**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: Стандартная библеотека Си

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Шевченко Д.В. |
| Преподаватель |  | Чайка К.В. |

Санкт-Петербург

2017

**Цель работы.**

Написать 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.

**Описание работы программы.**

Программа представлена в виде одной вспомогательной функции

и основной функции.

1. Вспомогательная функция int compare(const void\* arg\_1, const void\* arg\_2)

Сравнивает числа, используется для работы qsort (сортировки)

1. Основная функция int main()

Содержит основную логику программы, содержит в себе алгоритмы бинарного поиска и поиска «пузырьком», так-же вычисляет время за которое был совершен поиск в отсортированном массиве.

Приложение 1. Исходный код программы

#include <stdio.h>

|  |  |
| --- | --- |
|  | #include <stdlib.h> |
|  | #include <time.h> // работа со временем |
|  |  |
|  | #define S\_NUMBER 0 // искомое число |
|  | #define ARR\_SIZE 1000 // для тестирования мне была необходима возможность установки меньшего размера массива в кратчайшие сроки |
|  |  |
|  | int compare(const void\* arg\_1, const void\* arg\_2); // прототип функции сравнения чисел для qsort |
|  |  |
|  | int main() { |
|  | short exist = 0; // принимает 1 или 0 в зависимости от того существует ноль или нет |
|  | int low = 0; // нижняя граница при бинарном поиске |
|  | int high = ARR\_SIZE-1; // верхняя граница при бинарном поиске |
|  | int count; // счетчик, в целом используется несколько раз в качестве счетчика в разных местах |
|  | int nums[ARR\_SIZE]; // массив чисел |
|  |  |
|  | for (int i = 0; i<ARR\_SIZE; i++) |
|  | { |
|  | scanf("%d", &nums[i]); // цикл заполняющий массив числами с потока ввода |
|  | } |
|  |  |
|  | qsort(nums, ARR\_SIZE, sizeof(int), compare); // сортировка массива quicksort-ом |
|  |  |
|  | time\_t time\_start = time(NULL); // запись времени начала поиска |
|  |  |
|  | while (low != high) |
|  | { |
|  | count = (low + high) / 2; // если число больше искомого числа, значит мы находимся выше этого числа и это становится |
|  | if (nums[count] > S\_NUMBER) // верхней границей в дальнейшем поиске |
|  | { |
|  | high = count; |
|  | continue; |
|  | } |
|  | if (nums[count] < S\_NUMBER) // меньше искомого числа - нижняя граница |
|  | { |
|  | low = count; |
|  | continue; |
|  | } |
|  | if (nums[count] == S\_NUMBER) // если число равно искомому, то exist принимает значение "1" |
|  | { //что эквивалентно "ноль существует" в логике программы |
|  | exist = 1; |
|  | break; |
|  | } |
|  | } |
|  |  |
|  | time\_t time\_finish = time(NULL); // запись времени окончания поиска |
|  |  |
|  | if (exist == 1) |
|  | printf("exists\n"); // вывод exists если флаг поднят (ноль найден) и doesn't exist если нет |
|  | else |
|  | printf("doesn't exist\n"); |
|  | exist = 0; // установка флага равным нулю, для дальнейшего его использования в повторном поиске |
|  |  |
|  | printf("%f\n", difftime(time\_finish, time\_start)); // вывод разницы между временем начала и конца поиска |
|  |  |
|  | time\_start = time(NULL); // запись времени старта поиска |
|  |  |
|  | for (count = 0; count<ARR\_SIZE; count++) // поиск пузырьком |
|  | { |
|  | if (nums[count] == S\_NUMBER) |
|  | { |
|  | exist = 1; // установка флага 1 если ноль найден |
|  | break; |
|  | } |
|  | } |
|  |  |
|  | time\_finish = time(NULL); // запись времени окончания поиска |
|  |  |
|  | if (exist == 1) // вывод exists если флаг поднят (ноль найден) и doesn't exist если нет |
|  | printf("exists\n"); |
|  | else |
|  | printf("doesn't exist\n"); |
|  |  |
|  | printf("%f\n", difftime(time\_finish, time\_start)); // вывод времени поиска пузырьком |
|  | return 0; |
|  | } |
|  |  |
|  | int compare(const void\* arg\_1, const void\* arg\_2) |
|  | { |
|  | return (\*(int\*)arg\_1 - \*(int\*)arg\_2); // функция сравнивающая числа, используется для qsort-а |
|  | } |

**Составление отчёта.**

По лабораторной работе составлен отчет Lr\_1\_orchet.docx и файл с кодом программы Lab\_1.c  
В репозитории на Github создана от мастера ветка Shevchenko\_Danil\_Lab\_1 с помощью команды «git checkout -b Shevchenko\_Danil\_Lab\_1».

Файлы отчета и кода были помещены в ветку репозитория командами:  
«git add Lab\_1.c»

«git add Lr\_1\_otchet.docx»  
«git commit»   
«git push origin Shevchenko\_Danil\_Lab\_1»  
После чего был сделан pull request на ветку master.