**Оглавление**

[Введение 2](#_Toc139354804)

[1 Постановка задачи 4](#_Toc139354805)

[2 Выбор решения 6](#_Toc139354806)

[3 Описание программы 7](#_Toc139354807)

[4 Схемы программы 9](#_Toc139354808)

[5 Совместная разработка 12](#_Toc139354809)

[Заключение 15](#_Toc139354810)

[Список используемой литературы 16](#_Toc139354811)

[Приложение А 17](#_Toc139354812)

[Приложение B “Листинг” 19](#_Toc139354813)

# Введение

Язык программирования С++ представляет высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений. На сегодняшний день С++ является одним из самых популярных и распространенных языков.

Своими корнями он уходит в язык Си, который был разработан в 1969—1973 годах в компании Bell Labs программистом Деннисом Ритчи. В начале 1980-х годов датский программист Бьерн Страуструп, который в то время работал в компании Bell Labs, разработал С++ как расширение к языку Си. Фактически вначале C++ просто дополнял язык Си некоторыми возможностями объектно-ориентированного программирования. И поэтому сам Страуструп вначале называл его как "C with classes" ("Си с классами").

С++ является мощным языком, унаследовав от Си богатые возможности по работе с памятью. Поэтому нередко С++ находит свое применение в системном программировании, в частности, при создании операционных систем, драйверов, различных утилит, антивирусов и т.д. К слову сказать, ОС Windows большей частью написана на С++. Но только системным программированием применение данного языка не ограничивается. С++ можно использовать в программах любого уровня, где важны скорость работы и производительность. Нередко он применяется для создания графических приложений, различных прикладных программ. Также особенно часто его используют для создания игр с богатой насыщенной визуализацией. Кроме того, в последнее время набирает ход мобильное направление, где С++ тоже нашел свое применение. И даже в веб-разработке также можно использовать С++ для создания веб-приложений или каких-то вспомогательных сервисов, которые обслуживают веб-приложения. В общем С++ – язык широкого пользования, на котором можно создавать практически любые виды программ.

QuickSort является существенно улучшенным вариантом алгоритма сортировки с помощью прямого обмена (его варианты известны как «Пузырьковая сортировка» и «Шейкерная сортировка»), известного в том числе своей низкой эффективностью. Принципиальное отличие состоит в том, что в первую очередь производятся перестановки на наибольшем возможном расстоянии и после каждого прохода элементы делятся на две независимые группы (таким образом улучшение самого неэффективного прямого метода сортировки дало в результате один из наиболее эффективных улучшенных методов).

Общая идея алгоритма состоит в следующем:

-Выбрать из массива элемент, называемый опорным. Это может быть любой из элементов массива. От выбора опорного элемента не зависит корректность алгоритма, но в отдельных случаях может сильно зависеть его эффективность (см. ниже).

-Сравнить все остальные элементы с опорным и переставить их в массиве так, чтобы разбить массив на три непрерывных отрезка, следующих друг за другом: «элементы меньшие опорного», «равные» и «большие».

-Для отрезков «меньших» и «больших» значений выполнить рекурсивно ту же последовательность операций, если длина отрезка больше единицы.

На практике массив обычно делят не на три, а на две части: например, «меньшие опорного» и «равные и большие»; такой подход в общем случае эффективнее, так как упрощает алгоритм разделения (см. ниже).

Хоар разработал этот метод применительно к машинному переводу; словарь хранился на магнитной ленте, и сортировка слов обрабатываемого текста позволяла получить их переводы за один прогон ленты, без перемотки её назад. Алгоритм был придуман Хоаром во время его пребывания в Советском Союзе, где он обучался в Московском университете компьютерному переводу и занимался разработкой русско-английского разговорника.

# 1 Постановка задачи

Необходимо разработать алгоритм быстрой сортировки.

Необходимо заполнить массив из n-ого количества элементов случайными числами, записать данные элементы в отдельный файл. После этого выполнить выборочную сортировку над данными, находящимися в массиве, записать отсортированные данные в другой файл.

Использовать сервис GitHub для совместной работы.

Оформить отчет по проведенной практике.

1.1 Достоинства алгоритма:

* Один из самых быстродействующих (на практике) из алгоритмов внутренней сортировки общего назначения.
* Прост в реализации.
* Требует лишь O(log⁡ n) дополнительной памяти для своей работы. (Не улучшенный рекурсивный алгоритм в худшем случае O(n) памяти)
* Хорошо сочетается с механизмами кэширования и виртуальной памяти.
* Допускает естественное распараллеливание (сортировка выделенных подмассивов в параллельно выполняющихся подпроцессах).
* Допускает эффективную модификацию для сортировки по нескольким ключам (в частности — алгоритм Седжвика для сортировки строк): благодаря тому, что в процессе разделения автоматически выделяется отрезок элементов, равных опорному, этот отрезок можно сразу же сортировать по следующему ключу.
* Работает на связных списках и других структурах с последовательным доступом, допускающих эффективный проход как от начала к концу, так и от конца к началу.

1.2 Недостатки алгоритма:

* Сильно деградирует по скорости (до O(n2) в худшем или близком к нему случае, что может случиться при неудачных входных данных.
* Прямая реализация в виде функции с двумя рекурсивными вызовами может привести к ошибке переполнения стека, так как в худшем случае ей может потребоваться сделать O(n)вложенных рекурсивных вызовов.
* Неустойчив.

1.3 Типичные сценарии применения:

Зачастую мы заочно сортируем данные. Также если у вас небольшой понятный массив, можно воспользоваться встроенной функцией языка. Однако не всегда условия просты в исполнения. Cложности начинаются, когда:

* это работа с массивами данных на десятки или сотни тысяч элементов;
* затруднён доступ к этим данным;
* нужно оптимизировать время выполнения или требуемый объём памяти. Тогда уже имеет место воспользоваться алгоритмом сортировки.

# 2 Выбор решения

В качестве среды разработки выбрана Visual Studio 2022 на языке C++.

Сначала пользователю предложится ввести размер массива >1.

Массив данных заполняется случайными элементами с использованием цикла for и функции rand().

После заполнения массива, данные переписываются в файл “mas.txt”, а массив сортируется с помощью функции qsortRec.

После того, как массив будет отсортирован, данные из массива переписываются в другой файл “sortedmas.txt”.

Программа завершает свою работу.

# 3 Описание программы

В программе для быстрой сортировки подключены следующие заголовочные файлы: *iostream* – заголовочный файл с классами, функциями и переменными для организации ввода-вывода; *Windows.h* – специфичный заголовочный файл, необходимый для использования в программе функционала, предоставляемого операционной системой Windows; ***fstream*** – **заголовочный файл,** предоставляющий функционал для считывания данных из файла и для записи в файл; *ctime*– заголовочный файл стандартной библиотеки C++, содержащий типы и функции для работы с датой и временем; *using namespace std* – использование стандартного пространства имен.

Далее идет текст функции быстрой сортировки qsortRec. Выбирается разрешающий элемент, и сохраняются левая и правая границы массива.

После в цикле с предусловием, пока границы не сомкнутся, осуществляется перестановка элементов в массиве относительно разрешающего элемента и сдвиг границ: элементы меньше разрешающего располагаются левее него, а элементы, больше или равные разрешающему, – правее разрешающего элемента.

Затем ставим на место разрешающий элемент.

Потом выполняются рекурсивные вызовы функции qsortRec для левой и правой частей массива при условии, что они содержат более одного элемента

Далее идет текст функции main.

Потом выполняется инициализация генератора случайных чисел *rand* и объявление переменной целого типа *size* – размера массива, *\* mas* – указателя на массив, *i* – счетчика цикла, две переменные строкового типа *filename1* и *filename2.*

Далее осуществляется ввод и проверка ввода размера массива *size* в цикле с постусловием: пока ввод неверный, выводится сообщение о неверном вводе, и вводится значение переменной *size.*

Затем выделяется память для динамического массива *mas*.

Потом происходит генерация массива случайными числами в диапазоне от -5000 до 5000, записывается в исходный файл, и выводится сообщение в консоль о том, в каком файле хранится сгенерированный массив.

Далее объявляется и инициализируется переменная временного типа *start*, вызывается функция быстрой сортировки *qsortRec*, объявляются и инициализируются переменная временного типа *stop* и переменная вещественного типа *time.*

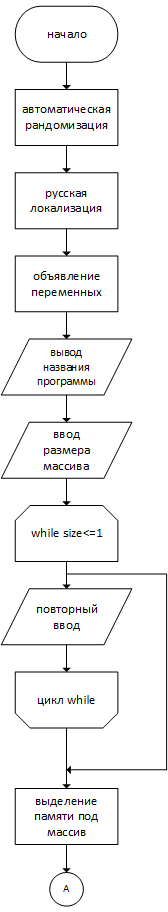
После этого отсортированный массив записывается другой файл.

Затем, и выводится сообщение в консоль о том, где хранится исходный массив и отсортированный по возрастанию массив.

Потом выводится в консольное окно время выполнения быстрой сортировки.

# 4 Схемы программы

4.1 Блок-схема программы



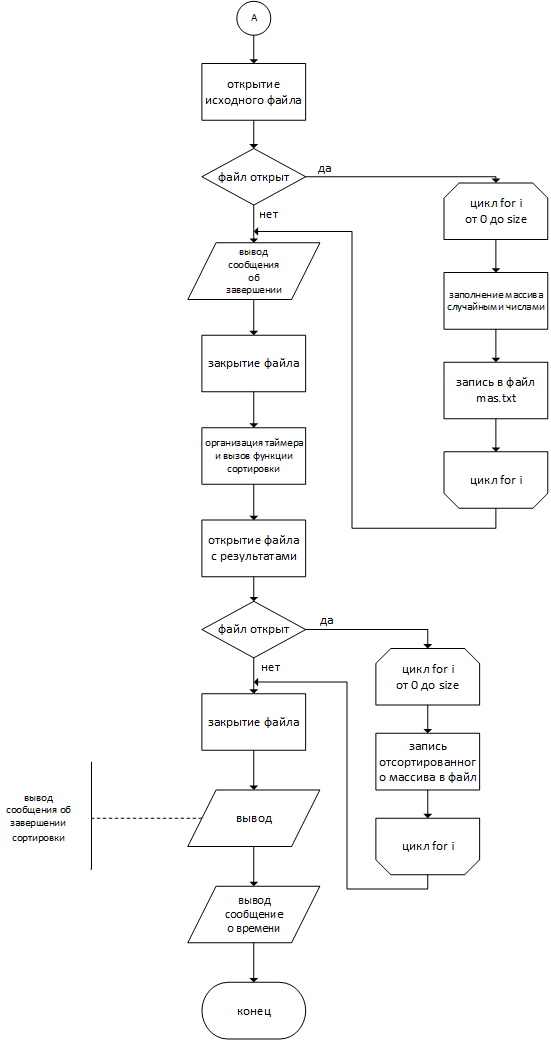


Рисунок 1. - “Блок-схема программы”

4.2 Блок-схема алгоритма

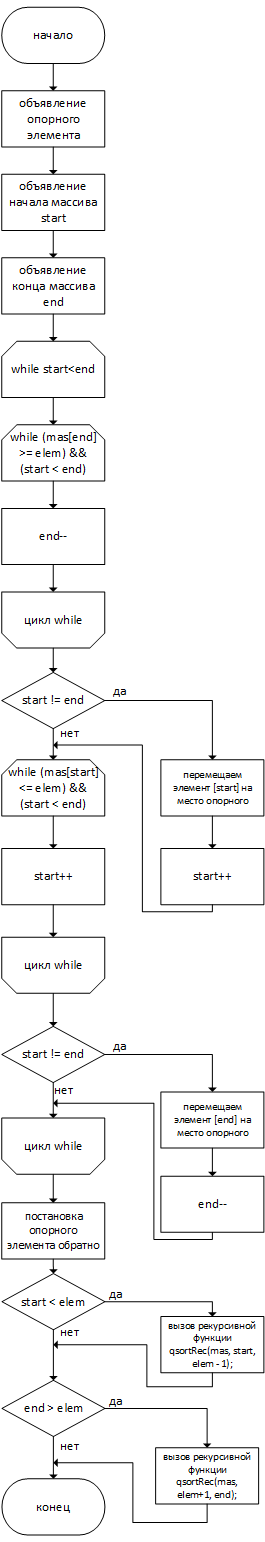


Рисунок 2. - “Блок-схема алгоритма”

# 5 Совместная разработка

Во время работы над данной практикой, нашей бригадой осуществлялась совместная работа в GitHub.

Данная программа была написана владельцем репозитория – Гурьяновым Д.И.

В ходе ее написания она несколько раз усовершенствовалась.

Сначала связываем локальный репозиторий с удаленным (см. Рисунок 3).



Рисунок 3

Затем фиксируем изменения, добавляя папку программы и делаем коммит (см. Рисунок 4).

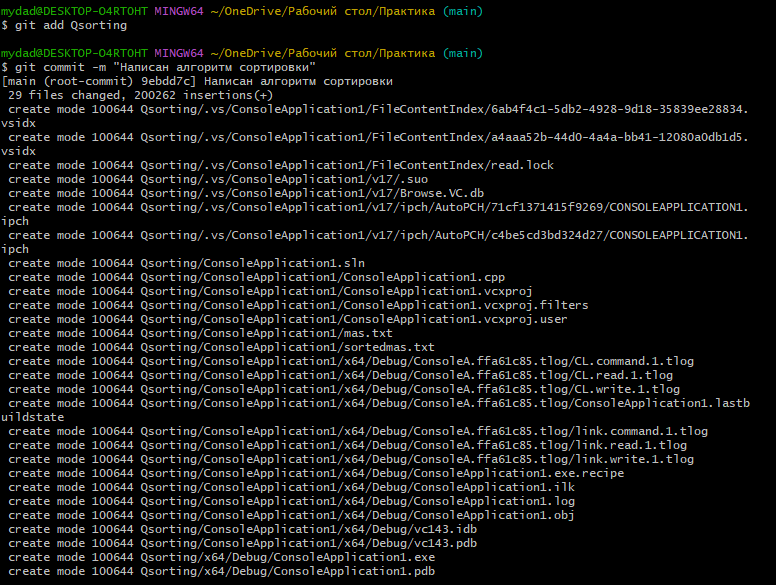


Рисунок 4

Отправил зафиксированные изменения в главную ветку (см. Рисунок 5).

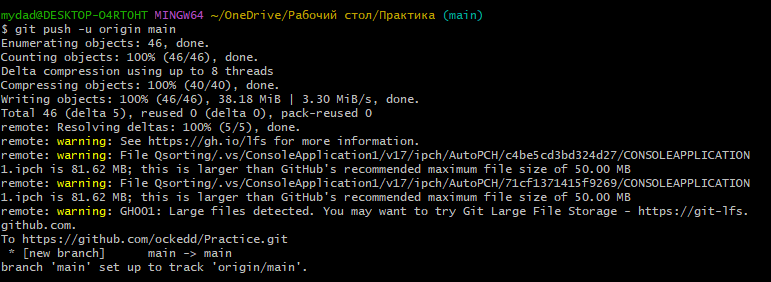


Рисунок 5

Проверил успешность отправки:

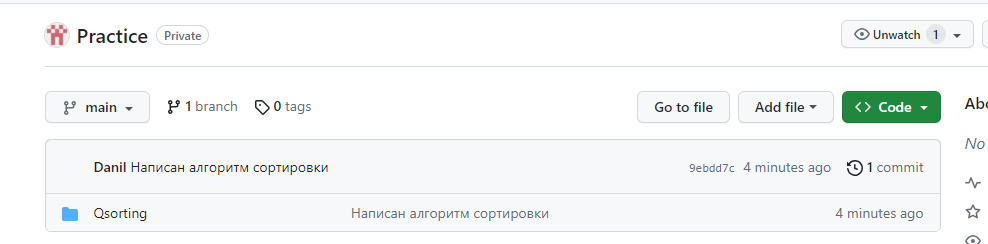


Рисунок 6. – “Вид в репозитории”

Далее были сделаны несколько коммитов в соответствии с улучшениями кода:

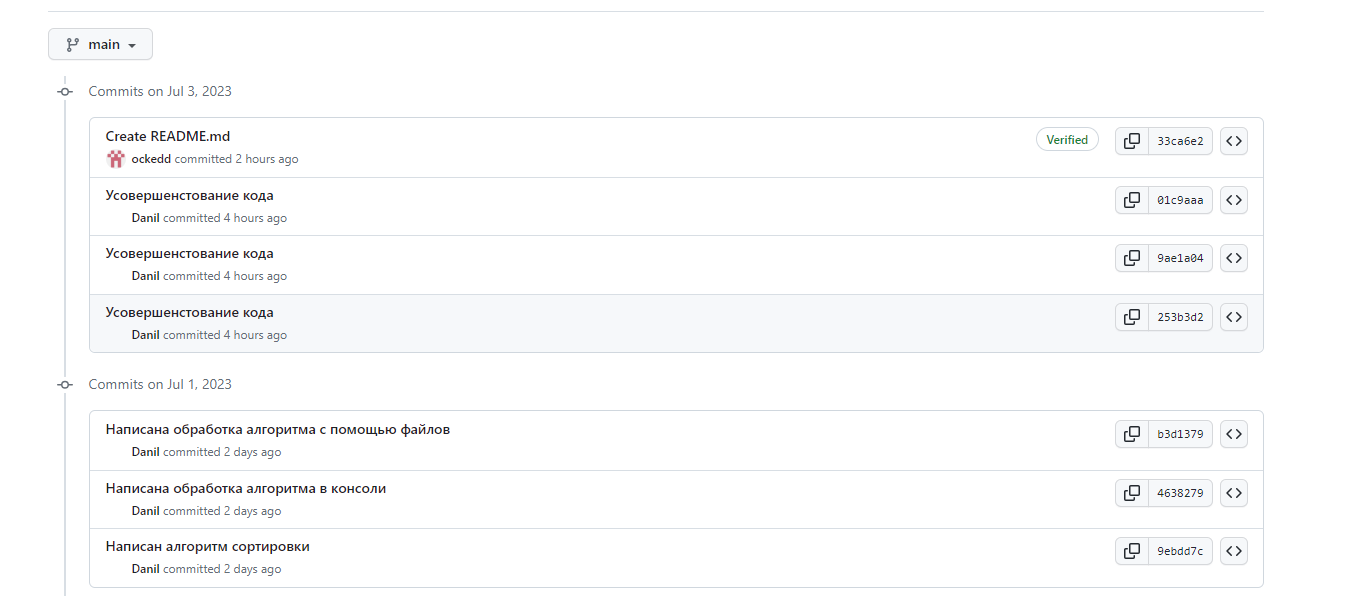


Рисунок 7. – “Список коммитов”

После чего второй участник бригады, должен был загрузить программу к себе на компьютер с помощью git clone и провести тестирование.

Ссылка на удаленный репозиторий:

https://github.com/ockedd/Practice.git

# Заключение

В ходе работы нами был изучен алгоритм быстрой сортировки. Гурьянов Д.И написал программу, выполняющую данную сортировку над массивом случайно сгенерированных чисел, выполнил работу с файлами, и оформил отчет по данной практике. Крупнов В.Е выполнил тестирование и отладку данной программы. Также были усовершенствованы навыки совместной работы с помощью сервиса GitHub, навыки использования программы Git Bash.

Так же при выполнении практической работы были улучшены наши базовые навыки программирования на языке C++. Улучшены навыки отладки, тестирования программ и работы со сложными типами данных.

В дальнейшем программу можно улучшить путем добавления графического интерфейса.

# 

# 

# Список используемой литературы

1. Прата С. Язык программирования С++. 2019 г.

2.Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. 2017 г.

# Приложение А



Рисунок 8. – “Ввод размера массива”

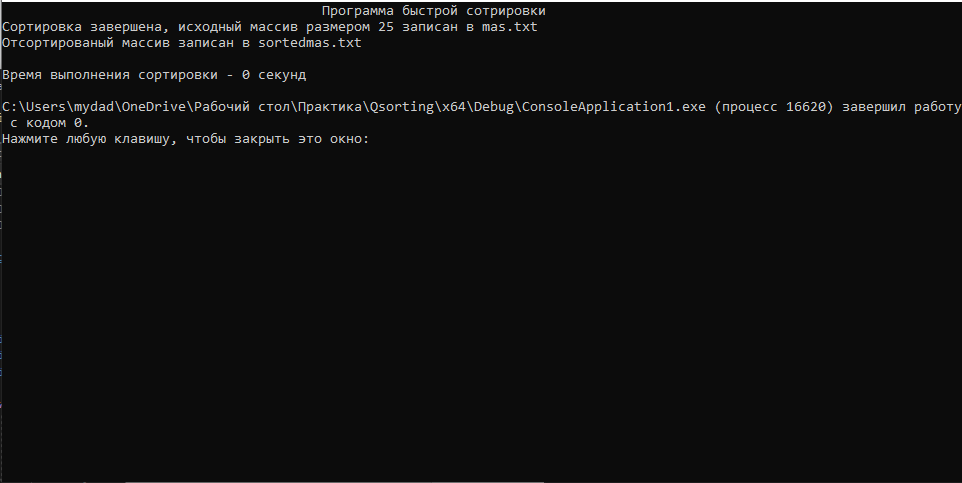


Рисунок 9. – “Результат работы в консоли”

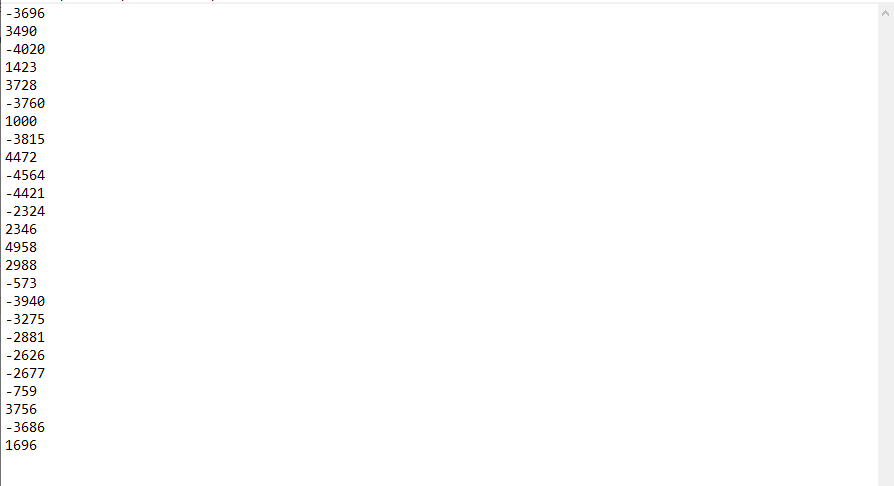


Рисунок 10. – “Результат в файле mas.txt”

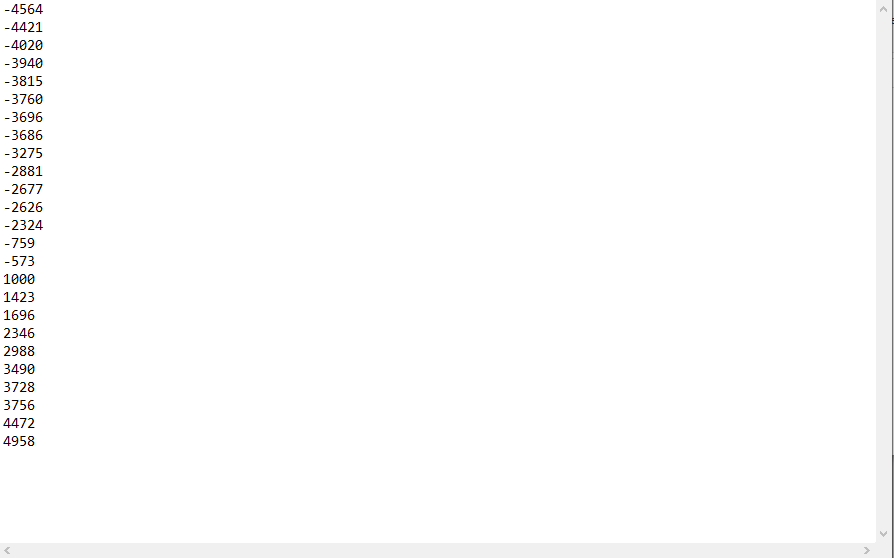


Рисунок 11. – “Результат в файле sortedmas.txt”

# Приложение B “Листинг”

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include <locale.h>

#include <iostream>

#include <fstream>

using namespace std;

void qsortRec(int\* mas, int start, int end) {

int elem = mas[start]; //опорный элемент

int i = start; //начало массива

int j = end; //конец массива

while (start < end) { //пока границы не сомкнутся

while ((mas[end] >= elem) && (start < end)) //пока конечный элемент массива больше опорного

end--; //сдвигаем правую границу влево

if (start != end) //границы не сомкнулись

{

mas[start] = mas[end]; //перемещаем элемент [start] на место разрешающего

start++;//сдвигаем левую границу вправо

}

while ((mas[start] <= elem) && (start < end)) //пока начальный элемент массива меньше опорного

start++; //сдвигаем левую границу вправо

if (start != end) //границы не сомкнулись

{

mas[end] = mas[start]; //перемещаем элемент [end] на место разрешающего

end--;//сдвигаем правую границу влевоw

}

}

mas[start] = elem;

elem = start;

start = i;

end = j;

//Рекурсивные вызовы, если осталось, что сортировать

if (start < elem) {

//"Левый кусок"

qsortRec(mas, start, elem - 1);

}

if (end > elem) {

//"Правый кусок"

qsortRec(mas, elem + 1, end);

}

}

int main() {

srand(time(NULL)); //автоматическая рандомизация на основе текущего времени

setlocale(LC\_ALL, "Russian");//русская локализация консоли

int size; //размер массива

int\* mas; //указатель на массив

int i; //счетчик цикла

string filename1 = "mas.txt"; //имя файла исходного массива

string filename2 = "sortedmas.txt"; // имя файла отсортированного массива

cout << "\t\t\t\t\t" << "Программа быстрой сотрировки" << endl;

cout << "Введите размер массива(>1): ";//

cin >> size; //ввод размера массива с клавиатуры

cout << "\n"; //

while(size <= 1){ // проверка равен ли массив 1 и меньше

system("cls"); // очистка консоли

cout << "\t\t\t\t\t" << "Программа быстрой сотрировки" << endl;

cout << "Массив не может быть <= 1, введите снова: "; //

cin >> size; //повторный ввод размера массива

cout << "\n";

}

int\* mas1 = new int[size];//выделение памяти для массива

ofstream out; //поток для записи

out.open(filename1); //открываем файл для записи

if (out.is\_open()) { //если файл открыт

cout << "Генерация массива и запись в файл...";

for (i = 0; i < size; i++) { //цикл i от 0 до размера массива

mas1[i] = (rand() % 10000) - 5000; //запись в i-ый элемент массива случайное число от -5000 до 5000

out << mas1[i] << '\n'; //запись i-ого элемента массива в файл с новой строки

}

}

system("cls");

cout << "\t\t\t\t" << "Программа быстрой сотрировки" << endl;

cout << "Генерация завершена, массив размером " << size << " записан в " << filename1 << endl;

cout << endl << "Сортируем...";

out.close(); //закрываем файл

time\_t start = clock(); //время до сортировки

qsortRec(mas1, 0, size - 1); //вызов функции сортировки

time\_t stop = clock(); //время после сортировки

double time = (stop - start) / 1000.0; //время сортировки

ofstream sorted; //поток для записи

sorted.open(filename2); //открываем файл для записи

if (sorted.is\_open()) { //если файл открыт

for (i = 0; i < size; i++) { //цикл i от 0 до размера массива

sorted << mas1[i] << '\n'; //запись i-ого элемента отсортированного массива в файл с новой строки

}

}

sorted.close(); //закрываем файл

system("cls");

cout << "\t\t\t\t\t" << "Программа быстрой сотрировки" << endl;

cout << "Сортировка завершена, исходный массив размером " << size << " записан в " << filename1 << endl;

cout << "Отсортированый массив записан в " << filename2 << endl;

cout << endl << "Время выполнения сортировки – " << time << " секунд " << endl; //вывод времени выполнения сортировки

getchar();//запрашивает нажатие любой клавиши для продолжения работы

return 0;

}