**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П.О.СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №5

по дисциплине: «Введение в разработку программного обеспечения»

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

Выполнила: студентка гр. ИТП-21

Волосевич М. В.

Проверил: ст. преподаватель

Гуменников Е. Д.

Гомель 2025

**Цель работы:** освоить методы алгоритмизации и моделирования с использованием блок-схем. Научиться реализовывать алгоритмы средствами языка *Java*, тестировать их с помощью модульных тестов, создавать консольные приложения для демонстрации работы алгоритма, а также составлять отчёт о проделанной работе.

**Задание:**

1. Описать блок-схему алгоритма согласно варианта (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.

Таблица 3

|  |  |
| --- | --- |
| **Вариант** | **Условие задачи** |
| 5 | Двоичный поиск |

**Ход работы:**

Блок-схема алгоритма представлена на рисунке 1.

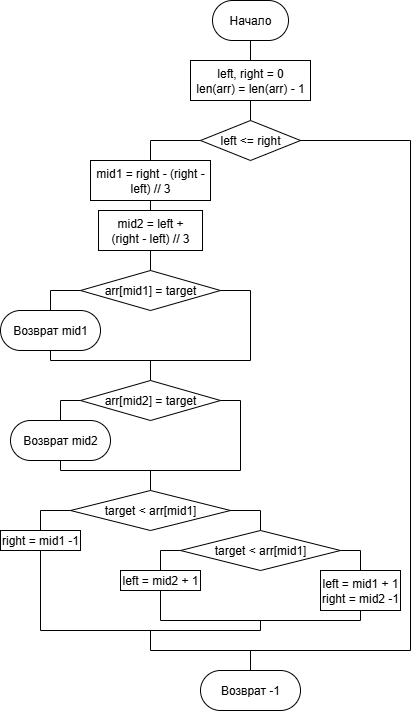


Рисунок 1 – Блок-схема алгоритма

 **Описание программы и созданных файлов:**

В рамках лабораторной работы были созданы следующие файлы:

|  |  |
| --- | --- |
| **Файл** | **Назначение** |
| *TernarySearch*.*java* | Реализация алгоритма троичного поиска. Метод *ternarySearch* принимает отсортированный массив и значение, возвращает индекс найденного элемента или -1. Переменные именованы так же, как в *Python*-примере: *left*, *right*, *mid*1, *mid*2, *arr*, *target*. |
| *App*.*java* | Консольное приложение для демонстрации работы алгоритма. Пользователь может использовать примерный массив по умолчанию или ввести собственный. Программа выводит индекс найденного элемента или сообщение о том, что элемент не найден. |
| *TernarySearchTest*.*java* | Набор модульных тестов для проверки корректности реализации алгоритма. Тестирует поиск в разных массивах, включая пустой массив и различные позиции искомого элемента. |

**Описание работы программы:**

При запуске *App*.*java* пользователю предлагается использовать примерный массив или ввести собственный.

Пользователь вводит искомое целое число.

Программа вызывает метод *TernarySearch*.*ternarySearch*, который ищет элемент в массиве.

Выводится результат: индекс найденного элемента и значение или сообщение о том, что элемент не найден.

**Примеры работы:**

**Пример 1:** используем примерный массив *[*1,2,3,5,8,13,21,34,55,89*]*, ищем 13. **Вывод:** Найдено: индекс = 5, значение = 13. (рис 2)

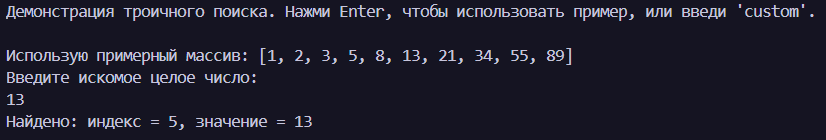


Рисунок 2 – Ищем 13 в исходном массиве (найдено)

**Пример 2:** используем примерный массив *[*1,2,3,5,8,13,21,34,55,89*]*, ищем 7. **Вывод:** Элемент не найден (возвращено -1) (рис 3)

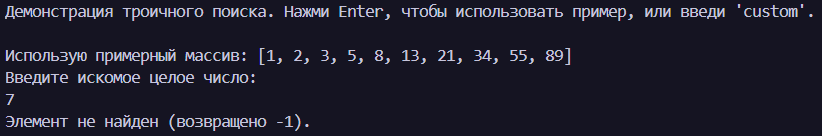


Рисунок 3 – Ищем 7 в предлагаемом массиве (не найдено)

**Пример 3:** Пользователь ввёл массив *[*2,4,6,8,10*]*, ищем 8.

**Вывод:** Найдено: индекс = 3, значение = 8 (рис 4)

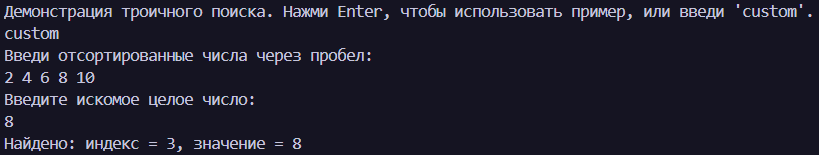


Рисунок 4 – Ищем 8 в массиве пользователя (найдено)

**Пример 4:** Выполнение всех модульных тестов *TernarySearchTest*.*java*. **Вывод:** *Tests* *passed*: 7/7 (рис 5)



Рисунок 5 – Результат успешно пройденных тестов

**Выводы:** в ходе выполнения лабораторной работы:

* была построена блок-схема алгоритма троичного поиска;
* алгоритм реализован на языке *Java*;
* создано консольное приложение *App*.*java* для демонстрации работы алгоритма;
* разработан набор модульных тестов *TernarySearchTest*.*java* для проверки корректности работы;
* проверка работы алгоритма показала правильность всех тестовых случаев;
* полученные навыки позволяют применять моделирование и алгоритмизацию для разработки корректных и проверяемых программ.

ПРИЛОЖЕНИЕ А  
(обязательное)  
Листинг *App*.*java*

*package* *com*.*university*.*lab*5;

*import* *java*.*util*.*Scanner*;

*import* *java*.*util*.*Arrays*;

*/\*\**

*\* Простое консольное приложение для демонстрации работы TernarySearch.*

*\* Читает массив и значение из консоли или использует пример по умолчанию.*

*\*/*

*public* *class* *App* {

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

*// try-with-resources автоматически закроет Scanner*

*try* (*Scanner* *sc* = *new* *Scanner*(*System*.*in*)) {

*System*.*out*.*println*("Демонстрация троичного поиска. Нажми *Enter*, чтобы использовать пример, или введи '*custom*'.");

*String* *line* = *sc*.*nextLine*().*trim*();

*int[]* *arr*;

*if* ("*custom*".*equalsIgnoreCase*(*line*)) {

*System*.*out*.*println*("Введи отсортированные числа через пробел:");

*String* *nums* = *sc*.*nextLine*();

*String[]* *parts* = *nums*.*trim*().*split*("*\\s*+");

*arr* = *new* *int[parts*.*length]*;

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

*arr[i]* = *Integer*.*parseInt*(*parts[i]*);

                }

            } *else* {

*arr* = *new* *int[]*{1, 2, 3, 5, 8, 13, 21, 34, 55, 89}; *// пример*

*System*.*out*.*println*("Использую примерный массив: " + *Arrays*.*toString*(*arr*));

            }

*System*.*out*.*println*("Введите искомое целое число:");

*int* *target* = *Integer*.*parseInt*(*sc*.*nextLine*().*trim*());

*int* *idx* = *TernarySearch*.*ternarySearch*(*arr*, *target*);

*if* (*idx* >= 0) {

*System*.*out*.*printf*("Найдено: индекс = %*d*, значение = %*d*%*n*", *idx*, *arr[idx]*);

            } *else* {

*System*.*out*.*println*("Элемент не найден (возвращено -1).");

            }

        } *// Scanner автоматически закрыт здесь*

    }

}

ПРИЛОЖЕНИЕ Б  
(обязательное)  
Листинг *TernarySearch*.*java*

*package* *com*.*university*.*lab*5;

*/\*\**

*\* Реализация троичного поиска (ternary search) над отсортированным массивом.*

*\* Переменные именованы в соответствии с твоим python-примером:*

*\* left, right, mid1, mid2, arr, target.*

*\*/*

*public* *class* *TernarySearch* {

*/\*\**

*\* Выполняет троичный поиск.*

*\**

*\* @param arr    отсортированный по возрастанию массив int*

*\* @param target искомое значение*

*\* @return индекс найденного элемента или -1, если не найден*

*\*/*

*public* *static* *int* *ternarySearch*(*int[]* *arr*, *int* *target*) {

*int* *left* = 0;

*int* *right* = *arr*.*length* - 1;

*while* (*left* <= *right*) {

*int* *mid*1 = *left* + (*right* - *left*) / 3;

*int* *mid*2 = *right* - (*right* - *left*) / 3;

*if* (*arr[mid*1*]* == *target*)

*return* *mid*1;

*if* (*arr[mid*2*]* == *target*)

*return* *mid*2;

*if* (*target* < *arr[mid*1*]*) {

*right* = *mid*1 - 1;

            } *else* *if* (*target* > *arr[mid*2*]*) {

*left* = *mid*2 + 1;

            } *else* {

*left* = *mid*1 + 1;

*right* = *mid*2 - 1;

            }

        }

*return* -1;

    }

}

ПРИЛОЖЕНИЕ В  
(обязательное)  
Листинг *TernarySearchTest*.*java*

*package* *com*.*university*.*lab*5;

*/\*\**

*\* Небольшой тестовый класс для проверки корректности реализации.*

*\* Запускается как обычная java-программа и печатает результат тестов.*

*\*/*

*public* *class* *TernarySearchTest* {

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

*int* *passed* = 0;

*int* *total* = 0;

*total*++;

*if* (*assertEq*(*TernarySearch*.*ternarySearch*(*new* *int[]* { 1, 2, 3, 4, 5 }, 3), 2))

*passed*++;

*total*++;

*if* (*assertEq*(*TernarySearch*.*ternarySearch*(*new* *int[]* { 1, 2, 3, 4, 5 }, 1), 0))

*passed*++;

*total*++;

*if* (*assertEq*(*TernarySearch*.*ternarySearch*(*new* *int[]* { 1, 2, 3, 4, 5 }, 5), 4))

*passed*++;

*total*++;

*if* (*assertEq*(*TernarySearch*.*ternarySearch*(*new* *int[]* { 1, 2, 3, 4, 5 }, 6), -1))

*passed*++;

*total*++;

*if* (*assertEq*(*TernarySearch*.*ternarySearch*(*new* *int[]* {}, 1), -1))

*passed*++;

*total*++;

*if* (*assertEq*(*TernarySearch*.*ternarySearch*(*new* *int[]* { 2, 4, 6, 8, 10, 12 }, 8), 3))

*passed*++;

*total*++;

*if* (*assertEq*(*TernarySearch*.*ternarySearch*(*new* *int[]* { 2, 4, 6, 8, 10, 12 }, 2), 0))

*passed*++;

*System*.*out*.*printf*("*Tests* *passed*: %*d*/%*d*%*n*", *passed*, *total*);

*if* (*passed* != *total*)

*System*.*exit*(1); *// ненулевой код если есть провал*

    }

*private* *static* *boolean* *assertEq*(*int* *a*, *int* *b*) {

*boolean* *ok* = *a* == *b*;

*if* (!*ok*)

*System*.*out*.*printf*("*FAILED*: *expected* %*d* *but* *got* %*d*%*n*", *b*, *a*);

*return* *ok*;

    }

}