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
1.	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.
2.	lmdlld root_@cd19: ~ # try1 lmlld rt@-: ~ # try2 lmlld rt-@-: ~ # 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]);
      }
}
```