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: Регулярные выражения

Студент гр. 3344 _____ Фоминых Е.Г.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Целью работы является использование регулярных выражений в программе на языке Си для нахождения и вывода искомой информации во входящем потоке символов.

Задание

Вариант 1. На вход программе подается текст, представляющий собой набор предложений с новой строки. Текст заканчивается предложением "Fin." В тексте могут встречаться ссылки на различные файлы в сети интернет. Требуется, используя регулярные выражения, найти все эти ссылки в тексте и вывести на экран пары <название_сайта> - <имя_файла>. Гарантируется, что если предложение содержит какой-то пример ссылки, то после ссылки будет символ переноса строки.

Ссылки могут иметь следующий вид:

- Могут начинаться с названия протокола, состоящего из букв и :// после
- Перед доменным именем сайта может быть www
- Далее доменное имя сайта и один или несколько доменов более верхнего уровня
- Далее возможно путь к файлу на сервере
- И, наконец, имя файла с расширением.

Выполнение работы

Для выполнения работы подключаем все необходимые библиотеки: <stdio.h>, <stdlib.h>, <string.h>, <regex.h>.

В строке char *sentence = malloc(size_sent * sizeof(char)) выделяем динамическую память под массив символов sentence, размер которого определяется переменной size_sent. Во строке char **text = malloc(size_text * sizeof(char*)) выделяем динамическую память под массив указателей на символы text, размер которого определяется переменной size_text. Считываем посимвольно, увеличивая счетчик символов и записывая каждый символ в массив символов sentence. Когда встречаем символ ‘\n’, то предложение заканчивается, добавляем его в массив предложений text. Одновременно проверяем не равен ли массив символов sentence строке “Fin.” с помощью функции стандартной библиотеки strcmp(). Если равен, то ввод закончен. Объявляем переменную regex_t regex для хранения скомпилированного регулярного выражения. Переменная size_t nmatch = 7 отвечает за максимальное количество групп захвата. Массив size_t match используется для хранения информации о соответствиях каждой группы в регулярном выражении. Функция regcomp(&regex, str, REG_EXTENDED) принимает на вход адрес структуры regex_t, в которую будет сохранено скомпилированное регулярное выражение, указатель на строку с регулярным выражением, флаг, который указывает на использование расширенного (extended) синтаксиса регулярных выражений. Если компиляция успешна, то функция возвращает 0. Функция regexec(&regex, text[b], nmatch, match, 0) принимает на вход адрес структуры regex_t, в которую сохранено скомпилированное регулярное выражение, элемент массива указателей text, максимальное кол-во групп захвата, массив match, состоящий из 7 элементов(7 групп захвата), 0 в качестве флага означает, что выполняется стандартное сопоставление регулярного выражения без дополнительных настроек. Если соответствие найдено, то функция возвращает 0.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	https://wandbox.org/arriruyu.txt super eto tochno https://www.bilibili.com/video/BV1ox411R7KJ/?spm_id_from=333.337.search-card.all.click https://1.shkolkovo.online/dz.pdf Fin.	wandbox.org - arriruyu.txt shkolkovo.online - dz.pdf	Вывод верный
2.	Однажды, в далеком лесу, где все казалось знакомым и безопасным, обитала небольшая группа удивительных животных - куниц. Они были не похожи на других лесных обитателей, их шерсть была мягкой и блестящей, а глаза - яркими и умными. https://ru.wikipedia.org/wiki/Kunici.jpg Fin.	ru.wikipedia.org - Kunici.jpg	Вывод верный

Выводы

Изучена работа с регулярными выражениями и программами для использования созданных шаблонов.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: src.c

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

int main() {
    int size_sent = 10;
    int size_text = 10;
    char *sentence = malloc(size_sent * sizeof(char));
    char **text = malloc(size_text * sizeof(char*));
    int a = 0;
    int i = size_sent;
    int j = size_text;
    int sents_count = 0;
    while (1) {
        char symbol = getchar();
        if (symbol == '\n'){
            sentence[a] = '\0';
            text[sents_count] = malloc((a + 1) * sizeof(char));
            strcpy(text[sents_count], sentence);
            sents_count++;
            a = 0;
            if (sents_count >= j){
                j += size_text;
                text = realloc(text, j * sizeof(char*));
            }
        } else{
            sentence[a] = symbol;
            a++;
            if (a >= i){
                i += size_sent;
                sentence = realloc(sentence, i * sizeof(char));
            }
        sentence[a] = '\0';
        if (strcmp(sentence, "Fin.\0") == 0){
            break;
            }

        }
    }
regex_t regex;
    char *str = "([a-zA-Z]+:\\/\\/)?(www\\.)?(([a-zA-Z0-9]+\\.)+[a-zA-Z0-
9]+)\\/([a-zA-Z0-9]+\\/)*([a-zA-Z0-9]+\\.[a-zA-Z0-9]+)";
    size_t nmatch = 7;
    regmatch_t match[7];
    if (regcomp(&regex, str, REG_EXTENDED) == 0){
        for (int b=0; b<sents_count; b++){
            if (regexec(&regex, text[b], nmatch, match, 0) == 0){
                for (int t = match[3].rm_so; t<match[3].rm_eo; t++){
                    printf("%c", text[b][t]);
                }
                printf(" - ");
```

```
                for (int t = match[6].rm_so; t<match[6].rm_eo; t++){
                    printf("%c", text[b][t]);
                }
            printf("\n");
            }
        }
    }
    for (int b = 0; b < sents_count; b++){
        free(text[b]);
    }
    regfree(&regex);
    return 0;
}
```