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. 3344 _____ Ханнанов А.Ф.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Получить навыки в составлении регулярных выражений. Научиться применять их в работе на языке Си.

Задание.

Вариант 2

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 слеующий 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

Программа начинается с того, что задаёт регулярное выражение в массив regexString, после компиляции выражения оно будет находиться в regexCompiled. Далее идёт проверка на успешную компиляцию. После успешной компиляции выражения задаётся динамический массив text_line, в котором по очереди будут храниться строки входных данных. Следующим шагом идёт цикл, который будет выполняться, пока не будет встречена строка “Fin.”. В цикле идёт поочерёдный ввод строк. Эти строки проверяются на соответствие регулярному выражению. Если строка ему соответствует, то запускаются циклы, которые выводят первую и четвёртую группы выражения (имя пользователя и команда соответственно). После вывода этих данных массив обнуляется. После выполнения цикла высвобождается память массива и регулярного выражения.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Коммен тарии
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	-

Выводы

Были изучены правила составления регулярных выражений. Получены навыки написания регулярных выражений и их применения на практике, используя язык Си.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Khannanov_Artem_lb1.c

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

#define MAX_GROUPS 5


int main() {

    char *regexString = "(\\w+)@([A-Za-z0-9_-]+)(: ?\\~ ?\\# )(.+\n)";

    regex_t regexCompiled;
    regmatch_t groupArray[MAX_GROUPS];

    if (regcomp(&regexCompiled, regexString, REG_EXTENDED)) {
        printf("[Can't compile expression]\n");
        return 0;
    }

    char *text_line = (char *)malloc(sizeof(char) * 1000);


    while (fgets(text_line, 1000, stdin) != NULL) {

        if (strncmp(text_line, "Fin.\n", 5) == 0) break;

        if (regexec(&regexCompiled, text_line, 5, groupArray, 0) == 0) {
            for (size_t i = groupArray[1].rm_so; i < groupArray[1].rm_eo;
i++) {
                printf("%c", text_line[i]);
            }

            printf(" - ");

            for (size_t j = groupArray[4].rm_so; j < groupArray[4].rm_eo;
j++) {
                printf("%c", text_line[j]);
            }
        }
            free(text_line);
            text_line = (char *)malloc(sizeof(char) * 1000);
    }

    free(text_line);
    regfree(&regexCompiled);

    return 0;
}
```