МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Белгородский государственный технологический университет им. В.Г. Шухова»

**ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ Кафедра программного обеспечения вычислительной техники и автоматизированных систем**

Лабораторная работа №1

по дисциплине: Математическая логика и теория алгоритмов:

Логика высказываний

Работу выполнил:

Студент группы ПВ-21:

Донцов Александр Алексеевич

Проверил:

Бондаренко Татьяна Владимировна

Белгород – 2018

Модуль main

import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
  
class Main {  
 public static void main(String[] args) throws IOException {  
 String input;  
 int output;  
 while(true) {  
 System.out.print("Enter infix: ");  
 System.out.flush();  
 input = getString(); // Ввод строки с клавиатуры  
 if( input.equals("") ) // Завершение, если нажата клавиша  
 break; // [Enter]  
 // Создание объекта-преобразователя  
 InToPost theTrans = new InToPost(input);  
 input = theTrans.doTrans(); // Преобразование  
 System.out.println("Postfix: " + input);  
 GenerateBinary end = new GenerateBinary(input);  
 end.rec(0);//генерация двоичных векторов  
 }  
 }  
  
 public static String getString() throws IOException {//ввод строки  
 BufferedReader br = new BufferedReader(new InputStreamReader(System.in));  
 String s = br.readLine();  
 return s;  
 }  
}

Модуль InToPost

// Преобразование инфиксных арифметических выражений в постфиксные  
import java.io.\*; // Для ввода/вывода  
  
class InToPost { // Преобразование инфиксной записи в постфиксную  
 private StackChar theStack;  
 private String input;  
 private String output = "";  
  
 public InToPost(String in) { // Конструктор  
 input = in;  
 int stackSize = input.length();  
 theStack = new StackChar(stackSize);  
 }  
  
 public String doTrans() { // Преобразование в постфиксную форму  
 for (int j = 0; j < input.length(); j++) {  
 char ch = input.charAt(j);  
 //theStack.displayStack("For " + ch + " "); // \*диагностика\*  
 switch (ch) {  
 case '+': // дизюнкция  
 gotOper(ch, 1); // Извлечение операторов  
 break; // (приоритет 1)  
 case '\*': // конъюнкция  
 gotOper(ch, 2); // Извлечение операторов  
 break; // (приоритет 2)  
 case '!':  
 gotOper(ch, 3);  
 break;//(приоритет 3)  
 case '(': // Открывающая круглая скобка  
 theStack.push(ch); // Занести в стек  
 break;  
 case ')': // Закрывающая круглая скобка  
 gotParen(ch); // Извлечение операторов  
 break;  
 default: // Остается операнд  
 output = output + ch; // Записать в выходную строку  
 break;  
 }  
 }  
 while (!theStack.isEmpty()) { // Извлечение оставшихся операторов  
 //theStack.displayStack("While "); // \*диагностика\*  
 output = output + theStack.pop(); // write to output  
 }  
 //theStack.displayStack("End "); // \*диагностика\*  
 return output; // Возвращение постфиксного выражения  
 }  
  
 public void gotOper(char opThis, int prec1) { // Чтение оператора из входной строки  
 while (!theStack.isEmpty()) {  
 char opTop = theStack.pop();  
 //System.out.print((char) opTop);  
 if (opTop == '(') { // Если это '('  
 theStack.push(opTop); // Вернуть '('  
 break;  
 } else { // Оператор  
 int prec2; // Приоритет нового оператора  
 if (opTop == '+') // Определение приоритета  
 prec2 = 1;  
 else if(opTop == '\*')  
 prec2 = 2;  
 else  
 prec2 = 3;  
 if (prec2 < prec1) { //Если приоритет нового оператора меньше приоритета старого  
 theStack.push(opTop); // Сохранить новый оператор  
 break;  
 } else // Приоритет нового оператора  
 output = output + opTop; // не меньше приоритета старого  
 }  
 }  
 theStack.push(opThis); // Занесение в стек нового оператора  
 }  
  
 public void gotParen(char ch) { // Прочитана закрывающая скобка  
 while (!theStack.isEmpty()) {  
 char chx = theStack.pop();  
 if (chx == '(') // Если извлечен элемент ‘(‘  
 break; // Прервать выполнение  
 else // Если извлечен оператор  
 output = output + chx; // Вывести в постфиксную строку  
 }  
 }  
}

Модуль ParsePost

class ParsePost {//вычисление постфиксного выражения  
 private StackInt theStack;  
 private String input;  
  
 public ParsePost(String s) {  
 input = s;  
 }  
  
 public int doParse() {  
 theStack = new StackInt(20); // Создание объекта стека  
 char ch;  
 int j;  
 int num1 = 0, num2, interAns;  
 for(j = 0; j < input.length(); j++) { // Для каждого символа  
 ch = input.charAt(j); // Чтение символа  
 //theStack.displayStack(ch + " "); // \*диагностика\*  
 if(ch >= '0' && ch <= '9') // Если это цифра  
 theStack.push( (ch-'0') ); // Занести в стек  
 else { // Если это оператор  
 num2 = theStack.pop(); // Извлечение операндов  
 if(!theStack.isEmpty())  
 num1 = theStack.pop();  
 switch(ch) // Выполнение арифметической  
 { // операции  
 case '+':  
 if (num1 + num2 > 0)  
 interAns = 1;  
 else  
 interAns = 0;  
 break;  
 case '!':  
 if (num2 == 0)  
 interAns = 1;  
 else  
 interAns = 0;  
 break;  
 case '\*':  
 if (num1 \* num2 != 0)  
 interAns = 1;  
 else  
 interAns = 0;  
 break;  
 default:  
 interAns = 0;  
 }  
 // Занесение промежуточного  
 // результата в стек  
 theStack.push(interAns);  
 }  
 }  
 interAns = theStack.pop(); // Получение результата  
 return interAns;  
 }  
}

Модуль GenerateBinary

import java.io.IOException;  
import java.util.Scanner;  
  
class GenerateBinary {  
 private static int[] *arr*;  
 private static String *str*;  
 private static int *n*;  
  
 public GenerateBinary(String in) throws IOException {  
 *str* = in;  
 System.*out*.print("Input n = ");  
 *n* = *getInt*();  
 *arr* = new int[*n*];  
 }  
  
 public void rec(int i){  
 for (int x = 0; x <= 1; x++) {  
  
  
 if (i == (*n*)) {  
 *test*(*arr*);  
 break;  
 } else  
 {  
 *arr*[i] = x;  
 rec(i + 1);  
 }  
 }  
 }  
  
 public static void test(int[] arr){  
 /\*Заменяем буквы цифрами\*/  
 String out = new String();  
 out = *str*;  
 int k = 0;  
 char c = 0, ch = 0;  
 for (int i = 0; i < out.length(); i++) {  
 String path = new String();  
 c = out.charAt(i);  
 if ( c >= 'A' && c <= 'Z') {  
 String a = Integer.*toString*(arr[k++]);  
 for (int j = 0; j < out.length(); j++) {  
 ch = out.charAt(j);  
 if ( c == ch)  
 path += a;  
 else  
 path += ch;  
  
 }  
 out = path;  
 }  
 }  
  
 /\*Решаем уравнение\*/  
 ParsePost aParser = new ParsePost(out);  
 int output = aParser.doParse();  
 for (int i = 0; i < *n*; i++) {  
 System.*out*.print(' ');  
 System.*out*.print(arr[i]);  
 }  
 System.*out*.println("= " + output);  
 }  
  
 public static int getInt() throws IOException {  
 Scanner in = new Scanner(System.*in*);  
 int a = in.nextInt();  
 return a;  
 }  
}

Модуль StackChar

class StackChar {//реализация стека с символами  
 private int maxSize;  
 private char[] stackArray;  
 private int top;  
 //--------------------------------------------------------------  
 public StackChar(int s) // Конструктор  
 {  
 maxSize = s;  
 stackArray = new char[maxSize];  
 top = -1;  
 }  
 //--------------------------------------------------------------  
 public void push(char j) { // Размещение элемента на вершине стека  
 stackArray[++top] = j;  
 }  
 //--------------------------------------------------------------  
 public char pop() {// Извлечение элемента с вершины стека  
 return stackArray[top--];  
 }  
 //--------------------------------------------------------------  
 public char peek() {// Чтение элемента с вершины стека  
 return stackArray[top];  
 }  
 //--------------------------------------------------------------  
 public boolean isEmpty() {// true, если стек пуст  
 return (top == -1);  
 }  
 //-------------------------------------------------------------  
 public int size() {// Текущий размер стека  
 return top+1;  
 }  
 //--------------------------------------------------------------  
 public char peekN(int n) {  
 return stackArray[n];  
 }  
 //--------------------------------------------------------------  
 public void displayStack(String s)  
 {  
 System.*out*.print(s);  
 System.*out*.print("Stack (bottom-->top): ");  
 for(int j=0; j<size(); j++)  
 {  
 System.*out*.print( peekN(j) );  
 System.*out*.print(' ');  
 }  
 System.*out*.println("");  
 }  
//--------------------------------------------------------------  
}

Модуль StackInt

class StackInt//реализация стека с целыми числами  
{  
 private int maxSize;  
 private int[] stackArray;  
 private int top;  
  
 public StackInt(int size) {// Конструктор  
 maxSize = size;  
 stackArray = new int[maxSize];  
 top = -1;  
 }  
  
 public void push(int j) {// Размещение элемента на вершине стека  
 stackArray[++top] = j;  
 }  
  
 public int pop() {// Извлечение элемента с вершины стека  
 return stackArray[top--];  
 }  
  
 public int peek() {// Чтение элемента на вершине стека  
 return stackArray[top];  
 }  
 public boolean isEmpty() {// true, если стек пуст  
 return (top == -1);  
 }  
 public boolean isFull() {// true, если стек заполнен  
 return (top == maxSize - 1);  
 }  
  
 public int size() {// Текущий размер стека  
 return top + 1;  
 }  
  
 public int peekN(int n) {// Чтение элемента с индексом n  
 return stackArray[n];  
 }  
  
 public void displayStack(String s) {  
 //System.out.print(s);  
 //System.out.print("Stack (bottom-->top): ");  
 for (int j = 0; j < size(); j++) {  
 //System.out.print(peekN(j));  
 //System.out.print(' ');  
 }  
 //System.out.println("");  
 }  
}

Пример работы:

