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

Студентка гр. 0382 _____ Чегодаева Е.А.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2021

Цель работы.

Изучение стандартной библиотеки Си.

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.

Основные теоретические положения.

Библиотека stdlib.h:

- time.h

В заголовочном файле time.h можно найти объявления типов и функций для работы с датой и временем. В том числе:

- Функция, позволяющая получить текущее календарное время
- Функция, позволяющая получить время в тактах процессора с начала выполнения программы
- Функция для вычисления разности в секундах между двумя временными штампами
- Функции для вывода значения даты и времени на экран

А также структура tm, содержащая компоненты календарного времени и функция для преобразования значения времени в секундах в объект такого типа.

- qsort

```
void qsort (void* base, size_t num, size_t size,
        int (*compar)(const void*,const void*));
```

Функция принимает указатель на начальный элемент массива, количество элементов и размер одного элемента, а также указатель на функцию для сравнения двух элементов.

Так как тип элементов может быть любым, то и указатель на первый элемент массива имеет тип void. Это позволяет, зная адрес первого элемента и размер каждого элемента вычислить адрес любого элемента массива в памяти и обратиться к нему. Остается только сравнить 2 элемента имея 2 указателя на них. Это выполняет функция compar, указатель на которую передается функции qsort в качестве одного из параметров.

Функция compar принимает 2 указателя типа void, но в своей реализации может привести их к конкретному типу (так как её реализация остается за программистом, он точно знает элементы какого типа он сортирует) и сравнивает их. Результат сравнения определяется знаков возвращаемого функций qsort числа.

Выполнение работы.

В функцию *main(){}* посредством цикла *for()* и функции *scanf()* подаётся целочисленный массив *arr* размера *N (N = 1000)*. Далее при помощи *qsort* реализован алгоритм необходимой сортировки. В функции *cmp(){}* (являющейся одним из параметров *qsort'a*) указатели, передающиеся в функцию, преобразовываются в указатели на целое число (переменные *a* и *b* хранят два числа заданного массива) и далее посредством *abs* функция возвращает либо отрицательное число (в таком случае “первый ” поданный элемент встаёт на место “второго”), либо положительное (в таком случае оба числа сохраняют свои позиции), либо 0 (что свидетельствует о равенстве чисел), следовательно, после реализации данного алгоритма мы получаем массив, отсортированной по невозрастанию модулей элементов. Вместе с тем, в переменные *time1* и *time2* — типа *clock_t*, который определён в *time.h* — записываются результаты, возвращаемые функцией *clock()* (количество временных тактов, прошедших с

начала запуска программы до начала сортировки и после окончания сортировки соответственно). При помощи макроса *CLOCKS_PER_SEC* и разницы *time2* и *time1*, приведённых к типу *int,* в переменную *t*, объявленную ранее, записывается количество секунд, за которое была произведена сортировка. Затем при помощи функции *printf()* реализован вывод данных, в заданном формате.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	*При #define N 10 1 2 3 4 5 6 8 9 10	10 9 8 7 6 5 4 3 2 1 0	Ответ верный.
2.	*При #define N 10 -1 2 3 -4 5 -6 7 -8 9 -10	-10 9 -8 7 -6 5 -4 3 2 -1 0	Ответ верный.
3.	*При #define N 25 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5	5 5 5 5 5 4 4 4 4 4 3 3 3 3 3 2 2 2 2 2 1 1 1 1 1 0	Ответ верный.
4.	*При #define N 15 1 1 -1 1 1 10 10 -10 10 10 5 5 -5 5 5	10 10 10 -10 10 5 5 5 -5 5 1 1 1 -1 1 0	Ответ верный.
5.	*При #define N 10 24 33 64 10 7 105 333 843 8 32	843 333 105 64 33 32 24 10 8 7 0	Ответ верный.

Выводы.

Была изучена стандартная библиотека Си.

Разработана программа, выполняющая сортировку (quick sort) по невозрастанию модулей элементов введённого массива, а так же

подсчитывающая время, за которое была совершена данная сортировка. Для реализации были применены функции стандартной библиотеки.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb1.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define N 1000
int cmp(const void* it1, const void* it2){
   int a = *(int *)it1;
   int b = *(int *)it2;
   return abs(b)-abs(a);
}
int main(){
   int t = 0;
   int arr[N];
   for (int i = 0; i<N; i++){
      scanf("%d", &arr[i]);
   }
   clock_t time1 = clock();
   qsort(arr, N, sizeof(int), cmp);
   clock_t time2 = clock();
   t = ((int)time2-(int)time1)/CLOCKS_PER_SEC;
   for (int j = 0; j<N; j++){
      printf("%d", arr[j]);
      printf(" ");
   }
   printf("\n%d", t);
   return 0;
}
```