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 Мукатанов А.В.

Преподаватель Берленко Т.А.

Санкт-Петербург
2021

Цель работы.

Изучить стандартную библиотеку языка Си, освоить ее возможности, используя основные функции этой библиотеки в программном коде.

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.

Основные теоретические положения.

Основной функцией, благодаря которой работает данный код, является компаратор.Компаратор работает по следующему принципу: если элементы равны, результатом сравнения будет 0, если первый больше - результат 1 или любое положительное число, иначе -1 или любое отрицательное.

Так же используется функция для сортировки и поиска в массиве – qsort.Функция принимает указатель на начальный элемент массива, количество элементов и размер одного элемента, а также указатель на функцию для сравнения двух элементов.

Выполнение работы.

1) Для того, чтобы реализовать ввод текста и слова, выделим память под строку ***char text1***, которая будет хранить исходный текст, и под строку ***char* str***, которая будет хранить введенное пользователем слово.

2)Считаем строки с помощью функций ***fgets()***, которые считывают символы из потока и сохраняют их в виде строки до тех пор, пока не наступит конец строки. После ввода текста пользователь перейдет на новую строку..

3) Так как перед нуль-терминатором запишется символ перевода строки, то присвоим ему значение нуль-терминатора.

4) Разбивать текст на слова будем с помощью алгоритма с использованием функции ***strtok()*** из заголовочного файла ***<string.h>***. Инициализируем переменную char* tok, которая будет хранить возвращаемый функцией ***strtok()*** указатель .

5) Далее выделим память на динамический массив указателей на строки ***char** text2***, который будет хранить указатели на найденные лексемы, с помощью функции стандартной библиотеки языка ***calloc().***Если выделенная память заканчивается, расширяем ее с помощью функции ***realloc().*** Таким образом, вызывая функцию ***strtok()*** в цикле, мы заполним массив указателями на слова из текста.

6) Далее для сортировки массива с указателями используем функцию qsort().В качестве аргументов в данную функцию передается: массив(text2), количество элементов и функцию компаратор.

7)Объявим функцию-компаратор compare. Возвращаемое значение int регулируется с помощью функции ***strcmp()*** из заголовочного файла <string.h> .Аргументами функции будут два элемента ***const void* str1, const void* str2***, для того, чтобы компаратору могли передаваться указатели любого типа.

8) После вызова функции ***qsort()*** массив ***char** text2*** отсортируется. И можно приступать к двоичному поиску слова в тексте.

9) Инициализируем переменную ***char** item***, которая будет хранить в себе возвращенный функцией ***bsearch()*** указатель на слово, совпадающее с ключом

поиска, если таковое нашлось. Если искомое слово не найдется, то вернется ***NULL***.

10) В аргументы функции ***bsearch()*** передадим указатель на слово-ключ поиска ***&str***, указатель на первый элемент массива, в котором будет выполняться поиск ***text2***, количество элементов массива ***i***, размер элементов массива ***sizeof(char**),*** а также функцию-компаратор, которая будет сравнивать элементы массива с самим ключом ***compare***(определенная нами на предыдущем шаге функция).

11) Таким образом, если в ***char** item*** запишется ненулевой указатель, то значит, слово, совпадающее с ключом поиска присутствует в тексте, т.е. выводим “exists”; если же ***bsearch()*** вернул ***NULL***, то выводим “doesn’t exist”. Эти действия можно записать с помощью конструкции.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	Java is a general-purpose computer programming language that is concurrent class-based object-oriented and specifically designed to have as few implementation dependencies as	exists	Ответ верный

	possible. It is intended to let application developers "write once, run anywhere" (WORA) meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2016 Java is one of the most popular programming languages in use particularly for client-server web applications, with a reported 9 million developers. Java was originally developed		

	by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems Java platform. is		
2.	I was walking in park. forest	Doesn’t exist	Ответ верный

Выводы.

Была изучена стандартная библиотека языка программирования Си и использована при написании программы.

Разработана программа, выполняющая поставленную задачу, а именно считывание с клавиатуры исходного текста и поиск в нем вводимого слова .

Для решения этой задачи были использованы следующие функции стандартной библиотеки : для разрезания текста на слова была использована функция strtok(), для сортировки слов была использована функция qsort() , а для поиска вводимого слова использовалась функция bsearch().

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb1_prog.c

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define BUF 10

int compare(const void* str1, const void* str2);

int main(){
    char* text1 = malloc(1001*sizeof(char));
    char* str = malloc(31*sizeof(char));
    fgets(text1, 1001, stdin);
    fgets(str, 31, stdin);
    text1[strlen(text1)] = '\0';
    str[strlen(str)] = '\0';
    char* tok;
    tok = strtok(text1, " .");
    int buf = BUF;
    int i = 0;
    char** text2 = calloc(buf, sizeof(char*));
    while(tok != NULL){
        text2[i] = tok;
        i++;
        if(i == buf - 1){
            buf += BUF;
            text2 = realloc(text2,buf*sizeof(char*));
        }
        tok = strtok(NULL, " .");
    }
    qsort(text2, i, sizeof(char*), compare);
    char** item = bsearch(&str, text2, i, sizeof(char**), compare);

    if(item != NULL){
        printf("exists");
    }
    else{
        printf("doesn't exist");
    }
    free(text1);
    free(text2);
    free(str);
    return 0;
}

int compare(const void* str1, const void* str2){
```

```
    return strcmp(*(char**)str1, *(char**)str2);
}
```