**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра МО ЭВМ**

# ОТЧЕТ

**по лабораторной работе №1**

**по дисциплине «Программирование»**

**Тема: Обзор стандартной библиотеки**

Студент гр. 1304 _____ Мусаев А. И.

Преподаватель _____ Чайка К. В.

Санкт-Петербург

2022

**Цель работы.**

Изучить сортировку qsort и её время работы с помощью функций стандартной библиотеки time.h.

**Задание.**

Вариант 4.

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

Программа должна совершать следующие действия:

- отсортировать массив по невозрастанию модулей элементов 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 быстрая сортировка

*Отсортированный массив, время быстрой сортировки должны быть выведены с новой строки, при этом элементы массива должны быть разделены пробелами.*

**Выполнение работы.**

В начале кода объявляются 4 стандартные библиотеки: stdio.h, stdlib.h, time.h. Из этих библиотек мы будем использовать функцию abs(), которая возвращает модуль числа и clock(), чтобы замерить время работы функции.

В функции main() происходит объявление массива arr[n] типа int, переменной i типа int, которая будет выполнять роль счётчика в цикле и переменной n, которая определяет размер массива. Далее следует объявление переменных start_time и end_time типа unsigned int. Затем происходит считывание входных данных с помощью цикла. После

считывания массива переменной start_time присваивается количество временных тактов процессора с начала работы программы. Далее вызывается функция qsort. В качестве аргументов сортировка принимает массив arr, количество элементов в размере n, размер одного элемента массива (тип int) и компаратор cmp.

В функции cmp сравниваемые элементы приводятся к типу указатель на константное значение типа int. Далее происходит разыменование обоих элементов и сравнение их по модулю. Если модуль первого элемента больше модуля второго, то функции возвращается значение -1. Если модуль первого элемента меньше модуля второго, то функции возвращается значение 1. При равенстве модулей обоих элементов функции возвращается 0.

После завершения работы функции qsort переменной end_time присваивается количество временных тактов процессора с начала работы программы. Далее выводится отсортированный массив с помощью цикла. В следующей строке выводится время работы функции qsort согласно следующей формуле: (end_time – start_time).

**Тестирование.**

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

| № п/п | Входные данные | Выходные данные | Комментарии |
|---|---|---|---|
| 1. | 1 2 3 4 5 6 7 8 9 10 | 10 9 8 7 6 5 4 3 2 1<br>0.000004 | Вводилось 10 значений для удобства тестирования<br>Ответ верный |
| 2. | 1 2 3 4 5 -1 -2 -3 -4 -5 | 5 -5 4 -4 3 -3 2 -2 1 -1<br>0.000005 | Вводилось 10 значений для удобства тестирования<br>Ответ верный |
| 3. | -100 9 8 6 50 -90 7 -1000 10000 67 8 | 10000 -1000 -100 -90 67 50 | Вводилось 10 значений для удобства |

| | | | |
|---|---|---|---|
| | | 9 8 7 6<br>0.000006 | тестирования<br>Ответ верный |
| 4. | 100 9 8 6 50 -90 7 -1000 10000 67 8 1 2 3 4 5 -1 -2 -3 -4 -5 | 10000 -1000 100 -90 67 50 9 8 8 7 6 5 4 -4 3 -3 2 -2 1 -1<br>0.000012 | Вводилось 20 значений для удобства тестирования<br>Ответ верный |
| 5. | 100 9 8 6 50 -90 7 -1000 10000 67 8 1 2 3 4 5 -1 -2 -3 -4 -5 1 2 3 4 5 6 7 8 9 10 | 10000 -1000 100 -90 67 50 9 9 8 8 8 7 7 6 6 5 -5 5 4 -4 4 3 -3 3 2 -2 2 1 -1 1<br>0.000007 | Вводилось 30 значений для удобства тестирования<br>Ответ верный |

**Выводы.**

Была написана программа, сортирующая массив по невозрастанию модулей и считающая время своей работы.

## ПРИЛОЖЕНИЕ А
## ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
int cmp(const void* a, const void* b){

    const int *f = (const int*)a;
    const int* s = (const int*)b;
    if (abs(*f) > abs(*s))
        return -1;
    if (abs(*f) < abs(*s))
        return 1;
    return 0;
}

int main(){

    int arr[1000];
    int i;
    clock_t start, end;
    for(i = 0; i < 1000; i++)
        scanf("%d", &arr[i]);
    start = clock();
    qsort(arr, 1000, sizeof(int), cmp);
    end = clock();
    for(i = 0; i < 1000; i++)
        printf("%d ", arr[i]);
    printf("\n");
    printf("%f\n", (float)(end-start)/CLOCKS_PER_SEC);
    return 0;
}
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int cmp(const void* a, const void* b)
{
    const int* k = (const int*)a;
    const int* f = (const int*)b;
    if (abs(*k)>abs(*f)){
        return -1;
    }
    if (abs(*k)<abs(*f)){
        return 1;
    }
    return 0;
}

int main()
{
    int i;
    int n=1000;
```

```
    int a[n];
    for(i=0;i<n;i++){
        //a[i]=n-i;
        scanf("%d",&a[i]);
    }
    unsigned int start_time=clock();
    qsort(a,n,sizeof(int),cmp);
    unsigned int end_time=clock();
    for(i=0;i<n;i++){
        printf("%d ",a[i]);
    }
    printf("%c",'\n');
    printf("%ld",end_time-start_time);
}
```