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: Лабораторная работа № 1. Регулярные выражения

Студент гр. 3343 _____ Пименов П.В.

Преподаватель _____ Государкин Я.С.

Санкт-Петербург

2024

Цель работы

Изучить способы работы с регулярными выражениями, создать программу на языке C, которая с помощью регулярных выражений извлекает из ссылок на различные файлы в сети Интернет название сайта и имя файла.

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

Описание переменных:

1. *size_t size* – размер текста
2. *size_t capacity* – текущий максимально допустимый размер текста
3. *char *text* – текст
4. *int last_char* – последний считанный символ
5. *size_t maxGroups* – максимальное количество групп захвата
6. *regex_t regexCompiled* – скомпилированное на математический язык регулярное выражение

7. *regmatch_t groupArray[maxGroups]* – массив значений выделенных групп захвата
8. *int result* – характеристика успешности выполнения операции компиляции и применения регулярного выражения
9. *char *line* –* выделенная из текста строка до символа перевода строки

Программа считывает текст до символа конца строки, либо подстроки "Fin.", компилирует регулярное выражение на математический язык, разделяет текст на строки по символам перевода строки, применяет к ней регулярное выражение, и в случае, если совпадение обнаружилось, выводит в поток вывода пары вида <название_сайта> – <имя_файла>.

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
1.	http://www.google.com/ track.mp3 Fin.	google.com - track.mp3	Программа работает корректно.
2.	https://vk.com/feed/test.sf Fin.	vk.com - test.sf	Программа работает корректно.
3.	http:// www.qwe.edu.etu.yahooo.org. net.ru/qwe.q Fin.	qwe.edu.etu.yahooo.org. net.ru - qwe.q	Программа работает корректно.
4.	http://www.google.com.edu/ hello.avi Fin.	google.com.edu - hello.avi	Программа работает корректно.

Выводы

Были изучены способы работы с регулярными выражениями, создана программа на языке C, которая с помощью регулярных выражений извлекает из ссылок на различные файлы в сети Интернет название сайта и имя файла.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

#define MEMORY_BLOCK_SIZE 10
#define TEXT_END "Fin."
#define REGEX_STRING "([a-zA-Z]+:\\/\\/)?(www\\.)?(([a-zA-Z0-9.\\-])
+\\.([a-zA-Z0-9])+)(\\/[a-zA-Z0-9_\\-]+)*\\/([a-zA-Z0-9_\\-]+\\.[a-zA-
Z0-9]+)"

int main()
{
    size_t size = 0;
    size_t capacity = MEMORY_BLOCK_SIZE;
    char *text = (char *)malloc(capacity * sizeof(char));
    int last_char = getchar();
    if (last_char == '\0' || last_char == EOF)
    {
        return 0;
    }
    text[size++] = last_char;
    text[size] = '\0';
    last_char = getchar();
    while (last_char != '\0' && last_char != EOF)
    {
        text[size++] = last_char;
        text[size] = '\0';
        if (strstr(text, TEXT_END) != NULL)
        {
            break;
        }
        if (size == capacity - 1)
        {
            capacity += MEMORY_BLOCK_SIZE;
            text = (char *)realloc(text, capacity);
        }
        last_char = getchar();
    }

    size_t maxGroups = 8;

    regex_t regexCompiled;
    regmatch_t groupArray[maxGroups];

    int result = regcomp(&regexCompiled, REGEX_STRING, REG_EXTENDED);
    if (result)
    {
```

```
        free(text);
        regfree(&regexCompiled);
        return 0;
    };
    char *line;
    line = strtok(text, "\n");
    while (line != NULL)
    {
        result = regexec(&regexCompiled, line, maxGroups, groupArray,
0);

        if (!result)
        {
            for (int j = groupArray[3].rm_so; j < groupArray[3].rm_eo;
j++)
            {
                printf("%c", line[j]);
            }
            printf(" - ");
            for (int k = groupArray[7].rm_so; k < groupArray[7].rm_eo;
k++)
            {
                printf("%c", line[k]);
            }
            printf("\n");
        }
        line = strtok(NULL, "\n");
    }

    free(text);
    regfree(&regexCompiled);

    return 0;
}
```