**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

**ЛАБОРАТОРНАЯ РАБОТА №3**

по дисциплине: «Технологии разработки программного обеспечения»

на тему: «Моделирование и алгоритмизация как средства проектирования программного обеспечения»

Выполнил: студент гр. ИТП-21

Царьков В.П.

Принял: преподаватель-стажер

Малиновский И.Л.

Гомель 2023

**Цель работы:**

1. Описать блок-схему алгоритма согласно варианту (таблица 3).

2. На основе описанной блок-схемы реализовать алгоритм средствами языка *Java*.

3. Протестировать разработанный алгоритм при помощи модульных тестов.

4. Создать консольное приложение для демонстрации работы алгоритма.

5. Составить отчет о проделанной работе.

**Условие:**

Вариант 5 – Двоичный поиск.

**Решение:**

Двоичный поиск (или бинарный поиск) - это алгоритм поиска значения в упорядоченном массиве данных. Суть алгоритма заключается в том, что мы сравниваем значение, которое мы ищем, с значением, находящимся в середине массива. Если значение совпадает, то мы заканчиваем поиск. Если значение, которое мы ищем, меньше, чем значение, находящееся в середине массива, то мы продолжаем поиск только в левой половине массива (то есть в той, которая находится перед серединой). Если значение, которое мы ищем, больше, чем значение, находящееся в середине массива, то мы продолжаем поиск только в правой половине массива (то есть в той, которая находится после середины). Мы продолжаем делить массив на половины до тех пор, пока не найдем значение, которое мы ищем, или пока не закончится массив. Листинг класса *BinarySearch* указан в приложении А. Блок схема данного алгоритма указана на рисунке 1.

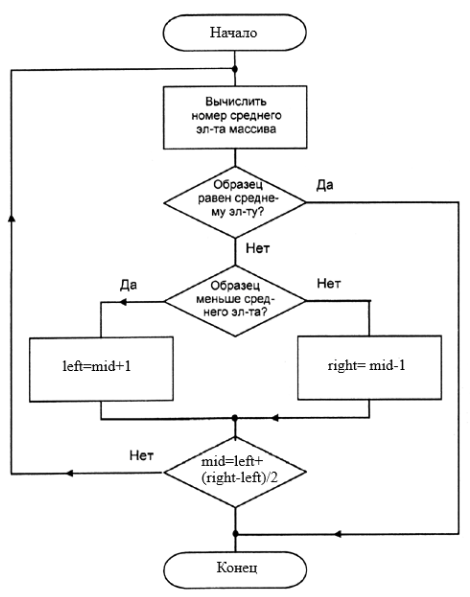


Рисунок 1 – Блок-схема алгоритма бинарного поиска

Пример работы двоичного поиска указан на рисунке 2

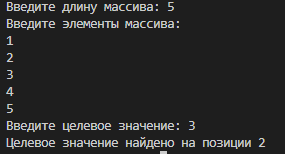


Рисунок 2 – Пример работы программы

В задание 3 необходимо было разработать модульные тесты и протестировать разработанный алгоритм. Пример работы тестов указан на рисунке 3.

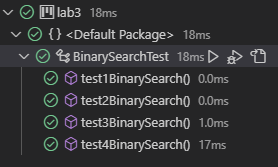


Рисунок 3 – Результат работы модульных тестов

**Вывод**: в результате выполнения лабораторной работы был изучен алгоритм бинарного поиска. Также были изучены способы создания модульных тестов на языке программирования *Java* при помощи библиотеки *Junit*.

**ПРИЛОЖЕНИЕ A**

Листинг класса *BinarySearch*

import java.util.Scanner;

public class BinarySearch {

public static int binarySearch(int[] arr, int target) {

    int left = 0;

    int right = arr.length - 1;

    while (left <= right) {

        int mid = left + (right - left) / 2;

            if (arr[mid] == target) {

                return mid;

            }

        else if (arr[mid] < target) {

            left = mid + 1;

        }

        else {

            right = mid - 1;

        }

    }

    return -1;

}

public static void main(String[] args) {

    Scanner sc = new Scanner(System.in);

    System.out.print("Введите длину массива: ");

    int length = sc.nextInt();

    int[] arr = new int[length];

    System.out.println("Введите элементы массива:");

    for (int i = 0; i < length; i++) {

        arr[i] = sc.nextInt();

    }

    System.out.print("Введите целевое значение: ");

    int target = sc.nextInt();

    int result = binarySearch(arr, target);

    if (result == -1) {

        System.out.println("Целевое значение не найдено");

    }

    else {

        System.out.println("Целевое значение найдено на позиции " + result);

    }

}

}