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. 1304 _____ Нго Тхи Йен

Преподаватель _____ Чайка К.В.

Санкт-Петербург

2022

Цель работы.

Освоить работу со стандартными библиотека языка С и научиться использовать функции данных библиотек в своих программах.

Задачи рабочи

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
- Отсортивать слова, используя алгоритм быстрой сортировки
- Определить, присутсвует ли в тескте str, используя алгоритм двоичного поиска
- Вывести строку “exits”, если str в тескте есть и “doesn’t exits” в противном случае

Основные теоретические положения.

qsort *<stdlib.h>*

Как мы уже говорили, в stdlib.h есть функции для сортировки и поиска в массиве любого типа. Давайте рассмотрим как такое возможно на примере функции qsort:

```
void qsort (void* base, size_t num, size_t size,int (*compar)(const void*,const void*));
```

Функция принимает указатель на начальный элемент массива, количество элементов и размер одного элемента, а также указатель на функцию для сравнения двух элементов.

Так как тип элементов может быть любым, то и указатель на первый элемент массива имеет тип void. Это позволяет, зная адрес первого элемента и размер каждого элемента вычислить адрес любого элемента массива в памяти и обратиться к нему. Остается только сравнить 2 элемента имея 2 указателя на них. Это выполняет функция compar, указатель на которую передается функции qsort в качестве одного из параметров.

Выполнение работы.

Для выполнения поставленной задачи, необходимо подключить дополнительные библиотеки. Библиотеку <stdlib.h> и <*string.h*>.
В главной функции *int main()* объявляется массив символов *char text* размером 1000, в котором будет храниться вводимый пользователем текст, и массив символов *char str* размером 30, в котором будет храниться вводимая пользователем строка, которая будет искаться в вводимом пользователем тексте. С помощью функции *fgets()* считываются вводимый пользователем текст в *text* и вводимая пользователем строка в *str*. В *char*str_new* сохраняется строка *str* без пробелов и знаков переноса строки с помощью функции *strtok()*.

С помощью цикла *while()* и функции *strtok()* массив *text* разбивается на строки, не содержащие точку и пробелы. Данное разбиение сохраняется в массив строк *words*. С помощью функции *qsort()* происходит сортировка массива words. В функцию подается первым аргументом массив, который необходимо отсортировать, вторым аргументом – длина данного массива, третьим аргументом – размер одного элемента массива и последним аргументом – функция *cmp*.

С помощью функции *bsearch()* происходит бинарный поиск элемента *str_new* в массиве *words*. В функцию подается первым аргументом элемент, который необходимо найти в массиве, вторым аргументом – массив, в котором необходимо найти данный элемент, третьим аргументом – длина данного массива, четвертым аргументом – размер одного элемента массива и последним аргументом – функция *cmp*.

Результат работы функции *bsearch()* сохраняется в переменную *char** *result*.

Ход работы

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

```
int cmp(const void* a, const void* b)
{
	const char** word_1 = (const char**) a;
	const char** word_2 = (const char**) b;
	return strcmp(*word_1, *word_2);
}
int main(){
	char text[1000];
	char str[30];
	fgets(text, sizeof(char)*1000, stdin);
	fgets(str, sizeof(char)*30, stdin);
	char* str_new = strtok(str, " .\n");
	char* word = strtok(text, " .");
	int count_word = 0;
	char** words;
	words = malloc(sizeof(char*) * count_word);
	while(word){
		count_word++;
		words = realloc(words, sizeof(char*) * count_word);
		words[count_word-1] = word;
		word = strtok(NULL, " .");
	}
	qsort(words, count_word, sizeof(char*), cmp);
	char** result = bsearch(&str_new, words, count_word,
	sizeof(char*), cmp);
	free(words);
	if(result)
		printf("exists");
	else
		printf("doesn't exist");
	return 0;
}
```

Вывод

В ходе лабораторной работы были изучены способ сортировки массива, функцию стандартной библиотеки для подсчёта времени работы части хода программы.