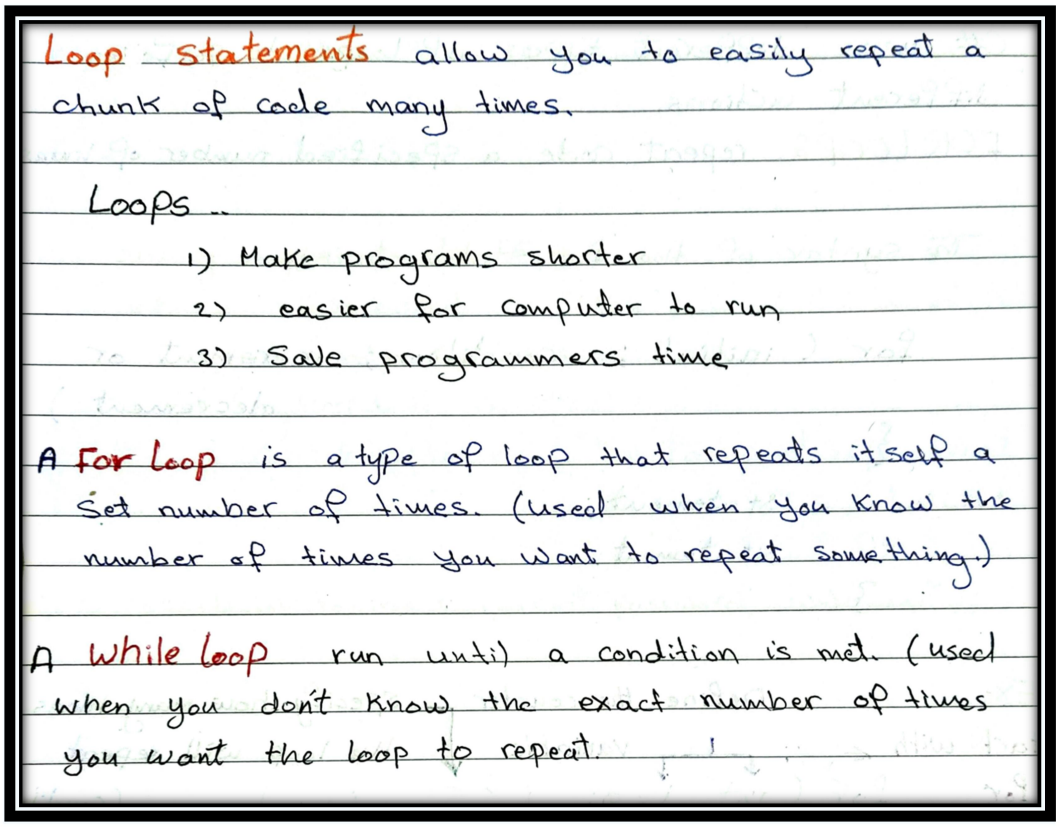
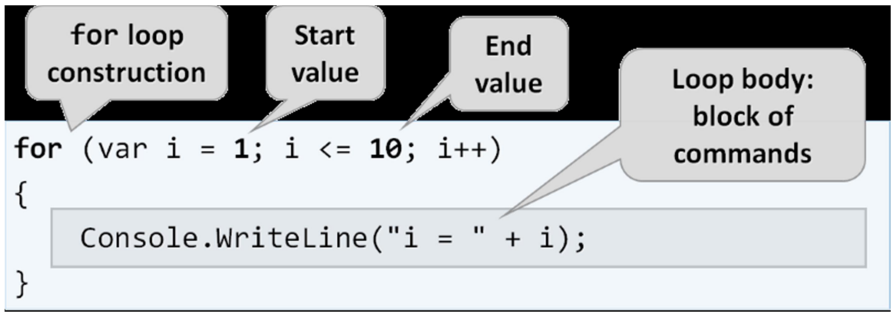
LOOPS

A **loop** is a logical structure used to repeatedly execute a block of code. This repetition can either iterate an exact number of times (definite) or continuously until a condition is met (indefinite).

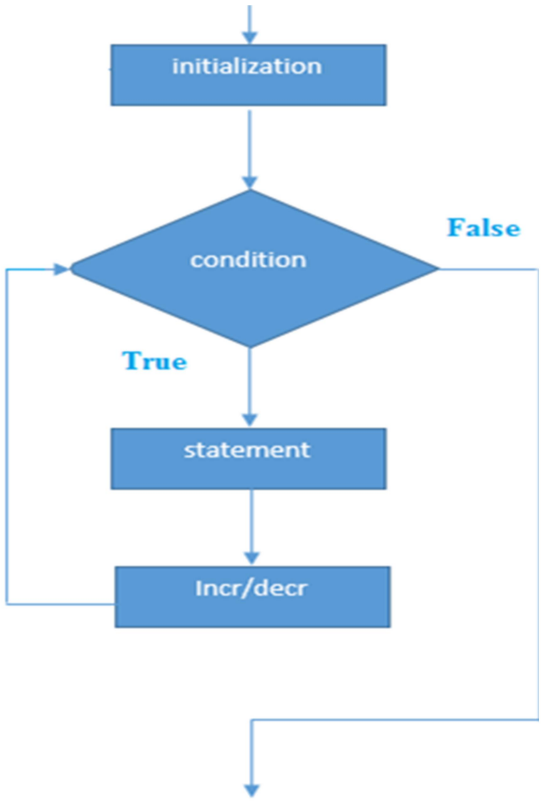


FOR LOOP

In C# the for loop uses definite repetition to run a block of code a specified number of times and consists of three statements separated by semicolons.




```
for(initialization; condition; incr/decr){
    //code to be executed
}
```




```
for (int i = 5; i > 0; i--)
{
  // Repeated code here
}
```

Write a program that inputs n integers and sums them up.

```
int n = int.Parse(Console.ReadLine());
int sum = 0;
for (int i = 0; i < n; i++)
{
    int num = int.Parse(Console.ReadLine());
    sum += num;
}
Console.WriteLine("Sum = {0}", sum);
```

Write a program that enters n integers (n > 0) and finds the max number among them (the largest).

```
Console.WriteLine("n = ");
int n = int.Parse(Console.ReadLine());
int max = -1000000000000000;

for (int i = 1; i <= n; i++)
{
    int num = int.Parse(Console.ReadLine());
    if (num > max)
    {
        max = num;
    }
}

Console.WriteLine("max = " + max);
```

Using Nested Loops

1. The body of the following loop executes **six** times:

```
for(a = 1; a < 4; ++a)
    for(b = 2; b < 3; ++b)
```

2. The body of the following loop executes **four** times:

```
for(c = 1; c < 3; ++c)
    for(d = 1; d < 3; ++d)
```

3. The body of the following loop executes **15** times:

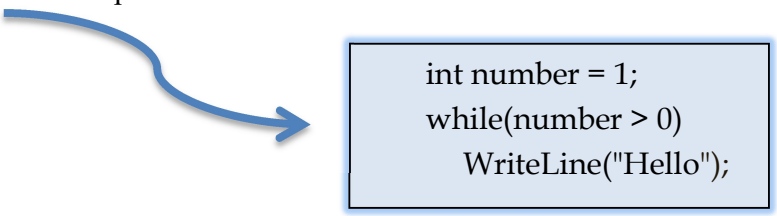
```
for(e = 1; e <= 5; ++e)
    for(f = 2; f <= 4; ++f)
```

WHILE LOOP

The while loop in C# executes an unspecified number of times until the given condition returns false. The condition is tested before each iteration of the loop. If the condition is false when it is tested the first time, the code block is never run.

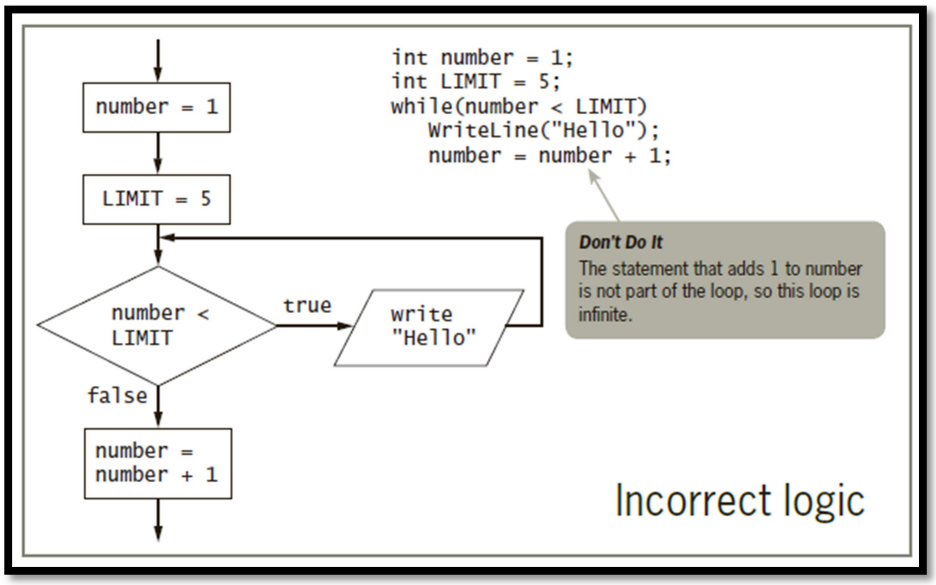
```
int i = 0;
while (i > -5)
{
  // Repeated code here
  i--;
}
```

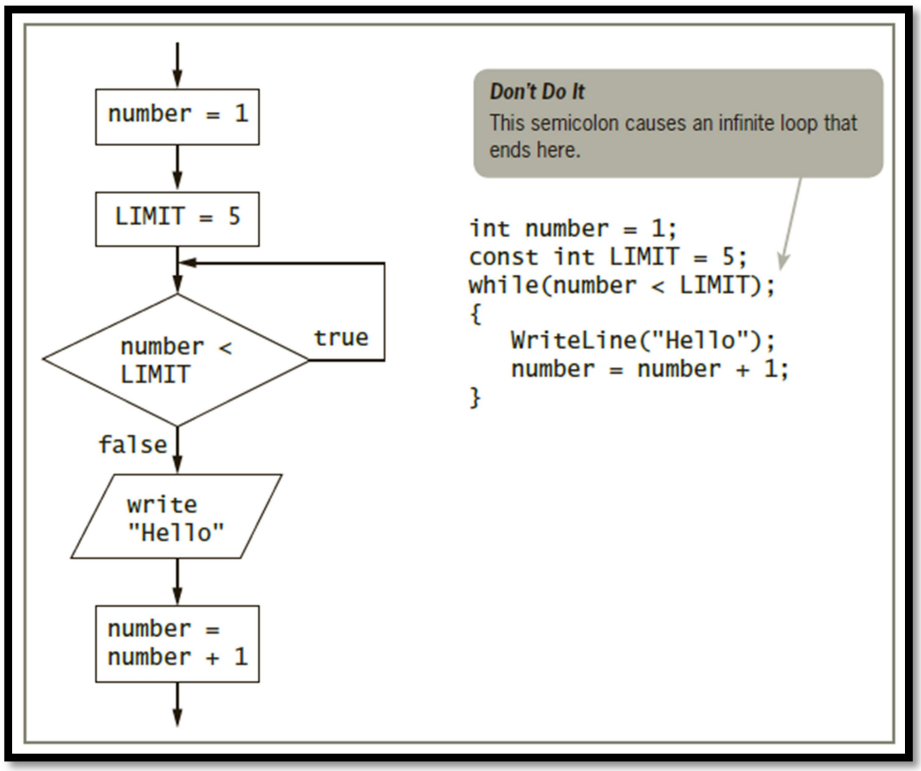
Note: The variable tested in the condition should be updated within the code block to avoid an **infinite** loop.



To make a while loop end **correctly**, three separate actions should occur:

- A variable, the loop control variable, is initialized (before entering the loop).
- The loop control variable is tested in the while expression.
- The body of the loop must take some action that alters the value.


number = 1
LIMIT = 5
number < LIMIT
true
write "Hello"
false
number = number + 1
int number = 1;
int LIMIT = 5;
while(number < LIMIT)
WriteLine("Hello");
number = number + 1;
Don't Do It
The statement that adds 1 to number is not part of the loop, so this loop is infinite.
Incorrect logic


number = 1
LIMIT = 5
number < LIMIT
true
false
write "Hello"
number = number + 1
Don't Do It
This semicolon causes an infinite loop that ends here.
int number = 1;
const int LIMIT = 5;
while(number < LIMIT);
{
WriteLine("Hello");
number = number + 1;
}

```
// Declare loop control variable and limit
int x;
int LIMIT = 10;

// Using a while loop to display 1 through 10
x = 1;
while(x <= LIMIT)
{
   WriteLine(x);
   ++x;
}

// Using a for loop to display 1 through 10
for(x = 1; x <= LIMIT; ++x)
   WriteLine(x);
```

DO WHILE LOOP

This form of loop uses the do keyword, followed by the code block, followed by the while keyword and condition. Unlike the while loop it checks the condition after the code block is executed. This means the loop will always iterate at least once, but the condition must be true for it to continue.

```
int i = 1;
do
{
  // Repeated code here
  i++;
} while (i <= 3);
```