МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Программирование»

Тема: «Обзор стандартной библиотеки».

Студент гр. 1304 _____ Сенников К.Д.

Преподаватель _____ Чайка К.В.

Санкт-Петербург

2022

Цель работы

Узнать возможности, представляемые стандартной библиотекой языка Си, научиться использовать основные функции стандартной библиотеки.

Задание

Вариант 2

Напишите программу, на вход которой подается массив целых чисел длинны 1000, при этом число 0 либо встречается один раз, либо не встречается.

Программа должна совершать следующие действия:

- отсортировать массив, используя алгоритм быстрой сортировки (см. функции стандартной библиотеки) определить, присутствует ли в массиве число 0, используя алгоритм двоичного поиска (для реализации алгоритма двоичного поиска используйте функцию стандартной библиотеки)
- посчитать время, за которое совершен поиск числа 0, используя при этом функцию стандартной библиотеки
- вывести строку "*exists*", если ноль в массиве есть и "*doesn't exist*" в противном случае
- вывести время, за которое был совершен двоичный поиск
- определить, присутствует ли в массиве число 0, используя перебор всех чисел массива
- посчитать время, за которое совершен поиск числа 0 перебором, используя при этом функцию стандартной библиотеки
- вывести строку "*exists*", если 0 в массиве есть и "*doesn't exist*" в противном случае
- вывести время, за которое была совершен поиск перебором.

Результат двоичного поиска, время двоичного поиска, результат поиска перебором и время поиска перебором должны быть выведены именно в таком порядке и разделены символом перевода строки.

Выполнение работы

Подключение заголовочных файлов *time.h, stdio.h, stdlib.h.* В функции *main* с помощью цикла *for* и функции *scanf* в массив *arr* записываются числа. Затем с помощью функции *qsort* массив *arr* сортируется по возрастанию, используя функцию compare, принимающую на вход две переменные типа

*void**. Внутри функции compare происходит сравнивание двух чисел, поданных на вход и возвращается одно из трех значений: -1 – если первое число меньше второго, 0 – если они равны, 1 – если первое число больше второго. Затем вводится переменная *key* типа *int*, в которой хранится значение 0. Чтобы посчитать время, за которое выполниться операция, использована переменна типа *clock_t time_1,* в которой храниться значение функции *clock*. После производится бинарный поиск с помощью функции *bsearch*, которая тоже использует функцию compare для поиска 0 в массиве. Переменной *time_1* присваивается разность между текущим значением функции *clock* и предыдущим значением этой переменной. Значение, которое возвращает функция *bsearch* записывается в переменную *index* типа *int**. С помощью конструкции *if-else,* которой на вход подается *index,* выводится соответствующая строка *exists* – если значение *index* не равно *NULL*, *doesn't exist* – в ином случае. Затем выводится строка с временем, за которое была совершена данная операция. Значение *time_1* приводится к типу *float* и делится на константу *CLOCKS_PER_SEC.*

Объявляется переменная *flag* типа *int*, значение которой равно 0. Чтобы посчитать время, за которое выполниться операция, использована переменна типа *clock_t time_2,* в которой храниться значение функции *clock.* Производится поиск числа 0 с помощью цикла *for*, и как только оно находится в массиве переменная *flag* принимает значение 1 цикл завершается с помощью оператора *break*. Переменной *time_2* присваивается разность между текущим значением функции *clock* и предыдущим значением этой переменной. С помощью конструкции *if-else,* которой на вход подается *flag,* выводится соответствующая строка *exists* – если значение *flag* не равно 0, *doesn't exist* – в ином случае. Затем выводится строка с временем, за которое была совершена данная операция. Значение *time_2* приводится к типу *float* и делится на константу *CLOCKS_PER_SEC.* Функция main возвращает 0.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования.

№ п/п	Входные данные	Выходные данные	Комментарии
1.	1 2 3 4 5 6 7 8 9 10	doesn’t exist 0.000002 doesn’t exist 0.000001	n = 10
2.	1 2 3 4 5 6 7 8 9 10 11 12 13 -7 -8 -16 -17 -18 -19 -20 -21 -22 -23 -24 0 -25 -26 -27 -28 29 30	exists 0.000002 exists 0.000001	n = 30
3.	1 2 3 4 5 7 32 25 43 -23 32 -44 234 -23 4 35 232 -77 45 12 130 -22 323 34 454 100 -111 34 -34 1 22 34 19 10 231 -23 -19 -123 -22 -40 1 2 3 4 5 7 32 25 43 -23 32 -44 234 -23 4 35 232 -77 45 12 130 -22 323 34 454 100 -111 34 -34 1 22 34 19 10 231 -23 -19 -123 -22 -40 -213 0 32 2 5 7 5 98 5 4 54 34 56 6 -123 34 56 77 99 343	exists 0.000002 exists 0.000001	n = 100

Выводы

В ходе выполнения лабораторной работы были изучены возможности, которые предоставляет стандартная библиотека языка Си. Была написана программа, сортирующая массив из 1000 целых чисел по возрастанию и выясняющая есть ли среди данных чисел 0 с помощью двух вариантов поиска: используя *bsearch* и перебором, - а также выводящая время, за которое были выполнены обе операции.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Sennikov_Kirill_lb1.c

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

int compare(const void* a, const void* b){
    const int* f = (const int*)a;
    const int* s = (const int*)b;
    if(*f > *s){
        return 1;
    }
    if(*f < *s){
        return -1;
    }
    return 0;
}

int main(){
    int n = 1000;
    int arr[n];
    for(int i = 0; i < n; i++){
        scanf("%d", &arr[i]);
    }
    qsort(arr, n, sizeof(int), compare);
    int key = 0;
    clock_t time_1 = clock();
    int* index = bsearch(&key, arr, n, sizeof(int), compare);
    time_1 = clock() - time_1;
    if(index){
        printf("exists\n");
    }
    else{
        printf("doesn't exist\n");
    }
    printf("%f\n", ((float)time_1) / CLOCKS_PER_SEC);
    int flag = 0;
    clock_t time_2 = clock();
    for(int i = 0; i < n; i++){
        if(key == arr[i]){
            flag = 1;
            break;
        }
    }
    time_2 = clock() - time_2;
    if(flag){
        printf("exists\n");
    }
    else{
        printf("doesn't exist\n");
    }
    printf("%f\n", ((float)time_2) / CLOCKS_PER_SEC);
```

```
    return 0;
}
```