**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 _____ Павлов Д. Р.

Преподаватель _____ Чайка К. В.

Санкт-Петербург
2022

**Цель работы.**

Целью данной лабораторной работы является изучение стандартных функций библиотеки time.h и использование функции сортировки qsort.

**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.**

Сначала объявляются 3 стандартные библиотеки: stdio.h, time.h, stdlib.h — и, для собственного удобства, константу SIZE 1000 (размер массива). Далее идет функция print_arr, которая отвечает за вывод массива. У нее есть два аргумента — int* arr(сам массив) и int max_size(его размер). Внутри нее пишется цикл for (int i = 0; i < max_size; i++){printf(«%d», arr[i])}, пока i

меньше max_size — увеличиваем i и выводим элемент массива. Далее пишется функция scan_array, с аргументом int max_size. Это функция отвечает за ввод массива. Она так же возвращает массив. Внутри нее объявляется массив int* scaned_arr — место, куда мы будем записывать целые числа. Далее выделяется память для массива. Далее мы пишем цикл, который будет посимвольно выделять память в массиве и заносить введенное число в этот массив. После цикла возвращается данный массив.

Далее идет компаратор который будет отвечать за конкретный процесс сортировки при помощи qsort. Он возвращает целое число и его аргументы — i и j. Фактически, данная функция проверяет, если ли разность между числами отрицательная или положительная.

Далее идет функция main. Сначала инициализируется две переменные типа clock_t start и end. Они отвечают за время до сортировки и после; приравниваем start к функции clock(). Далее инициализируем массив целых чисел — array и приравниваем его возвращаемому значению функции scan_array с аргументом SIZE. Далее сортируем массив при помощи функции qsort и функции-компаратора. После приравнивается end к функции clock(). После выводится массив при помощи функции print_arr. Далее выводится разность тактов и делим ее на макрос CLOCKS_PER_SEC. Таким образом получится время работы сортировки в секундах.

**Тестирование.**

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

| № п/п | Входные данные | Выходные данные | Комментарии |
|---|---|---|---|
| 1. | 2424 243 5 632 65 753664 64565 56464 757 6536464 | 6536464 753664 64565 56464 2424 757 632 243 65 | True |

| | | | |
|---|---|---|---|
| | | 5<br>0 | |
| 2. | 999 222 111 555 444 666 333 777 888 101010 | 101010 999 888 777 666 555 444 333 222 111<br>0 | True |
| 3. | 1 2 3 4 5 6 7 8 9 10 | 10 9 8 7 6 5 4 3 2 1<br>0 | True |
| ... | | | |

**Выводы.**

Была написана программа, сортирующая массив в порядке убывания и считающая время своей работы.

# ПРИЛОЖЕНИЕ А
# ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <math.h>
#define SIZE 1000

void print_arr(int* arr, int max_size){
    for (int i = 0; i < max_size; i++){
        printf("%d ", arr[i]);
    }
}

int* scan_array(int max_size){
    int* scaned_arr;
    scaned_arr = malloc(sizeof(int));

    for (int i = 0; i < max_size; i++){
        scaned_arr = realloc(scaned_arr, sizeof(int)*(i+1));
        scanf("%d", &scaned_arr[i]);
    }

    return scaned_arr;
}

int cmp(const void *i, const void *j)
{
    return *(int *)j - *(int *)i;
}


int main() {
    clock_t start, end;
    start = clock();
    int *array = scan_array(SIZE);
    qsort(array, SIZE, sizeof(int),  (int(*) (const void *, const
void *)) cmp);
    end = clock();
    print_arr(array, SIZE);
    //printf("\n0");
    printf("\n%lu", (end - start) / (CLOCKS_PER_SEC));

    return 0;
}
```