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

Студентка гр. 3344 _____ Коняева М.В.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Целью работы является освоение работы с регулярными выражениями в языке Си на примере использующей их программы.

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

Подключим стандартные библиотеки *stdio.h* для работы с вводом, *string.h* для работы со строками, а также *regex.h* для работы с регулярными выражениями.

В главной функции задается переменная *char *regexString* – строка, являющаяся регулярным выражением, а также переменная *size_t maxGroups*, равная 9 и отвечающая за максимальное количество групп в шаблоне. Объявлена структура для хранения информации о скомпилированном регулярном выражении *regex_t regexCompiled*. Был объявлен массив *regmatch_t groupArray[maxGroups]* размером *maxGroups*, используемый для хранения информации о совпадениях групп в регулярном выражении.

Далее делаем проверку на компилируемость регулярного выражения с помощью функции *regcomp*, которая принимает на вход структуру *regexCompiled*, регулярное выражение *regexString*, и флаг *REG_EXTENDED*. Если выражение невозможно скомпилировать, то программа завершает работу.

Объявлен массив символов *s* для записи вводимого текста. Считываем текст с помощью *fgets*, пока не встретится строка *«Fin.»*. Далее в функции *if* вызываем функцию *regexec*, которая сопоставляет регулярное выражение скомпилированное и помещённое в структуру *regexCompiled* со строкой *regexString*. Если соответствие найдено, функция возвращает 0, иначе - ненулевое значение. Для вывода были запущены циклы, которые посимвольно выводят строки, нужной группы.

После цикла освобождаем память, запрошенную при компилировании регулярного выражения и завершаем программу.

Разработанный программный код см. в приложении А. Результаты тестирования см. в приложении Б.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	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 Fin.	google.com - track.mp3 google.com.edu - hello.aviists	Данные обработаны корректно
2.	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.	qwe.edu.etu.yahooo.org.net.ru - qwe.q skype.com - qwe.avi	Данные обработаны корректно

Выводы

Были изучена работа с регулярными выражениями. А также была создана программа, в которой реализовано считывание и вывод строк, подходящих под заданный шаблон.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb1.c

```
#include <stdio.h>
#include <regex.h>
#include <string.h>


int main ()
{
    char *regexString = "((http|https|ftp):\\/\\/)?(www\\.)?(([a-zа-я0-
9-]+\\.)+[a-zа-я0-9]+)(\\/[a-zа-я0-9-]+)*(\\/([a-zа-я0-9-]+\\.[a-zа-
я0-9]+))";
    size_t maxGroups = 9;

    regex_t regexCompiled;
    regmatch_t groupArray[maxGroups];

     if (regcomp(&regexCompiled, regexString, REG_EXTENDED))
    {
        printf("Can't compile regular expression\n");
        return 0;
    };

    char s[100] = "";
    while (strcmp(s, "Fin."))
    {
        fgets(s, 100, stdin);
        if (regexec(&regexCompiled, s, 9, groupArray, 0) == 0)
        {
            for (int j = groupArray[4].rm_so; j < groupArray[4].rm_eo; j++)
                printf("%c", s[j]);
            printf(" - ");
            for (int j = groupArray[8].rm_so; j < groupArray[8].rm_eo; j++)
                printf("%c", s[j]);
            printf("\n");
        }
    }
    regfree(&regexCompiled);
    return 0;
}
```