**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 гр. ИТП-21

Бондаренко И.А.

Принял: преподаватель-стажёр

Башаримов Ю.С.

Гомель 2024

Цель работы: реализовать алгоритм быстрой сортировки на языке *Java,* протестировать его и создать консольное приложение для демонстрации работы.

**Ход работы**

**Задание.** 1. Описать блок-схему алгоритма согласно варианту 1.

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 к заданию 1 приведено на рисунке 1.



Рисунок 1 – Условие к заданию 1

Разработанная блок-схема алгоритма приведена в приложении А.

Код программы находится в приложении Б.

Результат тестирования разработанного алгоритма приведен на рисунке 2.

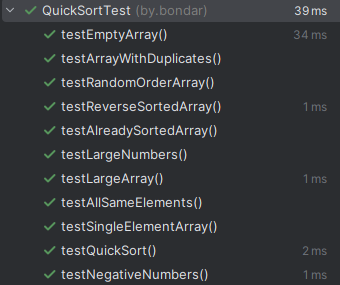


Рисунок 2 – Результат тестирования разработанного алгоритма

Результат выполнения задания 1 представлен на рисунке 3.

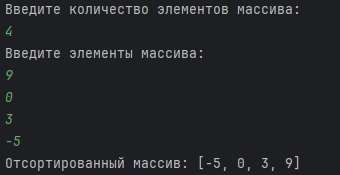


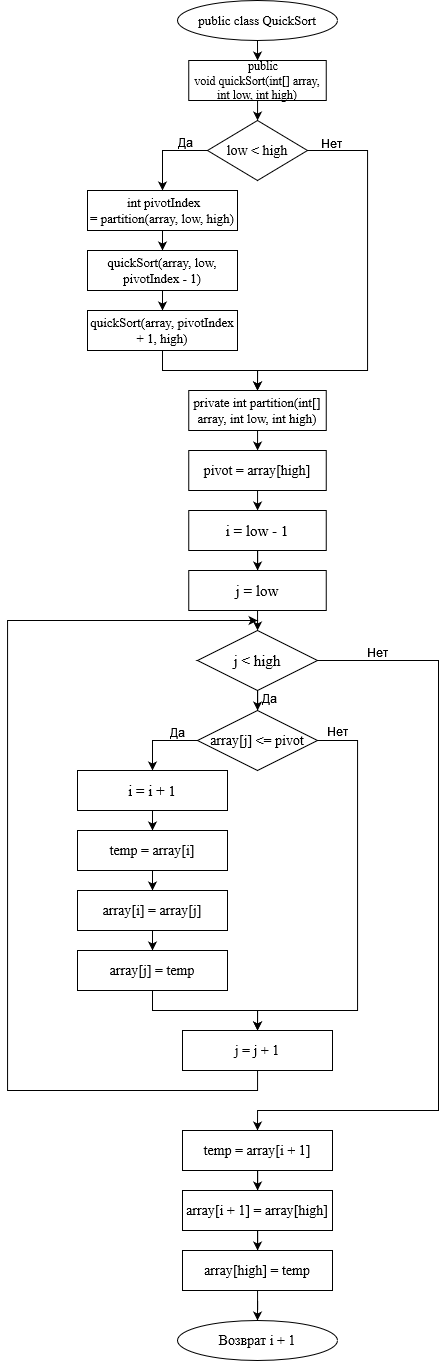
Рисунок 3 – Результат выполнения программы

Вывод: в ходе выполнения лабораторной работы мы реализовали алгоритм быстрой сортировки на языке *Java,* протестировали его и создали консольное приложение для демонстрации работы.  В заключение, моделирование и алгоритмизация являются неотъемлемыми инструментами в проектировании программного обеспечения. Их применение позволяет создать более качественные, устойчивые и легко поддерживаемые системы, что, в свою очередь, отвечает требованиям современного рынка и пользователей.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Блок-схема**



**ПРИЛОЖЕНИЕ Б**

(обязательное)

**Текст программы**

***QuickSort.java*:**

package by.bondar;  
  
public class QuickSort {  
  
 public void quickSort(int[] array, int low, int high) {  
 if (low < high) {  
 int pivotIndex = partition(array, low, high);  
 quickSort(array, low, pivotIndex - 1);  
 quickSort(array, pivotIndex + 1, high);  
 }  
 }  
  
 private int partition(int[] array, int low, int high) {  
 int pivot = array[high];  
 int i = low - 1;  
  
 for (int j = low; j < high; j++) {  
 if (array[j] <= pivot) {  
 i++;  
 int temp = array[i];  
 array[i] = array[j];  
 array[j] = temp;  
 }  
 }  
  
 int temp = array[i + 1];  
 array[i + 1] = array[high];  
 array[high] = temp;  
  
 return i + 1;  
 }  
}

***QuickSortApp.java*:**

package by.bondar;  
  
import java.util.Arrays;  
import java.util.Scanner;  
  
public class QuickSortApp {  
  
 public void run() {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Введите количество элементов массива:");  
 int n = scanner.nextInt();  
 int[] array = new int[n];  
  
 System.*out*.println("Введите элементы массива:");  
 for (int i = 0; i < n; i++) {  
 array[i] = scanner.nextInt();  
 }  
  
 QuickSort quickSort = new QuickSort();  
 quickSort.quickSort(array, 0, array.length - 1);  
 System.*out*.println("Отсортированный массив: " + Arrays.*toString*(array));  
 scanner.close();  
 }  
  
 public static void main(String[] args) {  
 new QuickSortApp().run();  
 }  
}

***QuickSortTest.java*:**

package by.bondar;  
  
import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.*assertArrayEquals*;  
  
public class QuickSortTest {  
  
 private final QuickSort quickSort = new QuickSort();  
  
 @Test  
 public void testQuickSort() {  
 int[] array = {3, 6, 8, 10, 1, 2, 1};  
 quickSort.quickSort(array, 0, array.length - 1);  
 *assertArrayEquals*(new int[]{1, 1, 2, 3, 6, 8, 10}, array);  
 }  
  
 @Test  
 public void testEmptyArray() {  
 int[] array = {};  
 quickSort.quickSort(array, 0, array.length - 1);  
 *assertArrayEquals*(new int[]{}, array);  
 }  
  
 @Test  
 public void testSingleElementArray() {  
 int[] array = {5};  
 quickSort.quickSort(array, 0, array.length - 1);  
 *assertArrayEquals*(new int[]{5}, array);  
 }  
  
 @Test  
 public void testArrayWithDuplicates() {  
 int[] array = {3, 1, 2, 1, 3};  
 quickSort.quickSort(array, 0, array.length - 1);  
 *assertArrayEquals*(new int[]{1, 1, 2, 3, 3}, array);  
 }  
  
 @Test  
 public void testAlreadySortedArray() {  
 int[] array = {1, 2, 3, 4, 5};  
 quickSort.quickSort(array, 0, array.length - 1);  
 *assertArrayEquals*(new int[]{1, 2, 3, 4, 5}, array);  
 }  
  
 @Test  
 public void testReverseSortedArray() {  
 int[] array = {5, 4, 3, 2, 1};  
 quickSort.quickSort(array, 0, array.length - 1);  
 *assertArrayEquals*(new int[]{1, 2, 3, 4, 5}, array);  
 }  
  
 @Test  
 public void testNegativeNumbers() {  
 int[] array = {-1, -3, -2, 0, 2};  
 quickSort.quickSort(array, 0, array.length - 1);  
 *assertArrayEquals*(new int[]{-3, -2, -1, 0, 2}, array);  
 }  
  
 @Test  
 public void testAllSameElements() {  
 int[] array = {2, 2, 2, 2, 2};  
 quickSort.quickSort(array, 0, array.length - 1);  
 *assertArrayEquals*(new int[]{2, 2, 2, 2, 2}, array);  
 }  
  
 @Test  
 public void testLargeArray() {  
 int[] array = {3, 6, 8, 10, 1, 2, 1, 9, 5, 7, 4, 0, 3, 2, 8, 6, 1};  
 quickSort.quickSort(array, 0, array.length - 1);  
 *assertArrayEquals*(new int[]{0, 1, 1, 1, 2, 2, 3, 3, 4, 5, 6, 6, 7, 8, 8, 9, 10}, array);  
 }  
 @Test  
 public void testLargeNumbers() {  
 int[] array = {Integer.*MAX\_VALUE*, Integer.*MIN\_VALUE*, 0, 100000, -100000};  
 quickSort.quickSort(array, 0, array.length - 1);  
 *assertArrayEquals*(new int[]{Integer.*MIN\_VALUE*, -100000, 0, 100000, Integer.*MAX\_VALUE*}, array);  
 }  
 @Test  
 public void testRandomOrderArray() {  
 int[] array = {2, 5, 3, 1, 4};  
 quickSort.quickSort(array, 0, array.length - 1);  
 *assertArrayEquals*(new int[]{1, 2, 3, 4, 5}, array);  
 }  
}