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 _____ Деткова А.С.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2021

Цель работы.

Познакомиться со стандартной библиотекой языка Си, её заголовочными файлами, основными функциями. Научиться применять функции заголовочных файлов в зависимости от ситуации.

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.

Стандартной библиотекой языка Си (также известная как libc, crt) называется часть стандарта ANSI C, посвященная заголовочным файлам и библиотечным подпрограммам. Является описанием реализации общих

операций, таких как обработка ввода-вывода и строк, в языке программирования Си. Стандартная библиотека языка Си — это описание программного интерфейса, а не настоящая библиотека, пригодная для использования в процессе компиляции.

В программе использовалось 3 заголовочных файла стандартной библиотеки языка Си:

1. <*stdio.h*> - содержит объявления функций ввода-вывода, а также функций для файловых операций. Использованные функции: *printf()* - форматный вывод, *scanf()* - форматный ввод.
2. <*stdlib.h*> - содержит объявления функций, которые занимаются преобразованием типов, генерацией псевдослучайных последовательностей, выделением и освобождением памяти, котролем процесса выполения программы, сортировкой и поиском, математическими операциями, многобайтовыми операциями/широкими символами. Использованные функции: *qsort()* - быстрая сортировка массива.
3. <*time.h*> - содержит типы и объявления функций для работы с датой и временем. Использованная функция: *clock()* - находит количество тактов процессора для выполнения программы. Использованная константа: *CLOCKS_PER_SEC* — определяет количество тактов системных часов в секунду, нужна для пересчета возвращаемого значения функцией *clock()* в секунды.

В программе использовался известный алгоритм: сортировка пузырьком. Простейший и неэффективный алгоритм сортировки.

Выполнение работы.

Функции:

- *void getarray(int * mas, int sz)* - в качестве аргумента получает указатель на массив чисел и его размер, считывает элементы массива и записывает их по адресу массива, ничего не возвращает.
- *double bubblesort(int arr[], int sz)* — в качестве аргумента получает массив целых чисел и его размер, выполняет пузырьковую сортировку и считает время ее выполнения, возвращает это время.
- *int comparr(const void *a, const void *b)* — функция компаратор, которая получает два элемента массива, сравнивая их, возвращает число, которое показывает, необходимо ли меня местами элементы массива.
- *int main()* - главная функция, собирает программу воедино и выводит результаты.

В начале работы программы создается буфер *arr[SIZE]* для считывания чисел размера *SIZE = 1000* (именованная константа, создана для упрощения, чтобы размер массива можно было легко менять). Далее числа считываются с помощью функции *getarray()*.

После происходит сортировка пузырьком через функцию *bubblesort()*, которая также вычисляет время этой сортировки в секундах и оно записывается в переменную *timebubble*. Как вычисляется время сортировки: в функции *bubblesort()* создается переменная *start* типа *clock_t*, в которой будет храниться счетчик в тактах процессора в начале выполнения сортировки, во время сортировки значение *start* изменится, в конце сортировки создана переменная *end* типа *clock_t*, считает количество тактов после выполнения сортировки. Далее в переменную *t* типа *double* записывается частное от разности количества тактов до сортировки и после (*end-start*) на количество тактов системных часов в секунду (*CLOCKS_PER_SEC*).

Далее выполняется функция *qsort()* - сортирует массив по возрастанию. А также считается время сортировки, аналогично, как и при сортировке

пузырьком. Время сортировки записывается в переменную *timeqsort* типа *double*.

Выводятся результаты на экран.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные (для SIZE=50)	Выходные данные	Комментарии
1.	2 3 4 5 6 4 3 2 1 2 2 20 9 0 9 7 6 3 4 5 6 7 8 7 6 5 1 2 9 0	0 0 1 1 2 2 2 2 2 3 3 3 4 4 4 5 5 5 6 6 6 6 7 7 7 8 9 9 9 20 0.000019 0.000011	Программа работает корректно
2.	0 9 7 66 5 55 6 5 4 3 2 1 5 22 4 15 47 23 6 5 0 1 0 4 88 5 2 7 1 5	0 0 0 1 1 1 2 2 3 4 4 4 5 5 5 5 5 5 6 6 7 7 7 9 15 22 23 47 55 66 88 0.000018 0.000011	Программа работает корректно

Выводы.

Была изучена стандартная библиотека языка Си, ее заголовочные файлы и функции.

Разработана программа, которая с помощью функций стандартной библиотеки языка Си, а также написанных алгоритмов, вычисляет время в секундах для сортировки пузырьком и быстрой сортировки qsort и выводит отсортированный массив, время сортировок на экран.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb1.c

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#define SIZE 30

void getarray(int * mas, int sz){
    for (int i=0; i<sz; i++){
        scanf(" %d", &mas[i]);
    }
}
double bubblesort(int arr[], int sz){
    clock_t start = clock();
    for (int i=0; i<sz; i++){
        for (int j=0; j<sz-i-1; j++){
            if (arr[j]>arr[j+1]){
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
    clock_t end = clock();
    double t = (double) (end - start)/CLOCKS_PER_SEC;
    return t;
}
int comparr(const void *a, const void *b){
    const int * aa = a;
    const int * bb = b;
    if (*aa > *bb) return 1;
    if (*aa == *bb) return 0;
    if (*aa < *bb) return -1;
}
int main(){
    int arr[SIZE];
    getarray(arr, SIZE);

    double timebuble = bubblesort(arr, SIZE);
    clock_t start = clock();
    qsort(arr, SIZE, sizeof(arr)/SIZE, *comparr);
    clock_t end = clock();
    double timeqsort = (double) (end - start)/CLOCKS_PER_SEC;

    for (int i=0; i<SIZE-1; i++){
        printf("%d ", arr[i]);
    }
    printf("%d\n", arr[SIZE-1]);
    printf("%f\n%f\n", timebuble, timeqsort);

    return 0;
}
```