**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 |  | Че М. Б. |
| Преподаватель |  | Глазунов С. А. |

Санкт-Петербург

2024

## Цель работы

Научиться составлять регулярные выражения, работать с группами, писать программы с использованием библиотеки regex.h для работы с регулярными выражениями, находить подходящие строки и извлекать из них необходимую информацию.

## 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

Первым делом необходимо привести текст к удобному для нас виду. Для этого создадим динамический массив text, пока в тексте не будет найдена строчка «Fin.».

После того, как был записан весь текст, необходимо разбить его на отдельные строчки. Для этого создадим динамический sentences, и с помощью strtok разобьем предложение на отдельные строчки и запишем их в массив sentences.

В reg\_sentence будет записано регулярное выражение, которое состоит из 6 групп. Первая группа проверяет, есть ли протокол из букв и (пример https://), вторая проверяет наличие «www.», следующая группа отвечает за полное доменное имя сайта, четвёртая сохраняет последний уровень домена (непятая отвечает за наличие или отсутствие пути к файлу на сервере, шестая отвечает за название файл на сервере

Из 6 групп для решения поставленной задачи необходимо использовать информацию из 3 и 6 групп (доменное имя и файл).

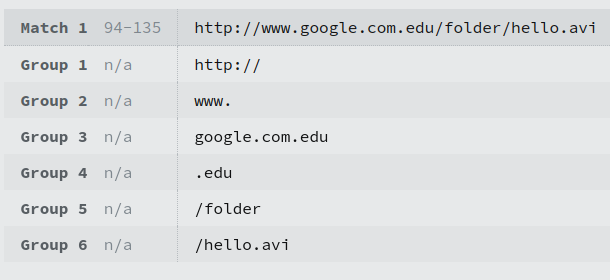


Рисунок 1 – Пример разбиения на группы

Следующим шагом необходимо скомпилировать выражение в форму, подходящую для последующего поиска с помощью функции regexec(). Для этого необходимо вызвать функцию regcomp(), которая скомпилирует выражение в переменную regexCompiled. Если по какой-то причине функция не сможет этого сделать, тогда будет выведено сообщение об ошибке и программа завершит свою работу.

Затем создаём массив groupArray, который будет хранить информацию о каждой строке. Далее пробегаемся по циклу и проверяем наличие ссылка в строке с помощью функции regexec(). Если ссылка была найдена, то с помощью цикла от начальной до конечной позиции посимвольно выводится информация из 3-ей группы (домен) и 6-ой группы (имя файла).

## Выводы

Были изучены принципы создания регулярных выражений, как использовать их в языке C, принципы и методы работы с библиотекой regex.h. Написана программа, которая разбивает текст на строки, находит ссылки с помощью регулярного выражения и выводит их в необходимом формате.

# Приложение А Исходный код программы

Название файла: main.с

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <regex.h>

int main(int argc, char const \*argv[])

{

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

char \*\*sentences = (char \*\*)malloc(sizeof(char \*));

int count = 0;

char symbol;

while (strstr(text, "Fin.") == NULL)

{

symbol = getchar();

text[count++] = symbol;

text = realloc(text, sizeof(char) \* (count + 2));

}

text[count] = '\0';

char \*sentence = strtok(text, "\n");

count = 0;

while (sentence != NULL)

{

sentences[count++] = sentence;

sentences = (char \*\*)realloc(sentences, sizeof(char \*) \* (count + 1));

sentence = strtok(NULL, "\n");

}

char \*reg\_sentence = "([a-zA-Z]+:\\/{2})?(w{3}\\.)?([a-zA-Z0-9\\-]+(\\.[a-zA-Z\\-]+)+)(\\/[A-z]+)\*(\\/[A-z0-9]+\\.[A-z0-9]+)";

size\_t maxGroups = 7;

regex\_t regexCompiled;

regmatch\_t groupArray[maxGroups];

if (regcomp(&regexCompiled, reg\_sentence, REG\_EXTENDED))

{

printf("Wowm no - can't compile regular expression\n");

return 0;

};

for (int i = 0; i < count; i++)

{

if (regexec(&regexCompiled, sentences[i], maxGroups, groupArray, 0) == 0)

{

for (int j = groupArray[3].rm\_so; j < groupArray[3].rm\_eo; j++)

printf("%c", sentences[i][j]);

printf(" - ");

for (int j = groupArray[6].rm\_so + 1; j < groupArray[6].rm\_eo; j++)

printf("%c", sentences[i][j]);

printf("\n");

}

}

free(text);

free(sentences);

regfree(&regexCompiled);

return 0;

}

# Приложение Б Тестирование

**Тест № 1.**

**Входные данные:**

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.

**Выходные данные:**

google.com - track.mp3

google.com.edu - hello.avi

qwe.edu.etu.yahooo.org.net.ru - qwe.q

skype.com - qwe.avi

**Тест № 2.**

**Входные данные:**

Hello

google.com/track.mp3

domain http://www.google.com.about.edu/folder/hell3o.mp4

Many of them.

Fin.

**Выходные данные:**

google.com - track.mp3

google.com.about.edu - hell3o.mp4