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. 1382 _____ Коренев Д.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2022

Цель работы.

Рассмотреть функции стандартной библиотеки языка программирования Си.

Задание (Вариант 2).

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

Программа должна совершать следующие действия:

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

Выполнение работы.

Подключение заголовочных файлов стандартной библиотеки: stdio.h, string.h, stdlib.h, time.h, а так же, с помощью директивы define, определение значения для TOTAL.

Функция cmpb необходимая для алгоритма двоичного поиска, возвращает «0» если значение равно ключу, и 1 или -1 в противном случае.

Функция cmpq необходима для алгоритма быстрой сортировки, возвращает разницу между двумя значениями, поступивших на вход.

Функция main считывает значения в массив nums, сортирует их при помощи функции qsort, четвертым аргументом которой является функция cmpq описанная выше. Переменная time равна времени начала выполнения двоичного алгоритма поиска, а после его окончания равна времени его работы. Выводится результат работы: найдено ли число, а также время его работы. Далее идут аналогичные действия, однако вместо двоичного алгоритма поиска был использован метод перебора при помощи цикла for . Программа завершена. Пользователь может наглядно увидеть разницу времени работы двух разных алгоритмов.

Тестирование.

Результаты тестирования представлены в таблице 1.

Для тестирования программы были использованы данные с меньшим количеством символов, количества которых достаточно для проверки работоспособности программы.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1	20 чисел 1 -23 124 42 52 -12 42 62 -63 -29 363 637 -363 -433 23 234 236 -23 629 34	doesn't exist 0.000001 doesn't exist 0.000002	Программа работает верно.
2	50 чисел 22 -10 -20 -53 -37 -50 82 -18 -42 -57 27 43 41 -22 -22 -72 76 65 -51 65 -99 -54 -44 -6 -49 39 71 -85 84	exists 0.000002 exists	Программа работает верно.

	-69 36 -73 -63 64 -35 -41 5 -57 -64 -93 35 11 25 -59 12 65 32 16 -30 -42	0.000003	

Выводы.

Была изучена стандартная библиотека языка Си. Разработана программа поиска числа в массиве двумя алгоритмами, наглядное сравнение времени их работы: алгоритм qsearch быстрее переборочного алгоритма.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название программы: PR_Korenev_Danil_lb1.c

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

#define TOTAL 1000

int cmpb (const void *key, const void *src){
    if ( *(int*)key >  *(int*)src) return 1;
    if ( *(int*)key == *(int*)src) return 0;
    if ( *(int*)key <  *(int*)src) return -1;
}

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

int main(){
    int nums [TOTAL];
    for (int i = 0; i < TOTAL; i++){
        scanf("%d", &nums[i]);
    }


    //sorted
    qsort(nums, TOTAL, sizeof(int), cmpq);

    //start time
    clock_t timer;
    timer = clock();


    //find zer by bsearch
    int* pItem;
    int key = 0;
    pItem = (int*) bsearch(&key, nums, TOTAL, sizeof (int), cmpb);


    //count time of algorithm
    timer = clock() - timer;

    //existing zero
    if (pItem == NULL){
        printf("doesn't exist\n");
    } else {
        printf("exists\n");
    }


    //print time of qsearch algorithm
    printf("%f\n", ((float)timer)/CLOCKS_PER_SEC);
```

```
    //start time
    timer = clock();

    //find zero by for
    int existing = 0;
    for(int i = 0; i < TOTAL; i++){
        if (nums[i] == 0){
            existing = 1;
            break;
        }
    }


    //count time of algorithm
    timer = clock() - timer;


    //existing zero
    if (!existing){
        printf("doesn't exist\n");
    } else {
        printf("exists\n");
    }

    //print time of qsearch algorithm
    printf("%f\n", ((float)timer)/CLOCKS_PER_SEC);

    return 0;
}
```