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

Лаптев Владимир Антонович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

**Цель работы**

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

**Формулировка задания (с вариантом)**

Вариант 12

Задание:

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).

**Описание алгоритма**

Были написаны 2 логические функции, которые встраиваются в функции сортировок, и в зависимости от их значения выводится отсортированный либо по возрастанию, либо по убыванию массив.

Были написаны процедуры сортировки подсчётом и быстрой сортировки.

**Сортировка подсчётом** — [алгоритм сортировки](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B8), в котором используется диапазон чисел сортируемого [массива](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) ([списка](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0))) для подсчёта совпадающих элементов.

**Быстрая сортировка**

На очередном шаге выбирается опорный элемент — им может быть любой элемент массива.

Все остальные элементы массива сравниваются с опорным и те, которые меньше него, ставятся слева от него, а которые больше или равны — справа.

Для двух получившихся блоков массива (меньше опорного, и больше либо равны опорному) производится точно такая же операция — выделяется опорный элемент и всё идёт точно так же, пока в блоке не останется один элемент.

Была написана процедура считывания массива из файла, также была написана процедура, которая выводит массив. Также была написана процедура, которая записывает массив в файл.

**Схема алгоритма с комментариями**

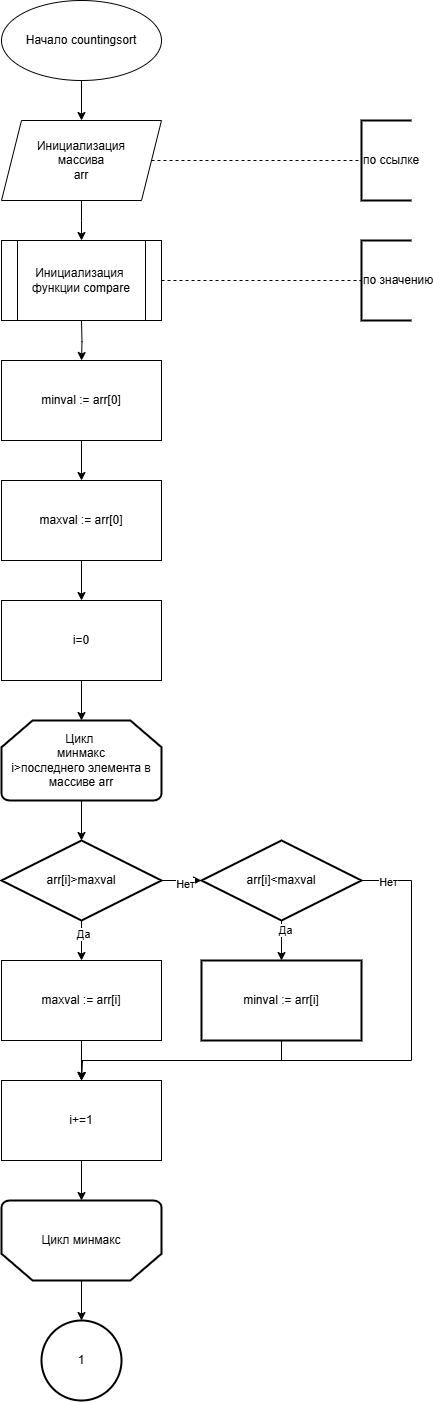
****

Рисунок 1 — Алгоритм решения программы

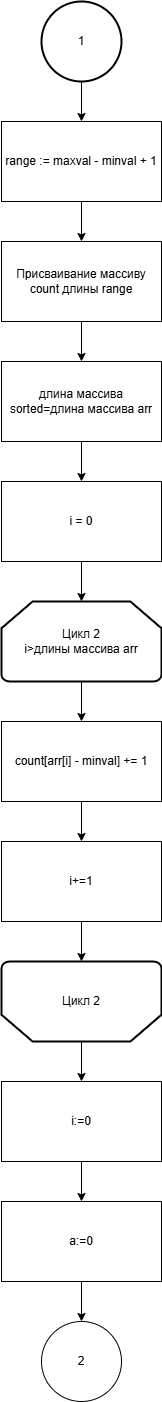
****

Рисунок 2 — Алгоритм решения программы

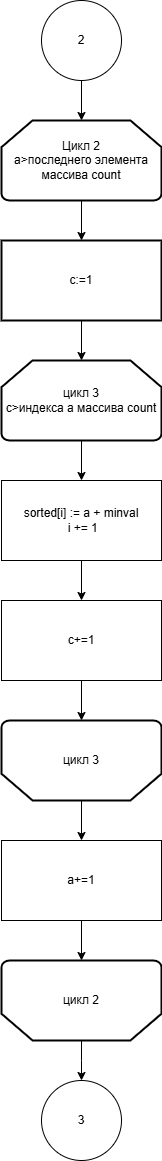


Рисунок 3 — Алгоритм программы

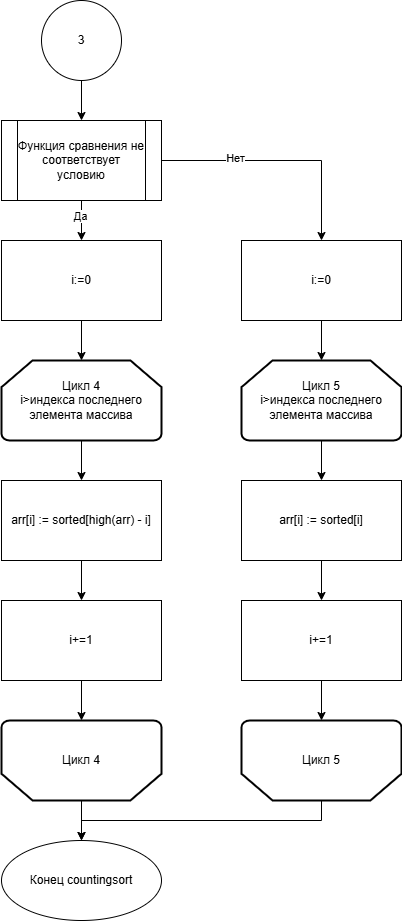
**

Рисунок 4 — Алгоритм программы

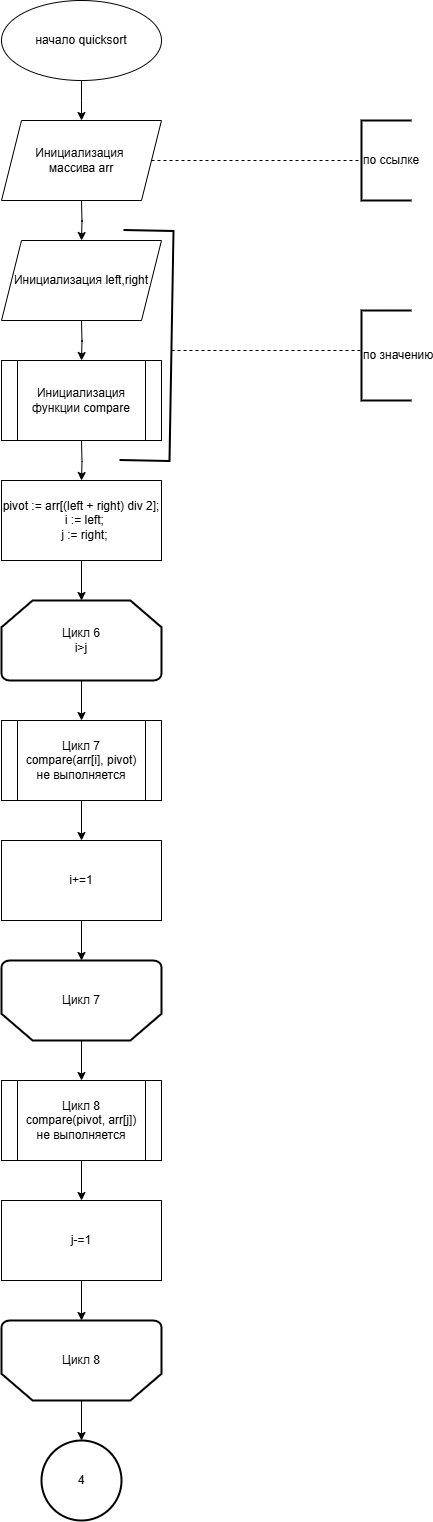
**

Рисунок 5 — Алгоритм программы

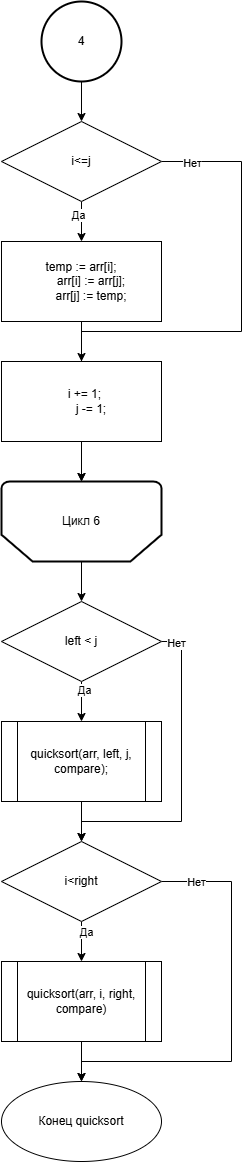


Рисунок 6 — Алгоритм программы

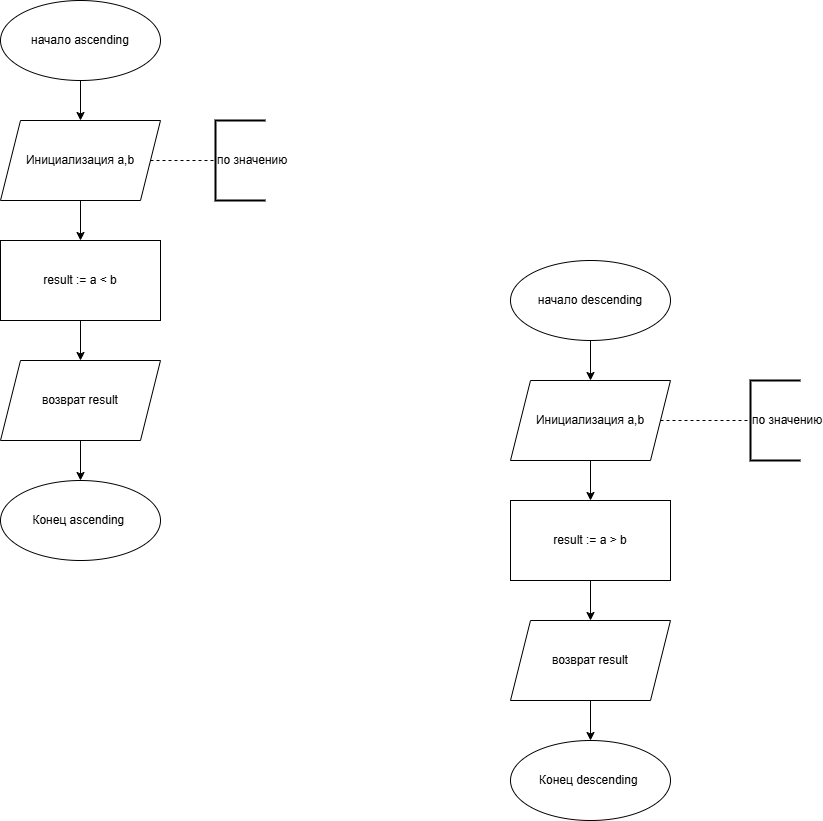
**

Рисунок 7 — Алгоритм программы

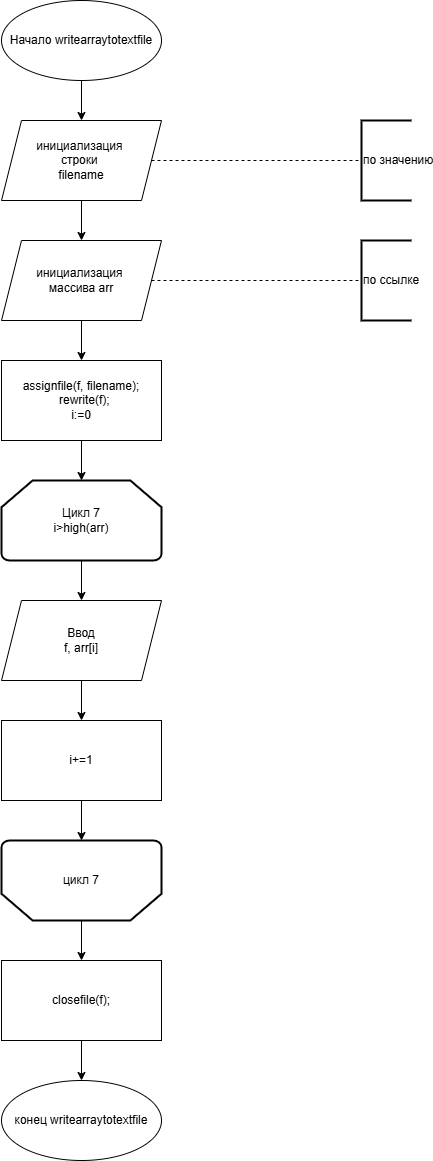


Рисунок 8 — Алгоритм программы

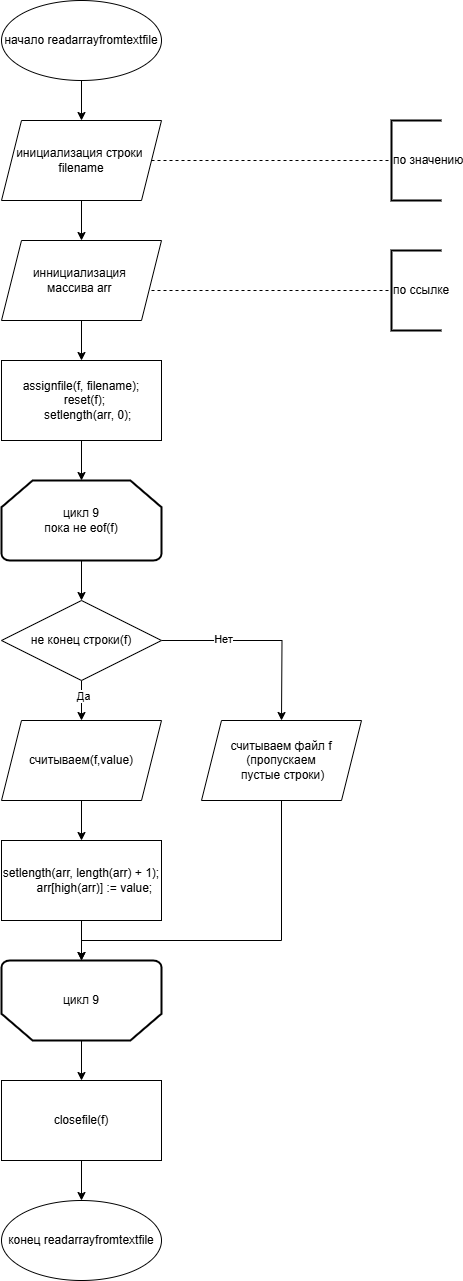


Рисунок 9 — Алгоритм программы

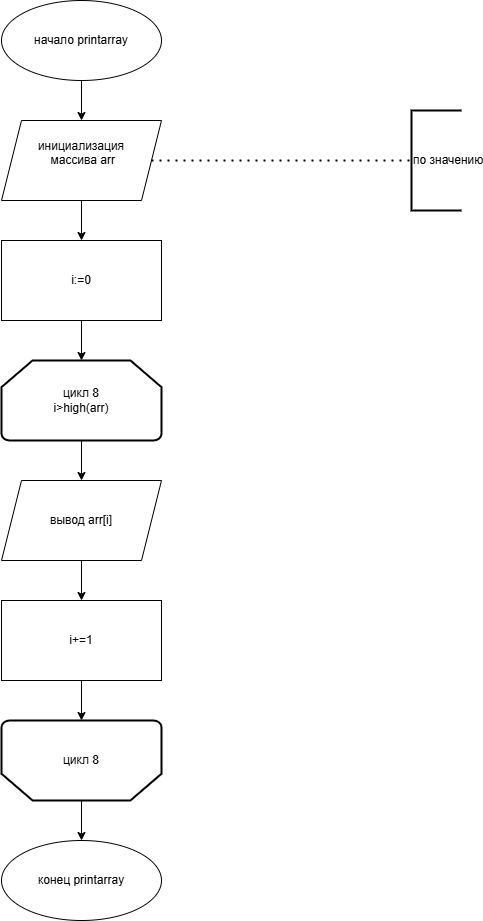


Рисунок 10 — Алгоритм программы

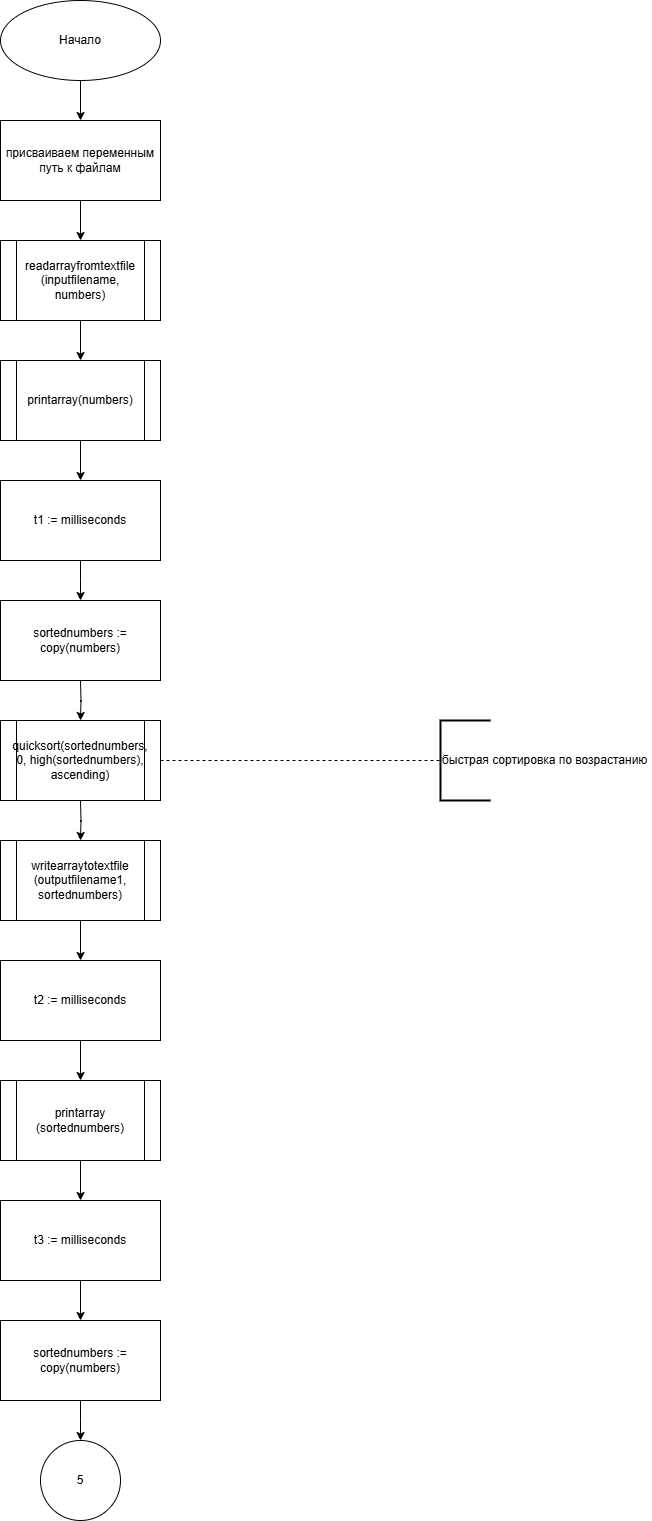


Рисунок 11 — Алгоритм программы

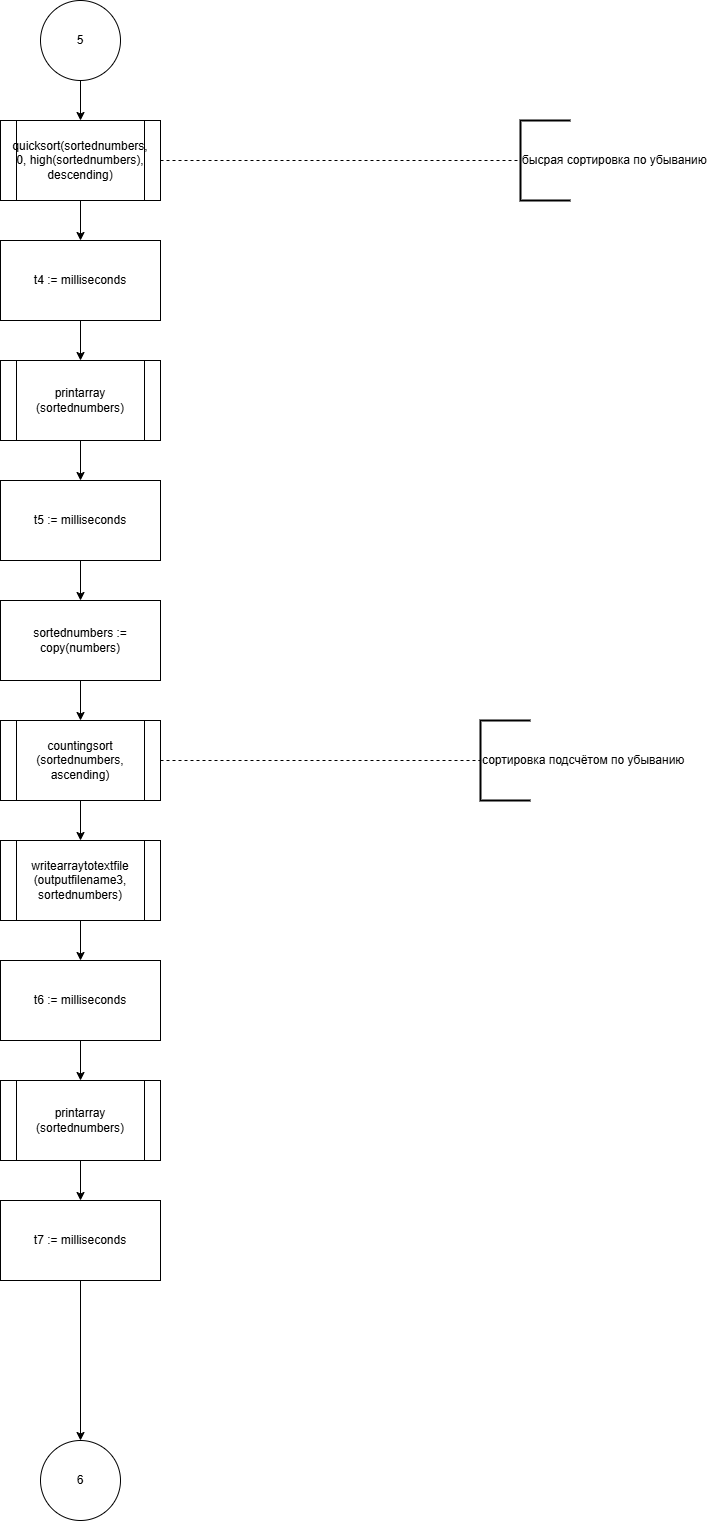


Рисунок 12 — Алгоритм программы

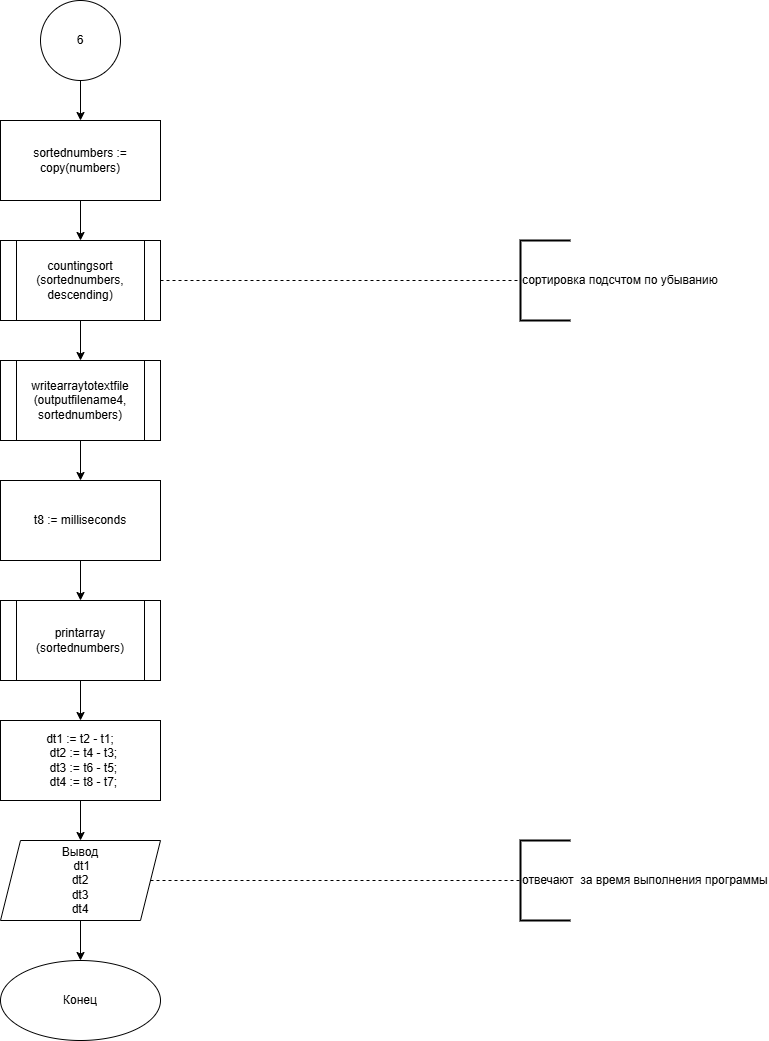


Рисунок 13 — Алгоритм программы

**Код программы**

program qq;

type

tintarray = array of integer;

tcomparefunc = function(a, b: integer): boolean;

function ascending(a, b: integer): boolean;

begin

result := a < b;

end;

function descending(a, b: integer): boolean;

begin

result := a > b;

end;

procedure countingsort(var arr: tintarray; compare: tcomparefunc);

var

maxval, minval, range, i, a, c: integer;

count, sorted: array of integer;

begin

minval := arr[0];

maxval := arr[0];

for i := 1 to high(arr) do

begin

if arr[i] > maxval then

maxval := arr[i];

if arr[i] < minval then

minval := arr[i];

end;

range := maxval - minval + 1;

setlength(count, range);

setlength(sorted, length(arr));

for i := 0 to high(arr) do

count[arr[i] - minval] += 1;

i := 0;

for a := 0 to high(count) do

for c := 1 to count[a] do

begin

sorted[i] := a + minval;

i += 1;

end;

if not compare(1, 0) then

begin

for i := 0 to high(arr) do

arr[i] := sorted[high(arr) - i];

end

else

for i := 0 to high(arr) do

arr[i] := sorted[i];

end;

procedure quicksort(var arr: tintarray; left, right: integer; compare: tcomparefunc);

var

pivot, i, j, temp: integer;

begin

pivot := arr[(left + right) div 2];

i := left;

j := right;

while i <= j do

begin

while compare(arr[i], pivot) do

begin

i += 1;

end;

while compare(pivot, arr[j]) do

begin

j -= 1;

end;

if i <= j then

begin

temp := arr[i];

arr[i] := arr[j];

arr[j] := temp;

i += 1;

j -= 1;

end;

end;

if left < j then

quicksort(arr, left, j, compare);

if i < right then

quicksort(arr, i, right, compare);

end;

procedure writearraytotextfile(filename: string; var arr: tintarray);

var

f: text;

i: integer;

begin

assignfile(f, filename);

rewrite(f);

for i := 0 to high(arr) do

writeln(f, arr[i]);

closefile(f);

end;

procedure readarrayfromtextfile(filename: string; var arr: tintarray);

var

f: text;

value: integer;

begin

assignfile(f, filename);

reset(f);

setlength(arr, 0);

while not eof(f) do

begin

if not eoln(f) then

begin

readln(f, value);

setlength(arr, length(arr) + 1);

arr[high(arr)] := value;

end

else

readln(f);

end;

closefile(f);

end;

procedure printarray(arr: tintarray);

var

i: integer;

begin

for i := 0 to high(arr) do

write(arr[i], ' ');

writeln;

end;

var

numbers, sortednumbers: tintarray;

inputfilename, outputfilename1, outputfilename2, outputfilename3, outputfilename4: string;

t1, t2, t3, t4, t5, t6, t7, t8, dt1, dt2, dt3, dt4: integer;

begin

inputfilename := 'D:\основы программирования и алгортмизации\дкр 5\input.txt';

outputfilename1 := 'D:\основы программирования и алгортмизации\дкр 5\output1.dat';

outputfilename2 := 'D:\основы программирования и алгортмизации\дкр 5\output2.dat';

outputfilename3 := 'D:\основы программирования и алгортмизации\дкр 5\output3.txt';

outputfilename4 := 'D:\основы программирования и алгортмизации\дкр 5\output4.txt';

readarrayfromtextfile(inputfilename, numbers);

writeln('Изначальный массив:');

printarray(numbers);

t1 := milliseconds;

sortednumbers := copy(numbers);

quicksort(sortednumbers, 0, high(sortednumbers), ascending);

writearraytotextfile(outputfilename1, sortednumbers);

t2 := milliseconds;

writeln('Отсортированный массив (быстрая сортировка по возрастанию):');

printarray(sortednumbers);

t3 := milliseconds;

sortednumbers := copy(numbers);

quicksort(sortednumbers, 0, high(sortednumbers), descending);

writearraytotextfile(outputfilename2, sortednumbers);

t4 := milliseconds;

writeln('Отсортированный массив (быстрая сортировка по убыванию):');

printarray(sortednumbers);

t5 := milliseconds;

sortednumbers := copy(numbers);

countingsort(sortednumbers, ascending);

writearraytotextfile(outputfilename3, sortednumbers);

t6 := milliseconds;

writeln('Отсортированный массив (сортировка подсчётом по возрастанию):');

printarray(sortednumbers);

t7 := milliseconds;

sortednumbers := copy(numbers);

countingsort(sortednumbers, descending);

writearraytotextfile(outputfilename4, sortednumbers);

t8 := milliseconds;

writeln('Отсортированный массив (сортировка подсчётом по убыванию):');

printarray(sortednumbers);

dt1 := t2 - t1;

dt2 := t4 - t3;

dt3 := t6 - t5;

dt4 := t8 - t7;

writeln;

writeln('Время выполнения:');

writeln('Быстрая сортировка (по возрастанию): ', dt1, ' мс');

writeln('Быстрая сортировка (по убыванию): ', dt2, ' мс');

writeln('Сортировка подсчётом (по возрастанию): ', dt3, ' мс');

writeln('Сортировка подсчётом (по убыванию): ', dt4, ' мс');

end.

**Результат выполнения программы**

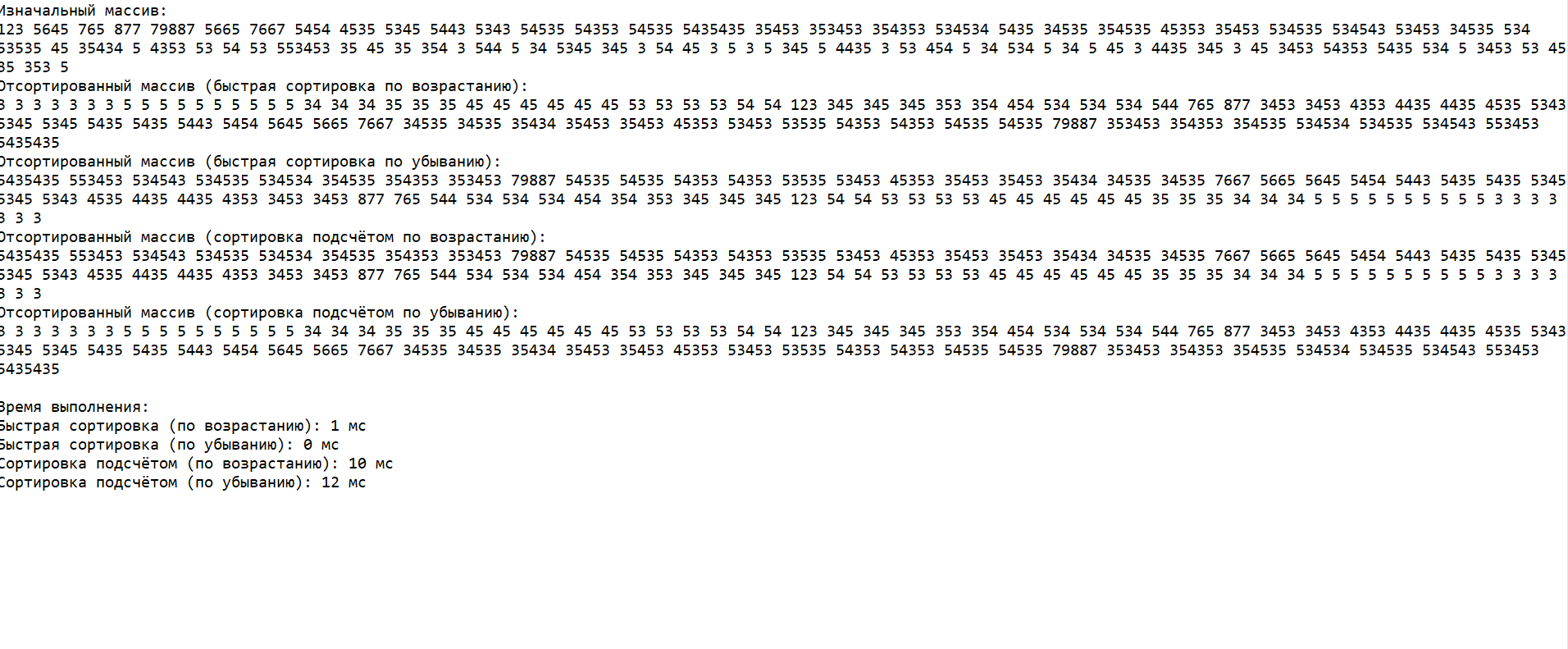
****

Рисунок 14 — Результат выполнения программы

**Вывод**

В рамках нашего исследования алгоритмов сортировки мы сосредоточились на двух методах — сортировке подсчётом и быстрой сортировке. Оба алгоритма имеют свои уникальные особенности, преимущества и недостатки, которые мы подробно рассмотрели.

Сортировка подсчётом является одним из самых эффективных алгоритмов для сортировки целых чисел, особенно в случаях, когда диапазон возможных значений не слишком велик. Этот алгоритм работает за время O(n + k), где n — количество элементов, а k — максимальное значение. Мы отметили, что данная сортировка идеально подходит для ситуаций, когда необходимо упорядочить большие массивы данных с небольшим диапазоном возможных значений. Примером может служить сортировка баллов студентов, где значения ограничены и предсказуемы.

С другой стороны, быстрая сортировка использует принцип «разделяй и властвуй» и в большинстве случаев демонстрирует отличные результаты. Средняя временная сложность быстрой сортировки составляет O(n log n), что делает её одним из самых быстрых алгоритмов сортировки для общего случая. Однако важно отметить, что в худшем случае, например, при уже отсортированном массиве, её сложность может достигать O(n²). Этот фактор делает быструю сортировку менее предпочтительной в ситуациях, где массив может быть уже отсортированным.

В ходе нашего исследования мы также провели эксперименты, сравнивая время выполнения обоих алгоритмов на различных наборах данных. Результаты подтвердили, что выбор алгоритма сортировки зависит от конкретной задачи и характеристик данных. Сортировка подсчётом продемонстрировала превосходную производительность на ограниченных диапазонах данных, тогда как быстрая сортировка проявила свою эффективность на больших объемах данных с разнообразными значениями.

Мы пришли к выводу, что важно понимать особенности и ограничения каждого алгоритма, чтобы выбрать наиболее подходящий метод сортировки в конкретной ситуации. Таким образом, правильный выбор алгоритма сортировки может значительно ускорить процесс обработки данных и повысить общую эффективность программного обеспечения.