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

Студентка гр. 3343 _____ Стрижков И. А.

Преподаватель _____ Государкин Я. С.

Санкт-Петербург

2024

Цель работы

Изучить и научиться применять функции библиотеки regex.h языка Си для поиска совпадений в строках при помощи регулярных выражений. Освоить навыки для написания регулярных выражений на языке Си.

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:

- printMatch(char* s, regmatch_t groupArray): Функция для вывода совпадения, найденного регулярным выражением.

- main(): Основная функция программы, выполняющая следующие задачи:

- Инициализация переменных и компиляция регулярного выражения: Создает строку для хранения регулярного выражения и использует regcomp для его компиляции в скомпилированную структуру regex_t.

- Считывание входных данных: Читает строки из стандартного ввода в массив s до тех пор, пока не будет введена строка "Fin.".

- Обработка ввода: Для каждой считанной строки использует regexec для выполнения регулярного выражения и поиска совпадений. Если совпадение найдено, outputs specific matches используя функцию printMatch - сначала домен, затем имя файла.

- Освобождение ресурсов: Вызывает regfree для освобождения ресурсов, связанных с скомпилированным регулярным выражением.

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	Выходные данные соответствуют ожиданиям.
2.	This is simple url: http://www.google.-aaaaaa.com/track.mp3 May be more than one upper level domain http://www.google.com.edu/hello.avi Many of them. youtube.en/file.f Rly. Look at this! Fin.	google.-aaaaaa.com - track.mp3 google.com.edu - hello.avi youtube.en - file.f	Выходные данные соответствуют ожиданиям.
3.	This is simple url: http://www.google.aaaaaa.com//track.mp3 May be more than one upper level	google.com.edu - hello.avi youtube.en - file.f google_google.com.edu	Выходные данные соответствуют

	domain http://www.google.com.edu/hello.avi Many of them. youtube.en/file.f Rly. Look at this! This is simple url: aaa://googleaaaaaacom/a.a May be more than one upper level domain http://www.google_google.com.edu/hello.avi Fin.	- hello.avi	ожиданиям.

Выводы

В ходе выполнения лабораторной работы были освоены необходимые навыки для использования регулярных выражений на языке Си с помощью библиотеки regex.h, а также для составления регулярных выражений согласно требованиям. Были изучены необходимые языковые конструкции и особенности записи регулярных выражений на языке Си.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

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

#define INPUT_SIZE 100
#define NUM_INPUTS 100

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

int main() {
    size_t maxGroups = 100;
    char* regexString = "([A-z]+:\\/\\/)?(w{3}\\.)?([A-z_-]+(\\.[A-
z_-]+){1,})((\\/[A-z]+){1,})?(\\/([A-z]+\\.[A-z0-9]+\n))";
    regex_t regexCompiled;
    char s[NUM_INPUTS][INPUT_SIZE];
    int numInputs = 0;

    if (regcomp(&regexCompiled, regexString, REG_EXTENDED)) {
        printf("can't compile regular expression\n");
        return 0;
    }

    while (fgets(s[numInputs], INPUT_SIZE, stdin) && numInputs <
NUM_INPUTS) {
        if (strstr(s[numInputs], "Fin.") != NULL){
            break;
        }
        numInputs++;
    }

    regmatch_t groupArray[maxGroups];
```

```
    for (int i = 0; i < numInputs; i++) {
        if (regexec(&regexCompiled, s[i], maxGroups, groupArray, 0)
== 0) {
            printMatch(s[i], groupArray[3]);
            printf(" - ");
            if (groupArray[8].rm_so != -1) {
                printMatch(s[i], groupArray[8]);
            }
        }
    }

    regfree(&regexCompiled);

    return 0;
}
```