БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Лабораторная работа № 7

по дисциплине «Структуры и алгоритмы обработки данных»

Вариант № 4

Выполнил студент: Ефименко Павел Викторович,

группа 981063

Минск 2020

**Условие:**

Используя стек, реализовать алгоритм преобразования алгебраического выражения из инфиксной формы записи в постфиксную форму представления.

Используя стек, реализовать алгоритм преобразования алгебраического выражения из инфиксной формы записи в префиксную форму представления.

Для обоих алгоритмов предусмотреть вхождение операций с различными приоритетами, а также наличие скобок в инфиксных выражениях.

**Листинг:**

public class ReversePolishNotation

{

static public double Calculate(string input)

{

var output = GetExpression(input.Replace(".", ","));

double result = Counting(output);

return result;

}

static private double Counting(string input)

{

var result = 0d;

var temp = new Stack<double>();

for (int i = 0; i < input.Length; i++)

{

if (char.IsDigit(input[i]))

{

string a = string.Empty;

while (!IsDelimeter(input[i]) && !IsOperator(input[i]))

{

a += input[i];

i++;

if (i == input.Length) break;

}

temp.Push(Convert.ToDouble(a));

i--;

}

else if (IsOperator(input[i]))

{

double a = temp.Pop();

double b = temp.Pop();

switch (input[i])

{

case '+':

result = b + a;

break;

case '-':

result = b - a;

break;

case '\*':

result = b \* a;

break;

case '/':

result = b / a;

break;

case '^':

result = Math.Pow(b, a);

break;

}

temp.Push(result);

}

}

return temp.Peek();

}

static private string GetExpression(string input)

{

var output = string.Empty;

var operStack = new Stack<char>();

for (int i = 0; i < input.Length; i++)

{

if (IsDelimeter(input[i]))

{

continue;

}

if (char.IsDigit(input[i]))

{

while (!IsDelimeter(input[i]) && !IsOperator(input[i]))

{

output += input[i];

i++;

if (i == input.Length) break;

}

output += " ";

i--;

}

if (IsOperator(input[i]))

{

if (input[i] == '(')

{

operStack.Push(input[i]);

}

else if (input[i] == ')')

{

char s = operStack.Pop();

while (s != '(')

{

output += s.ToString() + ' ';

s = operStack.Pop();

}

}

else

{

if (operStack.Count > 0)

{

if (GetPriority(input[i]) <= GetPriority(operStack.Peek()))

{

output += operStack.Pop().ToString() + " ";

}

}

operStack.Push(Convert.ToChar(input[i]));

}

}

}

while (operStack.Count > 0)

{

output += operStack.Pop() + " ";

}

return output;

}

static private bool IsDelimeter(char c)

{

if ((" =".IndexOf(c) != -1))

{

return true;

}

return false;

}

static private bool IsOperator(char с)

{

if (("+-/\*^()".IndexOf(с) != -1))

{

return true;

}

return false;

}

static private byte GetPriority(char s)

{

switch (s)

{

case '(':

return 0;

case ')':

return 1;

case '+':

return 2;

case '-':

return 3;

case '\*':

return 4;

case '/':

return 4;

case '^':

return 5;

default:

return 6;

}

}

}

class Program

{

static void Main(string[] args)

{

var exp = CreateNormalExpression("a / (b - c) \* (d + e)", 8.6, 2.4, 5.1, 0.3, 7.9);

var result = ReversePolishNotation.Calculate(exp);

Console.WriteLine($"Результат:\t{result.ToString("N3")}");

Console.ReadKey();

}

public static IEnumerable<object[]> ArrayData()

{

yield return new object[] { "a / (b - c) \* (d + e)", 8.6, 2.4, 5.1, 0.3, 7.9, "-26.119" };

yield return new object[] { "a \* (b - c) / (d + e)", 0.5, 6.1, 8.9, 2.4, 7.3, "-0.144" };

}

public static string CreateNormalExpression(string expression, double a, double b, double c, double d, double e)

{

var result = string.Empty;

var chars = expression.ToCharArray().Select(v => Convert.ToString(v)).ToArray();

for (int i = 0; i < chars.Length; i++)

{

switch (chars[i])

{

case "a":

chars[i] = a.ToString();

continue;

case "b":

chars[i] = b.ToString();

continue;

case "c":

chars[i] = c.ToString();

continue;

case "d":

chars[i] = d.ToString();

continue;

case "e":

chars[i] = e.ToString();

continue;

default:

continue;

}

}

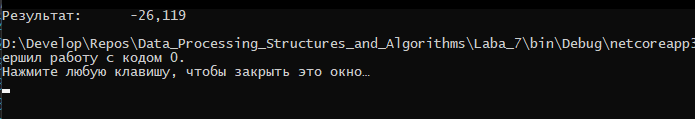
result = string.Join("", chars);

return result;

}

}

**Выполнение:**

****