

EXPERIMENT 7

NAME: Rahil Sharma

PRN: 18070123062

BATCH: EA-3

SUBJECT: ESRTOS

AIM: To use Loops and Conditional Statements in Linux

THEORY:

- **Operators in C:**

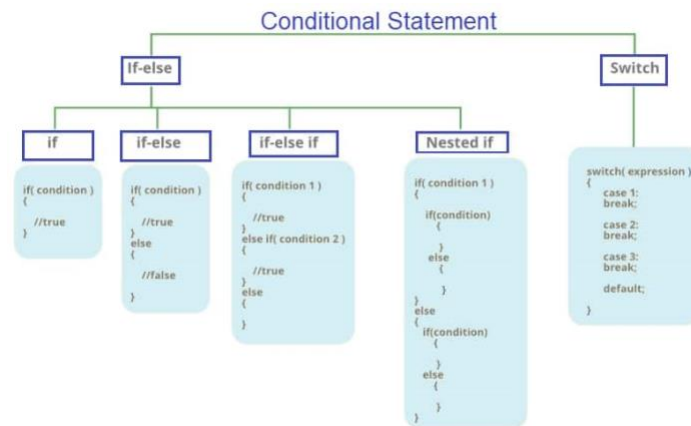
An operator is a symbol that operates on a value or a variable. For example: + is an operator to perform addition.

C has a wide range of operators to perform various operations.

	Operator	Type
Unary operator	+ +, --	Unary operator
	+, -, *, /, %	Arithmetic operator
	<, <=, >, >=, ==, !=	Relational operator
Binary operator	&&, , !	Logical operator
	&, , <<, >>, ~, ^	Bitwise operator
	=, +=, -=, *=, /=, %=	Assignment operator
Ternary operator	?:	Ternary or conditional operator

- Conditional Statements in C:

Conditional Statements in C programming are used to make decisions based on the conditions. Conditional statements execute sequentially when there is no condition around the statements. It is also called as branching as a program decides which statement to execute based on the result of the evaluated condition.

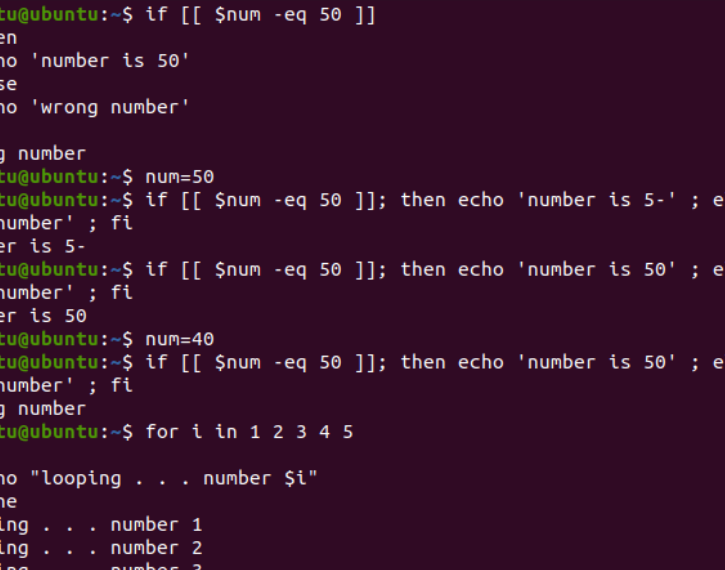


CODES & OUPUTS OF THE PROGRAM:

1. Task 1: Using if else loop on the Linux Terminal

```
ubuntu@ubuntu: ~  
ubuntu@ubuntu:~$ if [[ $num -eq 50 ]]  
> then  
> echo 'number is 50'  
> else  
> echo 'wrong number'  
> fi  
wrong number  
ubuntu@ubuntu:~$ num=50  
ubuntu@ubuntu:~$ if [[ $num -eq 50 ]]; then echo 'number is 5-' ; else echo 'wrong number' ; fi  
number is 5-  
ubuntu@ubuntu:~$ if [[ $num -eq 50 ]]; then echo 'number is 50' ; else echo 'wrong number' ; fi  
number is 50  
ubuntu@ubuntu:~$ num=40  
ubuntu@ubuntu:~$ if [[ $num -eq 50 ]]; then echo 'number is 50' ; else echo 'wrong number' ; fi  
wrong number  
ubuntu@ubuntu:~$ for i in 1 2 3 4 5  
> do  
> echo "looping . . . number $i"  
> done  
looping . . . number 1  
looping . . . number 2  
looping . . . number 3  
looping . . . number 4  
looping . . . number 5  
ubuntu@ubuntu:~$ for i in hello 1 * 2 goodbye  
> do
```

2. For Loop in Linux Terminal



```

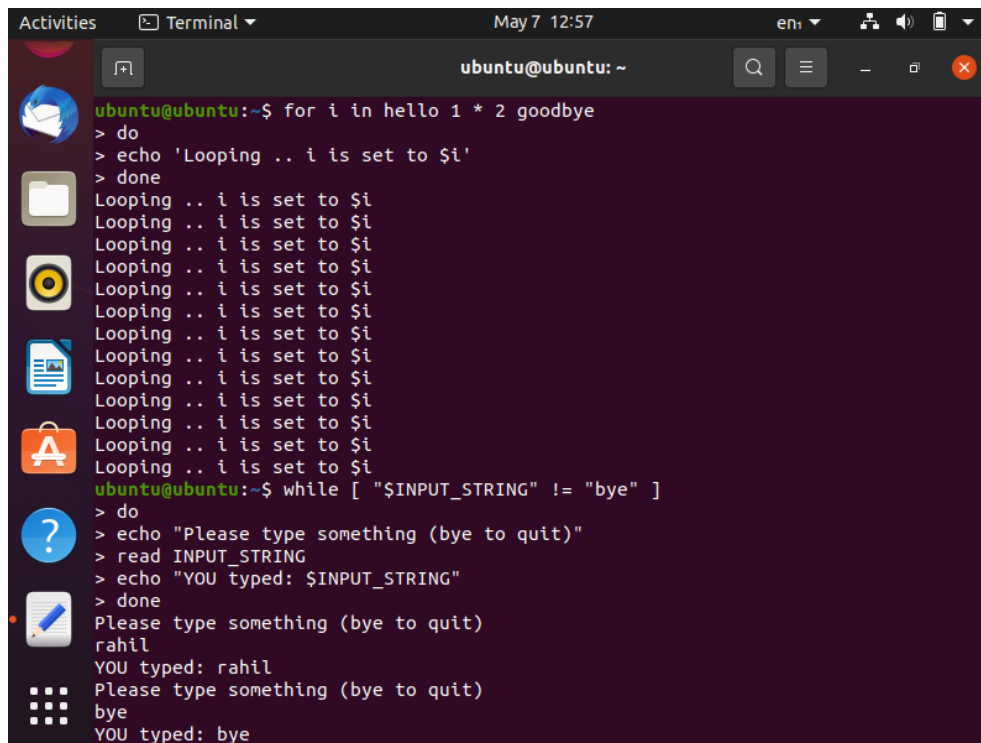
ubuntu@ubuntu:~$ if [[ $num -eq 50 ]]
> then
> echo 'number is 50'
> else
> echo 'wrong number'
> fi
wrong number
ubuntu@ubuntu:~$ num=50
ubuntu@ubuntu:~$ if [[ $num -eq 50 ]]; then echo 'number is 5-' ; else echo 'wr
ong number' ; fi
number is 5-
ubuntu@ubuntu:~$ if [[ $num -eq 50 ]]; then echo 'number is 50' ; else echo 'wr
ong number' ; fi
number is 50
ubuntu@ubuntu:~$ num=40
ubuntu@ubuntu:~$ if [[ $num -eq 50 ]]; then echo 'number is 50' ; else echo 'wr
ong number' ; fi
wrong number
ubuntu@ubuntu:~$ for i in 1 2 3 4 5
> do
> echo "looping . . . number $i"
> done
looping . . . number 1
looping . . . number 2
looping . . . number 3
looping . . . number 4
looping . . . number 5
ubuntu@ubuntu:~$ for i in hello 1 * 2 goodbye
> do

```

A screenshot of a Linux desktop environment with a dark theme. The top panel shows system status icons and the date/time "May 7 12:57". Below it is a dock with application icons. The main area is a terminal window titled "ubuntu@ubuntu: ~". It contains two shell scripts. The first script uses a `for` loop to print "Looping .. i is set to \$i" for values 1 through 10. The second script uses a `while` loop to prompt the user for input; if the input is anything other than "bye", it prints "YOU typed: \$INPUT_STRING" and loops back. The user has entered "rahil" and "bye" during the execution of the second script.

```
ubuntu@ubuntu:~$ for i in hello 1 * 2 goodbye
> do
> echo 'Looping .. i is set to $i'
> done
Looping .. i is set to $i
Looping .. i is set to $i
Looping .. i is set to $i
Looping .. i is set to $i
Looping .. i is set to $i
Looping .. i is set to $i
Looping .. i is set to $i
Looping .. i is set to $i
Looping .. i is set to $i
Looping .. i is set to $i
ubuntu@ubuntu:~$ while [ "$INPUT_STRING" != "bye" ]
> do
> echo "Please type something (bye to quit)"
> read INPUT_STRING
> echo "YOU typed: $INPUT_STRING"
> done
Please type something (bye to quit)
rahil
YOU typed: rahil
Please type something (bye to quit)
bye
YOU typed: bye
```

3. While Loop in Linux Terminal



The screenshot shows a Linux terminal window with a dark purple background. The terminal title bar indicates the user is 'ubuntu@ubuntu' and the time is 'May 7 12:57'. The terminal displays two loops. The first is a 'for' loop that iterates over the words 'hello', '1', '*', '2', and 'goodbye', printing 'Looping .. i is set to \$i' for each. The second is a 'while' loop that continues to prompt the user for input until they type 'bye'. The user's inputs 'rahil' and 'bye' are shown, along with the corresponding 'YOU typed:' output.

```
ubuntu@ubuntu:~$ for i in hello 1 * 2 goodbye
> do
> echo 'Looping .. i is set to $i'
> done
Looping .. i is set to $i
Looping .. i is set to $i
Looping .. i is set to $i
Looping .. i is set to $i
Looping .. i is set to $i
Looping .. i is set to $i
Looping .. i is set to $i
Looping .. i is set to $i
Looping .. i is set to $i
Looping .. i is set to $i
Looping .. i is set to $i
Looping .. i is set to $i
Looping .. i is set to $i
Looping .. i is set to $i
Looping .. i is set to $i
ubuntu@ubuntu:~$ while [ "$INPUT_STRING" != "bye" ]
> do
> echo "Please type something (bye to quit)"
> read INPUT_STRING
> echo "YOU typed: $INPUT_STRING"
> done
Please type something (bye to quit)
rahil
YOU typed: rahil
Please type something (bye to quit)
bye
YOU typed: bye
```

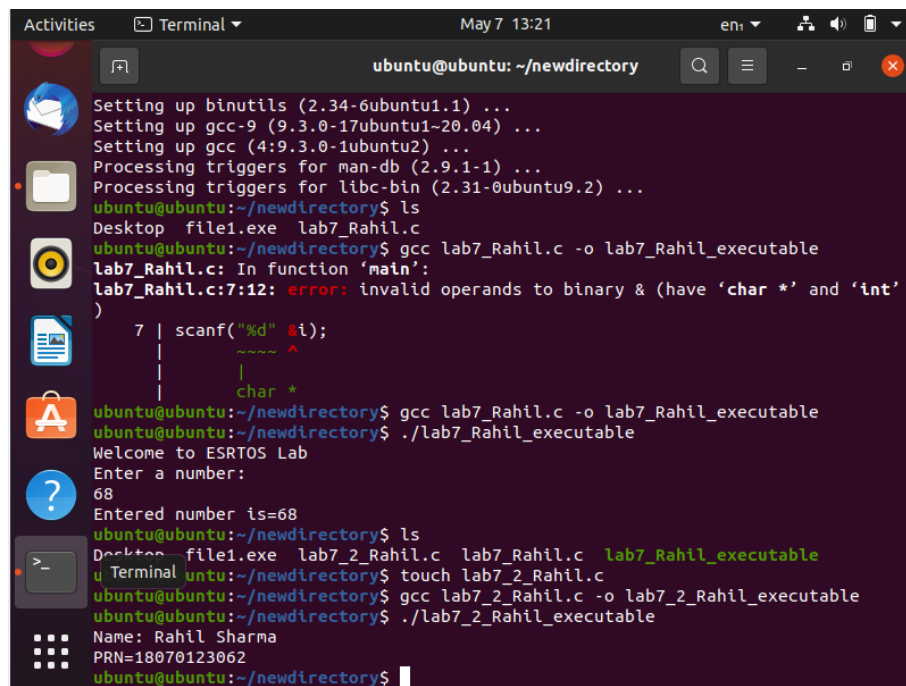
4. Displaying the C program Output in Linux Terminal

Code for C Program:

- a)

```
#include <stdio.h>
int main()
{
    int i;
    printf("Welcome to ESRTOS LAB \n");
    printf("Enter a number: \n");
    scanf("%d", &i);
    printf("Entered number is= %d \n",i);
    return 0
}
```
- b)

```
#include <stdio.h>
Void main()
{
    Printf("Name: Rahil Sharma");
    Printf("PRN: 18070123062");
}
```

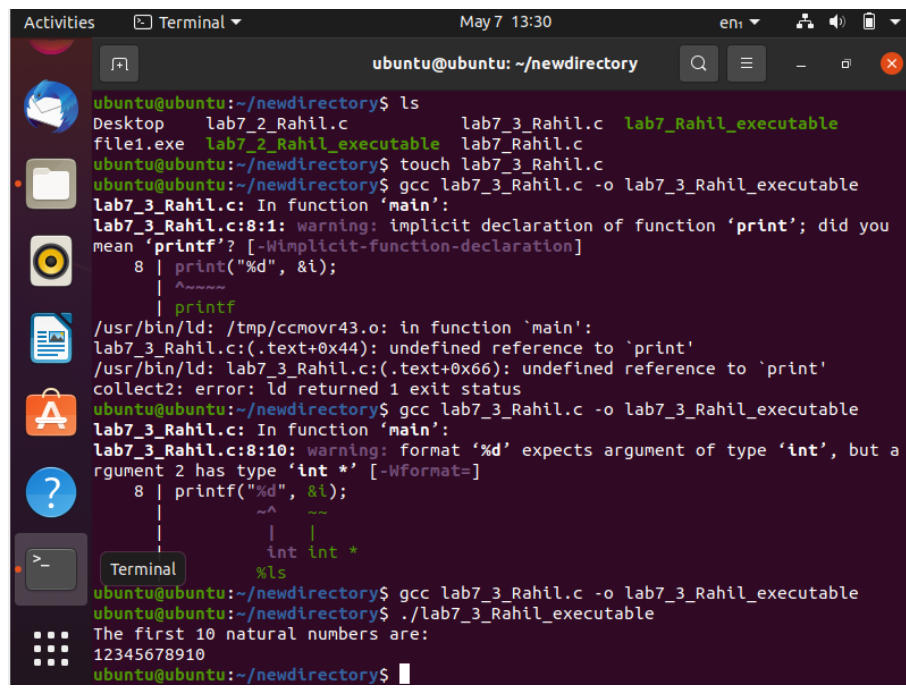


```
Activities Terminal May 7 13:21 en ubuntu@ubuntu: ~/newdirectory
Setting up binutils (2.34-6ubuntu1.1) ...
Setting up gcc-9 (9.3.0-17ubuntu1~20.04) ...
Setting up gcc (4:9.3.0-1ubuntu2) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
ubuntu@ubuntu:~/newdirectory$ ls
Desktop file1.exe lab7_Rahil.c
ubuntu@ubuntu:~/newdirectory$ gcc lab7_Rahil.c -o lab7_Rahil_executable
lab7_Rahil.c: In function 'main':
lab7_Rahil.c:7:12: error: invalid operands to binary & (have 'char *' and 'int'
)
7 | scanf("%d" &i);
  |             ^
  |             |
  |             char *
ubuntu@ubuntu:~/newdirectory$ gcc lab7_Rahil.c -o lab7_Rahil_executable
ubuntu@ubuntu:~/newdirectory$ ./lab7_Rahil_executable
Welcome to ESRTOS Lab
Enter a number:
68
Entered number is=68
ubuntu@ubuntu:~/newdirectory$ ls
Desktop file1.exe lab7_2_Rahil.c lab7_Rahil.c lab7_Rahil_executable
ubuntu@ubuntu:~/newdirectory$ touch lab7_2_Rahil.c
ubuntu@ubuntu:~/newdirectory$ gcc lab7_2_Rahil.c -o lab7_2_Rahil_executable
ubuntu@ubuntu:~/newdirectory$ ./lab7_2_Rahil_executable
Name: Rahil Sharma
PRN=18070123062
ubuntu@ubuntu:~/newdirectory$
```

5. For Loop in C program to be printed in Linux Terminal:

C Program to print first 10 Natural Numbers:

```
a) #include <stdio.h>
void main()
{
    int i;
    printf("The first 10 natural numbers are:\n");
    for (i=1;i<=10;i++)
    {
        printf("%d ",i);
    }
    printf("\n");
}
```

A terminal window on Ubuntu showing the compilation and execution of a C program. The user lists files in the directory, then compiles 'lab7_3_Rahil.c' with 'gcc'. The first compilation attempt fails with errors: 'implicit declaration of function 'print'' and 'undefined reference to 'print''. The user then corrects the code to use 'printf' and recompiles. The second compilation is successful. Finally, the user runs the executable './lab7_3_Rahil_executable', which outputs 'The first 10 natural numbers are: 12345678910'.

```
ubuntu@ubuntu: ~/newdirectory
ubuntu@ubuntu:~/newdirectory$ ls
Desktop  lab7_2_Rahil.c  lab7_3_Rahil.c  lab7_Rahil_executable
file1.exe lab7_2_Rahil_executable lab7_Rahil.c
ubuntu@ubuntu:~/newdirectory$ touch lab7_3_Rahil.c
ubuntu@ubuntu:~/newdirectory$ gcc lab7_3_Rahil.c -o lab7_3_Rahil_executable
lab7_3_Rahil.c: In function 'main':
lab7_3_Rahil.c:8:1: warning: implicit declaration of function 'print'; did you
mean 'printf'? [-Wimplicit-function-declaration]
   8 |   print("%d", &i);
     |   ~~~~~
     |   printf
/usr/bin/ld: /tmp/ccmovr43.o: in function 'main':
lab7_3_Rahil.c:(.text+0x44): undefined reference to 'print'
/usr/bin/ld: lab7_3_Rahil.c:(.text+0x66): undefined reference to 'print'
collect2: error: ld returned 1 exit status
ubuntu@ubuntu:~/newdirectory$ gcc lab7_3_Rahil.c -o lab7_3_Rahil_executable
lab7_3_Rahil.c: In function 'main':
lab7_3_Rahil.c:8:10: warning: format '%d' expects argument of type 'int', but a
rgument 2 has type 'int *' [-Wformat=]
   8 |   printf("%d", &i);
     |             ~^~
     |             |
     |             int int *
     |             %ls
ubuntu@ubuntu:~/newdirectory$ gcc lab7_3_Rahil.c -o lab7_3_Rahil_executable
ubuntu@ubuntu:~/newdirectory$ ./lab7_3_Rahil_executable
The first 10 natural numbers are:
12345678910
ubuntu@ubuntu:~/newdirectory$
```

6. Conditional Program in C:

C Program for displaying output:

```
#include<stdio.h>
int main()
{
    int num;
    printf("Enter Any Number : ");
    scanf("%d",&num);
    if(num<10)
        printf("\n%d is Less Than 10",num);
    return 0;
}
```

```
Activities Terminal May 7 13:36 en1
ubuntu@ubuntu: ~/newdirectory

| printf
/usr/bin/ld: /tmp/ccmovr43.o: in function 'main':
lab7_3_Rahil.c:(.text+0x44): undefined reference to `print'
/usr/bin/ld: lab7_3_Rahil.c:(.text+0x66): undefined reference to `print'
collect2: error: ld returned 1 exit status
ubuntu@ubuntu:~/newdirectory$ gcc lab7_3_Rahil.c -o lab7_3_Rahil_executable
lab7_3_Rahil.c: In function 'main':
lab7_3_Rahil.c:8:10: warning: format '%d' expects argument of type 'int', but a
rgument 2 has type 'int *' [-Wformat=]
8 | printf("%d", &i);
  |           ~^   ~~~
  |           int int *
  |           %ls
4 ubuntu@ubuntu:~/newdirectory$ gcc lab7_3_Rahil.c -o lab7_3_Rahil_executable
ubuntu@ubuntu:~/newdirectory$ ./lab7_3_Rahil_executable
The first 10 natural numbers are:
12345678910
ubuntu@ubuntu:~/newdirectory$ ls
Desktop          lab7_2_Rahil_executable  lab7_4_Rahil.c
file1.exe        lab7_3_Rahil.c           lab7_Rahil.c
lab7_2_Rahil.c   lab7_3_Rahil_executable  lab7_Rahil_executable
ubuntu@ubuntu:~/newdirectory$ touch lab7_4_Rahil.c
ubuntu@ubuntu:~/newdirectory$ gcc lab7_4_Rahil.c -o lab7_4_Rahil_executable
ubuntu@ubuntu:~/newdirectory$ ./lab7_4_Rahil_executable
Enter any Number:6
6 is less than 10ubuntu@ubuntu:~/newdirectory$
```

Conclusion: From this experiment we have understood how to work on LINUX terminal and C programs and siplay outputs for Conditional Statement programs.