

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И
МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
государственное бюджетное образовательное учреждение высшего
образования ордена Трудового Красного Знамени
«Московский технический университет связи и информатики»

Кафедра Математической кибернетики и информационных технологий

Отчет по лабораторной работе №2
Методы поиска.
по дисциплине «Структуры и алгоритмы обработки данных»

Выполнил: студент группы БФИ1902

Шацкий Е.И

Проверил:

Мкртчян Г. М

Москва 2021 г.

Реализовать методы поиска в соответствии с заданием. Организовать генерацию начального набора случайных данных.

Задание №1

Реализовать методы поиска:

- 1.1 Бинарный поиск
- 1.2 Бинарное дерево
- 1.3 Фибоначчиев
- 1.4 Интерполяционный

Результат выполнения задания №1.1 представлен на рисунке 1

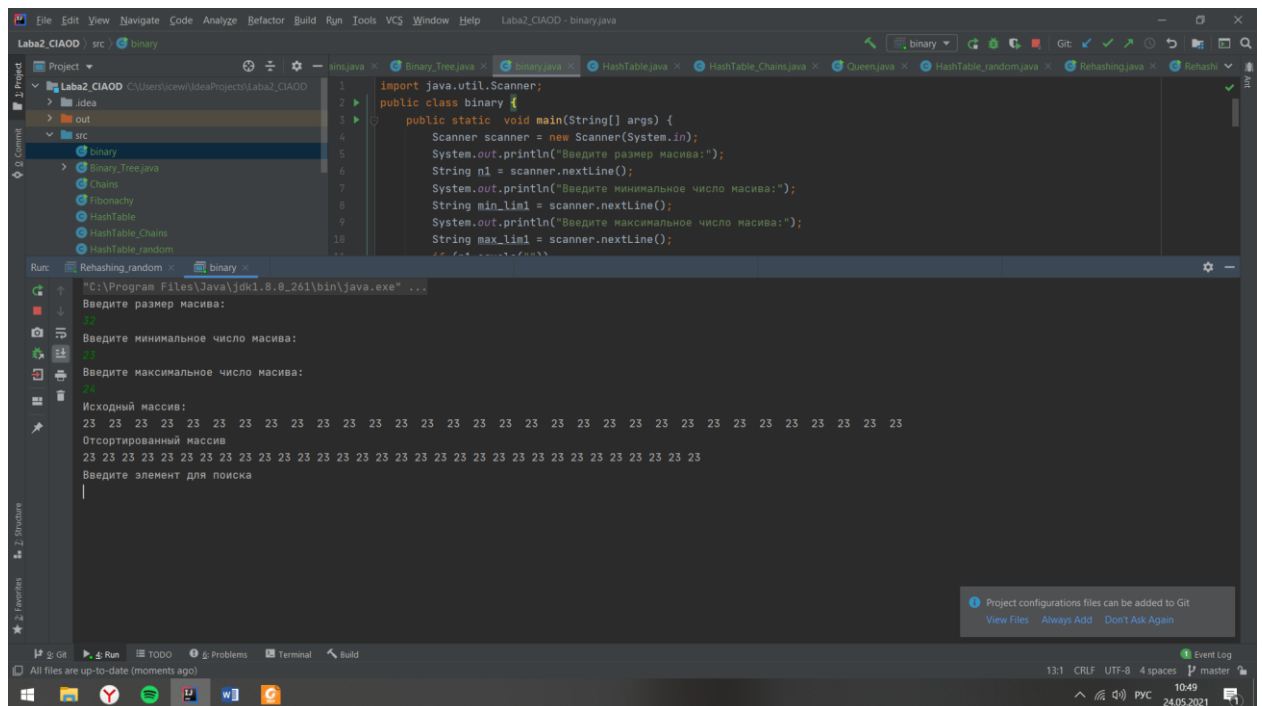


Рисунок 1 – бинарный поиск

Результат выполнения задания №1.2 представлен на рисунке 2

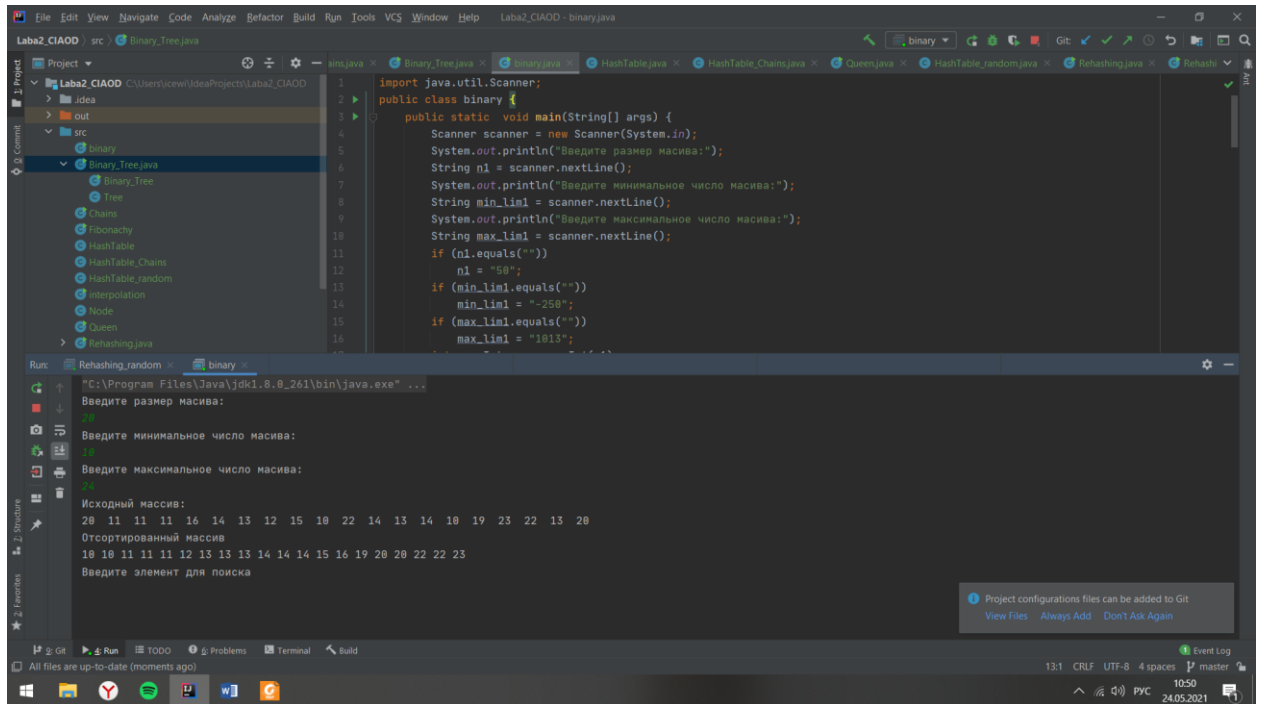


Рисунок 2 - Бинарное дерево

Результат выполнения задания №1.3 представлен на рисунке 3

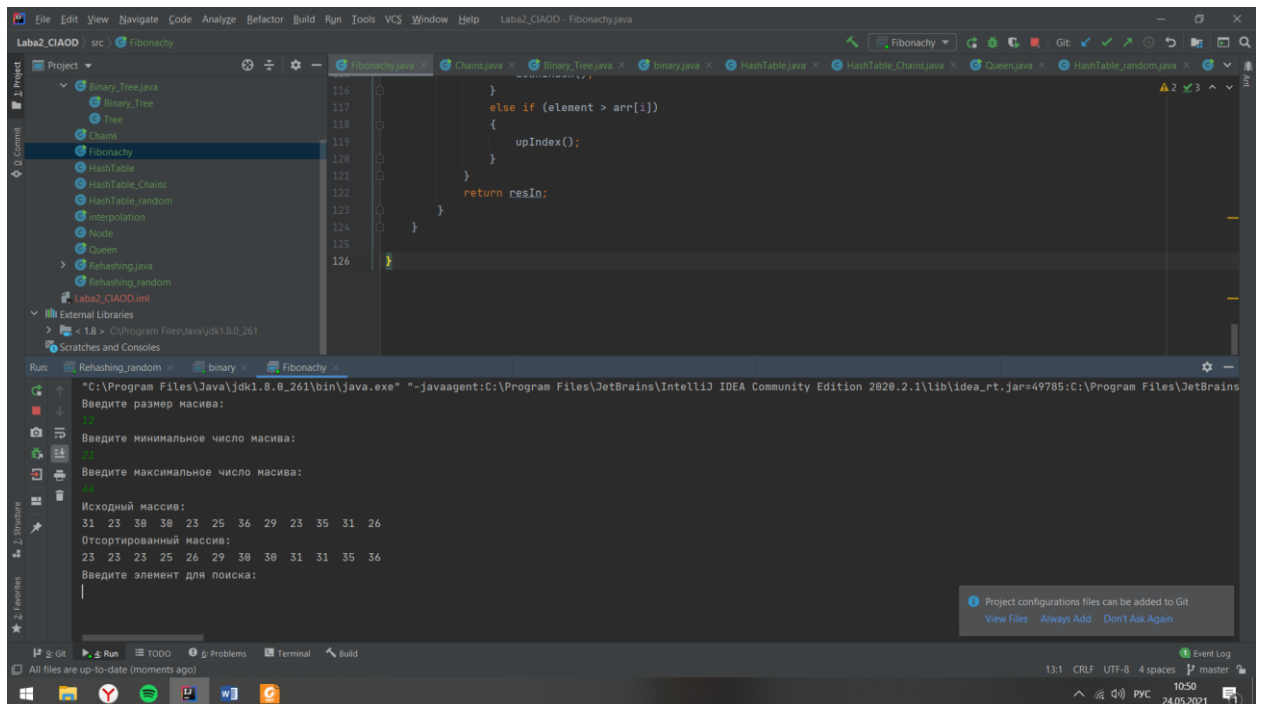


Рисунок 3 – Фибоначчиев

Результат выполнения задания №1.4 представлен на рисунке 4

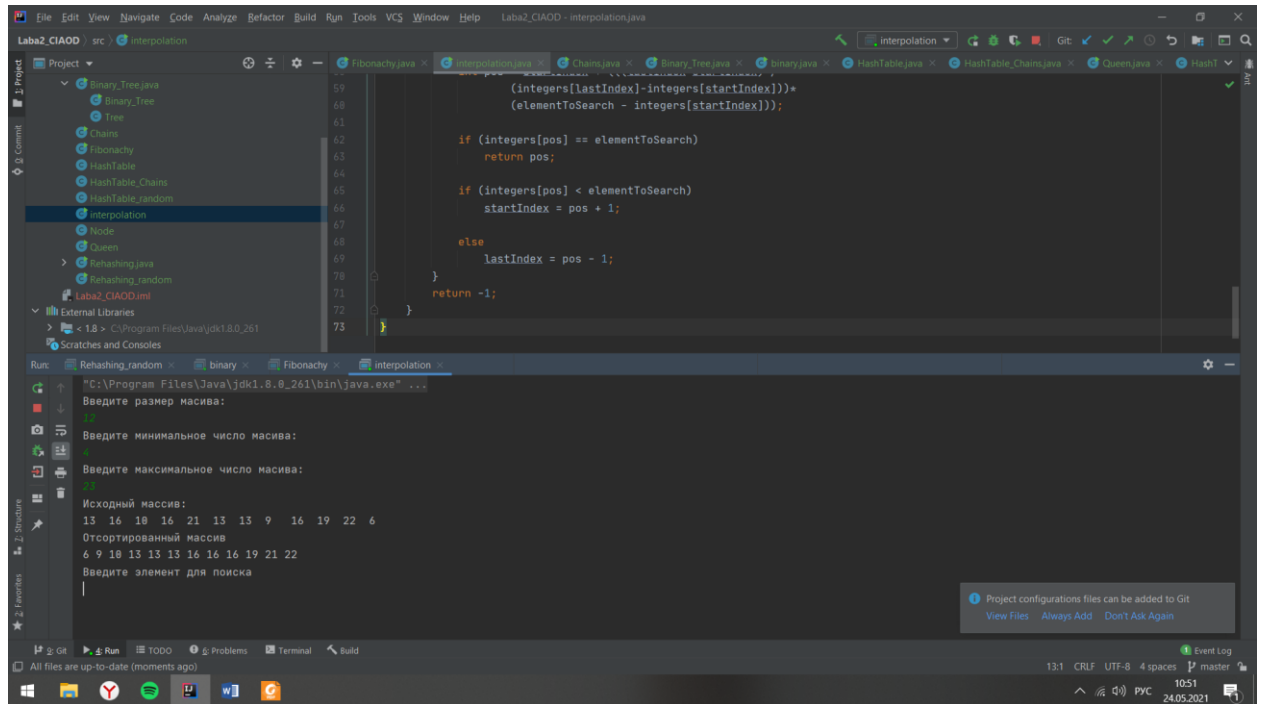


Рисунок 4 – Интерполяционный

Задание №2:

Реализовать методы поиска:

2.1 Простое рехэширование

2.2 Рехэширование с помощью псевдослучайных чисел

2.3 Метод цепочек

Результат выполнения задания №2.1 представлен на рисунке 5

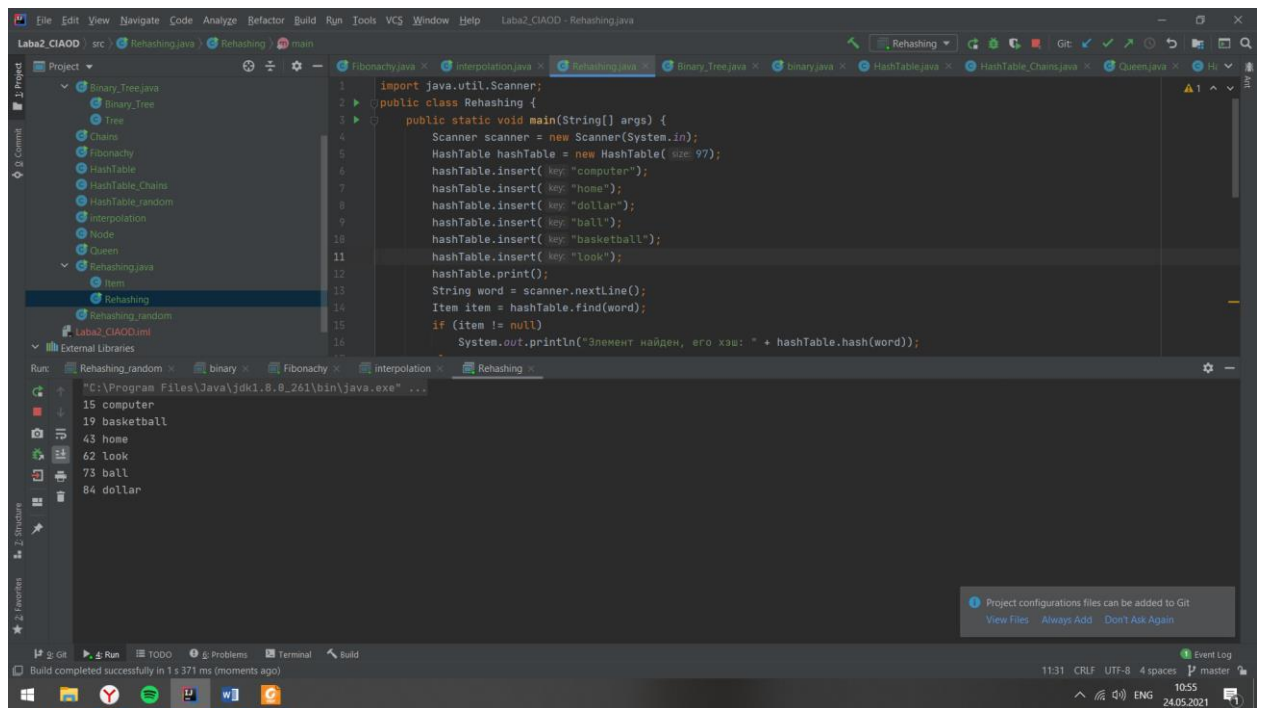


Рисунок 5 - Результат выполнения задания №2.1

Результат выполнения задания №2.2 представлен на рисунке 6

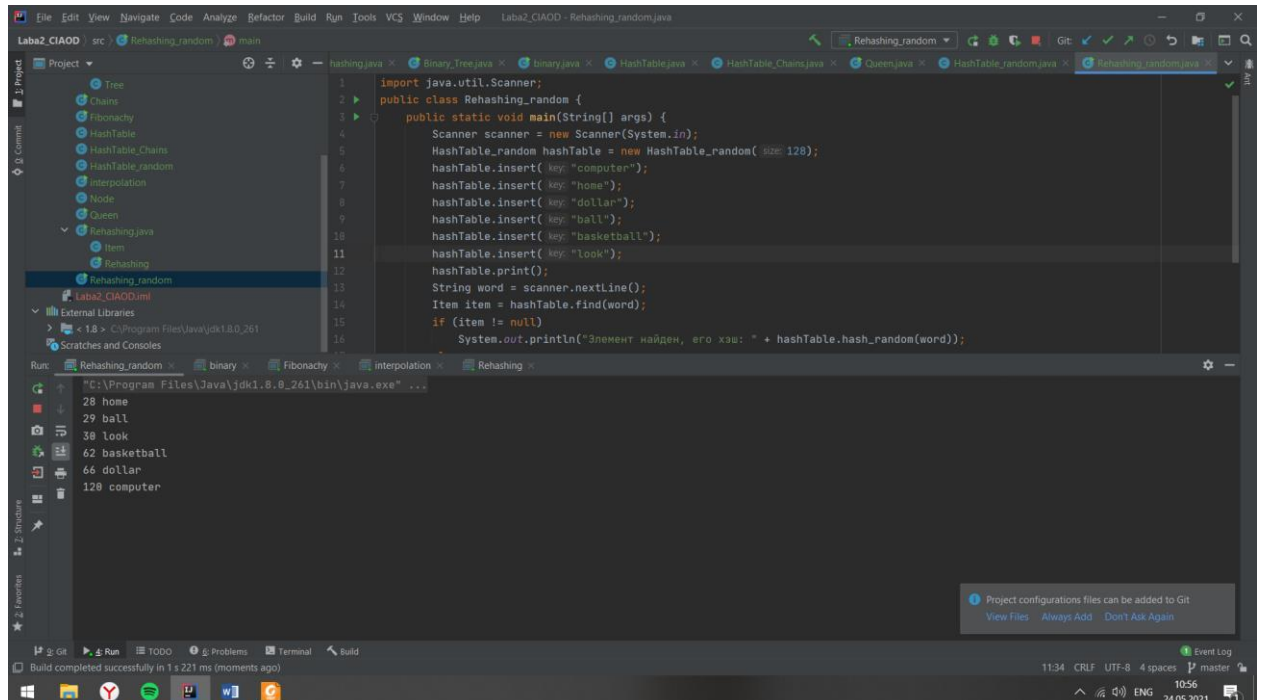


Рисунок 6 - Результат выполнения задания №2.2

Листинг программы:

```
import java.util.Scanner;
public class binary {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Введите размер массива:");
        String n1 = scanner.nextLine();
        System.out.println("Введите минимальное число массива:");
        String min_lim1 = scanner.nextLine();
        System.out.println("Введите максимальное число массива:");
        String max_lim1 = scanner.nextLine();
        if (n1.equals(""))
            n1 = "50";
        if (min_lim1.equals(""))
            min_lim1 = "-250";
        if (max_lim1.equals(""))
            max_lim1 = "1013";
        int n = Integer.parseInt(n1);
        int min_lim = Integer.parseInt(min_lim1);
        int max_lim = Integer.parseInt(max_lim1);
        int[] arr = new int[n];
        System.out.println("Исходный массив:");
        for (int i = 0; i < n; i++) {
            arr[i] = (int) ((Math.random() * (max_lim - min_lim)) + min_lim);
            System.out.print(arr[i] + "\t");
        }
        System.out.println();
        boolean needIteration = true;
        while (needIteration) {
            needIteration = false;
            for (int i = 1; i < n; i++) {
                if (arr[i] < arr[i - 1]) {
                    int tmp = arr[i];
                    arr[i] = arr[i - 1];
                    arr[i - 1] = tmp;
                    needIteration = true;
                }
            }
        }
        System.out.println("Отсортированный массив");
        for (int i=0;i<n;i++){
            System.out.print(arr[i]+" ");
        }
        System.out.println();
        System.out.println("Введите элемент для поиска");
        int item = scanner.nextInt();
        binarySearch(arr, 0, n-1, item);
    }
    public static void binarySearch(int[] array, int first, int last, int
item) {
        int position;
        int comparisonCount = 1;    // для подсчета количества сравнений

        // для начала найдем индекс среднего элемента массива
        position = (first + last) / 2;

        while ((array[position] != item) && (first <= last)) {
            comparisonCount++;
            if (array[position] > item) { // если число заданного для поиска
меньше текущего
                last = position - 1; // уменьшаем позицию на 1.
            } else {
```

```

        first = position + 1;    // иначе увеличиваем на 1
    }
    position = (first + last) / 2;
}
if (first <= last) {
    System.out.println(item + " является " + ++position + " элементом
в массиве");
    System.out.println("Метод бинарного поиска нашел число после " +
comparisonCount +
        " сравнений");
} else {
    System.out.println("Элемент не найден в массиве. Метод бинарного
поиска закончил работу после "
        + comparisonCount + " сравнений");
}
}
}
}

```

```

import java.util.Scanner;
import java.util.Stack;

public class Binary_Tree {
    public static void main(String[] args) {
        Tree tree = new Tree();
        Scanner scanner = new Scanner(System.in);
        System.out.println("Введите размер массива:");
        String n1 = scanner.nextLine();
        System.out.println("Введите минимальное число массива:");
        String min_lim1 = scanner.nextLine();
        System.out.println("Введите максимальное число массива:");
        String max_lim1 = scanner.nextLine();
        if (n1.equals(""))
            n1 = "50";
        if (min_lim1.equals(""))
            min_lim1 = "-250";
        if (max_lim1.equals(""))
            max_lim1 = "1013";
        int n = Integer.parseInt(n1);
        int min_lim = Integer.parseInt(min_lim1);
        int max_lim = Integer.parseInt(max_lim1);
        int[] arr = new int[n];

        System.out.println("Исходный массив:");
        for (int i = 0; i < n; i++) {
            arr[i] = (int) ((Math.random() * (max_lim - min_lim)) + min_lim);
            System.out.print(arr[i] + "\t");
        }
        System.out.println();
        System.out.println("Введите элемент для поиска:");
        int item = scanner.nextInt();
        System.out.println();
        for (int i=0;i<n;i++){
            tree.insertNode(arr[i]);
        }
        // отображение дерева:
        tree.printTree();
        // находим узел по значению и выводим его в консоли
        Node foundNode = tree.findNodeByValue(item);
        if (foundNode==null){
            System.out.println("Элемента нет в дереве");
        } else {
            foundNode.printNode();
        }
    }
}

```

```

    }
}
class Tree {
    private Node rootNode; // корневой узел

    public Tree() { // Пустое дерево
        rootNode = null;
    }

    public Node findNodeByValue(int value) { // поиск узла по значению
        Node currentNode = rootNode; // начинаем поиск с корневого узла
        while (currentNode.getValue() != value) { // поиск пока не будет
найден элемент или не будут перебраны все
            if (value < currentNode.getValue()) { // движение влево?
                currentNode = currentNode.getLeftChild();
            } else { // движение вправо
                currentNode = currentNode.getRightChild();
            }
            if (currentNode == null) { // если потомка нет,
                return null; // возвращаем null
            }
        }
        return currentNode; // возвращаем найденный элемент
    }

    public void insertNode(int value) { // метод вставки нового элемента
        Node newNode = new Node(); // создание нового узла
        newNode.setValue(value); // вставка данных
        if (rootNode == null) { // если корневой узел не существует
            rootNode = newNode; // то новый элемент и есть корневой узел
        }
        else { // корневой узел занят
            Node currentNode = rootNode; // начинаем с корневого узла
            Node parentNode;
            while (true) // мы имеем внутренний выход из цикла
            {
                parentNode = currentNode;
                if (value == currentNode.getValue()) { // если такой элемент
в дереве уже есть, не сохраняем его
                    return; // просто выходим из метода
                }
                else if (value < currentNode.getValue()) { // движение
влево?
                    currentNode = currentNode.getLeftChild();
                    if (currentNode == null) { // если был достигнут конец
цепочки,
                        parentNode.setLeftChild(newNode); // то вставить
слева и выйти из метода
                        return;
                    }
                }
                else { // Или направо?
                    currentNode = currentNode.getRightChild();
                    if (currentNode == null) { // если был достигнут конец
цепочки,
                        parentNode.setRightChild(newNode); //то вставить
справа
                        return; // и ВЫЙТИ
                    }
                }
            }
        }
    }
}

```



```

public void printTree() { // метод для вывода дерева в консоль
    Stack globalStack = new Stack(); // общий стек для значений дерева
    globalStack.push(rootNode);
    int gaps = 32; // начальное значение расстояния между элементами
    boolean isRowEmpty = false;
    String separator = "-----
-----";

    System.out.println(separator); // черта для указания начала нового
дерева
    while (isRowEmpty == false) {
        Stack localStack = new Stack(); // локальный стек для задания
потомков элемента
        isRowEmpty = true;

        for (int j = 0; j < gaps; j++)
            System.out.print(' ');
        while (globalStack.isEmpty() == false) { // покуда в общем стеке
есть элементы
            Node temp = (Node) globalStack.pop(); // берем следующий, при
этом удаляя его из стека
            if (temp != null) {
                System.out.print(temp.getValue()); // выводим его
значение в консоли
                localStack.push(temp.getLeftChild()); // сохраняем в
локальный стек, наследники текущего элемента
                localStack.push(temp.getRightChild());
                if (temp.getLeftChild() != null ||
                    temp.getRightChild() != null)
                    isRowEmpty = false;
            }
            else {
                System.out.print("__"); // - если элемент пустой
                localStack.push(null);
                localStack.push(null);
            }
            for (int j = 0; j < gaps * 2 - 2; j++)
                System.out.print(' ');
        }
        System.out.println();
        gaps /= 2; // при переходе на следующий уровень расстояние между
элементами каждый раз уменьшается
        while (localStack.isEmpty() == false)
            globalStack.push(localStack.pop()); // перемещаем все
элементы из локального стека в глобальный
    }
    System.out.println(separator); // подводим черту
}
}

```

```

import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class Chains
{
    public static void main( String args[] )
    {
        HashTable_Chains table=new HashTable_Chains(7);
    }
}

```

```

        try
        {
            File file = new
File("C:\\Users\\icewi\\IdeaProjects\\Laba2_CIAOD\\input.txt");
            Scanner scanner = new Scanner(file);
            while (scanner.hasNext())
            {
                table.addElement(scanner.next());
            }
            scanner.close();
        }

        catch (FileNotFoundException e)
        {
            e.printStackTrace();
        }

        table.printHashTable();
        Scanner sc = new Scanner(System.in);
        System.out.print("Введите слово для поиска: ");
        String answer = sc.nextLine();
        if(table.findElement(answer))
        {
            System.out.print("Такое слово есть.");
        }
        else
            System.out.print("Такого слова нету.");
    }
}

```

```

import java.util.Scanner;

public class Fibonachy {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Введите размер массива:");
        String n1 = scanner.nextLine();
        System.out.println("Введите минимальное число массива:");
        String min_lim1 = scanner.nextLine();
        System.out.println("Введите максимальное число массива:");
        String max_lim1 = scanner.nextLine();
        if (n1.equals(""))
            n1 = "50";
        if (min_lim1.equals(""))
            min_lim1 = "-250";
        if (max_lim1.equals(""))
            max_lim1 = "1013";
        int n = Integer.parseInt(n1);
        int min_lim = Integer.parseInt(min_lim1);
        int max_lim = Integer.parseInt(max_lim1);
        int[] arr = new int[n];
        System.out.println("Исходный массив:");
        for (int i = 0; i < n; i++) {
            arr[i] = (int) ((Math.random() * (max_lim - min_lim)) + min_lim);
            System.out.print(arr[i] + "\t");
        }
        System.out.println();
        boolean needIteration = true;
        while (needIteration) {
            needIteration = false;

```

```

        for (int i = 1; i < n; i++) {
            if (arr[i] < arr[i - 1]) {
                int tmp = arr[i];
                arr[i] = arr[i - 1];
                arr[i - 1] = tmp;
                needIteration = true;
            }
        }
    }
    System.out.println("Отсортированный массив:");
    for (int i = 0; i < n; i++) {
        System.out.print(arr[i] + "\t");
    }
    System.out.println();
    System.out.println("Введите элемент для поиска:");
    int item = scanner.nextInt();
    System.out.println();
    FibonacciSearch F = new FibonacciSearch();
    int index = F.search(arr, item);
    System.out.println("Индекс найденного элемента:");
    System.out.println(index);
}

public static class FibonacciSearch {
    private int i;
    private int p;
    private int q;
    private boolean stop = false;
    private void init(int[] arr) {
        stop = false;
        int k = 0;
        int n = arr.length;
        for (; getFibonacciNumber(k+1) < n+1; ) {
            k +=1;
        }
        int m = getFibonacciNumber(k+1) - (n+1);
        i = getFibonacciNumber(k) - m;
        p = getFibonacciNumber(k-1);
        q = getFibonacciNumber(k-2);
    }

    public int getFibonacciNumber(int k) {
        int firstNumber = 0;
        int secondNumber = 1;
        for (int i = 0; i < k; i++) {
            int temp = secondNumber;
            secondNumber += firstNumber;
            firstNumber = temp;
        }
        return firstNumber;
    }

    private void upIndex() {
        if (p==1)
            stop = true;
        i = i + q;
        p = p - q;
        q = q - p;
    }

    private void downIndex() {
        if (q==0)
            stop = true;
        i = i - q;
        int temp = q;

```

```

        q = p - q;
        p = temp;
    }

    public int search(int[] arr,int element){
        init(arr);
        int n = arr.length;
        int resIn = -1;
        for (; !stop;){
            if (i < 0){
                upIndex();
            }
            else if (i>=n){
                downIndex();
            }
            else if (arr[i]==element){
                resIn = i;
                break;
            }
            else if (element < arr[i]){
                downIndex();
            }
            else if (element > arr[i])
            {
                upIndex();
            }
        }
        return resIn;
    }
}

```

```

public class HashTable {

    //массив для хранения элементов
    private Item[] table;
    //количество элементов в таблице
    private int count;
    //размер таблицы
    private int size;

    public HashTable(int size) {
        this.size = size;
        table = new Item[size];
    }
    public int hash(String key)
    {
        int hash = 0;

        for(int i = 0; i < key.length(); i++)
            hash = (31 * hash + key.charAt(i)) % size;

        return hash;
    }

    public void insert(String key) {
        Item item = new Item(key);
        int hash = hash(key);
        while (table[hash] != null) {
            hash++;
            hash %= size;
        }
    }
}

```

```

    }
    table[hash] = item;
}
public void print()
{
    for(int i = 0; i < size; i++)
        if(table[i] != null)
            System.out.println(i + " " + table[i].getKey());
}
public Item find(String key)
{
    int hash = hash(key);
    while(table[hash] != null)
    {
        if(table[hash].getKey().equals(key))
            return table[hash];
        hash++;
        hash = hash % size;
    }

    return null;
}

```

```

import java.util.ArrayList;

public class HashTable_Chains
{
    private int size;
    private ArrayList<String>[] array;

    public HashTable_Chains(int number)
    {
        size=number;
        array= new ArrayList[size];
        for (int i=0; i<size; ++i)
            array[i]=new ArrayList<String>();
    }

    private int hashFunc(String str)
    {
        int result=0;
        for( int i=0; i<str.length(); i++)
            result+=(int)str.charAt(i);

        return result%size;
    }

    public void addElement(String str)
    {
        array[hashFunc(str)].add(str);
    }

    public boolean findElement(String str)
    {
        for (int j = 0; j < array[hashFunc(str)].size(); j++)
            if((array[hashFunc(str)].get(j)).equals(str))
                return true;
        return false;
    }

    public void printHashTable()
    {

```

```

        System.out.println("Ключ:  значение ");
        for (int i=0; i<size; ++i)
        {
            System.out.print(i + ":  ");
            for (int j = 0; j < array[i].size(); j++)
                System.out.print(array[i].get(j) + " ");
            System.out.println();
        }
    }
}

```

```

public class HashTable_random {

    //массив для хранения элементов
    private Item[] table;
    //количество элементов в таблице
    private int count;
    //размер таблицы
    private int size;

    public HashTable_random(int size) {
        this.size = size;
        table = new Item[size];
    }

    public int hash_random(String key)
    {
        double hash=0;
        double R = 1;

        for(int i = 0; i < key.length(); i++)
            R=5*R;
        R=R%(4*size);
        hash=Math.floor(R/4);

        return (int)hash;
    }

    public void insert(String key) {
        Item item = new Item(key);
        int hash = hash_random(key);
        while (table[hash] != null) {
            hash++;
            hash %= (4*size);
        }
        table[hash] = item;
    }

    public void print()
    {
        for(int i = 0; i < size; i++)
            if(table[i] != null)
                System.out.println(i + " " + table[i].getKey());
    }

    public Item find(String key)
    {
        int hash = hash_random(key);
        while(table[hash] != null)
        {
            if(table[hash].getKey().equals(key))
                return table[hash];
            hash++;
            hash = hash % (4*size);
        }
    }
}

```

```

        return null;
    }
}

```

```

import java.util.Scanner;

public class interpolation {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Введите размер массива:");
        String n1 = scanner.nextLine();
        System.out.println("Введите минимальное число массива:");
        String min_lim1 = scanner.nextLine();
        System.out.println("Введите максимальное число массива:");
        String max_lim1 = scanner.nextLine();
        if (n1.equals(""))
            n1 = "50";
        if (min_lim1.equals(""))
            min_lim1 = "-250";
        if (max_lim1.equals(""))
            max_lim1 = "1013";
        int n = Integer.parseInt(n1);
        int min_lim = Integer.parseInt(min_lim1);
        int max_lim = Integer.parseInt(max_lim1);
        int[] arr = new int[n];

        System.out.println("Исходный массив:");
        for (int i = 0; i < n; i++) {
            arr[i] = (int) (Math.random() * (max_lim - min_lim) + min_lim);
            System.out.print(arr[i] + "\t");
        }
        boolean needIteration = true;
        while (needIteration) {
            needIteration = false;
            for (int i = 1; i < n; i++) {
                if (arr[i] < arr[i - 1]) {
                    int tmp = arr[i];
                    arr[i] = arr[i - 1];
                    arr[i - 1] = tmp;
                    needIteration = true;
                }
            }
        }
        System.out.println();
        System.out.println("Отсортированный массив");
        for (int i=0;i<n;i++){
            System.out.print(arr[i]+" ");
        }
        System.out.println();
        System.out.println("Введите элемент для поиска");
        int item = scanner.nextInt();
        System.out.println("Индекс найденного
элемента:"+interpolationSearch(arr,item));
    }
    public static int interpolationSearch(int[] integers, int
elementToSearch) {

        int startIndex = 0;
        int lastIndex = (integers.length - 1);

        while ((startIndex <= lastIndex) && (elementToSearch >=

```

```

integers[startIndex]) &&
    (elementToSearch <= integers[lastIndex])) {
    // используем формулу интерполяции для поиска возможной лучшей
    позиции для существующего элемента
    int pos = startIndex + (((lastIndex-startIndex) /
        (integers[lastIndex]-integers[startIndex])) *
        (elementToSearch - integers[startIndex]));

    if (integers[pos] == elementToSearch)
        return pos;

    if (integers[pos] < elementToSearch)
        startIndex = pos + 1;

    else
        lastIndex = pos - 1;
}
return -1;
}
}

```

```

public class Node {
    private int value; // ключ узла
    private Node leftChild; // Левый узел потомок
    private Node rightChild; // Правый узел потомок

    public void printNode() { // Вывод значения узла в консоль
        System.out.println(" Выбранный узел имеет значение : " + value);
    }

    public int getValue() {
        return this.value;
    }

    public void setValue(final int value) {
        this.value = value;
    }

    public Node getLeftChild() {
        return this.leftChild;
    }

    public void setLeftChild(final Node leftChild) {
        this.leftChild = leftChild;
    }

    public Node getRightChild() {
        return this.rightChild;
    }

    public void setRightChild(final Node rightChild) {
        this.rightChild = rightChild;
    }

    @Override
    public String toString() {
        return "Node{" +
            "value=" + value +
            ", leftChild=" + leftChild +
            ", rightChild=" + rightChild +
            '}';
    }
}

```



```
}  
}
```

```
public class Queen {  
    /**  
     * размерность доски  
     */  
    /**  
     * хранит расстановку ферзей. каждый ферзь находится на отдельной линии,  
на  
     * одной линии находится не могут так как бьют друг друга.  
     */  
    private int[] state;  
    /**  
     * Порядковый номер комбинации  
     */  
    private int index = 1;  
  
    /**  
     * n - размерность доски и количество ферзей  
     */  
    public Queen(int n) {  
  
        state = new int[n];  
  
        for (int i = 0; i < state.length; i++) {  
            state[i] = 0;  
        }  
  
    }  
  
    /**  
     * генерирует следующую комбинацию (расстановку фигур)  
     */  
    public boolean next() {  
        index++;  
        return move(8 - 1);  
    }  
  
    /**  
     * Двигает фигуру в указанной линии на одну клетку вправо и возвращает  
true.  
     * Если фигура находится в крайнем положении, то фигура устанавливается в  
     * первое положение и двигается фигура находящаяся на линии выше и так  
далее.  
     * Если линий выше не осталось возвращает false.  
     */  
    private boolean move(int index) {  
        if (state[index] < 8 - 1) {  
            state[index]++;  
            return true;  
        }  
  
        state[index] = 0;  
        if (index == 0) {  
            return false;  
        } else {  
            return move(index - 1);  
        }  
    }  
  
    /**  
     * Возвращает порядковый номер комбинации, которая в данный момент
```

```

        * установлена.
        */
    public int getIndex() {
        return index;
    }

    //проверяем бьет ли наша королева другую фигуру если да то возвращаем
    фолс
    public boolean isPeace() {
        for (int i = 0; i < state.length; i++) {
            for (int j = i + 1; j < state.length; j++) {
                // бьет ли по вертикали
                if (state[i] == state[j]) {
                    return false;
                }
                // бьет ли по диагонали
                if (Math.abs(i - j) == Math.abs(state[i] - state[j])) {
                    return false;
                }
            }
        }

        return true;
    }

    /*
    * Выводит доску с фигурами.
    */
    public void printState() {
        for (int i = 0; i < state.length; i++) {
            int position = state[i];
            for (int j = 0; j < 8 ; j++) {
                System.out.print(j == position ? 'X' : '_');
            }
            System.out.println();
        }
    }

    public static void main(String[] args) {
        Queen c = new Queen(8);
        int counter = 0;
        do {
            if (c.isPeace()) {
                counter ++;
                c.printState();
                System.out.println("-----");
            }
        } while (c.next());

        System.out.println("Итого: " + counter);
    }

```

```

import java.util.Scanner;
public class Rehashing {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        HashTable hashTable = new HashTable(97);
        hashTable.insert("computer");
        hashTable.insert("home");
        hashTable.insert("dollar");
        hashTable.insert("ball");
        hashTable.insert("basketball");
        hashTable.insert("look");
    }
}

```

```

        hashTable.print();
        String word = scanner.nextLine();
        Item item = hashTable.find(word);
        if (item != null)
            System.out.println("Элемент найден, его хэш: " +
hashTable.hash(word));
        else
            System.out.println("Элемент не найден!");
    }
}
class Item{

    private String key;

    public Item(String key)
    {
        this.key = key;
    }

    public String getKey() {
        return key;
    }

    public void setKey(String key) {
        this.key = key;
    }
}

```

```

import java.util.Scanner;
public class Rehashing_random {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        HashTable_random hashTable = new HashTable_random(128);
        hashTable.insert("computer");
        hashTable.insert("home");
        hashTable.insert("dollar");
        hashTable.insert("ball");
        hashTable.insert("basketball");
        hashTable.insert("look");
        hashTable.print();
        String word = scanner.nextLine();
        Item item = hashTable.find(word);
        if (item != null)
            System.out.println("Элемент найден, его хэш: " +
hashTable.hash_random(word));
        else
            System.out.println("Элемент не найден!");
    }
}

```

Вывод:

В данной лабораторной работе были изучены основные методы поиска и выполнена их программная реализация на языке Java.