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 _____ Анахин Е.Д.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Научиться использовать регулярные выражения в языке программирования C.

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

Создается регулярное выражение и происходит его компиляция. Далее производится считывание предложений от пользователя. Каждая отдельная строка рассматривается как предложение. Для каждого предложения программа пытается найти строку, которая подходит под шаблон регулярного выражения. Далее производится проверка, что в выражении, которое нашла программа нет двух подряд идущих точек, выражение не начинается с точки и не заканчивается ею. И происходит проверка, что в выражении есть хотя бы одна точка, чтобы было разделение домена и доменного имени. Дальше, если все условия выполняются, то происходит вызов функции printResFormatted, которая находит третью и пятую группу и выводит результат в формате *домен — название файла.расширение файла*.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Коммен тарии
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	-

Выводы

Был получен опыт работы с регулярными выражениями в языке C и была создана программа, которая находит URL — пути до файлов.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main_lb1.c

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

#define REGEX "([A-Za-z]+:\\/\\/)?(www\\.)?([a-zA-Z\\.\\-\\_]+)\\/([A-Za-
z\\-\\_]+\\/)*([A-Za-z0-9\\-\\_]+\\.[A-Za-z0-9\\-]+)$"

void printResFormatted(regmatch_t match[8], char *url, int isNewLine) {
    if (match[3].rm_so != -1 && match[5].rm_so != -1) {
        char group2[match[3].rm_eo - match[3].rm_so + 1];
        char group6[match[5].rm_eo - match[5].rm_so + 1];
              strncpy(group2,  url  +  match[3].rm_so,  match[3].rm_eo  -
match[3].rm_so);
              strncpy(group6,  url  +  match[5].rm_so,  match[5].rm_eo  -
match[5].rm_so);
        group2[match[3].rm_eo - match[3].rm_so] = '\0';
        group6[match[5].rm_eo - match[5].rm_so] = '\0';
        if (!isNewLine && strstr(group2, ".")) {
            printf("%s - %s", group2, group6);
        } else if (strstr(group2, ".")) {
            printf("\n%s - %s", group2, group6);
        }
    } else {
        printf("Не удалось найти соответствия для доменного имени и имени
файла\n");
    }
}

int main() {
    char** sentences = malloc(10 * sizeof(char*));
    int len = 0;
    regex_t regex;
    regmatch_t match[8];

    regcomp(&regex, REGEX, REG_EXTENDED);

    while (1) {
        if (len % 10 == 8) {
            sentences = realloc(sentences, (len + 10) * sizeof(char*));
        }

        sentences[len] = malloc(1000);
        scanf("%s", sentences[len]);

        if (strcmp(sentences[len], "Fin.") == 0) {
            break;
        }

        len++;
```

```
    }

    int isfirst = 0;
    char* currentSentence = malloc(1000);

    for (int i = 0; i < len; i++) {
        if (regexec(&regex, sentences[i], 8, match, 0) == 0) {
            strcpy(currentSentence, sentences[i] + match->rm_so);
            if (strstr(currentSentence, "..") || !strstr(currentSentence,
"."))        ||        (currentSentence[0]        ==        '.'        ||
(currentSentence[strlen((currentSentence))-1] == '.'))) {
                continue;
            } else {
                printResFormatted(match, sentences[i], isfirst++);
            }
        }
    }

    for (int i = 0; i < len; i++) {
        free(sentences[i]);
    }
    free(sentences);

    return 0;
}
```