Iteration (Repetition) statements:-

For repetition structure, C++ provides four types of repetition structure:-

1) while statement.
2) do/while statement.
3) for statement.
4) Nested for statement.

1- while statement (while loop):-

In C++, you can use a while loop to repeat actions. The format of the while loop is as follows:

while (*expression*) **or** **while (expression)**
{block of one or more C++ statements} ; **statement;**

As long as condition holds, the block of statements will be repeatedly executed. Otherwise, the loop is terminated.

1. If condition is true (nonzero), statement is executed and condition is evaluated again.
2. If condition is false continues, continues with statement (if many statement) else the loop is terminated (if only one statement).

Example: W.P in C++ to summation the numbers from 1 to 10?
Solution:-

```
#include <iostream.h>
int main ( )
{
int i=1;                          // initialization i
int sum=0;
while(i<=10)                      // repetition condition (i<=10)
{                                 // start body of while loop
sum=sum+i;
i++;                             // increment
}
cout <<"The sum is"<<sum<<endl;
return 0;}
```

Example: W.P in C++ that computes the sum of ten numbers input by the user. Use while loop?
Solution:-

```
#include <iostream.h>
int main ( )
{
```

```
int n, sum=0, i=1;
while(i<=10)
{
cout<< "enter an integer number:";
cin>> n;
sum+=n;
i++;
}
cout <<"The sum is"<<sum<<endl;
return 0;}
```

Example: W.P in C++ that computes the following equation:-

$$\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + \ldots + n^2$$

Solution:-

```
#include <iostream.h>
int main ( )
{
int i=1, n, sum;

cout<<"enter a positive:";

cin>> n;

while(i<=n){

Sum+=i*i;

i++;}

cout<<" the sum :"<< n << "squares is" << sum << endl;

return 0;}
```

2- do/while statement (do/while loop):-

It is similar to the while statement, except that its body is executed first and then the loop condition is examined. The general form of the do statement is:

```
do {                          or        do
    statement 1;                        statement;
} while (expression);                 while (expression);
    statement2;
```

1. **Statement 1 are executed.**
2. **Condition is evaluated.**
3. **If condition is true goes to 1.**
4. **If condition is false continues with statement 2.**

The do/while loop is similar in syntax and purpose to the while loop. The construct moves the test that continues condition of the loop to the end of the code block so that the code block is executed at least once before any evaluation.

Example: consecutive integer numbers 1+2+3+.....+n, use do/while loop?

Solution:-

```
#include <iostream.h>
int main ( )
{
int n,  i=1;
long sum=0;
cout<< "enter a positive integer:";
cin>> n;
do
sum+=i++;
while (i<= n);
cout <<"The sum of the first"<< n << "integers is "<<sum<<endl;
return 0;}
```

Example: W.P in C++ using do/while repetition structure to print the numbers from 1 to 10?

Solution

```
#include <iostream>
using std::cout;
using std::endl;
int main ( )
{
int counter=1;
do
cout<<counter<<" ";
while (++counter<=10);
cout<<endl;
return 0;
}
```

```
#include<iostream.h>
```

Example: W.P in C++ that computes the following equation:-

$$\boldsymbol{n! = (n)(n-1) \ldots \ldots (3)(2)(1)}$$

Solution:-

```
#include <iostream.h>
int main ( )
{
```

```
int f=1, n;

cout<<"enter a positive:";

cin>> n;

cout<< n <<"factorial is:";

do{

f*=n;

n--;

} while (n>1);

cout<< f << endl;

}

return 0;}
```

3- for statement (for loop):-

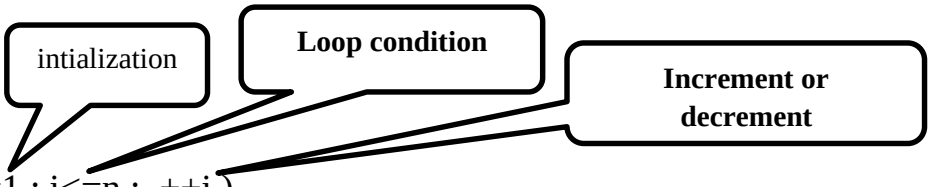
The for loop is designed to allow a counter variable that is initialized at the beginning of the loop and incremented (or decremented) on each iteration of the loop. The *for* loop works like the while loop but with some change in syntax:-

for (initialization; condition; iteration)

statement;

The for keyword is used as special case of a pre-conditional loop that supports constructors for repeating a loop only a certain number of times in the form of a step-expression that can be tested and used to set a step size (the rate of change) by incrementing or decrementing it in each loop.

Ex:




```
Sum=0;

for   ( i=1 ; i<=n ;  ++i )
Sum+=1;
```

Example: W.P in C++ that computes the following equation:-

$$\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + \ldots + n^2$$

Solution:-

```
#include <iostream.h>
int main ( )
{
int  n, sum=0;

cout<<"enter a positive:";

cin>> n;

for (int i=1; i<=n; i++)

sum+=i*i;

cout<<" the sum :"<< n << "squares is" << sum << endl;

return 0;}
```

Example: W.P in C++ using for structure to sum all the even integers from 2 to10?
Solution:-

```
#include <iostream.h>
int main ( )
{
int sum=0;
for (int number=2 ; number<=10; number+=2)
sum+=number;
cout <<"The sum of even number from 2 to 10 is"<<sum<<endl;
return 0;
}
```

Example:- W.P in C++ that prints the numbers from 1 to 20?

Solution:-

```
#include <iostream.h>
int main ( )
{
for (int i=2 ; i<=10; i++)
cout <<i<<endl;
return 0;}
```

Example: W.P in C++ that computes the following equation:-

$$n! = (n)(n-1) \ldots \ldots (3)(2)(1)$$

Solution:-

```
#include <iostream.h>
int main ( )
{
 int f=1, n;
cout<<"enter a positive:";
cin>> n;
for ( int i=2;i<=n;i++)
f*=n;
cout<< f << endl;
return 0;}
```

or

```
#include<iostream.h>
 int main( )
 {
  long f=1;
cout<<"enter a positive:";
 cin>> n;
for( int i=n; i>1; i--)
 f*=i;
cout<< f << endl;
 return 0;}
```

Nested for statement :-

the general format of the **for** structure:-

for (initialization; condition; iteration)

{ Block of one or more C++ statements;}

Example :-

If we have the following for structures:

```
for (int i=0;l<=10;i++)
for (int j=0; j<=5; j++)
cout<<i<<endl;
```

In this case the **first for** structure starts with initial value of i=0 and then test its condition if it is true, we will go to **the second for** structure. Here **the second** for takes the initial value of j=0 and test its condition j<=5, if it is true the statement; cout <<i<<endl; is executed. This process continue up the **second for** structure finish the 5 iterations (i.e, its condition j<=5 becomes false).

At this moment return to the **first for** structure and increment i by 1 and again test its condition (i<=0), if it is true go to the **second for** structure again, and this **for** repeats the process up the condition j<=5 become false then return to **first for** loop again and so on up

the condition in the **first for** becomes false. In this step, the **two for** structures are terminated.

Example:- The output from this program follows: The inside loop repeats five times (as outer counts down from 5 to 1) and prints from five numbers to one numbers?

Solution:-

```
#include <iostream.h>
int main ( )
{
int outer, inner;
for (outer=5;outer>=1;outer--)
{
for (inner=1;inner<=outer; inner++)
{
cout <<inner;
}
cout <<endl;
}
return 0;
}
```

Jump Statements:-

C++ has three statements that perform an unconditional branch: **break**, **goto**, and **continue**. Of these, you can use **goto** anywhere inside a function. You can use the **break** and **continue** statements in conjunction with any of the loop statements.

1- break statement:-

The **break** statement has two uses. You can use it to terminate a **case** in the **switch** statement (covered in the section on the **switch**, earlier in this chapter). You can also use it to force immediate termination of a loop, bypassing the normal loop conditional test. This use is examined here.

Example:-

```
# include <iostream.h>

int main( )

{

for( int i=1; i<=10; i++)

{

if( i==5)

break;
```

```
cout<< i<< " ";

}

cout<< "\n broke out of loop at i=" << i << endl;

return 0;}
```

Example:- what will be the output of the following program?
Solution:-

```
#include <iostream.h>
int main ( )
{
int y;
for (y=1; y<=11;y++)
{
if (y==6)
break;
cout <<y<<" ";
}
cout <<"\nBroke out of loop at y equal "<<y<<endl;
return 0;}
```

2- continue statement:-

The continue statement is used to skip the remaining statements in the body of the loop and then continue with the next iteration of the loop.

Example:-

```
# include <iostream.h>

int main( )

{

for( int i=1; i<=10; i++)

{

if( i==5)

continue;

cout<< i<< " ";

}
```

```
cout<< "\n continue out of loop at i=" << i << endl;

return 0;}
```

3- goto statement:-

The **goto** statement requires a label for operation. (A *label* is a valid identifier followed by a colon.) Furthermore, the label must be in the same function as the **goto** that uses it- you cannot jump between functions.
One good use for the **goto** is to exit from several layers of nesting. For example:-

```
for(...) {
for(...) {
while(...) {
if(...) goto stop;
.
.
.
}
}
}
stop:
cout<< "error in program\n")<< endl;
```

Example:

```
for (i=0; i < attempts; ++i)
{
cout << "please enter your password";
cin >> password;
if (verify(password)) // check password for correctness
goto out; // drop out of the loop
cout << "incorrect\n";
}
out:
// etc....
```

Example:- W.P in C++ to read seven student marks then prints the number of pass and fail student.
Solution:-

```
#include <iostream.h>
int main()
{
int i=0,x,p=0,f=0;
```

```
cout <<"Input seven mark"<<endl;
again: cin >>x;
if (x>=50)
p+=1;
else f+=1;
i++;
if (i<7) goto again;
cout <<"The number of pass mark is "<<p<<endl;
cout <<"The number of fail mark is "<<f<<endl;
return 0;
}
```

One of the most interesting uses of the **for** loop is to create an ***infinite loop***. Since none of the three expressions that form the **for** loop are required, you can make an endless loop by leaving the conditional expression empty, as here:

```
for(;;) statement;
```

Example: Write a program in C++ language to print the message ”you are in outside of loop” when you enter the right password, otherwise print “you are inside the loop” take one character as the password.

Solution:

```
#include<iostream.h>
int main()
{
char f;
for(;;){
cout<<" Enter The Password:";
cin>>f;
if('D'==f)break;
cout<<"you are in loop:\n";
}
cout<<"you are in outside of loop";
return 0;}
```

H.W:-

1- w.p that computes the sum of integer numbers divisible by 6 that are from 20 to 100?

2- w.p that computes the following:-

a- $y = \sum_{k=1}^{b} \frac{x^k}{k!}$

b- $y = 1 + \frac{1}{x} + \frac{2}{x^2} + \cdots + \frac{n}{x^n}$