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 С

Студент гр. 1304 _____ Дешура Д.В.

Преподаватель _____ Чайка К.В.

Санкт-Петербург
2022

Цель работы.

Изучить возможности стандартной библиотеки языка C, научиться на практике использовать содержащиеся в ней функции.

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:

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.

В программе используются функции стандартной библиотеки из заголовочных файлов <stdio.h>, <stdlib.h>, <time.h>. Для сортировки массива используется функция стандартной библиотеки qsort(), а для поиска элемента 0 в массиве применяется функция bsearch(), для их работы реализована функция-компаратор cmp(). Время работы bsearch() и посимвольного перебора вычисляется при помощи функции clock(), возвращающей количество временных тактов процессора, прошедших с начала работы программы. Разделив разность тактов до запуска поиска и после его конца на макрос CLOCKS_PER_SEC, программа находит время выполнения поиска.

Разработанный программный код см. в приложении А.

Тестирование.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	1 2 4 5 6 0 4 8 10 12	exists 0.000002 exists 0.000000	SIZE = 10
2.	1 2 1111111 222 1565 153 5 4 54 789	doesn't exist 0.000002 doesn't exist 0.000001	SIZE = 10
3.	1 212 112 12 12 1 25 45 78 92 -1 8 -98 -56 45 7 785 -365 12 44 -82 12 4 8 6 98 12 45 67 98 -1 -8 -99 94 78 54 78 65 15 35 48 46 0 21 8 56 7 3 6 45 -4 -951 753 54 55 56 57 58 59 60 -61 62 -63 -64 65 66 67 -68 88 46 -71 52 89 65 15 24 0 4 7 3 123 156 457 49 85 86 87 -88 -456 -9 1 92 93 94 96 75 48 -8 99 -100	exists 0.000002 exists 0.000001	SIZE = 100
4.	96 99 90 28 106 48 99 105 62 34 75 40 103 72 76 39 53 39 85 49 24 81 80 42 95 43 75 36 80 48 42 15 35 71 82 80 106 69 24 55 42 86 34 32 97 50 111 37 28 83 26 39 104 93 69 86 75 83 109 94 18 38 97 40 49 18 59 42 26 70 37 108 95 58 27 80 47 77 56 63 100 21 89 91 101 97 16 64 67 112 45 73 89 81 52 25 39 99 107 52	doesn't exist 0.000002 doesn't exist 0.000001	SIZE = 100

Выводы.

Выполнив лабораторную работу №1, мы создали программу, осуществляющую поиск числа 0 среди элементов отсортированного массива при помощи встроенной функции бинарного поиска и методом перебора, а также выводящее время работы обоих способов.

Изучили возможности стандартной библиотеки языка С и научились на практике использовать содержащиеся в ней функции.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: 1304_PR_Дешура_ДВ_ЛР1.c

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

int cmp(const void * x1, const void * x2){
    if(*(int*)x1 > *(int*)x2)
        return 1;
    if(*(int*)x1 == *(int*)x2)
        return 0;
    if(*(int*)x1 < *(int*)x2)
        return -1;
}

int main(){
    int arr[SIZE], zero = 0;
    float start, end;
    for(int i = 0; i <= (SIZE - 1); i++)
        scanf("%d", &arr[i]);
    qsort(arr, SIZE, sizeof(int), cmp);

    start = clock();
    int *pointer = bsearch(&zero, arr, SIZE, sizeof(int), cmp);
    end = clock();
    if(pointer == NULL)
        printf("doesn't exist\n");
    else
        printf("exists\n");
    printf("%f\n", (end - start)/CLOCKS_PER_SEC);

    int i = 0;
    start = clock();
    while(arr[i] != 0 && i < SIZE)
        i++;
    end = clock();
    if(i == SIZE)
        printf("doesn't exist\n");
    else
        printf("exists\n");
    printf("%f\n", (end - start)/CLOCKS_PER_SEC);

    return 0;
}
```