

**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. 0382

\_\_\_\_\_

Тихонов С.В.

Преподаватель

\_\_\_\_\_

Берленко Т.А..

Санкт-Петербург

2021

## Цель работы.

Изучить и освоить функционал стандартной библиотеки языка программирования Си.

## Задание.

### Вариант-2.

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.

*void qsort (void\* base, size\_t num, size\_t size, int (\*compar)(const void\*,const void\*)) из stdlib.h*

Функция принимает указатель на начальный элемент массива, количество элементов и размер одного элемента, а также указатель на функцию для сравнения двух элементов.

Так как тип элементов может быть любым, то и указатель на первый элемент массива имеет тип *void*. Это позволяет, зная адрес первого элемента и размер каждого элемента вычислить адрес любого элемента массива в памяти и обратиться к нему. Остается только сравнить 2 элемента имея 2 указателя на них. Это выполняет функция *compar*, указатель на которую передается функции *qsort* в качестве одного из параметров.

Функция *compar* принимает 2 указателя типа *void*, но в своей реализации может привести их к конкретному типу (так как её реализация остается за программистом, он точно знает элементы какого типа он сортирует) и сравнивает их. Результат сравнения определяется знаков возвращаемого функций *qsort* числа.

*clock\_t clock( void ) из time.h*

Возвращает количество временных тактов, прошедших с начала запуска программы. С помощью макроса *CLOCKS\_PER\_SEC* функция получает количество пройденных тактов за 1 секунду. Таким образом, зная сколько выполняется тактов в секунду, зная время запуска программы можно посчитать время работы всей программы или отдельного её фрагмента, что и делает данная функция.

## Выполнение работы.

Считываем числа в массив. Производим сортировку функцией `qsort()` (использует `comp()` для сравнения чисел), с помощью функции `bsearch()` (использует `comp()` для сравнения чисел) определяем есть ли 0. При помощи функции `clock()` записываем время начала и конца поиска. С помощью макроса `CLOCKS_PER_SEC` получаем представление времени в секундах. Аналогично находим время поиска числа 0 путём перебора всех чисел в массиве. Выводим полученные результаты.

Функция `int comp(const void *a, const void *b)` - функция, указатель на которую передается функциям `qsort` и `bsearch` в качестве одного из параметров. Функция принимает 2 указателя типа `void`, но в своей реализации приводит их к типу `(int)` и сравнивает их. Результат сравнения определяется знаком возвращаемого функцией `(int)` числа.

Код программы смотрите в Приложение А.

## Тестирование.

Результаты тестирования представлены в табл.1.

Таблица 1-Результаты тестирования

№п/п	Входные данные	Выходные данные	Комментарии
1.	1 2 3 4 5 6 7 8 10 9	1 2 3 4 5 6 7 8 9 10 0.000002 0.000002	Работает верно

## Вывод.

В ходе работы были изучены основные функции стандартных библиотек Си. А также была разработана программа для сравнения времени сортировки массива методом пузырька и функцией быстрой сортировки.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

#define kol 1000

int comp(const void *a, const void *b) {
    return *(const int *) a - *(const int *) b;
}

int main() {

    clock_t start_t, end_t;
    double time_spent_on_bsearch, time_spent_on_fullsearch;
    int nums[kol];
    for (int i = 0; i < kol; ++i) {
        scanf("%d", &nums[i]);
    }

    int tagret_number = 0;

    start_t = clock();

    qsort(nums, kol, sizeof(int), comp);
    int zero_in_array = bsearch(&tagret_number, nums, kol, sizeof(int), comp) !=
    NULL;

    end_t = clock();

    time_spent_on_bsearch = ((double) (end_t - start_t)) / CLOCKS_PER_SEC;

    if (zero_in_array)printf("exists\n");
    else printf("doesn't exist\n");
    printf("%f\n", time_spent_on_bsearch);
```

```

start_t = clock();

zero_in_array = 0;
for (int i = 0; i < kol; ++i) {
    if (nums[i] == 0) {
        zero_in_array = 1;
    }
}

end_t = clock();
time_spent_on_fullsearch = ((double) (end_t - start_t)) / CLOCKS_PER_SEC;

if(zero_in_array)printf("exists\n");
else printf("doesn't exist\n");

printf("%f", time_spent_on_fullsearch);
}

```