

Министерство цифрового развития, связи и массовых коммуникаций
Государственное образовательное учреждение высшего образования

Ордена Трудового Красного Знамени

«Московский технический университет связи и информатики»

Задачи для самостоятельного решения

по дисциплине «Структура и алгоритмы обработки данных»

Выполнил студент группы БФИ 1901:

Соколовский Никита

Проверил:

Кутейников Иван Алексеевич

Москва 2021

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 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:
 $\langle \text{ЛВ} \rangle ::= \text{T} \mid \text{F} \mid (\text{N}\langle \text{ЛВ} \rangle) \mid (\langle \text{ЛВ} \rangle \text{A} \langle \text{ЛВ} \rangle) \mid (\langle \text{ЛВ} \rangle \text{X} \langle \text{ЛВ} \rangle) \mid (\langle \text{ЛВ} \rangle \text{O} \langle \text{ЛВ} \rangle),$
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:
 $\langle \text{Формула} \rangle ::= \langle \text{Терм} \rangle \mid \langle \text{Терм} \rangle + \langle \text{Формула} \rangle \mid \langle \text{Терм} \rangle - \langle \text{Формула} \rangle$
 $\langle \text{Терм} \rangle ::= \langle \text{Имя} \rangle \mid (\langle \text{Формула} \rangle)$
 $\langle \text{Имя} \rangle ::= x \mid y \mid z$

Код программы

```
package fourthLab;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.*;

public class twoDec {

    public static void main(String[] args) throws IOException {
        System.out.println("Задание №1:");
        System.out.println(firstTask("F:\\TestLab4\\Task1.txt"));

        System.out.println();
        System.out.println("Задание №2:");
        System.out.println(secondTask("helloworld", "F:\\TestLab4\\Task2.txt",
true));

        System.out.println();
        System.out.println("Задание №3:");
        System.out.println(ThirdTask(3, 1, 2, 3, true));

        System.out.println();
        System.out.println("Задание №4:");
        System.out.println(fourthTask("F:\\TestLab4\\Task4.txt", '{', '}'));

        System.out.println();
        System.out.println("Задание №5:");
        System.out.println(fifthTask("F:\\TestLab4\\Task5.txt", '[', ']'));

        System.out.println();
        System.out.println("Задание №6:");
        System.out.println(sixTask("F:\\TestLab4\\Task6.txt"));

        System.out.println();
        System.out.println("Задание №7:");
        System.out.println(sevenTask("F:\\TestLab4\\Task7.txt"));

        System.out.println();
        System.out.println("Задание №8:");
        System.out.println(eightTask("F:\\TestLab4\\Task8.txt"));

        System.out.println();
        System.out.println("Задание №9:");
        nineTask("F:\\TestLab4\\Task9.txt");

        System.out.println();
        System.out.println("Задание №10:");
        System.out.println(tenTask("F:\\TestLab4\\Task10.txt"));

        System.out.println();
        System.out.println("Задание №11:");
        System.out.println(elevenTask("F:\\TestLab4\\Task11.txt"));
    }

    /*
    Task 1
    */
```

```

public static Deque<String> firstTask(String file) throws IOException {
    Scanner sc = new Scanner(new File(file));
    List<String> lines = new ArrayList<>();
    while (sc.hasNextLine()) {
        lines.add(sc.nextLine());
    }

    String[] arr = lines.toArray(new String[0]);

    int n = arr.length;
    // for (int i = 0; i<arr.length;i++){
    //     System.out.println(arr[i]);
    // }

    Deque<String> BooksDeque = new ArrayDeque<>();

    if (n < 1){
        return BooksDeque;
    } else if(n == 1){
        BooksDeque.addFirst(arr[0]);
        return BooksDeque;
    }
    Deque<String> SortedDeque = new ArrayDeque<>();

    BooksDeque.addFirst(arr[0]);

    for (int i = 1; i < n; i++){
        while ((BooksDeque.size() > 0) &&
isEquals(BooksDeque.getFirst().toLowerCase(), arr[i].toLowerCase()))
        {
            SortedDeque.addLast(BooksDeque.removeFirst());
        }
        BooksDeque.addFirst(arr[i]);
        while (!SortedDeque.isEmpty())
        {
            BooksDeque.addFirst(SortedDeque.removeLast());
        }
    }

    return BooksDeque;
}

static Boolean isEquals(String book, String book2){
    for (int i = 0; i < Math.min(book2.length(), book.length()); i++)
    {
        if (book.charAt(i) != book2.charAt(i))
        {
            return book.charAt(i) <= book2.charAt(i); //book2 раньше
        }
    }
    return book.length() <= book2.length();
}

/*
Task 2
*/

public static String secondTask(String str, String file, Boolean encrypt)
throws FileNotFoundException {

    char[] arr = readCharFromFile(file);

    int n = arr.length;

```

```

Deque<Character> parens = new ArrayDeque<>();

for (int i = 0; i < arr.length; i++)
{
    parens.addFirst(arr[i]);
}
if (encrypt)
    return encryption(parens, str);
else
    return decryption(parens, str);
}

public static String encryption(Deque<Character> deque, String str){    //
Шифрование
    String enc = "";
    for (int i = 0; i < str.length(); i++)
    {
        enc += getSymbol(deque, str.charAt(i), -1);
    }
    return enc;
}

static char getSymbol(Deque<Character> deque, char a, int i) {
    while (deque.getFirst() != a)
    {
        deque = Spin(deque, 1);
    }
    deque = Spin(deque, i);
    return deque.getFirst();
}

public static Deque<Character> Spin(Deque<Character> deque, int
spinCount){
    if (spinCount == 1){
        deque.offerLast(deque.removeFirst());
    }else {
        deque.offerFirst(deque.removeLast());
    }
    return deque;
}

public static String decryption(Deque<Character> deque, String str){    //
Расшифровка
    String dec = "";
    for (int i = 0; i < str.length(); i++)
    {
        dec += getSymbol(deque, str.charAt(i), 1);
    }
    return dec;
}

public static char[] readCharFromFile(String file) throws
FileNotFoundException {
    Scanner sc = new Scanner(new File(file));
    String word = "";
    while (sc.hasNextLine()) {
        word = sc.next();
    }
    char[] DeqMass = new char[word.length()];
    for (int i = 0; i < word.length(); i++){
        DeqMass[i] = word.charAt(i);
    }
    return DeqMass;
}

```

```

    }

    /*
    Task 3
    */

    public static String ThirdTask(int count, int a, int b, int c, boolean
flag){
        if (flag && count == 2)
        {
            count--;
            return ThirdTask(count, a, c, b, false) + "\n" + a + " - " + c +
"\n" + ThirdTask(count, b, a, c, false);
        }
        if (count > 3)
        {
            count--;
            return ThirdTask(count, a, c, b, false) + "\n" + a + " - " + c +
"\n" + ThirdTask(count, b, a, c, false);
        }
        else
        {
            switch (count)
            {
                case 1:
                    return a + " - " + c;
                case 2:
                    count--;
                    return ThirdTask(count, a, b, c, false) + "\n" + a + " -
" + b + "\n" + ThirdTask(count, c, a, b, false);
                case 3:
                    count--;
                    return ThirdTask(count, a, b, c, false) + "\n" + a + " -
" + c + "\n" + ThirdTask(count, b, c, a, false);
                default:
                    return "";
            }
        }
    }

    public static boolean fourthTask(String file, char start, char end)
throws FileNotFoundException {
        String arr = "";
        Scanner in = new Scanner(new File(file));
        while(in.hasNext())
            arr += in.nextLine() + "\r\n";
        in.close();

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

        for (int i = 0; i < arr.length(); i++)
        {
            if (arr.charAt(i) == start)
            {
                stack.push('+');
            }
            if (arr.charAt(i) == end)
            {
                try
                {
                    stack.pop();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
    }

```

```

    }
    }
    return stack.isEmpty();
}

/*
    Task 5
*/

    public static boolean fifthTask(String file, char start, char end) throws
FileNotFoundException {
    String arr = "";
    Scanner in = new Scanner(new File(file));
    while(in.hasNext())
        arr += in.nextLine() + "\r\n";
    in.close();

    Deque<Character> deque = new ArrayDeque<>();

    for (int i = 0; i < arr.length(); i++)
    {
        if (arr.charAt(i) == start)
        {
            deque.push('+');
        }
        if (arr.charAt(i) == end)
        {
            try
            {
                deque.pop();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
    return deque.isEmpty();
}

/*
    Task 6
*/

    public static String sixTask(String file) throws FileNotFoundException {
    String arr = "";
    Scanner in = new Scanner(new File(file));
    while(in.hasNext())
        arr += in.nextLine() + "\r\n";
    in.close();

    StringBuilder sb = new StringBuilder();
    Stack<Character> stack = new Stack<>();
    Stack<Character> stack2 = new Stack<>();

    for (int i = 0; i < arr.length(); i++)
    {
        if (arr.charAt(i) >= 48 && arr.charAt(i) <= 57)    // 0 до 10
        {
            sb.append(arr.charAt(i));
        }
        else
        {
            if ((arr.charAt(i) >= 65 && arr.charAt(i) <= 90) ||
(arr.charAt(i) >= 97 && arr.charAt(i) <= 122))    // Буквы

```

```

        {
            stack.push(arr.charAt(i));
        }
        else
        {
            stack2.push(arr.charAt(i));
        }
    }
}
int count = sb.length();
while (!stack.isEmpty())
{
    sb.insert(count, stack.pop());
}
count = sb.length();
while (!stack2.isEmpty())
{
    sb.insert(count, stack2.pop());
}
return sb.toString();
}

/*
    Task 7
*/

public static String sevenTask(String file) throws FileNotFoundException
{
    String arr = "";
    Scanner in = new Scanner(new File(file));
    while(in.hasNext())
        arr += in.nextLine();
    in.close();

    int count = 0;
    String[] s = arr.split(" ");
    StringBuilder sb = new StringBuilder();
    Deque<Integer> deque = new ArrayDeque<>();
    for (String value : s) {
        int a = Integer.parseInt(value);
        if (a >= 0) {
            deque.addLast(a);
        } else {
            deque.addFirst(a);
            count++;
        }
    }
    for(int i = 0; i<count; i++)
    {
        sb.insert(0, deque.removeFirst() + " ");
    }
    while (!deque.isEmpty())
    {
        sb.append(deque.removeFirst()).append(" ");
    }
    return sb.toString().trim();
}

/*
    Task 8
*/

public static String eightTask(String file) throws FileNotFoundException

```



```

{
    Scanner sc = new Scanner(new File(file));
    List<String> lines = new ArrayList<>();
    while (sc.hasNextLine()) {
        lines.add(sc.nextLine());
    }

    String[] arr = lines.toArray(new String[0]);

    Stack<String> stack = new Stack<>();

    for(String i: arr){
        stack.push(i);
    }
    String result = "";
    while (!stack.isEmpty()){
        result +=stack.pop() + "\n";
    }
    return result;
}

/*
Task 9
*/

public static void nineTask(String file) throws FileNotFoundException{

    char[] text = readCharFromFile(file);

    Stack<Character> opstack = new Stack<>();
    Stack<Character> vstack = new Stack<>();

    int cur = 0;

    while(true) {
        boolean read = false;
        if (!opstack.isEmpty()) {
            char elem = opstack.pop();
            if (elem == 'N') {
                opstack.push(elem);
                if (vstack.isEmpty()) {
                    read = true;
                } else {
                    if (vstack.pop() == 'T') {
                        vstack.push('F');
                    } else {
                        vstack.push('T');
                    }
                }
                opstack.pop();
            }
        } else if (elem == 'A') {
            opstack.push(elem);
            if (vstack.size() < 2) {
                read = true;
            } else {
                char a = vstack.pop();
                char b = vstack.pop();
                if (a == b && b == 'T') {
                    vstack.push('T');
                } else {
                    vstack.push('F');
                }
                opstack.pop();
            }
        }
    }
}

```

```

        } else if (elem == 'O') {
            opstack.push(elem);
            if (vstack.size() < 2) {
                read = true;
            } else {
                char a = vstack.pop();
                char b = vstack.pop();
                if (a == 'T' || b == 'T') {
                    vstack.push('T');
                } else {
                    vstack.push('F');
                }
                opstack.pop();
            }
        } else if (elem == 'X') {
            opstack.push(elem);
            if (vstack.size() < 2) {
                read = true;
            } else {
                char a = vstack.pop();
                char b = vstack.pop();
                if (a != b) {
                    vstack.push('T');
                } else {
                    vstack.push('F');
                }
                opstack.pop();
            }
        } else if (elem == '(') {
            opstack.push(elem);
            read = true;
        } else if (elem == ')') {
            opstack.push(elem);
            opstack.pop();
            opstack.pop();
        }
    } else {
        read = true;
    }
}

if (read) {
    char i = text[cur];
    if ("FT".contains(Character.toString(i))) {
        vstack.push(i);
    }
    else if ("AXON".contains(Character.toString(i))) {
        opstack.push(i);
    }
    cur++;
}

if (cur == text.length && opstack.size() == 0) {
    break;
}

}

while (!vstack.isEmpty()) {
    System.out.println(vstack.pop());
}

}

/*
Task 10
*/

public static int tenTask(String file) throws FileNotFoundException {

```

```

String str = "";
Scanner in = new Scanner(new File(file));
while(in.hasNext())
    str += in.nextLine();
in.close();
return MinMax(str);    // возвращает итоговый результат
}

public static int MinMax(String str){
    int a, b = 0;
    int minMax = -1;    // равно 1, если ищем
минимум; равно 0, если ищем максимум

    int f = str.indexOf("(");    // Индекс первой
встречающейся в строке скобка
    int l = str.length()-1;    // Индекс скобки, закрывающей
предыдущую ')'
    if (str.charAt(f-1) == 'M'){    // Если M(3,5)
        minMax = 0;
    }else if(str.charAt(f-1) == 'N'){    // Если N(3,5)
        minMax = 1;
    }
    if(str.charAt(f+1) == 'M' || str.charAt(f+1) == 'N') {    //
Если M(M(2,4),5) или M(N(2,4),5)
        int endSkobka = checkSkobka(str.substring(f+2, l - 2));    //
При помощи стека (и 4-ой задачи лабораторной работы) определяем индекс
закрывающей скобки
        a = MinMax(str.substring(f + 1, f + 3 + endSkobka));    //
находим минимум или максимум в M(2,4) или N(2,4)
        String firstStr = Integer.toString(a);
        str = str.replace(str.substring(f + 1, f + 3 + endSkobka),
firstStr);    // Заменяем получившийся минмакс в исходной строке: M(4,5) или
M(2,5)
        l = str.length()-1;
// переопределяем индекс закрывающей скобки, т.к. удалили часть строки
    }else {
        a = Integer.parseInt(String.valueOf(str.charAt(f + 1)));    //
присваиваем первое число
    }
    if(str.charAt(f+3) == 'M' || str.charAt(f+3) == 'N') {    // Если
M(5,M(2,4)) или M(5,N(2,4))
        b = MinMax(str.substring(f + 3, l));    //
находим минимум или максимум в M(2,4) или N(2,4)
    }else{
        b = Integer.parseInt(String.valueOf(str.charAt(l - 1)));    //
присваиваем второе число
    }
    if (minMax == 0)    // если ищем максимум, то
        return Math.max(a, b);
    else
        return Math.min(a, b);
}

public static int checkSkobka(String arr){
    char start = '(';
    char end = ')';
    Stack<Character> stack = new Stack<Character>();

    for (int i = 0; i < arr.length(); i++)
    {
        if (arr.charAt(i) == start)    // если встретили "(", то добавляем
в стек "+"
        {
            stack.push('+');

```

```

        }
        if(arr.charAt(i) == end)           // если встретили ")", то извлекаем
из стека "+"
        {
            try
            {
                stack.pop();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        if (stack.isEmpty())               // проверяем, не является ли
текущий символ закрывающим нужную нам скобку
            return i;
    }
    return 0;
}

/*
Task 11
*/

public static Boolean elevenTask(String file) throws
FileNotFoundException{

    String str = "";
    Scanner in = new Scanner(new File(file));
    while(in.hasNext())
        str += in.nextLine();
    in.close();

    if (!BracketsValid(str))
        return false;

    str = str.replace('(', ' ');
    str = str.replace(')', ' ');
    str = str.replace(" ", "");

    int i = 0;
    while (i < str.length()){
        if("xyz".contains(Character.toString(str.charAt(i))) &&
i!=str.length()-1) {
            if (!("+-".contains(Character.toString(str.charAt(i + 1)))))
                return false;
            else
                i++;
        }else
            return (i == str.length() - 1) &&
"xyz".contains(Character.toString(str.charAt(i)));
        i++;
    }
    return false;
}

public static Boolean BracketsValid(String arr){
    char start = '(';
    char end = ')';
    Stack<Character> stack = new Stack<Character>();

    for (int i = 0; i<arr.length(); i++)
    {
        if (arr.charAt(i) == start)           // если встретили "(", то добавляем
в стек "+"
        {

```

```
        if ("+-".contains(Character.toString(arr.charAt(i+1))))
            return false;
        else
            stack.push('+');
    }
    if (arr.charAt(i) == end)           // если встретили ")", то извлекаем
из стека "+"
    {
        if (stack.isEmpty())
            return false;
        else if ("+-".contains(Character.toString(arr.charAt(i-1))))
            return false;
        else
            stack.pop();
    }
}
return true;
}
```

Вывод: в ходе выполнения данной работы мною были получены знания о разнице и структуре типов данных таких как стэк и дэка, реализованы эти структуры, выполнены задачи на работы со структурами.