**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 |  | Иванов В. С. |
| Преподаватель |  | Кринкин К.В. |

Санкт-Петербург

2017

Оглавление

[**Цель работы** 2](#_Toc477292053)

[**Задание** 2](#_Toc477292054)

[**Ход работы** 2](#_Toc477292055)

[1. Вводим обозначения 2](#_Toc477292056)

[2. Выполняем разбиение исходного текста на отдельные слова (*strtok*) 3](#_Toc477292057)

[3. Сортируем слова, используя алгоритм быстрой сортировки (*qsort*) 3](#_Toc477292058)

[4. Выполняем бинарный поиск str в массиве отдельных слов (*bsearch*) 3](#_Toc477292059)

[**Вывод** 3](#_Toc477292060)

[**Приложение** 3](#_Toc477292061)

# **Цель работы**

Написание программы с использованием функций стандартной библиотеки языка программирования Си.

# **Задание**

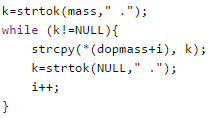
Напишите программу, на вход которой подается текст на **английском** языке (длина текста не превышает **1000** символов) и слово **str** (длина слова не превышает **30** знаков). Слова в тексте разделены пробелами или точкой. Программа должна вывести строку "exists", если **str** в тексте есть и "doesn't 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. Вводим обозначения:
   1. исходный текст;
   2. искомое слово;
   3. дополнительный массив для хранения отдельных слов
2. Выполняем разбиение исходного текста на отдельные слова (*strtok*):



1. Сортируем слова, используя алгоритм быстрой сортировки (*qsort*):



1. Выполняем бинарный поиск str в массиве отдельных слов (*bsearch*):



# **Вывод**

Выполнив данную лабораторную работу только с помощью функций стандартной библиотеки языка Си, мы научились не «изобретать велосипед», что ускорило время работы программы, а также упростило код программы и уменьшило размер файла.

# **Приложение**

//Vipolnil: Ivanov Vladimir 6304

//Zadanie predstavleno v otchete

|  |
| --- |
|  |
|  | #include <stdio.h> |
|  | #include <stdlib.h> |
|  | #include <string.h> |
|  | int funccmp(const void \* x1, const void \* x2) |
|  | { |
|  | return strcmp((char\*)x1,(char\*)x2); |
|  | } |
|  | int main() { |
|  | char mass[1000]; //text |
|  | fgets(mass,1000,stdin); |
|  | char str[30]; //iskomoe slovo |
|  | fgets(str,30,stdin); |
|  | char \*k; |
|  | int i=0; |
|  | char dopmass[1000][1000]; // dopolnitel'niy massiv dlya hraneniya otdel'nih slov |
|  | k=strtok(mass," ."); |
|  | while (k!=NULL){ //razbienie na slova |
|  | strcpy(\*(dopmass+i), k); |
|  | k=strtok(NULL," ."); |
|  | i++; |
|  | } |
|  | qsort(dopmass,i,1000,funccmp); //sortirovka |
|  | char \* ptr = (char\*) bsearch(str, dopmass, i, 1000, funccmp); //poisk (binarniy) str v massive |
|  | if (ptr != NULL) |
|  | printf("exists"); |
|  | else printf("doesn't exist"); |
|  | return 0; |
|  | } |
|  |  |
|  |  |