Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №5**

**«ИССЛЕДОВАНИЕ АЛГОРИТМОВ СОРТИРОВКИ»**

**ПО ДИСЦИПЛИНЕ «ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ»**

Выполнил: студент учебной группы

ИСПк-203-52-00

Вальковская Арина Константиновна

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

1. **Цель работы:**

Получить базовые сведения о наиболее известных алгоритмах сортировки, изучить принципы работы с текстовыми файлами.

1. **Формулировка задания:**

Вариант: 4

1. Реализовать сортировку данных с помощью алгоритма подсчетом.
2. Реализовать сортировку данных с помощью пирамидального алгоритма.
3. В обоих случаях необходимо предусмотреть возможность изменения компаратора (реализация компаратора в виде передаваемой в подпрограмму функции).
4. Считывание и вывод данных необходимо производить из текстового файла.
5. Для демонстрации работы программных реализаций самостоятельно подготовить варианты входных данных (при этом объем текстовых файлов должен позволять оценить скорость работы программ).
6. **Описание алгоритма**

**Задание 1**

Программа начинается с определения типа arr, который представляет собой массив из 100 целых чисел. Затем объявляются переменные f1 и f2 как файловые переменные, а также переменные i, n и a.

Далее следует процедура countSort, которая принимает массив a и его размер n в качестве параметров. В этой процедуре создается массив count из 100 элементов и инициализируется нулями. Затем происходит подсчет количества вхождений каждого элемента массива a и сохранение этой информации в массив count. Затем элементы массива a перезаписываются в упорядоченном порядке на основе информации из массива count.

В основной части программы открывается файл inputt.txt для чтения данных, считывается значение n (размер массива) и элементы массива a из файла. Файл затем закрывается, и вызывается процедура countSort для сортировки массива a. Затем открывается файл output.txt для записи отсортированных элементов массива a, и они записываются в файл. Файл затем закрывается, и на экран выводится сообщение о успешной сортировке.

**Задание 2**

Программа начинается с определения типа данных arr - это массив из 100 целых чисел. Затем объявляются файловые переменные f1 и f2, а также переменные i и n типа Integer и массив a типа arr.

Затем следуют две процедуры: heapify и heapSort. Процедура heapify используется для преобразования массива в кучу, а процедура heapSort выполняет собственно сортировку пирамидальным методом.

В основном блоке программы происходит открытие файла input.txt, считывание размера массива и его элементов из этого файла, сортировка массива с помощью процедуры heapSort, запись отсортированных элементов в файл output.txt и вывод сообщения о успешном завершении сортировки.

1. **Схемы алгоритмов**

**Задание 1**

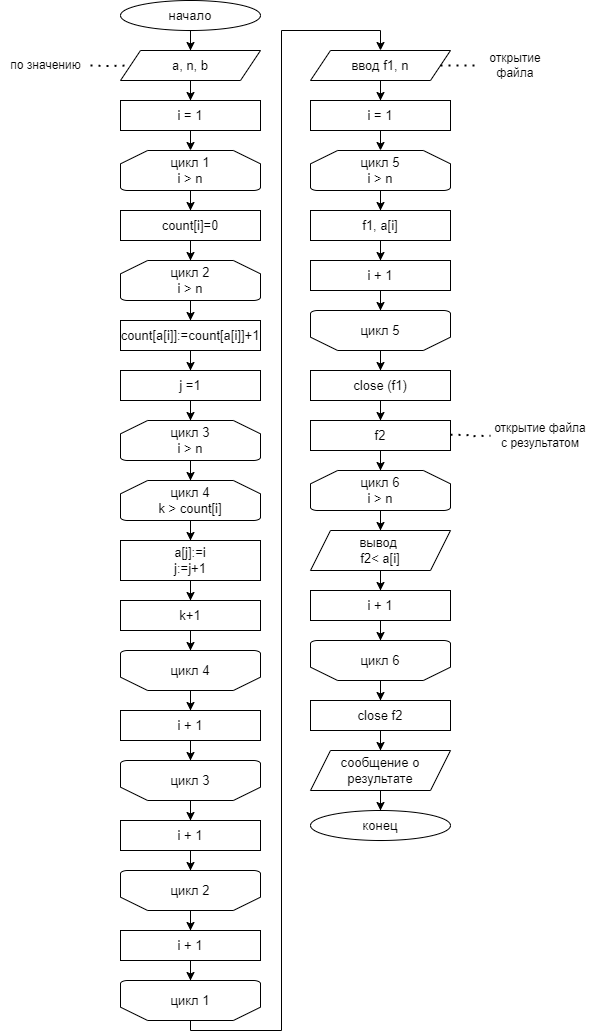


Рисунок 1 – Схема алгоритма 1

**Задание 2**

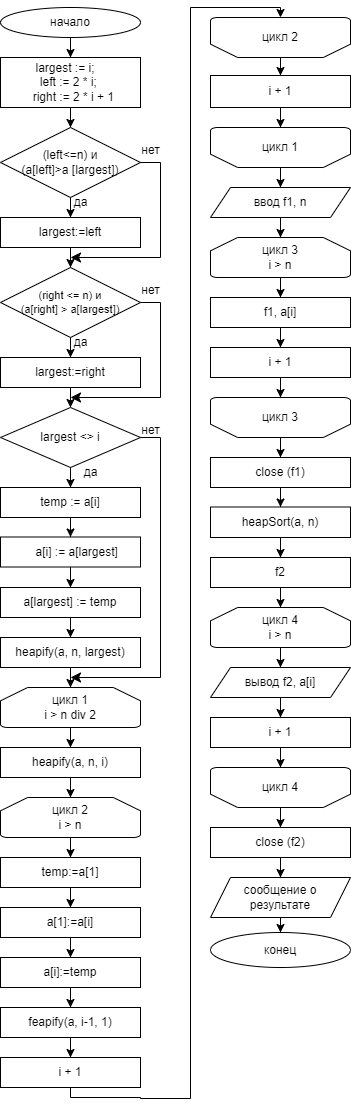
****

Рисунок 2 – Схема алгоритма 2

1. **Код программы**

**Задание 1**

**program** z1;

**type**

arr = **array**[1..100] **of** Integer;

**var**

f1, f2: text; // файловые переменные

i, n: Integer;

a: arr;

**procedure** countSort(**var** a: arr; n: Integer); // упорядочивание элементов массива

**var**

i, j, k: Integer;

count: **array**[1..100] **of** Integer;

**begin**

**for** i := 1 **to** n **do**

count[i] := 0;

**for** i := 1 **to** n **do**

count[a[i]] := count[a[i]] + 1;

j := 1;

**for** i := 1 **to** 100 **do**

**begin**

**for** k := 1 **to** count[i] **do**

**begin**

a[j] := i;

j := j + 1;

**end**;

**end**;

**end**;

**begin**

assign(f1, 'inputt.txt'); // открытие файла с данными

reset(f1);

readln(f1, n); // счетчик элементов в файле

**for** i := 1 **to** n **do**

read(f1, a[i]);

close(f1);

countSort(a, n); // сортировка алгоритмом подсчета

assign(f2, 'output.txt'); // открытие файла с результатом

rewrite(f2);

**for** i := 1 **to** n **do** // запись отсортированных в файл

write(f2, a[i], ' ');

close(f2);

writeln('Сортировка прошла успешно. Результат записан в файл output.txt.');

**end**.

**Задание 2**

**program** z2;

**type**

arr = **array**[1..100] **of** Integer;

**var**

f1, f2: text; // файловые переменные

i, n: Integer;

a: arr;

**procedure** heapify(**var** a: arr; n, i: Integer);

**var**

largest, left, right, temp: Integer;

**begin**

largest := i;

left := 2 \* i;

right := 2 \* i + 1;

**if** (left <= n) **and** (a[left] > a[largest]) **then**

largest := left;

**if** (right <= n) **and** (a[right] > a[largest]) **then**

largest := right;

**if** largest <> i **then**

**begin**

temp := a[i];

a[i] := a[largest];

a[largest] := temp;

heapify(a, n, largest);

**end**;

**end**;

**procedure** heapSort(**var** a: arr; n: Integer);

**var**

i, temp: Integer;

**begin**

**for** i := n **div** 2 **downto** 1 **do**

heapify(a, n, i);

**for** i := n **downto** 2 **do**

**begin**

temp := a[1];

a[1] := a[i];

a[i] := temp;

heapify(a, i - 1, 1);

**end**;

**end**;

**begin**

assign(f1, 'inputt.txt'); // открытие файла с данными

reset(f1);

readln(f1, n); // считывание размера массива

**for** i := 1 **to** n **do** // считывание элементов массива

read(f1, a[i]);

close(f1);

heapSort(a, n); // сортировка алгоритмом пирамидальной сортировки

assign(f2, 'output.txt'); // открытие файла с результатом

rewrite(f2);

**for** i := 1 **to** n **do** // запись отсортированных в файл

write(f2, a[i], ' ');

close(f2);

writeln('Сортировка прошла успешно. Результат записан в файл output.txt.');

**end**.

1. **Результат выполнения программ**

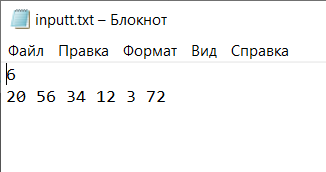
****

Рисунок 3 – Массив для программы 1

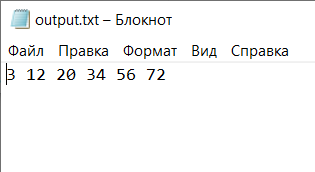


Рисунок 4 – Результат выполнения программы 1

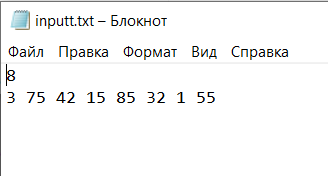


Рисунок 5 – Массив для программы 2

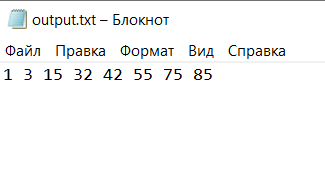


Рисунок 6 – Результат выполнения программы 2

1. **Вывод**

В ходе данной работы мы познакомились с сортировками, они показались мне очень сложными, но зато теперь мы знаем несколько способов и алгоритмов сортировки. Было не мало проблем, но мы отважно их решали. Было прочитано несколько сайтов в поисках необходимой информации, так что эта работа результат упорных трудов! Так же мы познакомились с текстовыми файлами, с которыми можно работать через паскаль.

В процессе выполнения отчета, для создания блок схемы мы работали с программой draw.io. Draw.io — это бесплатный онлайн-сервис, который помогает создавать алгоритмы, прототипы, инфографику и диаграммы любого вида. Чаще всего его используют именно для построения диаграмм. Сервисом пользуются разные IT-специалисты — от бизнес-аналитиков до разработчиков и дизайнеров.