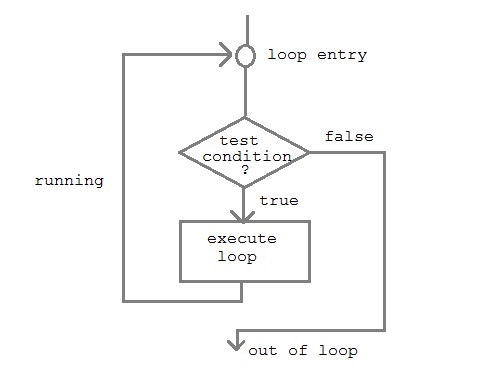
**Loop types**

In any programming language, loops are used to execute a set of statements repeatedly until a particular condition is satisfied.

**How it works**



A sequence of statement is executed until a specified condition is true. This sequence of statement to be executed is kept inside the curly braces { } known as loop body. After every execution of loop body, condition is checked, and if it is found to be **true** the loop body is executed again. When condition check comes out to be **false**, the loop body will not be executed.

**There are 3 type of loops in C++ language**

1. *while* loop
2. *for* loop
3. *do-while* loop

**while loop**

**while** loop can be address as an **entry control** loop. It is completed in 3 steps.

* Variable initialization.( e.g int x=0; )
* condition( e.g while( x<=10) )
* Variable increment or decrement ( x++ or x-- or x=x+2 )

**Syntax :**

variable initialization ;

while (condition)

{

statements ;

variable increment or decrement ;

}

**for loop**

**for** loop is used to execute a set of statement repeatedly until a particular condition is satisfied. we can say it an **open ended loop.** General format is,

for(**initialization**; **condition** ; **increment/decrement**)

{

statement-block;

}

In **for** loop we have exactly two semicolons, one after initialization and second after condition. In this loop we can have more than one initialization or increment/decrement, separated using comma operator. **for** loop can have only one **condition**.

**Nested for loop**

We can also have nested **for** loop, i.e one **for** loop inside another **for** loop. Basic syntax is,

for(**initialization**; **condition**; **increment/decrement**)

{

for(**initialization**; **condition**; **increment/decrement**)

{

statement ;

}

}

**do while loop**

In some situations it is necessary to execute body of the loop before testing the condition. Such situations can be handled with the help of **do-while** loop. **do** statement evaluates the body of the loop first and at the end, the condition is checked using **while** statement. General format of **do-while** loop is,

do

{

....

.....

}

while(condition);

**Jumping out of loop**

Sometimes, while executing a loop, it becomes necessary to skip a part of the loop or to leave the loop as soon as certain condition becocmes true, that is jump out of loop. C language allows jumping from one statement to another within a loop as well as jumping out of the loop.

**1) break statement**

When **break** statement is encountered inside a loop, the loop is immediately exited and the program continues with the statement immediately following the loop.

**2) continue statement**

It causes the control to go directly to the test-condition and then continue the loop process. On encountering continue, cursor leave the current cycle of loop, and starts with the next cycle.