**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 |  | Русанов А.И. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Изучение и применение регулярных выражений в языке программирования 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

В начале работы программы компилируется регулярное выражение в соответствии с условие работы. В случае ошибки, программа выводит строку «Error: can't compile regular expression. Code », после чего выводит код возникшей ошибки.

Затем происходит построчное считывание текста в динамический массив. Считывание происходит до ввода терминального предложения «Fin.».

После этого с помощью цикла for перебираются все предложения в тексте. Если в текущем предложении есть совпадение с регулярным выражением, то запускается цикл, который проходится по всем требуемым группам, и выводит результат на экран в формате <название\_сайта> - <имя\_файла>.

В конце выполняется очистка памяти, выделенной для хранения текста и регулярного выражения.

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

Были изучены и применены на практике регулярные выражения в языке программирования Си. Была разработана программа, которая находит в тексте ссылки на различные файлы в сети интернет.

# Приложение А Исходный код программы

Название файла: main.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <regex.h>

#define END\_OF\_INPUT "Fin.\n"

#define SIZE 250

char \*\*get\_text(int \*number\_of\_sentences)

{

int sentence\_count = 0;

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

char \*sentence = malloc(sizeof(char) \* SIZE);

if (sentence == NULL)

{

printf("Memory allocation error!");

exit(1);

}

while (fgets(sentence, SIZE, stdin))

{

if (strcmp(sentence, END\_OF\_INPUT) == 0)

{

break;

}

sentence\_count++;

text = realloc(text, sizeof(char \*) \* sentence\_count);

if (text == NULL)

{

printf("Memory allocation error!");

exit(1);

}

text[sentence\_count - 1] = strdup(sentence);

free(sentence);

sentence = malloc(sizeof(char) \* SIZE);

if (sentence == NULL)

{

printf("Memory allocation error!");

exit(1);

}

}

free(sentence);

\*number\_of\_sentences = sentence\_count;

return text;

}

void print\_match(char \*sentence, regmatch\_t group\_array[], int index\_of\_group)

{

for (int k = group\_array[index\_of\_group].rm\_so; k < group\_array[index\_of\_group].rm\_eo; k++)

{

printf("%c", sentence[k]);

}

}

void free\_text(char \*\*text, int sentence\_count)

{

for (int i = 0; i < sentence\_count; i++)

{

free(text[i]);

}

free(text);

}

int main()

{

char \*regex\_string = "([a-zA-Z]+://)?(www\\.)?([a-zA-Z0-9-]+(\\.[a-zA-Z0-9]+)+)/((\\w+/)\*)([a-zA-Z0-9-]+(\\.[a-zA-Z0-9-]+))\*\n$";

size\_t max\_groups = 9;

regex\_t regex\_compiled;

regmatch\_t group\_array[max\_groups];

int rc;

if (0 != (rc = regcomp(&regex\_compiled, regex\_string, REG\_EXTENDED)))

{

printf("Error: can't compile regular expression. Code %d\n", rc);

return 0;

};

int sentence\_count = 0;

char \*\*text = get\_text(&sentence\_count);

int flag = 1;

for (int i = 0; i < sentence\_count; i++)

{

if (regexec(&regex\_compiled, text[i], max\_groups, group\_array, 0) == 0)

{

if (flag)

{

flag = 0;

}

else

{

printf("\n");

}

print\_match(text[i], group\_array, 3);

printf(" - ");

print\_match(text[i], group\_array, 7);

}

}

free\_text(text, sentence\_count);

regfree(&regex\_compiled);

return 0;

}