**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. 3341 |  | Шуменков А.П. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Целью работы является освоение работы с регулярными выражениями на языке C.

Для достижения поставленной цели требуется решить следующие задачи:

* ознакомиться с регулярными выражениями;
* научиться их использовать;
* написать программу, решающую задачу в соответствии с индивидуальным условием с использованием регулярных выражений.

## Задание

2 вариант.

На вход программе подается текст, представляющий собой набор предложений с новой строки. Текст заканчивается предложением "Fin." В тексте могут встречаться примеры запуска программ в командной строке Linux. Требуется, используя регулярные выражения, найти только примеры команд в оболочке суперпользователя и вывести на экран пары <имя пользователя> - <имя\_команды>. Если предложение содержит какой-то пример команды, то гарантируется, что после нее будет символ переноса строки.

Примеры имеют следующий вид:

* Сначала идет имя пользователя, состоящее из букв, цифр и символа \_
* Символ @
* Имя компьютера, состоящее из букв, цифр, символов \_ и -
* Символ : и ~
* Символ $, если команда запущена в оболочке пользователя и #, если в оболочке суперпользователя. При этом между двоеточием, тильдой и $ или # могут быть пробелы.
* Пробел
* Сама команда и символ переноса строки.

## Выполнение работы

1. Директивы препроцессора и константы:

- #include <stdlib.h>, #include <stdio.h>, #include <string.h>, #include <regex.h>: Подключают нужные заголовочные файлы для работы с динамической памятью, вводом/выводом, строками и регулярными выражениями.

- #define PATTERN "([a-zA-Z0-9\_]+)@[a-zA-Z0-9\_-]+: \*~ \*# (.\*)": Определяет шаблон регулярного выражения для поиска заданного формата текста.

- #define END\_WORD "Fin.": Определяет строку-маркер для завершения ввода текста.

- #define START\_TEXT\_SIZE 10: Определяет начальный размер буфера для текста.

2. Основная функция main():

- Создается структура regex\_comp для компиляции регулярного выражения из шаблона.

- Запускается основной цикл while, который продолжается до ввода END\_WORD.

- При каждой итерации выделяется память под text, считывается ввод и проверяется соответствие регулярному выражению.

- Если введен END\_WORD, устанавливается флаг flag для завершения цикла.

3. Функция input\_text():

- Считывает текст из ввода пользователя и расширяет буфер при необходимости.

- При обнаружении END\_WORD прерывает чтение и завершает строку.

4. Функция regex():

- Производит поиск совпадений регулярного выражения в тексте.

- Если совпадения найдены, вызывает функцию print\_regex() для вывода найденных выражений.

5. Функция print\_regex():

- Выводит соответствующий фрагмент текста, который совпадает с найденным регулярным выражением.

Разработанный программный код см. в приложении А.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | Run docker container:  kot@kot-ThinkPad:~$ docker run -d --name  stepik stepik/challenge-avr:latest You can get into running /bin/bash  command in interactive mode:  kot@kot-ThinkPad:~$ docker  exec -it stepik "/bin/bash" Switch user: su : root@84628200cd19: ~ # su box box@84628200cd19: ~ $ ^C Exit from box: box@5718c87efaa7: ~ $ exit exit from container:  root@5718c87efaa7: ~ # exit kot@kot-ThinkPad:~$ ^C Fin. | root - su box  root - exit | root@84628200cd19: ~ # su box и root@5718c87efaa7: ~ # exit - команды в оболочке суперпользователя, нужно вывести пары <имя пользователя> - <имя команды>, т.е в первой строке root - su box, во второй root – exit. |
|  | lmdlld  root\_@cd19: ~ # try1  lmlld [rt@-](mailto:root@cd19): ~ # try2  lmlld [rt-@-](mailto:root@cd19): ~ # try3  Fin. | Fin.root\_ - try1  rt - try2 | 1) проверка наличия пробелов между знаком :, ~ и #.  2) проверка наличия символов до имени пользователя |

## Выводы

Была освоена работа с регулярными выражениями на языке C.

Для достижения поставленной цели были решены следующие задачи:

* ознакомление с регулярными выражениями;
* их использование;
* написана программа, которая, используя регулярные выражения, находит только примеры команд в оболочке суперпользователя и выводит на экран пары <имя пользователя> - <имя команды>.

# Приложение А Исходный код программы

Название файла: main.c

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <regex.h>

#define PATTERN "([a-zA-Z0-9\_]+)@[a-zA-Z0-9\_-]+: \*~ \*# (.\*)"

#define END\_WORD "Fin."

#define START\_TEXT\_SIZE 10

int main();

void regex(char\* text, regex\_t regex\_comp);

void input\_text(char\*\* text);

void print\_regex(char\* text, regmatch\_t match);

int main(){

regex\_t regex\_comp;

regcomp(&regex\_comp, PATTERN, REG\_EXTENDED);

int flag = 0;

while (flag != 1) {

char\* text = (char\* )malloc(START\_TEXT\_SIZE \* sizeof(char));

input\_text(&text);

regex(text, regex\_comp);

if(strcmp(text, END\_WORD) == 0){

flag = 1;

}

}

}

void input\_text(char\*\* text){

char c;

int text\_size = START\_TEXT\_SIZE, i = 0;

while ((c = getchar()) != '\n'){

if(i + 1 == text\_size){

text\_size += START\_TEXT\_SIZE;

(\*text) = (char\*)realloc((\*text), text\_size \* sizeof(char));

}

(\*text)[i++] = c;

if(strcmp((\*text), END\_WORD) == 0){

break;

}

(\*text)[i] = '\0';

}

}

void regex(char\* text, regex\_t regex\_comp){

regmatch\_t matches[3];

if (regexec(&regex\_comp, text, 3, matches, 0) == 0) {

print\_regex(text, matches[1]);

printf(" - ");

print\_regex(text, matches[2]);

printf("\n");

}

}

void print\_regex(char\* text, regmatch\_t match) {

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

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

}

}