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. 3341 _____ Кудин А.А.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Эта работа направлена на практическое применение и понимание регулярных выражений через разработку соответствующего программного обеспечения на языке C. Чтобы достичь этой цели, перед нами стоят следующие задачи:

- Исследовать общепринятые шаблоны и операторы, используемые в регулярных выражениях;
- Разработать эффективное регулярное выражение для специфической задачи;
- Программировать на языке C, включив в код функционал созданного регулярного выражения для решения поставленной задачи.

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

1. Инициализация необходимых библиотек:
    - stdio.h для ввода-вывода;
    - stdlib.h для общих функций языка C;
    - string.h для работы со строками;
    - regex.h для работы с регулярными выражениями.
2. Определение констант:
    - URL_PATTERN содержит регулярное выражение для поиска URL.
    - GROUPS_COUNT определяет количество групп захвата в регулярном выражении.
    - BUFFER_LENGTH задает размер буфера для хранения вводимого текста.
    - END_OF_INPUT задает маркер конца ввода текста.
3. Функция main:
    - Компилирует регулярное выражение и готовит его к использованию.
    - Считывает текст построчно из стандартного ввода до тех пор, пока не встретит строку "Fin.".
    - Для каждой считанной строки выполняет поиск URL с помощью регулярного выражения.
    - Если URL найден, вызывает функцию displayMatches для отображения результатов.
4. Функция displayMatches:
    - Получает текст и массив с информацией о позициях найденных соответствий регулярному выражению.
    - Выводит на экран название сайта и имя файла для каждой найденной ссылки.

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

В процессе выполнения работы было изучено использование регулярных выражений для анализа и извлечения данных из текста. Были освоены основные конструкции и паттерны, необходимые для составления эффективных регулярных выражений, что позволило успешно решить задачу поиска и выделения интернет-ссылок в предоставленном тексте. Это знание может быть применено для решения широкого спектра задач, связанных с обработкой текста, и подтверждает ценность регулярных выражений как инструментария для программирования и анализа данных.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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


const char* URL_PATTERN = "([a-z]+\\:\\/\\/)?(www\\.)?(([a-z]+\\.)+[a-
z]+)\\/([a-z]+\\/)*([a-z]+\\.[a-z0-9]+)";
const int GROUPS_COUNT = 7;
const int BUFFER_LENGTH = 1000;
const char* END_OF_INPUT = "Fin.";

void displayMatches(char* text, regmatch_t* matches);

int main() {
    regex_t regexCompiled;
    regmatch_t matchGroups[GROUPS_COUNT];
    char inputBuffer[BUFFER_LENGTH];

    regcomp(&regexCompiled, URL_PATTERN, REG_EXTENDED);

    while(fgets(inputBuffer, BUFFER_LENGTH, stdin)) {
        if(strncmp(inputBuffer, END_OF_INPUT, strlen(END_OF_INPUT)) == 0)
break;

        if(regexec(&regexCompiled, inputBuffer, GROUPS_COUNT, matchGroups,
0) == 0) {
            displayMatches(inputBuffer, matchGroups);
        }
    }
    regfree(&regexCompiled);
    return 0;
}

void displayMatches(char* text, regmatch_t* matches) {
    printf("%.*s - %.*s\n",
           (int)(matches[3].rm_eo          -          matches[3].rm_so),
&text[matches[3].rm_so],
           (int)(matches[6].rm_eo          -          matches[6].rm_so),
&text[matches[6].rm_so]);
}
```