**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 |  | Ваганов Н.А. |
| 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. Программа считывает элементы массива.

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

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

1. Затем сортирует их «пузырьковым» алгоритмом. Используя стандартную функцию time(NULL) , сохраняем время, затраченное на сортировку.

bubblesort\_start = clock();

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

{

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

{

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

{

int tmp = numbers[j];

numbers[j] = numbers[j + 1];

numbers[j + 1] = tmp;

}

}

}

bubblesort\_end = clock();

1. Далее используется функция сравнения для быстрой сортировки compare и вызывается сама функция быстрой сортировки. Перед сортировкой и после нее было сохранено время с момента запуска программы.

int compare(const void \* x1, const void \* x2)

{

return (\*(int\*)x1 - \*(int\*)x2);

}

1. Программа выводит уже отсортированные элементы массива.

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

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

1. Далее выводится время, затраченное на сортировку (в секундах). В первой строке время, затраченное на сортировку «пузырьком», во второй – функцией qsort.

printf("%f\n", (float)(bubblesort\_end - bubblesort\_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 Vaganov\_sem2\_lab1*
* Добавлены файлы для загрузки с помощью команды *git add lab1.c otchet.docx*
* Добавлен комментарий коммита командой *git commit –m “vaganovlab1”*
* Файлы были загружены командой *git push origin Vaganov\_sem2\_lab1*

**Вывод:** В ходе лабораторной работы получены навыки работы с сортировкой массивов разными методами. Определено, что в данном случае скорость сортировки функцией qsort значительно выше чем методом «пузырька».