A C++ program is given below to display number 1, 2, 3, 4 and 6 on the screen.

```
#include<iostream.h>
void main()
{
   cout<<1<<endl;
   cout<<2<<endl;
   cout<<3<<endl;
   cout<<4<<endl;
   cout<<5<<endl;
   cout<<6<<endl;
}
```

A different program is given below which displays numbers 1, 2, 3, 4, 5 and 6 on the screen.

```
#include<iostream.h>
void main()
{
   int k=1;
   cout<<k<<endl;
   k++;
   cout<<k<<endl;
   k++;
   cout<<k<<endl;
   k++;
   cout<<k<<endl;
   k++;
   cout<<k<<endl;
   k++;
   cout<<k<<endl;
   k++;
}
```

But if I want to display number 1, 2, 3, …, 99, 100 on the screen then the above programs will be neither suitable nor practical to produce desired output. So there has to a better way or an elegant way of producing same output on the screen and this is possible by using loop in C++. In the second program, pair of statements `cout<<k<<endl;` and `k++;` are getting repeated 5 times. Using a loop in C++ it is possible to repeat statement or block. C++ supports three (3) types of loops: **`while`** loop, **`for`** loop and **`do-while`** loop. Generally any loop in C++ has three (3) components:

a) **Control Variable and its Initialisation**: Generally a loop in C++ has a control variable. Control variable is generally an integer type or character type or floating point type. But we can have loop in C++ without a control variable. We will discuss such loops later. If a loop has a control variable then the control variable has to be initialized.

b) **Terminating Condition**: Generally a loop should terminate after certain number of iterations. Terminating condition of the loop in C++ is a condition (logical expression) involving the Control Variable. Condition or logical expression plays a very important in the working of a loop in C++ since number of times loop is repeated depends on the condition.

c) **Updation of Control Variable**: Every C++ loop repeats statement or block. Inside the block or statement of a loop, control variable is updated (value of the control variable is either incremented or decremented), so that the loop terminates after certain number of iterations.

- **while loop**

Rule: **while** (Condition)
Block / Statement;

The Condition is a logical expression involving control variable. First the Condition is evaluated; if the Condition is **TRUE** (nonzero), the Block / Statement is executed. Inside the Block / Statement, control variable is updated and Condition is tested once again. These three steps are repeated till Condition is **FALSE**. When the condition is evaluated to be false, the loop is terminates.

Usage of **while** loop:

```
#include<iostream.h>
void main()
{
   int k=1;
   while (k<=8)
   {
      cout<<k<<endl;
      k++;
   }
}
```

Running of the program produces following output:

```
1
2
3
4
5
6
7
8
```

Explanation of output and working of the program:
Initial value of k=1

k<=8	cout<<k<<endl;	k++
TRUE	1	2
TRUE	2	3
TRUE	3	4
TRUE	4	5
TRUE	5	6
TRUE	6	7
TRUE	7	8
TRUE	8	9
FALSE	9	

The **while** loop is an example of entry controlled loop, since first the Condition is tested and then the Block or the Statement is executed.

Use of **while** loop

```
#include<iostream.h>
void main()
{
   int n;
   cout<<"Input a positive integer? ";
   cin>>n;
   int k=2;
   while (k<=n)
   {
      cout<<k<<endl;
      k+=2;
   }
}
```

Running of the program

```
Input a positive integer? 20
2
4
6
8
10
12
14
16
18
20
```

Explanation of output and working of the output

Inputted value of `n` is `20` and initial value of `k=2`.

k<=20	cout<<k<<endl;	k+=2
TRUE	2	4
TRUE	4	6
TRUE	6	8
TRUE	8	10
TRUE	10	12
TRUE	12	14
TRUE	14	16
TRUE	16	18
TRUE	18	20
TRUE	20	22
FALSE		

Running of the program

```
Input a positive integer? 9
2
4
6
8
```

Explanation of output and working of the output
Inputted value of `n` is `9` and initial value of `k=2`.

k<=9	cout<<k<<endl;	k+=2
TRUE	2	4
TRUE	4	6
TRUE	6	8
TRUE	8	10
FALSE		

Running of the program

```
Input a positive integer? -5
```

Explanation of output
Inputted value of `n` is `-5` and initial value of `k=2`. Therefore condition is **FALSE** and the loop is not executed at all and hence no output.

- **`for` loop**

The **`for`** loop is almost similar to **`while`** loop but it is more compact than **`while`** loop. The while loop has three important components: an initialization of control variable, a terminating condition and update of control variable. In a **`for`** loop all three components are together immediately after the keyword **`for`**. A **`for`** loop is also an example of entry controlled loop.

Rule:
```
for (Initialisation; Condition; Update)
        Block / Statement;
```

Usage of **`for`** loop

```
#include<iostream.h>
void main()
{
   int n, sum=0;
   cout<<"Input positive integer? "; cin>>n;
   int x=1;              //Initialisation of control variable
   while (x<=n)          //Terminating Condition
   {
      sum+=x*x;
      x++;               //Update of control variable
   }
   cout<<"Sum="<<sum<<endl;
}
```

```
#include<iostream.h>
void main()
{
   int n, sum=0;
   cout<<"Input positive integer? "; cin>>n;
   for (int k=1; k<=n; k++)    //all together
      sum+=k*k;
   cout<<"Sum="<<sum<<endl;
}
```

Running of the program

```
Input positive integer? 7
Sum=140
```

Explanation of output and working of the output
Inputted value in `n` is `10` and initial value of `k=1`

k<=10	k	k*k	sum+=k*k	k++
TRUE	1	1	1	2
TRUE	2	4	5	3
TRUE	3	9	14	4
TRUE	4	16	30	5
TRUE	5	25	55	6
TRUE	6	36	81	7
TRUE	7	49	140	8
FALSE				

- **do-while loop**

The **do-while** loop is similar to **while** loop but only difference is **while** and the terminating condition is at the end. That loop is executed first and then the condition is tested. The **do-while** loop is called exit control loop. Exit control loop is executed at least once.

Rule:

```
do
{
   //C++ Statements
}
while (Condition);
```

The do starts the beginning of the loop. After that the `Block` or the `Statement` is executed. Next, the control variable is updated and the terminating condition is tested.

Usage of **do-while**

```
#include<iostream.h>
void main()
{
   int n, k=1, fact=1;
   cout<<"Input positive integer? "; cin>>n;
   do
   {
      fact*=k;
      k++;
   }
   while (k<=n);
   cout<<"Factorial="<<fact<<endl;
}
```

Running of the program

```
Input positive integer? 10
Factorial=3628800
```

Explanation of output and working of the output
Inputted value in n is 10 and initial value of k=1

k	fact*=k	k++	k<=10
1	1	2	**TRUE**
2	2	3	**TRUE**
3	6	4	**TRUE**
4	24	5	**TRUE**
5	120	6	**TRUE**
6	720	7	**TRUE**
7	5040	8	**TRUE**
8	40320	9	**TRUE**
9	362880	10	**TRUE**
10	3628800	11	**FALSE**

```
//Program to find HCF and LCM of two integers
#include<iostream.h>
void main()
{
   int a, b;
   cout<<"Input 1st integer? ";cin>>a;
   cout<<"Input 2nd integer? ";cin>>b;
   int prod=a*b, r;
   do
   {
      r=a%b;
      a=b;
      b=r;
   }
   while(r>0);
   cout<<"HCF="<<a<<endl;
   cout<<"LCM="<<(prod/a)<<endl;
}

//Count digits, sum of digits, product of digits
#include<iostream.h>
void main()
{
   int n, digit=0, sum=0, prod=1;
   cout<<"Input an integer? ";cin>>n;
   while (n!=0)
   {
      int r=n%10;
      digit++;
      sum+=r;
      prod*=r;
      n/=10;
   }
   cout<<"Number of digits="<<digit<<endl;
   cout<<"Sum of digits="<<sum<<endl;
   cout<<"Product of digits="<<prod<<endl;
}
```

```
//Program to reverse an inputted integer
#include<iostream.h>
void main()
{
   int n, m=0;
   cout<<"Input an integer? ";cin>>n;
   while (n!=0)
   {
      m=10*m+n%10;
      n/=10;
   }
   cout<<"Reversed integer="<<m<<endl;
}

//Check for Prime Number
#include<iostream.h>
void main()
{
   int n;
   cout<<"Input an integer? ";cin>>n;
   int k=2, prime=1;
   while (k<n && prime==1)
   {
      if (n%k==0)
         prime=0;
      k++;
   }
   if (prime==1)
      cout<<n<<" Prime Number"<<endl;
   else
      cout<<n<<" Composite Number"<<endl;
}

//Check for Armstrong Number
#include<iostream.h>
void main()
{
   int n;
   cout<<"Input an integer? ";cin>>n;
   int t=n, s=0;
   while (n!=0)
   {
      int digit=n%10;
      s+=digit*digit*digit;
      n/=10;
   }
   if (s==t)
      cout<<t<<" Armstrong Number"<<endl;
   else
      cout<<t<<" Not Armstrong Number"<<endl;
}
```

```
//Check for Palindromic Integer
#include<iostream.h>
void main()
{
   int n;
   cout<<"Input an integer? ";cin>>n;
   int t=n, m=0;
   while (n!=0)
   {
      m=10*m+n%10;
      n/=10;
   }
   if (t==m)
      cout<<t<<" Palindromic integer"<<endl;
   else
      cout<<t<<" Not Palindromic integer"<<endl;
}
```

- **Nested loop**

A loop may contain another loop in its block or statement. This form of a loop is called nested loop. But in a nested loop, the inner loop must terminate before the outer loop. An example is given below:

```
#include<iostream.h>
void main()
{
   for (int k=1; k<=4; k++)          //Outer loop
   {
      for (int j=1; j<=k; j++)       //Inner loop
         cout<<'*';
      cout<<endl;
   }
}
```

Running of the program

```
*
**
***
****
```

Explanation of the output

The block of outer loop (**for** k-loop) contains inner loop (**for** j-loop). Outer loop is executed 4 times. Therefore the block with outer loop is also executed 4 times. The block of outer loop contains inner loop. The table given below explains execution of inner loop.

k	Iteration of Inner loop	Output
1	1	*
2	2	**
3	3	***
4	4	****

```
#include<iostream.h>   //Same program using while loop
void main()
{
   int k=1;
   while (k<=4)        //Start of Outer loop
   {
      int j=1;
      while (j<=k)     //Start of Inner loop
      {
         cout<<'*';
         j++;
      }               //End of Inner loop
      cout<<endl;
      k++;
   }                  //End of Outer loop
}

#include<iostream.h>   //Same program using do-while loop
void main()
{
   int k=1;
   do                             //Start of Outer loop
   {
      int j=1;
      do               //Start of Inner loop
      {
         cout<<'*';
         j++;
      }
      while (j<=k);    //End of Inner loop
      cout<<endl;
      k++;
   }
   while (k<=4);                  //End of Outer loop
}
```

- **Infinite loop**

  A loop that never terminates is called an infinite loop. Kind of program we will do using loops, will involve only finite loop (loop that terminates) but there are no harm in learning about infinite loop. Infinite loop can be implemented with **while** or **for** or **do-while** loop.

  An important point to remember is that in C++ any non-zero value is considered to be **TRUE** and zero (0) value is **FALSE**.

  Rule:
  ```
  while (NonZeroValue)        //Infinite while loop
     Block / Statement;

  for (; ;)
     Block / Statement;       //Infinite for loop
  ```

```
do
{                              //Infinite do-while loop
   //C++ Statements
}
while (NonZeroValue);
```

Examples of infinite loops are given below:

a) Using **while** loop:

```
#include<iostream.h>
void main()
{
   while (1)
      cout<<"*";
}
```

The program displays stars (*) on the screen infinitely. To terminate the program and the loop click the Close Icon of the DOS Window.

b) Using **do-while** loop:

```
#include<iostream.h>
void main()
{
   do
   {
      cout<<"*";
   }
   while (1);
}
```

c) Using **for** loop:

```
#include<iostream.h>
void main()
{
   for ( ; ; )
      cout<<"*";
}
```