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

Студентка гр. 2384 _____ Валеева А. А.

Преподаватель _____ Гаврилов А.В.

Санкт-Петербург
2023

Цель работы.

Изучить стандартную библиотеку *stdlib.h* языка Си, а так же познакомиться с ее функциями. Получить практические навыки работы *qsort*, *besearch*.

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.

Выполнение работы.

Объявим заголовки требуемых библиотек.

Для начала создаем двумерный динамический массив для сохранения в него слов. Для выделения памяти используем функцию *malloc()*. Принимаем на вход строку с помощью функции *fgets()* и с помощью цикла *while()* и функция *strtok()* делим ее на отдельные слова, которые помещаем в массив. Так же подсчитывает количество слов (переменная *count_words*), которую

используем за индекс. Далее принимаем слово (переменная key), которое будем искать в массиве слов. Выделяем для него динамическую память с помощью *malloc()*. Принимаем строку с помощью *fgets(),* которая в конец слова дописывает «\n», которое мы заменяем на «\0». Далее для использования бинарного поиска сортируем массив с помощью функции *qsort(),* которой на вход мы передаем адрес нулевого элемента массива слов (переменная *text*), количество слов (переменная *count_words),* размер *char**, а так же компаратор *compare.* В нем использую функцию сравнения строк *strcmp()*, которая возвращает следующие значения: 1, 0 либо -1. Теперь в отсортированном массиве может применить *besearch()*. В нее мы передаем следующие аргументы: адрес слова, которое мы ищем, адрес нулевого элемента текста, количество слов и компаратор *cmp*. Записываем результат в переменную *res*, которая примет значение *NULL*, если слово в массиве не найдено и не NULL в противоположной ситуации. Далее очищаем память, выделенную динамически.

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 possible. It is intended to let application developers	exists	Верный ответ

	"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.	Java is a general-purpose computer programming language that is concurrent class-based. `hello`	doesn't exist	Верный ответ

Выводы.

Была изучена стандартная библиотека *stdlib.h* языка Си, а так же мы научились работать с ее функциями. Получены практические навыки работы с *qsort*, *besearch*.

Разработана программа, выполняющая считывание строки, разбиение ее на слова, а также сортировка массива слов и бинарный поиск введенного слова.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Valeyeva_Alina_lb1.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define TEXT_SIZE 1000

int cmp(const void *key, const void *b){ // ключ, который передан
- указатель имеет тип char *
    const char *k = (const char *) key;
    const char **p2 = (const char **) b;
    return strcmp(k, *p2);
}

int compare(const void* a, const void* b){
    const char** first = (const char**)a;
    const char** second = (const char**)b;
    return strcmp(*first, *second);
}

int main(){
    char** text = (char**) malloc((TEXT_SIZE) * sizeof (char *));
    int count_words = 0;
    char str[1000];
    fgets(str, 1000, stdin);
    char* word;
    word = strtok(str, ". ");
    while (word != NULL){
        char* word_to_dict = malloc((strlen(word)+1)*sizeof
(char));
        strcpy(word_to_dict, word);
        text[count_words] = word_to_dict;
        count_words++;
        word = strtok(NULL, ". \n"); // получаем новое слово
    }
    char* key = malloc(sizeof (char) * 31);
    fgets(key, 31, stdin);
    if(key[strlen(key)-1] == '\n')
        key[strlen(key)-1] = '\0';
    qsort(text, count_words, sizeof (char *), compare);
    char **res = bsearch(key, text, count_words, sizeof (char *),
cmp);

    if (res != NULL){
        puts("exists");
    }
    else{
        puts("doesn't exist");
    }

    for (int i = 0; i < count_words; i++){
```

```
            //printf("%s\n" , text[i]);
            free(text[i]);
        }
        free(text);
        free(key);
    }
```