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. 3383 _____ Логинова А.Ю.

Преподаватель _____ Гаврилов А.В.

Санкт-Петербург

2024

Цель работы.

Ознакомиться с регулярными выражениями, использовать полученные знания в написании программы, реализующую обработку текста на выявление в ней корректно написанной ссылки и захват названия сайта и файла.

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 программе были реализованы функции, считывающие текст из потока ввода, обрабатывающие текст с помощью регулярного выражения, а также структуры для хранения данных о строке, полученной на входе, и о корректно написанной ссылке.

Структура String состоит из трех полей: capacity, length, chars. В них хранятся данные о размере массива, длине строки и указатель на массив символов соответственно. Структура Link состоит из двух полей: URL, file. В них хранятся указатели на данные, полученные после обработки ссылки: название адреса сайта и файла.

Программа начинает работу с компиляции регулярного выражения: если оно не скомпилировалось, программа завершается с ошибкой. Далее создается и инициализируется объект типа String с помощью функции create(). Для поля chars выделяется блок памяти на куче с помощью функции calloc(), размер массива при создании равен единице и длина строки равна нулю.

Считывание текста реализовано в функции read_input(String *s), в которую передается указатель на ранее созданный объект типа String. Считывается текст посимвольно с помощью функции read_next_char(String *s), которая проверяет размер массива, при необходимости выделяет для него память с помощью функции resize (String* s) и записывает символ в массив. В конец строки записывается нуль-терминатор.

Считанная строка и регулярное выражение передаются в функцию parse(String *s, regex_t *regex) для проверки на наличие в строке корректно написанной ссылки. В ней создается и возвращается объект типа Link, в которой будут храниться указатели на адрес сайта и название файла, полученные с помощью регулярного выражения. Данные поля инициализируются в функции set_link(regmatch_t *groups, char *s, Link *link), которая записывает начало подстроки с адресом или файлом, и нуль-терминатор в конец подстроки по индексам, полученным с помощью функции regexec().

В функции parse() адрес и название файла выводятся сразу — это обусловлено тем, что в одной строке может быть несколько ссылок. После обработки строки вызывается функция clear_string(String *s), которая обнуляет длину строки, чтобы записывать в нее новую строку.

В конце программы освобождается память, выделенная для строки и регулярного выражения, с помощью функций free() и regfree().

Результаты тестирования программы представлены в таблице 1.

Тестирование.

Таблица 1 – Результаты тестирования

Входные данные	Выходные данные	Комментарии
http:// www.qwe.edu.etu.yahooo.org.net.ru/qwe.q ftp://secondlink.com/qqwe/qweqw/qw11.avi 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! Some other protocols ftp://skype.com/qqwe/qweqw/qwe.avi http://skype.com./qawe/qqw/qwde.avja Fin.	qwe.edu.etu.yahooo.org.net.ru - qwe.q secondlink.com - qw11.avi google.com - track.mp3 google.com.edu - hello.avi skype.com - qwe.avi	Программа работает корректно.

Выводы.

Были изучены регулярные выражения. Разработана программа, выполняющая считывание с клавиатуры исходных данных и обрабатывающая их с помощью регулярного выражения. Задача была решена с помощью функций regcomp(), regexec(), структур, циклов while и for. Было обработано некорректное поведение программы: если не выделилась память с помощью realloc() или не скомпилировалось регулярное выражение, программа выводит сообщение об ошибке.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <regex.h>
#define GROUPS_AMOUNT 7
#define REGEX "(\\w+://)?(www.)?(([A-Za-z0-9]+\\.)+[A-Za-z0-9]+)/([A-Za-z0-9]+/)*([A-Za-z0-9]+\\.[A-Za-z0-9]+)"
#define URL_GROUP 3
#define FILE_GROUP 6


typedef struct String {
    size_t capacity, length;
    char* chars;
} String;

typedef struct Link {
    char *URL, *file;
} Link;

void resize (String* s) {
    if (s->length + 1 == s->capacity) {
        s->capacity *= 2;
        char *temp =  realloc(s->chars, s->capacity);
        if(!temp) {
            free(s->chars);
            printf("Memory allocation error");
            exit(1);
        }
        s->chars = temp;
    }
}

char read_next_char(String *s) {
    char c = (char)getchar();
    resize(s);
    s->chars[s->length++] = c;
    s->chars[s->length] = '\0';
    return c;
}

String create() {
    String s = { 1, 0,calloc(s.capacity, sizeof(char)) };
    return s;
}

void clear_string(String *s) {
    s->length = 0;
}
```

```
     void read_input(String *s) {
          while(read_next_char(s) != '\n' && strcmp(s->chars, "Fin.") !=
0);
     }

     void set_link(regmatch_t *groups, char *s, Link *link) {
         s[groups[URL_GROUP].rm_eo] = '\0';
         s[groups[FILE_GROUP].rm_eo] = '\0';
         link->URL = &s[groups[URL_GROUP].rm_so];
         link->file = &s[groups[FILE_GROUP].rm_so];
     }

     void print(Link *link) {
         static int need_lf = 0;
         if(link->URL && link->file) {
             if (need_lf)
                 printf("\n");
             else
                 ++need_lf;
             printf("%s - %s", link->URL, link->file);
         }
     }

     Link parse(String *s, regex_t *regex) {
         Link link;
         regmatch_t groups[GROUPS_AMOUNT];
         int res;

          for (char *sub_s = strtok(s->chars, " "); sub_s != NULL; sub_s
= strtok(NULL, " ")) {
             res = regexec(regex, sub_s, GROUPS_AMOUNT, groups, 0);
                    if(res  ==  0  &&  groups[URL_GROUP].rm_so  !=  -1  &&
groups[FILE_GROUP].rm_so != -1) {
                 set_link(groups, sub_s, &link);
                 print(&link);
             } else {
                 link.URL = NULL; link.file = NULL;
             }
         }
         return link;
     }

     int main() {
         regex_t regex;
         if (regcomp(&regex, REGEX, REG_EXTENDED) == 0) {
             String s = create();
                    for(read_input(&s);  strcmp(s.chars,  "Fin.")  !=  0;
read_input(&s)) {
                 parse(&s, &regex);
                 clear_string(&s);
             }
             free(s.chars);
         }
         else
             printf("Regex compilation error");
         regfree(&regex);
         return 0; }.
```