LOOPS in C++

Loops are used when we want a particular piece of code to run multiple times. We use loops to execute the statement of codes repeatedly until a specific condition is satisfied. It eases the work of the programmer and also shortens the code length. Loops can execute a block of code as long as a specified condition is reached.

Advantages of Loops

Loops are handy because they **save time, reduce errors**, and they **make code more readable.**

Loops

For example, if we want to print numbers from 1 to 1000, and we don't use loops, we have to write 1000 different print statements for printing numbers from 1 to 1000. With the help of loops, we can write this code in 2 lines. We need to run the loop and give iteration conditions.

There are 3 types of loops **for loop**, **while loop** and **do-while loop**.

C++ for loop

A for loop is the repetition control structure that is generally used to write a code more efficiently which is supposed to be executed a specific number of times.

How for loop works

1. An initialization statement is executed only once at a beginning.
2. Then, test expression is evaluated.
3. If a test expression is false, then for loop is terminated. But if a test expression is true, codes inside the body of **for loop** is executed, and update expression is updated.
4. A test expression is evaluated, and this process repeats until the test expression is false.

Syntax of for loop

```
for(initialization; condition; increment/decrement)
{
   //statements
}
```

The **initialization step** is executed to initialize the program counter it done once only. The program counter is also known as loop control variables.

The next parameter is the **condition** where the condition is checked for which the loop is supposed to run.

After that, there is an **increment/decrement** counter which increases or decreases the counter once the statements are executed.

Example 1
Write a program to print from 1 to 6.

```
#include <iostream>
using namespace std;
int main(){
   for(int i=1; i<=6; i++){

      cout<<"Value of variable i is: "<<i<<endl;
   }
   return 0;
}
```

Output:

```
Value of variable i is: 1
Value of variable i is: 2
Value of variable i is: 3
Value of variable i is: 4
Value of variable i is: 5
Value of variable i is: 6
```

Example 2

Write a program to print all the even numbers from 1 to 20 using for loop.

```
#include<iostream>
using namespace std;
int main()
{
    cout<<"We will print even numbers from 1 to 20 using for
loop"<<endl;
    for(int i=2;i<=20;i=i+2)
    {
        cout<<"The even number is:"<<i<<endl;
    }
    return 0;

}
```

Output

```
We will print even numbers from 1 to 20 using for loop
The even number is:2
The even number is:4
The even number is:6
The even number is:8
The even number is:10
The even number is:12
The even number is:14
The even number is:16
The even number is:18
The even number is:20

--------------------------------
Process exited after 0.1643 seconds with return value 0
Press any key to continue . . .
```

Example 3: Display a text 5 times

```
// C++ Program to display a text 5 times

#include <iostream>

using namespace std;

int main() {
    for (int i = 1; i <= 5; ++i) {
```

```
        cout <<  "Hello World! " << endl;
    }
    return 0;
}
```

Output

```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

Example 4

Write a program to print all the odd numbers from 1 to 20 using for loop.

```
#include<iostream>
using namespace std;
int main()
{
    cout<<"We will print odd numbers from 1 to 20 using for
loop"<<endl;
    for(int i=1;i<=20;i=i+2)
    {
        cout<<"The odd number is:"<<i<<endl;
    }
    return 0;

}
```

Output

```
We will print odd numbers from 1 to 20 using for loop

The odd number is:1
The odd number is:3
The odd number is:5
The odd number is:7
The odd number is:9
The odd number is:11
The odd number is:13
The odd number is:15
The odd number is:17
The odd number is:19

--------------------------------
Process exited after 0.3402 seconds with return value 0
```

```
Press any key to continue . . .
```

Example 5

Write a program to print the number of cars from 10 to 0.

```
#include <iostream>
using namespace std;

int main( ) {
      for (int car = 10; car >= 0; car = car-2){
            cout<< car << endl;
      }
      return 0;
}
```

Output

```
10
8
6
4
2
0

--------------------------------
Process exited after 0.2496 seconds with return value 0
Press any key to continue . ..
```

while loop

Let's first look at the **syntax** of **while** loop.

```
while(condition)
{
   statement(s)
}
```

while loop checks whether the **condition** written in ( ) is true or not. If the condition is true, the statements written in the body of the while loop i.e., inside the braces { }

are executed. Then again the condition is checked, and if found true, again the statements in the body of the while loop are executed. This process continues until the condition becomes false.

Example

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

Output

```
1
2
3
4
5
6
7
8
9
10

--------------------------------
Process exited after 3.322 seconds with return value 0
```

Example

Write a program to print from 5 to 1.

```
#include<iostream>
using namespace std;
int main()
{
    int t=5;

    while(t>0)
    {
```

```
        cout<<t<<endl;
        t=t-1;
    }
    return(0);
}
```

Output

```
5
4
3
2
1

--------------------------------
Process exited after 0.8417 seconds with return value 0
Press any key to continue . . .
```

C++ do...while loop

do-while loop

The do-while loop iterates a section of the C++ program several times. In the do-while loop, test expression is added at the bottom of the loop. The loop body comes before the test expression. That's why the loop body must execute for once, even when test expression evaluates to false in the first test.

When to use a do-while loop

The do-while loop should be used when the number of iterations is not fixed, and the loop must execute for at least once. The C++ compiler executes the loop body first before evaluating the condition. That means the loop must return a result. This is the case even when the test condition evaluates to a false on the first evaluation. Since the loop body has already been executed, it must return the result.

Syntax

The basic syntax of C++ do while loop is as follows:

```
do{
//code
}while(condition);
```

Here,

- The body of the loop is executed at first. Then the condition is evaluated.
- If the condition evaluates to true, the body of the loop inside the do statement is executed again.
- The condition is evaluated once again.
- If the condition evaluates to true, the body of the loop inside the do statement is executed again.
- This process continues until the condition evaluates to false. Then the l

Example 1

Program to display numbers from 1 to 4

```
#include <iostream>
using namespace std;
int main() {
	// Local variable
	int x = 1;
	do {
		cout << "X is: " << x << endl;
		x = x + 1;
	} while (x < 5);
	return 0;
}
```

Output:

```
X is: 1
X is: 2
X is: 3
X is: 4

--------------------------------
Process exited after 0.6603 seconds with return value 0
Press any key to continue . . .
```

Example 2

Display Numbers from 1 to 10

```
// C++ Program to print numbers from 1 to 10

#include <iostream>

using namespace std;

int main() {
    int i = 1;

    // do...while loop from 1 to 10
    do {
        cout << i << " ";
        ++i;
    }
    while (i <= 10);

    return 0;
}
```

Output

```
1 2 3 4 5 6 7 8 9 10
--------------------------------
Process exited after 0.3074 seconds with return value 0
Press any key to continue . . .
```