**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

Долгих Дмитрий Константинович

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 9

1. Реализовать сортировку данных с помощью вставок.
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**

Для выполнения задания был использован модуль *CRT*, с помощью которого выводилась консоль на экран пользователя. Перемещение по консоли сделано с помощью оператора выбора *case…of*.

Для работы с текстовыми файлами использовались процедуры *assign* (связывает файловую переменную с файлом на диске), *reset* (открывает файл на чтение), *rewrite* (открывает текстовый файл на запись), *close* (закрывает файл).

Для сортировки массивов из текстовых файлов были использованы процедуры сортировок вставками и быстрым алгоритмом.

Сортировка вставками предполагает такой алгоритм сортировки, при котором элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов.

Сортировка быстрым алгоритмом или быстрая сортировка предполагает такой алгоритм, в котором из массива выбирается один элемент, называемый опорным. Затем все остальные элементы сравниваются с опорным и подразделяются на 3 категории: «меньше опорного», «равные опорному» и «большие опорного», для отрезков «меньших» и «больших» значений рекурсивно выполняется та же последовательность операций, если длина отрезка больше единицы.

**Схема алгоритма с комментариями**

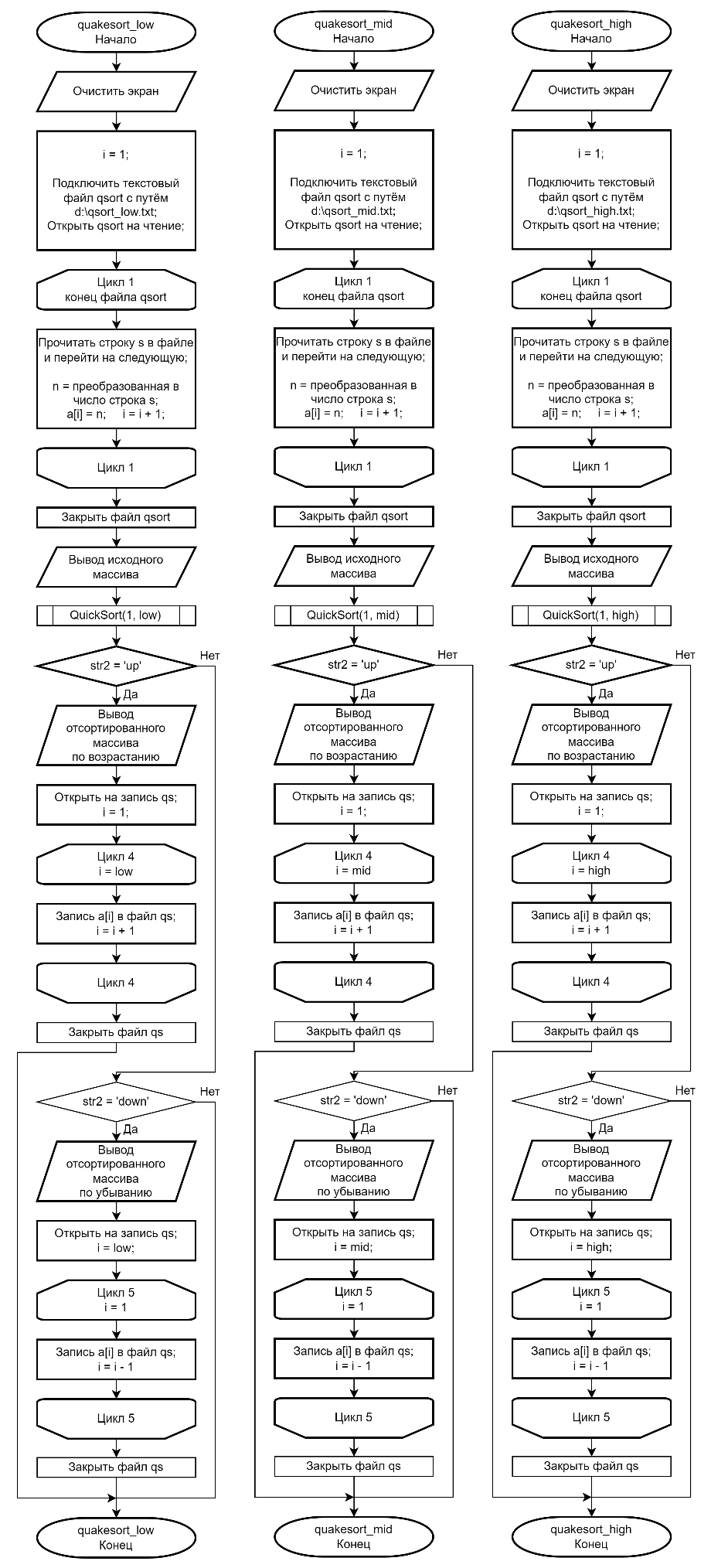


Рис. 1 «Схема сортировки быстрым алгоритмом для файлов разного объема»

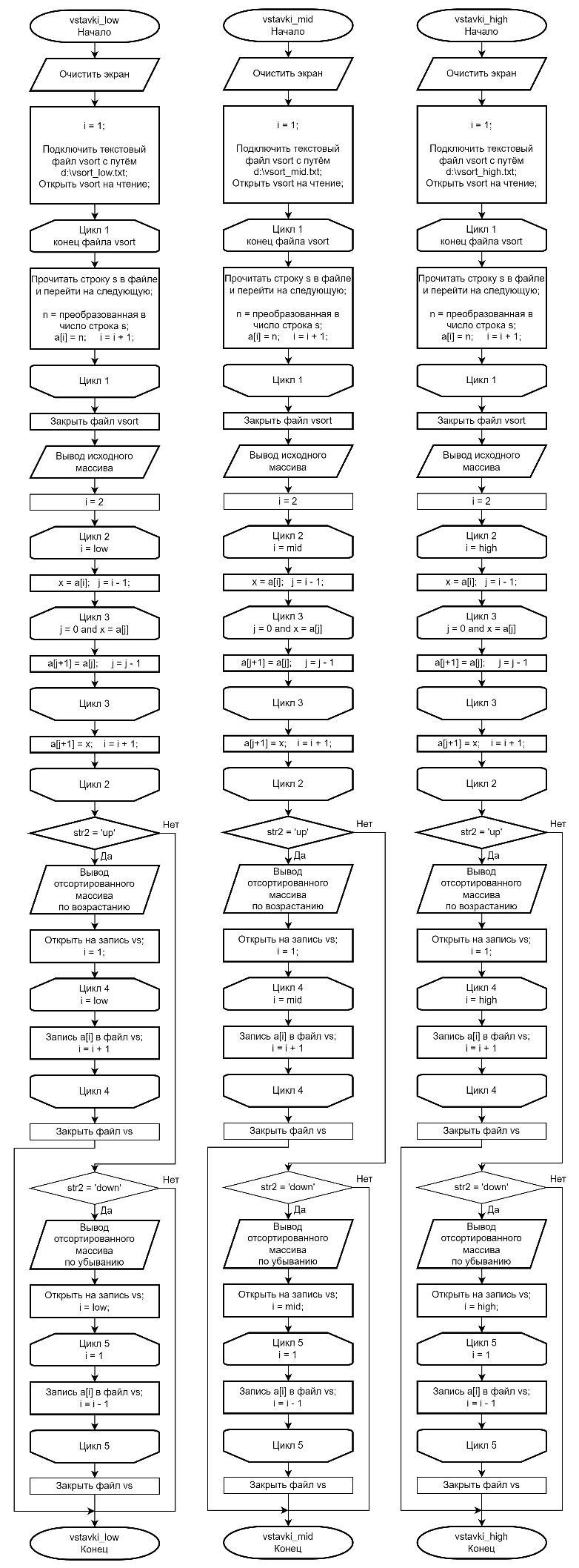


Рис. 2 «Схема сортировки вставками для файлов разного объема»

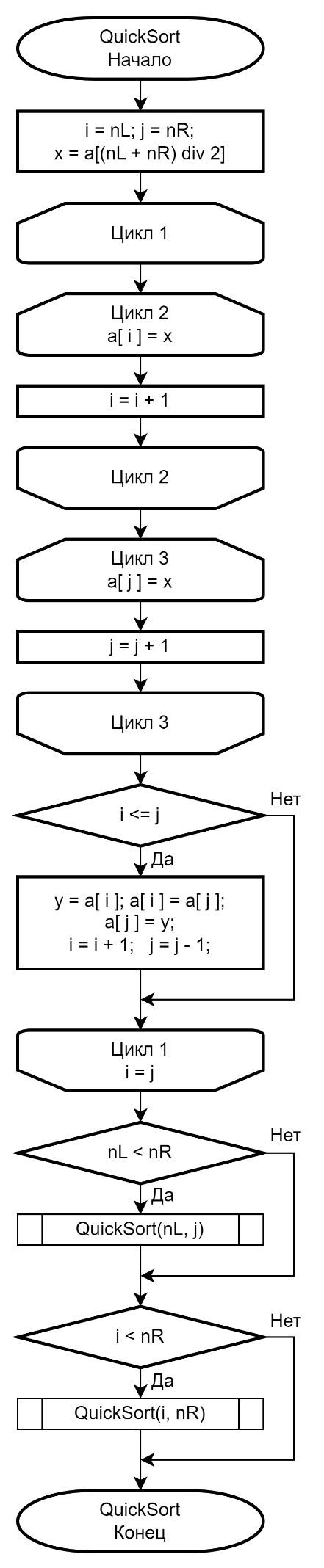


Рис. 3 «Схема алгоритма быстрой сортировки»



Рис. 4 «Схема процедуры вывода меню на экран»

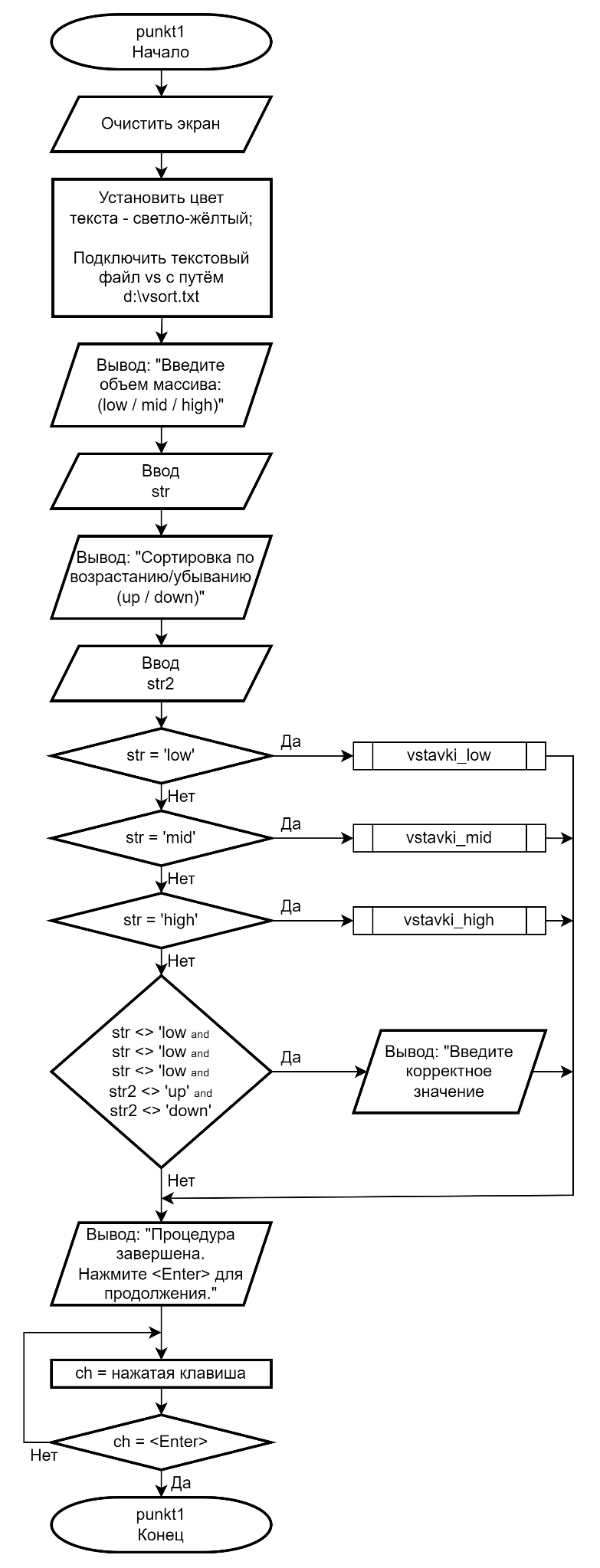


Рис. 5 «Схема алгоритма выбора объема и компаратора сортировки вставками»

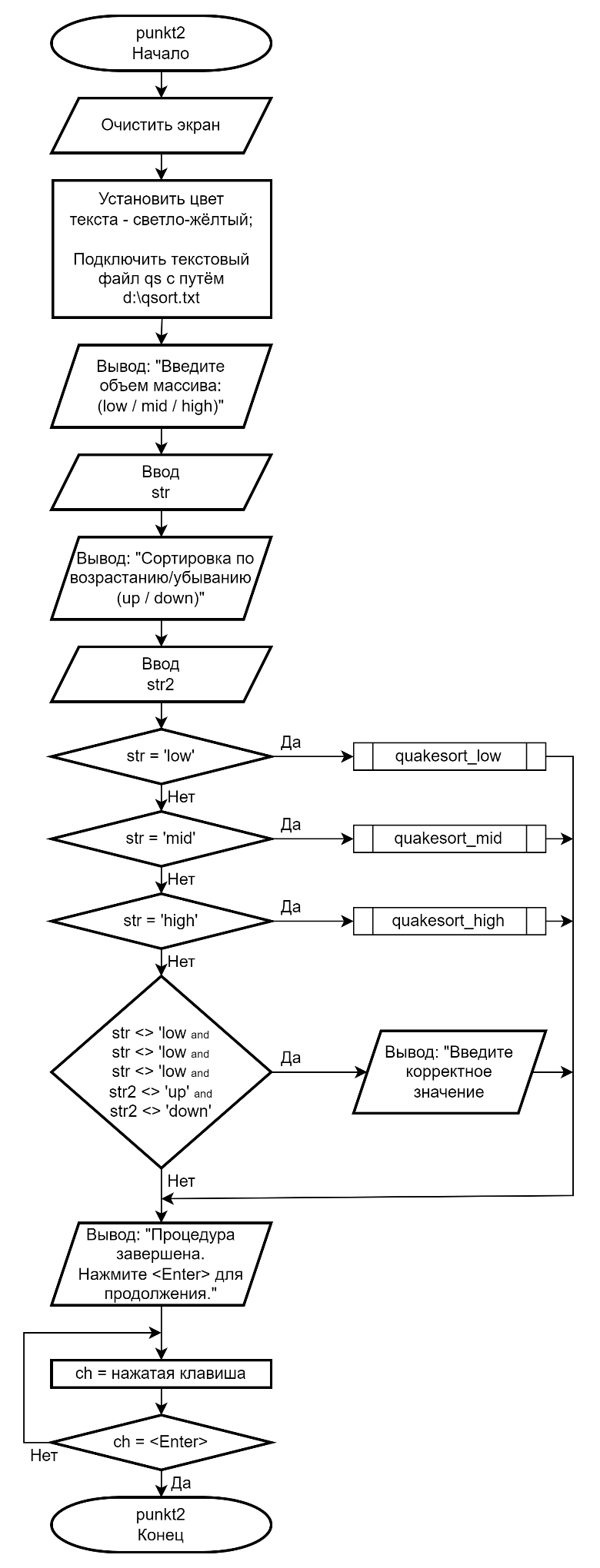


Рис. 6 «Схема алгоритма выбора объема и компаратора сортировки быстрым алгоритмом»

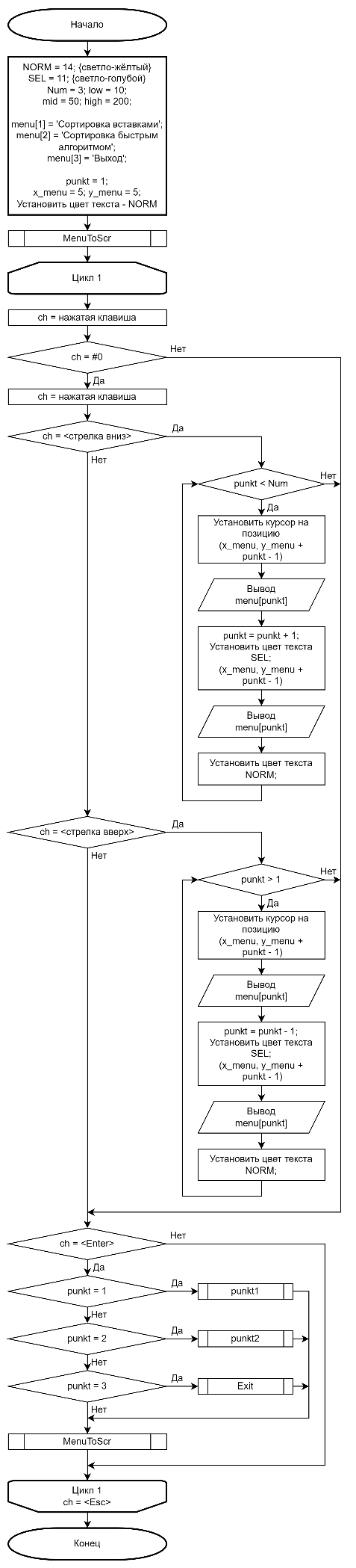


Рис. 7 «Схема алгоритма главной программы»

**Код программы**

**program** dkr5;

**uses** Crt;

**const**

NORM = 14; { цвет невыделеного пункта }

SEL = 11; { цвет выделенного пункта }

Num = 3;

low = 10;

mid = 50;

high = 200;

**var**

menu: **array**[1..Num] **of** string[36];{ названия пунктов меню }

punkt: integer;

ch: char; str, str2: string;

x\_menu, y\_menu: byte;

qsort, qs, vsort, vs: text;

**procedure** vstavki\_low;

**var**

a: **array**[1..low] **of** integer;

x, i, j, n: integer;

s: string;

**begin**

ClrScr;

assign(vsort, 'd:\vsort\_low.txt');

i:= 1;

reset(vsort);

**while not** eof(vsort) **do**

**begin**

readln(vsort, s);

n:= StrToInt(s);

a[i]:= n;

i+=1;

**end**;

close(vsort);

writeln('Исходный массив: ', a);

**for** i := 2 **to** low **do**

**begin**

x := a[i];

j := i - 1;

**while** (j > 0) **and** (x < a[j]) **do**

**begin**

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

j := j - 1;

**end**;

a[j + 1] := x;

**end**;

**if** str2 = 'up' **then begin**

writeln('Отсортированный массив: ', a);

rewrite(vs);

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

writeln(vs, a[i]);

close(vs);

**end**;

**if** str2 = 'down' **then begin**

write('Отсортированный массив: [');

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

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

write(']');

rewrite(vs);

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

writeln(vs, a[i]);

close(vs);

**end**;

**end**;

**procedure** vstavki\_mid;

**var**

a: **array**[1..mid] **of** integer;

x, i, j, n: integer;

s: string;

**begin**

ClrScr;

assign(vsort, 'd:\vsort\_mid.txt');

i:= 1;

reset(vsort);

**while not** eof(vsort) **do**

**begin**

readln(vsort, s);

n:= StrToInt(s);

a[i]:= n;

i+=1;

**end**;

close(vsort);

writeln('Исходный массив: ', a);

**for** i := 2 **to** mid **do**

**begin**

x := a[i];

j := i - 1;

**while** (j > 0) **and** (x < a[j]) **do**

**begin**

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

j := j - 1;

**end**;

a[j + 1] := x;

**end**;

**if** str2 = 'up' **then begin**

writeln('Отсортированный массив: ', a);

rewrite(vs);

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

writeln(vs, a[i]);

close(vs);

**end**;

**if** str2 = 'down' **then begin**

write('Отсортированный массив: [');

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

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

write(']');

rewrite(vs);

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

writeln(vs, a[i]);

close(vs);

**end**;

**end**;

**procedure** vstavki\_high;

**var**

a: **array**[1..high] **of** integer;

x, i, j, n: integer;

s: string;

**begin**

ClrScr;

assign(vsort, 'd:\vsort\_high.txt');

i:= 1;

reset(vsort);

rewrite(vs);

**while not** eof(vsort) **do**

**begin**

readln(vsort, s);

n:= StrToInt(s);

a[i]:= n;

writeln(vs, n);

i+=1;

**end**;

close(vs);

close(vsort);

writeln('Исходный массив: ', a);

**for** i := 2 **to** high **do**

**begin**

x := a[i];

j := i - 1;

**while** (j > 0) **and** (x < a[j]) **do**

**begin**

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

j := j - 1;

**end**;

a[j + 1] := x;

**end**;

**if** str2 = 'up' **then begin**

writeln('Отсортированный массив: ', a);

rewrite(vs);

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

writeln(vs, a[i]);

close(vs);

**end**;

**if** str2 = 'down' **then begin**

write('Отсортированный массив: [');

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

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

write(']');

rewrite(vs);

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

writeln(vs, a[i]);

close(vs);

**end**;

**end**;

**procedure** quakesort\_low;

**var**

a: **array**[1..low] **of** integer;

n, i: integer;

s: string;

**procedure** QuickSort(nL, nR: Integer);

**var**

i, j, x, y: integer;

**begin**

i := nL; j := nR;

x := a[(nL+nR) **div** 2];

**repeat**

**while** (a[i] < x) **do** inc(i);

**while** (x < a[j]) **do** dec(j);

**if** (i <= j) **then**

**begin**

y:= a[i];

a[i]:= a[j];

a[j]:= y;

inc(i);

dec(j);

**end**;

**until** (i > j);

**if** (nL < nR) **then** QuickSort(nL, j);

**if** (i < nR) **then** QuickSort(i, nR);

**end**;

**begin**

ClrScr;

assign(qsort, 'd:\qsort\_low.txt');

i:= 1;

reset(qsort);

**while not** eof(qsort) **do**

**begin**

readln(qsort, s);

n:= StrToInt(s);

a[i]:= n;

i+=1;

**end**;

close(qsort);

writeln('Исходный массив: ', a);

QuickSort(1, low);

**if** str2 = 'up' **then begin**

writeln('Отсортированный массив: ', a);

rewrite(qs);

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

writeln(qs, a[i]);

close(qs);

**end**;

**if** str2 = 'down' **then begin**

write('Отсортированный массив: [');

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

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

write(']');

rewrite(qs);

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

writeln(qs, a[i]);

close(qs);

**end**;

**end**;

**procedure** quakesort\_mid;

**var**

a: **array**[1..mid] **of** integer;

n, i: integer;

s: string;

**procedure** QuickSort(nL, nR: Integer);

**var**

i, j, x, y: integer;

**begin**

i := nL; j := nR;

x := a[(nL+nR) **div** 2];

**repeat**

**while** (a[i] < x) **do** inc(i);

**while** (x < a[j]) **do** dec(j);

**if** (i <= j) **then**

**begin**

y:= a[i];

a[i]:= a[j];

a[j]:= y;

inc(i);

dec(j);

**end**;

**until** (i > j);

**if** (nL < nR) **then** QuickSort(nL, j);

**if** (i < nR) **then** QuickSort(i, nR);

**end**;

**begin**

ClrScr;

assign(qsort, 'd:\qsort\_mid.txt');

i:= 1;

reset(qsort);

**while not** eof(qsort) **do**

**begin**

readln(qsort, s);

n:= StrToInt(s);

a[i]:= n;

i+=1;

**end**;

close(qsort);

writeln('Исходный массив: ', a);

QuickSort(1, mid);

**if** str2 = 'up' **then begin**

writeln('Отсортированный массив: ', a);

rewrite(qs);

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

writeln(qs, a[i]);

close(qs);

**end**;

**if** str2 = 'down' **then begin**

write('Отсортированный массив: [');

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

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

write(']');

rewrite(qs);

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

writeln(qs, a[i]);

close(qs);

**end**;

**end**;

**procedure** quakesort\_high;

**var**

a: **array**[1..high] **of** integer;

n, i: integer;

s: string;

**procedure** QuickSort(nL, nR: Integer);

**var**

i, j, x, y: integer;

**begin**

i := nL; j := nR;

x := a[(nL+nR) **div** 2];

**repeat**

**while** (a[i] < x) **do** inc(i);

**while** (x < a[j]) **do** dec(j);

**if** (i <= j) **then**

**begin**

y:= a[i];

a[i]:= a[j];

a[j]:= y;

inc(i);

dec(j);

**end**;

**until** (i > j);

**if** (nL < nR) **then** QuickSort(nL, j);

**if** (i < nR) **then** QuickSort(i, nR);

**end**;

**begin**

ClrScr;

assign(qsort, 'd:\qsort\_high.txt');

i:= 1;

reset(qsort);

**while not** eof(qsort) **do**

**begin**

readln(qsort, s);

n:= StrToInt(s);

a[i]:= n;

i+=1;

**end**;

close(qsort);

writeln('Исходный массив: ', a);

QuickSort(1, high);

**if** str2 = 'up' **then begin**

writeln('Отсортированный массив: ', a);

rewrite(qs);

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

writeln(qs, a[i]);

close(qs);

**end**;

**if** str2 = 'down' **then begin**

write('Отсортированный массив: [');

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

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

write(']');

rewrite(qs);

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

writeln(qs, a[i]);

close(qs);

**end**;

**end**;

**procedure** punkt1;

**begin**

ClrScr;

assign(vs, 'd:\vsort.txt');

writeln('Введите объем массива: (low / mid / high)');

readln(str);

writeln('Сортировка по возрастанию/убыванию (up / down)');

readln(str2);

**if** str = 'low' **then** vstavki\_low;

**if** str = 'mid' **then** vstavki\_mid;

**if** str = 'high' **then** vstavki\_high;

**if** (str <> 'low') **and** (str <> 'mid') **and** (str <> 'high') **and** (str2 <> 'up') **and** (str2 <> 'down') **then** writeln('Введите корректное значение');

writeln;

writeln('Процедура завершена. Нажмите <Enter> для продолжения.');

**repeat**

ch := readkey;

**until** ch = #13;

**end**;

**procedure** punkt2;

**begin**

ClrScr;

assign(qs, 'd:\qsort.txt');

writeln('Введите объем массива: (low / mid / high)');

readln(str);

writeln('Сортировка по возрастанию/убыванию (up / down)');

readln(str2);

**if** str = 'low' **then** quakesort\_low;

**if** str = 'mid' **then** quakesort\_mid;

**if** str = 'high' **then** quakesort\_high;

**if** (str <> 'low') **and** (str <> 'mid') **and** (str <> 'high') **and** (str2 <> 'up') **and** (str2 <> 'down') **then** writeln('Введите корректное значение');

writeln;

writeln('Процедура завершена. Нажмите <Enter> для продолжения.');

**repeat**

ch := readkey;

**until** ch = #13;

**end**;

**procedure** MenuToScr;{ вывод меню на экран }

**var**

i: integer;

**begin**

ClrScr;

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

**begin**

GoToXY(x\_menu, y\_menu + i - 1);

write(menu[i]);

**end**;

TextColor(SEL);

GoToXY(x\_menu, y\_menu + punkt - 1);

write(menu[punkt]);{ выделим строку меню }

TextColor(NORM);

**end**;

**begin**

ClrScr;

menu[1] := ' Сортировка вставками ';

menu[2] := ' Сортировка быстрым алгоритмом ';

menu[3] := ' Выход ';

punkt := 1; x\_menu := 5; y\_menu := 5;

TextColor(NORM);

MenuToScr;

**repeat**

ch := ReadKey;

**if** ch = #0 **then begin**

ch := ReadKey;

**case** ch **of**

#40:{ стрелка вниз }

**if** punkt < Num **then begin**

GoToXY(x\_menu, y\_menu + punkt - 1); write(menu[punkt]);

punkt := punkt + 1;

TextColor(SEL);

GoToXY(x\_menu, y\_menu + punkt - 1); write(menu[punkt]);

TextColor(NORM);

**end**;

#38:{ стрелка вверх }

**if** punkt > 1 **then begin**

GoToXY(x\_menu, y\_menu + punkt - 1); write(menu[punkt]);

punkt := punkt - 1;

TextColor(SEL);

GoToXY(x\_menu, y\_menu + punkt - 1); write(menu[punkt]);

TextColor(NORM);

**end**;

**end**;

**end**

**else**

**if** ch = #13 **then begin**{ нажата клавиша <Enter> }

**case** punkt **of**

1: punkt1;

2: punkt2;

3: **exit** { выход }

**end**;

MenuToScr;

**end**;

**until** ch = #27;{ 27 - код <Esc> }

**end**.

**Результат выполнения программы**

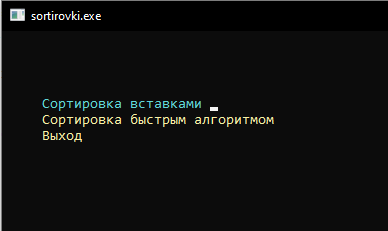


Рис. 8 «Case-меню»

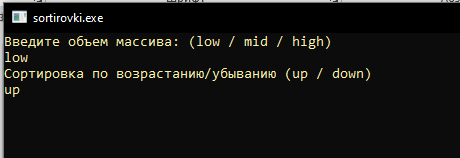


Рис. 9 «Выбор объема массива и компаратора»

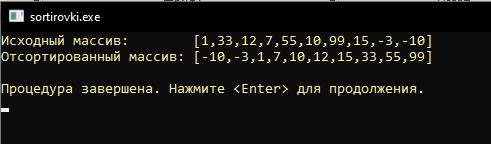


Рис. 10 «Результат сортировки»

**Вывод**

Данная работа прошла успешно, в ней мы получили базовые сведения о наиболее известных алгоритмах сортировки, изучили принципы работы с текстовыми файлами.

Во время выполнения работы возникли трудности с написанием кода и построением схемы алгоритма. Одна из таких проблем:

Проблема с динамическим массивом. В алгоритме под каждый файл используется отдельный статический массив. Используя динамический массив появилась бы возможность поместить всё в одну процедуру и значительно уменьшить код, но после поиска и нескольких неудачных попыток решения данной проблемы было принято решение оставить несколько процедур со статическими массивами.

В итоге, работа улучшила знания в языке программирования Pascal, научила работать с текстовыми файлами в Pascal, а также различными алгоритмами сортировок. Основная цель работы была достигнута, все задачи выполнены, все ошибки и недочёты исправлены.