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-202-52-00

Ситников Иван Владимирович

Преподаватель:

Кузьминых Ангелина Владимировна

Киров

2023

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

**Задание**

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

**Описание алгоритма.**

Алгоритм основан на двух наиболее известных алгоритмах сортировки (сортировка выбором и быстрой сортировки).

1. Визуализация меню:

Реализуется меню для более удобного ввода параметров сортировки.

1. Сортировка выбором:

Алгоритм сортировки выбором заключается в поиске на необработанном срезе массива или списка минимального значения и в дальнейшем обмене этого значения с первым элементом необработанного среза (поиск минимума и перестановка). На следующем шаге необработанный срез уменьшается на один элемент.

1. Быстрая сортировка:

Алгоритм состоит из трёх шагов:

1)Выбрать элемент из массива. Назовём его опорным.

2)Разбиение: перераспределение элементов в массиве таким образом, что элементы, меньшие опорного, помещаются перед ним, а большие или равные - после.

3)Рекурсивно применить первые два шага к двум подмассивам слева и справа от опорного элемента. Рекурсия не применяется к массиву, в котором только один элемент или отсутствуют элементы.

**Схема алгоритма**

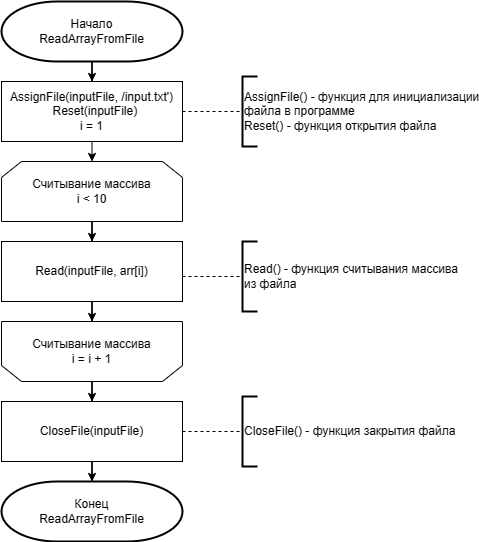
****

Рисунок 1.1 – Процедура ReadArrayFromFile

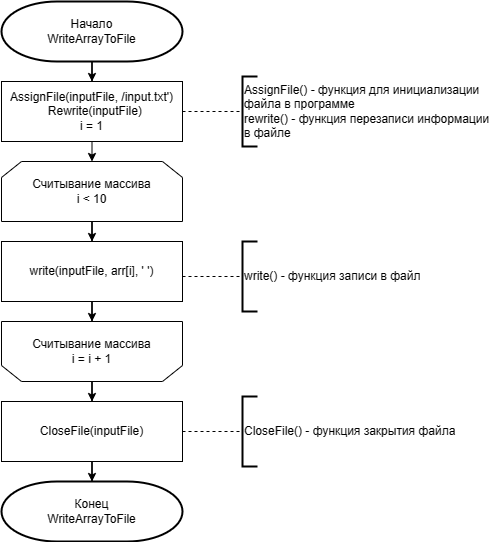


Рисунок 1.2 – Процедура WriteArrayToFile

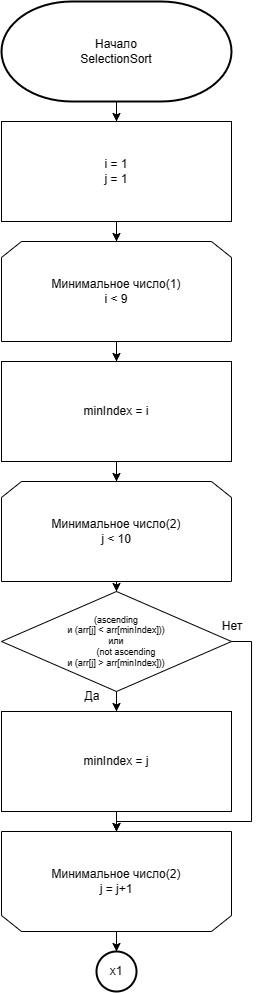


Рисунок 1.3.1 – Процедура SelectionSort

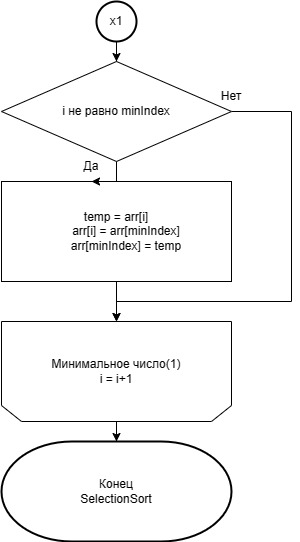


Рисунок 1.3.2 – Процедура SelectionSort

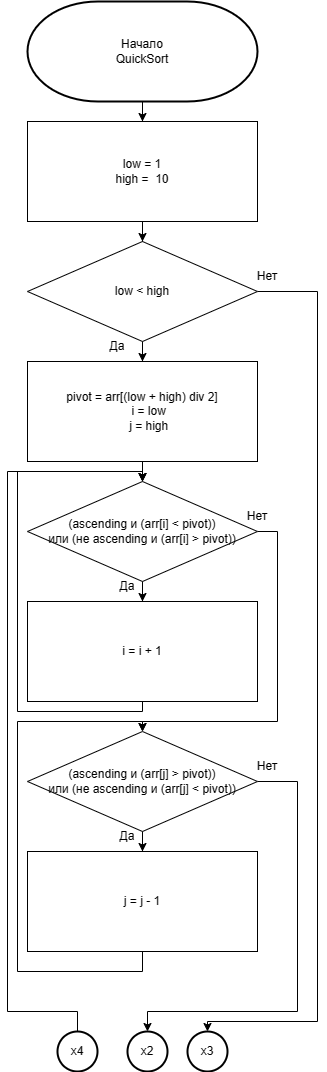


Рисунок 1.4.1 – Процедура QuickSort

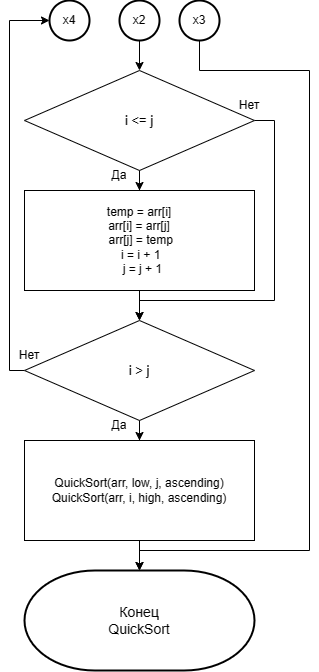


Рисунок 1.4.2 – Процедура QuickSort

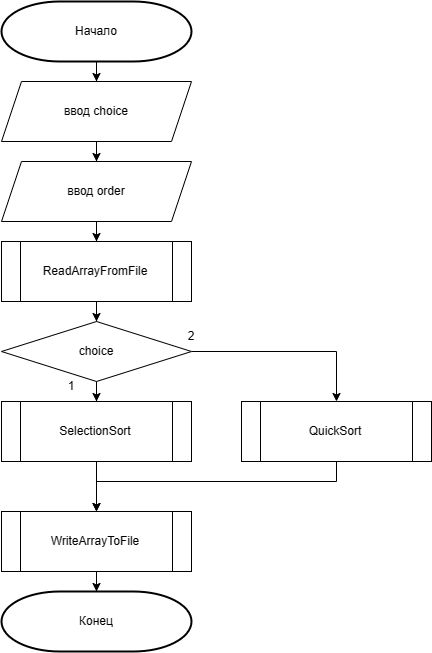
****

Рисунок 1.5 – Тело программы

**Код программы**

program choice\_sort;

uses Crt;

const

ARRAY\_SIZE = 10;

type

IntArray = array [1..ARRAY\_SIZE] of Integer;

procedure ReadArrayFromFile(var arr: IntArray);

var

inputFile: TextFile;

i: Integer;

begin

AssignFile(inputFile, 'E:/лабы/Основы алгоритмизации и программирования/input.txt');

Reset(inputFile);

for i := 1 to ARRAY\_SIZE do

Read(inputFile, arr[i]);

CloseFile(inputFile);

end;

procedure WriteArrayToFile(const arr: IntArray);

var

outputFile: TextFile;

i: Integer;

begin

AssignFile(outputFile, 'E:/лабы/Основы алгоритмизации и программирования/output.txt');

Rewrite(outputFile);

for i := 1 to ARRAY\_SIZE do

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

CloseFile(outputFile);

end;

procedure SelectionSort(var arr: IntArray; ascending: Boolean);

var

i, j, minIndex, maxIndex, temp: Integer;

begin

for i := 1 to ARRAY\_SIZE - 1 do

begin

minIndex := i;

for j := i + 1 to ARRAY\_SIZE do

begin

if (ascending and (arr[j] < arr[minIndex])) or

(not ascending and (arr[j] > arr[minIndex])) then

minIndex := j;

end;

if i <> minIndex then

begin

temp := arr[i];

arr[i] := arr[minIndex];

arr[minIndex] := temp;

end;

end;

end;

procedure QuickSort(var arr: IntArray; low, high: Integer; ascending: Boolean);

var

i, j, pivot, temp: Integer;

begin

if low < high then

begin

pivot := arr[(low + high) div 2];

i := low;

j := high;

repeat

while (ascending and (arr[i] < pivot)) or

(not ascending and (arr[i] > pivot)) do

Inc(i);

while (ascending and (arr[j] > pivot)) or

(not ascending and (arr[j] < pivot)) do

Dec(j);

if i <= j then

begin

temp := arr[i];

arr[i] := arr[j];

arr[j] := temp;

Inc(i);

Dec(j);

end;

until i > j;

QuickSort(arr, low, j, ascending);

QuickSort(arr, i, high, ascending);

end;

end;

var

data: IntArray;

choice, order,ch: Char;

begin

repeat

ClrScr;

WriteLn('Выберите алгоритм сортировки:');

WriteLn('1. Сортировка выбором');

WriteLn('2. Быстрая сортировка');

ReadLn(choice);

ClrScr;

WriteLn('Выберите порядок сортировки:');

WriteLn('A. По возрастанию');

WriteLn('D. По убыванию');

ReadLn(order);

ClrScr;

ReadArrayFromFile(data);

case choice of

'1': SelectionSort(data, order = 'A');

'2': QuickSort(data, 1, ARRAY\_SIZE, order = 'A');

end;

WriteArrayToFile(data);

WriteLn('Результат успешно записан в файл output.txt.');

WriteLn('Нажмите Enter, чтобы задать новые параметры');

ch:= ReadKey;

until ch = #27;

end.

**Результаты выполнения программы**

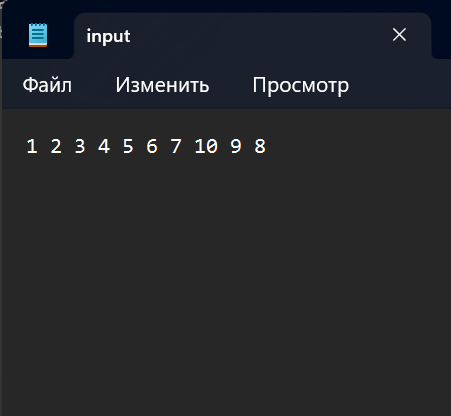
****

Рисунок 2.1 – Исходный массив

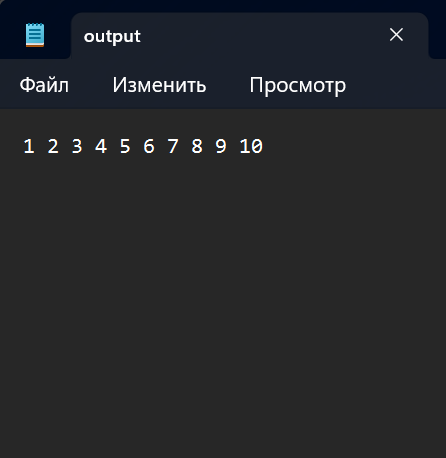


Рисунок 2.2 – Отсортированный по возрастанию массив

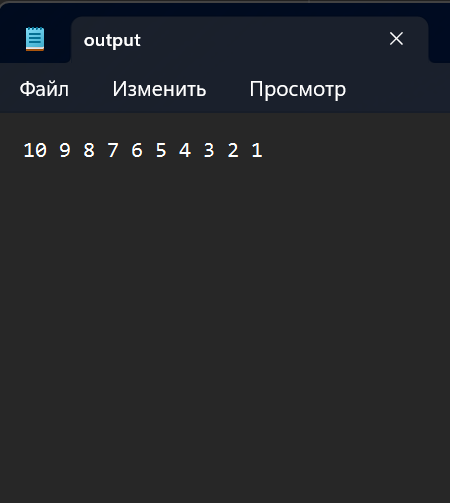
****

Рисунок 2.3 – Отсортированный по убыванию массив

**Вывод**

Программы и алгоритмы, разработанные в ходе работы, представляют собой пример работы с файлами на языке программирования Pascal.

В ходе написания программы для сортировки массива были изучены наиболее известные алгоритмы сортировки и реализованы в языке программирования Pascal. Был реализован алгоритм, благодаря которому тестовый набор данных поступает в программу из файла. В процессе выполнения программы, присутствует возможность выбора алгоритма сортировки массива, с помощью быстрой сортировки или сортировки выбором. А также присутствует возможность выбрать вид сортировки: по убыванию или по возрастанию.

В итоге, была разработана программа, сортирующая массив, находящийся в файле, записывая результат в другой файл. А также была реализована функция выбора компаратора сортировки и минимальный пользовательский интерфейс.