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. 3342 _____ Колесниченко М.А.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Целью работы является изучение принципов работы регулярных выражений и использование их в программе на языке 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

Было написано регулярное выражение, с помощью которого проверялась валидность ссылки на файл. Программа считывает предложения, пока не будет введено предложение, означающее конец ввода. Каждое предложение проверяется с помощью регулярного выражения на предмет присутствия валидной ссылки, и затем с помощью отдельной функции и групп захвата выводится нужная информация.

Программа выводит подходящие ссылки в формате <название_сайта> - <имя_файла>.

Разработанный программный код см. в приложении А.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
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

Было написано регулярное выражение, изучены способы работы с ним и с группами захвата в языке программирования C.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

int match(char *string, char *pattern, regex_t *re, regmatch_t
matchptr[], size_t nmatch)
{
    int s;
    s = regcomp(re, pattern, REG_EXTENDED);
    if (s != 0) {
        return s;
    }
    s = regexec(re, string, nmatch, matchptr, 0);
    return s;
}

void print_matched_substring(char *string, regmatch_t match)
{
    for (int i = match.rm_so; i < match.rm_eo; i++) {
        printf("%c", string[i]);
    }
}

int main()
{
    regex_t re;
    regmatch_t matchptr[9];
    int retval;
    char *pattern = "([a-zA-Z]+://)?(www\\.)?([a-zA-Z0-9-]+(\\.[a-
zA-Z0-9]+)+)/((\\w+/)*)([a-zA-Z0-9-]+(\\.[a-zA-Z0-9-]+))*\n$";
    int sentence_counter = 0;
    char **text = malloc(sizeof(char *));
    char *sentence = malloc(sizeof(char) * 1000);

    if (text == NULL || sentence == NULL) {
        fprintf(stderr, "Memory allocation failed\n");
        return 1;
    }

    while (fgets(sentence, 1000, stdin))
    {
        if (strcmp(sentence, "Fin.\n") == 0)
        {
            break;
        }
        sentence_counter++;
        char **temp_text = realloc(text, sizeof(char *) *
sentence_counter);
        if (temp_text == NULL) {
```

```
            fprintf(stderr, "Memory allocation failed\n");
            free(sentence);
            free(text);
            return 1;
        }
        text = temp_text;
        text[sentence_counter - 1] = strdup(sentence);
        if (text[sentence_counter - 1] == NULL) {
            fprintf(stderr, "Memory allocation failed\n");
            free(sentence);
            for (int i = 0; i < sentence_counter - 1; i++) {
                free(text[i]);
            }
            free(text);
            return 1;
        }
        sentence = malloc(sizeof(char) * 1000);
        if (sentence == NULL) {
            fprintf(stderr, "Memory allocation failed\n");
            for (int i = 0; i < sentence_counter; i++) {
                free(text[i]);
            }
            free(text);
            return 1;
        }
    }

    for (int i = 0; i < sentence_counter; i++)
    {
        retval = match(text[i], pattern, &re, matchptr, 9);
        if (retval == 0)
        {
            print_matched_substring(text[i], matchptr[3]);
            printf(" - ");
            print_matched_substring(text[i], matchptr[7]);
            if (i != sentence_counter -1){
                printf("\n");
            }
        }
    }
    regfree(&re);

    free(sentence);
    for (int i = 0; i < sentence_counter; i++) {
        free(text[i]);
    }
    free(text);

    return 0;
}
```