ОТЧЕТ

о лабораторной работе №4

по теме: Алгоритмы сортировки

по дисциплине: Разработка программных модулей

УНИВЕРСИТЕТ ИТМО

Факультет среднего профессионального образования

09.02.07 Информационные системы и программирование

Выполнил:

Студент группы Y2333

\_\_\_\_\_\_\_\_ Харченко Д.И.

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 | 47.3065 сек | 189.182 сек | 440.171 сек | 798.54 сек | 1213.23 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**:

**“main.cpp”:**

/\*\*

@file main.cpp

@author Popovskaya Angelina <anponew@mail.com>

@brief Программа с двумя способами сортировки массива: пузырьком и быстрой

\*/

#include <iostream>

#include <locale.h>

#include <cstdlib>

#include <time.h>

#include <fstream>

#include <sstream>

#include "qsort.h"

#include "random.h"

#include "psort.h"

using namespace std;

const int limit=100;

int main()

{

unsigned int n\_P, n\_B;

ofstream ishod1, ishod2, result1, result2;

ishod1.open("ishod1.txt");

if (!ishod1.is\_open())

{

return 11; // выход в случае неудачи открытия файла для записи первого исходного массива

}

cout << "Введите размерность масива для пузырька: " << endl;

cin >> n\_P;

int \*array\_P = new int[n\_P];

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

for(int i=0; i<n\_P; i++)

{

/\*\*

Вызов функции заполнения массива рандомным числом

\*/

array\_P[i] = random(-limit, limit);

}

ishod1 << "Сгенерированный массив для пузырьковой сортировки: " << endl;

for (int i=0; i<n\_P; i++)

{

ishod1 << array\_P[i] << " ";

}

ishod1.close();

double t1 = clock();

/\*\*

Вызов функции сортировки пузырьком

\*/

psort(array\_P, n\_P);

cout << "Время сортировки пузырьком: " << (clock()-t1) / CLOCKS\_PER\_SEC << endl;

result1.open("result1.txt");

if (!result1.is\_open())

{

return 12; // выход в случае неудачи открытия файла для записи результата сортировки пузырьком

}

result1 << "Массив, отсортированный пузырьком: " << endl;

for (int i=0; i<n\_P; i++)

{

result1 << array\_P[i] << " ";

}

result1.close();

delete [] array\_P;

cout << "Введите размерность масива для быстрой сортировки: " << endl;

cin >> n\_B;

int \*array\_B = new int[n\_B];

for(int i=0; i<n\_P; i++)

{

/\*\*

Вызов функции заполнения массива рандомным числом

\*/

array\_B[i] = random(-limit, limit);

}

ishod2.open("ishod2.txt");

if (!ishod2.is\_open())

{

return 13; // выход в случае неудачи открытия файла для записи второго исходного массива

}

ishod2 << "Сгенерированный массив для быстрой сортировки: " << endl;

for (int i=0; i<n\_B; i++)

{

ishod2 << array\_B[i] << " ";

}

ishod2.close();

double t2 = clock();

/\*\*

Вызов функции быстрой сортировки

\*/

qsort(array\_B, 0, n\_B-1);

cout << "Время быстрой сортировки: " << (clock()-t2) / CLOCKS\_PER\_SEC << endl;

result2.open("result2.txt");

if (!result2.is\_open())

{

return 14; // выход в случае неудачи открытия файла для записи результата быстрой сортировки

}

result2 << "Массив, отсортированный быстрой сортировкой: " << endl;

for (int i=0; i<n\_P; i++)

{

result2 << array\_P[i] << " ";

}

result2.close();

delete [] array\_B;

return 0;

}

**“random.h”:**

/\*\*

@file random.h

@author Samoletov Petr Y2333 <samoletovp@gmail.com>

@brief Функция заполнения массива рандомным числом

\*/

#ifndef random\_h

#define random\_h

#endif /\* random\_h \*/

/\*\*

Функция заполнения элемента массива случайным числом

@param range\_min минимальное число для заполнения

@param range\_max максимальное число для заполнения

@return рандомное число

\*/

int random(int range\_min, int range\_max)

{

return rand() % (range\_max - range\_min + 1) + range\_min;

}

**“psort.h”:**

/\*\*

@file psort.h

@author Samoletov Petr Y2333 <samoletovp@gmail.com>

@brief Функция сортировки пузырьком

\*/

#ifndef psort\_h

#define psort\_h

#endif /\* psort\_h \*/

/\*\*

Функция сортировки пузырком

@param array\_P массив для сортировки пузырьком

@param n\_P количество элементов массива для сортировки пузырьком

\*/

void psort(int \*array\_P, int n\_P)

{

int temp;

for (int i=0; i<n\_P; i++)

{

for(int j=0; j<n\_P; j++)

{

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

{

temp = array\_P[j];

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

array\_P[j + 1] = temp;

}

}

}

}

**“qsort.h”:**

/\*\*

@file qsort.h

@author Samoletov Petr Y2333 <samoletovp@gmail.com>

@brief Функция быстрой сортировки

\*/

#ifndef qsort\_h

#define qsort\_h

#endif /\* qsort\_h \*/

/\*\*

@param array\_B передаваемый массив для быстрой сортировки

@param left крайний левый элемент

@param right крайний правый элемент

\*/

void qsort(int \*array\_B, int left, int right)

{

int pivot;

int l\_hold=left;

int r\_hold=right;

pivot=array\_B[left];

while (left<right)

{

while ((array\_B[right] >= pivot) && (left < right))

{

right--;

}

if (left != right)

{

array\_B[left] = array\_B[right];

left++;

}

while ((array\_B[left] <= pivot) && (left < right))

{

left++;

}

if (left != right)

{

array\_B[right] = array\_B[left];

right--;

}

}

array\_B[left]=pivot;

pivot=left;

left=l\_hold;

right=r\_hold;

if (left < pivot)

{

qsort(array\_B, left, pivot - 1);

}

if (right > pivot)

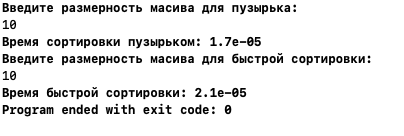
{

qsort(array\_B, pivot + 1, right);

}

}

**Результат работы программы.**

Протокол программы представлен на рисунке 1.

*Рисунок 1 – Протокол программы*

Изображение выглядит как снимок экрана

Автоматически созданное описаниеФайлы с записанными и отсортированными массивами представлены на рисунке 2.

*Рисунок 2 – Файлы с массивами*