**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАИНДУСТРИИ**

***Институт Принтмедиа и информационных технологий***

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**ЛАБОРАТОРНАЯ РАБОТА № 16-17**

**Дисциплина:** Основы алгоритмизации и программирования.

**Тема:** Алгоритм сортировки «быстрая сортировка»

**Выполнила:**

**студентка группы 201-723**

Круглова А.М.

17.12.20  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Проверил:** преп. Хуснулина Д.Р.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Москва**

**2020**

**Оглавление**

[Цель 3](#_Toc59159778)

[Постановка задачи 3](#_Toc59159779)

[Идея алгоритма 3](#_Toc59159780)

[Словесное представление алгоритма 4](#_Toc59159781)

[Блок-схема программы 5](#_Toc59159782)

[Листинг программы 6](#_Toc59159783)

[Результат работы программы 7](#_Toc59159784)

# **Цель**

Получить практические навыки разработке алгоритмов и их программной реализации.

# **Постановка задачи**

Необходимо выполнить и оформить описание следующих пунктов:

1. Сформулировать идею алгоритма
2. Выполнить словесное представление алгоритма
3. Выполнить полнить представление алгоритма с помощью блок схем с использованием элемента модификации и без него.
4. Выполнить программную реализацию алгоритмов на языке С с использованием параметрического цикла и цикла с предусловием.

# **Идея алгоритма**

Выбираем из массива элемент, называемый опорным, и запоминаем его значение. Это может быть любой из элементов массива. От выбора опорного элемента не зависит корректность алгоритма, но в отдельных случаях может сильно зависеть его эффективность.

Далее начинаем двигаться от начала массива по возрастающей, а потом от конца массива по убывающей. Цель: переместить в правую часть элементы больше опорного, а в левую – элементы меньше опорного. Если во время движения по возрастающей находится элемент со значением больше опорного, то мы выходим из цикла, прибавляем единицу к индексу элемента, на котором остановились, и переходим к циклу с движением по убывающей. В этом цикле мы остаемся до тех пор, пока не находится элемент со значением меньше опорного.

Как только такой элемент найден, мы отнимаем единицу от его индекса, и меняем значение элемента со значением элемента, на котором мы остановились в предыдущем цикле. Делаем так до тех пор, пока индекс левого элемента (найденного в первом цикле) меньше либо равен индексу правого элемента (найденного во втором цикле). В итоге получаем два подмассива (от начала до индекса правого элемента и от индекса левого элемента до конца). С этими подмассивами мы рекурсивно проделываем все то же самое, что и с большим массивом до тех пор, пока все элементы окончательно не отсортируются.

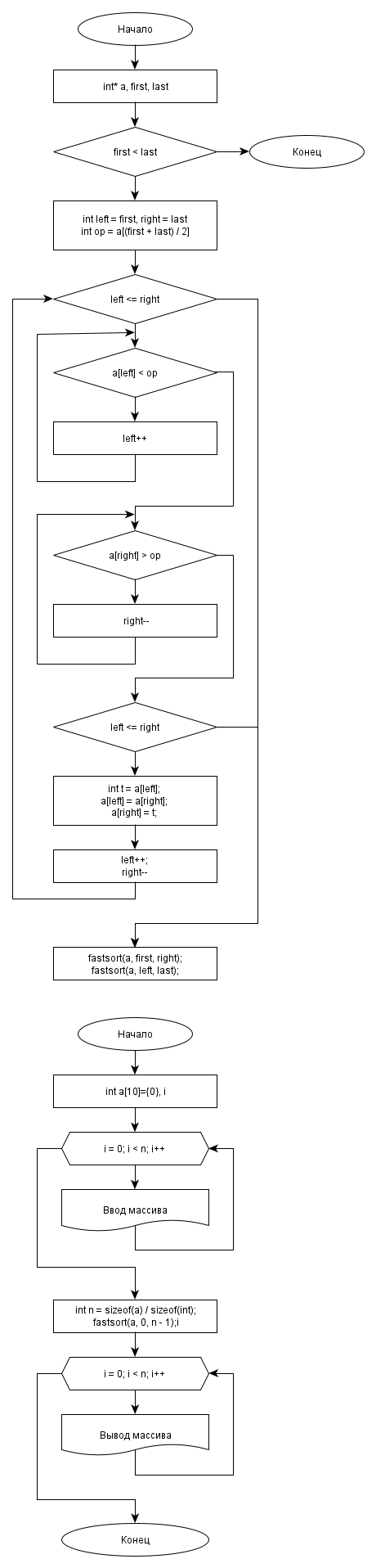
Словесное представление алгоритма

a – массив, op- номер опорного элемента массива, i - индекс первого элемента

массива, j – индекс последнего элемента массива

1. Номер опорного элемента- целая часть от (i+j)/2
2. Если t<= k (где t= i, a k = j), то переходим к пункту 3, иначе к пункту 13
3. Если a[t] < a[op], то переходим к пункту 4, иначе к пункту 6.
4. t++
5. Возвращаемся к 3
6. Если a[t] > a[op], то 7, иначе 9
7. k –
8. Возвращаемся к 6
9. Если t <= k, то переходим к 10, иначе переходим к 13.
10. Меняем значения элементов с индексами t и k местами
11. Прибавляем единицу к индексу левого элемента и убавляем на единицу индекс правого элемента.
12. Возвращаемся к пункту 2.
13. Если индекс первого элемента массива меньше индекса правого элемента массива (i<k), то 14, иначе 15
14. Вызываем a[],i,k
15. Если индекс левого элемента меньше индекса последнего элемента массива(t<j), то 16, иначе 17
16. Вызываем a[],t,j
17. Конец

# **Блок-схема программы**



Нет

Да

Нет

Да

Нет

Да

Нет

Да

Нет

Да

Рисунок 1 — Блок-схема программы

Листинг программы

Листинг 1 — Исходный код программы

#include <iostream>

#include <stdlib.h>

#include <locale.h>

void fastsort(int\* a, int first, int last)

{

if (first < last) // тк программа работает через рекурсия, то создаем новый метод, для ссылки не на массив,

//а на его имя, оставшиеся две перменные - номера первого и последнего элементов в массиве

{

int left = first, right = last; // чтобы идти от самого левого и самого правого элементов относительно нашего опорного

int op = a[(first + last) / 2]; // запоминаем номер опорного элемента

do

{

while (a[left] < op) // пока значение левого относительного опорного элемента, меньше значения опорного, увеличиваем номер левого элемента

left++;

while (a[right] > op) // все с точность до наоборот, как для левого

right--;

if (left <= right) // если номер левого меньше или равен номеру правого, то переприсваиваем

{

int t = a[left];

a[left] = a[right];

a[right] = t;

left++;

right--;

}

} while (left <= right); // пока номер левого меньше или равен номеру правого, делаем все, что выше, а после вызываем ещё 2 такие же программы

fastsort(a, first, right);

fastsort(a, left, last);

}

}

int main()

{

setlocale(LC\_ALL, "Russian"); //установка русского языка

int a[10]={0}, i; // Объявляем массив из 10 элементов

printf("Введите 10 цифр в столбик:\n");

for (i = 0; i < 10; i++) // Вводим значения элементов массива через клавиатуру

{

scanf\_s("%d", &a[i]);

}

int n = sizeof(a) / sizeof(int);

fastsort(a, 0, n - 1); //а- массив, 0-номер первого элемента, n-1 номер последнего элемента(кол - во элементов массива - 1)

printf("Преобразованный массив:\n");

for (int i = 0; i < n; i++) //выводим массив на экран

printf("%d ", a[i]);

printf("\n"); //выводим пустую строку

system("pause");

return 0;

}

Результат работы программы

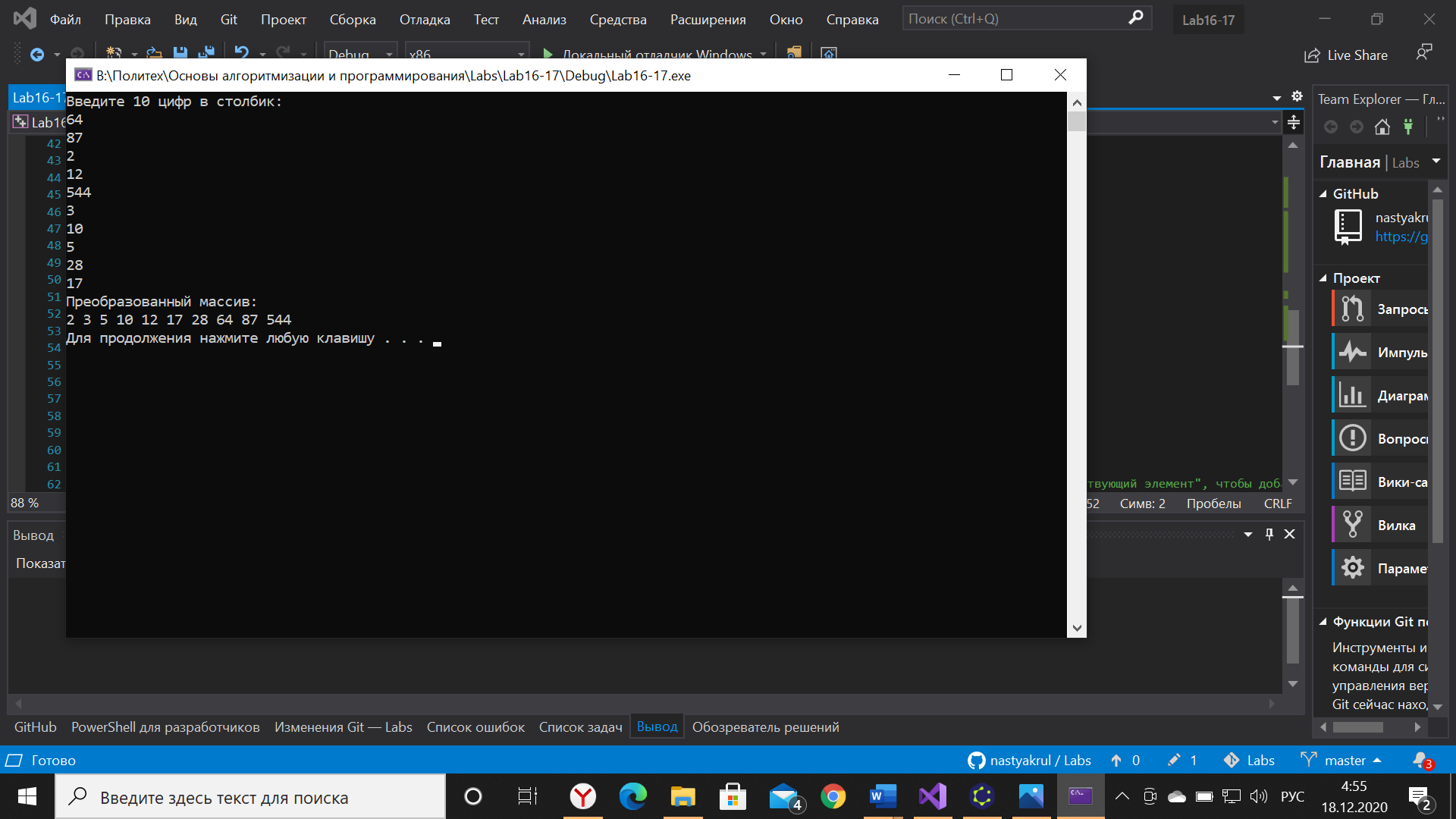


Рисунок 2 — Результат работы программы