**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

Иванов Д. А.

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 10 – Сортировка подсчетом.

**Решение:**

Сортировка подсчетом (*Counting Sort*) - это алгоритм сортировки, который использует массивы для сортировки целых чисел. Он работает за линейное время и эффективен для сортировки чисел в небольшом диапазоне. Блок схема данного алгоритма указана на рисунке 1.

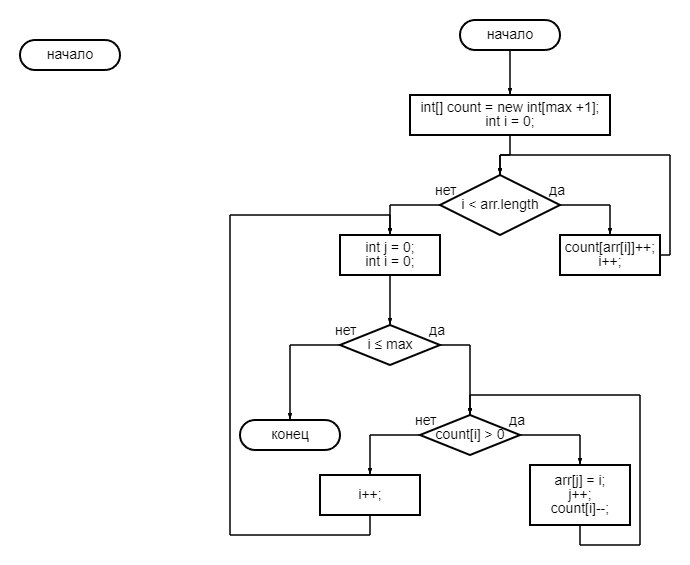


Рисунок 1 – Блок-схема алгоритма

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

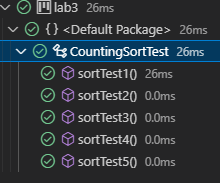


Рисунок 2 – Пример работы тестов

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

**ПРИЛОЖЕНИЕ A**

Листинг класса *CountingSort*

public class CountingSort {

    public static void sort(int[] arr, int max) {

        int[] count = new int[max + 1]; // создаем массив для подсчета

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

            count[arr[i]]++; // увеличиваем счетчик в соответствии с числом

        }

        int j = 0;

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

            while (count[i] > 0) {

                arr[j] = i; // заполняем массив отсортированными элементами

                j++;

                count[i]--;

            }

        }

    }

}

Листинг класса *CountingSortTest*

import org.junit.Test;

import static org.junit.Assert.assertArrayEquals;

public class CountingSortTest {

    @Test

    public void sortTest1() {

        int[] arr = {2, 3, 1, 5, 4};

        int[] expected = {1, 2, 3, 4, 5};

        CountingSort.sort(arr, 5);

        assertArrayEquals(expected, arr);

    }

    @Test

    public void sortTest2() {

        int[] arr = {7, 4, 5, 2, 1};

        int[] expected = {1, 2, 4, 5, 7};

        CountingSort.sort(arr, 7);

        assertArrayEquals(expected, arr);

    }

    @Test

    public void sortTest3() {

        int[] arr = {5, 4, 3, 2, 1};

        int[] expected = {1, 2, 3, 4, 5};

        CountingSort.sort(arr, 5);

        assertArrayEquals(expected, arr);

    }

    @Test

    public void sortTest4() {

        int[] arr = {2, 3, 4, 1, 5};

        int[] expected = {1, 2, 3, 4, 5};

        CountingSort.sort(arr, 5);

        assertArrayEquals(expected, arr);

    }

    @Test

    public void sortTest5() {

        int[] arr = {1, 2, 3, 4, 5};

        int[] expected = {1, 2, 3, 4, 5};

        CountingSort.sort(arr, 5);

        assertArrayEquals(expected, arr);

    }

}