**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 |  | Хайруллов Д.Л. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Целью лабораторной работы является изучение применения регулярных выражений в языке программирования C.

## 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](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

С помощью функции input\_text реализуется считывание текста из стандартного потока ввода и его разделение на предложения с помощью двумерного динамического массива.

Объявляется указатель на строку регулярного выражения, которое затем компилируется. В цикле for перебираются индексы предложений текста, каждое предложение проверяется на наличие совпадений с регулярным выражение с помощью regexec. При нахождении совпадений производится вывод символов, заключенных в определенных группах захвата регулярного выражения.

В конце работы программы с помощью функции clean\_memory и regfree очищается ранее занятая динамическая память.

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 основы работы с регулярными выражениями в языке программирования C. С помощью функций библиотеки regex.h была написана программа, с помощью которого была решена поставленная задача.

# Приложение А Исходный код программы

Название файла: main.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <regex.h>

#define END\_OF\_THE\_TEXT "Fin."

#define MEMORY\_ERROR "Error: error reallocating memory\n"

#define COMPILE\_REGEX\_ERROR "Cant't compile regular expression\n"

#define MEMORY\_ALLOCATING\_ERROR\_PRINT "Error: memory allocating error\n"

#define MEMORY\_ALLOCATING\_ERROR\_EXIT 40

char\*\* text\_input(int\* num\_of\_sentences);

void clean\_memory(char\*\* text, int\* num\_of\_sen);

char\*\* text\_input(int\* num\_of\_sentences){

int current\_symbol\_index = 0;

int current\_sentence\_index = 0;

char symbol;

char\*\* text = (char\*\*)malloc(1\*sizeof(char\*));

if(text == NULL)

{

printf(MEMORY\_ALLOCATING\_ERROR\_PRINT);

exit(MEMORY\_ALLOCATING\_ERROR\_EXIT);

}

text[current\_sentence\_index] = (char\*)malloc(1\*sizeof(char));

if(text[current\_sentence\_index] == NULL)

{

printf(MEMORY\_ALLOCATING\_ERROR\_PRINT);

exit(MEMORY\_ALLOCATING\_ERROR\_EXIT);

}

while(strstr(text[current\_sentence\_index], END\_OF\_THE\_TEXT) == NULL){

symbol = getchar();

if(symbol == '\n'){

text[current\_sentence\_index][current\_symbol\_index] = '\0';

current\_symbol\_index = 0;

current\_sentence\_index++;

text = (char\*\*)realloc(text, (current\_sentence\_index + 1)\*sizeof(char\*));

if(text == NULL)

{

printf(MEMORY\_ALLOCATING\_ERROR\_PRINT);

exit(MEMORY\_ALLOCATING\_ERROR\_EXIT);

}

text[current\_sentence\_index] = (char\*)malloc(1\*sizeof(char));

if(text[current\_sentence\_index] == NULL)

{

printf(MEMORY\_ALLOCATING\_ERROR\_PRINT);

exit(MEMORY\_ALLOCATING\_ERROR\_EXIT);

}

}

else{

text[current\_sentence\_index][current\_symbol\_index] = symbol;

current\_symbol\_index++;

text[current\_sentence\_index] = (char\*)realloc(text[current\_sentence\_index], (current\_symbol\_index + 1)\*sizeof(char));

if(text[current\_sentence\_index] == NULL)

{

printf(MEMORY\_ALLOCATING\_ERROR\_PRINT);

exit(MEMORY\_ALLOCATING\_ERROR\_EXIT);

}

}

}

text[current\_sentence\_index][current\_symbol\_index] = '\0';

\*num\_of\_sentences = current\_sentence\_index;

return text;

}

void clean\_memory(char\*\* text, int\* num\_of\_sen){

for(int i = 0; i <= \*num\_of\_sen; i++){

free(text[i]);

}

free(text);

}

int main(){

char\*\* text;

int num\_of\_sentences = 0;

text = text\_input(&num\_of\_sentences);

char\* regexString = "(:\\/\\/)?(www\\.)?(([A-z0-9]+\\.)+[A-z0-9]+)\\/([A-z0-9]+\\/)\*([A-z0-9]+\\.[A-z0-9]+)\0";

size\_t maxGroups = 7;

regex\_t regexCompiled;

regmatch\_t groupArray[maxGroups];

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

printf(COMPILE\_REGEX\_ERROR);

return 0;

}

for(int i = 0; i <= num\_of\_sentences; i++){

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

if(groupArray[3].rm\_so == -1 || groupArray[6].rm\_so == -1){

break;

}

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

printf("%c", text[i][y]);

}

printf(" - ");

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

printf("%c", text[i][y]);

}

printf("%c", '\n');

}

}

clean\_memory(text, &num\_of\_sentences);

regfree(&regexCompiled);

return 0;

}