CHAPTER 6

DECISION MAKING AND LOOPING

INTRO:

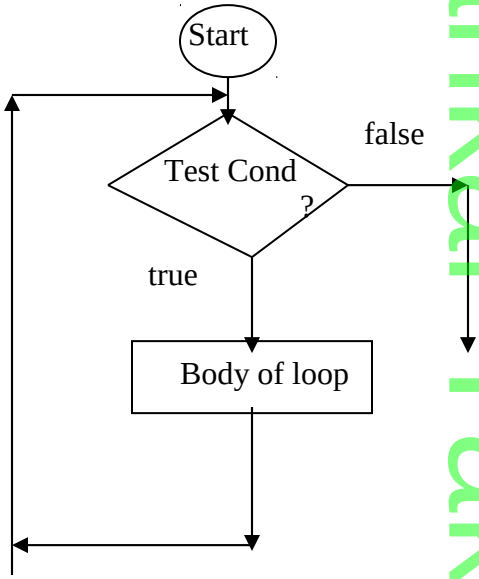
PROGRAM)

To calculate the sum of squares of all integers between 1 and 10.

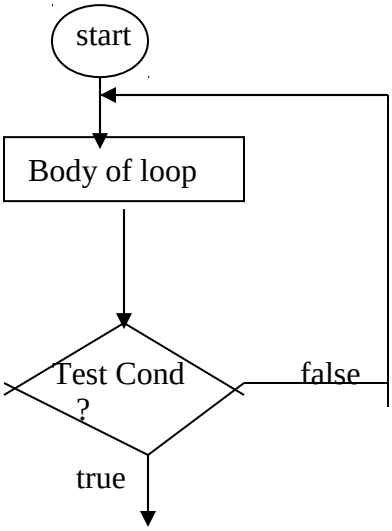
```
Main()
{
        int sum, n;
        n = 1;
        loop:
        sum = sum + n * n;
        if(n == 10)
                goto print;
        else
        {
                n = n+1;
                goto loop;
        }
        print:
                printf("%d",sum);
}
```

There are two types of loops. They are,
*) Entry controlled loop
*) Exit controlled loop

Entry Controlled Loop



Exit Controlled Loop



In case of exit controlled loop, the test is performed at end of body of loop and therefore the body is executed unconditionally for the first time.

A looping process, in general, include four steps,

1) Setting and initialization of a counter.
2) Execution of the statements in the loop.
3) Test for a specified condition for execution of the loop.
4) Incrementing the counter.

The C language provides three loop constructs for performing loop operations.

1) While Statement.
2) DO Statement.
3) FOR Statement

(*) **<u>While Statement</u>**:-

syntax:-

```
while (test Condition )
{
        body of loop;
}
```

It is an entry controlled loop statement. The test condition is evaluated first and if true, body of the loop is executed. After the execution of the body, the test condition is once again evaluated and the process begins as same. If condition is false the control is transferred out of the loop.

```
Prog) sum = 0
        n = 1;
while ( n  <= 10)
{
        sum = sum + n*n;
        n = n + 1;
}
printf("Sum = %d",sum);
```

prog) wap to evaluate the equation
y = x to the power of n

sol:

```
y = 1;
counter =1;
while(counter <=n)
{
        y = y * x;
        counter = counter + 1;
}
printf("x = %d to power of n =%d is y = %d \n",x,n,y);
```

(*) **Do Statement**:-

syntax:-

```
do
{
        body of loop;
}while(test condition);
```

It is an exit controlled loop statement. The body of the loop is executed first, at the end of loop, the test condition in the while statement is evaluated. If the condition is true, the program continues to evaluate the body of the loop once again. If the condition is false, the loop is terminated and control goes to the statement immediately after the while statement.

(*) **FOR Statement:-**

it is an entry controlled loop that provides a more concise loop control structure.

Syntax:-

```
for(initialization; test condition; increment)
{
        body of the loop;
}
```

ex:;

```
for(x = 0; x <= 10; x++)
{
printf("%d",x);
}
```

Features of for loop:-

1) More than one variables can be initialized at a time in the for statement.

Ex:

```
        P = 1;
    for(x = 0; x <= 10; x++)
            {
            printf("%d",x);
            }
```

can be written as,

```
    for(p = 1, x = 0; x <= 10; x++)
            {
            printf("%d",x);
            }
```

2) The increment section may also have more than one part.

Ex:

```
            for(p=1,x = 0; x <= 10; x++,p++)
            {
            printf("%d %d",x,p);
            }
```

3) The test condition may have compund relation and the testing need not be limited only to the loop control variable.

Ex:

```
            for(sum =1,x = 0; x <= 10 && sum < 50; x++)
            {
            sum = sum + x;
            printf("%d",x);
            }
```

4) One or more sections can be omitted, if necessary.

Ex:

```
            x = 0
            for(; x <= 10; x++)
            {
            printf("%d",x);
            }

            x = 0
            for(; x <= 10; )
            {
            printf("%d",x);
            x = x + 1;
            }
```

Nesting of for loops:-

i-e one for statement within another.

Ex:

```
for(x = 0;x <= 10; x++)
{
        printf("%d",x);
        for(y = 0 ; y<10; y++)
        {
                sum = x + y;
        }
}
```

The nesting may continue upto 15 levels according to ANSI C.

Prog) Wap to print multiplication table from 1 to 10.

```
for(row = 1; row <= ROWMAX;row++)
{
        for(col = 1 ; col <= COLMAX; col++)
        {
                prod = row * col;
               printf("%d * %d = %d\n",row,col,prod);
        }
        printf("\n");
}
```

Jumping Out of Loop(Using **break** statement or **goto** statement)

When **break** statement is encountred inside a loop, the loop is immediately exited and the program continues with statement immediately following the loop.
goto is used to exit from deeply nested loops when an error occurs. A simple **break** will not work out here.

Prog) to illustrate use of break

```
main()
{
        int m;
        float x , sum , avg;
        printf("THE PROG COMPUTES THE AVG OF SET OF NO'S\n");
        printf("Enter a negative number to end the series\n");
        sum =0;
```

```
        for(m =1; m <= 100;m++)
{
        scanf("%f",&x);
if (x < 0)
{
        break;
}
sum = sum + x;
}
avg = sum / (float)(m-1);
printf("No of values = %d\n",m –1);
printf("sum = %f \n",sum);
printf("avg=%f\n",avg);
} /* EOM*/


/*88888888888888888888888888888888888888888888888888*/
```