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 языка Си

Студентка гр. 0382 _____ Охотникова Г.С.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2021

Цель работы.

Освоить возможности стандартной библиотеки языка Си, научиться применять на практике основные функции.

Задание.

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.

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.

В заголовочном файле *stdlib.h* собраны объявления различных функций, частью из которых мы уже пользовались ранее.

- функции для работы с динамической памятью
- функции для преобразования строки в число
- генерации псевдослучайных чисел
- функции для управления процессом выполнения программы
- функции для вычисления абсолютного значения и деления целых чисел
- функции для сортировки и поиска

Для написания функции нахождения минимума в массиве элементов неизвестного типа указатель на функцию нужен, чтобы сравнивать элементы. Похожая логика используется во многих языках программирования, а функцию сравнения двух элементов обычно называют компаратор (англ *compare* - сравнивать). Компаратор работает по следующему принципу: если элементы равны, результатом сравнения будет 0, если первый больше - результат 1 или любое положительное число, иначе -1 или любое отрицательное.

В *stdlib.h* есть функции для сортировки и поиска в массиве любого типа. Давайте рассмотрим как такое возможно на примере функции *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.

Макрос:

- *#define C 100* — количество вводимых чисел.

Переменные:

- *int i* — счетчик для ввода массива чисел.
- *int arr[C]* — массив чисел.
- *clock_t start_time* — счетчик времени для начала отсчета.
- *clock_t end_time* — счетчик времени для окончания отсчета.

Функция *compfunction1(const void* a1, const void* a2)* является функцией-компаратором для быстрой сортировки массива чисел *qsort()*. Она принимает на вход указатели на два числа и при помощи оператора сравнения производит сортировку по убыванию.

В функции *main()* в цикле *for* вводится массив целых чисел. Затем применяется функция *clock()*, которая помогает получить количество временных тактов, прошедших с начала выполнения программы. Происходит вызов функции *qsort()* для сортировки массива чисел, после этого еще раз используется функция *clock()* и переменной *end_time* присваивается разность значения, полученного помощью функции *clock()*, и времени, полученного до начала сортировки и сохраненного в переменную *start_time*. Отсортированный массив выводится на экран. Также выводится время выполнения сортировки с использованием макроса *CLOCKS_PER_SEC* для получения тактов за одну секунду.

Разработанный программный код см. в приложении А.

Тестирование.

Тестирование представлено на примерах массивов из 10 и 5 элементов.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	-3 2 1 8 10 -2 013 9 6	13 10 9 8 6 -3 -2 2 1 0 0.000002	Программа работает верно.
2.	-9 -1 -4 -2 -21 -5 -4 -8 -7 -10	-21 -10 -9 -8 -7 -5 -4 -4 -2 -1 0.000020	Программа работает верно.
3.	4 2 7 1 9	9 7 4 2 1 0.000010	Программа работает верно.

Выводы.

Было исследованы возможности стандартной библиотеки языка Си, изучены основные ее функции.

Разработана программа, выполняющая считывание с клавиатуры массива целых чисел. Задача была решена с помощью функции *clock()* из заголовочного файла *time.h* и с помощью функции *qsort().*

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb4.c

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

int compfunction1(const void* a1, const void* a2){
    return abs(*(int*)a1) <= abs(*(int*)a2);
}

int main() {
    int i;
    int arr[C];
    for (i = 0; i < C; i++) {
        scanf("%d", &arr[i]);
    }
    clock_t start_time;
    start_time = clock();
    qsort(arr,   C,   sizeof(int),   (int(*)(const   void*,   const
void*))compfunction1);
    clock_t end_time;
    end_time = clock() - start_time;
    for (i = 0; i < C; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n%f", (double)end_time/CLOCKS_PER_SEC);
    return 0;
}
```