Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет

«Московский институт электронной техники»

институт системной и программной инженерии

и информационных технологий»

Курс «Теория алгоритмических языков и компиляторов»

Лабораторная работа № 1

Построение простейшего синтаксического анализатора выражений

Выполнил:

Студент группы ПИН-41Д

Джугели Дмитрий Александрович

Москва, 2025

Задание на лабораторную работу

Написать программу-аналог калькулятора (с некоторыми вариациями, в зависимости от желаемой оценки).

Входные данные: произвольная строка символов.

Выходные данные: заключение о корректности входной строки как арифметического выражения, вычисленное значение этого выражения, в случае некорректной входной строки информацию о типе ошибки (например, «Несоответствие количества (и)»). Выходные данные зависят от сложности задания.

Синтаксический анализатор должен распознавать следующие лексеммы:

1. Знаки арифметических операций: ‘+’, ‘-‘, ‘*’, ‘/’
2. Скобки ‘(‘ и ‘)’
3. Действительные числа (т.е. дробные). Например: 12, 66.6, .54, 221.
4. Имя встроенной функции (см. задание максимум)

Задание минимум (на 3)**:** считать строку символов, дать заключение о её соответствии арифметическому выражению, в случае несоответствия указать причину.

Задание максимум (на 5)**:** вычислить значение введённого выражения с учётом приоритета операций. При этом необходимо обрабатывать исключительные ситуации (например, деление на 0). Кроме того, нужно реализовать поддержку одной встроенной функции от двух переменных (например, log(a, b)). При этом аргументы функции сами являются выражениями (в том числе содержат эту функцию). Должна быть реализована возможность сравнительно простого изменения встроенной функции по требованию преподавателя (например, заменить log(a, b) на pow(a, b)). Встроенная функция имеет наивысший приоритет. Скобки служат для изменения порядка действий (обычная математика).

Код:

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

    class Program
    {
        //true, если оператор
        static private bool IsOperator(char c)
        {
            if ("+-/*()#".IndexOf(c) != -1)
                return true;
            return false;
        }

        //true, если пробел
        static private bool IsDelimeter(char c)
        {
            if (c == ' ' || c=='\t')
                return true;
            return false;
        }

        //priority
        static private byte GetPriority(char s)
        {
            switch (s)
            {
                case '(': return 1;
                case ')': return 1;
                case '+': return 2;
                case '-': return 2;
                case '*': return 3;
                case '/': return 3;
                default: return 4;
            }
        }

        static private string GetExpression(string input)
        {
            input = input.Replace(".", ",");
            while (input.IndexOf("pow") != -1)
                input = Regex.Replace(input, @"pow\((?<first>.*?),(?<second>.*?)\)",
"(({first})#({second}))");
            if (input.IndexOf(",,") != -1)
                return "Некоректный формат";

            string output = string.Empty;
            Stack<char> operStack = new Stack<char>();
            for (int i = 0; i < input.Length; i++)
            {
                if (IsDelimeter(input[i]))
                    continue;
                else if (char.IsDigit(input[i]) || input[i] == ',')
                {
                    string tempOut = string.Empty;

                    while ((!IsDelimeter(input[i])) && !IsOperator(input[i]))
                    {
                        if (!char.IsDigit(input[i]) && input[i] != ',')
                        return "Ошибка! " + i + 1 + "символ не является символом / цифрой";
                        tempOut += input[i];
                        i++;
```

```
                    if (i == input.Length) break;
                }
                output += tempOut + " ";
                i--;
            }


            else if (IsOperator(input[i]))
            {
                if (i + 1 < input.Length)
                    if (input[i] == '(' && input[i + 1] == '-')
                        output += "0 ";

                if (input[i] == '(')
                    operStack.Push(input[i]);

                else if (input[i] == ')')
                {
                    try
                    {
                        char s = operStack.Pop();
                        while (s != '(')
                        {
                            output += s.ToString() + ' ';
                            s = operStack.Pop();
                        }
                    }
                    catch (InvalidOperationException)
                    {
                        return "Неправильно расставлены скобки";
                    }
                }
                else
                {
                    if (operStack.Count > 0)
                        if (GetPriority(input[i]) <= GetPriority(operStack.Peek()))
                            output += operStack.Pop().ToString() + " ";

                    operStack.Push(char.Parse(input[i].ToString()));

                }
            }
            else return "Ошибка! " + (i + 1) + "символ не является символом / цифрой";
        }

        while (operStack.Count > 0)
            output += operStack.Pop() + " ";

        return output;
    }

    static private double Counting(string input)
    {
        double result = double.NaN;
        Stack<double> temp = new Stack<double>();

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

            if (char.IsDigit(input[i])||input[i]==',')
            {
                string a = string.Empty;

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

```
                        a += input[i];
                        i++;
                        if (i == input.Length) break;
                    }
                    a = a.Replace(",", "0,");
                    temp.Push(Convert.ToDouble(a));
                    i--;
                }
                else if (IsOperator(input[i]))
                {
                    try
                    {
                        double a = temp.Pop();
                        double b;
                        if (input[i] == '-' && temp.Count == 0)
                            b = 0;
                        else
                            b = temp.Pop();

                        switch (input[i])
                        {
                            case '+':
                                result = b + a; break;
                            case '-':
                                result = b - a; break;
                            case '*':
                                result = b * a; break;
                            case '#':
                                if (b<0)
                                {
                                    Console.WriteLine("Нельзя возводить отрицательное число в
степень");
                                    return double.NaN;
                                }
                                result = Math.Pow(b,a); break;
                            case '/':
                                if (a == 0)
                                {
                                    Console.WriteLine("Нельзя делить на 0");
                                    return double.NaN;
                                }
                                result = b / a; break;
                        }
                        temp.Push(result);
                    }
                    catch (InvalidOperationException)
                    {
                        Console.WriteLine("Неверное количество служебных символов");
                        return double.NaN;
                    }
                }
            }
            try
            {
                return temp.Peek();
            }
            catch (InvalidOperationException)
            {
                Console.WriteLine("Отсутствуют арифметичесие операции");
                return double.NaN;
            }
        }

        static public double Calculate(string input)
        {
```

```
            string output = GetExpression(input);
            Console.WriteLine("Преобразованное выражение: " + output.Replace("#", "pow"));
            double result = Counting(output);
            return result;
        }

        static void Main(string[] args)
        {
            while (true)
            {
                Console.Write("Введите выражение: ");
                string parsestr = Console.ReadLine();
                if (parsestr == "") break;

                Console.WriteLine("Результат: " + Calculate(parsestr));
            }
        }
    }
```

Тестовые данные: