|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 **Информатика и вычислительная техника**

**Отчет**

**по домашней работе № 1**

**Дисциплина:** Разработка приложений на языке C#

**Название работы:** Разработка веб-сервера на языке C#



Студент гр. ИУ6-72Б **\_\_**16.12.2022**\_\_\_\_\_\_\_** И.С.Марчук

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** А.М. Минитаева

(Подпись, дата) (И.О. Фамилия)

Москва, 2022

**Задание:**

Нужно реализовать консольный калькулятор, умеющий вычислять выражение, подаваемое на STDIN.

Требуется реализовать:

* сложение;
* вычитание;
* умножение;
* деление;
* поддержка скобок.

Нужно написать тесты, которые покрывают все операции.

Вариант: Нечетный по списку – Стековым методом

**Выполнение работы:**

Код программы:

static void Main(string[] args)

{

//Console.WriteLine("Input (valid) Task:");

//string input = Console.ReadLine();

//Console.WriteLine(new Calculator().Solve(input));

Calculator calc = new Calculator();

string[] inputs = { "1+2", "(234-11)\*34", "6\*6/6", "(9+1)/10", "(1+2+3+4+6)", "1+(100\*2,6)" };

string[] outputs = { "3", "7582", "6", "1", "16", "261" };

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

{

Console.WriteLine(inputs[i] + "=" + outputs[i]);

if (calc.Solve(inputs[i]) == outputs[i])

{

Console.WriteLine("OK");

}

else

{

Console.WriteLine("Failed");

}

}

}

}

class Calculator

{

private double makeOperation(double a, double b, string operation)

{

switch (operation)

{

case "+":

return a + b;

case "-":

return a - b;

case "\*":

return a \* b;

case "/":

return a / b;

default:

return 0;

}

}

public string Solve(string task)

{

Stack<string> parsedString = new Stack<string>();

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

char[] taskSymbols = task.ToCharArray();

string currentVariable = "";

foreach (char symbol in taskSymbols)

{

switch (symbol)

{

case '+':

case '-':

case '/':

case '\*':

case ')':

parsedString.Push(currentVariable + symbol.ToString());

currentVariable = "";

break;

case '(':

if (currentVariable.Length != 0)

parsedString.Push(currentVariable);

parsedString.Push(symbol.ToString());

currentVariable = "";

break;

default:

currentVariable += symbol;

break;

}

}

if (currentVariable.Length != 0)

parsedString.Push(currentVariable);

string[] variables = parsedString.ToArray();

Array.Reverse(variables);

int idx = 1;

stack.Push(variables[0]);

while (true)

{

if (variables[idx] == "+" || variables[idx] == "-" || variables[idx] == "\*" || variables[idx] == "/")

{

idx++;

continue;

}

if (stack.Count == 0 && idx == variables.Length - 1)

{

return variables[idx];

}

if (stack.Count == 0)

{

stack.Push(variables[idx]);

idx++;

continue;

}

currentVariable = stack.Pop();

if (currentVariable[currentVariable.Length - 1].ToString() == "(" && variables[idx][variables[idx].Length - 1].ToString() == ")")

{

variables[idx] = variables[idx].Remove(variables[idx].Length - 1);

if (idx != variables.Length - 1)

variables[idx] += variables[idx + 1];

continue;

}

if (variables[idx][variables[idx].Length - 1].ToString() == ")")

{

double first = double.Parse(currentVariable.Remove(currentVariable.Length - 1));

double second = double.Parse(variables[idx].Remove(variables[idx].Length - 1));

variables[idx] = this.makeOperation(first, second, currentVariable[currentVariable.Length - 1].ToString()).ToString() + ')';

continue;

}

//5+4+

string curOperations = currentVariable[currentVariable.Length - 1].ToString() + variables[idx][variables[idx].Length - 1].ToString();

string validOperations = "++|--|-+|+-|\*\*|//|/\*|\*/";

if (validOperations.Contains(curOperations))

{

double first = double.Parse(currentVariable.Remove(currentVariable.Length - 1));

double second = double.Parse(variables[idx].Remove(variables[idx].Length - 1));

variables[idx] = this.makeOperation(first, second, currentVariable[currentVariable.Length - 1].ToString()).ToString() + variables[idx][variables[idx].Length - 1].ToString();

continue;

}

//5+4

if (!"+-/\*(".Contains(variables[idx][variables[idx].Length - 1].ToString()))

{

double first = double.Parse(currentVariable.Remove(currentVariable.Length - 1));

double second = double.Parse(variables[idx]);

variables[idx] = this.makeOperation(first, second, currentVariable[currentVariable.Length - 1].ToString()).ToString();

continue;

}

stack.Push(currentVariable);

stack.Push(variables[idx]);

idx++;

}

return "Something Went Wrong :(";

}

}

Ниже на рисунке 1 показана работа программы:

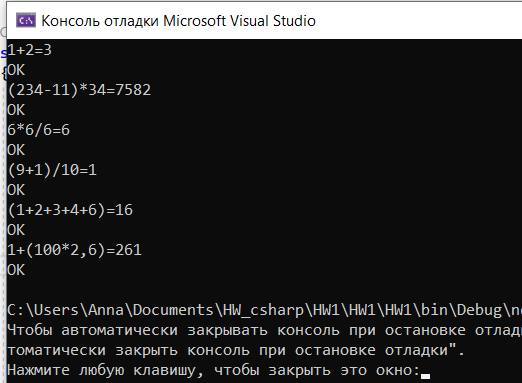


Рисунок 1 – Выполнение работы

**Вывод:** была написана программа, которая получив на вход стоку, вычисляет значения выражения с помощью распознавания строки стековым методом.