**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 |  | Шаповаленко Е.В |
| 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 *pattern* записывается необходимое регулярное выражение.

В функции *main* регулярное выражение компилируется. После этого функцией *input* построчно считывается текст. Каждая строка проходит проверку функцией *checkString*, выводятся найденные ссылки функцией *printGroup*. Если введенная строка является конечной ("Fin."), программа завершает работу.

Функция *input* посимвольно считывает строку текста в переменную *string*. При выходе за границы памяти она перевыделяется. Функцией *isFin* проверяется, является ли строка конечной ("Fin.") путем их сравнения.

В функции *checkString* производится проверка строки при помощи регулярного выражения. Искомые адрес сайта и имя файла выводятся на экран функцией *printGroup*.

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 |
|  | https://ru.wikipedia.org/wiki/cooltext.txt  Fin. | ru.wikipedia.org – cooltext.txt | Проверка на наличие доменов более высокого уровня и на наличие пути до файла на сервере |
|  | ftp://skype.com/qqwe/qweqw/qwe.avi  Fin. | skype.com – qwe.avi | Проверка исправности с другим протоколом и на наличие пути до файла на сервере |

## 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 <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <regex.h>

const char\* pattern = "([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]+)";

int isFin(char\* string) {

return (strcmp(string, "Fin.") == 0);

}

void input(char\*\* string) {

int used\_memory = 0, allocated\_memory = 1;

char ch = ' ';

while (ch != '\n') {

ch = getchar();

if(used\_memory >= allocated\_memory) {

allocated\_memory \*= 2;

(\*string) = (char\*)realloc(\*string, allocated\_memory \* sizeof(char));

}

(\*string)[used\_memory] = ch;

used\_memory++;

if(isFin(\*string)) break;

}

}

void printGroup(char\* string, regmatch\_t group) {

for (int i = group.rm\_so; i < group.rm\_eo; i++) {

printf("%c", string[i]);

}

}

void checkString(char\* string, regex\_t regexCompiled) {

regmatch\_t groups[8];

if (regexec(&regexCompiled, string, 8, groups, 0) == 0) {

printGroup(string, groups[3]);

printf(" - ");

printGroup(string, groups[7]);

printf("\n");

}

}

int main() {

regex\_t regexCompiled;

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

while(1) {

char\* string = (char\*)calloc(1, sizeof(char));

input(&string);

checkString(string, regexCompiled);

if(isFin(string)) {

break;

}

}

return 0;

}