Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

**Сортировки**

**Отчет по лабораторной работе №6**

**По дисциплине**

**«Структуры и алгоритмы обработки данных в ЭВМ»**

Выполнил: студент гр. 439-1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Зозуля Е.Д.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Проверил: ассистент каф. АСУ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Яблонский Я.В.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Томск 2021

# Задание на лабораторную работу

Вариант 5.

# Пусть даны три файла вещественных чисел, состоящих из 100, 1000 и 10000 чисел. Выполнить сортировку данных с помощью метода слияния. Подсчитать количество операций сравнения для каждого из трех файлов.

# Алгоритм решения задачи

Выполняется обход с поиском элемента Е, где одновременно с поиском элемента ведётся подсчёт количества проб.

# Листинг программы

Lab6.cpp

#include <iostream>

#include <fstream>

#include <stdio.h>

#include <stdlib.h>

#include <ctime>

using namespace std;

// Функция сортировки слиянием

int comp = 0;

void mergeSort(int\* a, int l, int r)

{

if (l == r) return; // границы сомкнулись

int mid = (l + r) / 2; // определяем середину последовательности

// и рекурсивно вызываем функцию сортировки для каждой половины

mergeSort(a, l, mid);

mergeSort(a, mid + 1, r);

int i = l; // начало первого пути

int j = mid + 1; // начало второго пути

// !!! Освободить динамическую память

int\* tmp = (int\*)malloc(r \* sizeof(int)); // дополнительный массив

for (int step = 0; step < r - l + 1; step++) // для всех элементов дополнительного массива

{

// записываем в формируемую последовательность меньший из элементов двух путей

// или остаток первого пути если j > r

if ((j > r) || ((i <= mid) && (a[i] < a[j])))

{

comp += 1;

tmp[step] = a[i];

i++;

}

else

{

comp += 1;

tmp[step] = a[j];

j++;

}

}

// переписываем сформированную последовательность в исходный массив

for (int step = 0; step < r - l + 1; step++)

a[l + step] = tmp[step];

}

int main()

{

setlocale(LC\_CTYPE, "RUSSIAN");

srand(static\_cast<unsigned int>(time(0)));

ifstream FileIn("Numbers.txt");

int\* a = nullptr;

int Size = 0;

// Заполняем массив

for(int x; FileIn >> x;)

{

++Size;

int\* array\_failsafe = a;

a = static\_cast<int\*>(realloc(a, sizeof(x) \* Size));

if (a == nullptr)

{

std::cerr << "realloc() failed!" << std::endl;

free(array\_failsafe);

return EXIT\_FAILURE;

}

a[Size - 1] = x;

}

for (int i = 0; i < Size; ++i)

{

if (i % 10 == 0)

cout << endl;

std::cout << a[i] << " ";

}

cout << endl<<endl;

FileIn.close();

comp = 0;

mergeSort(a, 0, Size-1); // вызываем функцию сортировки

// Выводим отсортированный массив

// !!! Выводить результат и в файл тоже (либо только в файл)

// !!! Обобщить код так, чтобы не было дублирований (т.е. чтобы и считывание\сортировка файла из 100 чисел и эти же действия с файлом из 1000 чисел производились одним участком кода)

ofstream Out("NumbOut.txt");

for (int i = 0; i < Size; ++i)

{

Out << a[i];

Out << endl;

if (i % 10 == 0)

cout << endl;

std::cout << a[i] << " ";

}

Out.close();

free(a);

cout << endl;

cout << "\nКол-во сравнений:" << comp << endl;

system("pause");

return 0;

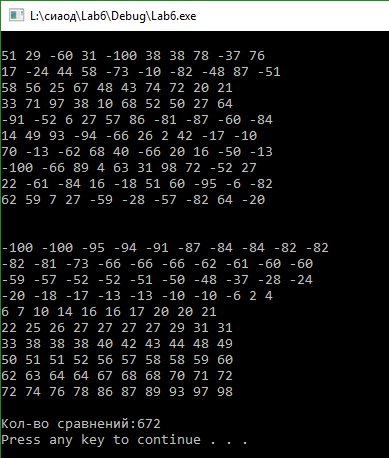
}

# Пример решения

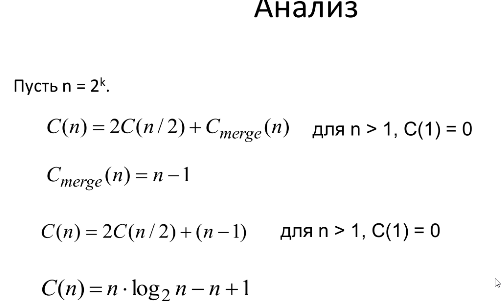
Скриншоты успешной работы программы, входные данные тестового примера, выходные данные на рисунке 4.1 и 4.2.,



*Рисунок 4.1 – Входные данные*



*Рисунок 4.2 – Отсортированный массив из 100 чисел*

**

*Рисунок 4.3 – Формулы расчёта эффективности алгоритма*

С(100) = 645, по формулам, что значит алгоритм выполнен эффективно

# Вывод

Был изучен алгоритм сортировки слиянием и использован для решения задачи.