

|  |  |
| --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  федеральное государственное бюджетное образовательное учреждение высшего образования  **«МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ ГОСУДАРСТВЕННЫЙ**  **ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (МАДИ)»** |  |

**Кафедра «Высшая математика»**

**Лабораторная работа № 2**

по дисциплине

«Структуры и алгоритмы обработки данных»

на тему:

**«**Постфикс/префикс**»**

**Выполнил:**

Учебная группа: 1бПМ   
ФИО: Кузнецов А.-С. О.

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Руководитель лабораторной работы:**

Должность: старший преподаватель

Звание: б/з

ФИО: Кутейников И. А.

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

Цель:

Решение программой примеров в префиксном/постфиксном виде, а также их преобразование из обычного вида.

Код:

import java.util.\*;  
import java.awt.Toolkit;  
import java.awt.datatransfer.StringSelection;  
public class Main2 {  
 public static Double Post(String[] expression) {  
 Stack<Double> stack = new Stack<>();  
 for (int i = 0; i < expression.length; i++) {  
 if (expression[i].matches("-?\\d+")) {  
 stack.push(Double.*parseDouble*(expression[i]));  
 } else {  
 double operand1 = 0;  
 double operand2 = 0;  
 if (!expression[i].equals("sin") && !expression[i].equals("cos")) {  
 operand2 = stack.pop();  
 operand1 = stack.pop();  
 } else {  
 stack.push(Double.*parseDouble*(expression[i + 1]));  
 operand2 = stack.pop();  
 }  
 switch (expression[i]) {  
 case "+":  
 stack.push(operand1 + operand2);  
 break;  
 case "-":  
 stack.push(operand1 - operand2);  
 break;  
 case "\*":  
 stack.push(operand1 \* operand2);  
 break;  
 case "/":  
 stack.push(operand1 / operand2);  
 break;  
 case "sin":  
 operand2 = Math.*sin*(operand2);  
 stack.push(operand2);  
 i = i + 1;  
 break;  
 case "cos":  
 operand2 = Math.*cos*(operand2);  
 stack.push(operand2);  
 i = i + 1;  
 break;  
 case "^":  
 double d = operand1;  
 for (double i1 = 0; i1 < operand2; i1++) {  
 d = d \* operand1;  
 }  
 stack.push(d);  
 break;  
 }  
 }  
 }  
 return stack.pop();  
 }  
  
 public static double Pref(String[] expression) {  
 Stack<Double> stack = new Stack<>();  
  
 for (int i = expression.length - 1; i >= 0; i--) {  
 String token = expression[i];  
  
 if (*isOperator*(token)) {  
 double operand1 = stack.pop();  
 double operand2 = stack.pop();  
 double result = *applyOperator*(token, operand1, operand2);  
 stack.push(result);  
 } else if (token.equals("sin") || token.equals("cos")) {  
 double operand = stack.pop();  
 double result = *applyTrigonometricFunction*(token, operand);  
 stack.push(result);  
 } else if (token.equals("^")) {  
 double base = stack.pop();  
 double exponent = stack.pop();  
 double result = Math.*pow*(base, exponent);  
 stack.push(result);  
 } else {  
 stack.push(Double.*parseDouble*(token));  
 }  
 }  
  
 return stack.pop();  
 }  
  
 private static boolean isOperator(String token) {  
 return token.equals("+") || token.equals("-") || token.equals("\*") || token.equals("/");  
 }  
  
 private static double applyOperator(String operator, double operand1, double operand2) {  
 switch (operator) {  
 case "+":  
 return operand1 + operand2;  
 case "-":  
 return operand1 - operand2;  
 case "\*":  
 return operand1 \* operand2;  
 case "/":  
 return operand1 / operand2;  
 default:  
 throw new IllegalArgumentException("Invalid operator: " + operator);  
 }  
 }  
  
 private static double applyTrigonometricFunction(String function, double operand) {  
 switch (function) {  
 case "sin":  
 return Math.*sin*(Math.*toRadians*(operand));  
 case "cos":  
 return Math.*cos*(Math.*toRadians*(operand));  
 default:  
 throw new IllegalArgumentException("Invalid function: " + function);  
 }  
 }  
  
 public static void Perest(String[] exp) {  
 Stack<String> znaki = new Stack<>();  
 ArrayList<String> ar = new ArrayList<>();  
 String str = "";  
 for (int i = 0; i < exp.length; i++) {  
 if (exp[i].equals("(")) {  
 i++;  
 do {  
 if (exp[i].matches("-?\\d+") || exp[i].equals("sin") || exp[i].equals("cos")) {  
 ar.add(exp[i]);  
 if (i > 0 && exp[i - 1].equals("\*") || exp[i - 1].equals("/")) {  
 ar.add(znaki.pop());  
 }  
 } else {  
 znaki.push(exp[i]);  
 }  
 if (exp[i + 1].equals(")")) {  
 for (int i1 = 0; i1 < znaki.size(); i1++) {  
 ar.add(znaki.pop());  
 }  
 }  
 if ((exp[i + 1].equals("+") || exp[i + 1].equals("-")) && !znaki.isEmpty() && (znaki.peek().equals("+") || znaki.peek().equals("-"))) {  
 ar.add(znaki.pop());  
 znaki.push(exp[i + 1]);  
 i++;  
 }  
 i++;  
 } while (!exp[i].equals(")"));  
 }  
 if (i != 0) {  
 i++;  
 }  
 do {  
 if (exp[i].matches("-?\\d+") || exp[i].equals("sin") || exp[i].equals("cos")) {  
 ar.add(exp[i]);  
 if (i > 0) {  
 if (exp[i - 1].equals("\*") || exp[i - 1].equals("/")) {  
 ar.add(znaki.pop());  
 }  
 }  
 } else {  
 znaki.push(exp[i]);  
 }  
 if (i > 0) {  
 if (i == exp.length - 1) {  
 for (int i1 = 0; i1 < znaki.size(); i1++) {  
 ar.add(znaki.pop());  
 }  
 }  
 }  
 if (i + 1 < exp.length) {  
 if ((exp[i + 1].equals("+") || exp[i + 1].equals("-")) && !znaki.isEmpty() && (znaki.peek().equals("+") || znaki.peek().equals("-"))) {  
 ar.add(znaki.pop());  
 znaki.push(exp[i + 1]);  
 i++;  
 }  
 }  
 i++;  
 } while (i <= exp.length - 1);  
 break;  
 }  
 for (int i=0; i < ar.size(); i++){  
 if(ar.get(i).equals("(")||ar.get(i).equals(")")){  
 ar.remove(ar.get(i));  
 }  
 }  
 for (String i1 : ar) {  
 System.*out*.print(i1 + " ");  
 str = str + i1 + ' ';  
 }  
 System.*out*.println();  
 StringSelection selection = new StringSelection(str);  
 Toolkit.*getDefaultToolkit*().getSystemClipboard().setContents(selection, null);  
 }  
 public static void applyOperation(Stack<Double> numbers, Stack<String> operations) {  
 double num1 =0;  
 double num2=0;  
 String operation = operations.pop();  
 if(!operation.equals("sin")&&!operation.equals("cos")) {  
 num2 = numbers.pop();  
 num1 = numbers.pop();  
 }  
 else{  
 num1 = numbers.pop();  
 }  
 double result = 0;  
 switch(operation) {  
 case "+":  
 result = num1 + num2;  
 break;  
 case "-":  
 result = num1 - num2;  
 break;  
 case "\*":  
 result = num1 \* num2;  
 break;  
 case "/":  
 result = num1 / num2;  
 break;  
 case "sin":  
 result = Math.*sin*(num1);  
 break;  
 case "cos":  
 result = Math.*cos*(num1);  
 break;  
 }  
  
 numbers.push(result);  
 }  
  
 public static int precedence(String op) {  
 if (op.equals("+") || op.equals("-")) {  
 return 1;  
 } else if (op.equals("\*") || op.equals("/")) {  
 return 2;  
 } else {  
 return 0;  
 }  
 }  
  
 public static void main(String[] args) {  
 int ans = 0;  
 String expression;  
 do {  
 System.*out*.println("1.Префикс");  
 System.*out*.println("2.Постфикс");  
 System.*out*.println("3.Из инфикса в префикс");  
 System.*out*.println("4.Вычисление обычного выражения");  
 System.*out*.print("Выберете пункт меню: ");  
 Scanner in = new Scanner(System.*in*);  
 ans = in.nextInt();  
 switch (ans) {  
 case 1:  
 System.*out*.println("Введите выражение:");  
 in.nextLine();  
 expression = in.nextLine();  
 String[] tokens = expression.split(" ");  
 double result1 = *Post*(tokens);  
 System.*out*.println(result1);  
 break;  
 case 2:  
 System.*out*.println("Введите выражение:");  
 in.nextLine();  
 expression = in.nextLine();  
 String[] tokens1 = expression.split(" ");  
 double result2 = *Pref*(tokens1);  
 System.*out*.println(result2);  
 break;  
 case 3:  
 System.*out*.println("Введите выражение:");  
 in.nextLine();  
 expression = in.nextLine();  
 String[] tokens2 = expression.split(" ");  
 *Perest*(tokens2);  
 break;  
 case 4:  
 System.*out*.println("Введите математическое выражение через пробелы:");  
 in.nextLine();  
 String input = in.nextLine();  
 String[] elements = input.split(" ");  
 Stack<Double> numbers = new Stack<>();  
 Stack<String> operations = new Stack<>();  
 for (String element : elements) {  
 if (element.equals("(")) {  
 operations.push(element); // Обработка открывающей скобки  
 } else if (element.equals(")")) {  
 while (!operations.peek().equals("(")) {  
 *applyOperation*(numbers, operations);  
 }  
 operations.pop(); // Удаляем открывающую скобку из стека  
 } else if (element.equals("cos") || element.equals("sin")) {  
 operations.push(element);  
 } else if (element.equals("+") || element.equals("-") || element.equals("\*") || element.equals("/")) {  
 while (!operations.isEmpty() && *precedence*(operations.peek()) >= *precedence*(element)) {  
 *applyOperation*(numbers, operations);  
 }  
 operations.push(element);  
 } else {  
 numbers.push(Double.*parseDouble*(element));  
 }  
 }  
 while (!operations.isEmpty()) {  
 *applyOperation*(numbers, operations);  
 }  
 System.*out*.println("Результат: " + numbers.pop());  
 break;  
 case 5:  
 break;  
 }  
 } while (ans != 5);  
 }  
}

Результат:

1.Префикс

2.Постфикс

3.Из инфикса в префикс

4.Вычисление обычного выражения

Выберете пункт меню: 3

Введите выражение:

( 9 + 8 - 5 ) \* 8 - 12 + 3

Результат:

9 8 + 5 - 8 \* 12 - 3 +

Выберете пункт меню: 1

Введите выражение:

9 8 + 5 - 8 \* 12 - 3 +

Результат:

87.0

Вывод:

В ходе выполнения данной лабораторной работы были изучены и реализованы методы для работы с выражениями в префиксной и постфиксной формах. Программа успешно демонстрирует возможности преобразования стандартных инфиксных выражений в префиксные и постфиксные формы, что является важным навыком в области компьютерных наук.