**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

Демиденко Д.В.

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:**

Для написания алгоритма двоичного поиска был создан класс *Main*. В этот класс передаётся массив целых числе, который сразу сортируется при помощи метода *sort*. Метод *sort* выполняет сортировку массива методом пузырька. Далее был создан сам метод двоичного поиска. Суть этого метода заключается в том, что программа каждый раз делить область проверки данных пополам до тех пор, пока не найдет номер нужного элемента. Если такого элемента нет в массиве метод вернет значение -1 в другом случае метод вернет индекс переданного элемента в массиве. Сложность этого алгоритма равна *O*(*log*(*n*)). Листинг класса *Main* указан в приложении А. Блок схема данного алгоритма указана на рисунке 1.

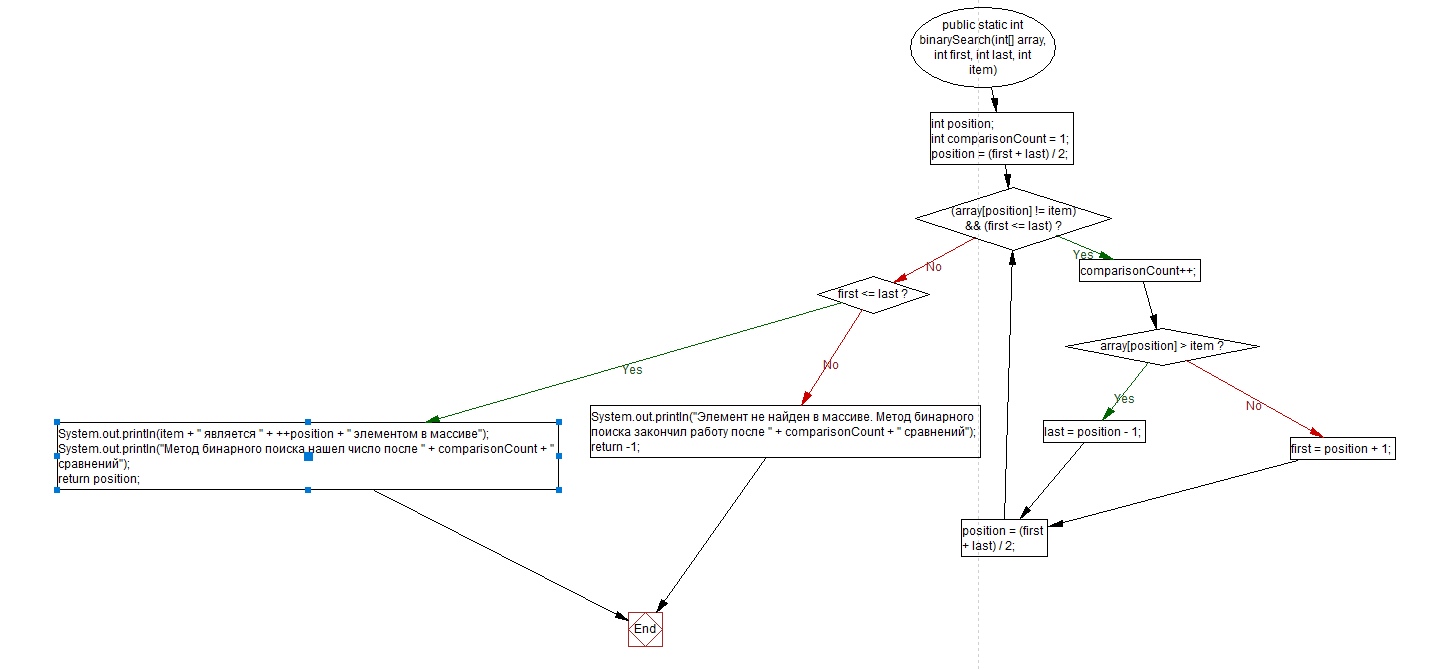


Рисунок 1 – Блок-схема алгоритма бинарного поиска

После написания алгоритма были написаны *Unit* тесты при помощи библиотеки *Junit*. Для написания тестов был создан класс *MainTest*.В котором указана изначальный массив и список тестов. Листинг данного класса указан в приложении А. Результат работы данных тестов указан на рисунке 2.

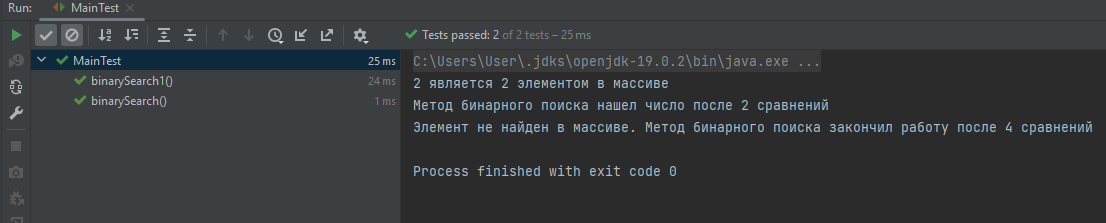


Рисунок 2 – Пример работы тестов

**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 *Main*

import java.util.Arrays;  
import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String args[]) {  
 int counter, num, item, array[], first, last;  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.println("Введите количество элементов массива: ");  
 num = input.nextInt();  
 array = new int[num];  
  
 System.*out*.println("Введите " + num + " чисел");  
  
 for (counter = 0; counter < num; counter++)  
 array[counter] = input.nextInt();  
  
 Arrays.*sort*(array);  
  
 System.*out*.println("Введите элемент для бинарного поиска: ");  
 item = input.nextInt();  
 first = 0;  
 last = num - 1;  
 *binarySearch*(array, first, last, item);  
 }  
  
 public static int binarySearch(int[] array, int first, int last, int item) {  
 int position;  
 int comparisonCount = 1;  
 position = (first + last) / 2;  
 while ((array[position] != item) && (first <= last)) {  
 comparisonCount++;  
 if (array[position] > item) {  
 last = position - 1;  
 } else {  
 first = position + 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");  
 return position;  
 }  
 else {  
 System.*out*.println("Элемент не найден в массиве. Метод бинарного поиска закончил работу после "  
 + comparisonCount + " сравнений");  
 }  
 return -1;  
 }  
}