Loop Constructs available in 'C' are as:

1. **while loop construct**

Multiple statements

```
while(<condition>)
{
<statements>;
----------;
<new_value_statement>;
}
```

Single statement

```
while(<condition>)
<single_line_statement_with_new_value>;
```

Note: Remember before coding the loop construct the value of the variable using in the condition of the loop construct must be set. Also, the value of the variable used in the condition must be changed from inside the loop construct that will make the result of condition of loop construct false. while loop construct is called "Entry Checking loop construct". Once the loop body is entered after condition resulting true it will be exited when the result of condition used becomes false.

2. **do-while loop construct**

Multiple statements

```
do
{
<statements>;
----------;
<new_value_statement>;
} while(<condition>);
```

Single statement

```
do
<single_line_statement_with_new_value>;
while(<condition>);
```

Note: Remember before coding the loop construct the value of the variable using in the condition of the loop construct must be set. Also, the value of the variable used in the condition must be changed from inside the loop construct that will make the result of condition of loop construct false. do-while loop construct is called "Exit Checking loop construct". Once the loop body is entered without checking any condition it will be exited when the result of condition used becomes false.

3. **for loop construct**

Multiple statements

```
for(<value>;<condition>;<new_value>)
{
<statement>;
----------;
}
```

Single statement

```
for(<value>;<condition>;<new_value>)
<statement>;
```

Note: The execution of for loop starts with evaluating the value statement coded inside the parenthesis "()" and then the condition is checked. Once the condition checked results true, the body of for loop is entered by the compiler and the set of statements or statement coded for the loop gets executed. When the last statement of the loop construct is executed, compiler directs the flow of execution to the new value statement that evaluates some different value for the variable used in the condition of

the loop. After evaluating new value statement, the condition is checked and the process stated above is performed by the compiler until the condition becomes false.

Variety of for loop construct

Multiple statements

```
<value_of_the_required_variable>;
for(;<condition>;<new_value>)
{
<statement>;
----------;
}
```

Single statement

```
<value_of_the_required_variable>;
for(;<condition>;<new_value>)
<statement>;
```

Multiple statements

```
<value_of_the_required_variable>;
for(;<condition>;)
{
<statement>;
----------;
<new_value_statement>;
}
```

Single statement

```
<value_of_the_required_variable>;
for(;<condition>;)
<Single_line_statement_with_new_value>;
```

Multiple statements

```
for(<value>;<condition>;)
{
<statement>;
----------;
}
```

Single statement

```
for(<value>;<condition>;)
<Single_line_statement_with_new_value>;
```

Multiple statements

```
<value_of_the_required_variable>;
for(;;<new_value>)
{
if(<condition>)
{
<statement>;
----------;
}
else
{
<statement>;
----------;
break;
}
}
```

Single statement

```
<value_of_the_required_variable>;
for(;;<new_value>)
{
if(<condition>)
<statement>;
else
break;
}
```

Remember if the programmer is using this for loop syntax then break statement is required for the termination of loop. The programmer may also use return statement or may make a function call to exit(0); or exit(1);. In the absence of this the loop becomes infinite.

Multiple statements

```
<value_of_the_required_variable>;
for(;;)
{
```

Single statement

```
<value_of_the_required_variable>;
for(;;)
{
```

```
if(<condition>)
{
<statement>;
----------;
}
else
{
<statement>;
----------;
break;
}
<new_value>;
}
```

```
if(<condition>)
<single_statement_with_new_value>;
else
break;
}
```

Remember inside the parenthesis of for loop construct double semi-colon is required in the absence of <value>,<condition> and <new_value> instruction.