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. 3342 _____ Антипина В.А.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Изучение регулярных выражений и реализация программы, которая осуществляет поиск строк текста, удовлетворяющих заданному виду и выводит их фрагменты на экран.

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.

Основные теоретические положения.

Регулярные выражения (их еще называют regexp, или regex) — это механизм для поиска и замены текста.

С помощью regex можно искать как конкретные выражения, так и что-то более общее (например, любую букву или цифру).

Для обозначения второй категории существуют специальные символы. Вот некоторые из них:

«.» - любой символ;

«[...]» - любой символ из тех, что представлены в скобках;

«[^...]» - любой символ, кроме тех, что представлены в скобках;

«^» - начало строки;

«$» - конец строки;

«\» - экранирование специальных символов;

«|» - логическое «ИЛИ».

Также существуют специальные символы, наиболее часто из которых встречаются +, * и ?. + используют, чтобы обозначить, что группа или символ используются 1 и более раз, * - 0 и более, а ? сигнализирует о том, что символ или группа или отсутствуют, или встречаются не более 1 раза в этом месте.

\w\d\s используют, чтобы обозначить буквы, цифры или пробельные символы.

Когда в квадратных скобках указывается диапазон, подразумевается наличие одного из этих символов (или 0, или большего числа в зависимости от знака после, если он есть), причём в регулярных выражениях используется расположение символов в таблице ASCII. Таким образом, если написать, например, [A-z], будет считаться, что символ ^ также может входить в строку и быть на данном месте.

В Си для работы с регулярными выражениями подключается библиотека regex.h.

Выполнение работы

Были подключены библиотеки stdio.h, stdlib.h для работы с динамической памятью, string.h и regex.h. Было реализовано считывание текста неизвестной длины. Две функции input и split_text, которые возвращают указатели на char и char*.

В первой динамически выделяется память на массив символов text типа char*. В переменную ch записывается символ из потока ввода, переменные ch_one, ch_two, ch_three, ch_four хранят последние четыре считанных символа. Это было сделано для того, чтобы прекратить считывание текста после ввода «Fin.». Если хотя бы одна из переменных содержит «неправильный» символ, считывание продолжается (это реализовано с помощью цикла while). При записи символа в массив увеличивается счётчик k, и если он становится равным capacity-1, изменяется значение последней на константу BLOCK_SIZE, а память перевыделяется с помощью realloc. После считывания в конец текста записывается символ окончания строки.

Функция split_text получает на вход указатель на массив символов text, а также указатель на целое (там будет записано количество предложений). Функцией malloc выделяется память на двумерный массив символов text_done, а в массив символов sentence записывается результат вызова функции strtok, на вход которой были поданы текст и символ переноса строки. С помощью strcpy копируется содержимое sentence в text_done[(*k)]. После этого счётчик k увеличивается и, аналогично предыдущей функции, при необходимости выделяется новый участок памяти. Затем снова вызывается strtok, но на этот раз ей на вход подаётся NULL, а не text.

В функции main вызываются описанные ранее функции и результат их вызова сохраняется в переменных text и text_done соответственно. В переменную типа char* записывается регулярное выражение, соответствующее условию задачи. В переменной maxGroups записано максимальное число групп в выражении. Создаются переменные

regexCompiled (где будет храниться скомпилированное регулярное выражение) типа regex_t (это псевдоним для структуры) и groupArray типа regmatch_t.

Если компиляция не прошла успешно, выводится ошибка.

В цикле for проверяется каждое предложение считанного текста. Если для этого предложения результат вызова функции regexec равен нулю, то выводятся все символы групп, в которых содержится имя домена и название файла. Между этими выводами на экран выводится тире, после — символ переноса строки.

Далее с помощью функций free, regfree осуществляется освобождение памяти.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	Hello! That is my program Try this simple URL: http://www.google.com/kjwfwk/kitten.jpg And what about this one? ftp://jshkjshk.net.ru.edu/kjwhdjkh/kjhkhl/jh/ha-ha.zip Here is the one you should NOT choose: hgjhghjp:/i-am-the-wrong-choice.boo/jjjjjjj/whats'up.doc Fin.	google.com - kitten.jpg jshkjshk.net.ru.edu - ha-ha.zip	Всё правильно!
2.	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/qweq	google.com - track.mp3 google.com.edu - hello.avi qwe.edu.etu.yahooo.org.net.ru - qwe.q skype.com - qwe.avi	

	w/qwe.avi Fin.		

Выводы

Были изучены регулярные выражения, их использование.

Разработана программа, выполняющая считывание с клавиатуры текста. Используя регулярные выражения, программа находит URL файла и выводит в консоль название домена и имя файла. Для этого использовалась библиотека regex.h и циклы for, while, динамическое выделение памяти и её освобождение.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Antipina_Veronika_lb1.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <regex.h>
#define BLOCK_SIZE 10
#define END_OF_STRING '\0'

char* input(){
        int capacity = BLOCK_SIZE;
        char* text = malloc(capacity*sizeof(char));
        if(text==NULL){
            printf("ERROR: couldn't allocate memory\n");
            return 0;
        }
        int k = 0;

        char ch = getchar();
        char ch_one = 'a';
        char ch_two = 'a';
        char ch_three = 'a';
        char ch_four = 'a';

      while(ch_one!='F'||ch_two!='i'||ch_three!='n'||ch_four!='.')
{
           text[(k)++] = ch;
            if((k) == (capacity)-1){
                 (capacity)+=BLOCK_SIZE;
                 text                                              =
(char*)realloc(text,(capacity)*sizeof(char));
                 if(text==NULL){
                      printf("ERROR: could not find the memory! And
it's just the beginning...\n");
                      return 0;
                 }
            }
            ch_one = ch_two;
            ch_two = ch_three;
            ch_three = ch_four;
            ch_four = ch;
            ch = getchar();
        }
        text[(k)] = END_OF_STRING;
        return text;
}

char** split_text(char* text, int* k){
       int capacity = BLOCK_SIZE;
       char** text_done = (char**)malloc(capacity*sizeof(char*));
       if(text_done==NULL){
```

```
                printf("ERROR:  No  memory  left!  Buy  me  Ginkoum,
please!!");
                  return 0;
              }
              char* sentence = strtok(text,"\n");

              while(sentence!=NULL){
                 int sent_len = strlen(sentence);
                 text_done[(*k)] = malloc((sent_len+1)*sizeof(char));
                 if(text_done[(*k)]==NULL){
                      printf("ERROR:  Sorry,  I  don't  have  enough
memory:(\n");
                      return 0;
                 }

                 strcpy(text_done[(*k)], sentence);

                 (*k)++;
                 if((*k) == capacity-1){
                      capacity+=BLOCK_SIZE;
                      text_done                                      =
(char**)realloc(text_done,capacity*sizeof(char*));
                      if(text_done==NULL){
                           printf("ERROR:  Oh  no!  Could  not  find  the
memory!\n");
                           return 0;
                      }
                 }
                 sentence = strtok(NULL,"\n");
              }
              return text_done;
      }

      int main(){
              char* text = input();
              int k = 0;

              char** text_done = split_text(text,&k);

              char*  regexString  =  "(([A-z]*):\\/\\/)?(www.)?([A-z0-
9]+([_\\-\\.]+[A-z]+)+)\\/((([A-z]*)\\/)*)([A-z0-9_\\-]+\\.[A-z0-
9_\\-]+)";


              regex_t regexCompiled;
              regmatch_t groupArray[BLOCK_SIZE];

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

              for(int index = 0; index < k; index++){

if(regexec(&regexCompiled,text_done[index],BLOCK_SIZE,groupArray,0)==0
){
```

```
                                int i = 4;

                                for(int                 j                 =
groupArray[i].rm_so;j<groupArray[i].rm_eo;j++)
                                        printf("%c",text_done[index][j]);
                                printf(" - ");

                                i = 9;
                                for(int                 j                 =
groupArray[i].rm_so;j<groupArray[i].rm_eo;j++)
                                        printf("%c",text_done[index][j]);
                                printf("\n");
                        }
                }

                free(text);
                for(int i = 0;i<k;i++)
                        free(text_done[i]);
                free(text_done);

                regfree(&regexCompiled);

                return 0;
        }
```