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. 0382 _____ Ильин Д.А.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2021

Цель работы.

Изучить и освоить функционал стандартной библиотеки языка программирования Си.

Задание.

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

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.

Основные теоретические положения.

void qsort (void base, size_t num, size_t size, int (*compar)(const void*,const void*)) из stdlib.h*

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.

clock_t clock(void) из time.h

Возвращает количество временных тактов, прошедших с начала запуска программы. С помощью макроса *CLOCKS_PER_SEC* функция получает количество пройденных тактов за 1 секунду. Таким образом, зная сколько выполняется тактов в секунду, зная время запуска программы можно посчитать время работы всей программы или отдельного её фрагмента, что и делает данная функция.

Выполнение работы.

В программе имеются функция compare() (вспомогательная функция, для qsort()), а так же функция main().

Сначала считываются числа в массив длинны 1000, заполненный нулями изначально. Далее записывается время в переменную time_s1, после чего выполняется «сортировка пузырьком» полученного массива, после чего рассчитывается время данной сортировки и сохраняется в переменную time_e1.

После чего аналогично записываем время в переменную time_s2, выполняем сортировку при помощи qsort(), считаем сколько времени было на это затрачено и записываем его в переменную time_e2.

В конце производится вывод элементов массива чисел, при помощи цикла for().

Выводы.

Были изучен и освоен функционал стандартной библиотеки языка программирования Си.

Разработана программа, показывающая скорость поиска путём бинарного поиска и поиска полным перебором.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb1.c

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

int compare(const void * x1, const void * x2)
{
  return ( *(int*)x1 - *(int*)x2 );
}

int main() {
    double time_s1, time_e1, time_s2, time_e2, help;
    int list [1000] = {0};
    for(int i = 0; i < 1000; i++){
        scanf("%d", &list[i]);
    }
    time_s1 = clock();
    for(int i = 0; i < 999; i++){
        for(int j = i; i < 1000; i++){
            if (list[i] > list[j]){
                int a = list[i];
                list[i] = list[j];
                list[j] = a;
            }
        }
    }
    time_e1 = (clock() - time_s1)/CLOCKS_PER_SEC;
    time_s2 = clock();
    qsort(list, 1000, sizeof(int), compare);
```

```
    time_e2 = (clock() - time_s2)/CLOCKS_PER_SEC;
    for(int i = 0; i<1000; i++){
        printf("%d ", list[i]);
    }
    printf("\n%o\n%o", time_e1, time_e2);
    return 0;
}
```