Федеральное агентство связи

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Московский технический университет связи и информатики»

Кафедра «Информатики»

Дисциплина «СиАОД»

Лабораторная работа №4

Выполнил: студент группы БВТ1901

Адедиха Коффи Жермен

Руководитель:

Мелехин А.

Москва 2021

Лабораторная работа 4.

Реализация стека/дека

Реализовать следующие структуры данных:

● Стек (stack): операции для стека: инициализация, проверка на пустоту, добавление нового элемента в начало, извлечение элемента из начала;

● 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. Результат работы программы вывести на экран и сохранить в отдельном txt-файле

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 A на стержень C, 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 B. Реализовать алгоритм, используя три стека вместо стержней A, B, C. 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) | (A) | (X) | (O), 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: ::= | M(,) | N(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

Выполнение

Операции для стека : инициализация, проверка на пустоту, добавление нового элемента в начало, извлечение элемента из начала

```
public class MyStackImpl {

    private int stackSize;
    private int[] stackArr;
    private int top;


    public MyStackImpl(int size) {
        this.stackSize = size;
        this.stackArr = new int[stackSize];
        this.top = -1;
    }


    public void push(int entry) throws Exception {
        if(this.isStackFull()){
            throw new Exception("Stack is already full. Can not add element.");
        }
        System.out.println("Adding: "+entry);
        this.stackArr[++top] = entry;
    }



    public int pop() throws Exception {
        if(this.isStackEmpty()){
            throw new Exception("Stack is empty. Can not remove element.");
        }
        int entry = this.stackArr[top--];
        System.out.println("Removed entry: "+entry);
        return entry;
    }


    public int peek() {
        return stackArr[top];
    }
```

```
    public boolean isStackEmpty() {
        return (top == -1);
    }


    public boolean isStackFull() {
        return (top == stackSize - 1);
    }

    public static void main(String[] args) {
        MyStackImpl stack = new MyStackImpl(5);
       try {
            stack.push(10);
            stack.push(8);
            stack.push(3);
            stack.push(89);
            stack.pop();
            stack.push(34);
            stack.push(45);
            stack.push(78);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
       try {
           // stack.pop();
            stack.pop();
            stack.pop();
            stack.pop();
            stack.pop();
           // stack.pop();
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

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

```
public class Deque
{
    static final int MAX = 100;
    int arr[];
    int front;
    int rear;
    int size;

    public Deque(int size)
    {
        arr = new int[MAX];
        front = -1;
        rear = 0;
        this.size = size;
    }

    // Operations on Deque:void insertfront(int key);void insertrear(int key);voi
d deletefront();
    //  void deleterear();bool isFull();bool isEmpty();int getFront();int getRear
();


    // Checks whether Deque is full or not.
    boolean isFull()
    {
        return ((front == 0 && rear == size-1)||
            front == rear+1);
    }

    // Checks whether Deque is empty or not.
    boolean isEmpty ()
    {
        return (front == -1);
    }

    // Inserts an element at front
    void insertfront(int key)
    {
```

```
        // check whether Deque if full or not
        if (isFull())
        {
            System.out.println("Overflow");
            return;
        }

        // If queue is initially empty
        if (front == -1)
        {
            front = 0;
            rear = 0;
        }

        // front is at first position of queue
        else if (front == 0)
            front = size - 1 ;

        else // decrement front end by '1'
            front = front-1;

        // insert current element into Deque
        arr[front] = key ;
    }

    // function to inset element at rear end
    // of Deque.
    void insertrear(int key)
    {
        if (isFull())
        {
            System.out.println(" Overflow ");
            return;
        }

        // If queue is initially empty
        if (front == -1)
        {
            front = 0;
            rear = 0;
        }

        // rear is at last position of queue
        else if (rear == size-1)
```

```
            rear = 0;

        // increment rear end by '1'
        else
            rear = rear+1;

        // insert current element into Deque
        arr[rear] = key ;
    }

    // Deletes element at front end of Deque
    void deletefront()
    {
        // check whether Deque is empty or not
        if (isEmpty())
        {
            System.out.println("Queue Underflow\n");
            return ;
        }

        // Deque has only one element
        if (front == rear)
        {
            front = -1;
            rear = -1;
        }
        else
            // back to initial position
            if (front == size -1)
                front = 0;

            else // increment front by '1' to remove current
                // front value from Deque
                front = front+1;
    }

    // Delete element at rear end of Deque
    void deleterear()
    {
        if (isEmpty())
        {
            System.out.println(" Underflow");
            return ;
        }
```

```
        // Deque has only one element
        if (front == rear)
        {
            front = -1;
            rear = -1;
        }
        else if (rear == 0)
            rear = size-1;
        else
            rear = rear-1;
    }

    // Returns front element of Deque
    int getFront()
    {
        // check whether Deque is empty or not
        if (isEmpty())
        {
            System.out.println(" Underflow");
            return -1 ;
        }
        return arr[front];
    }

    // function return rear element of Deque
    int getRear()
    {
        // check whether Deque is empty or not
        if(isEmpty() || rear < 0)
        {
            System.out.println(" Underflow\n");
            return -1 ;
        }
        return arr[rear];
    }

    public static void main(String[] args)
    {

        Deque dq = new Deque(5);

        System.out.println("Insert element at rear end : 5 ");
        dq.insertrear(5);
```

```
        System.out.println("insert element at rear end : 10 ");
        dq.insertrear(10);

        System.out.println("get rear element : "+ dq.getRear());

        dq.deleterear();
        System.out.println("After delete rear element new rear become : " +
                                dq.getRear());

        System.out.println("inserting element at front end");
        dq.insertfront(15);

        System.out.println("get front element: " +dq.getFront());

        dq.deletefront();

        System.out.println("After delete front element new front become : " +
                                    + dq.getFront());

    }
}
```

- **Задания**

```
import java.util.*;

public class BalancedBrackets {

    // function to check if brackets are balanced
    static boolean areBracketsBalanced(String expr)
    {
        // Using ArrayDeque is faster than using Stack class
        Deque<Character> stack
            = new ArrayDeque<Character>();

        // Traversing the Expression
        for (int i = 0; i < expr.length(); i++)
        {
            char x = expr.charAt(i);
```

```
        if (x == '(' || x == '[' || x == '{')
        {
            // Push the element in the stack
            stack.push(x);
            continue;
        }


        if (stack.isEmpty())
            return false;
        char check;
        switch (x) {
        case ')':
            check = stack.pop();
            if (check == '{' || check == '[')
                return false;
            break;

        case '}':
            check = stack.pop();
            if (check == '(' || check == '[')
                return false;
            break;

        case ']':
            check = stack.pop();
            if (check == '(' || check == '{')
                return false;
            break;
        }
    }

    // Check Empty Stack
    return (stack.isEmpty());
}


public static void main(String[] args)
{
    String expr = "([{}])";


    if (areBracketsBalanced(expr))
```

```
            System.out.println("Balanced ");
        else
            System.out.println("Not Balanced ");
    }
}
```

```
import java.io.* ; import java.util.* ;
public class RangeNumber  {
    public static void main(String[] args) { int[] array = null;
        try (BufferedReader in = new BufferedReader(new FileReader("C:\\Users\\so
ny\\Desktop\\sem4\\СиАОД\4- Реализация стека и дека\\input.txt")))
        {
            array = in.lines().mapToInt(Integer::parseInt).toArray();
        }
        catch (IOException | NumberFormatException e)
        {
            e.printStackTrace();
        }
        Deque deq = new Deque(array.length);

        for (int i = 0; i < array.length; i++) { deq.addElementBot(array[i]);
        }
        String str = ""; String str1 = "";
        while (deq.isEmpty() == false) { if (deq.getBot() < 0) {
            str += " " + Integer.toString(deq.getBot());
        }
        else {
            str1 += " " + Integer.toString(deq.getBot());
        }
            deq.deleteElementBot();
        }
        System.out.println(str1 + str);
        try(FileWriter writer = new FileWriter("C:\\Users\\sony\\Desktop\\sem4\\С
иАОД\4- Реализация стека и дека\\output.txt", false))
        {

            writer.write(str1+str);


            writer.flush();
```

```
        }
        catch(IOException ex){

            System.out.println(ex.getMessage());
        }
    }
}
```

```
import java.util.Date;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Random;
import java.util.Scanner;
import java.util.Collections;
import java.io.*;

public class Zadania {
    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        System.out.print("Введите номер таска: ");
        String task = input.nextLine();
        if(task.equals("1")) {
            task1(input);
        }
        if(task.equals("2")) {
            task2(input);
        }
        if(task.equals("3")) {
            task3(input);
        }
        if(task.equals("4")) {
            task4(input);
        }
        if(task.equals("5")) {
            task5(input);
        }
        if(task.equals("6")) {
            task6(input);
        }
        if(task.equals("7")) {
            task7(input);
```

```
        }
        if(task.equals("8")) {
            task8(input);
        }
        if(task.equals("9")) {
            task9(input);
        }
        if(task.equals("10")) {
            task10(input);
        }
        if(task.equals("11")) {
            task11(input);
        }
        input.close();
    }


    //таск 1
    //Отсортировать строки файла в алфавитном порядке с
    //использованием двух деков.

    public static void task1(Scanner input) {
        BufferedReader reader;
        Deque<String> deq1 = new Deque<>();
        while(true) {
            try {
                System.out.print("Введите путь до файла: ");
                String path = input.nextLine();
                input.close();
                reader = new BufferedReader(new FileReader(path));
                break;
            }
            catch(IOException ioExc) {
                System.out.println("Неверный путь");
            }
        }
        try {
            String line = reader.readLine();
            while(line != null) {
                deq1.addLast(line);
                line = reader.readLine();

            }
        }
```

```
        catch(IOException ioExc) {
            ioExc.printStackTrace();
        }

        Deque<String> deq2 = new Deque<>();
        deq2.addFirst(deq1.removeFirst());

        while(!deq1.isEmpty()) {
            String first = deq1.getFirst().toLowerCase();
            String second = deq2.getFirst().toLowerCase();
            boolean compareWithLast = false;
            if (deq1.getFirst().length() >= deq2.getFirst().length()) {
                for (int i = 0; i < second.length(); i++) {
                    if (first.charAt(i) < second.charAt(i)) {
                        deq2.addFirst(deq1.getFirst());
                        deq1.removeFirst();
                        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)) {
                        deq2.addFirst(deq1.getFirst());
                        deq1.removeFirst();
                        break;
                    }
                    if (first.charAt(i) > second.charAt(i)) {
                        compareWithLast = true;
                        break;
                    }
                }
            }
            if (compareWithLast) {
                second = deq2.getLast().toLowerCase();
            }
            if (deq1.getFirst().length() >= deq2.getFirst().length() && compareWi
thLast) {
                for (int i = 0; i < second.length(); i++) {
                    if (first.charAt(i) > second.charAt(i)) {
```

```
                    deq2.addLast(deq1.getFirst());
                    deq1.removeFirst();
                    break;
                }
                if (first.charAt(i) < second.charAt(i)) {
                    deq1.addLast(deq2.removeLast());
                    break;
                }
            }
        }
        else if (compareWithLast) {
            for(int i = 0; i < first.length(); i++) {
                if (first.charAt(i) > second.charAt(i)) {
                    deq2.addLast(deq1.getFirst());
                    deq1.removeFirst();
                    break;
                }
                if (first.charAt(i) < second.charAt(i)) {
                    deq1.addLast(deq2.removeLast());
                    break;
                }
            }
        }
    }
    System.out.println(deq2.toString());
}


//таск2
/* 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 */

public static void task2(Scanner input) {
    BufferedReader reader;
    Deque<Character> deq = new Deque<>();
    System.out.print("Введите строку: ");
    String decoder = input.nextLine().toLowerCase();
    for(int i = 0; i < decoder.length(); i++) {
        deq.addLast(decoder.charAt(i));
    }
    while(true) {
        try {
```

```
                System.out.print("Укажите путь до файла: ");
                String path = input.nextLine();
                input.close();
                reader = new BufferedReader(new FileReader(path));
                break;
            }
            catch(IOException ioExc) {
                System.out.println("Неверный путь до файла");
            }
        }
        String line = "";
        try {
            String newLine = reader.readLine();
            while (newLine != null) {
                line += newLine + " ";
                newLine = reader.readLine();
            }
        }
        catch (IOException ioExc) {
            ioExc.printStackTrace();
        }
        System.out.println("Закодированное сообщение:");
        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.getFirst() == line.charAt(index)) {
                    canDecode = true;
                    break;
                }
                deq.addLast(deq.removeFirst());
            }
            if (!canDecode) {
                System.out.println("Не удается декодировать входное сообщение из-
за отсутствия символов в декодере");
```

```
                break;
            }
            deq.addLast(deq.removeFirst());
            deq.addLast(deq.removeFirst());
            decodedMessage += deq.getFirst();
            index++;
        }
        if (canDecode) {
            System.out.println("Расшифрованное сообщение:");
            System.out.println(decodedMessage);
        }
    }


//таск3
    /*Даны три стержня и n дисков различного размера. Диски можно надевать на
стержни, образуя из них башни. Перенести n дисков со стержня A на стержень C,
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 B.
Реализовать алгоритм, используя три стека вместо стержней A, B, C. Информация
о дисках хранится в исходном файле.
*/

    public static void task3 (Scanner input) {
        BufferedReader reader;
        ArrayList<Stack<Integer>> stacks = new ArrayList<Stack<Integer>>();
        stacks.add(new Stack<Integer>());
        stacks.add(new Stack<Integer>());
        stacks.add(new Stack<Integer>());
        while(true) {
            try {
                System.out.print("Укажите путь до файла: ");
                String path = input.nextLine();
                input.close();
                reader = new BufferedReader(new FileReader(path));
                break;
            }
            catch (IOException ioExc) {
                System.out.println("Неверный путь до файла");
            }
        }
```

```
        try {
            ArrayList<Integer> disks = new ArrayList<>();
            String line = reader.readLine();
            String[] numbers;
            while (line != null) {
                line = line.trim();
                numbers = line.split(" ");
                for(int i = 0; i < numbers.length; i++) {
                    disks.add(Integer.parseInt(numbers[i]));
                }
                line = reader.readLine();

            }
            Collections.sort(disks, Collections.reverseOrder());
            for (int i = 0; i < disks.size(); i++) {
                stacks.get(0).push(disks.get(i));
            }
        }
        catch(IOException ioExc) {
            ioExc.printStackTrace();
        }
        catch(NumberFormatException numExc) {
            System.out.println("Некорректный формат чисел в файле");
        }
        System.out.println(stacks.get(0).toString());
        int count = stacks.get(0).getSize();
        hanoiTowers(count, 0, 2, 1, stacks);
        System.out.print(stacks.get(0).toString());
        System.out.print(stacks.get(1).toString());
        System.out.println(stacks.get(2).toString());
    }

    public static void hanoiTowers(int count, int start, int middle, int end, Arr
ayList<Stack<Integer>> stacks) {
        if(count > 0) {
            hanoiTowers(count-1, start, end, middle, stacks);
            stacks.get(middle).push(stacks.get(start).pop());
            hanoiTowers(count-1, end, middle , start, stacks);

        }
    }


    //таск4
```

```java
/*Дан текстовый файл с программой на алгоритмическом языке. За один просмотр
файла проверить баланс круглых скобок в тексте, используя стек.*/
public static boolean task4(Scanner input) {
    BufferedReader reader;
    while(true) {
        try {
            System.out.print("Укажите путь до файла: ");
            String path = input.nextLine();
            input.close();
            reader = new BufferedReader(new FileReader(path));
            break;
        }
        catch(IOException ioExc)
        {
            System.out.println("Неверный путь до файла");
        }
    }
    String line = "";
    try {
        String newLine = reader.readLine();
        while(newLine != null) {
            line += newLine + "\n";
            newLine = reader.readLine();
        }
    }
    catch(IOException ioExc) {
        ioExc.printStackTrace();
    }
    System.out.println("Код программы:");
    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("Ожидается '('");
                return false;
            }
```

```
            }
        }
        if (stack.getSize() != 0) {
            System.out.println("Ожидается ')'");
            return false;
        }
        System.out.println("Код верен");
        return true;
    }


    //таск5
    /*Дан текстовый файл с программой на алгоритмическом языке. За один просмотр
    файла проверить баланс квадратных скобок в тексте, используя дек*/
    public static boolean task5(Scanner input) {
        BufferedReader reader;
        while(true) {
            try {
                System.out.print("Укажите путь до файла: ");
                String path = input.nextLine();
                input.close();
                reader = new BufferedReader(new FileReader(path));
                break;
            }
            catch(IOException ioExc) {
                System.out.println("Неверный путь до файла");
            }
        }
        String line = "";
        try {
            String newLine = reader.readLine();
            while(newLine != null) {
                line += newLine + "\n";
                newLine = reader.readLine();
            }
        }
        catch(IOException ioExc) {
            ioExc.printStackTrace();
        }
        System.out.println("Код программы:");
        System.out.println(line);

        Deque<Character> deq = new Deque<>();
        for(int i = 0; i < line.length(); i++) {
```

```
            if(line.charAt(i) == '[') {
                deq.addLast('[');
            }
            if(line.charAt(i) == ']') {
                if(deq.getSize() != 0) {
                    deq.removeLast();
                }
                else {
                    System.out.println("Ожидается '['");
                    return false;
                }
            }
        }
        if(deq.getSize() != 0) {
            System.out.println("Ожидается ']'");
            return false;
        }
        System.out.println("Код верен");
        return true;
    }


    //таск6
    /*Дан файл из символов. Используя стек, за один просмотр файла напечатать
    сначала все цифры, затем все буквы, и, наконец, все остальные символы, сохран
яя
    исходный порядок в каждой группе символов*/

    public static void task6(Scanner input) {
        BufferedReader reader;
        while(true) {
            try {
                System.out.print("Укажите путь до файла: ");
                String path = input.nextLine();
                input.close();
                reader = new BufferedReader(new FileReader(path));
                break;
            }
            catch(IOException ioExc) {
                System.out.println("Неверный путь до файла");
            }
        }
        String line = "";
        try {
```

```
            String newLine = reader.readLine();
            while (newLine != null) {
                line += newLine;
                newLine = reader.readLine();
            }
        }
        catch (IOException ioExc) {
            ioExc.printStackTrace();
        }

        System.out.println("Исходный текст:");
        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("Новый порядок символов:");
    System.out.println(stack.toString());
}


//таск7
/*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*/

public static void task7(Scanner input) {
    BufferedReader reader;
    while(true) {
        try {
            System.out.print("Укажите путь к файлу: ");
            String path = input.nextLine();
            input.close();
            reader = new BufferedReader(new FileReader(path));
            break;
        }
        catch(IOException ioExc) {
            System.out.println("Неверный путь");
        }
    }
    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("Исходный порядок чисел:");
        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.addFirst(numbers.get(i));
            }
            else {
                deq.addLast(numbers.get(i));
            }
        }
        while (deq.getFirst() >= 0) {
            deq.addLast(deq.removeFirst());
        }
        while (deq.getSize() != 0) {
            if(deq.getFirst() < 0) {
                System.out.print(deq.removeFirst());
                System.out.print(" ");
            }
            if (deq.getFirst() >= 0) {
                System.out.print(deq.removeLast());
                System.out.print(" ");
            }
        }
    }
```

```java
//таск8
/*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. */

public static void task8(Scanner input) {
    Stack<String> stack = new Stack<>();
    BufferedReader reader;
    FileWriter writer;
    while(true) {
        try {
            System.out.print("Укажите путь до файла: ");
            String path = input.nextLine();
            reader = new BufferedReader(new FileReader(path));

            System.out.print("Укажите путь к выходному файлу: ");
            path = input.nextLine();
            input.close();
            writer = new FileWriter(path, false);
            break;
        }
        catch(IOException ioExc)
        {
            System.out.println("Неверный путь до файла");
        }
    }
    String line = "";
    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();
    }
}
```

```java
    //таск9
    /*Дан текстовый файл. Используя стек, вычислить значение логического выражени
я,
    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. */

    public static boolean task9(Scanner input) {
        BufferedReader reader;
        while(true) {
            try {
                System.out.print("Укажите путь до файла: ");
                String path = input.nextLine();
                input.close();
                reader = new BufferedReader(new FileReader(path));
                break;
            }
            catch(IOException ioExc) {
                System.out.println("Неверный путь до файла");
            }
        }
        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;
}
```

```java
//таск10
/*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.*/

public static int task10(Scanner input) {
    BufferedReader reader;
    while(true) {
        try {
            System.out.print("Укажите путь до файла: ");
            String path = input.nextLine();
            input.close();
            reader = new BufferedReader(new FileReader(path));
            break;
        }
        catch(IOException ioExc) {
            System.out.println("Неверный путь до файла");
        }
    }
    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;

    }


    //таск11
    /*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 */

    public static boolean task11(Scanner input) {
        BufferedReader reader;
```

```
        while(true) {
            try {
                System.out.print("Укажите путь к файлу: ");
                String path = input.nextLine();
                input.close();
                reader = new BufferedReader(new FileReader(path));
                break;
            }
            catch (IOException ioExc) {
                System.out.println("Указан неверный путь");
            }
        }
        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();
                Character var = stack.pop();
                Character elem2 = stack.peek();
                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;
    }
}
```

№ Теста	Исходный массив	Результат работы программы
1	155 -13 -4 25 -1 -15 7	155 25 7 -13 -4 -1 -15
2	-18 -2 7 -7 2 8 15 -45 -6 -4	-18 -2 -7 -45 -6 -4 7 2 8 15
3	15 78 -8 -5 -4 -15 5 17 25	-8 -5 -4 -15 15 78 5 17 25

.....

Работы программ

input.txt – Блокнот

Файл Правка Формат Вид Справка

155
-13
-4
25
-1
-15
7

output.txt – Блокнот

Файл Правка Формат Вид Справка

155 25 7 -13 -4 -1 -15

input.txt – Блокнот

Файл Правка Формат Вид Справка

18 -2 7 -7 2 8 15 -45 -6 -4

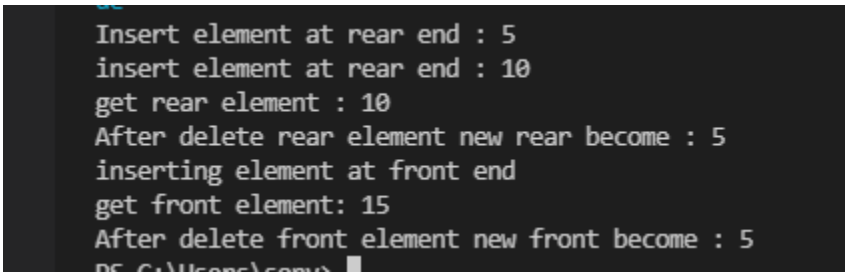
output.txt – Блокнот

Файл Правка Формат Вид Справка

-18 -2 -7 -45 -6 -4 7 2 8 15

```
Adding: 10
Adding: 8
Adding: 3
Adding: 89
Removed entry: 89
Adding: 34
Adding: 45
Stack is already full. Can not add element.
Removed entry: 45
Removed entry: 34
Removed entry: 3
Removed entry: 8
```

- Стек: инициализация, проверка на пустоту, добавление нового элемента в начало, извлечение элемента из начала;



- Дек : инициализация, проверка на пустоту, добавление нового элемента в начало, добавление нового элемента в конец, извлечение элемента из начала, извлечение элемента из конца.

Вывод

С этой лабораторной работы ,были получены знание и реализация структуры данных как стеки и деки