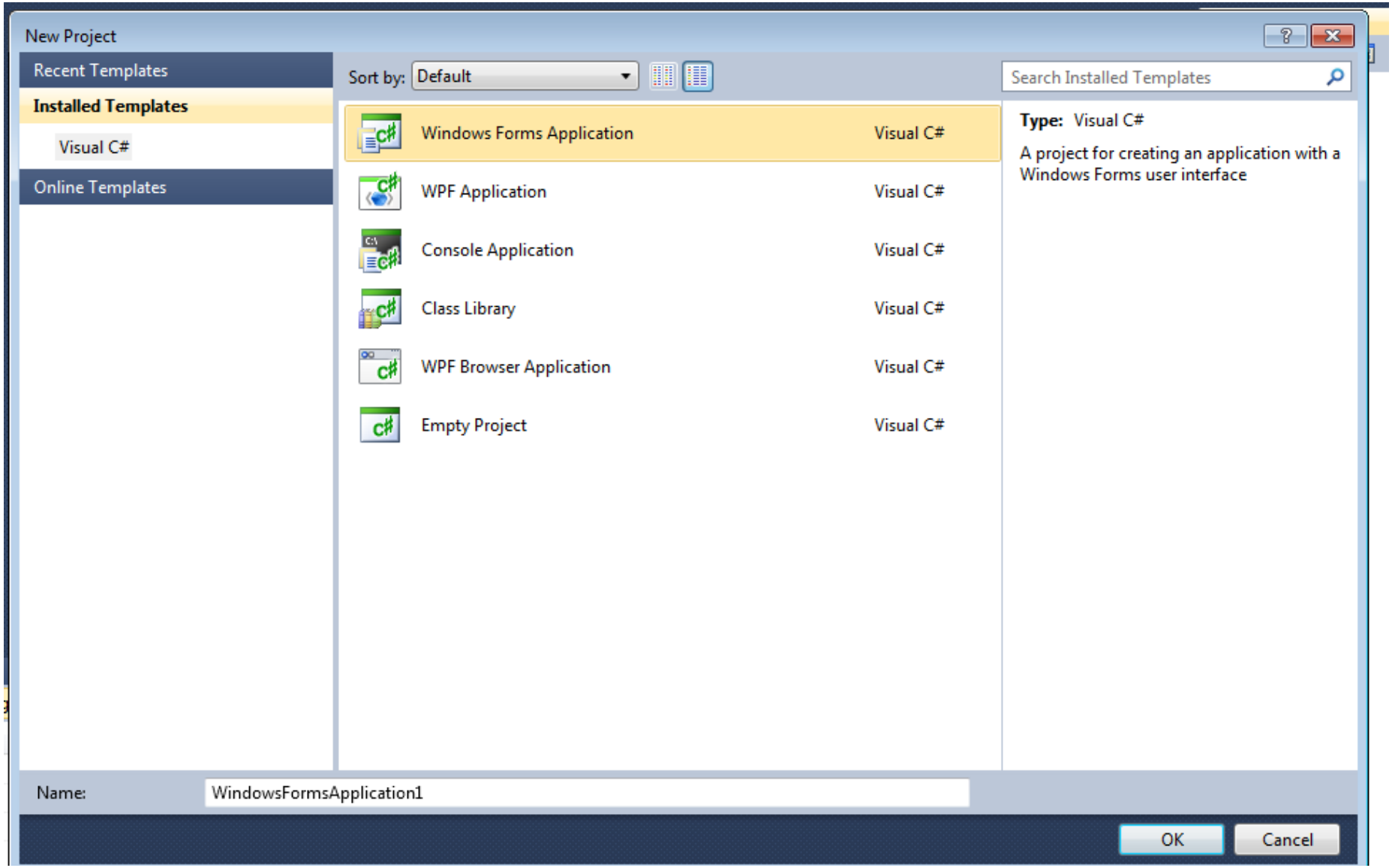
Obsługa cd1..

```
namespace ConsoleApplication1{
    class ExceptionTest    {
      static double SafeDivision(double x, double y)       {
         if (y == 0)
            throw new System.DivideByZeroException();
         return x / y;
      }
      static void Main()
      {
         // Input for test purposes. Change the values to see
         // exception handling behavior.
         double a = 98, b = 1;
         double result = 0;
```

Obsługa cd2..

```
try
        {
           result = SafeDivision(a, b);
           Console.WriteLine("{0} divided by {1} = {2}", a, b, result);
        }
        catch (DivideByZeroException e)
        {
           Console.WriteLine("Attempted divide by zero.");
        }
        Console.ReadKey();
    }   }       }
```

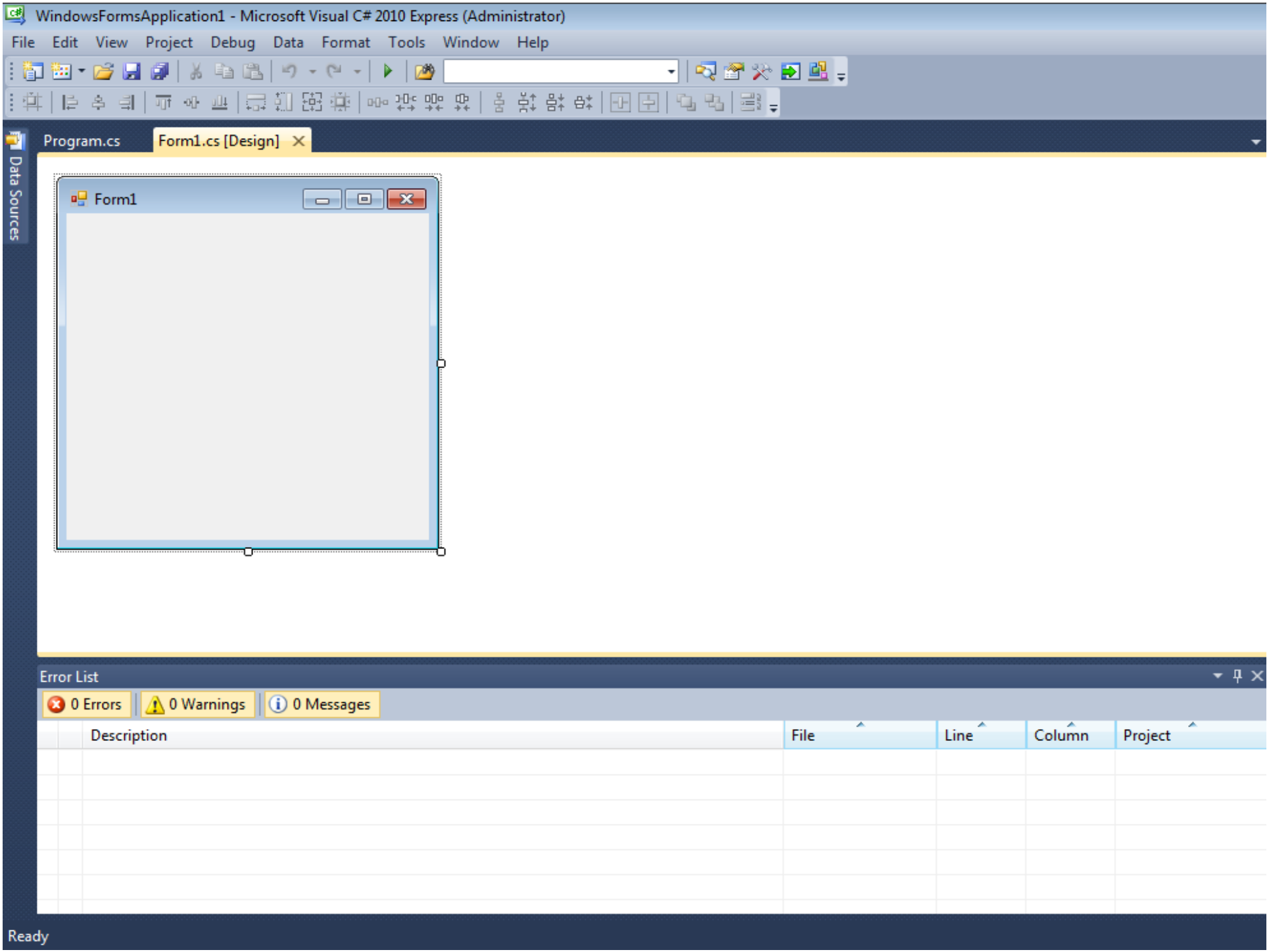
Pierwsze okno



Pusty projekt

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace WindowsFormsApplication1{
   static class Program    {
      /// <summary>
      /// The main entry point for the application.
      /// </summary>
      [STAThread]
      static void Main()       {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new Form1());
      }    }}
```


WindowsFormsApplication1 - Microsoft Visual C# 2010 Express (Administrator)
File Edit View Project Debug Data Format Tools Window Help
Program.cs
Form1.cs [Design]
Data Sources
Form1
Error List
0 Errors
0 Warnings
0 Messages
Description
File
Line
Column
Project
Ready

Wybrane właściwości klasy Form 1

Typ	Nazwa właściwości	Znaczenie
bool	AutoScaleMode	Ustala tryb automatycznego skalowania okna.
bool	AutoScroll	Określa, czy w oknie mają się automatycznie pojawiać paski przewijania.
bool	AutoSize	Określa, czy forma (okno) może automatycznie zmieniać rozmiary zgodnie z trybem określonym przez AutoSizeMode.
AutoSizeMode	AutoSizeMode	Określa tryb automatycznej zmiany rozmiarów formy.
Color	BackColor	Określa aktualny kolor tła.

Wybrane właściwości klasy Form 2

Image	BackgroundImage	Określa obraz tła okna.
Bounds	Bounds	Określa rozmiar oraz położenie okna.
Size	ClientSize	Określa rozmiar obszaru roboczego okna.
ContextMenu	ContextMenu	Określa powiązane z oknem menu kontekstowe.
Cursor	Cursor	Określa rodzaj kursora wyświetlanego, kiedy wskaźnik myszy znajdzie się nad oknem.
Font	Font	Określa rodzaj czcionki, którą będzie wyświetlany tekst znajdujący się w oknie.
Color	ForeColor	Określa kolor używany do rysowania obiektów w oknie (kolor pierwszoplanowy).

Wybrane właściwości klasy Form 3

FormBorderStyle	FormBorderStyle	Ustala typ ramki okalającej okno.
int	Height	Określa wysokość okna.
Icon	Icon	Ustala ikonę przypisaną do okna.
int	Left	Określa w pikselach położenie lewego górnego rogu w poziomie.
Point	Location	Określa współrzędne lewego górnego rogu okna.
MainMenu	Menu	Menu główne przypisane do okna.
bool	Modal	Decyduje, czy okno ma być modalne.
string	Name	Określa nazwę okna.
Control	Parent	Referencja do obiektu nadrzędnego okna.

Wybrane właściwości klasy Form 4

bool	ShowInTaskbar	Decyduje, czy okno ma być wyświetlane na pasku narzędziowym.
Size	Size	Określa wysokość i szerokość okna.
String	Text	Określa tytuł okna (tekst na pasku tytułu).
int	Top	Określa w pikselach położenie lewego górnego rogu w pionie.
bool	Visible	Określa, czy okno ma być widoczne.
int	Width	Określa w pikselach szerokość okna.
FormWindowState	WindowState	Reprezentuje bieżący stan okna.

Określanie rozmiaru okna

```
using System;
using System.Windows.Forms;

public class MainForm : Form{
   public MainForm()    {
      this.Width = 300;
      this.Height = 100;
      this.Text = "Tytuł okna”;    }
   static void Main()    {
      Application.Run(new MainForm());
   }}
```

Wyświetlanie komunikatów

```
using System;
using System.Windows.Forms;

public class Aplikacja
{
    public static void Main()
    {
        MessageBox.Show("Przykładowy teks");
    }
}
```

Zadanie

- Zbuduj swoje okno o wysokości 500, szerokości 800, oraz nazwie okna Imię i Nazwisko
- Przed oknem głównym zostanie wyświetlony komunikat „Witaj”

Okno dialogowe przed oknem głównym

```
using System;
using System.Windows.Forms;

public class Aplikacja
{
   public static void Main()
   {
      MessageBox.Show("Przykładowy teks");
      Application.Run(new Form());
   }
}
```

Obsługa zamykania aplikacji

```
using System;
using System.Windows.Forms;

public class MainForm : Form{
   Button button = new Button();
   public MainForm()    {
      Application.ApplicationExit += new
    EventHandler(this.onExit);
   }
   private void onExit(object sender, EventArgs ea)    {
      MessageBox.Show("Aplikacja zostanie zamknięta !");    }
   public static void Main()    {
      Application.Run(new MainForm());
   }}
```

Etykiety (Label)

Wybrane właściwości klasy Label

Typ	Nazwa właściwości	Znaczenie
bool	AutoSize	Określa, czy etykieta ma automatycznie dopasowywać swój rozmiar do zawartego w niej tekstu.
Color	BackColor	Określa kolor tła etykiety.
BorderStyle	BorderStyle	Określa styl ramki otaczającej etykietę.
Bounds	Bounds	Określa rozmiar oraz położenie etykiety.

Cursor	Cursor	Określa rodzaj kursora wyświetlanego, kiedy wskaźnik myszy znajdzie się nad etykietą.
Font	Font	Określa rodzaj czcionki, którą będzie wyświetlany tekst znajdujący się na etykiecie.
Color	ForeColor	Określa kolor tekstu etykiety.
int	Height	Określa wysokość etykiety.
Image	Image	Obraz wyświetlany na etykiecie.
int	Left	Określa położenie lewego górnego rogu w poziomie, w pikselach.
Point	Location	Określa współrzędne lewego górnego rogu etykiety.

sting	Name	Nazwa etykiety.
Control	Parent	Referencja do obiektu zawierającego etykietę (nadrzędnego).
Size	Size	Określa wysokość i szerokość etykiety.
string	Text	Określa tekst wyświetlany na etykiecie.
ContentAlignment	TextAlign	Określa położenie tekstu na etykiecie.
int	Top	Określa położenie lewego górnego rogu w pionie, w pikselach.
bool	Visible	Określa, czy etykieta ma być widoczna.
int	Width	Określa rozmiar etykiety w poziomie.

Wartości typu FontStyle

Nazwa	Znaczenie
Bold	tekst pogrubiony
Italic	tekst pochylony
Regular	tekst zwyczajny
Strikeout	tekst przekreślony
Underline	tekst podkreślony

Okno z etykietą

```
using System;
using System.Windows.Forms;
public class MainForm : Form {
Label label = new Label(); public MainForm(){
this.Width = 320;
this.Height = 200;
this.Text = "Okno z etykietą";
label.Text = "Przykładowa etykieta...";
label.AutoSize = true;
label.Left = (this.ClientSize.Width - label.Width) / 2;
label.Top = (this.ClientSize.Height - label.Height) / 2;
   this.Controls.Add(label);}
public static void Main(){
       Application.Run(new MainForm());    }}
```

Wykorzystanie własnego kroju czcionki

```
using System.Windows.Forms;
using System.Drawing;
public class MainForm : Form {
Label label = new Label();
public MainForm(){
this.Width = 500;
this.Height = 200;
this.Text = "Okno z etykietą";
label.Font = new Font("Courier", 30, FontStyle.Bold);
label.Text = "Czcionka Courier";
label.AutoSize = true; this.Controls.Add(label);
label.Left = (this.ClientSize.Width - label.Width) / 2;
label.Top = (this.ClientSize.Height - label.Height) / 2;}
public static void Main(){
Application.Run(new MainForm());}}
```

Zadanie

- Zbuduj okno o rozmiarze 500x500
- Przy użyciu etykiety wyświetlamy swoje Imię i Nazwisko (czcionka Arial rozmiar 40)

Wybrane właściwości klasy Button 1

Typ	Nazwa właściwości	Znaczenie
Color	BackColor	Określa kolor tła przycisku.
Bounds	Bounds	Określa rozmiar oraz położenie przycisku.
Cursor	Cursor	Określa rodzaj kursora wyświetlanego, kiedy wskaźnik myszy znajdzie się nad przyciskiem.
FlatStyle	FlatStyle	Modyfikuje styl przycisku.
Font	Font	Określa rodzaj czcionki, którą będzie wyświetlany tekst znajdujący się na przycisku.

Wybrane właściwości klasy Button 2

Color	ForeColor	Określa kolor tekstu przycisku.
int	Height	Określa wysokość przycisku.
Image	Image	Obraz wyświetlany na przycisku.
int	Left	Określa położenie lewego górnego rogu w poziomie, w pikselach.
Point	Location	Określa współrzędne lewego górnego rogu przycisku.
string	Name	Nazwa przycisku.
Control	Parent	Referencja do obiektu zawierającego przycisk (obiektu nadrzędnego).
Size	Size	Określa wysokość i szerokość przycisku.

Wybrane właściwości klasy Button 3

Typ	Nazwa właściwości	Znaczenie
string	Text	Tekst wyświetlany na przycisku.
Content ↳Alignment	TextAlign	Określa położenie tekstu na przycisku.
int	Top	Określa położenie lewego górnego rogu w pionie, w pikselach.
bool	Visible	Określa, czy przycisk ma być widoczny.
int	Width	Określa rozmiar przycisku w poziomie, w pikselach.

Umieszczanie przycisku a oknie aplikacji

```
using System;
using System.Windows.Forms;
public class MainForm : Form{
      Button button = new Button();
   public MainForm()    {
            this.Width = 320;
            this.Height = 200;
            this.Text = "Okno z przyciskiem";
            button.Top = 60;
            button.Left = 120;
            button.Text = "Kliknij mnie";
            this.Controls.Add(button);          }
      public static void Main()    {
      Application.Run(new MainForm());
            }}
```

Zadanie

- Do poprzedniego programu dopisz obsługę kliknięcia i zamknięcia aplikacji.

Rozwiazanie

```
button.Text = "Kliknij mnie";
        EventHandler eh = new
    EventHandler(this.CloseClicked);
         button.Click += eh;
         this.Controls.Add(button);
       }
       public void CloseClicked(Object sender, EventArgs e)
       {
          this.Close();
       }
       public static void Main()    {
       Application.Run(new MainForm());
```

Pole tekstowe (TextBox) 1

Typ	Nazwa właściwości	Znaczenie
bool	AutoSize	Określa, czy pole tekstowe ma automatycznie dopasowywać swój rozmiar do zawartego w nim tekstu.
Color	BackColor	Określa kolor tła pola tekstowego.
Image	BackgroundImage	Obraz znajdujący się w tle okna tekstowego.
BorderStyle	BorderStyle	Określa styl ramki otaczającej pole tekstowe.
Bounds	Bounds	Określa rozmiar oraz położenie pola tekstowego.
Cursor	Cursor	Rodzaj kursora wyświetlanego, kiedy wskaźnik myszy znajdzie się nad polem tekstowym.
Font	Font	Określa rodzaj czcionki, którą będzie wyświetlany tekst znajdujący się w polu.

Pole tekstowe (TextBox) 2

Color	ForeColor	Określa kolor tekstu pola tekstowego.
int	Height	Określa wysokość pola tekstowego.
int	Left	Określa położenie lewego górnego rogu w poziomie, w pikselach.
string[]	Lines	Tablica zawierająca poszczególne linie tekstu zawarte w polu tekstowym.
Point	Location	Określa współrzędne lewego górnego rogu pola tekstowego.
int	MaxLength	Maksymalna liczba znaków, które można wprowadzić do pola tekstowego.
bool	Modified	Określa, czy zawartość pola tekstowego była modyfikowana.
bool	Multiline	Określa, czy pole tekstowe ma zawierać jedną, czy wiele linii tekstu.
string	Name	Nazwa pola tekstowego.
Control	Parent	Referencja do obiektu zawierającego pole tekstowe (obiektu nadrzędnego).

Pole tekstowe (TextBox) 3

Typ	Nazwa właściwości	Znaczenie
`Char`	`PasswordChar`	Określa, jaki znak będzie wyświetlany w polu tekstowym w celu zamaskowania wprowadzanego tekstu; aby skorzystać z tej opcji, właściwość `Multiline` musi być ustawiona na `false`.
`bool`	`ReadOnly`	Określa, czy pole tekstowe ma być ustawione w trybie tylko do odczytu.
`string`	`SelectedText`	Zaznaczony fragment tekstu w polu tekstowym.
`int`	`SelectionLength`	Liczba znaków w zaznaczonym fragmencie tekstu.
`int`	`SelectionStart`	Indeks pierwszego znaku zaznaczonego fragmentu tekstu.

Pole tekstowe (TextBox) 4

`Size`	`Size`	Określa rozmiar pola tekstowego.
`string`	`Text`	Tekst wyświetlany w polu tekstowym.
`Content` ↪`Alignment`	`TextAlign`	Określa położenie tekstu w polu tekstowym.
`int`	`Top`	Określa położenie lewego górnego rogu w pionie, w pikselach.
`bool`	`Visible`	Określa, czy pole tekstowe ma być widoczne.
`int`	`Width`	Określa rozmiar pola tekstowego w poziomie.
`bool`	`WordWrap`	Określa, czy słowa mają być automatycznie przenoszone do nowej linii, kiedy nie mieszczą się w bieżącej; aby zastosować tę funkcję, właściwość `Multiline` musi być ustawiona na `true`.

Obsługa pola tekstowego 1

```
using System;
using System.Windows.Forms;
public class MainForm : Form {
Button button = new Button();
TextBox textBox = new TextBox();
public MainForm(){
this.Width = 320;
 this.Height = 200;
this.Text = "Aplikacja";
button.Top = 120;
button.Left = (this.ClientSize.Width - button.Width) / 2;
button.Text = "Kliknij mnie";
```

Obsługa pola tekstowego 2

```
textBox.Top = 60;
textBox.Left = (this.ClientSize.Width - textBox.Width) / 2;
EventHandler eh = new EventHandler(this.ButtonClicked);
        button.Click += eh;
        this.Controls.Add(button);
        this.Controls.Add(textBox);
}
public void ButtonClicked(Object sender, EventArgs e){
MessageBox.Show(textBox.Text);
}
public static void Main(){
Application.Run(new MainForm());
}}
```

Sposób użycia pola tekstowego 1

```
using System;
using System.Drawing;
using System.Windows.Forms;
public class MainForm : Form{
   Button button = new Button();
   Label label = new Label();
   TextBox textBox = new TextBox();
   public MainForm()    {
      this.Width = 320;
      this.Height = 200;
      this.Text = "Apli kacja";
      button.Top = 60;
   button.Left = (this.ClientSize.Width - button.Width) / 2;
      button.Text = "Kliknij mnie";
```

Sposób użycia pola tekstowego 2

```
label.Top = 30;
        label.Text = "Etykieta";
        label.TextAlign = ContentAlignment.MiddleCenter;
        label.Left = (this.ClientSize.Width - label.Width) / 2;
        textBox.Top = 120;
        textBox.Left = (this.ClientSize.Width - textBox.Width) / 2;
        EventHandler eh = new EventHandler(this.ButtonClicked);
        button.Click += eh;
        this.Controls.Add(button);
        this.Controls.Add(label);
        this.Controls.Add(textBox);    }
    public void ButtonClicked(Object sender, EventArgs e)    {
        label.Text = textBox.Text;
        label.Left = (this.ClientSize.Width - label.Width) / 2;    }
    public static void Main()    {
        Application.Run(new MainForm());    }}
```

Pole wyboru cz.1.

Nazwa właściwości	Typ	Znaczenie
AutoCheck	bool	Określa, czy pole ma być automatycznie zaznaczane lub odznaczane, kiedy użytkownik kliknie je myszą.
BackColor	Color	Określa kolor tła pola.
Bounds	Bounds	Określa rozmiar oraz położenie pola.
Checked	bool	Pozwala stwierdzić, czy pole jest zaznaczone.
CheckState	CheckState	Określa sposób zaznaczania pola.
Cursor	Cursor	Rodzaj kursora wyświetlanego, kiedy wskaźnik myszy znajdzie się nad polem.
FlatStyle	FlatStyle	Określa styl, w jakim pole będzie wyświetlane.
Font	Font	Określa rodzaj czcionki, którą będzie wyświetlany tekst znajdujący się przy polu.

Pole wyboru cz.2.

ForeColor	Color	Określa kolor tekstu pola.
Height	int	Określa wysokość pola.
Left	int	Określa położenie lewego górnego rogu w poziomie, w pikselach.
Location	Point	Określa współrzędne lewego górnego rogu pola.
Name	sting	Nazwa pola.
Parent	Control	Referencja do obiektu zawierającego pole
Size	Size	Określa wysokość i szerokość pola.
Text	string	Tekst wyświetlany przy polu.
TextAlign	Content ↪Alignment	Określa położenie tekstu znajdującego się przy polu.
ThreeState	bool	Określa, czy pole ma być dwu- czy trójstanowe.
Top	int	Określa położenie pola w pionie, w pikselach.
Visible	bool	Określa, czy pole ma być widoczne.
Width	int	Określa rozmiar pola w poziomie.

Obsługa CheckBox 1

```
using System;
using System.Drawing;
using System.Windows.Forms;

public class MainForm : Form {
Button button = new Button();
CheckBox chbl, chb2, chb3;
   public MainForm()
{

this.Width = 320;
this.Height = 200;
this.Text = "Pola wyboru";
button.Top = 120;
button.Left = (this.ClientSize.Width - button.Width) /2;
button.Text = "Kliknij mnie";
```

Obsługa CheckBox 2

```
chbl = new CheckBox(); chbl.Left = 120; chbl.Top = 20;
chbl.Text = "CheckBox nr l";
chb2 = new CheckBox(); chb2.Left = 120; chb2.Top = 40;
chb2.Text = "CheckBox nr 2";
chb3 = new CheckBox(); chb3.Left = 120; chb3.Top = 60;
chb3.Text = "CheckBox nr 3";
EventHandler eh = new EventHandler(this.ButtonClicked);
button.Click += eh; this.Controls.Add(button);
this.Controls.Add(chbl);
this.Controls.Add(chb2);
this.Controls.Add(chb3);}
```

Obsługa CheckBox 3

```
public void ButtonClicked(Object sender,EventArgs
    e){
string s1 = "", s2 = "", s3 = "";
if(chbl.Checked){
s1 = " 1 ";}
if(chb2.Checked){
s2 = " 2 ";}
if(chb3.Checked){
    s3 = " 3 ";}
MessageBox.Show("Zostały zaznaczone opcje: " + s1
    + s2 +s3);}
public static void Main(){
        Application.Run(new MainForm());}}
```

Obsługa pól wyboru typu radio 1

```
using System;
using System.Drawing;
using System.Windows.Forms;
public class MainForm : Form {
Button button = new Button();
RadioButton rb1, rb2, rb3; public MainForm(){
this.Width = 320;
this.Height = 200;
this.Text = "Pola wyboru";
button.Top = 120;
button.Left = (this.ClientSize.Width - button.Width) /2;
    button.Text = "Kliknij mnie'";
```

Obsługa pól wyboru typu radio 2

```
rb1 = new RadioButton();
rb1.Left = 120;rb1.Top = 20;
rb1.Text = "Opcja nr 1";
rb2 = new RadioButton(); rb2.Left = 120;
rb2.Top = 40; rb2.Text = "Opcja nr 2";
rb3 = new RadioButton(); rb3.Left = 120;
   rb3.Top = 60; rb3.Text = "Opcja nr 3";
EventHandler eh = new EventHandler(this.ButtonClicked);
button.Click += eh;
this.Controls.Add(button);
this.Controls.Add(rb1);
this.Controls.Add(rb2);
this.Controls.Add(rb3);}
```

Obsługa pól wyboru typu radio 3

```
public void ButtonClicked(Object sender, EventArgs e){
string s1 = "", s2 = "", s3 = "";
   if(rb1.Checked){
s1 = " 1 ";}
else if(rb2.Checked)
{s2 = " 2 ";}
else if(rb3.Checked){
       s3 = " 3 ";}
MessageBox.Show("Zaznaczona została opcja: " + s1 + s2
    +s3);}
public static void Main(){
   Application.Run(new MainForm());}}
```

Memu

```
using System;
using System.Windows.Forms;
public class MainForm : Form{
    MainMenu mainMenu = new MainMenu();
        MenuItem menuItem1 = new MenuItem("Plik");
    MenuItem menuItem2 = new MenuItem("Zamknij");
    public MainForm()    {
        this.Text = "Okno z menu";
        this.Width = 320;
        this.Height = 200;
        mainMenu.MenuItems.Add(menuItem1);
        menuItem1.MenuItems.Add(menuItem2);
        this.Menu = mainMenu;    }
    public static void Main()    {
        Application.Run(new MainForm());    }}
```

Zadanie

- Utwórz menu z opcją Plik i Edycja
- Każde menu ma posiadać po 3 podmenu

Obsługa pola zamknij

```
using System;
using System.Windows.Forms;
public class MainForm : Form    {
   MainMenu mainMenu = new MainMenu();
MenuItem menuIteml;
MenuItem menuItem2; public MainForm(){
this.Text = "Okno z menu";
   this.Width = 320; this.Height = 200;
menuIteml = mainMenu.MenuItems.Add("Plik");
EventHandler eh = new EventHandler(this.CloseClicked);
   menuItem2 = menuIteml.MenuItems.Add("Zamknij", eh);
   this.Menu = mainMenu;}
public void CloseClicked(Object sender, EventArgs e){
this.Close();}
public static void Main(){
Application.Run(new MainForm());}}
```