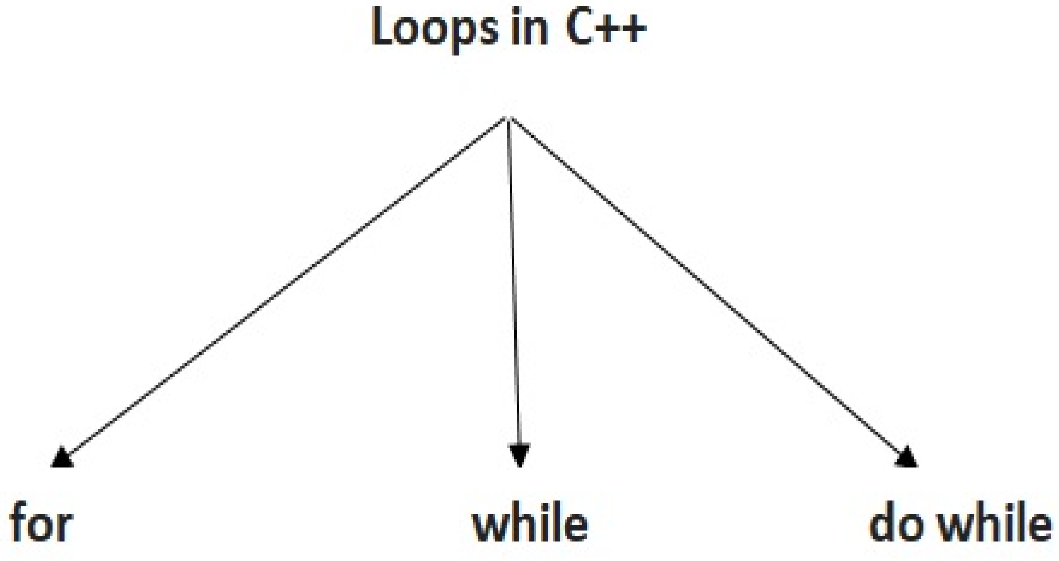
Department of Networks

First Year

Problem Solving and Programming 1

Loops in C++

- In Programming, sometimes there is a need to perform some operation **more than once** or (say) **n number** of times.
- Loops come into use when we need to repeatedly execute a block of statements.
- Loops can execute a block of code as long as a specified condition is reached.
- Loops need counter and stopping condition
- Loops can be achieved in C++ via three statements:



for Loop

- A *For* loop is a repetition control structure that allows us to write a loop that is executed a specific number of times.
- The loop enables us to perform n number of steps together in one line.
- When you know exactly how many times you want to loop through a block of code, use the **`for loop`** to repeat that block:

Syntax

```
for (initialization; condition; update)
{
  // code block to be executed
}
```

- **`initialization`** is executed (one time) before the execution of the code block.
- **`condition`** defines the condition for executing the code block. This statement is evaluated before each execution of the loop body, and aborts the execution if the given condition is false.

- **`update`** is executed (every time) after the code block has been executed.

for loop Examples

Example 1:

- The example below will print the numbers 0 to 4:

```
for (int i = 0; i < 5; i++)

{

  cout<<i<<endl;

}
```

Example 2:

- This example will only print even values between 0 and 10:

```
for (int i = 0; i <= 10; i += 2)

{

  cout<<i<<endl;

}
```

Example 3:

- This example will print the numbers from 10 to 1:

```
for (int i = 10; i > 0; i--)
```

```
{
  cout<< i <<endl;
}
```

Example 3

- This example will find and print the sum of the odd numbers between 1 and 10:

```
 int sum = 0;
for (int i = 1; i < 10; i += 2)
{
   sum = sum + i;
}
  cout<<sum;
```

Nested for loops

- Loops can be nested (i.e., a loop can include another loop inside)

```
for (initialization; condition; update)
{
    for(initialization; condition; update)
    {
           // inner loop statements.
    }
    // outer loop statements.
}
```

Example 1

- Write a C++ program to print the following pattern:

```
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
```

```
int n = 3;
for (int i = 1; i <= n; i++)  // outer loop
 {
  for (int j = 1; j <= 10; j++) // inner loop
   {
     cout<<(i * j)<<" ";
   }
  cout<<endl;
  }
```

Example 2

- Write a C++ program to print the following pattern:

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

```
int n = 4;
for (int i = 1; i <= n; i++)  // outer loop
  {
   for (int j = 1; j <= 8; j++)  // inner loop
     {
      cout<< "*“ ;
     }
      cout<<endl;
  }
```

Example 3

- Write a C++ program to print the following pattern:

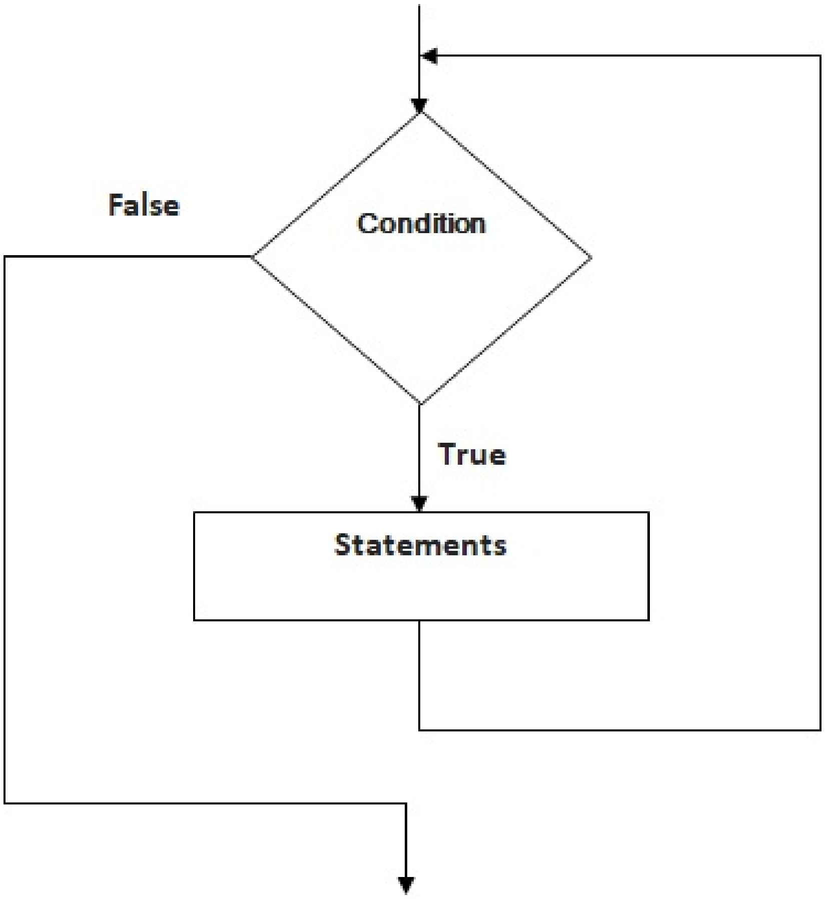
```
*
**
***
****
*****
```

```
int n = 5;
for (int i = 1; i <= n; i++)  // outer loop
 {
   for (int j = 1; j <= i; j++)  // inner loop
    {
     cout<<"*";
    }
   cout<<endl;
 }
```

while loop

- The `while` loop loops through a block of code as long as a specified condition is True
- `while` loops are used in situations where we do not know the exact number of iterations of the loop in advance.

- The loop execution is terminated on the basis of the test conditions.



Syntax

```
while (condition)

{

  // code block to be executed

}
```

- In the example below, the code in the loop will run, over and over again, as long as a variable (i) is less than 5:

Example 1

```
int i = 0;

while (i < 5)

{

  cout<<i<<endl;

  i++;

}
```

Example 2

- This C++ program asks to input unlimited number of integer numbers and finds the summation of odd numbers only. The program ends by entering -999.

```
int sum = 0;

int number;

cin>>number;

while (number != -999)

  {

   if (number % 2 != 0)

      {
          sum += number;
        }
     cin>>number;
    }
 cout<<sum;
```

Example 3

- This program prints the numbers from 10 to 19:

```
int a = 10;
while (a < 20)
  {
    cout<< a <<endl;
    a++;
  }
```

The do while Loop

- The do/while loop is a variant of the while loop.
- This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

- In a do-while loop, the loop body will **execute at least once** irrespective of the test condition.

Syntax

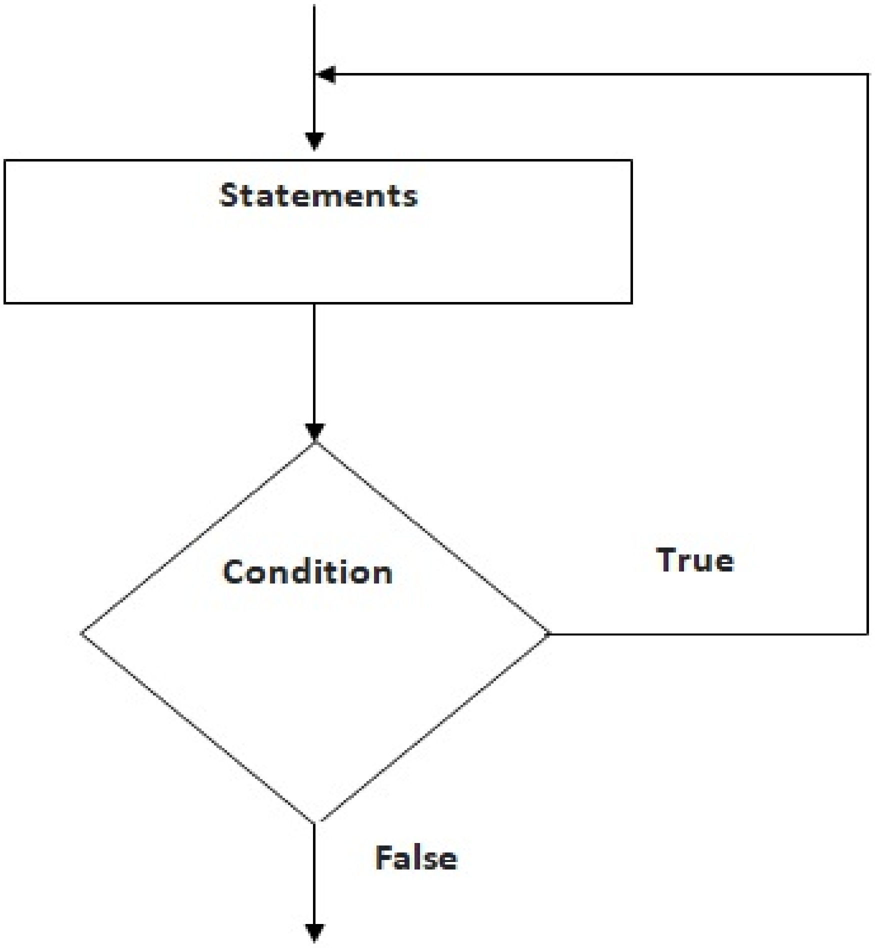
```
do

{

  // code block to be executed

}while (condition);
```

Do while structure



Examples

- The example below uses a do/while loop.

- The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:

Example 1

```
int i = 0;

do

{

  cout<<i<<endl;

  i++;

} while (i < 5);
```

- Note: Do not forget to increase the variable used in the condition, otherwise the loop will never end!

Example 2

- The example below uses a do/while loop to enter numbers and sum them and the program stops by entering 0

```
int number, sum = 0;
```

```
// the body of the loop is executed at
least once
do
 {
  cout<<"Enter a number: ";
  cin>>number;
  sum += number;
 }while (number != 0);
 cout<< "Sum = " << sum;
```

Examples on Loops in C++

1. Finding x^y using **for** statement.

```
int x, y;
cin>>x;
cin>>y;
int result = 1;
  for(int i=1; i<=y; i++)
    {
      result *= x;
    }
  cout<<result;
```

2. Finding the average of 7 marks using while statement

```
int sum = 0;
int mark; int i=1;
float average;
while (i <= 7)
{
  cin>>mark;
  sum += mark;
  i++;
}
average = sum / 7.0;
cout<<"Average = " << average;
```

3. Finding the maximum value from 100 random values (positive integers) entered from keyboard using do while statement.

```
int i = 1;
int Max = 0;
int num;
do{
    cin>>num;
    i++;
    if (num > Max)
      {
        Max = num;
      }
   } while (i <= 100);
cout<<“The Maximum is " << Max;
```

<u>**Homework**</u>**:**

Write C++ programs for the following problems:

1. Input 100 random numbers and count the odd and even numbers using `for` statement.
2. Input 100 random integer numbers (positive and negative) and sum the positive and negative numbers using `while` statement.
3. Output this series using `do while` statement:

   1 2 4 8 16 1024

4. Print this pattern using **`for`** statement:

```
1

1 2

1 2 3

1 2 3 4

1 2 3 4 5
```

5. Sum this series using **`while`** statement:

3 5 7 …. 99

6. Print this pattern using **`for`** statement:

```
*****

****

***

**

*
```

7. Calculate this series using **`while`** statement:

$$\boldsymbol{Y = \frac{1}{x^2} + \frac{2}{x^3} + \frac{3}{x^4} \dots \dots + \frac{n}{x^{n+1}}}$$

8. Print the times table from 3-8 only:

3 6 9 12 15 18 21 24 27 30

….

…

8 16 24 32 40 48 56 64 72 80

9. Enter an integer number and check whether it is **prime** or not using while statement.

10. Enter an integer number and find the **factors** of that number. (Example: input: 12, output: 1 2 3 4 6)