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-206-52-00

Долинин Владислав Александрович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

1. Цель работы
2. Формулировка задания (с вариантом)
3. Описание алгоритма и ответы на вопросы
4. Схема алгоритма с комментариями
5. Код программы
6. Результат выполнения программы
7. Вывод

**1. Цель работы:** получить базовые сведенья о наиболее известных алгоритмах сортировки, изучить принципы работы с текстовыми файлами.

**2.Формулировка заданий, 2 Вариант.**

1. Реализовать сортировку данных с помощью вставок.

2. Реализовать сортировку данных с помощью быстрого алгоритма.

3. В обоих случаях необходимо предусмотреть возможность изменения компаратора.

4. Считывание и вывод данных необходимо производить из текстового файла.

5. Для демонстрации работы программных реализаций самостоятельно подготовить варианты входных данных (при этом объём текстовых файлов должен позволять оценить скорость работы программ).

**3. Описание алгоритма.**

Программа №1

AscendingComparator (a < b)

DescendingComparator (a > b)

InsertionSort принимает массив arr, его длину n и функцию сравнения comparator.

Проходит по элементам массива с индекса 1 до n-1.

Для каждого элемента:

Сохраняет текущий элемент как ключ (key).

Сравнивает ключ с предыдущими элементами массива через функцию comparator.

Если предыдущий элемент больше ключа (в зависимости от используемой функции сравнения), то сдвигает элементы вправо, пока не найдет подходящее место для вставки ключа.

Вставляет ключ на правильное место.

ReadDataFromFile открывает файл filename для чтения.

Читает первое число из файла – это длина массива n.

Выделяет память под массив длиной n.

Читает все элементы массива из файла.

Закрывает файл.

WriteDataToFile открывает файл filename для записи.

Записывает все элементы отсортированного массива в файл, разделяя их пробелами.

Закрывает файл.

Вызывается процедура ReadDataFromFile для считывания данных из файла input.txt.

Вызывается процедура InsertionSort для сортировки массива по возрастанию с использованием функции AscendingComparator.

Вызывается процедура WriteDataToFile для записи отсортированных данных в файл output.txt.

Программа №2

AscendingComparator (a < b)

DescendingComparator (a > b)

QuickSort принимает массив arr, границы диапазона для сортировки (low, high), а также функцию сравнения comparator.

Если диапазон для сортировки содержит более одного элемента (low < high), выполняется следующая последовательность действий:

Выбирается последний элемент массива как опорный (pt).

Переменная i инициализируется значением low - 1.

Цикл проходит по всем элементам от low до high - 1:

Если текущий элемент меньше опорного (согласно функции сравнения), увеличивается индекс i и происходит обмен текущего элемента с элементом на позиции i.

После завершения цикла происходит обмен элемента на позиции i+1 с опорным элементом.

Рекурсивно вызываются две части массива: левая часть (от low до i) и правая часть (от i+2 до high).

ReadDataFromFile открывает файл filename для чтения.

Читает первое число из файла – это длина массива n.

Выделяет память под массив длиной n.

Читает все элементы массива из файла.

Закрывает файл.

WriteDataToFile открывает файл filename для записи.

Записывает все элементы отсортированного массива в файл, разделяя их пробелами.

Закрывает файл.

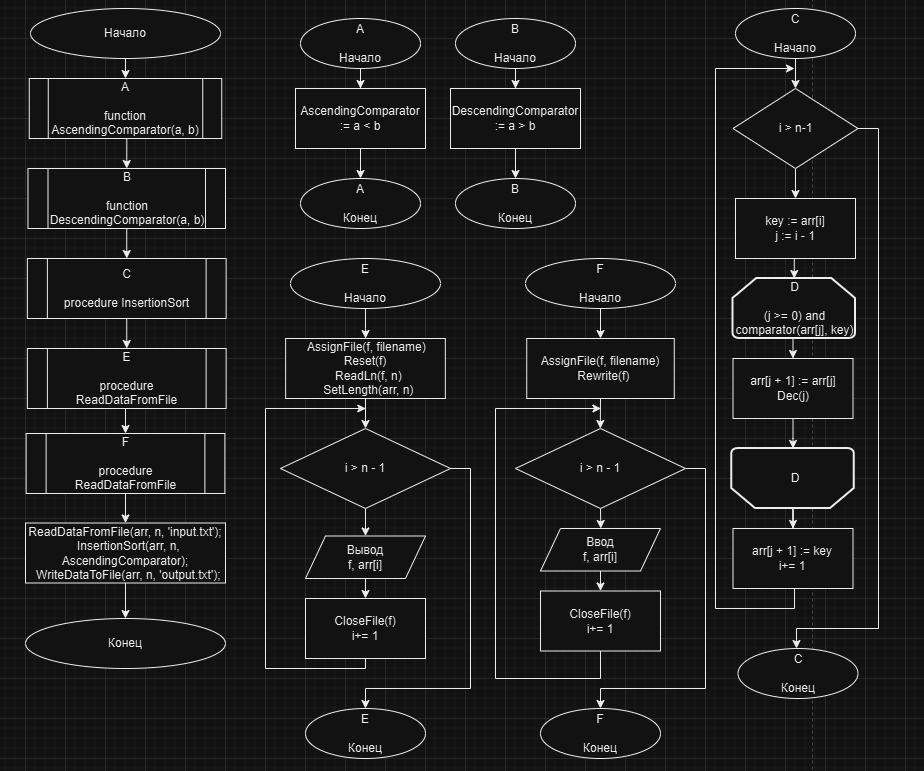
Вызывается процедура ReadDataFromFile для считывания данных из файла input.txt.

Вызывается процедура QuickSort для сортировки массива с использованием функции AscendingComparator (по возрастанию). Здесь можно изменить функцию на DescendingComparator для сортировки по убыванию.

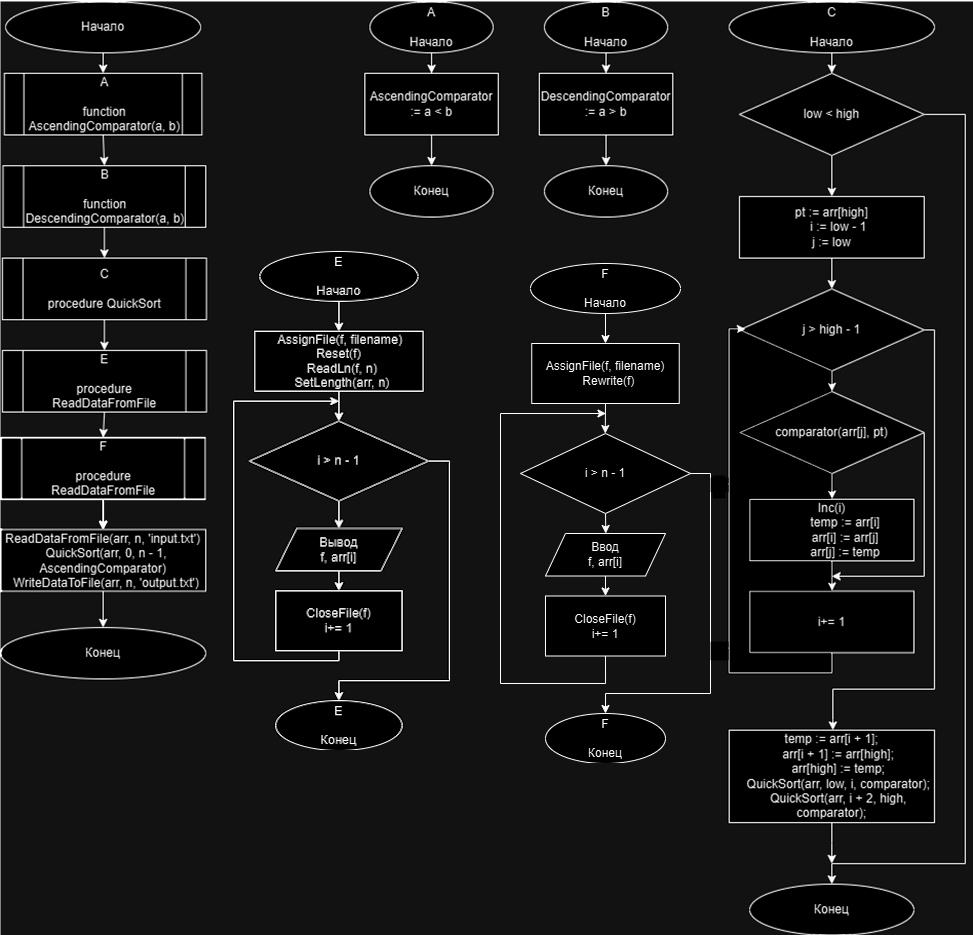
Вызывается процедура WriteDataToFile для записи отсортированных данных в файл output.txt.

**4. Схема алгоритмов.**

**Программа №1**



**Программа №2**



**5. Код программы**

**Программа №1**

**type**

TComparator = **function**(a, b: Integer): Boolean;

**var**

arr: **array of** Integer;

n: Integer;

f: TextFile;

**function** AscendingComparator(a, b: Integer): Boolean;

**begin**

AscendingComparator := a < b;

**end**;

**function** DescendingComparator(a, b: Integer): Boolean;

**begin**

DescendingComparator := a > b;

**end**;

**procedure** InsertionSort(**var** arr: **array of** Integer; n: Integer; comparator: TComparator);

**var**

i, j, key: Integer;

**begin**

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

**begin**

key := arr[i];

j := i - 1;

**while** (j >= 0) **and** comparator(arr[j], key) **do**

**begin**

arr[j + 1] := arr[j];

Dec(j);

**end**;

arr[j + 1] := key;

**end**;

**end**;

**procedure** ReadDataFromFile(**var** arr: **array of** Integer; **var** n: Integer; filename: string);

**var**

i: Integer;

**begin**

AssignFile(f, filename);

Reset(f);

ReadLn(f, n);

SetLength(arr, n);

**for** i := 0 **to** n - 1 **do**

Read(f, arr[i]);

CloseFile(f);

**end**;

**procedure** WriteDataToFile(arr: **array of** Integer; n: Integer; filename: string);

**var**

i: Integer;

**begin**

AssignFile(f, filename);

Rewrite(f);

**for** i := 0 **to** n - 1 **do**

Write(f, arr[i], ' ');

CloseFile(f);

**end**;

**begin**

ReadDataFromFile(arr, n, 'input.txt');

InsertionSort(arr, n, AscendingComparator);

WriteDataToFile(arr, n, 'output.txt');

**end**.

**Программа №2**

**type**

TComparator = **function**(a, b: Integer): Boolean;

**var**

arr: **array of** Integer;

n: Integer;

f: TextFile;

**function** AscendingComparator(a, b: Integer): Boolean;

**begin**

AscendingComparator := a < b;

**end**;

**function** DescendingComparator(a, b: Integer): Boolean;

**begin**

DescendingComparator := a > b;

**end**;

**procedure** QuickSort(**var** arr: **array of** Integer; low, high: Integer; comparator: TComparator);

**var**

pt, i, j, temp: Integer;

**begin**

**if** low < high **then**

**begin**

pivot := arr[high];

i := low - 1;

**for** j := low **to** high - 1 **do**

**begin**

**if** comparator(arr[j], pt) **then**

**begin**

Inc(i);

temp := arr[i];

arr[i] := arr[j];

arr[j] := temp;

**end**;

**end**;

temp := arr[i + 1];

arr[i + 1] := arr[high];

arr[high] := temp;

QuickSort(arr, low, i, comparator);

QuickSort(arr, i + 2, high, comparator);

**end**;

**end**;

**procedure** ReadDataFromFile(**var** arr: **array of** Integer; **var** n: Integer; filename: string);

**var**

i: Integer;

**begin**

AssignFile(f, filename);

Reset(f);

ReadLn(f, n);

SetLength(arr, n);

**for** i := 0 **to** n - 1 **do**

Read(f, arr[i]);

CloseFile(f);

**end**;

**procedure** WriteDataToFile(arr: **array of** Integer; n: Integer; filename: string);

**var**

i: Integer;

**begin**

AssignFile(f, filename);

Rewrite(f);

**for** i := 0 **to** n - 1 **do**

Write(f, arr[i], ' ');

CloseFile(f);

**end**;

**begin**

ReadDataFromFile(arr, n, 'input.txt');

QuickSort(arr, 0, n - 1, AscendingComparator);

WriteDataToFile(arr, n, 'output.txt');

**end**.

**6. Результат выполнения программы**

Программа №1

Ввод.

100

292 973 698 785 330 787 685 544 392 480 468 724 457 159 389 411 560 596 356 554 443 826 639 18 719 6 947 397 841 702 133 912 1000 945 179 289 278 87 704 878 278 646 983 66 192 875 168 788 186 507 568 992 622 347 790 466 334 680 67 325 840 289 703 691 347 556 725 513 980 101 134 474 892 479 277 797 844 574 827 844 839 159 211 335 135 141 290 653 155 713 955 598 21 380 38 989 944 293 719 636 151 655 808 86 374 95 100 144 71 993 623 761 854 223 487 682 79 369 557 207

Вывод.

1000 992 989 983 980 973 955 947 945 944 912 892 878 875 844 844 841 840 839 827 826 797 790 788 787 785 725 724 719 719 713 704 703 702 698 691 685 680 653 646 639 636 622 598 596 574 568 560 556 554 544 513 507 480 479 474 468 466 457 443 411 397 392 389 380 356 347 347 335 334 330 325 293 292 290 289 289 278 278 277 211 192 186 179 168 159 159 155 141 135 134 133 101 87 67 66 38 21 18 6

Программа №2

Ввод.

100

292 973 698 785 330 787 685 544 392 480 468 724 457 159 389 411 560 596 356 554 443 826 639 18 719 6 947 397 841 702 133 912 1000 945 179 289 278 87 704 878 278 646 983 66 192 875 168 788 186 507 568 992 622 347 790 466 334 680 67 325 840 289 703 691 347 556 725 513 980 101 134 474 892 479 277 797 844 574 827 844 839 159 211 335 135 141 290 653 155 713 955 598 21 380 38 989 944 293 719 636 151 655 808 86 374 95 100 144 71 993 623 761 854 223 487 682 79 369 557 207

Вывод.

6 18 21 38 66 67 87 101 133 134 135 141 155 159 159 168 179 186 192 211 277 278 278 289 289 290 292 293 325 330 334 335 347 347 356 380 389 392 397 411 443 457 466 468 474 479 480 507 513 544 554 556 560 568 574 596 598 622 636 639 646 653 680 685 691 698 702 703 704 713 719 719 724 725 785 787 788 790 797 826 827 839 840 841 844 844 875 878 892 912 944 945 947 955 973 980 983 989 992 1000

1. **Вывод**.

В ходе работы получили базовые сведенья о наиболее известных алгоритмах сортировки, изучить принципы работы с текстовыми файлами. Возникли проблемы со схемой алгоритма, при написании и создании ковров, но используя методичку, нашел проблему и исправил ее. Так же написана сам код программы.