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, 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;
}
```