**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №1**

**по дисциплине «Программирование»**

Тема: **Стандартная библиотека языка С.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Григорьев И.С. |
| Преподаватель |  | Кринкин К.В. |

Санкт-Петербург

2017

**Содержание**

[**Цель работы** 3](#_Toc477182460)

[**Задание** 3](#_Toc477182461)

[**Ход работы** 3](#_Toc477182462)

[**Функция быстрой сортировки** 3](#_Toc477182463)

[**Функция двоичного поиска** 4](#_Toc477182464)

[**Перебор чисел массива** 4](#_Toc477182465)

[**Время работы функций двоичного поиска и перебора чисел массива** 4](#_Toc477182466)

[**Вывод** 4](#_Toc477182467)

[**Приложение** 5](#_Toc477182468)

# **Цель работы**

Написание программного кода с использованием функций стандартной библиотеки языка СИ.

# **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.*

# **Ход работы**

## **Функция быстрой сортировки**

Функция **qsort** предназначена упорядочивать множества блоков байтов равной длины. Второй аргумент функции — это число таких блоков, третий аргумент — длина каждого блока. Первый аргумент — это адрес, где находится начало первого блока (предполагается, что блоки в памяти расположены друг за другом подряд).

Четвёртый аргумент функции **qsort** — это имя функции, которая умеет сравнивать два элемента массива. В нашем случае это:

int funccmp(const void \* x1, const void \* x2) // функция сравнения

{

return ( \*(int\*)x1 - \*(int\*)x2 ); // если результат вычитания - 0, значит числа равны

}

В силу указанной универсальности функции сортировки, функция сравнения получает в качества аргумента адреса двух блоков, которые нужно сравнить и возвращает 1, 0 или -1:

* положительное значение, если x1 > x2
* 0, если x1 == x2
* отрицательное значение, если x1 < x2

Поскольку у нас блоки байт -- это целые числа (в 32-битной архитектуре это четырёхбайтовые блоки), то необходимо привести данные указатели типа (const void\*) к типу (int \*) и осуществляется это с помощью дописывания перед указателем выражения «(const int\*)». Затем нужно получить значение переменной типа int, которая лежит по этому адресу. Это делается с помощью дописывания спереди звездочки. Функция qsort в нашем случае:

qsort(arr, 1000, sizeof(int), funccmp); //функция сортировки

## **Функция двоичного поиска**

int \*ptr=(int\*)bsearch(&key, arr, 100, sizeof(int), funccmp);

Вывод результата:

if(ptr!=NULL) //если мы нашли совпадение с ключом, то указатель на него не будет пуст

printf("exists\n");

else

printf("doesn't exist\n");жэы

## **Перебор чисел массива**

for(i=0;i<1000;i++){

if(arr[i]==key){

j=arr[i];

break; //если мы находим элемент массива равный нашему ключу, то цикл обрывается

}

}

Вывод результата:

if(j==key) //сравниваем с ключом и выводим соответствующий результат

printf("exists\n");

else

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

## **Время работы функций двоичного поиска и перебора чисел массива**

#include <time.h>

time\_t t1,t2,t3,t4; //t1 && t2 – начало и конец двоичного поиска, t3 && t4 – перебора чисел массива

ti=time(NULL); //где i=1,2,3,4

Время работы фрагмента кода равно разности времен конца его выполнения и начала.

# **Вывод**

Данная лабораторная работа была выполнена с помощью функций стандартной библиотеки (string.h, stdlib.h), благодаря этому отсутствовала необходимость реализации собственных процедур; заметно уменьшился объем кода и увеличилось быстродействие программы за счет использования проверенных готовых решений.

# **Приложение**

/\*

Лабораторная работа №1. Второй семестр. Григорьев Иван гр.6304

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

int funccmp(const void \* x1, const void \* x2) // функция сравнения

{

return ( \*(int\*)x1 - \*(int\*)x2 ); // если результат вычитания - 0, значит числа равны

}

int main() {

int i;

time\_t t1,t2,t3,t4;

int j=0;

int key=0;

int arr[1000];

for(i=0;i<1000;i++)

scanf("%d\n", &arr[i]);

qsort(arr, 1000, sizeof(int), funccmp); //Сортировка массива с помощью алгоритма быстрой сортировки

t1=time(NULL);

int \*ptr=(int\*)bsearch(&key, arr, 100, sizeof(int), funccmp); //Двоичный поиск ключа

t2=time(NULL);

if(ptr!=NULL)

printf("exists\n");

else

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

printf("%I\n", t2-t1); //Время двоичного поиска ключа

t3=time(NULL);

for(i=0;i<1000;i++){

if(arr[i]==key){

j=arr[i];

break;

}

}

t4=time(NULL);

if(j==key)

printf("exists\n");

else

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

printf("%I\n", t4-t3); //Время поиска перебором

return 0;

}