МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное

образовательное учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

Кафедра информационно-аналитических систем безопасности

имени Л.С. Берштейна

**ОТЧЕТ**

**По Лабораторной работе № 7**

***Методы сортировки***

**Вариант - 20**

Выполнил:

студент гр. КТбо1-10

Мельман Д.Э

Проверил:

Профессор кафедры ИАСБ

Барковский С.А.

«\_\_\_» 2020 г.

Таганрог – 2021

Код Программы:

#include <iostream>

#include <fstream>

using namespace std;

// Лабараторная работа | методы сортировки

// Вариант 10

int temp = 0, pivot = 0;

void Bubble\_Sort(int\* mass, int size) {

for (int i = 0; i < size - 1; i++) {

for (int j = 0; j < size - i - 1; j++) {

if (mass[j] > mass[j + 1]) {

temp = mass[j];

mass[j] = mass[j + 1];

mass[j + 1] = temp;

}

}

}

}

void Quick\_Sort(int\* mass, int size) {

//Указатели в начало и в конец массива

int i = 0;

int j = size - 1;

//Центральный элемент массива

int mid = mass[size / 2];

//Делим массив

do {

// В левой части чила, которые меньше центрального

while (mass[i] < mid) {

i++;

}

// В правой части пропускаем элементы, которые больше центрального

while (mass[j] > mid) {

j--;

}

//Меняем элементы местами

if (i <= j) {

int temp = mass[i];

mass[i] = mass[j];

mass[j] = temp;

i++;

j--;

}

} while (i <= j);

//Рекурсивные вызовы, если осталось, что сортировать

if (j > 0) {

//"Левый кусок"

Quick\_Sort(mass, j + 1);

}

if (i < size) {

//"Првый кусок"

Quick\_Sort(&mass[i], size - i);

}

}

void Merge(int mass[], int first, int last, int size) {

int middle, start, final, j;

int\* mas = new int[size];

middle = (first + last) / 2; //вычисление среднего элемента

start = first; //начало левой части

final = middle + 1; //начало правой части

for (j = first; j <= last; j++) { //выполнять от начала до конца

if ((start <= middle) && ((final > last) || (mass[start] < mass[final]))) {

mas[j] = mass[start];

start++;

}

else {

mas[j] = mass[final];

final++;

}

}

//возвращение результата в список

for (j = first; j <= last; j++) { mass[j] = mas[j]; }

delete[]mas;

};

void Merge\_Sort(int\* mass,int first, int last) {

int size = last;

if (first < last)

{

Merge\_Sort(mass, first, (first + last) / 2); //сортировка левой части

Merge\_Sort(mass, (first + last) / 2 + 1, last); //сортировка правой части

Merge(mass, first, last, size); //слияние двух частей

}

}

void Insertion\_Sort(int\* mass, int size) {

for (int j = 1; j < size; j++) {

temp = mass[j];

pivot = j - 1;

while (pivot >= 0 && mass[pivot] > temp) {

mass[pivot + 1] = mass[pivot];

pivot = pivot - 1;

mass[pivot + 1] = temp;

}

}

}

int main(){

setlocale(LC\_ALL, "ru");

unsigned rand\_value = 18; //зерно для создания \*рандомных чисел

ofstream file\_out; // Создаём не сортированный файл

file\_out.open("Sumvol\_no\_sort.txt");

ofstream end\_file\_out; // Создаём сортированный файл

end\_file\_out.open("Sumvol\_sort.txt");

int\* mass; // Указатель для выделения памяти под массив

int size; // Длина Значений

cout << "Введите количество символов = ";

cin >> size;

mass = new int[size] {}; // Динамический массив

if (!file\_out.is\_open()) {

cout << "Ошибка, файл Nсортированный не открылся!" << endl;

}

else {

cout << "Файл Nсортированный открылся" << endl;

srand(rand\_value); // Необходимо для создания рандомных чисел

for (int i = 0; i < size; i++) {

mass[i] = rand() % 10;

file\_out << mass[i];

}

cout << "Массив заполнен числами" << endl;

file\_out.close(); // Закрываем файл с \*рандомными числами

}

if (!end\_file\_out.is\_open()) {

cout << "Ошибка, файл-сортированный не открылся!" << endl;

}

else {

cout << "Файл-сортированный открылся" << endl;

int state;

cout << "Выберите нужную сортировку: " << endl <<

"1 - Пузырьком (Bubble sort)" << endl <<

"2 - Слиянием (Merge sort)" << endl <<

"3 - Вставками (Insertion sort)" << endl <<

"4 - Быстрая сортировка (Quick sort)" << endl;

cin >> state;

switch (state) {

//Bubble sort

case 1:

Bubble\_Sort(mass, size);

for (int i = 0; i < size; i++) { end\_file\_out << mass[i]; }

break;

//Merge sort

case 2:

Merge\_Sort(mass, 0, size);

for (int i = 0; i < size; i++) { end\_file\_out << mass[i]; }

break;

//Insertion sort

case 3:

Insertion\_Sort(mass, size);

// вывод в сортированный массив в файл

for (int i = 0; i < size; i++) { end\_file\_out << mass[i]; }

break;

//Quick sort

case 4:

Quick\_Sort(mass,size);

for (int i = 0; i < size; i++) { end\_file\_out << mass[i]; }

break;

}

cout << "Отсортировано!" << endl;

end\_file\_out.close(); // Закрываеи отсортированный файл

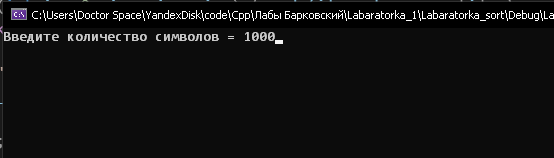
return 0;

}

}

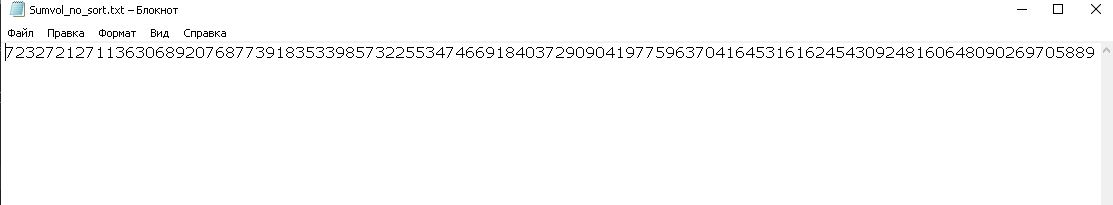
Пример:

При запуске программы необходимо ввести количество чисел для сортировки. (*size = 1000*)

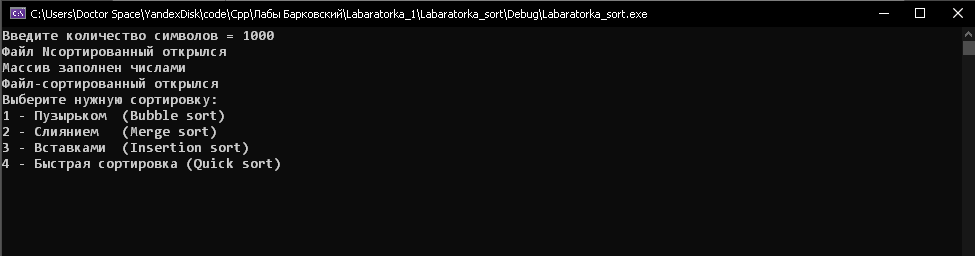


Дальше через зерно (*rand\_value = 18),* значение можно поменять на любое другое, чтобы выводилась другая последовательность \*рандомных чисел.

Далее создаётся файл Sumvol\_no\_sort.txt в котором хранится не отсортированный массив чисел



Дальше программа спрашивает метод сортировки:

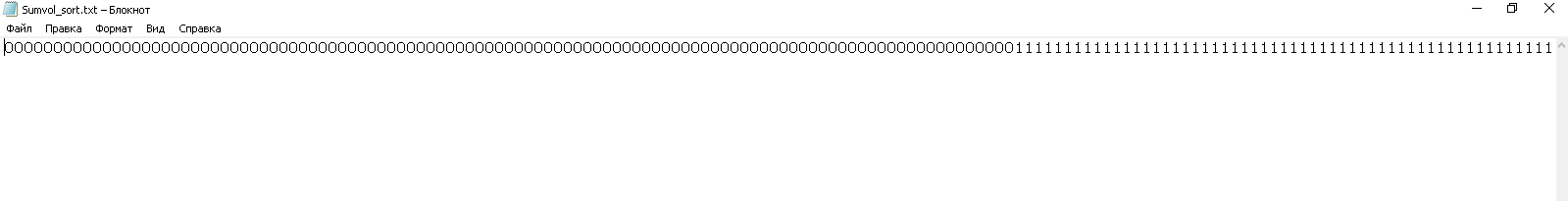


И после выбора определённой сортировки, допустим Quick\_Sort

Программа сорирует данным методом и выводит сообщение:



Так же создавая отсортированный файл Sumvol\_sort.txt, в котором: числа отсортированы в строчку



Для удобства смещю числа:

