**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: **Регуляторные выражения**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3342 |  | Галеев А.Д. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Цель данной работы состоит в исследовании и практическом применении регулярных выражений в языке программирования C. Основная задача заключается в изучении синтаксиса, возможностей и применений регулярных выражений для обработки текстовых данных в программировании

## **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](https://e.moevm.info/mod/lesson/view.php?id=509), 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

Для решения задач в программе использовались функции стандартной библиотеки языка си, а так-же библиотека regex.h, которая предоставляет возможность использовать регулярные выражения в программировании на языке C. Она является стандартной частью библиотеки языка C и предназначена для работы с регулярными выражениями в стиле POSIX

## **Выполнение работы**

Функция main()

* Вызывается функция read\_text(), которая читает вводимый текст построчно из стандартного ввода и сохраняет его в массиве строк text. Функция выделяет динамическую память для массива text и каждой строки текста.
* Затем определяется строковая переменная pattern, содержащая регулярное выражение для поиска ссылок в тексте. Это выражение задает шаблон для поиска строк, соответствующих URL-адресам.
* Создается экземпляр структуры regex\_t под названием ptrn, который будет хранить скомпилированное регулярное выражение. Функция regcomp() компилирует регулярное выражение из строки pattern в эту структуру. Если компиляция не удалась, программа выведет сообщение об ошибке и завершится.
* Далее происходит итерация по всем строкам в массиве text, и для каждой строки вызывается функция print\_link(), которая ищет в строке совпадения с регулярным выражением и выводит найденные ссылки в формате "ссылка - файл".
* После обработки всех строк освобождается выделенная динамическая память для массива text и каждой строки текста.
* Вызывается функция regfree() для освобождения ресурсов, занятых компилированным регулярным выражением.

Функция read\_text() - отвечает за чтение текста

* Выделяется динамическая память под массив указателей на строки text с помощью функции malloc().
* В цикле while считываются строки из стандартного ввода с помощью функции fgets() и сохраняются в buffer.
* Если считанная строка не пустая копируется содержимое buffer в ново-выделенную память с помощью функции strcpy().
* Если считанная строка совпадает с "Fin.", цикл чтения текста завершается.
* После завершения чтения текста, массив text переразмечается с использованием функции realloc(), чтобы освободить любую неиспользуемую память, которая могла быть выделена сверх count. В этот момент переменная count содержит количество фактически считанных строк.
* Функция возвращает указатель на массив строк text.

Функция print\_link - отвечает за поиск и вывод ссылок из строки текста согласно регулярному выражению.

* Функция принимает два аргумента regex\_t \*ptrn (указатель на скомпилированное регулярное выражение) и char \*sentence (указатель на строку текста, в которой нужно искать ссылки).
* Выполняется вызов функции regexec(), которая ищет совпадения между регулярным выражением, заданным в ptrn, и строкой sentence. Если совпадение найдено, программа продолжает работу
* Из массива match\_str извлекаются подстроки, соответствующие группам регулярного выражения.
* Далее происходит вывод согласно заданию

**Тестирование**

Результаты тестирования представлены в табл. 1  
Таблица 1 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| № проверки | Входные данные | Выходные данные |
| 1. | This is simple url:   http://www.google.com/track.mp3 | google.com - track.mp3 |
| 2. | 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 | google.com.edu - hello.avi qwe.edu.etu.yahooo.org.net.ru - qwe.q |
| 3. | Some other protocols  ftp://skype.com/qqwe/qweqw/qwe.avi  Fin. | skype.com - qwe.avi |

## Выводы

Было изучено понятие регуляторных выражений.

Разработана программа выполняющая поиск и вывод форматированных ссылок в заданном тексте.

# Приложение А Исходный код программы

Название файла: main\_lb1

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <regex.h>

#define MAX\_TEXT 1000

#define MAX\_LENGTH 100

char \*\*read\_text() {

char \*\*text = (char \*\*)malloc(MAX\_TEXT \* sizeof(char \*));

if (!text) {

fprintf(stderr, "ошибка выделения памяти\n");

exit(EXIT\_FAILURE);

}

char buffer[MAX\_LENGTH];

int count = 0;

while (fgets(buffer, sizeof(buffer), stdin) != NULL) {

buffer[strcspn(buffer, "\n")] = '\0';

text[count] = (char \*)malloc(strlen(buffer) + 1);

if (!text[count]) {

fprintf(stderr, "ошибка выделения памяти в предложении [%d]\n", count);

exit(EXIT\_FAILURE);

}

strcpy(text[count], buffer);

if (strcmp(buffer, "Fin.") == 0) {

break;

}

count++;

}

text = (char \*\*)realloc(text, (count + 1) \* sizeof(char \*));

if (!text) {

fprintf(stderr, "ошибка выделения памяти\n");

exit(EXIT\_FAILURE);

}

text[count] = NULL;

return text;

}

void print\_link(regex\_t \*ptrn, char \*sentence) {

int N;

regmatch\_t match\_str[7];

N = regexec(ptrn, sentence, 7, match\_str, 0);

regmatch\_t link = match\_str[3];

regmatch\_t file = match\_str[6];

if (!N) {

for (int i = link.rm\_so; i < link.rm\_eo; i++) {

printf("%c", sentence[i]);

}

printf(" - ");

for (int i = file.rm\_so; i < file.rm\_eo; i++) {

printf("%c", sentence[i]);

}

printf("\n");

}

}

int main() {

char \*\*text;

text = read\_text();

char \*pattern = "([a-z]+://)?(www\\.)?([a-zA-Z0-9]+(\\.[a-zA-Z0-9]+)+)/([a-zA-Z0-9]+/)\*([a-zA-Z0-9]+\\.[a-zA-Z0-9]+)";

regex\_t ptrn;

if (regcomp(&ptrn, pattern, REG\_EXTENDED) != 0) {

fprintf(stderr, "ошибка компиляции регуляторного выражения\n");

exit(EXIT\_FAILURE);

}

for (int i = 0; text[i] != NULL; i++) {

print\_link(&ptrn, text[i]);

free(text[i]);

}

regfree(&ptrn);

free(text);

return 0;

}