Lekcja 2 - funkcje

## **Zadanie z lekcji**

Napisać program kalkulator. Użytkownik podaje 2 liczby oraz wpisuje symbol operatora lub funkcji: +,-,\*,/,%,&,|,^,POW,SIN,COS. Następnia program wyświetla wynik operacji. Funkcje 1 parametrowe ignorują parametr y. Działania mają odbywać się na liczbach typu double. Dla każdego operatora/funkcji zbudować osobną funkcję w C# która będzie wywoływana na podstawie danych wprowadzonych przez użytkownika.

\*\*Napisać kalkulator działający wg. następującego algorytmu:

Na wejściu mamy pojedynczy napis będący kolejnymi liczbami lub operatorami (2 argumentowe) oddzielonymi znakami spacji, np. 5 6 \* 3 +

Postępować wg. algorytmu:

1. Podziel napis na tokeny (używając funkcji Split)
2. Dla każdego tokena wykonuj operacje a-c
   1. Jeśli token jest liczbą: dodaj go na stos
   2. Jeśli token jest operatorem, zdejmij ze stosu 2 liczby i oblicz wynik dla nich wynik używając pobranego operatora
   3. Wynik umieść na stosie
3. Końcowy wynik to wartość ze szczytu stosu: wypisz go na ekranie

Nie sprawdzamy poprawności wprowadzonego wyrażenia, np. 4 + - będzie niepoprawny.

Wykorzystaj metodę Compute i funkcje operatorów z I zadania. Dla stosu wykorzystaj Stack<double>.

**Dodatkowo**

\*\*\*Zmodyfikować program \*\* tak żeby można było używać funkcji 1 argumentowych (SIN, COS). Dodatkowo zliczać sumę wszystkich liczb w ciągu (wykorzystaj operator +=) i na końcu podać ich średnią, odchylenie standardowe oraz medianę (posortować liczby i znaleźć element środkowy).

## **Zadanie domowe**

**1. Kalkulator dla stringów**

Napisać kalkulator z \*\* dla ciągów znaków. Operatory będą działały wg. następującej reguły (tylko te operatory definiujemy):

1. + - konkatenacja napisów, np. Ala Ma + da w wyniku MaAla (2 arg).
2. - - odejmij jeden znak z napisu np. AlaMaKota - da w wyniku AlaMaKot (1 arg.)
3. \* - dodaj tekst do siebie, np. AlaMaKota \* da w wyniku AlaMaKotaAlaMaKota (1 arg.)
4. / - zwraca połowę tekstu np. AlaMaKota / da w wyniku AlaM (1 arg.), wykorzystaj konstrukcję napis.Substring(0, napis.Length/2)

Input

|  |
| --- |
| a Ala Ma Koty + + AlaMaKoty + / - + |

Output

|  |
| --- |
| AlaMaKota |

Ćwiczenie: Przeanalizować przykład bez pisania kodu. Następnie napisać program z możliwością śledzenia kolejnych zawartości stosu: Console.WriteLine(string.Join(" ", stack.ToArray())). Jakie wnioski?

**2. Monte Carlo**

Napisać program obliczający pole koła o promieniu R metodą Monte Carlo.

Zobacz: <https://pl.wikipedia.org/wiki/Metoda_Monte_Carlo#Przyk.C5.82ad_ca.C5.82kowania_metod.C4.85_Monte_Carlo>

Przyjąć, że program będzie losował ilość prób określoną w parametrze wywołania (zobacz: <https://msdn.microsoft.com/pl-pl/library/cb20e19t.aspx>)

Następnie zmodyfikować program tak, żeby pozwalał obliczać:

* Pole kwadratu o boku R
* Objętość sfery o promieniu R

# Dodatek (program z lekcji)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lekcja\_06032017

{

class Program

{

private static Dictionary<string, Func<double, double, double>> \_operators = new Dictionary

<string, Func<double, double, double>>

{

{"+", Add},

{"-", Substract}

};

static double Add(double x, double y)

{

return x + y;

}

static double Substract(double x, double y)

{

return x - y;

}

static void Main(string[] args)

{

CreateOperators();

string expression = Console.ReadLine();

var tokens = expression.Split();

Stack<int> stack = new Stack<int>();

foreach (var token in tokens)

{

int value = 1;

if (int.TryParse(token, out value))

{

stack.Push(value);

}

else

{

Console.WriteLine(value);

int a= stack.Pop();

int b = stack.Pop();

stack.Push((int)\_operators[token](a, b));

}

}

Console.WriteLine(stack.Peek());

}

private static void Kalk1()

{

double x = double.Parse(Console.ReadLine());

double y = double.Parse(Console.ReadLine());

string @operator = Console.ReadLine();

Console.WriteLine(\_operators[@operator](x, y));

}

private static void CreateOperators()

{

\_operators.Add("\*", (x, y) => x \* y);

\_operators.Add("/", (x, y) => x / y);

\_operators.Add("%", (x, y) => x % y);

\_operators.Add("&", (x, y) => (int) x & (int) y);

\_operators.Add("SIN", (x, y) => Math.Sin(x));

}

}

}