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 ПРОГРАМИРОВАНИЯ»**

Выполнил: студент учебной группы

ИСПк-203-52-00

Шумилов Иван Андреевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

**1.** **Цель работы:**

Цель работы получить базовые сведения о наиболее известных алгоритмах сортировки, изучить принципы работы с текстовыми файлами

**2. Формулировка задания:**

**Вариант 24**

Задание состоит из пяти частей:

1. Реализовать сортировку данных с помощью алгоритма подсчетом
2. Реализовать сортировку данных с помощью быстрого 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 тестовых файлов должен позволять оценить скорость работы программ).

**3. Описание алгоритма:**

1. **Исходные данные:** Программа работает с файлами, содержащими числа. Есть три файла: 'маленький файл.txt', 'средний файл.txt' и 'большой файл.txt', которые содержат 12, 50 и 200 чисел соответственно.
2. **Сортировка подсчетом (SortCount):** Эта процедура читает числа из одного из файлов, используя массив **arr**, чтобы подсчитать количество каждого числа. Затем она записывает отсортированные числа в файл 'output.txt'. Но в текущей реализации не производится сортировка чисел.
3. **Быстрая сортировка (QuickSort и QuickSortWrapper):** В коде присутствует быстрая сортировка, реализованная с использованием рекурсивного алгоритма быстрой сортировки. Процедура **QuickSort** выполняет сортировку массива чисел. Процедура **QuickSortWrapper** служит оболочкой для удобства вызова.
4. **Опции взаимодействия с пользователем:** Программа предоставляет меню с тремя пунктами: сортировка подсчетом, быстрая сортировка и выход. Для каждого типа сортировки предоставляется возможность выбора направления сортировки (по возрастанию или убыванию).
5. **Выбор типа файла:** Программа предоставляет опцию выбора типа файла (маленький, средний, большой) при вызове процедур сортировки. Однако в текущей версии программы эти опции не используются полностью, так как внутри процедур сортировки прописаны фиксированные пути к файлам.
6. **Изменение названий файлов:** Программа позволяет изменять названия файлов в процедурах сортировки, а также предоставляет возможность выбора направления сортировки (по возрастанию или убыванию).
7. **Основной цикл программы:** Основная часть программы представляет собой бесконечный цикл, который продолжается до выбора пункта выхода (пункт 3).
8. **Замечания:** В коде есть несколько мест, где происходит вызов **clrscr**, что очищает экран консоли. Это может быть нецелесообразным в некоторых средах выполнения.

**4. Схема алгоритма с комментариями:**

Изображение выглядит как снимок экрана, текст, черно-белый, Графика

Автоматически созданное описание

Рис. 5 – Схема основной программы

Изображение выглядит как Красочность, снимок экрана, шаблон, черно-белый

Автоматически созданное описание

Рис. 4 – Схема алгоритма для сортировки алгоритмом подсчета

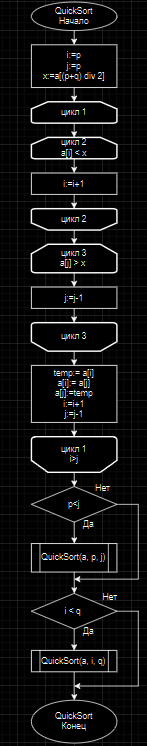


Рис.1 Схема алгоритма быстрой сортировки

**5. Код программы:**

**program** DKR\_5;

**Uses** CRT;

**type**

mas = **array** [1..1000] **of** integer;

**var**

m, out: text;

ch: integer;

**procedure** SortCount;

**var**

arr: **array**[-100..100] **of** integer; //arr[i] - количество чисел i в массиве

i, j, x: integer;

**begin**

Assign(m, 'маленький файл.txt');

Reset(m);

Writeln('Исходный массив: ');

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

arr[i] := 0;

**while not** eof(m) **do**

**begin**

readln(m, x);

write(x, ' ');

inc(arr[x]);

**end**;

Assign(out, 'output.txt');

ReWrite(out);

Writeln;

Writeln('Отсортированный массив: ');

**if** ch = 1 **then**

**begin**

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

**for** j := 1 **to** arr[i] **do**

**begin**

writeln(out, i);

print(i);

**end**;

**end**;

**if** ch = 2 **then**

**begin**

**for** i := 100 **downto** -100 **do**

**for** j := 1 **to** arr[i] **do**

**begin**

writeln(out, i);

print(i);

**end**;

**end**;

writeln;

writeln;

writeln('Отсортированные данные записаны в c:\output.txt');

close(out);

close(m);

**end**;

**procedure** QuickSort(**var** a: mas; p, q: integer);

**var**

i, j, x, temp: integer;

**begin**

i := p;

j := q;

x := a[(p + q) **div** 2];

**repeat**

**while** a[i] < x **do**

i := i + 1;

**while** a[j] > x **do**

j := j - 1;

**if** i <= j **then**

**begin**

temp := a[i];

a[i] := a[j];

a[j] := temp;

i := i + 1;

j := j - 1;

**end**;

**until** i > j;

**if** p < j **then**

QuickSort(a, p, j);

**if** i < q **then**

QuickSort(a, i, q);

**end**;

**procedure** QuickSortWrapper;

**var**

a: mas;

n, i, x: integer;

**begin**

clrscr;

Writeln('Исходный массив: ');

Assign(m, 'маленький файл.txt');

Reset(m);

n := 0;

**repeat**

readln(m, x);

inc(n);

**until** eof(m);

Reset(m);

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

**begin**

readln(m, a[i]);

print(a[i]);

**end**;

close(m);

writeln;

QuickSort(a, 1, N);

writeln('Отсортированный массив:');

Assign(out, 'output.txt');

ReWrite(out);

**if** ch = 1 **then**

**begin**

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

**begin**

Writeln(out, a[i]);

print(a[i]);

**end**;

**end**;

**if** ch = 2 **then**

**begin**

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

**begin**

Writeln(out, a[i]);

print(a[i]);

**end**;

**end**;

writeln;

writeln;

writeln('Отсортированные данные записаны в c:\output.txt');

close(out);

**end**;

**begin**

**repeat**

ClrScr;

Writeln('1. Сортировка подсчетом');

Writeln('2. Сортировка быстрым алгоритмом');

Writeln('3. Выход');

Write('Выберите пункт: ');

Readln(ch);

**case** ch **of**

1:

**begin**

clrscr;

Writeln('1. Отсортировать по возрастанию');

Writeln('2. Отсортировать по убыванию');

Write('Выберите пункт: ');

Readln(ch);

**case** ch **of**

1:

**begin**

ClrScr;

SortCount;

Readln;

**end**;

2:

**begin**;

ClrScr;

SortCount;

Readln;

**end**;

**end**;

**end**;

2:

**begin**

clrscr;

Writeln('1. Отсортировать по возрастанию');

Writeln('2. Отсортировать по убыванию');

Write('Выберите пункт: ');

Readln(ch);

**case** ch **of**

1:

**begin**

ClrScr;

QuickSortWrapper;

Readln;

**end**;

2:

**begin**;

ClrScr;

QuickSortWrapper;

Readln;

**end**;

**end**;

**end**;

3: **exit**;

**end**;

**until** ch = 3;

**end**.

**6. Результат выполнения программы**

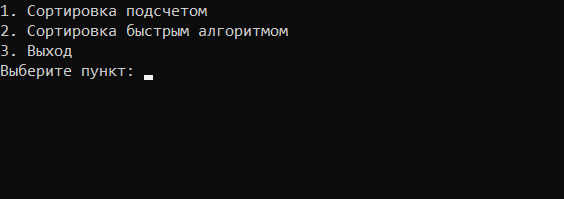
****

Рис.4 Меню выбора.

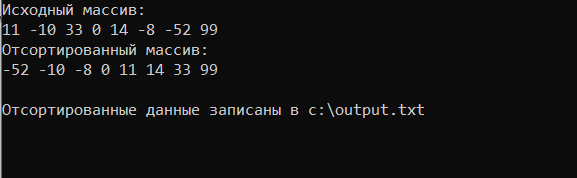
****

Рис.5 Результат сортировки.

**7. Вывод:**

В процессе выполнения работы по созданию программы сортировки на языке Pascal, мы изучили принципы организации кода с использованием процедур и функций. Программа включает в себя два алгоритма сортировки: подсчетом и быструю сортировку. Мы также реализовали возможность изменения порядка сортировки через компаратор.

Каждый из алгоритмов включает этапы обработки данных, такие как нахождение минимального и максимального значений (подсчет), разделение массива и рекурсивная сортировка (быстрая сортировка).

Для демонстрации работы программы созданы варианты входных данных в виде тестовых файлов. Это позволяет оценить скорость и корректность работы алгоритмов на различных наборах данных.

Таким образом, работа не только закрепила навыки работы с процедурами и функциями, но и продемонстрировала их использование в решении конкретной задачи сортировки данных. Полученный опыт может быть полезен для решения более сложных задач на языке Pascal и разработки эффективных алгоритмов.