Flow of Control

Looping statement

It is also called a Repetitive control structure. Sometimes we require a set of statements to be executed a number of times by changing the value of one or more variables each time to obtain a different result. This type of program execution is called looping. C++ provides the following construct

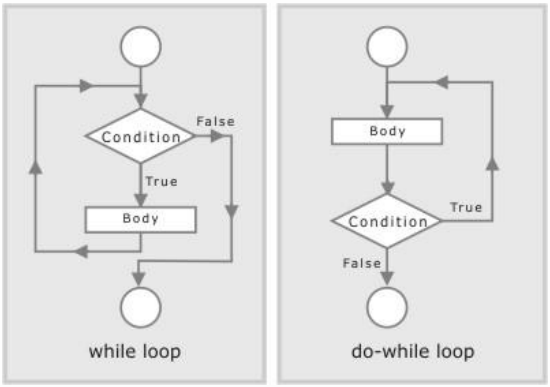
- while loop
- do-while loop
- for loop

While loop

Syntax of while loop

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

The flow diagram indicates that a condition is first evaluated. If the condition is true, the loop body is executed and the condition is re-evaluated. Hence, the loop body is executed repeatedly as long as the condition remains true. As soon as the condition becomes false, it comes out of the loop and goes to the statement next to the 'while' loop.

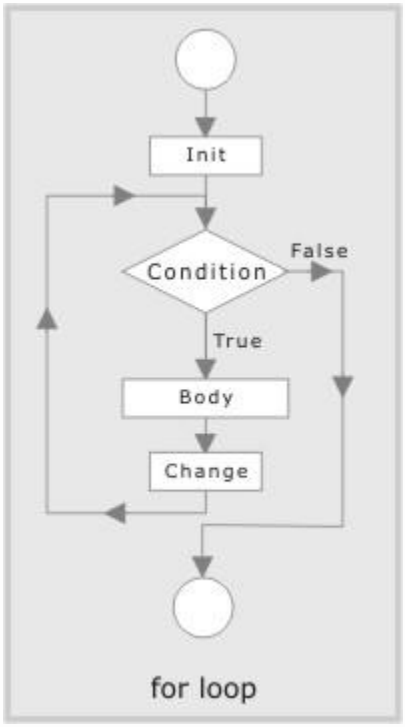


do-while loop

Syntax of do-while loop

```
do
{
  statements;
} while (condition);
```

Note : That the loop body is always executed at least once. One important difference between the while loop and the do-while loop the relative ordering of the conditional test and loop body execution. In the while loop, the loop repetition test is performed before each execution the loop body; the loop body is not executed at all if the initial test fail. In the do-while loop, the loop termination test is Performed after each execution of the loop body. hence, the loop body is always executed least once.



for loop

It is a count controlled loop in the sense that the program knows in advance how many times the loop is to be executed.

syntax of for loop

```
for (initialization; decision; increment/decrement)
{
   statement(s);
}
```

The flow diagram indicates that in for loop three operations take place:

- Initialization of loop control variable
- Testing of loop control variable
- Update the loop control variable either by incrementing or decrementing.

Operation (i) is used to initialize the value. On the other hand, operation (ii) is used to test whether the condition is true or false. If the condition is true, the program executes the body of the loop and then the value of loop control variable is updated. Again it checks the condition and so on. If the condition is false, it gets out of the loop.

Jump Statements

The jump statements unconditionally transfer program control within a function.

- goto statement
- break statement
- continue statement

The goto statement
goto allows to make jump to another point in the program.
`goto pqr;`
`pqr:` pqr is known as label. It is a user defined identifier. After the execution of goto statement, the control transfers to the line after label pqr.

The break statement
The break statement, when executed in a switch structure, provides an immediate
exit from the switch structure. Similarly, you can use the break statement in any of the loop. When the break statement executes in a loop, it immediately exits from the loop.

The continue statement

The continue statement is used in loops and causes a program to skip the rest of the body of the loop.

```
while (condition)
{
  Statement 1;
    If (condition)
      continue;
    statement;
}
```

The continue statement skips rest of the loop body and starts a new iteration.

The exit ( ) function

The execution of a program can be stopped at any point with exit ( ) and a status code can be informed to the calling program. The general format is

```
exit (code) ;
```

where code is an integer value. The code has a value 0 for correct execution. The value of the code varies depending upon the operating system.