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 Белаид Фарук

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

#define MAX_SENTENCE_LENGTH 1000
#define MAX_MATCHES 9

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

int findMatch(char *text, char *pattern, regex_t *compiledRegex,
regmatch_t matches[], size_t maxMatches) {
    int result = regcomp(compiledRegex, pattern, REG_EXTENDED);
    if (result != 0) {
        return result;
    }
    result = regexec(compiledRegex, text, maxMatches, matches, 0);
    return result;
}

int main() {
    regex_t compiledRegex;
    regmatch_t matchRegions[MAX_MATCHES];
    int returnValue;
    int sentenceCount = 0;
    char **inputText = NULL;
    char sentence[MAX_SENTENCE_LENGTH];
    char *pattern = "([a-zA-Z]+://)?(www\\.)?([a-zA-Z0-9-]+(\\.[a-
zA-Z0-9]+)+)/((\\w+/)*)([a-zA-Z0-9-]+(\\.[a-zA-Z0-9-]+))*\n$";

    while (fgets(sentence, MAX_SENTENCE_LENGTH, stdin)) {
        if (strcmp(sentence, "Fin.\n") == 0) {
            break;
        }
        sentenceCount++;
        char **tempInputText = realloc(inputText, sizeof(char *) *
sentenceCount);
        if (tempInputText == NULL) {
            fprintf(stderr, "Memory allocation failed\n");
            return 1;
        }
        inputText = tempInputText;
        inputText[sentenceCount - 1] = strdup(sentence);
```

```
        if (inputText[sentenceCount - 1] == NULL) {
            fprintf(stderr, "Memory allocation failed\n");
            for (int i = 0; i < sentenceCount - 1; i++) {
                free(inputText[i]);
            }
            free(inputText);
            return 1;
        }
    }

    for (int i = 0; i < sentenceCount; i++) {
        returnValue = findMatch(inputText[i], pattern,
&compiledRegex, matchRegions, MAX_MATCHES);
        if (returnValue == 0) {
            printMatchedSubstring(inputText[i], matchRegions[3]);
            printf(" - ");
            printMatchedSubstring(inputText[i], matchRegions[7]);
            if (i != sentenceCount - 1) {
                printf("\n");
            }
        }
    }
    regfree(&compiledRegex);

    for (int i = 0; i < sentenceCount; i++) {
        free(inputText[i]);
    }
    free(inputText);

    return 0;
}
```