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

Студентка гр. 3344 _____ Щербак М.С.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Целью работы является освоение работы с регулярными выражениями в языке Си на примере использующей их программы.

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

Подключим стандартные библиотеки *stdio.h* для работы с вводом, *string.h* для работы со строками, а также *regex.h* для работы с регулярными выражениями.

Объявлены переменные, включая массивы для хранения текста, ввода, регулярного выражения и результатов сопоставления с регулярным выражением. С помощью функции regcomp() компилируется регулярное выражение, переданное в виде строки regexString. В цикле while происходит чтение текста из стандартного ввода (пользователь вводит текст до появления строки "Fin.") и конкатенация текста в переменную input. Затем начинается поиск URL-адресов в тексте. Цикл while продолжается до тех пор, пока есть совпадения с регулярным выражением. Если найдено совпадение, то извлекаются части URL-адреса (домен и путь) и выводятся на экран с помощью функции printf(). После обработки текущего URL адреса указатель url сдвигается к следующему совпадению. Если не найдено ни одного URL-адреса, программа выводит сообщение "No URLs found." После завершения поиска URL-адресов освобождается память, занимаемая регулярным выражением, с помощью функции regfree(). Программа завершает свою работу и возвращает 0.

Таким образом, данная программа анализирует ввод пользователя на наличие URL-адресов и выводит найденные адреса на экран

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	This is simple url: http://www.google.com/track.mp3 May be more than one upper level domain http://www.google.com.edu/hello.avi Many of them. Rly. Look at this! http://www.qwe.edu.etu.yahooo.org.net.ru/qwe.q Fin.	google.com - track.mp3 google.com.edu - hello.avi qwe.edu.etu.yahooo.org.net.ru - qwe.q	Данные обработаны корректно
2.	Some other protocols ftp://skype.com/qqwe/qweqw/qwe.avi https://www.twitch.tv/bratishkinoff Fin.	skype.com - qwe.avi	Данные обработаны корректно

Выводы

Были изучена работа с регулярными выражениями. А также была создана программа, в которой реализовано считывание и вывод строк, подходящих под заданный шаблон.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb1.c

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

int main() {
    char text[10000];
    char input[10000] = "";
      char *regexString = "((http|https|ftp):\\/\\/)?(www\\.)?(([a-zA-Z]
+\\.)+[a-zA-Z]+)(\\/[a-zA-Z]+)*(\\/([a-zA-Z]+\\.[a-zA-Z0-9_+=!-]+))";
    regex_t regex;
    regmatch_t groupArray[9];
    int match;

    regcomp(&regex, regexString, REG_EXTENDED);

    while (1) {
        fgets(text, 10000, stdin);

        if (strstr(text, "Fin.") != NULL)
            break;

        strcat(input, text);
    }
    char *url = input;
    int urlCount = 0;
    while (1) {
        match = regexec(&regex, url, 9, groupArray, 0);
        if (match == REG_NOMATCH)
            break;

        if (groupArray[4].rm_so != -1 && groupArray[4].rm_eo != -1 &&
            groupArray[8].rm_so != -1 && groupArray[8].rm_eo != -1) {
            printf("%.*s - %.*s\n",
                   (int)(groupArray[4].rm_eo - groupArray[4].rm_so),
                   url + groupArray[4].rm_so,
                   (int)(groupArray[8].rm_eo - groupArray[8].rm_so),
                   url + groupArray[8].rm_so);
        }

        url += groupArray[0].rm_eo;
        urlCount++;
    }

    if (urlCount == 0) {
        printf("No URLs found.\n");
    }

    regfree(&regex);
    return 0;
}
```