**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. 6303 |  | Куценок Л.А. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

**Цель работы.**

Написать 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. В локальном репозитории git была создана новая ветка *kutsenok\_leonid\_sem2\_lr1* командой

git checkout -b kutsenok\_leonid\_sem2\_lr1

1. Создана папка *kutsenok\_leonid\_sem2\_lr1*

mkdir kutsenok\_leonid\_sem2\_lr1

и файл *main.c*

touch main.c

1. Далее приведено краткое описание работы программы.
2. Из входного потока считывается текст и слово *str*

**char\*** str **=** **(char\*)** malloc**(**STR\_SIZE **\*** **sizeof(char));**

**char\*** text **=** **(char\*)** malloc**(**TEXT\_SIZE **\*** **sizeof(char));**

fgets**(**text**,** TEXT\_SIZE**,** stdin**);**

scanf**(**"%s"**,** str**);**

1. С помощью функции стандартной библиотеки *strtok* выполняется разбиение текста на отдельные слова и запись их в массив

**char\*\*** words\_arr **=** **(char\*\*)** malloc**(sizeof(char\*)** **\*** WORDS\_COUNT**);**

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

**size\_t** i**;**

**for(**i **=** **0;** token **!=** NULL**;** i**++)** **{**

**if(**i **==** WORDS\_COUNT **\*** k **-** **1)** **{**

k**++;**

words\_arr **=** **(char\*\*)** realloc**(**words\_arr**,**

WORDS\_COUNT **\*** k **\*** **sizeof(char\*));**

**}**

words\_arr**[**i**]** **=** token**;**

token **=** strtok**(**NULL**,** " ."**);**

**}**

1. С помощью функции стандартной библиотеки *qsort* выполняется сортировка слов по возрастанию

qsort**(**words\_arr**,** i**,** **sizeof(char\*),** cmpstr**);**

1. С помощью функции стандартной библиотеки *bsearch* выполняется бинарный поиск нужного слова в массиве

**char\*** ptr **=** **(char\*)** bsearch**(&**str**,** words\_arr**,** i**,**

**sizeof(char\*),** cmpstr**)**

1. В качестве вспомогательной функции, передающейся в качестве аргумента в *qsort* иbsearch реализована функция-компаратор, срывающая две строки

**int** cmpstr**(const** **void\*** a**,** **const** **void\*** b**)** **{**

**return** **(**strcmp**(\*(char\*\*)** a**,** **\*(char\*\*)** b**));**

**}**

1. После поиска слова в массиве программа возвращает результат поиска

**if(**ptr **!=** NULL**)**

printf**(**"exists"**);**

**else**

printf**(**"doesn't exist"**);**

1. Полный код программы
2. *#include <stdio.h>*
3. *#include <stdlib.h>*
4. *#include <string.h>*
5. *#define TEXT\_SIZE 1001 //макс размер текста*
6. *#define STR\_SIZE 31 //макс размер слова*
7. *#define WORDS\_COUNT 10 //начальное количество слов в массиве*
8. **int** cmpstr**(const** **void\*** a**,** **const** **void\*** b**)** **{***//функция преобразования void\* в char\**
9. **return** **(**strcmp**(\*(char\*\*)** a**,** **\*(char\*\*)** b**));**
10. **}**
11. **int** main**()** **{**
12. **int** k **=** **1;**
13. **char\*** str **=** **(char\*)** malloc**(**STR\_SIZE **\*** **sizeof(char));**
14. **char\*** text **=** **(char\*)** malloc**(**TEXT\_SIZE **\*** **sizeof(char));**
15. fgets**(**text**,** TEXT\_SIZE**,** stdin**);***//читаем текст*
16. scanf**(**"%s"**,** str**);***//читаем слово*
17. text**[**strlen**(**text**)-1]** **=** '\0'**;**
18. **char\*\*** words\_arr **=** **(char\*\*)** malloc**(sizeof(char\*)** **\*** WORDS\_COUNT**);**
19. **char\*** token **=** strtok**(**text**,** " ."**);**
20. **size\_t** i**;**
21. **for(**i **=** **0;** token **!=** NULL**;** i**++)** **{***//каждое слово заносим в массив*
22. **if(**i **==** WORDS\_COUNT **\*** k **-** **1)** **{***//если массив заканчивается, увеличиваем его размер*
23. k**++;**
24. words\_arr **=** **(char\*\*)** realloc**(**words\_arr**,** WORDS\_COUNT **\*** k **\*** **sizeof(char\*));**
25. **}**
26. words\_arr**[**i**]** **=** token**;**
27. token **=** strtok**(**NULL**,** " ."**);**
28. **}**
29. qsort**(**words\_arr**,** i**,** **sizeof(char\*),** cmpstr**);***//сортируем слова*
30. **char\*** ptr **=** **(char\*)** bsearch**(&**str**,** words\_arr**,** i**,** **sizeof(char\*),** cmpstr**);***//и ищем нужное слово в массиве*
31. **if(**ptr **!=** NULL**)**
32. printf**(**"exists"**);**
33. **else**
34. printf**(**"doesn't exist"**);**
35. free**(**text**);**
36. free**(**str**);**
37. free**(**words\_arr**);**
38. **return** **0;**
39. **}**
40. В локальный репозиторий git была добавлена папка с исходным кодом программы

git add kutsenok\_leonid\_sem2\_lr1

1. Сделан коммит в локальный репозитрий

git commit

1. Изменения отправлены в удаленный репозиторий

git push origin kutsenok\_leonid\_sem2\_lr1

1. Сделан *pull request* в ветку *master*

**Выводы.**

В ходе лабораторной работы изучены возможности стандартной библиотеки языка Си. Полученные знания закреплены в ходе разработки программы для анализа текста на предмет наличия в нем указанного слова с использованием функций стандартной библиотеки.