CHAPTER – 6 DECISION MAKING AND LOOPING

In looping a sequence of statements are executed until some conditions for termination of the loop are satisfied.

A program loop consists of two segments, one known as the body of the loop and the other known as the control statement. The control statement tests certain conditions and then directs the repeated execution of the statements in the body of the loop.

Depending on the position of the control statement in the loop, a control structure is classified into one of the following types:

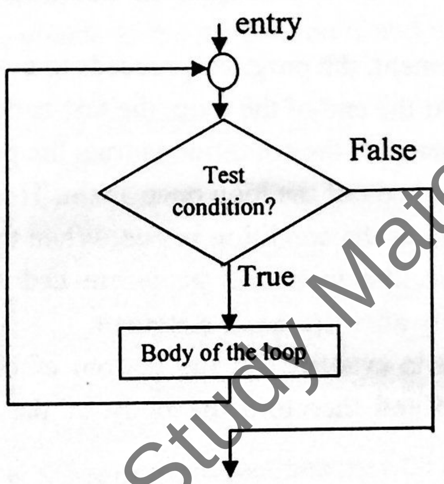
i) entry-controlled loop

ii) exit –controlled loop.

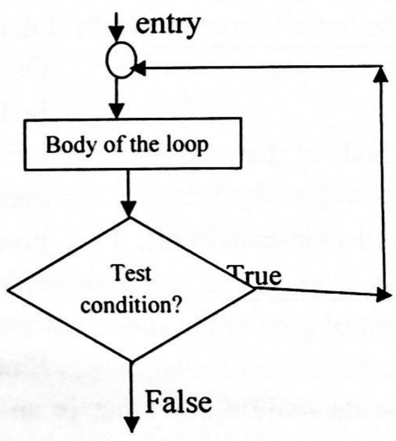
In the entry-controlled loop, the control statements are tested before the start of the loop execution. If the conditions are not satisfied, then the body of the loop will not be executed.

In an exit-controlled loop, the test is performed at the end of the body of the loop and therefore the body is executed unconditionally for the first time.

Flowcharts of loop control structures:



(entry-control)



(exit-control)

A looping process includes the following four steps:

i) Setting and initialization of a counter.

ii) Execution of the statements in the loop.

iii) Test for a specified condition for execution of the loop.

iv) Incrementing the counter.

C language provides for three loop constructs for performing loop operations. They are:

i) The while statement.

ii) The do statement.

iii) The for statement.

<u>The while statement</u>:

The while is an entry-controlled loop statement.

The basic format of the while statement is:

```
while(test condition)
{
body of the loop
}
```

The test condition is evaluated and if the condition is true, then the body of the loop is executed. After the execution of the body, the test-condition is once again evaluated and if it is true, the body is executed once again. This process of repeated Execution of the body continues until the test-condition finally becomes false and the control is transferred out of the loop. On exit, the program continues with the statement immediately after the body of the loop.

The body of the loop may have one or more statements. The braces are needed only if the body contains two or more statements. However it is a good practice to use braces even if the body has only one statement.

<u>The do statement</u>:

The basic format of the do statement is:

```
do
{
  body of the loop
}
while(test-condition);
```

On reaching the do statement, the program proceeds to evaluate body of the loop first. At the end of the loop, the test-condition while statement is evaluated. If the condition is true, the program continues to evaluate the body of the loop once again. This process continues as long as the condition is true. When the condition becomes false, the loop will be terminated and the control goes to the statement that appears immediately after the while statement.

Since the test-condition is evaluated at the bottom of the loop, the do---while construct is an exit-controlled loop and therefore the body of the loop is always executed at least once.

<u>The for statement</u>:

The for loop is an entry-controlled loop statement.

The general form of the for loop is:

```
for(initialization;test-conditon;increment)
{
   body of the loop
}
```

The execution of the for statement is as follows:

i) Initialization of the control variable is done first, using assignment statements. These variables are known as loop-control variables.

ii) The value of the control variable is tested using the test-condition. The test-condition is a relational expression, that determines when the loop will exit. If the condition is true, the body of the loop is executed; otherwise the loop is terminated and the execution continues with the statement that immediately follows the loop.

iii) When the body of the loop is executed, the control is transferred back to the for statement after evaluating the last statement in the loop. Now, the control variable is incremented using an assignment statement and the new value of the control variable is again tested to see whether it satisfies the loop condition. If the condition is satisfied, the body of the loop is again executed. This process continues till the value of the control variable fails to satisfy the test-condition.

Note:

i) The three sections enclosed within parentheses must be separated by semicolons.
ii) There is no semicolon at the end of the increment section.
iii) The for statements allows for negative increments.
iv) The braces are optional, when the body of the loop contains only one statement.
v) Since the conditional test is always performed at the beginning of the loop, the body of the loop may not be executed at all, if the condition fails at the start.

One of the important points about the for loop is that all the three actions namely initialization, testing and incrementing are placed in the for statement itself, thus making them visible to the programmers and users in one place.

Additional features of for loop:

- More than one variable can be initialized at a time in the for statement.
- The increment section may also have more than one part. The multiple arguments in the increment section are separated by commas.
- The test-condition may have any compound relation and the testing need not be limited only to the loop control variables.
- Expressions are permitted in the assignment statements of initialization and increment sections.
- One or more sections may be omitted.
  Consider the following statements:

```
-----------
-----------
m=5;
for(;m!=100;)
{
  -----------
  -----------
  -----------
  m=m+5;
 }
 ----------
 ----------
```

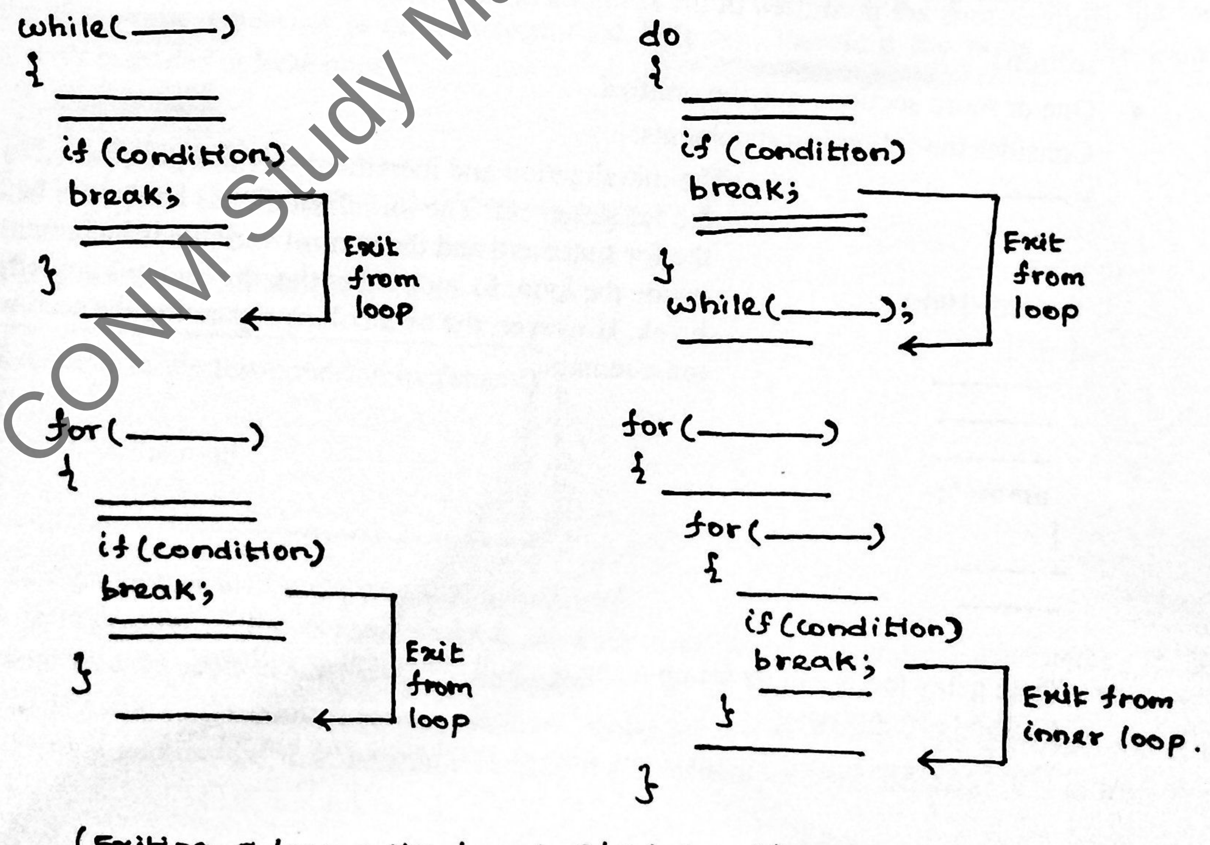
The initialization and increment sections are omitted in the for statement. The initialization has been done before the for statement and the control variable is incremented inside the loop. In such cases the, the sections are left blank. However, the semicolons separating the sections must remain.

- Time delay loops can be set up using the null statement as follows:
  for(i=0;i<10000;i++);
  This loop is executed 10000 times without producing any output.

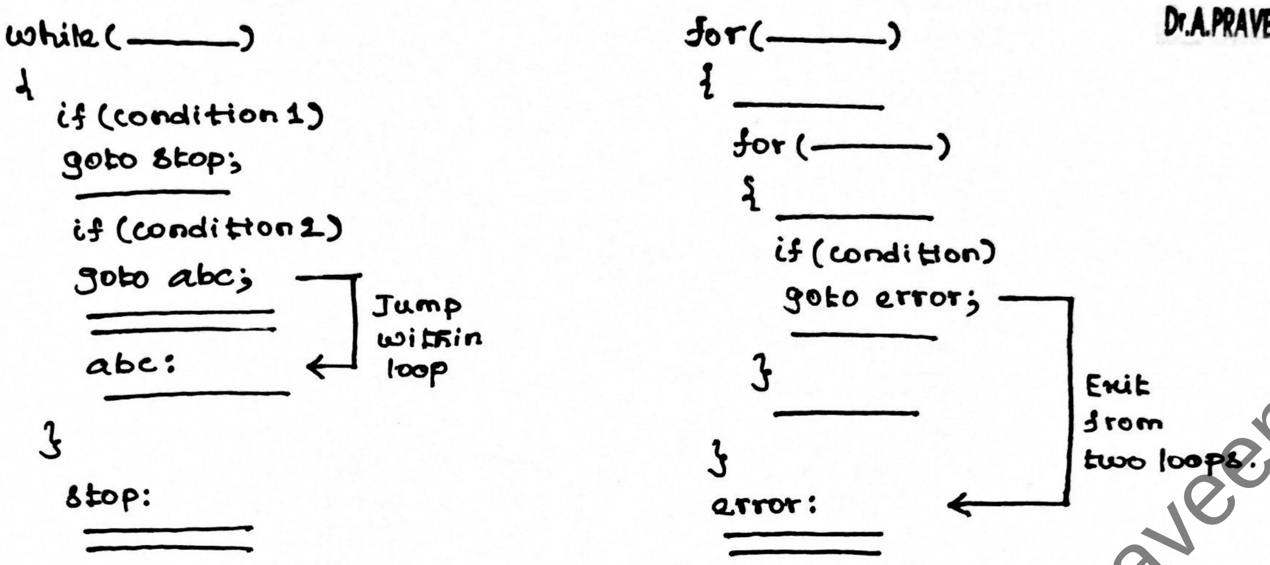
Note: Nesting of for loops, that is one for statement within another for statement, is allowed in C.
For example, two loops can be nested as follows:

```
-----------
-----------
for(i=1;i<6;i++)
 {
  ----------
  ----------
   for(j=0;j<5;j++)
     {
       -----------
       -----------
       -----------
     }
    -----------
    -----------
 }
 -----------
 -----------
```

Jumping out of a loop:
An early exit from a loop can be accomplished by using the *break* statement or the *goto* statement.
When the break statement is encountered inside a loop, the loop is immediately exited and the program continues with the statement immediately following the loop. When the loops are nested, the break would only exit from the loop containing it.
A goto statement can transfer the control to any place in a program. It is useful to provide branching within a loop.



(Exiting a loop with break statement)



```
while(______)
{
  if (condition 1)
  goto stop;
  ______
  if (condition 2)
  goto abc;
  ______
  ______
  abc:
  ______
}
 stop:
  ______
  ______
```

```
for(______)
{ ______
  for (______)
  { ______
    if (condition)
    goto error;
    ______
  } ______
}
error:
______
______
```

(Jumping within and exiting from the loops with goto statement)

<u>Skipping a part of a loop:</u>

The <u>continue</u> statement tells the compiler, <u>"SKIP THE FOLLOWING STATEMENTS AND CONTINUE WITH THE NEXT ITERATION"</u>. The continue statement causes the loop to be continued with the next iteration after skipping any statements in between.

The format of the continue statement is :

$\boxed{\textit{continue;}}$