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. 3341 _____ Лодыгин И.А.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Целью данной работы является изучение и освоение использования регулярных выражений на языке программирования C. Для этого разрабатывается программа, которая принимает на вход текст, состоящий из предложений с возможными ссылками на файлы в сети интернет.

Программа должна применять регулярные выражения для обнаружения всех таких ссылок в тексте. Кроме того, необходимо разработать механизм вывода на экран пар, где каждая пара содержит название сайта и имя файла, соответствующие найденной ссылке. Важным аспектом данной работы является развитие навыков программирования на языке C и исследование возможностей работы с регулярными выражениями.

Задание

1 вариант.

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 (Regular Expressions) – это мощный инструмент для работы с текстовыми данными, который позволяет задать шаблон поиска текста. Эти шаблоны могут содержать символы и специальные конструкции, которые определяют правила поиска соответствий. В языке программирования C для работы с регулярными выражениями используется библиотека regex.h.

Библиотека regex.h содержит функции для работы с регулярными выражениями, такие как компиляция, сопоставление и освобождение регулярного выражения. Для использования функций библиотеки regex.h необходимо включить заголовочный файл <regex.h> и скомпилировать программу с флагом -lregex.

С помощью функций из библиотеки regex.h можно осуществлять поиск, замену, разбиение и извлечение данных из текстовых строк в соответствии с заданным шаблоном. В языке C для описания регулярных выражений используется специальный синтаксис, который позволяет указывать символы и конструкции для определения шаблона поиска.

Использование регулярных выражений и библиотеки regex.h позволяет обрабатывать текстовые данные эффективно, удобно и с минимальными усилиями.

Выполнение работы

Пользователь вводит текст, содержащий ссылки, который заканчивается строкой "Fin.". Функция input() считывает текст построчно и возвращает указатель на выделенную память, содержащую введенный текст.

Функция text_processing() компилирует регулярное выражение для поиска ссылок в тексте. Регулярное выражение составлено таким образом, чтобы соответствовать ссылкам, которые могут начинаться с протокола (например, "http://"), иметь префикс "www." перед доменным именем сайта и содержать путь к файлу и имя файла с расширением.

После компиляции регулярного выражения функция text_processing() разбивает текст на слова с помощью разделителей (пробел, табуляция, новая строка). Для каждого слова в тексте проверяется соответствие регулярному выражению.

Если слово соответствует регулярному выражению, вызывается функция link_processing() для обработки ссылки. Функция link_processing() извлекает название сайта и имя файла из ссылки.

Для извлечения названия сайта функция link_processing() сначала проверяет, начинается ли ссылка с протокола. Если да, то она извлекает название сайта как часть после протокола и до первого слеша. Если ссылка не начинается с протокола, то функция link_processing() извлекает название сайта как часть до первого слеша.

После извлечения названия сайта функция link_processing() проверяет, содержит ли название сайта префикс "www.". Если да, то она удаляет префикс "www." из названия сайта.

Для извлечения имени файла функция link_processing() находит последний слеш в ссылке и извлекает часть после последнего слеша.

После извлечения названия сайта и имени файла функция link_processing() выводит название сайта, символ "-", имя файла и символ новой строки.

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 Many of them. Fin.	google.com track.mp3 google.com.edu - hello.avi	Пример успешной работы программы
2.	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.	qwe.edu.etu.yahooo.org.net.ru - qwe.q skype.com - qwe.avi	Пример успешной работы программы

Выводы

В ходе выполнения данной работы были усвоены основы работы с регулярными выражениями на языке C через создание и анализ примерной программы, использующей эту технику. Основные функциональности и принципы работы с регулярными выражениями были изучены, а также были рассмотрены практические примеры их применения.

Результатом стала разработанная программа, способная анализировать текстовый ввод и выделять в нем ссылки, применяя регулярные выражения. Затем программа выводит необходимую информацию, такую как название сайта и имя файла, связанные с каждой найденной ссылкой.

Важным аспектом данной работы является не только само освоение работы с регулярными выражениями, но и приобретение опыта в применении этого инструмента для анализа и обработки текстовых данных в программном контексте. Этот проект стимулировал развитие навыков программирования и позволил погрузиться в область работы с текстовой информацией и методами ее обработки в программном коде.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

#define PATTERN
"\\(\\w\\+://\\)\\?\\(\\([wW]\\)\\{3\\}.\\)\\?\\(\\w\\+\\.\\)\\+\\w\\+\\(
\\w\\+/\\)\\+\\w\\+\\.\\w\\+"
#define SEP "\n\t "
#define WWW_LEN 4

char* input() {
     char* text = calloc(10, sizeof(char));
     char symbol;
     int i = 0;
     while(1) {
          symbol = getchar();
          text = realloc(text, (i+10)*sizeof(char));
          text[i] = symbol;
          if(i>2
          && text[i-3] == 'F'
          && text[i-2] == 'i'
          && text[i-1] == 'n'
          && text[i] == '.'){
               text[i+1] = '\0';
               break;
          }
          i++;
     }
     return text;
}

void link_processing(char* link) {
    char* host_start = strstr(link, "://");
    if (host_start) {
        host_start += 3;
    } else {
        host_start = link;
    }

    if (strstr(host_start, "www.") == host_start) {
        host_start += 4;
    }

    char* host_end = strchr(host_start, '/');
    if (!host_end) {
        host_end = host_start + strlen(host_start);
    }

    while (host_start < host_end) {
```

```
            printf("%c", *host_start);
            host_start++;
        }
        printf(" - ");
        char* file_name_start = strrchr(link, '/');
        if (file_name_start) {
            file_name_start++;
        } else {
            file_name_start = link;
        }
        printf("%s\n", file_name_start);
    }


    void text_processing(char *text) {
        regex_t regular_expression;
        regcomp(&regular_expression, PATTERN, 0);
        size_t nmatch = 1;
        regmatch_t pmatch[nmatch];
        char *word;
        word = strtok(text, SEP);
        while(1) {
            if(strcmp(word, "Fin.") == 0) break;
            if((regexec(&regular_expression, word, nmatch, pmatch, 0)
== 0)) {
                link_processing(word);
            }
            word = strtok(NULL, SEP);
        }
    }

    int main() {
        char* text = input();
        text_processing(text);
        return 0;
    }
```