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 _____ Маркуш А.Е.

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 3.

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

Программа должна совершать следующие действия:

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

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

Выполнение работы.

В функции main() происходит объявление массива arr[1000] типа int и его заполнение с помощью цикла for. Далее следует объявление переменных start и end типа time_t. После переменной start присваивается текущее календарное время. Далее реализуется пузырьковая сортировка с помощью вложенных циклов for. После сортировки переменной end присваивается текущее календарное время, затем объявляется переменная bubble_time типа double, которой присваивается значение функции difftime(start, end), считающую разность между двумя переменными типа time_t. После

переменной start снова присваивается текущее календарное время. Далее вызывается функция qsort. В качестве аргументов сортировка принимает массив arr, количество элементов в размере 1000, 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. Если первый элемента больше второго, то функции возвращается значение 1. Если модуль первого элемента меньше модуля второго, то функции возвращается значение -1. При равенстве обоих элементов функции возвращается 0.

После завершения работы функции qsort переменной end присваивается каледарное время. Затем объявляется переменная double quick_time, которой присваивается результат функции difftime(start, end). Далее выводится отсортированный массив с помощью цикла for. На следующей строке выводится время пузырьковой сортировки, а после с новой строки время работы функции qsort. Программа заканчивается возвращением значения 0 функции main().

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	1 3 4 5 2 6 3 -5 4 -10	-10 -5 1 2 3 3 4 4 5 6 0.000004 0.000004	Вводилось 10 значений для удобства тестирования Ответ верный
2.	-2 29 43 -34 23 324 5 -234 3 5	-234 -34 -2 3 5 5 23 29 43 324 0.000005 0.000004	Вводилось 10 значений для удобства тестирования Ответ верный
3.	345 5 3 2 32 -324 432 2 3 -32	-324 -32 2 2 3 3 5 32 345 432 0.000006	Вводилось 10 значений для удобства

		0.000006	тестирования Ответ верный

Выводы.

Была написана программа, сортирующая массив по возрастанию с помощью двух видов сортировки и считающая время работы сортировок.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#define N 10

int compare(const void* num1, const void* num2){
    const int* a = (const int*)num1;
    const int* b=  (const int*)num2;

    if(*a > *b){
        return 1;
    }
    else if(*a < *b){
        return -1;
    }
    return 0;
}


int main(){
    int arr[N];
    for(int i = 0; i < N; i++){
        scanf("%d", &arr[i]);
    }

    time_t start, end;

    start = time(NULL);
    for(int i = 0; i < N; i++){
        for(int j = 0; j < N - i -1; j++){
            if(arr[j] > arr[j+1]){
                int buffer = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = buffer;
            }
        }
    }
    end = time(NULL);
    double bubble_time = difftime(end, start);

    start = time(NULL);
    qsort(arr,  N, sizeof(int),compare);
    end = time(NULL);
    double quick_time = difftime(end, start);

    for(int i = 0; i < N; i++){
        printf("%d ", arr[i]);
    }
    printf("\n%lf", bubble_time);
    printf("\n%lf", quick_time);

    return 0;
}
```