**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по лабораторной работе №1**

**по дисциплине «Программирование»**

Тема: ОБЗОР СТАНДАРТНОЙ БИБЛИОТЕКИ

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Матюшина М.Е. |
| 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.

**Ход работы.**

1. **Используемые функции стандартной библиотеки.**
   * Функция для сортировки элементов массива (алгоритм быстрой сортировки).

**qsort(element, SIZE, sizeof(int), cmpfunc**);

библиотеки <stdlib.h>

где:

element - указатель на первый элемент сортируемого массива,

SIZE – количество элементов массива,

sizeof(int) – размер элемента массива,

cmpfunc – функция, которая сравнивает два элемента массива. Она принимает два параметра  — указатели на элементы массива, типа void\*. Возвращаемое значение этой функции может быть отрицательным, равным нулю или положительным.

**int cmpfunc**(const void \* a, const void \* b) {

return ( \*(int\*)a - \*(int\*)b );

}

* + Функция bsearch библиотеки <stdlib.h> (Алгоритм двоичного поиска)

**bsearch (&key, element, SIZE, sizeof (int), cmpfunc);**

где:

&key - указатель на искомый объект типа void\*,

element - указатель на первый элемент массива, типа void\*,

sizeof(int) – размер элемента массива,

cmpfunc – функция, которая сравнивает два элемента массива.

Возвращаемое значение: указатель на элемент массива, который равен значению в key – 0. Если искомый элемент  не найден, возвращается нулевой указатель.

int\* search;

search = (int\*) bsearch (&key, element, SIZE, sizeof (int), cmpfunc);

* + Функция clock библиотеки <time.h> (Считает время, за которое совершается поиск числа 0)

Прототип: **clock\_t clock( void );**

Функция возвращает количество временных тактов.

**2.Полный код программы.**

#include <stdio.h>   
#include <stdlib.h>   
#include <time.h>   
  
#define SIZE 1000   
  
int cmpfunc(const void \* a, const void \* b) {   
 return ( \*(int\*)a - \*(int\*)b );   
}   
  
int main() {   
int \* element = (int\*)malloc(SIZE\*sizeof(int));   
int i;   
for ( i=0; i<SIZE; i++){   
 scanf("%d", &element[i]);   
 }   
  
qsort(element, SIZE, sizeof(int), cmpfunc);   
int key=0;   
int time;   
time = (int)clock();   
int\* search;   
search = (int\*) bsearch (&key, element, SIZE, sizeof (int), cmpfunc);   
time=(int)clock()-time;   
if (search!=NULL) {   
 printf("exists\n");   
}   
else {   
 printf ("doesn't exist\n");   
}   
printf("%d\n", time);   
//перебор   
int control=0;   
time = (int)clock();   
for (i=0; i<SIZE; ++i) {   
 if (element[i]==0) {   
 control=1;   
 break;   
 }   
 }   
time=(int)clock()-time;   
if (control==1) {   
 printf("exists\n");   
 }   
else {   
 printf("doesn't exist\n");   
 }   
printf("%d\n", time);   
free(element);   
return 0;   
}

**3.Загрузка файла на GitHub.**

* В локальном репозитории была создана новая ветка matyushina\_2sem\_lr1

git checkout -b matyushina\_2sem\_lr1

* Создана директория matyushina\_2s\_lr1

mkdir matyushina\_2s\_lr1

* В вышеуказанную папку был перемещен файл LAB1.c
* Исходный файл добавлен в локальный репозиторий

git add LAB1.c

* Командой <git commit –m> были зафиксированы изменения в проекте

git commit –m «matyushina\_2sem\_lr1»

* Командой <git push> последний commit был отправлен на удаленный репозиторий

git push origin matyushina\_2sem\_lr1

**Вывод:** в ходе работы были изучены функции стандартной библиотеки, получены навыки поиска элементов двумя различными способами: используя алгоритм двоичного поиска и используя перебор всех чисел массива, а также подсчет времени, за которое совершается поиск каждым из методов.