CONTROL OF FLOW OF C PROGRAMMING - I

For statement

The general format of a *for* statement is :

```
for ( expression1; loop_condition; expression2)
{
  program statements;
}
```

The first component of the *for* statement, expression1 is used to set the intial values before an iteration begins. The second component (loop_condition) is a boolean expression (a logical expression that can be evaluated only to true or false) and specifies the condition(s) that has(have) to be met for the loop to continue. The final component (expression2) specifies the expression that is executed each time after the body of loop is executed.

Program 5.1

```
/* Usage of various arithmetic operators */

#include <stdio.h>

main ()
{
   int counter, number, triangular_number;
   triangular_number = 0;
   for (number=1; number<=100; ++number)
     triangular_number = triangular_number + number;
   printf("Triangular number of 100 is %d",
triangular_number);
}
```

Depending on the requirements of a program , you can have one *for* loop within another. This is known as nesting of loops. In such a case the complete

inner loop is executed for every pass of the outer loop. Indenting the statements within a *for* loop is important in order to improve readability.

While statement

The syntax of the *while* statement is

```
while (expression)
{
   program statements;
}
```

The expression specified inside the parentheses is evaluated, if the result of the expression is true then the program statements inside the loop is executed. After one iteration the expression is evaluated again and the loop is executed depending on whether the result is true. If the result is false, then the loop is terminated and the program execution continues with the statements following the while loop.

Program 5.2

```
/* Program to print multiples of 2 */
#include <stdio.h>
main()
{
  int count = 2;

  printf("Multiples of 2\n");
  while(count <= 50){

    printf("%d\n", count);
    count += 2;
  }
}
```

The program first sets the initial value to 2, then the loop begins to execute, since the value of count is less than 50 , it enters the loop prints the value of count , which is 2 and then increments it to 4. This continues till the value of count reaches 50.

The above program can also be implemented using a for loop,

```
for (count = 2; count <= 50; count += 2)
{
  printf("%d\n", count);
}
```

Note: Before writing a complete program, arrive at an algorithm which accomplishes the required end result you are seeking.

Program 5.3

```
/* Program to reverse the digits of a number */
#include <stdio.h>
main()
{

  int number;
  int reversed_digit;

  printf("Enter a number \n");
  scanf("%d",&number);

  while(number != 0) {
    reversed_digit = number % 10;
    printf("%d", reversed_digit);
```

```
    number = number / 10;
  }
printf("/n");
}
```

Do loop

Both statements, *for* and *while*, evaluate a condition or expression before executing the loop. Therefore the body of the loop is never executed if the conditions are not met at the start. Sometimes it is necessary to evaluate the condition at the end of a loop. This is done by using a *do* statement. The syntax of this statement is as follows:

```
do
  {
  program statements;
  }
while (expression1)
```

Here the program statements are first executed once, then the expression1 is evaluated. If it is true, then it proceeds to the next iteration. This continues till the expression1 evaluates to false.

Source : http://www.peoi.org/Courses/Coursesen/cprog/frame5.html