**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

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

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

по дисциплине: «Введение в разработку программного обеспечения»

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

Выполнил студент

группы ИТД-21

Белковец А. В.

Проверил преподаватель

Малиновский И. Л.

Гомель 2024

**Цель работы:** Создать консольное приложение для демонстрации работы алгоритма.

**Задание:**



Рисунок 1 – Вариант задания

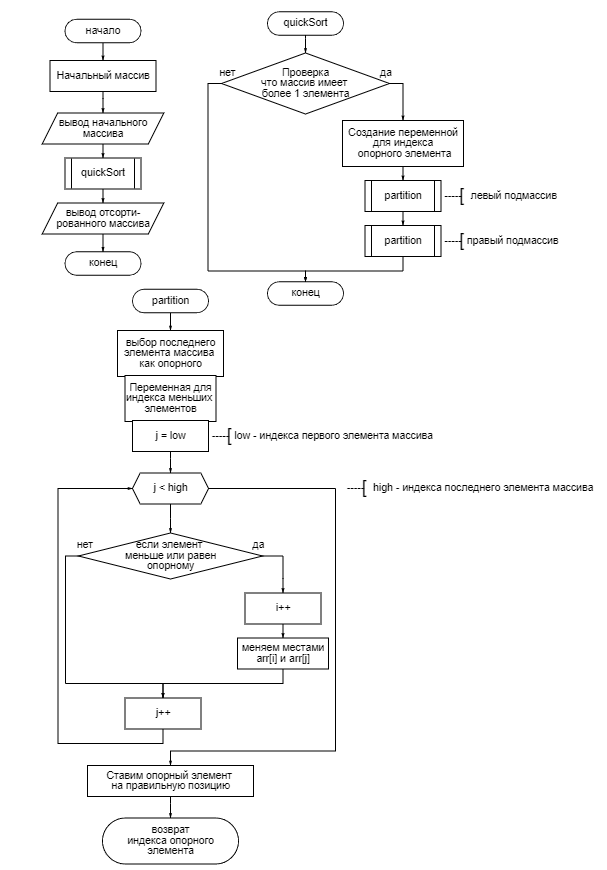


Рисунок 3 – Блок-схема программы

Ход выполнения задания

1. В приложении А указан код программы*.*
2. На рисунке 2 показан запуск программы.

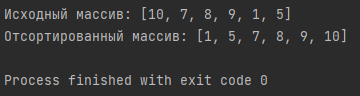


Рисунок 2 – Результат выполнения программы

**Вывод:** была создана программа с быстрой сортировкой на языке программирования *Java*.

**ПРИЛОЖЕНИЕ А**

(Обязательное)

**Текст программы**

***Main.java*:**

import java.util.Arrays;

import static sort.QuickSortExample.quickSort;

public class Main {

public static void main(String[] args) {

int[] arr = {10, 7, 8, 9, 1, 5};

System.out.println("Исходный массив: " + Arrays.toString(arr));

quickSort(arr, 0, arr.length - 1);

System.out.println("Отсортированный массив: " + Arrays.toString(arr));

}

}

**QuickSortExample.java:**

package sort;

public class QuickSortExample {

// Метод быстрой сортировки

public static void quickSort(int[] arr, int low, int high) {

if (low < high) {

// Находим индекс опорного элемента

int pivotIndex = partition(arr, low, high);

// Рекурсивно сортируем левую и правую части массива

quickSort(arr, low, pivotIndex - 1); // Левый подмассив

quickSort(arr, pivotIndex + 1, high); // Правый подмассив

}

}

// Метод для разбиения массива на две части

private static int partition(int[] arr, int low, int high) {

// Выбираем последний элемент в качестве опорного (pivot)

int pivot = arr[high];

int i = low - 1; // Индекс меньших элементов

for (int j = low; j < high; j++) {

// Если текущий элемент меньше или равен опорному

if (arr[j] <= pivot) {

i++;

// Меняем местами arr[i] и arr[j]

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

// Ставим опорный элемент на правильную позицию

int temp = arr[i + 1];

arr[i + 1] = arr[high];

arr[high] = temp;

return i + 1; // Возвращаем индекс опорного элемента

}

}