**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 |  | Охрименко Д. И. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Целью работы является использование регулярных выражений в программе на языке Си для нахождения и вывода искомой информации во входящем потоке символов.

## 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](https://e.moevm.info/mod/lesson/view.php?id=509), 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

Для выполнения работы подключаем все необходимые библиотеки: <stdio.h>, <stdlib.h>, <string.h>, <regex.h>. Создаём массивы, прежде всего input\_buffer для чтения текста из стандартного потока ввода stdin и text, в который будем добавлять (с помощью конкатенации строк) каждую считанную строку. Вводим регулярное выражение, в нём создано 8 групп, из которых две нам действительно нужны, остальные созданы для верного отличия ссылок на интернет ресурсы от прочего материала. Последний массив group\_match будет хранить стуктуры regmatch\_t с интересующими нас группами.

Чтобы начать работу с регулярными выражениями необходимо скомпилировать введённое выражение, преоразовав его до понятного компьютеру машинного кода, учтём, что из-за большого количества логических выражений программа будет работать медленно. Компилируем с флагом REG\_EXTENDED, расширяющий исходный синтаксис выражения.

В последующем цикле while будем сопоставлять регулярное выражение с заданной строкой, если подходящая подстрока будет присутствовать, отловим это с помощью указателей на начальный и конечные символы.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | https://wandbox.org/arriruyu.txt  super eto tochno  https://www.bilibili.com/video/BV1ox411R7KJ/?spm\_id\_from=333.337.search-card.all.click  https://1.shkolkovo.online/dz.pdf  Fin. | wandbox.org - arriruyu.txt  shkolkovo.online - dz.pdf | Вывод верный |
|  | Однажды, в далеком лесу, где все казалось знакомым и безопасным, обитала небольшая группа удивительных животных - куниц. Они были не похожи на других лесных обитателей, их шерсть была мягкой и блестящей, а глаза - яркими и умными.  https://ru.wikipedia.org/wiki/Kunici.jpg  Fin. | ru.wikipedia.org - Kunici.jpg | Вывод верный |

## Выводы

Изучена работа с регулярными выражениями и программами для использования созданных шаблонов.

# Приложение А Исходный код программы

Название файла: Okhrimenko\_Denis.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <regex.h>

int main(){

char\* input\_buffer = (char\*)malloc(sizeof(char) \* 100000);

char\* text = (char\*)malloc(sizeof(char) \* 100000);

char\* pattern = "((http|https|ftp):\\/\\/)?(www\\.)?(([a-zA-Z]+\\.)+[a-zA-Z]+)(\\/[a-zA-Z]+)\*(\\/([a-zA-Z]+\\.[a-zA-Z0-9\_+=!-]+))";

regmatch\_t group\_match[9]; //структура с матчами и группами

regex\_t pointer\_to\_regex;

//Скомпилируем регулярное выражение:

regcomp(&pointer\_to\_regex, pattern, REG\_EXTENDED); //Расширенный синтаксис regex (icase - w\out registr)

while(strstr(fgets(input\_buffer, 100000, stdin), "Fin.") == NULL){

strcat(text, input\_buffer);

}

//finding matches

while(1){

int check = regexec(&pointer\_to\_regex, text, 9, group\_match, 0);

if(check != REG\_NOMATCH)

{

if (group\_match[4].rm\_so != -1 && group\_match[4].rm\_eo != -1 &&

group\_match[8].rm\_so != -1 && group\_match[8].rm\_eo != -1)

{

printf("%.\*s - %.\*s\n",

(int)(group\_match[4].rm\_eo - group\_match[4].rm\_so),

text + group\_match[4].rm\_so,

(int)(group\_match[8].rm\_eo - group\_match[8].rm\_so),

text + group\_match[8].rm\_so);

text += group\_match[0].rm\_eo;

}

} else {

break;

}

}

}