PODSTAWY PROGRAMOWANIA OBIEKTOWEGO W JĘZYKU C#

NOTATKA

```
//====DEKLAROWANIE ZMIENNYCH=====================================
        int calkowita = 1;
        bool prawdaFalsz = true;
        float ulamek = 0.5f;
        string ciagZnakow = "lubie placki";
        char znak = 'a';

//====WYPISYWANIE ZA POMOCĄ KONSOLI==============================
        Console.WriteLine();
        //Wypisuje pustą linię i przechodzi do kolejnej

        Console.WriteLine(calkowita);
        //Wypisuje zawartość zmiennej 'calkowita'

        Console.WriteLine(ciagZnakow);
        //Wypisuje zawartość zmiennej 'ciagZnakow'

        Console.WriteLine("nie lubie placków");
        //Wypisuje tekst wpisany wprost jako argument funkcji

        Console.Write(ciagZnakow);
        //Wypisuje zawartość 'ciagZnakow' ale bez przejścia do kolejnej linijki

        Console.WriteLine(ciagZnakow + "jakis tekst " + ulamek + "znou coś" + calkowita);
        //Można łączyć tekst do wypisania w jednym poleceniu za pomocą +

        Console.Write("nie lubie plackow \n tekst w nowej linijce");
        //Znak specjalny \n powoduje przejście do następnej linijki w miejscu w którym został użyty

//====WPROWADZANIE ZA POMOCĄ KONSOLI=============================

        ciagZnakow = Console.ReadLine();
        //Funkcja Console.ReadLine(); czeka na tekst wprowadzony przez użytkownika i
        //zatwierdzenie enterem. Tekst zostaje umieszczony (znak =) w zmiennej ciagZnakow.

        calkowita = int.Parse(Console.ReadLine());
        //Funkcja Parse przekształca ciąg znaków zwracany przez ConsoleReadLine()
        //na zmienną liczbową (na której można wykonywać operacje matematyczne)

        ulamek = float.Parse(Console.ReadLine());
        //analogicznie do int.Parse.

        bool czyKonwersjaSieUdala = int.TryParse(Console.ReadLine(), out calkowita);
        //funkcja TryParse próbuje wykonać to co robi Parse
        //ale w przypadku gdy użytkownik wpisze bezsensowne dane
        //funkcja int.Parse zepsuje program. Funkcja int.TryParse
        //zamieni się na false (bool) i powiadomi o tym program.

        Console.ReadLine();
        //Funkcja czeka na wciśnięcie entera.

        Console.ReadKey();
        //Funkcja czeka na wciśnięcie jakiegokolwiek klawisza
```

```
//==========OPERACJE MATEMATYCZNE==============================
        float liczba1 = 2.0f;
        float liczba2 = 5.0f;
        float wynik = 0.0f;

        wynik = liczba1 + liczba2;  //Dodawanie
        wynik = liczba1 - liczba2;  //Odejmowanie
        wynik = liczba1 * liczba2;  //Mnożenie
        wynik = liczba1 / liczba2;  //Dzielenie

        wynik = ((liczba1 / liczba2) + (liczba1 * liczba2)) / 3.0f;
        //Kolejność wykonywania działań tak jak w matematyce

//== RZUTOWANIE ==============================================

        int wynikCalkowity = (int)(liczba1 / liczba2);
        //Chcąc wpakować wynik ułamkowy, powstały po liczba1/liczba2,
        //w pudełko dla liczby całkowitej, stracimy bezpowrotnie część ułamkową!
        //Musimy o tym specjalnie powiadomić kompilator i własnoręcznie uciąć
        //część ułamkową za pomocą rzutowania (int)(rzutowana wartość).
        //Niejako osobiście przeciskamy ułamek przez sito dla liczb całkowitych.
//========== INSTRUKCJE WARUNKOWE ==============================
        bool prawda = true;
        bool falsz = false;
        float liczba = 5.0f;

        //======= PODSTAWOWA KONSTRUKCJA IF / ELSE ======================
        if(prawda)
        {
           //Jeśli warunek w nawiasie jest prawdą, wykonaj instrukcję między tymi klamrami
        }
        else
        {
           //w przeciwnym razie wykonaj instrukcje zawarte tutaj
        }

        //===WIELOWARIANTOWA KONSTRUKCJA IF/ELSE ====================
        if (liczba > 5.0f)
        {
        //Jeśli liczba >5.0f to wykonaj instrukcje między klamrami i nie sprawdzaj poniższych zapytań if.
        }
        else if (liczba == 5.0f)
        {
           //Jeśli powyższy if się nie wykonał, sprawdź warunek tego ifa. jeśli on się spełni,
           //wykonaj to co jest tutaj i nie sprawdzaj kolejnych 'if' poniżej
        }
        else if (liczba == 5.0f)
        {
           //Jeśli żaden z powyższych ifów nie został wykonany, sprawdź tego
        }
        else
        {
           //Jeśli totalnie nic z powyższych się nie sprawdziło, wykonaj instrukcje zawarte tutaj.
        }
```

//========== OPERATORY LOGICZNE ==========

```
// && - iloczyn logiczny
// || - suma logiczna
// !  - zaprzeczenie
// == - rowne
//Różnica między == a = to: = przypisuje wartość do zmiennej a == ją porównuje.
// <  - mniejsze od
// >  - większe od
// <= - mniejsze lub równe
// >= - większe lub równe
// != - nie równe

if(prawda == true && liczba == 5.0f)
{
   //Jeśli prawda jest utawiona na true a liczba wynosi równo
   //Obydwa warunki muszą zostać spełnione jednocześnie
}

if (prawda == true || liczba == 15.0f)
{
   //Jeśli prawda ustawiona jest na true lub liczba wynosi równo 15.0f
   //Tylko jeden z warunków musi zostać spełniony. Drugi może być fałszem
}

if (liczba< 10.0f && liczba > 0.1f)
{
   //Jeśli liczba jest mniejsza od 10 i jednocześnie większa od 0.1
}

if (prawda != false)
{
   //Jeśli prawda nie będzie fałszem (będzie prawdą) wtedy warunek zostanie spełniony
}
```

//======INSTRUKCJA WARUNKOWA SWITCH ==========

```
// W przeciwieństwie do if/else tutaj porównujemy ze stałymi wartościami, np '10', '40.31f'
// a nie ze zmiennymi.Switch dzięki temu jest szybszy od if/else

switch (liczba)
{
   case 10:
      //Jeśli liczba wynosi 10, wtedy wykonaj instrukcje zawarte między case a break  (czyli tu);
      Console.WriteLine("Liczba wynosi dziesięć więc wykonuję instrukcję");
      break;
   case 20:
      //Jeśli liczba wynosi 20... to samo
      break;
   case 40.31f:
      //Analogicznie do poprzednich
      break;
   default:
      //Jeśli liczba nie spełni żadnego z powyższych warunków
      break;
}
```

```
//===== PĘTLA WHILE ORAZ DO/WHILE ================================
        while (prawda == true)
        {
           //Pętla będzie wykonywała instrukcje zawarte między jej klamrami w nieskończoność,
           //o ile tylko zmienna 'prawda' ma wartość 'true'.

           //Jeśli przy wejściu w pętlę while, prawda ma wartość false, pętla nie wykona się ani razu
           //i zostanie ominięta przez program.
        }

        do
        {
           //Ta pętla sprawdza warunek dopiero po wykonaniu. Zatem nawet jeśli prawda manwartość false,
           //pętla wykona się przynajmniej 1 raz zakończy się dopiero po sprawdzeniu warunku
           //- na dole przy while(warunek).
        } while (prawda == true);

        //======= INSTRUKCJA BREAK ======================================
        while (prawda == true)
        {
           liczba += 1;    //Co każde powtórzenie się pętli, do liczby (równej 5), zostanie dodane 1.

           if(liczba>100)  //W momencie gdy liczba osiągnie wartość większą niż 100
           {
              break; //Pętla zostanie przerwana, mimo że jej warunek 'prawda' nadal ma wartość true.

              //podobny efekt można uzyskać ustawiając w tym miejscu prawdę na false:
              prawda = false;
           }
        }

        //=========================PĘTLA FOR ============================
        //=================Z INKREMENTACJĄ ZMIENNEJ „i” ================

        for (int i=0; i<10; i++)
        {
           //Instrukcje zawarte w tym miejscu zostaną wykonana 10 razy.
           //Wewnątrz tej pętli istnieje zmienna lokalna 'i', która za każdym powtórzeniem pętli jest
           //zwiększana o 1 'i++', <- tzw inkrementacja.
           //Pętla powtarza się jeśli tylko warunek i<10 jest spełniony. Potem zostaje przerwana
           Console.WriteLine("I wynosi teraz: " + i);
        }

        //========== PODOBNA PĘTLA, TYLE ŻE WHILE ZAMIAST FOR============

        int licznik = 0;
        while(licznik<10) //Ta pętla wykona się tak samo jak pętla for powyżej
        {
           licznik++;
           Console.WriteLine("I wynosi teraz: " + licznik);
        }
```

```
//==========Z DEKREMENTACJĄ ==================================

for (int i = 10; i > 0; i--)
{
 //To samo co powyżej, tyle że tutaj pętla odlicza zaczynajac od 10 i zmniejszając i o jeden.
 //Dopóki i jest większe od zera (i>0), funkcja będzie się powtarzać.
   Console.WriteLine("I wynosi teraz: " + i);
}
```

//======ZAGNIEŻDŻONE PĘTLE FOR ==================================

```
for(int y=0; y<10; y++)
{
   for(int x=0; x<10; x++)
   {
      //Pętle można zagnieżdżać, odwzorowując tak jakby dwa wymiary (lub więcej).
      //W konsekwencji, wewnętrzna pętla zostanie wykonana 100 razy
      //(po 10 razy dla każdej iteracji - powtórzenia - pętli zewnętrznej. 10*10 = 100.
   }
}

//=======NIESKOŃCZONA PĘTLA FOR I INSTRUKCJA BREAK ============
for (; ; )
{
   // Brak warunków ? nic nie szkodzi. pętla będzie wykonywać się w nieskończoność
   // chyba że napotka break;
   if(prawda == false)
   {
      break; // instrukcja break jest w stanie przerwać nieskończoną pętlę, w której się znajduje
   }
}

//============PĘTLA FOR Z INSTRUKCJĄ CONTINUE ==================
for (int i = 10; i > 0; i--)
{
   if(i==5)
   {
      continue; // Jeśli i wyniesie w którymś momencie 5, wtedy pomiń wszystko co znajduje
                //się pod instrukcją continue i powróć do początku pętli.

      Console.WriteLine("To nie zostanie wypisane na ekran!");
                //dlatego ta instrukcja jest wyblakła.
   }
}
```

```
//======================TABLICE=====================================
        int[] tablicaLiczbCalkowitych = new int[10];
        // Deklaracja tablicy - tworzy tablicę 10 elementową.
        //Każdy element tablicy ma swój 'numerek' nazywany indexem.
        // Elementy numerowane są od zera! czyli w tablicy mamy elementy ponumerowane
        //jako: 0,1,2,3,4,5,6,7,8,9 (nie ma elementu z indeksem 10!)

        float[] tablicaUlamkow = new float[10];
        // Tak samo tworzymy tablicę10 floatów.

        string[] tablicaWyrazow = new string[10];
        // Analogicznie, deklarujemy tablicę, która ma 10 miejsc na wyrazy.

        bool[] tablicaBooli = new bool[10];
        char[] tablicaZnakow = new char[10];

        //Początkowo każda tablica zapełniona jest zerami.

        //Można jednak od razu podczas tworzenia tablicy, wrzucić do nie jakieś wartości:
         int[] tablicaLiczbCalkowitychZapelniona = { 1, 4, 5,};
        // tworząc tablicę, od razu wrzucamy do niej dane.
        // Tablica widzi, że wrzucamy do niej 3 liczby
        //więc automatycznie ustawia się jako tablica 3 elementowa z indeksami: 0, 1, 2.

        //Wpisywanie danych do tablicy.
        //Używając tablicy, myśl o komórkach w excellu.

        tablicaLiczbCalkowitych[0] = 1;
        // Wstawia do pierwszej komórki tablicy liczb całkowitych wartość 1.
        // (pierwszy element tablicy ma index [0])

        tablicaUlamkow[0] = 0.5f;
        // Wstawia do pierwszej komórki tablicy ułamków wartość 0.5

        tablicaWyrazow[5] = "Szósty element tablicy wyrazów :D";
        //Wstawia do szóstego elementu (którego indeks to [5]) tekst.

        //Odczytywanie z tablic
        Console.WriteLine(tablicaLiczbCalkowitych[0]);
        //Wypisuje do konsoli zawartość pierwszego elementu tablicy liczb całkowitych

        Console.WriteLine(tablicaWyrazow[5]);
        //Wypisuje do konsoli zawartość szóstego elementu tablicy wyrazów

        //Iteracja po elementach tablicy

        //Początkowo wszystkie elementy tablicy mają wartość "0".
        for(int i=0; i<10; i++)
        {
            Console.WriteLine(tablicaLiczbCalkowitych[i]);
            //Pętla wypisze na ekran wszystkie elementy tablicy liczb całkowitych (czyli same zera)
        }
```

```
//Iteracja z wpisywaniem wartości
for(int i=0; i<10; i++)
{
   tablicaLiczbCalkowitych[i] = 5;
   // Pętla, która powtarza się 10 razy i zapełnia całą tablicę liczb całkowitych liczbą 5.
   // zmienna "i" została wykorzystana do wskazania,
   // który element tablicy chcmy zmienić przy danym powtórzeniu pętli.
}

//Jak spytać program o długość naszej tablicy?
Console.WriteLine(tablicaLiczbCalkowitych.Length);
// Wypisze na ekran "10" bo ilość elementów tablicy jednowymiarowej to 10.
//Możemy użyć właściwości Length, którą posiada każda tablica
// i zapisać poprzedniego fora tak:

for(int i=0; i<tablicaLiczbCalkowitych.Length; i++)
   // Pętla zadziała tak: "Zaczynając od 0 a kończąc na
   // [długość tablicyLiczbCalkowitych, czyli 10],
  // zwiększaj licznik o jeden i wykonuj pętlę.
{
   tablicaLiczbCalkowitych[i] = 1;
   // Przy każdym powtórzeniu pętli użyj zmiennej i do określenia,
   // w który element tablicy chcemy wrzucić wartość
}

//Do iterowania po elementach tablicy można użyć pętli foreach.
foreach(int LiczbaCalkowita in tablicaLiczbCalkowitych)
{
   Console.WriteLine(LiczbaCalkowita);
   // Ta pętla wypisze nam wszystkie elementy tablicy. For each
   // czyli - dla każdego elementu tablicy (wykonaj jakąś czynność)
}

foreach(int LiczbaCalkowita in tablicaLiczbCalkowitych)
{
   tablicaLiczbCalkowitych[LiczbaCalkowita] = 3;
   // Ta pętla zapełni całą tablicę liczbami 3
}

//Zapełnianie tablicy liczbami losowymi:
Random rand = new Random();
foreach (int LiczbaCalkowita in tablicaLiczbCalkowitych)
{
   tablicaLiczbCalkowitych[LiczbaCalkowita] = rand.Next(1, 10);
   // Tablica zostanie wypełniona losowymi liczbami z zakresu 1-10
}
```

```
//========== TABLICE DWUWYMIAROWE ==============================
        int[,] tablicaCalkowitych2D = new int[5,5];
        //Deklarujemy tablicę, której wymiary to 5 x 5 (jak pole minowe w "saperze")
        /*
          [] [] [] [] []
          [] [] [] [] []
          [] [] [] [] []
          [] [] [] [] []
          [] [] [] [] []

        */
        string[,] tablicaWyrazow2D = new string[2,5];
        //Deklarujemy tablicę wyrazów o wymiarze 5 x 2.
        /*
          [] []
          [] []
          [] []
          [] []
          [] []

        */

        //Iterowanie po tablicach dwuwymiarowych

        for(int y=0; y<tablicaCalkowitych2D.GetLength(1);y++)
           // pobierz wymiar y tablicy - GetLenghth(1)
        {
           for (int x = 0; x < tablicaCalkowitych2D.GetLength(0); x++)
              // pobierz wymiar x tablicy - GetLenghth(0)
           {
                 Console.WriteLine(" Element o współrzędnych: " + x + " , " + y + "
                                         ma wartość: " + tablicaCalkowitych2D[x, y]);
           }
        }
```

```
//==========================FUNKCJE=========================
        //========= funkcja nie zwracająca, bez argumentów==========
        void funkcjaNieZwracajacaWartosci()
        {
           Console.WriteLine("Funkcja coś robi i nic nie zwraca.");
        }

        //Od teraz, możemy używać sobie naszej funkcji wywołując ją:
        funkcjaNieZwracajacaWartosci();

        //======================== funkcja nie zwracająca, z argumentem=========
        void funkcjaPrzyjmujacaArgument(int liczba)
        {
           Console.WriteLine(liczba);
        }

        //Do funkcji możemy przesłać argumenty i wywołać ją tak:
        funkcjaPrzyjmujacaArgument(5); // Funkcja wydrukuje nam na ekranie 5.

        //==== funkcja nie zwracająca, z wieloma argumentami
        void funkcjaPrzyjmujacaArgumenty(int liczba, string zdanie, bool niewiemCo)
        {
           Console.WriteLine(liczba);
           Console.WriteLine(zdanie);
           Console.WriteLine(niewiemCo);
        }

        //======================== funkcja zwracająca wartość i pobierająca argumenty
        float obliczPoleTrojkata(float podstawa, float wysokosc)
        {
           float poleTrojkata = (podstawa * wysokosc) / 2.0f;

           return poleTrojkata;
           // Polecenie return mówi, w którym miejscu funkcja ma zakończyć działanie
           // i 'zamienić się' w daną wartość, tutaj będzie to poleTrojkata.
        }

        //Funkcję obliczPoleTrojkata wywołujemy tak:
        float poleJakiegosTrojkata = obliczPoleTrojkata(10.3f, 40.1f);
        // Od tego miejsca, zmienna poleJakiegosTrojkata otrzyma
        // informacje zwrotną z funkcji, czyli wynik jej działania.



        //==== funkcja mnożąca wszystkie elementy tablicy razy określoną wielkość i zwracająca ją
        static void funkcjaPrzyjmujacaArgumenty(float[] tablicaWejsciowa, float mnoznik)
        {
                for (int i = 0; i < tablicaWejsciowa.Length; i++)
                {
                        tablicaWejsciowa[i] *= mnoznik;
                }

        }
```