**The Loop Control Structure**

**Looping Statements in C** execute the sequence of statements many times until the stated condition becomes false. A loop in C consists of two parts, a body of a loop and a control statement.

The control statement is a combination of some conditions that direct the body of the loop to execute until the specified condition becomes false. The purpose of the C loop is to repeat the same code a number of times.

**Types of Loops in C**

Depending upon the position of a control statement in a program, looping statement in C is classified into two types:

1. Entry controlled loop

2. Exit controlled loop

In an **entry control loop in C,** a condition is checked before executing the body of a loop. It is also called as a pre-checking loop.

In an **exit controlled loop**, a condition is checked after executing the body of a loop. It is also called as a post-checking loop.



The control conditions must be well defined and specified otherwise the loop will execute an infinite number of times. The loop that does not stop executing and processes the statements number of times is called as an **infinite loop**.

An infinite loop is also called as an “**Endless loop**.” Following are some characteristics of an infinite loop:

1. No termination condition is specified.

2. The specified conditions never meet.

The specified condition determines whether to execute the loop body or not.

‘C’ programming language provides us with three types of loop constructs:

1. The while loop

2. The do-while loop

3. The for loop

| **Sr. No.** | **Loop Type** | **Description** |
| --- | --- | --- |
| 1. | While Loop | In while loop, a condition is evaluated before processing a body of the loop. If a condition is true then and only then the body of a loop is executed. |
| 2. | Do-While Loop | In a do…while loop, the condition is always executed after the body of a loop. It is also called an exit-controlled loop. |
| 3. | For Loop | In a for loop, the initial value is performed only once, then the condition tests and compares the counter to a fixed value after each iteration, stopping the for loop when false is returned. |