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 Докучаев Р.А.

Преподаватель Чайка К.В.

Санкт-Петербург

2021

Цель работы.

Изучить функции стандартной библиотеки языка С, описанные в заголовочных файлах *stdio.h*, *stdlib.h*, *time.h*, и решить поставленную задачу при помощи функции стандартной библиотеки.

Задание.

Вариант №3.

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.

Для решения поставленной задачи необходимо были использованы следующие функции, описанные в заголовочных файлах *stdio.h*, *stdlib.h* и *time.h*: *clock()* – получение процессорного времени, за которое выполняется часть вышеописанного кода; *qsort()* – выполнения быстрой сортировки по заданному условию; *printf()* – вывод результата. Также необходимо реализовать в задаче *пузырьковую сортировку*: самый простой алгоритм сортировки, который работает путем многократной замены соседних элементов, если они находятся в неправильном порядке.

Выполнение работы.

1. Подключаем заголовочные файлы *stdio.h*, *stdlib.h*, *time.h* и создаём константу *ARR_SIZE*, которая отвечает за размер массива, при помощи директивы *define*.
2. Описывается функция типа *void* для пузырьковой сортировки *bubble_sort*.
3. Описывается функция comp, по которой будет проводится сортировка в *qsort*`е. Функция получает на вход два элемента массива, а возвращает разницу между ними либо 0.
4. Написание основной части программы.
   1) Создание переменных *t0*, *t1*, *t2* типа *clock_t* для записи в них результата функции *clock()*;
   2) Создание двух массивов *arr* и *temp*, размеров *ARR_SIZE*, которые и будут сортироваться;
   3) При помощи цикла *for* осуществляется запись значений в массив *arr* и копирование полученных значений в массив *temp*;
   4) В *t0* записывается процессорное время, прошедшее с начала работы программы;
   5) Осуществляется сортировка массива temp при помощи функции *bubble_sort*;
   6) Записывается время работы функции в переменную *t1*;
   7) В *t0* снова записывается процессорное время, прошедшее с начала работы программы;
   8) Осуществляется сортировка массива *arr* при помощи функции *qsort()*;
   9) В *t2* записывается время работы функции;
5. В конце основной части программы при помощи функции *for* происходит вывод отсортированного массива, за циклом выводится время работы обеих функций.

Вывод.

В ходе выполнения работы были изучены и использованы функции стандартной библиотеки, такие как *clock()* и *qsort()*, а также создана функция пузырьковой сортировки.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название исходного файла: laba1.c

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

void bubble_sort(int* arr){
for(int i = 1; i < ARR_SIZE; i++){
    int k = i;
    while(k > 0 && arr[k-1] > arr[k]){
         int tmp = arr[k-1];
         arr[k-1] = arr[k];
         arr[k] = tmp;
         k--;
         }
    }
}

int comp(const void* a, const void* b){
return(*(int*) a - *(int*) b);
}

int main(){
clock_t t0, t1, t2;
int arr[ARR_SIZE];
int temp[ARR_SIZE];
for(int i = 0; i < ARR_SIZE; i++){
    scanf("%d", &arr[i]);
    temp[i] = arr[i];
    }
t0 = clock();
bubble_sort(temp);
t1 = clock() - t0;
t0 = clock();
qsort(arr, ARR_SIZE, sizeof(int), comp);
t2 = clock() - t0;
for(int i = 0;i < ARR_SIZE;i++){
    printf("%d ", arr[i]);
    }
printf("\n");
printf("%f\n", ((float)t1 / CLOCKS_PER_SEC));
printf("%f\n", ((float)t2 / CLOCKS_PER_SEC));
return 0;
}
```