**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по лабораторной работе №1**

**по дисциплине «Программирование»**

ТЕМА: СОРТИРОВКА.

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Ильяшук Д.И. |
| Преподаватель |  | Берленко Т. А. |

Санкт-Петербург

2017

**Цель работы:**

Научиться использовать алгоритмы сортировки элементов массива, определить время работы каждого из них, сравнить быстродействие алгоритмов.

**Задание:**

Написать программу, на вход которой подается массив целых чисел длины **1000**.

Программа должна совершать следующие действия:

* отсортировать массив с помощью алгоритма "сортировка пузырьком"
* посчитать время, за которое будет совершена сортировка, используя при этом **функцию стандартной библиотеки**
* отсортировать массив с помощью алгоритма "быстрая сортировка" (quick sort), используя при этом **функцию стандартной библиотеки**
* посчитать время, за которое будет совершена сортировка, используя при этом **функцию стандартной библиотеки**
* вывести отсортированный массив (элементы массива должны быть разделены пробелом)
* вывести время, за которое была совершена сортировка пузырьком
* вывести время, за которое была совершена быстрая сортировка

**Ход работы:**

1. Была объявлена константа LENGTH, которая отвечает за количество элементов в массиве.

#define LENGTH 1000

1. Был написан алгоритм считывания элементов массива.

for (i = 0; i < LENGTH; i++)

scanf("%d", &array1[i]);

1. Массив с числами был продублирован для второго алгоритма сортировки.

memcpy(array2, array1, LENGTH \* sizeof(int));

1. Затем был написан алгоритм сортировки «пузырьком», при этом перед началом и после сортировки были сохранены количества тактов процессора с момента запуска программы.

sort\_start = clock();

for (i = 0; i < (LENGTH - 1); i++)

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

if (array1[j] > array1[j + 1])

{

int tmp = array1[j];

array1[j] = array1[j + 1];

array1[j + 1] = tmp;

}

sort\_end = clock();

1. Далее была написана функция сравнения для быстрой сортировки compare и вызвана сама функция быстрой сортировки. Перед началом и после быстрой сортировки были сохранены количества тактов процессора с момента запуска программы.

int compare(const void \* a, const void \* b)

{

return (\*(int\*)a - \*(int\*)b);

}

1. Далее был написан алгоритм вывода вывода элементов массива, элементы выводятся в порядке возрастания т.к. уже отсортированы.

for (i = 0; i < LENGTH; i++)

printf("%d ", array2[i]);

1. Далее производится подсчет времени работы циклов с помощью константы CLOCKS\_PER\_SEC и вывод этого времени в секундах.

printf("%f\n", (float)(sort\_end - sort\_start) / CLOCKS\_PER\_SEC);

printf("%f\n", (float)(qsort\_end - qsort\_start) / CLOCKS\_PER\_SEC);

1. Затем файл с лабораторной и отчёт работой был загружен на github c помощью консоли:

* Создана новая ветка с помощью команды *git checkout –b ilyashuk\_sem2\_lab1*
* Добавлены файлы для загрузки с помощью команды *git add lab1.c otchet.docx*
* Добавлен комментарий коммита командой *git commit –m “lab1”*
* Файлы были загружены командой *git push origin ilyashuk\_sem2\_lab1*

**Вывод:** В ходе лабораторной работы получены навыки работы с сортировкой массивов разными методами. На практике определено что скорость сортировки функцией qsort значительно выше чем методом «пузырька».