**UNIT-4:** Iterative statements :
4.1 Use of goto statement for iteration
4.2 while loop
4.3 do..while loop
4.4 for loop
4.5 Nested while, do..while and for loops
4.6 Jumping statement: (break and continue)

**UNIT-4:** Iterative statements:

4.1 Use of goto statement for iteration

A **goto** statement in C programming provides an unconditional jump from the 'goto' to a labelled statement in the same function. It used to jump forward and backword.

Syntax

| Forward jump | Back word jump |
|---|---|
| goto lable;<br><br>lable:<br>statements; | lable:<br>statements;<br><br>goto lable; |

```
// example of forward jump
#include <stdio.h>
#include <conio.h>

int main()
{
  int a=10;
  clrscr();
  goto x;
  printf(" %d ",a);

  x:
  printf("%d",a+10);

  getch();
  return 0;
 }

// example of backward jump
#include <stdio.h>
#include <conio.h>

int main()
{
  int i=1;
```

```
 clrscr();
 x:
 if(i<=10)
 {
  printf("\n%d",i);
  i=i+1;
  goto x;
 }

 getch();
 return 0;
}
```

4.2 while loop

**Syntax:** While (test condition)

```
        {
            Body of the loop
         }
```

- The while loop is an **entry controlled loop** statement. The control conditions tested before the start of the loop execution. If the conditions are not satisfied then the body of the loop does not executed.
- The test condition is checked and if it the condition is true then body of the loop executed. After execution of the body, the test condition is once again check and if it is true, the body has executed once again. This process of repeated execution of the body continues until the test condition finally becomes false and the control has transferred out of the loop.
- The body of the loop may have one or more statements. The braces has needed only if the body contains two or more statements.
- On exit, the program continues with statement immediately after the body of the loop.

- //sum of n number

```
#include <stdio.h>
#include <conio.h>
void main()
{
        int n,s=0,i;
        clrscr();
        printf("\n enter number:");
        scanf("%d",&n);
        i=1;
       while(i<=n)
       {
              s=s+i;
              i=i+1;
       }
  printf("\nsum is=%d",s);
  getch();
}
```

Output:

Enter number:5
Sum is=15

4.3 do..while loop

**Syntax:** do

```
{
    Body of the loop
} while (test condition);
```

- It first executes body of the loop. At the end of the loop, the test-condition has evaluated. If the condition is true, the program continues to execute the body of the loop once again. This process continues as long as the condition is true.
- When the condition becomes false the loop will be terminated and the control goes to the statement that appears immediately after the while statement.
- Since the test condition has evaluated at the bottom of the loop, the do…. while construct provides an **exit-controlled loop** and therefore the body of the loop always executed at least once.
- //sum of n number

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int n,s=0,i;
    clrscr();
    printf("\n enter number:");
    scanf("%d",&n);
     i=1;
    do
    {
            s=s+i;
            i=i+1;
     }while(i<=n);

    printf("\nsum is=%d",s);

  getch();
}
```

Output:
Enter number:5
Sum is=15

4.4 For loop

The for loop is an **entry controlled loop**.

**Syntax:** for (initialization; test-condition; increment/decrement)

```
{
```

```
        Body of the loop
    }
```

- Initialization of the control variable done first, using assignment statement. This statement executed only once.
- The value of the control variable tested using the test-condition. If the condition is true the body of the loop is executed otherwise, the loop is terminated and the execution continues with the statement that immediately follows the loop.
- When the body of the loop executed, the control transferred back to for statement. Now control variable is incremented /decremented using an assignment statements and the new value of the control variable again tested to see whether it satisfies the loop condition. If the condition satisfied, the body of the loop again executed. This process continues until the condition is true.
- The braces are optional when the body of the loop contains only one statement.
- Initialization and increment section may have more than one part.
- E.g for(n=1,m=15;n<=m; n=n+1,m=m+1)
- One or more section omitted if necessary.
- //sum of n number

```
#include <stdio.h>
#include <conio.h>
void main()
{
        int n,s=0,i;
         clrscr();
         printf("\n enter number:");
         scanf("%d",&n);
         for(i=1;i<=n;i++)
         {
                s=s+i;
         }
         printf("\nsum is=%d",s);

   getch();
}
```

Output:
Enter number:5
Sum is=15

4.5 Nested while, do..while and for loops

- Nesting of loops, that is one loop within another loop.
- The nesting may continue up to 15 levels.
- // example of for loop pattern

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int n,i,j,s;
    clrscr();
```

```
        printf("\n enter number:");
        scanf("%d",&n);
        for(i=1;i<=n;i++)
        {
                for(s=1;s<=n-i;s++)
                {
                        printf(" ");
                }
                for(j=1;j<=i;j++)
                {
                        printf(" %d ",j);
                }
                printf("\n");
        }
  getch();
}
```

Output:

```
    1
  1 2
1 2 3
```

4.6 Jumping statement: (break and continue)

**Jumping out of a loop (break statement)**

- An early exit from loop done by using **break** statement.
- When the break statement encountered inside a loop, the loop immediately exited and the program continues with the statement immediately following the loop.
- When loop are nested, the break would only exit from the loop containing it. That is the break will exit only a single loop.

- //check whether number is prime or not

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int n,i,p=0;
  clrscr();
  printf("\n enter number:");
  scanf("%d",&n);
  for(i=2;i<=n-1;i++)
  {
    if(n%i==0)
       p=1;
       break;
  }
  if(p==0)
   printf("\n number is prime number:");
  else
   printf("\n number is not prime number:");
  getch();
```

```
}
```

Output:
Enter number: 10
Number is not prime number

**Skipping a part of a loop (continue statement)**

- During the loop operations, it may be necessary to skip a part of the body of the loop under certain conditions. The continue statement cause the loop to be continued with the next iteration after skipping any statements in between.
- Syntax: continue
- // print odd number

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int x;
  clrscr();
  for(x=1;x<=10;x++)
  {
    if(x%2==0)
      continue;
      printf("\n %d",x);
  }
  getch();
}
```

Output:
1
3
5
7
9