**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. 8381 |  | Киреев К.А. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2019

**Цель работы**

Научиться использовать регулярные выражения. Изучить библиотеку <regex.h> в языке СИ. Научиться использовать функции библиотеки, такие как regcomp, regexec, regerror, regfree.

**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**

Сначала в переменную regexString записывается регулярное выражение для поиска пар <название\_сайта> - <имя\_файла> в стандарте POSIX. Далее создается массив groupArray из maxGroups элементов, который будет заполнен смещениями подцепочек. Функция regcomp предназначена для компиляции регулярного выражения. Эта функция компилирует регулярное выражение regexString, учитывая флаг REG\_EXTENDED [использование расширенных регулярных выражений], и помещает его в переменную regexCompiled. Далее в цикле while (до терминального предложения) функция regexec сопоставляет регулярное выражение, скомпилированное и помещённое в переменную regexCompiled со строкой s. При этом в случае успешного сопоставления возвращается нулевое значение, в противном случае - код ошибки (пятый аргумент функции (в данной программе - ноль) - побитовое ИЛИ REG\_NOTBOL и REG\_NOTEOL он определяет, являются ли границы цепочки границами строки (для обработки фиксаторов ^ и $)). Далее, при успешном сопоставлении, группы 3 и 6, в которых находятся название сайта и имя файла, посимвольно выводятся через дефис. Функция regfree освобождает память, запрошенную при компилировании регулярного выражения.

**Выводы**

В ходе лабораторной работы были изучены правила составления регулярных выражений. Изучена библиотека <regex.h>. Получен опыт в применении регулярных выражении в языке СИ. Написана программа для поиска пар <название\_сайта> - <имя\_файла> с помощью регулярного выражения.

**Приложение А. Исходный код программы.**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <regex.h>

int main(){

char\* regexString = "([[:alpha:]]+:\\/\\/)?(www\\.)?([[:alnum:]]+(\\.[[:alnum:]]+)+)(\\/[[:alnum:]]+)\*\\/([[:alnum:]]+\\.[[:alnum:]]+)";

size\_t maxGroups = 7;

regex\_t regexCompiled;

regmatch\_t groupArray[maxGroups];

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

char s[300];

while (strcmp(fgets(s, 300, stdin), "Fin.")){

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

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

if (i == 3 | i == 6){

//printf("group %d: [%d-%d]: ", i + 1, groupArray[i].rm\_so, groupArray[i].rm\_eo);

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

printf("%c", s[j]);

if (i == 3)

printf(" - ");

else

puts("");

}

}

}

}

regfree(&regexCompiled);

return 0;

}