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 _____ Гусева Е.А.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Целью работы является освоение работы с регулярными выражениями в языке Си.

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

Были подключены библиотеки *stdio.h*, *string* и *regex.h*

В функции *main* задается переменная *char *reRegex* – регулярное выражение, а также переменная *size_t maxGroups*, равная 9 - максимальное количество групп в шаблоне. Объявлена *regex_t regexCompiled* – хранящая хранения информацию о скомпилированном регулярном выражении. Объявлен массив *regmatch_t groupArray[maxGroups]* размером *maxGroups*, хранящий информацию о совпадениях групп.

Далее делаем проверку на компилируемость регулярного выражения с помощью функции *regcomp*. Если проверка говорит о невозможности скомпилировать регулярное выражение, то программа завершает работу.

Объявлен массив символов *smtext* для записи вводимого текста, считывается текст с помощью *fgets*, пока не встретится строка *«Fin.»*. Далее вызывается функция *regexec*, сопоставляющую регулярное выражение скомпилированное и помещённое в структуру *regexCompiled* со строкой *reRegex*. Если соответствие найдено, функция возвращает 0, иначе - ненулевое значение. Далее посимвольно выводятся строки, нужной группы. После цикла освобождается памятьи после этого завершается программа.

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	

Выводы

Были изучена работа с регулярными выражениями. Выполнена лабораторная работа.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main_for_lb1.c

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


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

    regex_t regexCompiled;
    regmatch_t groupArray[maxGroups];

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

    char smtext[1000] = "";
    while (strcmp(smtext, "Fin."))
    {
       fgets(smtext, 1000, stdin);
       if (regexec(&regexCompiled, smtext, maxGroups, groupArray, 0) == 0)
       {
          for (int j = groupArray[4].rm_so; j < groupArray[4].rm_eo; j++)
             printf("%c", smtext[j]);
```

```
            printf(" - ");
            for (int j = groupArray[8].rm_so; j < groupArray[8].rm_eo; j++)
                printf("%c", smtext[j]);
            printf("\n");
        }
        if (strstr(smtext, "Fin."))
            break;
    }


    regfree(&regexCompiled);
    return 0;
}
```