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. 3342 _____ Русанов А.И.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Изучение и применение регулярных выражений в языке программирования Си.

Задание

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

В начале работы программы компилируется регулярное выражение в соответствии с условие работы. В случае ошибки, программа выводит строку «Error: can't compile regular expression. Code », после чего выводит код возникшей ошибки.

Затем происходит построчное считывание текста в динамический массив. Считывание происходит до ввода терминального предложения «Fin.».

После этого с помощью цикла for перебираются все предложения в тексте. Если в текущем предложении есть совпадение с регулярным выражением, то запускается цикл, который проходится по всем требуемым группам, и выводит результат на экран в формате <название_сайта> - <имя_файла>.

В конце выполняется очистка памяти, выделенной для хранения текста и регулярного выражения.

Разработанный программный код см. в приложении А.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
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 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.	google.com - track.mp3 google.com.edu - hello.avi qwe.edu.etu.yahooo.org.net.ru - qwe.q skype.com - qwe.avi

Выводы

Были изучены и применены на практике регулярные выражения в языке программирования Си. Была разработана программа, которая находит в тексте ссылки на различные файлы в сети интернет.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <regex.h>
#define END_OF_INPUT "Fin.\n"
#define SIZE 250

char **get_text(int *number_of_sentences)
{
    int sentence_count = 0;
    char **text = malloc(sizeof(char *));
    char *sentence = malloc(sizeof(char) * SIZE);
    if (sentence == NULL)
    {
        printf("Memory allocation error!");
        exit(1);
    }
    while (fgets(sentence, SIZE, stdin))
    {
        if (strcmp(sentence, END_OF_INPUT) == 0)
        {
            break;
        }
        sentence_count++;
        text = realloc(text, sizeof(char *) * sentence_count);
        if (text == NULL)
        {
            printf("Memory allocation error!");
            exit(1);
        }
        text[sentence_count - 1] = strdup(sentence);
        free(sentence);
        sentence = malloc(sizeof(char) * SIZE);
        if (sentence == NULL)
        {
            printf("Memory allocation error!");
            exit(1);
        }
    }
    free(sentence);
    *number_of_sentences = sentence_count;
    return text;
}

    void print_match(char *sentence, regmatch_t group_array[], int
index_of_group)
    {
        for (int k = group_array[index_of_group].rm_so; k <
group_array[index_of_group].rm_eo; k++)
        {
```

```
            printf("%c", sentence[k]);
        }
    }

    void free_text(char **text, int sentence_count)
    {
        for (int i = 0; i < sentence_count; i++)
        {
            free(text[i]);
        }
        free(text);
    }

    int main()
    {
        char    *regex_string    =    "([a-zA-Z]+://)?(www\\.)?([a-zA-Z0-
9-]+(\\.[a-zA-Z0-9]+)+)/((\\w+/)*)([a-zA-Z0-9-]+(\\.[a-zA-Z0-9-]+))*\n$";
        size_t max_groups = 9;
        regex_t regex_compiled;
        regmatch_t group_array[max_groups];
        int rc;

        if   (0   !=   (rc   =   regcomp(&regex_compiled,   regex_string,
REG_EXTENDED)))
        {
            printf("Error: can't compile regular expression. Code %d\n",
rc);
            return 0;
        };

        int sentence_count = 0;
        char **text = get_text(&sentence_count);
        int flag = 1;

        for (int i = 0; i < sentence_count; i++)
        {
            if     (regexec(&regex_compiled,     text[i],     max_groups,
group_array, 0) == 0)
            {
                if (flag)
                {
                    flag = 0;
                }
                else
                {
                    printf("\n");
                }
                print_match(text[i], group_array, 3);
                printf(" - ");
                print_match(text[i], group_array, 7);
            }
        }
        free_text(text, sentence_count);
        regfree(&regex_compiled);
        return 0;
    }
```