**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 библиотека Си

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Шевченко Д.В. |
| Преподаватель |  | Берленко Т.А. |

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. Вспомогательные функции int compare(const void\* arg\_1, const void\* arg\_2)

Сравнивает числа, используется для работы qsort (сортировки) и bsearch

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 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 и bsearch

int main() {

int key = 0; // искомое число

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

clock\_t t\_active = clock(); // запись времени начала поиска

int \* ptrItem = (int\*)bsearch(&key, nums, 1000, sizeof(int), compare);

t\_active = clock() - t\_active; // запись времени окончания поиска

if (ptrItem != NULL)

printf("exists\n"); // вывод exists если флаг поднят (ноль найден) и doesn't exist если нет

else

printf("doesn't exist\n");

ptrItem = NULL;

printf("%f\n", (double)t\_active / CLOCKS\_PER\_SEC); // вывод разницы между временем начала и конца поиска

t\_active = clock();

for (int i = 0; i<ARR\_SIZE; i++) // поиск пузырьком

{

if (nums[i] == key)

{

ptrItem = &key; // установка флага 1 если ноль найден

break;

}

}

t\_active = clock() - t\_active; // запись времени окончания поиска

if (ptrItem != NULL)

printf("exists\n"); // вывод exists если флаг поднят (ноль найден) и doesn't exist если нет

else

printf("doesn't exist\n");

printf("%f\n", (double)t\_active / CLOCKS\_PER\_SEC); // 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 и bsearch

}

**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 request на ветку master.

Вывод: Были изучены функции стандартной библиотеки, осуществляющие поиск и сортировку элементов в массиве (bsearch и qsort), также были изучены и исследованы их прототипы и принцип работы, необходимые условия при которых может быть осуществлена их корректная работа. Также в процессе работы были изучена библиотека time.h содержащая функции для работы со временем, такие как time\_t и clock\_t, они также были применены для реализации программного кода удовлетворяющего постановленной задаче.