**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра МОЭВМ**

# ОТЧЕТ

**по лабораторной работе № 1**
**по дисциплине «Программирование»**
**Тема: Обзор стандартной библиотеки.**

Студент гр. 7303 _____ Мищенко М.А.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2018

# Оглавление

## Цель работы:

Напишите программу, на вход которой подается текст на **английском** языке (длина текста не превышает **1000** символов) и слово **str**(длина слова не превышает **30** знаков). Слова в тексте разделены пробелами или точкой. Программа должна вывести строку "exists", если **str** в тексте есть и "doesn't exist" в противном случае.
Программа должна реализовать следующий алгоритм:

- разбить текст на слова, используя **функции стандартной библиотеки**
- отсортировать слова, используя алгоритм быстрой сортировки (см. **функции стандартной библиотеки**)
- определить, присутствует ли в тексте **str**, используя алгоритм двоичного поиска (для реализации алгоритма двоичного поиска используйте **функцию стандартной библиотеки)**
- вывести строку "exists", если **str** в тексте есть и "doesn't exist" в противном случае.

## Ход работы:

1. Были подключены библиотеки:

   **#include <stdio.h>** - библиотека для ввода-вывода;

   **#include <stdlib.h>** - библиотека для работы с памятью;

   **#include <string.h>** - библиотека для работы со строками.

2. Были определены макросы и выделена память под массивы (один из которых двумерный) (рис. 1):

```
#define text_size 1001
#define word_size 31
#define number_of_words 10
```

```
char* word = (char*) malloc(word_size * sizeof(char));
char* text = (char*) malloc(text_size * sizeof(char));
char** text2 = (char**) malloc(sizeof(char*) * number_of_words);
```

(Рис. 1)

3. Ввод текста и проверочного слова (рис. 2) :

```
fgets(text, text_size, stdin);
scanf("%s",word);
text[strlen(text)-1] = '\0';
```

(Рис. 2)

4. Разбиение текста на слова при помощи функции “strtok” и его перезапись в двумерный массив “text2”.Так же происходит перераспределение памяти если кол-во слов в тексте > 10.(рис. 3):

```
char* token = strtok(text, " .");
int i=0;
for(i = 0; token != NULL; i++) {
    if(i == number_of_words * count - 1) {
    count++;
    text2  = (char**) realloc(text2 , number_of_words* count * sizeof(char*));
    }
  text2 [i] = token;
    token = strtok(NULL, " .");
```

(Рис.3)

5. Сортировка текста при помощи функции “qsort”.(Рис. 4)

```
qsort(text2 , i, sizeof(char*), function);
```

(Рис. 4)

6. Проверка содержит ли текст искомое слово.(Рис. 5)

```
       char* ptr = (char*) bsearch(&word, text2 , i, sizeof(char*), function);
        if(ptr != NULL)
        printf("exists");
        else
        printf("doesn't exist");
```

(Рис. 5)

7. Функция “function” принимает два параметра: первый указывает на искомый объект, а второй — на один из элементов массива, типа void *. Функция приводит передаваемые параметры к типу (char*) и выполнить сравнение. (Рис. 6)

```
int function(const void* a, const void* b) {
return (strcmp(*(char**) a, *(char**) b));
}
```

(Рис.6)

**8.** создан Makefile для удобной и быстрой сборки программы: Содержимое,

Makefile-a:

```
all:
    gcc lr1.c -o lr1
```

**Вывод:** в данной л/р были реализованы функции: разбиения текста на слова, быстрой сортировки, двоичного поиска в массиве. Был закреплен материал по указателям, строкам, динамическим выделением памяти и работе со стандартными библиотеками.