ОТЧЕТ

о лабораторной работе №4

по теме: Алгоритмы сортировки

по дисциплине: Разработка программных модулей

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет среднего профессионального образования

09.02.07 Информационные системы и программирование

Выполнил:

Студент группы Y2333

\_\_\_\_\_\_\_\_ Максимов Д.О.

Проверил:

\_\_\_\_\_\_\_\_\_Антонов М.Б.

Дата: «\_\_» \_\_\_\_\_\_\_ 20 г.

Оценка:

Санкт-Петербург 2019

**Цель работы:** реализовать и замерить скорость работы алгоритмов сортировки.

**Задачи:**

* изучить несколько алгоритмов сортировки;
* научиться оценивать сложность программы;
* изучить работы с динамическими структурами данных;
* научиться устанавливать метрики на программное обеспечение.

**Задание на лабораторную работу:**

Общее ограничение на лабораторную работу:

* использовать разбитие на файлы основной программы (в main.cpp только функция main.cpp);
* документировать весь код используя аннотации Doxygen.

Требования к исходной программе:

1. Пользователь вводит размер массива в integer.
2. Массив для сортировки генерируется динамически случайно.
3. Массив сортируется, используя 2 разных алгоритма сортировки.
4. Массив до сортировки и после сортировки каждым методом выводится в отдельные текстовые файлы (что бы можно было сравнить и проконтролировать работу алгоритмов).
5. Вывод программы: алгоритм сортировки и сколько времени заняла сортировка в секундах.

Требования к лабораторной:

1. Выбрать 1 квадратичный алгоритм сортировки и любой другой из алгоритмов и реализовать его в исходной программе.
2. На массиве из 10 элементов доказывается что программа работает корректно (с выводом до и после сортировки).
3. Замеряется время сортировки массивов размерами: 100.000, 200.000, 300.000, 400.000 элементов.
4. Замерять только время выполнения сортировки, а не само время выполнения программы.
5. Проводится оценка сложности реализованных алгоритмов сортировки.

Заполняется таблица результатов лабораторной работы:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 100.000 | 200.000 | 300.000 | 400.000 | 500.000 | Тип зависимости | Сложность O(f) |
| Сортировка пузырьком | 30.8530 сек | 194.131 сек | 264.676 сек | 468.892 сек | 708.179 сек | Квадратичная | O(n2) |
| Быстрая сортировка | 0.073211 сек | 0.275883 сек | 0.615087 сек | 1.02565 сек | 1.62242 сек | Логарифмическая | O(nlog(n)) |

**Текст программы**:

#include <iostream>

#include <cstdlib>

#include <time.h>

#include <fstream>

using namespace std;

void bublesorting(int \*array, int size)

{

int temp;

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

{

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

{

if (array[j] > array[j + 1])

{

temp = array[j];

array[j] = array[j + 1];

array[j + 1] = temp;

}

}

}

}

void quicksorting(int \*array, int size, int first, int last)

{

int mid, temp;

int f = first, l = last;

mid = array[(f+l)/2];

do

{

while (array[f]<mid)

{

f++;

}

while (array[l]>mid)

{

l--;

}

if(f<=l)

{

temp = array[f];

array[f] = array[l];

array[f] = temp;

f++;

l--;

}

}

while (f<l);

{

if(first<l)

{

quicksorting(array, size, first, l);

}

if(f<last)

{

quicksorting(array, size, f, last);

}

}

}

int main()

{

int \*array;

int size, first = 0, last;

cout << "Enter size of array " << endl;

cin >> size;

last = size - 1;

array = new int [size];

srand(time(NULL)); // заполнение массива слуачайными числами

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

{

array[i] = 1 + rand() % 1000;

}

ofstream ent("entered.txt");

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

{

ent << array[i] << " ";

}

ent.close();

double start1 = clock();

bublesorting(array, size);

ofstream bl("bublesorted.txt");

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

{

bl << array[i] << " ";

}

bl.close();

double end1 = clock();

double seconds1 = (double)(end1 - start1) / CLOCKS\_PER\_SEC;

printf("The time of bublesorting: %f seconds\n", seconds1);

double start2 = clock();

quicksorting(array, size, first, last);

ofstream qs("quicksorted.txt");

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

{

qs << array[i] << " ";

}

qs.close();

double end2 = clock();

double seconds2 = (double)(end2 - start2) / CLOCKS\_PER\_SEC;

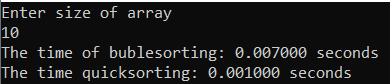
printf("The time quicksorting: %f seconds\n", seconds2);

return 0;

}

**Результат работы программы.**

Протокол программы представлен на рисунке 1.

**

*Рисунок 1 – Протокол программы*

Файлы с записанными и отсортированными массивами представлены на рисунке ниже.



*Рисунок 2 – Файл с исходным массивом*



*Рисунок 3 – Файл с отсортированным сортировкой пузырьком массивом*



*Рисунок 4 – Файл с отсортированным быстрой сортировкой массивом*