C++ while Loop

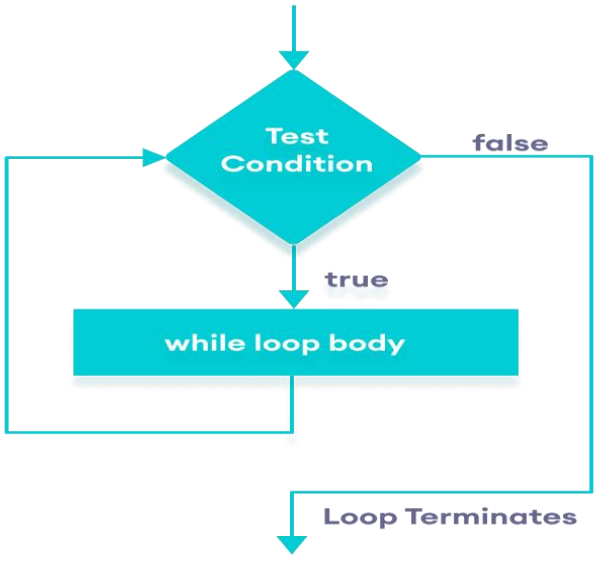
The syntax of the `while` loop is:

```
while (condition) {
  // body of the loop
}
```

Here,

- A `while` loop evaluates the `condition`
- If the `condition` evaluates to `true`, the code inside the `while` loop is executed.
- The `condition` is evaluated again.
- This process continues until the `condition` is `false`.
- When the `condition` evaluates to `false`, the loop terminates.

Flowchart of while Loop



Example 1: Display Numbers from 1 to 5

```
// C++ Program to print numbers from 1 to 5
#include <iostream>
using namespace std;
int main() {
   int i = 1;
   // while loop from 1 to 5
   while (i <= 5) {
      cout << i << "  ";
      ++i;
   }
   return 0;
}
```

Output

```
1 2 3 4 5
```

Here is how the program works.

Iteration	Variable	i <= 5	Action
1st	`i = 1`	`true`	`1` is printed and `i` is increased to `2`.
2nd	`i = 2`	`true`	`2` is printed and `i` is increased to `3`.
3rd	`i = 3`	`true`	`3` is printed and `i` is increased to `4`
4th	`i = 4`	`true`	`4` is printed and `i` is increased to `5`.
5th	`i = 5`	`true`	`5` is printed and `i` is increased to `6`.
6th	`i = 6`	`false`	The loop is terminated

Example 2: Sum of Positive Numbers Only

```
// program to find the sum of positive numbers
// if the user enters a negative number, the loop ends
// the negative number entered is not added to the sum
#include <iostream>
using namespace std;
int main() {
   int number;
   int sum = 0;
   // take input from the user
   cout << "Enter a number: ";
   cin >> number;
   while (number >= 0) {
      // add all positive numbers
      sum += number;
      // take input again if the number is positive
      cout << "Enter a number: ";
      cin >> number;
   }
   // display the sum
   cout << "\nThe sum is " << sum << endl;
   return 0;
}
```

Output

```
Enter a number: 6
Enter a number: 12
Enter a number: 7
```

```
Enter a number: 0

Enter a number: -2


The sum is 25
```

In this program, the user is prompted to enter a number, which is stored in the variable `number`.

In order to store the sum of the numbers, we declare a variable `sum` and initialize it to the value of `0`.

The `while` loop continues until the user enters a negative number. During each iteration, the number entered by the user is added to the `sum` variable.

When the user enters a negative number, the loop terminates. Finally, the total sum is displayed.

for vs while loops

A `for` loop is usually used when the number of iterations is known. For example,

```
// This loop is iterated 5 times
for (int i = 1; i <=5; ++i) {
  // body of the loop
}
```

Here, we know that the for-loop will be executed 5 times.

However, `while` loop is usually used when the number of

iterations is unknown. For example,

```
while (condition) {
   // body of the loop
}
```

C++ Program to Display Fibonacci Series

The Fibonacci sequence is a series where the next term is the sum of the previous two terms. The first two terms of the Fibonacci sequence is **0** followed by **1**.

```
The Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21
```

Example 1: Fibonacci Series up to n number of terms

```
#include <iostream>
using namespace std;
int main() {
   int n, t1 = 0, t2 = 1, nextTerm = 0;
   cout << "Enter the number of terms: ";
   cin >> n;
```

```
    cout << "Fibonacci Series: ";
   // Prints the first two terms.
   cout << t1 << ", ";
   cout << t2 << ", ";
    for (int i =3; i <= n; ++i) {
        nextTerm = t1 + t2;
        t1 = t2;
        t2 = nextTerm;
        cout << nextTerm << ", ";
    }
    return 0;
}
```

Output

```
Enter the number of terms: 10
Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,
```

Example 2: Program to Generate Fibonacci Sequence Up by using while statement

```
#include <iostream>
using namespace std;
int main() {
    int t1 = 0, t2 = 1, nextTerm = 0, n;
    cout << " Enter the number of terms: ";
    cin >> n;
```

```
  // displays the first two terms which is always 0 and 1
  cout << "Fibonacci Series: " << t1 << ", " << t2 << ", ";
  int  i = 3;
  while ( i  <= n) {
    nextTerm = t1 + t2;
     t1 = t2;
     t2 = nextTerm;
     cout << nextTerm << ", ";
     ++i;
  }
  return 0;
}
```

Output

```
Enter the number of terms: 10
Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,
```

C++ Program to Find GCD

The largest integer which can perfectly divide two integers is known as GCD of those two numbers.

For example, the GCD of **4** and **10** is **2** since it is the largest integer that can divide both **4** and **10**.

Example: 1. Find GCD using for loop

```
#include <iostream>

using namespace std;

int main() {

 int n1, n2, GCD;

 cout << "Enter two numbers: ";

 cin >> n1 >> n2;

 // swapping variables n1 and n2 if n2 is greater than n1.

 if ( n2 > n1) {

  int temp = n2;

  n2 = n1;

  n1 = temp;

 }


 for (int i = 1; i <=  n2; ++i) {

  if (n1 % i == 0 && n2 % i ==0) {

    GCD = i;

  }

 }


 cout << "GCD = " << GCD;

 return 0;

}
```

The logic of this program is simple.

In this program, the smaller integer between `n1` and `n2` is stored in `n2`. Then the loop is iterated from `i = 1` to `i <= n2` and in each iteration, the value of `i` is increased by 1.
If both numbers are divisible by `i` then, that number is stored in variable GCD. This process is repeated in each iteration. When the iteration is finished, GCD will be stored in variable `GCD`.

Example 2: Find GCD using while loop

```
#include <iostream>
using namespace std;
int main() {
 int n1, n2;
 cout << "Enter two numbers: ";
 cin >> n1 >> n2;

 while(n1 != n2) {
  if(n1 > n2)
    n1 -= n2;
  else
    n2 -= n1;
 }

 cout << "GCD = " << n1;
 return 0;
}
```

Output

```
Enter two numbers: 16

76

HCF = 4
```

In the above program, the smaller number is subtracted from the larger number and that number is stored in place of the larger number.

Here, `n1 -= n2` is the same as `n1 = n1 - n2`. Similarly, `n2 -= n1` is the same as `n2 = n2 - n1`.

This process is continued until the two numbers become equal which will be GCD.

Let us look at how this program works when `n1 = 16` and `n2 = 76`.

n1	n2	n1 > n2	n1 -= n2	n2 -= n1	n1 != n2
16	76	`false`	-	60	`true`
16	60	`false`	-	44	`true`
16	44	`false`	-	28	`true`
16	28	`false`	-	12	`true`
16	12	`true`	4	-	`true`
4	12	`false`	-	8	`true`
4	8	`False`	-	4	`false`

Here, the loop terminates when `n1 != n2` becomes `false`.

After the final iteration of the loop, `n1 = n2 = 4`. This is the value of the GCD since this is the greatest number that can divide both **16** and **76**.

H.W: write C++ program to find LCM. LCM of two integers `a` and `b` is the smallest positive integer that is divisible by both `a` and `b`.