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

Леушина Анна Станиславовна

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

**Цель работы:**

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

**Формулировка задания:**

Вариант: 15

1. Реализовать сортировку данных с помощью алгоритма выбором.
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).

**Схемы алгоритмов:**

**Задание 1.**

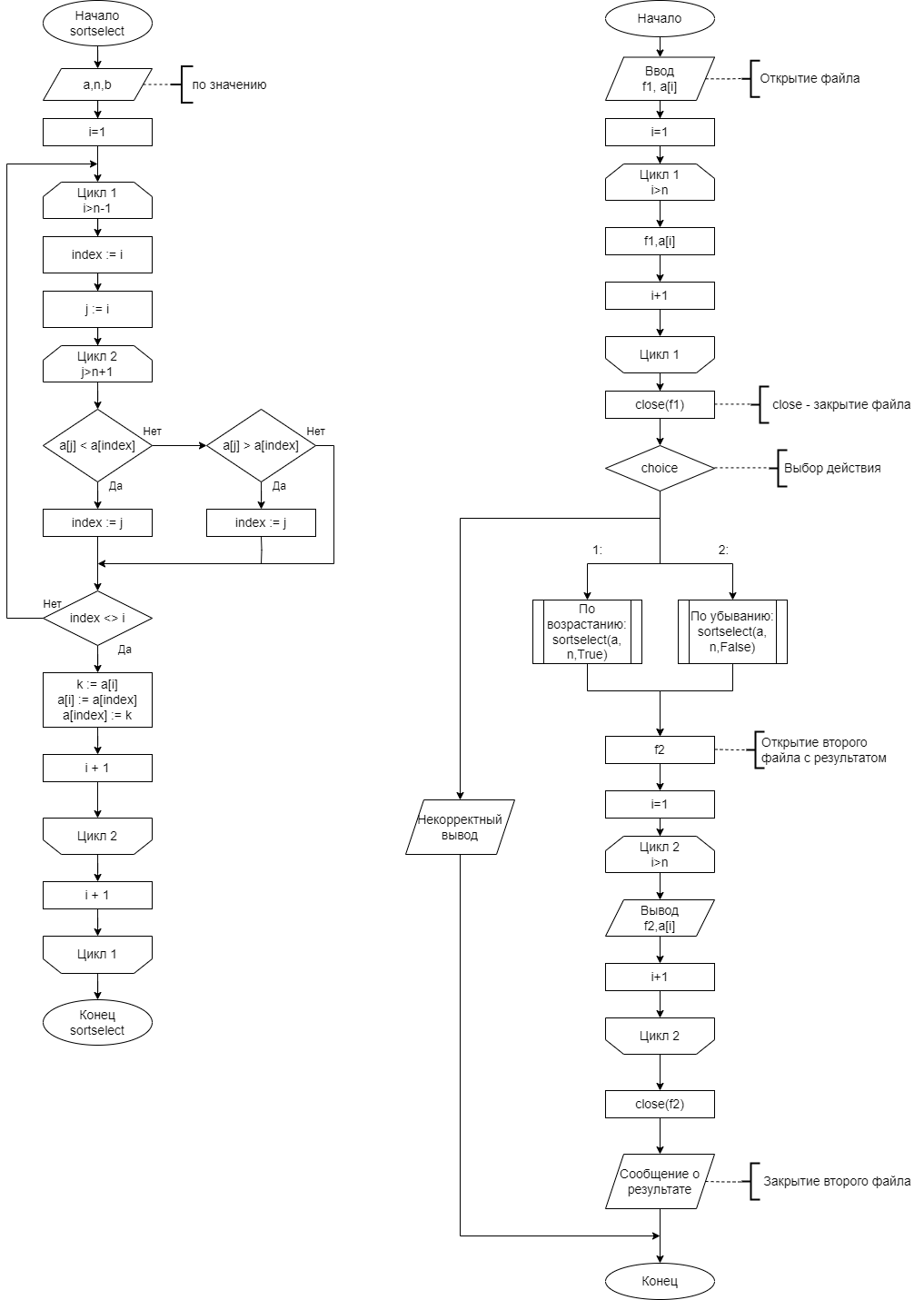
****

Рисунок 1 – Схема Алгоритма.

**Задание 2.**

Рисунок 2 – Схема Алгоритма.

**Код программы:**

**Задание 1.**

**program** n1;

**type**

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

**var**

f1, f2: text;

i, n: Integer;

a: arr;

**procedure** sortselect(**var** a: arr; n: Integer; b: Boolean);

**var**

i, j, index, k: Integer;

**begin**

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

**begin**

index := i;

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

**begin**

**if** b **then**

**begin**

**if** a[j] < a[index] **then**

index := j;

**end**

**else**

**begin**

**if** a[j] > a[index] **then**

index := j;

**end**;

**end**;

**if** index <> i **then**

**begin**

k := a[i];

a[i] := a[index];

a[index] := k;

**end**;

**end**;

**end**;

**begin**

assign(f1, 'zzz.txt');

reset(f1);

readln(f1, n);

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

read(f1, a[i]);

close(f1);

writeln('Выберите порядок сортировки:');

writeln('1. По возрастанию');

writeln('2. По убыванию');

write('Введите номер: ');

**var** choice: Integer;

readln(choice);

**case** choice **of**

1: sortselect(a, n, True); // сортировка по возрастанию

2: sortselect(a, n, False); // сортировка по убыванию

**else**

**begin**

writeln('Некорректный выбор.');

Halt;

**end**;

**end**;

assign(f2, 'xxx.txt');

rewrite(f2);

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

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

close(f2);

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

**end**.

**Задание 2.**

**program** n2;

**type**

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

**var**

f1, f2: text;

i, n: integer;

a: arr;

// Процедура пирамидной сортировки

**procedure** heapsort(**var** a: arr; n: integer; b: boolean);

**var**

i, j, index, k: integer;

// Процедура перестроения пирамиды

**procedure** heapify(**var** a: arr; n, i: integer; b: boolean);

**var**

largest, l, r: integer;

**begin**

largest := i;

l := 2 \* i;

r := 2 \* i + 1;

**if** b **then**

**begin**

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

largest := l;

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

largest := r;

**end**

**else**

**begin**

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

largest := l;

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

largest := r;

**end**;

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

**begin**

k := a[i];

a[i] := a[largest];

a[largest] := k;

heapify(a, n, largest, b);

**end**;

**end**;

**begin**

// Построение пирамиды

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

heapify(a, n, i, b);

// Перестановка элементов для получения отсортированного массива

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

**begin**

k := a[1];

a[1] := a[i];

a[i] := k;

heapify(a, i - 1, 1, b);

**end**;

**end**;

**begin**

assign(f1, 'aaa.txt');

reset(f1);

readln(f1, n);

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

read(f1, a[i]);

close(f1);

writeln('выберите порядок сортировки:');

writeln('1. по возрастанию');

writeln('2. по убыванию');

write('введите номер: ');

**var** choice: integer;

readln(choice);

**case** choice **of**

1: heapsort(a, n, true); // сортировка по возрастанию

2: heapsort(a, n, false); // сортировка по убыванию

**else**

**begin**

writeln('некорректный выбор.');

halt;

**end**;

**end**;

assign(f2, 'bbb.txt');

rewrite(f2);

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

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

close(f2);

writeln('сортировка прошла успешно! результат записан в файл bbb.txt!');

**end**.

**Результат выполнения программы:**

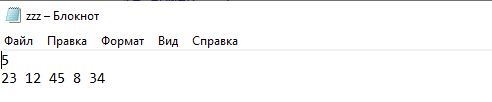
****

Рисунок 3 – Файл до сортировки выбором

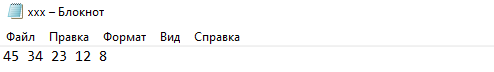
****

Рисунок 4 – Файл после сортировки выбором

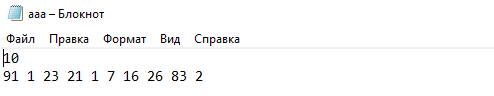
****

Рисунок 5 – Файл до пирамидальной сортировки

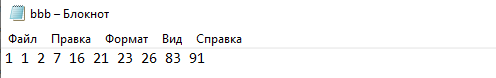
****

Рисунок 6 – Файл после пирамидальной сортировки

**Вывод.**

В ходе выполнения пятой домашней контрольной работы были изучены алгоритмы сортировки, а также принцип работы с текстовыми файлами при помощи вставок в языке PascalABS.NET. Освоен синтаксис построения процедур и выработан навык организации минимального пользовательского интерфейса.

В написании кода возникло множество трудностей, связанные с сортировкой данных с помощью пирамидального алгоритма. Путем решения данной проблемы стало выполнение 13, 14 и 15 лабораторной работы по предмету «Основы алгоритмизации и программирования». Использованы процедуры (procedure), которые прописываются заранее, а в дальнейшем вызывают их, что сокращает код программы и упрощает работу case-меню. Также в работе был использован case-меню, для наглядного представления выполнения программы.

Таким образом, домашняя контрольная работа была выполнена в полном объеме.