# Chapter 4-
# Nested If, Loop

# NESTED IF ELSE STATEMENT

```c
#include<stdio.h>

main()
{
int x=20,y=30;

    if(x==20)
    {
        if(y==30)
        {
            printf("value of x is 20, and value of y is 30.");
        }
    }
}
```

# LOOP

Every programming language has the feature to instruct to do such repetitive tasks with the help of certain form of statements. The process of repeatedly executing a collection of statement is called **Loop**.

# FOR LOOP(Syntax)

**FOR LOOP** continues to process a block of code until a statement becomes false, and everything is defined in a single line. The for loop is also called **entry-controlled loop**.

```
for ( init; condition; increment )
{
   statement(s);
}
```

# FOR LOOP(Program)

```c
#include<stdio.h>

int main ()
{
  /* local variable Initialization */  int n,times=5;;

   /* for loops execution */   for( n = 1; n <= times; n = n + 1 )
   {
      printf("C for loops: %d\n", n);
   }

   return 0;
}
```

# DO WHILE LOOP(Syntax)

**DO WHILE LOOP** always executes the code block at least once and furthermore as long as the condition remains true. This is an **exit-controlled loop**.

```
do
{
   statement(s);

}while( condition );
```

# DO WHILE LOOP(Program)

```c
#include<stdio.h>

int main ()
{
   /* local variable Initialization */   int n = 1,times=5;

   /* do loops execution */   do
   {
       printf("C do while loops: %d\n", n);
       n = n + 1;
   }while( n <= times );

   return 0;
}
```

# WHILE LOOP(Syntax)

**WHILE LOOP** has one control condition, and executes as long the condition is true. The condition of the loop is tested before the body of the loop is executed, hence it is called an **entry-controlled loop**.

```
While (condition)
{
   statement(s);
   Incrementation;
}
```

# WHILE LOOP(Program)

```c
#include<stdio.h>

int main ()
{
   /* local variable Initialization */   int n = 1,times=5;

   /* while loops execution */   while( n <= times )
   {
      printf("C while loops: %d\n", n);
      n++;
   }

   return 0;
}
```