**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 |  | Лапшов К.Н. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Изучение регулярных выражений и создание программы, которая выполняет поиск в тексте строк, соответствующих определенному образцу, и выводит их части на экран.

## 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

В начале работы программы компилируется регулярное выражение в соответствии с условие работы. В случае ошибки, программа предупреждает об этом и заканчивает свою работу.

Далее создается массив структур для хранения ответов. Структура состоит из переменной для хранения названия сайта и переменной названия файла.

После этого идет считывание текста построчно. Если в текущем предложении есть совпадение с регулярным выражением, то запускается цикл, который проходится по всем заданным группам, и сохраняет ответ в массив ответов. Если предложение состоит из одного слова “Fin”, считывание прекращается

В конце идет вывод ответа и очистка памяти.

Разработанный программный код см. в приложении А.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| № п/п | Входные данные | Выходные 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. | google.com - track.mp3  google.com.edu - hello.avi  qwe.edu.etu.yahooo.org.net.ru - qwe.q  skype.com - qwe.avi |

## Выводы

Было проведено изучение регулярных выражений и освоено их применение. В результате этого изучения была разработана навык по эффективному использованию регулярных выражений в различных сценариях программирования и обработки текста.

# Приложение А Исходный код программы

Название файла: main.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <regex.h>

struct answer{

char nameOfWebsite[100];

char nameOfFile[100];

};

struct answer createAnswer(){

struct answer newAnswer;

newAnswer.nameOfWebsite[0] = '\0';

newAnswer.nameOfFile[0] = '\0';

return newAnswer;

}

void memoryError(){

printf("Memory allocation error!");

exit(0);

}

int main(){

char \* regexString = "(:\\/\\/)?(www\\.)?([a-zA-Z0-9\\.]+)\\/([a-zA-Z0-9\\.]+\\/?)+";

size\_t maxGroups = 5;

regex\_t regexCompiled;

regmatch\_t groupArray[maxGroups];

if(regcomp(&regexCompiled, regexString, REG\_EXTENDED) ){

printf("cant compile regex(");

return 0;

}

struct answer \*answerArray = NULL;

size\_t quantityOfAnswer = 0;

char text[100];

while (1) {

fgets(text, sizeof(text), stdin);

text[strcspn(text, "\n")] = '\0';

if(regexec(&regexCompiled, text, maxGroups, groupArray, 0) == 0){

struct answer newAnswer = createAnswer();

size\_t isDotInName = 0;

size\_t isDotInFile = 0;

for (int i = 0; i <= maxGroups; i++) {

if(groupArray[i].rm\_so == -1){

continue;

}

if(i == 3){

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

if(text[j] == '.'){

isDotInName = 1;

}

strncat(newAnswer.nameOfWebsite, &text[j], 1);

}

}

if(i == 4){

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

if(text[j] == '.'){

isDotInFile = 1;

}

strncat(newAnswer.nameOfFile, &text[j], 1);

}

}

}

if(isDotInName && isDotInFile){

answerArray = realloc(answerArray, sizeof(struct answer) \* (quantityOfAnswer + 1));

if(answerArray == NULL){

memoryError();

}

answerArray[quantityOfAnswer] = newAnswer;

quantityOfAnswer++;

}

}

if (strcmp(text, "Fin.") == 0) {

break;

}

}

for (int i = 0; i < quantityOfAnswer; i++) {

printf("%s - %s\n", answerArray[i].nameOfWebsite, answerArray[i].nameOfFile);

}

free(answerArray);

regfree(&regexCompiled);

return 0;

}