**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 |  | Пискунов Я. А. |
| Преподаватель |  | Кринкин К.В. |

Санкт-Петербург

2017

Оглавление

[Цель 3](#_Toc477285416)

[Задание 3](#_Toc477285417)

[Ход работы 3](#_Toc477285418)

[Разбиение исходного текста на слова 3](#_Toc477285419)

[Сортировка слов 3](#_Toc477285420)

[Бинарный поиск 3](#_Toc477285421)

[Вывод 3](#_Toc477285422)

[Приложение 4](#_Toc477285423)

Цель:

Написание программы с использованием функций стандартной библиотеки языка 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:

Разбиение исходного текста на слова

Производится с помощью функции “Strtok”

char\* ptr = strtok(text, " .");

while (ptr){

strcpy(words[i], ptr);

ptr = strtok(NULL, " .");

i++;

}

Сортировка слов

Производится с помощью функции “Qsort”

qsort(words, i, 1000, comparator);

Бинарный поиск

Производится с помощью функции “Bsearch”

char\* pointer\_on\_str = (char\*)bsearch(str, words, i, 1000, comparator);

Вывод:

В данной работе использованы лишь функции стандартной библиотеки языка, что значительно ускорило работу, уменьшило размер файла и упростило сам код, так как отсутствовала необходимость самим писать функции.

|  |  |
| --- | --- |
|  | **Приложение** |
|  | #include <stdio.h> |
|  | #include <string.h> |
|  | #include <stdlib.h> |
|  |  |
|  | //Function to compare two values with type of char\* |
|  | //returns 0 if two strings are equal |
|  | int comparator (const void\* a, const void\* b){ |
|  | return strcmp ((char\*)a, (char\*)b); |
|  | } |
|  |  |
|  | //Main function |
|  | int main() { |
|  | int i = 0; |
|  | char text[1000]; |
|  | char str[30]; |
|  |  |
|  | fgets(text, 1000, stdin); |
|  | fgets(str, 30, stdin); |
|  |  |
|  | char words[1000][1000]; |
|  |  |
|  | char\* ptr = strtok(text, " ."); //Finding first word using standart function "Strtok" |
|  | while (ptr){ |
|  | strcpy(words[i], ptr); |
|  | ptr = strtok(NULL, " ."); //Finding next word |
|  | i++; |
|  | } |
|  |  |
|  | qsort(words, i, 1000, comparator); //Sorting Massive of words using standart function "qsort" |
|  |  |
|  | char\* pointer\_on\_str = (char\*)bsearch(str, words, i, 1000, comparator); //Searching "str" in Massive of words |
|  |  |
|  | if (pointer\_on\_str==NULL) |
|  | printf("doesn't exist\n"); |
|  | else printf("exists\n"); |
|  |  |
|  | return 0; |
|  | } |