**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 |  | Рябов М.Л. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Цель работы является изучение и использование регулярных выражений для обработки текстовых данных. Для этого необходимо изучить синтаксис и возможности регулярных выражений, а после применить полученные навыки на практике в ходе решения задачи.

## 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, 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

Подключаются необходимые библиотеки: *stdlib.h*, *stdio.h*, *string.h* и *regex.h*.

В переменную *regexString* записывается необходимое регулярное выражение.

Функция *getText* считывает текст из ввода пользователя, вызывает функцию *sepText* для разбиения цельного текста на строки и возвращает их в виде массива строк.

Функция *sepText* разделяет текст на отдельные предложения и возвращает их в виде массива строк.

Функция *checkMathReg* принимает массив предложений, применяет регулярное выражение *regexString* к каждому предложению и выводит предложения, которые соответствуют этому регулярному выражению.

В основной функции *main* происходит вызов этих функций: считывание текста, разделение на предложения и проверка совпадения с регулярным выражением.

Разработанный программный код см. в приложении А.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | This is simple url:  http://www.google.com/track.mp3  Fin. | google.com – track.mp3 | Проверка на наличие www перед доменным именем |
|  | 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 | Проверка на валидность выражений с доменами более высокого уровня и на наличие пути до файла |
|  | ftp://pepepupu.cheeck/qqwe/qweqw/qwe.avi  Fin. | pepepupu.cheeck – qwe.avi | Проверка исправности с протоколом ftp и на наличие пути до файла |

## Выводы

Цель данной работы заключалась в изучении и практическом применении регулярных выражений для обработки текстовых данных. Были изучены основные синтаксические конструкции и возможности регулярных выражений. Полученные знания были успешно применены для решения практической задачи, демонстрирующей использование регулярных выражений в реальной ситуации. Таким образом, цель данной работы была успешно достигнута.

# Приложение А Исходный код программы

Название файла: main.c

#include <stdio.h>

#include <stdlib.h>

#include <regex.h>

#include <string.h>

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

char\*\* getText(int\* countSen);

char\*\* sepText(char\* text, int countSen);

void checkMathReg(char\*\* sentences, int countSen);

int main(){

int countSen;

char\*\* sentences = getText(&countSen);

checkMathReg(sentences, countSen);

return 0;

}

char\*\* getText(int\* countSen){

int size = 0, capacity = 1;

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

char letter = getchar();

(\*countSen) = 1;

while (1){

text[size++] = letter;

if (letter == '\n')

(\*countSen) ++;

if (size >= 4 && text[size-1] == '.' && text[size-2] == 'n' && text[size-3] == 'i' && text[size-4] == 'F')

break;

if (size >= capacity){

capacity \*= 2;

text = (char\*)realloc(text, sizeof(char) \* capacity);

}

letter = getchar();

}

text[size] = '\0';

char\*\* sentences = sepText(text, \*countSen);

return sentences;

}

char\*\* sepText(char\* text, int countSen){

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

int size = 0;

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

while(line != NULL){

sentences[size++] = line;

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

}

return sentences;

}

void checkMathReg(char\*\* sentences, int countSen){

regex\_t regexCompiled;

regmatch\_t groups[8];

int size, sizeMatched = 0;

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

regcomp(&regexCompiled, regexString, REG\_EXTENDED);

for(int j = 0; j < countSen; j++){

if (regexec(&regexCompiled, sentences[j], 8, groups, 0) == 0) {

matchedSentences = realloc(matchedSentences, sizeof(char\*) \* (sizeMatched+1));

char\* regLine = (char\*)malloc(sizeof(char)\*100);

size = 0;

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

regLine[size++] = sentences[j][i];

regLine[size++] = ' ';

regLine[size++] = '-';

regLine[size++] = ' ';

for (int i=groups[7].rm\_so; i<groups[7].rm\_eo; i++)

regLine[size++] = sentences[j][i];

regLine[size] = '\0';

matchedSentences[sizeMatched++] = regLine;

}

}

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

if (i == sizeMatched - 1)

printf("%s", matchedSentences[i]);

else

printf("%s\n", matchedSentences[i]);

}

}