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

Студент гр. 1304 _____ Заика Т.П

Преподаватель _____ Чайка К.В.

Санкт-Петербург
2022

Цель работы.

Обзор стандартной библиотеки, практическое использование стандартной библиотеки

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.

Стандартная библиотека языка Си

Выполнение работы.

В ходе работы для решения поставленной задачи было принято создать несколько функций, выполняющих требуемый функционал. Реализованы функции ввода значений в массив, функция сравнения переменных в массиве, необходимая для работы стандартных библиотечных функций qsort и bsearch, а также функции бинарного поиска и полного перебора, выполняющие заданный в задаче функционал и выводящие соответствующие сообщения о выполнении. Все функции вызываются в главной функции программы.

Переменные:

int size — целочисленная переменная, хранящая значение количества элементов в массиве

int values[size] — целочисленный массив, хранящий количество элементов, заданное в переменной size

int key — целочисленная переменная, которая хранит значение, которое будет искать функция стандартной библиотеки bsearch в отсортированном массиве

int *f — указатель на целочисленную переменную, хрянящий приведённый к целочисленному типу указатель, переданный первым в функцию cmp

int *s - указатель на целочисленную переменную, хрянящий приведённый к целочисленному типу указатель, переданный вторым в функцию cmp

clock_t time_start — переменная типа clock_t, хранящее текущее количество тактов

int *pItem — указатель на целочисленную переменную, содержащий результат выполнения функции bsearch

clock_t time_end — переменная тип clock_t, хранящее текущее количество тактов минус количество тактов, записанное в переменную time_start. Используется для подсчета времени выполнения учатска кода

int foundKey — целочисленная переменная, служащая как флаг о нахождении заданной переменной в массиве при помощи полного перебора

Функции:

void scanNumbers(int *array, int n) — функция, принимающая указатель на массив и его размер для того, чтобы заполнить массив целочисленными значениями, введными с stdin

int cmp(const void *a, const void *b) — целочисленная функция, выполняющая сравнение элементов массива и необходимая для работы функций стандартной библиотеки qsort и bsearch

void binarySearch(int *values, int size, int key) — функция бинарного поиска, принимающая на вход отсортированный при помощи функции стандартной библиотеки qsort (сортировка происходит в теле функции main) массив, его размер, а также значение, которое нужно найти в массиве. В теле функции реализован функционал бинарного поиска при помощи функции стандартной библиотеки bsearch, вывода информации о результате поиска, а также вывода времени выполнения функции в секундах

void fullSearch(int *values, int size, int key) - функция полного перебора, принимающая на вход отсортированный при помощи функции стандартной библиотеки qsort (сортировка происходит в теле функции main) массив, его размер, а также значение, которое нужно найти в массиве. В теле функции реализован функционал поиска полным перебором, вывода информации о результате поиска, а также вывода времени выполнения функции в секундах

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	1 0 3 4 5 2 0 3 4 1	exists Время бинарного поиска - 0.000000 секунд exists Время полного перебора - 0.000001 секунд	Успешный тест
2.	1 4 2 3 4 5 1 7 4 1	doesn't exist Время бинарного поиска - 0.000001 секунд doesn't exist Время полного перебора - 0.000000 секунд	Успешный тест

Выводы.

Было исследована, изучена стандартная библиотека языка Си.

Разработана программа, выполняющая считывание с клавиатуры значений, записываемых в массив. Далее для работы бинарного поиска и поиска полным перебором, происходит сортировка массива. Программа осуществляет поиск 0 в заданном массиве с помощью бинарного поиска и поиска полным перебором, и в результате позволяет наглядно убедится в том, нашел ли каждый тип поиска искомое значение и за какое время в секундах была выполнена поставленная задача, выводя соответствующую информацию на экран пользователю.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void scanNumbers(int *array, int n){
      for(int i=0; i<n; i++){
            scanf("%d", &array[i]);
      }
}

int cmp (const void *a, const void *b){
      int *f = (int *) a;
      int *s = (int *) b;

      if (*f > *s) return 1;
      if (*f < *s) return -1;
      return 0;
}

void binarySearch(int *values, int size, int key){
      clock_t time_start = clock();
      int *pItem = (int *) bsearch(&key, values, size, sizeof(int),
cmp);
      clock_t time_end = clock() - time_start;
      if (pItem == NULL){
         printf("%s\n", "doesn't exist");
     } else {
         printf("%s\n", "exists");
     }
      printf("Время бинарного поиска - %f секунд\n", (double)time_end /
CLOCKS_PER_SEC);
}

void fullSearch(int *values, int size, int key){
      int foundKey = 0;
     clock_t time_start = clock();
     for(int i=0; i<size; i++){
      if (values[i] == key){
            foundKey = 1;
            break;
      }
     }
     clock_t time_end = clock() - time_start;
     if (foundKey == 0){
         printf("%s\n", "doesn't exist");
     } else {
         printf("%s\n", "exists");
     }
```

```
    printf("Время полного перебора - %f секунд\n", (double)time_end /
CLOCKS_PER_SEC);
}

int main(){
     int size = 10;
     int values[size];
     scanNumbers(values, size);

     qsort(values, size, sizeof(int), cmp);

     int key = 0;
     binarySearch(values, size, key);
     fullSearch(values, size, key);
}
```