Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Ордена Трудового Красного Знамени

федеральное государственное бюджетное образовательное учреждение высшего образования

МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ

Кафедра «Математической кибернетики и информационных технологий»

Лабораторная работа №4 по дисциплине

«Структуры и алгоритмы обработки данных»

Выполнил:

студент группы БВТ1902

Клюев А.П.

Цель данной лабораторной – получить знания и навыки реализовав

следующие структуры данных:

1. Стек (stack): операции для стека: инициализация, проверка на

пустоту, добавление нового элемента в начало, извлечение

элемента из начала;

2. Дек (двусторонняя очередь, deque): операции для дека: инициализация, проверка на пустоту, добавление нового элемента в начало, добавление нового элемента в конец, извлечение элемента из начала, извлечение элемента из конца. Разработать программу обработки данных, содержащихся в заранее подготовленном txt-файле, в соответствии с заданиями, применив указанную в задании структуру данных.

Задания:

1. Отсортировать строки файла, содержащие названия книг, в

алфавитном порядке с использованием двух деков.

2. Дек содержит последовательность символов для шифровки

сообщений. Дан текстовый файл, содержащий зашифрованное сообщение.

Пользуясь деком, расшифровать текст. Известно, что при шифровке каждый

символ сообщения заменялся следующим за ним в деке по часовой стрелке

через один.

3. Даны три стержня и n дисков различного размера. Диски можно

надевать на стержни, образуя из них башни. Перенести n дисков со стержня А

на стержень С, сохранив их первоначальный порядок. При переносе дисков

необходимо соблюдать следующие правила:

- на каждом шаге со стержня на стержень переносить только один диск;

- диск нельзя помещать на диск меньшего размера;

- для промежуточного хранения можно использовать стержень В.

Реализовать алгоритм, используя три стека вместо стержней А, В, С.

Информация о дисках хранится в исходном файле.

4. Дан текстовый файл с программой на алгоритмическом языке. За один просмотр файла проверить баланс круглых скобок в тексте, используя стек.

5. Дан текстовый файл с программой на алгоритмическом языке. За один

просмотр файла проверить баланс квадратных скобок в тексте, используя дек.

6. Дан файл из символов. Используя стек, за один просмотр файла напечатать сначала все цифры, затем все буквы, и, наконец, все остальные символы, сохраняя исходный порядок в каждой группе символов.

7. Дан файл из целых чисел. Используя дек, за один просмотр файла напечатать сначала все отрицательные числа, затем все положительные числа, сохраняя исходный порядок в каждой группе.

8. Дан текстовый файл. Используя стек, сформировать новый текстовый файл, содержащий строки исходного файла, записанные в обратном порядке: первая строка становится последней, вторая – предпоследней и т.д.

9. Дан текстовый файл. Используя стек, вычислить значение логического выражения, записанного в текстовом файле в следующей форме:

< ЛВ > ::= T | F | (N<ЛВ>) | (<ЛВ>A<ЛВ>) | (<ЛВ>X<ЛВ>) |

(<ЛВ>O<ЛВ>),

где буквами обозначены логические константы и операции:

T – True, F – False, N – Not, A – And, X – Xor, O – Or.

10. Дан текстовый файл. В текстовом файле записана формула

следующего вида:

<Формула> ::= <Цифра> | M(<Формула>,<Формула>) |

N(Формула>,<Формула>)

< Цифра > ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

где буквами обозначены функции:

M – определение максимума, N – определение минимума.

Используя стек, вычислить значение заданного выражения.

11. Дан текстовый файл. Используя стек, проверить, является ли

содержимое текстового файла правильной записью формулы вида:

< Формула > ::= < Терм > | < Терм > + < Формула > | < Терм > - < Формула >

< Терм > ::= < Имя > | (< Формула >)

< Имя > ::= x | y | z

Листинг программы

Класс Stack

import java.util.ArrayList;  
import java.util.Arrays;  
  
public class Stack<E>  
{  
  
 private ArrayList<E> stack;  
 private int size;  
  
 public Stack()  
 {  
 stack = new ArrayList<>();  
 size = 0;  
 }  
  
 public boolean isEmpty()  
 {  
 return this.size == 0;  
 }  
  
 public void push(E element)  
 {  
 if(element != null)  
 {  
 this.stack.add(element);  
 this.size++;  
 }  
 }  
  
 public E peek()  
 {  
 if(size != 0)  
 {  
 return this.stack.get(this.size-1);  
 }  
 System.*out*.println("Stack is empty");  
 return null;  
 }  
  
 public E pop()  
 {  
 if(size != 0)  
 {  
 E element = this.stack.get(this.size-1);  
 this.stack.remove(this.size-1);  
 this.size--;  
 return element;  
 }  
 System.*out*.println("Stack is empty");  
 return null;  
 }  
  
 public String toString()  
 {  
 return Arrays.*toString*(this.stack.toArray());  
 }  
  
 public int getSize()  
 {  
 return this.size;  
 }  
  
}

Класс Deque

import java.util.ArrayList;  
import java.util.Arrays;  
  
public class Deque<E>  
{  
 private ArrayList<E> deque;  
 private int size;  
  
 public Deque()  
 {  
 this.deque = new ArrayList<>();  
 this.size = 0;  
 }  
  
 public boolean isEmpty()  
 {  
 return this.size == 0;  
 }  
  
 public void pushLast(E element)  
 {  
 this.deque.add(element);  
 this.size++;  
 }  
  
 public void pushFirst(E element)  
 {  
 this.deque.add(0, element);  
 this.size++;  
 }  
  
 public E peekLast()  
 {  
 if(size != 0)  
 {  
 return this.deque.get(this.size-1);  
 }  
 System.*out*.println("Deque is empty");  
 return null;  
 }  
  
 public E peekFirst()  
 {  
 if(size != 0)  
 {  
 return this.deque.get(0);  
 }  
 System.*out*.println("Deque is empty");  
 return null;  
 }  
  
 public E popLast()  
 {  
 if(size != 0)  
 {  
 E element = this.deque.get(this.size-1);  
 this.deque.remove(this.size-1);  
 this.size--;  
 return element;  
 }  
 System.*out*.println("Deque is empty");  
 return null;  
 }  
  
 public E popFirst()  
 {  
 if(size != 0)  
 {  
 E element = this.deque.get(0);  
 this.deque.remove(0);  
 this.size--;  
 return element;  
 }  
 System.*out*.println("Deque is empty");  
 return null;  
 }  
  
 public String toString()  
 {  
 return Arrays.*toString*(this.deque.toArray());  
 }  
  
 public int getSize()  
 {  
 return this.size;  
 }  
}

Класс tasks

import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.Scanner;  
  
public class tasks {  
 public static void main(String[] args) {  
 *task1*();  
 *task2*();  
 *task3*();  
 *task4*();  
 *task5*();  
 *task6*();  
 *task7*();  
 *task8*();  
 *task9*();  
 *task10*();  
 *task11*();  
  
 }  
  
 public static void task1() {  
 Scanner input = new Scanner(System.*in*);  
 BufferedReader reader;  
 Deque<String> d1 = new Deque<>();  
 while(true) {  
 try {  
 System.*out*.print("Input path to task1 file : ");  
 String path = input.nextLine();  
 input.close();  
 reader = new BufferedReader(new FileReader(path));  
 break;  
 }  
 catch(IOException ioExc)  
 {  
 System.*out*.println("Wrong path to file");  
 }  
 }  
 try {  
 String line = reader.readLine();  
 while(line != null)  
 {  
 d1.pushLast(line);  
 line = reader.readLine();  
 }  
 }  
 catch(IOException ioExc)  
 {  
 ioExc.printStackTrace();  
 }  
  
 Deque<String> d2 = new Deque<>();  
 d2.pushFirst(d1.popFirst());  
 while(!d1.isEmpty()) {  
 String first = d1.peekFirst().toLowerCase();  
 String second = d2.peekFirst().toLowerCase();  
 boolean compareWithLast = false;  
  
 if(d1.peekFirst().length() >= d2.peekFirst().length()) {  
 for(int i = 0; i < second.length(); i++)  
 {  
 if(first.charAt(i) < second.charAt(i))  
 {  
 d2.pushFirst(d1.peekFirst());  
 d1.popFirst();  
 break;  
 }  
 if(first.charAt(i) > second.charAt(i))  
 {  
 compareWithLast = true;  
 break;  
 }  
 }  
 }  
 else  
 {  
 for(int i = 0; i < first.length(); i++)  
 {  
 if(first.charAt(i) < second.charAt(i))  
 {  
 d2.pushFirst(d1.popFirst());  
 break;  
 }  
 if(first.charAt(i) > second.charAt(i))  
 {  
 compareWithLast = true;  
 break;  
 }  
 }  
 }  
 if(d1.isEmpty()) {  
 break;  
 }  
 if(compareWithLast)  
 {  
 second = d2.peekLast().toLowerCase();  
 }  
 if(d1.peekFirst().length() >= d2.peekFirst().length() && compareWithLast)  
 {  
 for(int i = 0; i < second.length(); i++)  
 {  
 if(first.charAt(i) > second.charAt(i))  
 {  
 d2.pushLast(d1.peekFirst());  
 d1.popFirst();  
 break;  
 }  
 if(first.charAt(i) < second.charAt(i))  
 {  
 d1.pushLast(d2.popLast());  
 break;  
 }  
 }  
 }  
 else if(compareWithLast)  
 {  
 for(int i = 0; i < first.length(); i++)  
 {  
 if(first.charAt(i) > second.charAt(i))  
 {  
 d2.pushLast(d1.peekFirst());  
 d1.popFirst();  
 break;  
 }  
 if(first.charAt(i) < second.charAt(i))  
 {  
 d1.pushLast(d2.popLast());  
 break;  
 }  
 }  
 }  
 }  
 System.*out*.println(d2.toString());  
 }  
  
 public static void task2() {  
 Scanner input = new Scanner(System.*in*);  
 BufferedReader reader;  
 Deque<Character> deq = new Deque<>();  
 System.*out*.print("Input decoder string: ");  
 String decoder = input.nextLine().toLowerCase();  
 for(int i = 0; i < decoder.length(); i++)  
 {  
 deq.pushLast(decoder.charAt(i));  
 }  
 while(true)  
 {  
 try  
 {  
 System.*out*.print("Input path to task2 file: ");  
 String path = input.nextLine();  
 input.close();  
 reader = new BufferedReader(new FileReader(path));  
 break;  
 }  
 catch(IOException ioExc)  
 {  
 System.*out*.println("Wrong path to file");  
 }  
 }  
 String line = "";  
 try  
 {  
 String newLine = reader.readLine();  
 while(newLine != null)  
 {  
 line += newLine + " ";  
 newLine = reader.readLine();  
 }  
 }  
 catch(IOException ioExc)  
 {  
 ioExc.printStackTrace();  
 }  
 System.*out*.println("Encoded message is:");  
 System.*out*.println(line);  
  
 String decodedMessage = "";  
 line = line.toLowerCase();  
 line = line.trim();  
 boolean canDecode = true;  
 int index = 0;  
 while(decodedMessage.length() < line.length() && canDecode)  
 {  
 canDecode = false;  
 if(line.charAt(index) == ' ')  
 {  
 index++;  
 decodedMessage += " ";  
 }  
 for(int i = 0; i < deq.getSize(); i++)  
 {  
 if(deq.peekFirst() == line.charAt(index))  
 {  
 canDecode = true;  
 break;  
 }  
 deq.pushLast(deq.popFirst());  
 }  
 if(!canDecode)  
 {  
 System.*out*.println("Can't decode input message because of lack of symbols in decoder");  
 break;  
 }  
 deq.pushLast(deq.popFirst());  
 deq.pushLast(deq.popFirst());  
 decodedMessage += deq.peekFirst();  
 index++;  
 }  
 if(canDecode)  
 {  
 System.*out*.println("Decoded message is:");  
 System.*out*.println(decodedMessage);  
 }  
 }  
  
 public static void task3() {  
 Scanner input = new Scanner(System.*in*);  
 BufferedReader reader;  
 Stack<Integer> s1;  
 Stack<Integer> s2;  
 Stack<Integer> s3;  
  
 while(true)  
 {  
 try  
 {  
 System.*out*.print("Input path task3 to file: ");  
 String path = input.nextLine();  
 input.close();  
 reader = new BufferedReader(new FileReader(path));  
 break;  
 }  
 catch(IOException ioExc)  
 {  
 System.*out*.println("Wrong path to file");  
 }  
 }  
 try  
 {  
 String line = reader.readLine();  
 int count = Integer.*parseInt*(line);  
 System.*out*.println("Количество дисков: " + count);  
 s1 = new Stack<>();  
 s2 = new Stack<>();  
 s3 = new Stack<>();  
 for(int i = count; i > 0; i--){  
 s1.push(i);  
 }  
 *HanoiSolver*(count, s1, s2, s3);  
 System.*out*.println(s1.toString());  
 System.*out*.println(s2.toString());  
 System.*out*.println(s3.toString());  
 }  
 catch(IOException ioExc)  
 {  
 ioExc.printStackTrace();  
 }  
 catch(NumberFormatException numExc)  
 {  
 System.*out*.println("Incorrect format of numbers in file");  
 }  
  
 }  
  
 public static void HanoiSolver(int q, Stack<Integer> from, Stack<Integer> buf, Stack<Integer> to){  
 if(q == 1){  
 return;  
 }  
 *HanoiSolver*(q - 1, from, to, buf);  
 to.push(from.pop());  
 *HanoiSolver*(q - 1, buf, from, to);  
 }  
  
 public static boolean task4() {  
 Scanner input = new Scanner(System.*in*);  
 BufferedReader reader;  
 while(true)  
 {  
 try  
 {  
 System.*out*.print("Input path to task4 file: ");  
 String path = input.nextLine();  
 input.close();  
 reader = new BufferedReader(new FileReader(path));  
 break;  
 }  
 catch(IOException ioExc)  
 {  
 System.*out*.println("Wrong path to file");  
 }  
 }  
 String line = "";  
 try  
 {  
 String newLine = reader.readLine();  
 while(newLine != null)  
 {  
 line += newLine + "\n";  
 newLine = reader.readLine();  
 }  
 }  
 catch(IOException ioExc)  
 {  
 ioExc.printStackTrace();  
 }  
 System.*out*.println("Program code is:");  
 System.*out*.println(line);  
  
 Stack<Character> stack = new Stack<>();  
 for(int i = 0; i < line.length(); i++)  
 {  
 if(line.charAt(i) == '(')  
 {  
 stack.push('(');  
 }  
 if(line.charAt(i) == ')')  
 {  
 if(stack.getSize() != 0)  
 {  
 stack.pop();  
 }  
 else  
 {  
 System.*out*.println("Code is unbalanced. Some '(' expected");  
 return false;  
 }  
 }  
 }  
 if(stack.getSize() != 0)  
 {  
 System.*out*.println("Code is unbalanced. Some ')' expected");  
 return false;  
 }  
 System.*out*.println("Code is balanced.");  
 return true;  
 }  
  
 public static boolean task5() {  
 Scanner input = new Scanner(System.*in*);  
 BufferedReader reader;  
 while(true)  
 {  
 try  
 {  
 System.*out*.print("Input path to task5 file: ");  
 String path = input.nextLine();  
 input.close();  
 reader = new BufferedReader(new FileReader(path));  
 break;  
 }  
 catch(IOException ioExc)  
 {  
 System.*out*.println("Wrong path to file");  
 }  
 }  
 String line = "";  
 try  
 {  
 String newLine = reader.readLine();  
 while(newLine != null)  
 {  
 line += newLine + "\n";  
 newLine = reader.readLine();  
 }  
 }  
 catch(IOException ioExc)  
 {  
 ioExc.printStackTrace();  
 }  
 System.*out*.println("Program code is:");  
 System.*out*.println(line);  
  
 Deque<Character> deque = new Deque<>();  
 for(int i = 0; i < line.length(); i++)  
 {  
 if(line.charAt(i) == '[')  
 {  
 deque.pushLast(']');  
 }  
 if(line.charAt(i) == ']')  
 {  
 if(deque.getSize() != 0)  
 {  
 deque.popLast();  
 }  
 else  
 {  
 System.*out*.println("Code is unbalanced. Some '[' expected");  
 return false;  
 }  
 }  
 }  
 if(deque.getSize() != 0)  
 {  
 System.*out*.println("Code is unbalanced. Some ']' expected");  
 return false;  
 }  
 System.*out*.println("Code is balanced.");  
 return true;  
 }  
  
 public static void task6() {  
 Scanner input = new Scanner(System.*in*);  
 BufferedReader reader;  
 while(true)  
 {  
 try  
 {  
 System.*out*.print("Input path to task6 file: ");  
 String path = input.nextLine();  
 input.close();  
 reader = new BufferedReader(new FileReader(path));  
 break;  
 }  
 catch(IOException ioExc)  
 {  
 System.*out*.println("Wrong path to file");  
 }  
 }  
 String line = "";  
 try  
 {  
 String newLine = reader.readLine();  
 while(newLine != null)  
 {  
 line += newLine;  
 newLine = reader.readLine();  
 }  
 }  
 catch(IOException ioExc)  
 {  
 ioExc.printStackTrace();  
 }  
 System.*out*.println("Source text is:");  
 System.*out*.println(line);  
  
 Stack<Character> stack = new Stack<>();  
 for(int i = 0; i < line.length(); i++)  
 {  
 if(stack.peek() == null)  
 {  
 stack.push(line.charAt(i));  
 }  
 else  
 {  
 String storage = "";  
 if(Character.*isDigit*(line.charAt(i)))  
 {  
 while(stack.peek() != null && Character.*isDigit*(stack.peek()))  
 {  
 storage += stack.pop();  
 }  
 stack.push(line.charAt(i));  
 for(int j = storage.length() - 1; j >= 0; j--)  
 {  
 stack.push(storage.charAt(j));  
 }  
 }  
 if(Character.*isLetter*(line.charAt(i)))  
 {  
 while(stack.peek() != null && Character.*isLetterOrDigit*(stack.peek()))  
 {  
 storage += stack.pop();  
 }  
 stack.push(line.charAt(i));  
 for(int j = storage.length() - 1; j >= 0; j--)  
 {  
 stack.push(storage.charAt(j));  
 }  
 }  
 if(!Character.*isDigit*(line.charAt(i)) && !Character.*isLetter*(line.charAt(i)))  
 {  
 while(stack.peek() != null)  
 {  
 storage += stack.pop();  
 }  
 stack.push(line.charAt(i));  
 for(int j = storage.length() - 1; j >= 0; j--)  
 {  
 stack.push(storage.charAt(j));  
 }  
 }  
 }  
 }  
 System.*out*.println("New order of symbols is:");  
 System.*out*.println(stack.toString());  
 }  
  
 public static void task7() {  
 Scanner input = new Scanner(System.*in*);  
 BufferedReader reader;  
 while(true)  
 {  
 try  
 {  
 System.*out*.print("Input path to task7 file: ");  
 String path = input.nextLine();  
 input.close();  
 reader = new BufferedReader(new FileReader(path));  
 break;  
 }  
 catch(IOException ioExc)  
 {  
 System.*out*.println("Wrong path to file");  
 }  
 }  
 String line = "";  
 ArrayList<Integer> numbers = new ArrayList<>();  
 try  
 {  
 String newLine = reader.readLine();  
 while(newLine != null)  
 {  
 line += newLine + " ";  
 newLine = reader.readLine();  
 }  
 }  
 catch(IOException ioExc)  
 {  
 ioExc.printStackTrace();  
 }  
 String number = "";  
 for(int i = 0; i < line.length(); i++)  
 {  
 if(line.charAt(i) == '-' && number.length() == 0 || Character.*isDigit*(line.charAt(i)))  
 {  
 number += line.charAt(i);  
 }  
 else  
 {  
 if(!number.equals("-") && number.length() != 0)  
 {  
 numbers.add(Integer.*parseInt*(number));  
 number = "";  
 }  
 }  
 }  
  
 System.*out*.println("Source order of numbers is:");  
 System.*out*.println(Arrays.*toString*(numbers.toArray()));  
 Deque<Integer> deq = new Deque<>();  
 for(int i = 0; i < numbers.size(); i++)  
 {  
 if(numbers.get(i) >= 0)  
 {  
 deq.pushFirst(numbers.get(i));  
 }  
 else  
 {  
 deq.pushLast(numbers.get(i));  
 }  
 }  
 while(deq.peekFirst() >= 0)  
 {  
 deq.pushLast(deq.popFirst());  
 }  
 while(deq.getSize() != 0)  
 {  
 if(deq.peekFirst() < 0)  
 {  
 System.*out*.print(deq.popFirst());  
 System.*out*.print(" ");  
 }  
 if(deq.peekFirst() >= 0)  
 {  
 System.*out*.print(deq.popLast());  
 System.*out*.print(" ");  
 }  
 }  
 }  
  
 public static void task8() {  
 Scanner input = new Scanner(System.*in*);  
 Stack<String> stack = new Stack<>();  
 BufferedReader reader;  
 FileWriter writer;  
 while(true)  
 {  
 try  
 {  
 System.*out*.print("Input path to task8 file: ");  
 String path = input.nextLine();  
 reader = new BufferedReader(new FileReader(path));  
  
 System.*out*.print("Output path to file: ");  
 path = input.nextLine();  
 input.close();  
 writer = new FileWriter(path, false);  
 break;  
 }  
 catch(IOException ioExc)  
 {  
 System.*out*.println("Wrong path to file");  
 }  
 }  
 try  
 {  
 String newLine = reader.readLine();  
 while(newLine != null)  
 {  
 stack.push(newLine);  
 newLine = reader.readLine();  
 }  
 while(stack.peek() != null)  
 {  
 writer.write(stack.pop());  
 writer.append('\n');  
 }  
 writer.flush();  
 }  
 catch(IOException ioExc)  
 {  
 ioExc.printStackTrace();  
 }  
 }  
  
 public static boolean task9() {  
 Scanner input = new Scanner(System.*in*);  
 BufferedReader reader;  
 while(true)  
 {  
 try  
 {  
 System.*out*.print("Input path to task9 file: ");  
 String path = input.nextLine();  
 input.close();  
 reader = new BufferedReader(new FileReader(path));  
 break;  
 }  
 catch(IOException ioExc)  
 {  
 System.*out*.println("Wrong path to file");  
 }  
 }  
 String line = "";  
 try  
 {  
 line = reader.readLine();  
 }  
 catch(IOException ioExc)  
 {  
 ioExc.printStackTrace();  
 }  
 Stack<Character> stack = new Stack<>();  
  
 for(int i = 0; i < line.length();)  
 {  
 if(line.charAt(i) != ')')  
 {  
 if(line.charAt(i) != '(')  
 {  
 stack.push(line.charAt(i));  
 }  
 i++;  
 }  
  
 else if(stack.getSize() != 0)  
 {  
 char elem = stack.pop();  
 char var = stack.peek();  
 stack.push(elem);  
 switch(var)  
 {  
 case 'N':  
 {  
 if(stack.peek() == 'T')  
 {  
  
 i++;  
 stack.pop();  
 stack.pop();  
 stack.push('F');  
 break;  
 }  
 else  
 {  
 i++;  
 stack.pop();  
 stack.pop();  
 stack.push('T');  
 break;  
 }  
 }  
 case 'A':  
 {  
 if(stack.peek() == 'T')  
 {  
 stack.pop();  
 stack.pop();  
 if(stack.peek() == 'T')  
 {  
 i++;  
 stack.pop();  
 stack.push('T');  
 break;  
 }  
 else  
 {  
 i++;  
 stack.pop();  
 stack.push('F');  
 break;  
 }  
 }  
 else  
 {  
 stack.pop();  
 stack.pop();  
 i++;  
 stack.pop();  
 stack.push('F');  
 break;  
 }  
 }  
 case 'X':  
 {  
 char first = stack.peek();  
 stack.pop();  
 stack.pop();  
 char second = stack.peek();  
 if(first == second)  
 {  
 i++;  
 stack.pop();  
 stack.push('F');  
 break;  
 }  
 else  
 {  
 i++;  
 stack.pop();  
 stack.push('T');  
 break;  
 }  
 }  
 case 'O':  
 {  
 char first = stack.peek();  
 stack.pop();  
 stack.pop();  
 char second = stack.peek();  
 if(first == 'F' && second == 'F')  
 {  
 i++;  
 stack.pop();  
 stack.push('F');  
 break;  
 }  
 else  
 {  
 i++;  
 stack.pop();  
 stack.push('T');  
 break;  
 }  
 }  
 }  
 }  
 }  
  
 if(stack.peek() == 'T')  
 {  
 System.*out*.println("True");  
 return true;  
 }  
 System.*out*.println("False");  
 return false;  
 }  
  
 public static char task10() {  
 Scanner input = new Scanner(System.*in*);  
 BufferedReader reader;  
 while(true)  
 {  
 try  
 {  
 System.*out*.print("Input path to task10 file: ");  
 String path = input.nextLine();  
 input.close();  
 reader = new BufferedReader(new FileReader(path));  
 break;  
 }  
 catch(IOException ioExc)  
 {  
 System.*out*.println("Wrong path to file");  
 }  
 }  
 String line = "";  
 try  
 {  
 line = reader.readLine();  
 }  
 catch(IOException ioExc)  
 {  
 ioExc.printStackTrace();  
 }  
 Stack<Character> stack = new Stack<>();  
 for(int i = 0; i < line.length();)  
 {  
 if(line.charAt(i) != ')')  
 {  
 if(line.charAt(i) != '(')  
 {  
 stack.push(line.charAt(i));  
 }  
 i++;  
 }  
  
 else if(stack.getSize() != 0)  
 {  
 char elem1 = stack.pop();  
 stack.pop();  
 char elem2 = stack.pop();  
 char var = stack.pop();  
 switch(var)  
 {  
 case 'N':  
 {  
 if(elem1 > elem2)  
 {  
 i++;  
 stack.push(elem2);  
 break;  
 }  
 else  
 {  
 i++;  
 stack.push(elem1);  
 break;  
 }  
 }  
 case 'M':  
 {  
 if(elem1 > elem2)  
 {  
 i++;  
 stack.push(elem1);  
 break;  
 }  
 else  
 {  
 i++;  
 stack.push(elem2);  
 break;  
 }  
 }  
 }  
 }  
 }  
 if(Character.*isDigit*(stack.peek()))  
 {  
 System.*out*.println(stack.peek());  
 return stack.peek();  
 }  
 return 0;  
  
 }  
  
 public static boolean task11() {  
 Scanner input = new Scanner(System.*in*);  
 BufferedReader reader;  
 while(true)  
 {  
 try  
 {  
 System.*out*.print("Input path to task11 file: ");  
 String path = input.nextLine();  
 input.close();  
 reader = new BufferedReader(new FileReader(path));  
 break;  
 }  
 catch(IOException ioExc)  
 {  
 System.*out*.println("Wrong path to file");  
 }  
 }  
 String line = "";  
 try  
 {  
 line = reader.readLine();  
 }  
 catch(IOException ioExc)  
 {  
 ioExc.printStackTrace();  
 }  
 Stack<Character> stack = new Stack<>();  
 for(int i = 0; i < line.length();)  
 {  
 if(line.charAt(i) != ')')  
 {  
 if(line.charAt(i) != '(')  
 {  
 stack.push(line.charAt(i));  
 }  
 i++;  
 }  
  
 else if(stack.getSize() != 0)  
 {  
 Character elem1 = stack.pop(); // элемент  
 Character var = stack.pop(); // операция  
 Character elem2 = stack.peek(); // элемент  
 if(var == null || elem2 == null)  
 {  
 break;  
 }  
 if((elem1 != 'x' && elem1 != 'y' && elem1 != 'z') || (elem2 != 'x' && elem2 != 'y' && elem2 != 'z'))  
 {  
 break;  
 }  
 stack.push(var);  
 stack.push(elem1);  
 if(var == '+' || var == '-')  
 {  
 i++;  
 stack.pop();  
 stack.pop();  
 stack.pop();  
 stack.push('x');  
 }  
 }  
 }  
 if(stack.getSize() == 1 && (stack.peek() == 'x' || stack.peek() == 'y' || stack.peek() == 'z'))  
 {  
 System.*out*.println("True");  
 return true;  
 }  
 System.*out*.println("False");  
 return false;  
 }  
  
  
  
  
}

Вывод

В данной лабораторной работе я получил представление о таких структурах данных как Стек и Дек, реализовал их, и с помощью них решил задачи.