

Министерство цифрового развития связи и массовых коммуникаций РФ

Государственное бюджетное образовательное учреждение высшего
образования

Ордена Трудового Красного Знамени

«Московский технический университет связи и информатики»

Кафедра «Математическая кибернетика и информационные технологии»

Отчет по лабораторной работе №4
по дисциплине «Структуры и алгоритмы обработки данных»

Выполнил: студент группы БВТ1901

Кускова А. Е.

Руководитель:

Мелехин А. А.

Москва

2021

Задание:

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.

Техническое задание:

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:

$$\langle \text{ЛВ} \rangle ::= \text{T} \mid \text{F} \mid (\text{N}\langle \text{ЛВ} \rangle) \mid (\langle \text{ЛВ} \rangle \text{A} \langle \text{ЛВ} \rangle) \mid (\langle \text{ЛВ} \rangle \text{X} \langle \text{ЛВ} \rangle) \mid (\langle \text{ЛВ} \rangle \text{O} \langle \text{ЛВ} \rangle),$$

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:

$$\langle \text{Формула} \rangle ::= \langle \text{Цифра} \rangle \mid \text{M}(\langle \text{Формула} \rangle, \langle \text{Формула} \rangle) \mid \text{N}(\langle \text{Формула} \rangle, \langle \text{Формула} \rangle)$$
$$\langle \text{Цифра} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

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:

$$\langle \text{Формула} \rangle ::= \langle \text{Терм} \rangle \mid \langle \text{Терм} \rangle + \langle \text{Формула} \rangle \mid \langle \text{Терм} \rangle - \langle \text{Формула} \rangle$$
$$\langle \text{Терм} \rangle ::= \langle \text{Имя} \rangle \mid (\langle \text{Формула} \rangle)$$
$$\langle \text{Имя} \rangle ::= x \mid y \mid z$$

Выполнение:

Реализация стека:

```
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;
    }
    //добавить элемент в верхнюю часть стека
    public void push(E element) {
        if (element != null) {
            this.stack.add(element);
            this.size++;
        }
    }
    //возвратим верхний эл. без удаления
    public E peek() {
        if (size != 0) {
            return this.stack.get(this.size-1);
        }
        System.out.println("Stack is empty");
        return null;
    }
}
```

```

    }
    //снимаем верхний элемент (с удалением)
    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;
    }
}

```

Реализация дека:

```

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;
}
//добавим последним
public void addLast(E element) {
    this.deque.add(element);
    this.size++;
}
//добавим первым
public void addFirst(E element) {
    this.deque.add(0, element);
    this.size++;
}
//вернем последний
public E getLast() {
    if(size != 0) {
        return this.deque.get(this.size-1);
    }
    System.out.println("Deque is empty");
    return null;
}
//вернем первый
public E getFirst() {
    if(size != 0) {
        return this.deque.get(0);
    }
}
```

```
        System.out.println("Deque is empty");
        return null;
    }
    //удалим последний
    public E removeLast() {
        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;
    }
    //удалим первый
    public E removeFirst() {
        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;
    }
}
```

Реализация задач:

Задача 1:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;
```

```
public class task1 {
    public static void main(String[] args) {
        task();
    }
    public static void task() {
        Scanner input = new Scanner(System.in);
        BufferedReader reader;
        DequeU<String> deq1 = new DequeU<>();
        while(true) {
            try {
                System.out.print("Input path 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();
    while(line != null)
    {
        deq1.addLast(line);
        line = reader.readLine();
    }
}
catch(IOException ioExc)
{
    ioExc.printStackTrace();
}

```

```

DequeU<String> deq2 = new DequeU<>();
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());// доавляем вначало второго дека
первый элемент первого

```
    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(deq1.isEmpty()) {
    break;
}
if(compareWithLast)
{
    second = deq2.getLast().toLowerCase(); // toLowerCase потому что
аски разный
}
if(deq1.getFirst().length() >= deq2.getFirst().length() &&
compareWithLast)
{
    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());
}
}

```

Task 2:

```

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

public class task2 {
    public static void main(String[] args) {
        task();
    }
}

```

```

public static void task()
{
    Scanner input = new Scanner(System.in);
    BufferedReader reader;
    DequeU<Character> deq = new DequeU<>();
    System.out.print("Input decoder string: "); //(расшифровывает)
    String decoder = input.nextLine().toLowerCase();
    for(int i = 0; i < decoder.length(); i++)
    {
        deq.addLast(decoder.charAt(i));
    }
    while(true)
    {
        try
        {
            System.out.print("Input path 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");
        }
    }
    String line = "";
    try
    {

```

```

//заполняем строку содержимым файла
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);

//инициализируем новую строчку для расшифрованного сообщения
String decodedMessage = "";
line = line.toLowerCase();
line = line.trim(); //убираем лишние пробелы
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("Can't decode input message because of lack of
symbols in decoder");
    break;
}
deq.addLast(deq.removeFirst());
deq.addLast(deq.removeFirst());
decodedMessage += deq.getFirst();
index++;
}
if(canDecode)
{
    System.out.println("Decoded message is:");
    System.out.println(decodedMessage);
}
}
}

```

Такк 3:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

public class task3 {
    public static void main(String[] args) {
        task();
    }
    public static void task()
    {
        Scanner input = new Scanner(System.in);
        BufferedReader reader;
        ArrayList<StackU<Integer>> stacks = new ArrayList<StackU<Integer>>();
        stacks.add(new StackU<Integer>());
        stacks.add(new StackU<Integer>());
        stacks.add(new StackU<Integer>());
        while(true)
        {
            try
            {
                System.out.print("Input path 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
    {
        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();
        }
        // сортируем
        //записываем диски в первый стек
        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("Incorrect format of numbers in file");
}
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,
ArrayList<StackU<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);

        }
    }
}

```

Task 4:

```
import java.io.BufferedReader;
```

```
import java.io.FileReader;
```

```
import java.io.IOException;
```

```
import java.util.Scanner;
```

```
public class task4 {
```

```
    public static void main(String[] args) {
```

```
        System.out.println(task());
```

```
    }
```

```
    public static boolean task()
```

```
    {
```

```
        Scanner input = new Scanner(System.in);
```

```
        BufferedReader reader;
```

```
        while(true)
```

```
        {
```

```
            try
```

```
            {
```

```
                System.out.print("Input path 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");
```

```
            }
```

```
        }
```

```
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);
```

```
StackU<Character> stack = new StackU<>();
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;
}
}

```

Task 5:

```

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

public class task5 {
    public static void main(String[] args) {
        System.out.println(task());
    }
    public static boolean task()

```

```

{
    Scanner input = new Scanner(System.in);
    BufferedReader reader;
    while(true)
    {
        try
        {
            System.out.print("Input path 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");
        }
    }
    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);

DequeU<Character> deq = new DequeU<>();
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("Code is unbalanced. Some '[' expected");
            return false;
        }
    }
}
if(deq.getSize() != 0)
{
    System.out.println("Code is unbalanced. Some ']' expected");
}

```

```
        return false;
    }
    System.out.println("Code is balanced.");
    return true;
}
}
```

Task 6:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;
```

```
public class task6 {
    public static void main(String[] args) {
        task();
    }
    public static void task()
    {
        Scanner input = new Scanner(System.in);
        BufferedReader reader;
        while(true)
        {
            try
            {
                System.out.print("Input path to file: ");
                String path = input.nextLine();
                input.close();
                reader = new BufferedReader(new FileReader(path));
            }
            catch (IOException e) {
                System.out.println("Error: " + e.getMessage());
            }
        }
    }
}
```



```

        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);

StackU<Character> stack = new StackU<>();
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());
}
}

```

Tack 7:

```

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Scanner;

```

```

public class task7 {
    public static void main(String[] args) {
        task();
    }
    public static void task()
    {
        Scanner input = new Scanner(System.in);
        BufferedReader reader;
        while(true)
        {
            try
            {
                System.out.print("Input path 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");
            }
        }
        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()));

```

```
DequeU<Integer> deq = new DequeU<>();
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(" ");
    }
}
```

```
}  
}
```

Таск 8:

```
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.Scanner;
```

```
public class task8 {  
    public static void main(String[] args) {  
        task();  
    }  
    public static void task()  
    {  
        Scanner input = new Scanner(System.in);  
        StackU<String> stack = new StackU<>();  
        BufferedReader reader;  
        FileWriter writer;  
        while(true)  
        {  
            try  
            {  
                System.out.print("Input path to 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();
}

```



```
    }  
}
```

Task 9:

```
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
import java.util.Scanner;
```

```
public class task9 {  
    public static void main(String[] args) {  
        task();  
    }  
    public static boolean task()  
    {  
        Scanner input = new Scanner(System.in);  
        BufferedReader reader;  
        while(true)  
        {  
            try  
            {  
                System.out.print("Input path 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");
    }
}
String line = "";
try
{
    line = reader.readLine();
}
catch(IOException ioExc)
{
    ioExc.printStackTrace();
}
StackU<Character> stack = new StackU<>();

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;
}
}

```

Task 10:

```

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

```

```
public class task10 {  
    public static void main(String[] args) {  
        System.out.println(task());  
    }  
    public static char task()  
    {  
        Scanner input = new Scanner(System.in);  
        BufferedReader reader;  
        while(true)  
        {  
            try  
            {  
                System.out.print("Input path 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");  
            }  
        }  
        String line = "";  
        try  
        {  
            line = reader.readLine();  
        }  
        catch(IOException ioExc)
```

```

{
    ioExc.printStackTrace();
}

StackU<Character> stack = new StackU<>();
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;

}
}

```

Task 11:

```

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

public class task11 {
    public static void main(String[] args) {
        task();
    }
    public static boolean task()
    {
        Scanner input = new Scanner(System.in);
        BufferedReader reader;
        while(true)
        {
            try
            {
                System.out.print("Input path to file: ");
                String path = input.nextLine();
                input.close();
            }
            catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

```

```

        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();
}
StackU<Character> stack = new StackU<>();
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:

Разработали такие структуры данных, как стек и дек. Разработали программу обработки данных, содержащихся в заранее подготовленном txt-файле, в соответствии с заданиями, применив указанную в задании структуру данных.