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. 3384 _____ Рудаков А.Л.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы.

Изучить теорию работы с регулярными выражениями и интеграции их в язык С. Получить практические навыки путем написания программного кода.

Задание.

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.

В начале программы была подключен заголовочный файл *stdio.h*, в котором объявлены функции стандартного ввода и вывода. Также были подключены заголовочные файлы *stdlib.h*, в котором объявлены функции для работы с памятью и *regex.h*, в котором объявлены функции для работы с регулярными выражениями.

После это был создан макрос *BLOCK* размером 20 при помощи директивы *define*.

Далее инициализирована переменная *reegex* типа *regex_t* и инициализирован динамический массив строк *massive*. Далее при помощи функции *scan* выполняется считывание текста и его запись в *massive*. Считывание происходит до момента, пока не будет считана строка *Fin*. Переменная *csnt* отвечает за хранение количества считанных строк.

После этого инициализирована переменная *value* типа *int*, которой присваивается значение выполнения функции *regcomp* — 0, если успешно, значение макроса *REG_EXTENDED*, если не успешно. В функции *regcomp(&reegex,"([A-Za-z0-9,_]+)@[A-Za-z0-9,_,-]+\\: ?\\~ ?\\# (.+)\n$",REG_EXTENDED)* в *reegex* записывается регулярное выражение. Далее создается массив структур *regmatch_t matches[3]*.

После этого идет цикл по массиву строк, в котором при помощи функции r*egexec(&reegex,massive[i],3,matches,0)* проверяется, удовлетворяет ли строка регулярному выражению, записанному в *reegex*. Если да, то значение *value == 0*, тогда выполняется присваивание переменным *namebeg*, *nameend*, *combeg*, *comend*, значений *matches[1].rm_so* — индекс начала группы имени пользователя, *matches[1].rm_eo* — индекс конца группы имени пользователя, *matches[2].rm_so* — индекс начала группы команды, *matches[2].rm_eo* — индекс конца группы команды, соответственно. После чего происходит вывод на экран *printf("%.*s - %.*s\n", nameend-namebeg, massive[i]+namebeg, comend-combeg, massive[i]+combeg)*.

В конце выполняется очистка памяти переменной *reegex* посредством функции *regfree(&reegex)*.

Разработанный программный код см. в приложении А.

Тестирование см. в приложении Б.

Выводы.

Была изучена теория работы с регулярными выражениями и интегрированием их в язык С. В работе была использованы функции для работы с регулярными выражениями, такие как regcomp(), regexec(), regfree().

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main_lb1.c

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

#define BLOCK 20
int scan(char ***massive){
      int flag=0;
         int csnt=0;
         int csmb=0;
      int n=1;
      char symbol;
         while (flag!=1){
                  symbol=getchar();
                  if (symbol=='\n' || symbol==EOF){
                  (*massive)[csnt][csmb]='\n';
                  (*massive)[csnt][csmb+1]='\0';
                  csmb=0;
                  n=1;
                  if (strcmp((*massive)[csnt],"Fin.\n")==0){ flag=1;
break; }
                  csnt++;
                  (*massive)=(char **)realloc((*massive),sizeof(char
*)*(csnt+1));
                  (*massive)[csnt]=(char *)malloc(BLOCK*sizeof(char));
                  }else{
                  if (strlen((*massive)[csnt])>=BLOCK*n){
                        n+=1;
                        (*massive)[csnt]=(char
*)realloc((*massive)[csnt],BLOCK*n*sizeof(char));
                  }
                  (*massive)[csnt][csmb]=symbol;
                  csmb++;
            }
         }
      return csnt;
}
int main(){
      regex_t reegex;
      char ** massive = (char **)malloc(sizeof(char*));
      massive[0]=(char *)malloc(BLOCK*sizeof(char));
      int csnt=scan(&massive);
      int value;
      value=regcomp(&reegex,"([A-Za-z0-9,_]+)@[A-Za-z0-
9,_,-]+\\: ?\\~ ?\\# (.+)\n$",REG_EXTENDED);
      regmatch_t matches[3];
      for (int i=0; i<csnt; i++){
            value=regexec(&reegex,massive[i],3,matches,0);
            int namebeg,nameend,combeg,comend;
            if (value==0){
                  namebeg=matches[1].rm_so;
                  nameend=matches[1].rm_eo;
```

```
                combeg=matches[2].rm_so;
                comend=matches[2].rm_eo;
                printf("%.*s - %.*s\n", nameend-namebeg,
massive[i]+namebeg, comend-combeg, massive[i]+combeg);
            }
      }
      regfree(&reegex);
}
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

Таблица Б.1 - Примеры тестовых случаев

№ 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	Верно