САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Программирование»

Тема: Лабораторная работа № 1. Регулярные выражения

Студент гр. 3343 _____ Отмахов Д. В.

Преподаватель _____ Государкин Я. С.

Санкт-Петербург

2024

Цель работы

Научиться использовать функции библиотеки *regex.h* языка Си, написав программу.

Задание

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

Описание функций:

- Функия *int main()*: в *regexString* хранится регулярное выражение, по которому отбираются строчки. С помощью функии *regcomp* сохраняется обработанное регулярное выражение в *regexCompiled*. Далее выполняется построчное считывание текста при помощи функции *readLine()* до предложения «*Fin.*», и каждая строка проверяется по регулярному выражению с помощью функции *regexec*. Затем с помощью цикла поизводится проход по группам и выводится строчка в формате <название_сайта> - <имя_файла>.

- Функция *char *readLine()*: динамически считывает строку и возвращает ее.

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.	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 Some other protocols ftp://skype.com/qqwe/qweqw/qwe.avi Fin.	google.com - track.mp3 google.com.edu — hello.avi qwe.edu.etu.yahooo.org.net.ru - qwe.q skype.com - qwe.avi	Ожидаемый вывод.
2.	This is simple url: http://www.google.com/main/track.mp3 May be more than one upper level domain http://www.google.com.edu.eee/hello.avi Rly. Look at this! Fin.	google.com - track.mp3 google.com.edu.eee - hello.avi	Ожидаемый вывод.
3.	This is simple url: http://www.google.com/track.mp3 May be more than one upper level domain http://www.google.com.edu/hello.a	google.com - track.mp3 google.com.edu - hello.avi qwe.edu.bab.ddd.etu.yahooo.org.net.ru - qwe.q	Ожидаемый вывод.

	vi Rly. Look at this! This is simple url: http://www.qwe.edu.bab.ddd.etu.yahooo.org.net.ru/wwr/qwe.q Fin.		

Выводы

В ходе выполнения лабораторной работы были изучены основные функции библиотеки *regex.h*, а также освоен навык их использования.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

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

#define    REGEX_STRING    "([A-Za-z]+:\\/\\/)?(w{3}\\.)?(([A-Za-z0-
9\\-]+\\.)+[A-Za-z0-9\\-]+)\\/([A-Za-z0-9\\-]+\\/)*([A-Za-z0-9\\-]+\\.[A-
Za-z0-9\\-]+)"


char *readLine(){
    size_t cnt = 0, capacity = 1;
    char *line = (char *)malloc(sizeof(char) * capacity);
    char ch;

    while((ch = getchar()) != '\n'){
        line[cnt++] = ch;
        if(cnt == capacity){
            capacity += 1;
            line = (char *)realloc(line, sizeof(char) * capacity);
        }
        if(strcmp(line, "Fin.") == 0)
            return line;
    }

    return line;
}


int main(){
    char *line;
    char *regexString = REGEX_STRING;
    size_t maxGroups = 7;

    regex_t regexCompiled;
    regmatch_t groupArray[maxGroups];

    regcomp(&regexCompiled, regexString, REG_EXTENDED);

    while(strcmp((line = readLine()), "Fin.") != 0){
        if(regexec(&regexCompiled, line, maxGroups, groupArray, 0)
== 0){
            for(size_t    i    =    groupArray[3].rm_so;    i    <
groupArray[3].rm_eo; i++)
                printf("%c", line[i]);
            printf(" - ");
            for(size_t    i    =    groupArray[6].rm_so;    i    <
groupArray[6].rm_eo; i++)
                printf("%c", line[i]);
            printf("\n");
        }
        free(line);
    }
```

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