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

Студент гр. 0382 _____ Довченко М.К.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2021

Цель работы.

Научиться работать с функциями стандартной библиотеки stdlib.h, изучить работу функции clock библиотеки time.h.

Задание.

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.

Основные теоретические положения.

Функция *qsort* библиотеки *stdlib.h* – сортирует массив данных с помощью функции компаратора.

Функция *bsearch* библиотеки *stdlib.h* – находит определенный элемент в массиве данных с помощью функции компаратора.

Функция *clock()* библиотеки *time.h* – возвращает количество тактов процессора, прошедших с запуска программы.

Макрос *CLOCKS_PER_SEC* библиотеки *time.h,* принимающий значение 1 миллиона тактов процессора за 1 секунду.

Выполнение работы.

Для выполнения данной задачи необходимо подключить 3 библиотеки: *stdio.h, stdlib.h, time.h*. Также для удобства тестирования объявим макрос *N*, по условию задачи равный 1000.

В функции *main* объявляются следующие переменные:

- *int arr[N]* – массив для хранения чисел.
- *int zero* = 0 – переменная, нужная для работы функции *bsearch.*
- *int flag* = 0 – переменная, определяющая нахождение нуля в массиве чисел после выполнения цикла по перебору каждого числа в массиве.
- *clock_t start_b, end_b, start_c, end_c* – переменные для хранения количества тактов процессора во время выполнения программы.

После объявления переменных в функции выполняется сканирование массива чисел с помощью цикла *for* и функции *scanf* в цикле. Считывание заканчивается, когда переменная *i*, объявляемая в цикле, доходит до значения макроса *N*.

Перед выполнением сортировки с помощью функции *qsort* переменной *start_b* присваивается значение функции *clock()*, чтобы получить количество тактов процессора, прошедших с момента запуска программы. После этого с помощью функции *qsort* массив чисел сортируется для его последующего использования функцией *bsearch*. Аргументы, передаваемые в функции *qsort* и *bsearch,* состоят из массива чисел *arr,* количества элементов массива *N*, размера элементов массива *sizeof(int),* искомого элемента *zero* и функции компаратора *compare,* сравнивающая 2 числа и возвращающая 1, -1 и 0, если первое число больше, меньше или равно в сравнении со вторым соответственно. После этого переменной *end_b* присваивается значение функции *clock()*. Программа выводит

строку *“exists”*, если ноль в массиве был найден, *“doesn’t exist”*, если он не был найден, и количество секунд, прошедших с момента начала выполнения функций *qsort* и *bsearch*.

Далее с помощью перебора всех чисел массива происходит поиск нуля. Для нахождения времени, затраченного на перебор, переменным *start_c* и *end_c* также присваивается значение функции *clock()*. Программой выводится результат поиска и количество времени, потребовавшееся на него.

Разработанный программный код см. в приложении А.

Результаты тестирования см. в приложении Б.

Тестирование.

Здесь результаты тестирования, которые помещаются на одну страницу.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	-23 2 51 0 21 53 75 85 345	exists 0.000000 exists 0.000000	Программа работает корректно
2.	345 -63 84 32 86 2 97 1 45 75	doesn’t exist 0.000000 doesn’t exist 0.000000	Программа работает корректно
3.	0 84 142 967 36 -32 -2 53 78 64546	exists 0.000000 exists 0.000000	Программа работает корректно

Выводы.

Были исследованы функции стандартной библиотеки языка Си такие, как *qsort, bsearch, clock.*

Разработана программа, выполняющая считывание с клавиатуры исходных данных, сортировку массива чисел, нахождение нуля в массиве двумя способами и выводящая количество затраченного времени на оба способа.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Сначала указываем имя файла, в котором код лежит в репозитории:

Название файла: lab4.c

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

int compare(const void*, const void*);

int main(){
    int arr[N];
    int zero = 0;
    int flag = 0;
    clock_t start_b, end_b, start_c, end_c;

    for(int i = 0; i < N; i++){
        scanf("%d", &arr[i]);
    }
    start_b = clock();
    qsort(arr, N, sizeof(int), compare);
    int* zeroat = (int*)bsearch(&zero, arr, N, sizeof(int), compare);
    end_b = clock();

    if(zeroat)
        printf("exists\n");
    else
        printf("doesn't exist\n");
    printf("\n%f", (double)(end_b - start_b) / CLOCKS_PER_SEC);

    start_c = clock();
    for(int j = 0; j < N; j++){
        if(arr[j] == 0)
            printf("exists\n");
            flag = 1;
    }
    end_c = clock();
    if(!flag)
        printf("doesn't exist\n");
    printf("\n%f", (double)(end_c - start_c) / CLOCKS_PER_SEC);

    return 0;
}


int compare(const void* num1, const void* num2){
    int* b = (int*) num1;
    int* a = (int*) num2;
    if (*a > *b)
        return -1;
    else if(*a < *b)
        return 1;
    else
```

```
        return 0;
}
```