IT 105: Introduction to Programming

Dr. Manish Khare
Dr. Saurabh Tiwari

Lecture 10

More Complex Repetition

- X takes values from 1 to 10. For every X, Y takes values from 0.5 to 2.25 in the step of 0.5, and for every Y, Z goes from 100 to 55 in decrements of -2, and for every Z, M increases in multiples of 4 starting with 4.

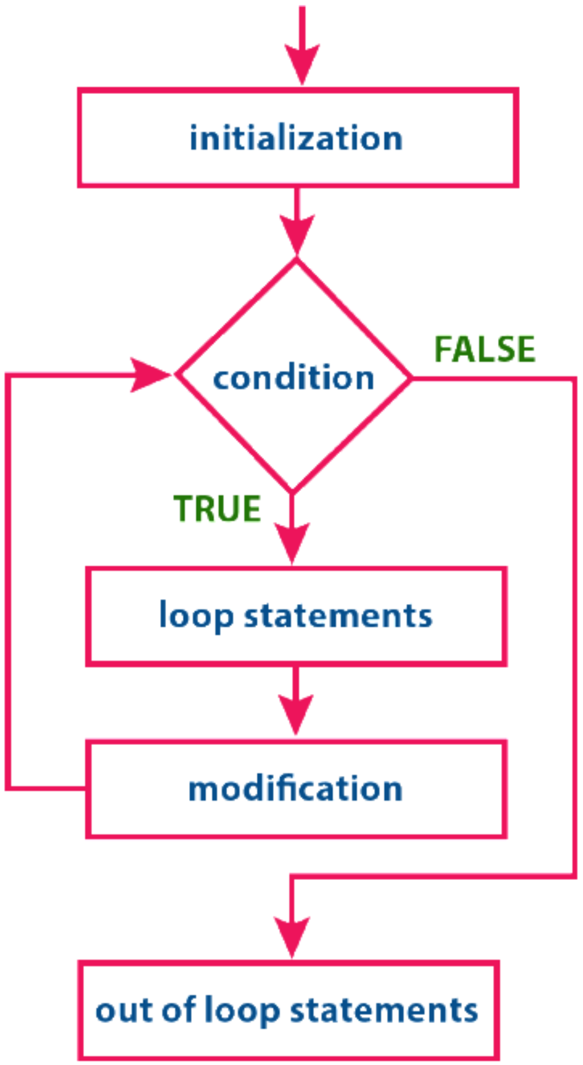
- Lost in the hoops??

for Loop

- The for loop allows us to specify things about the loop in a single line:
  - Setting a loop counter to an initial value.
  - Testing the loop counter to determine whether its value has reached the number of repetitions desired.
  - Increasing the value of loop counter each time the body of the loop has been executed.

- The general form of for statement is as under:

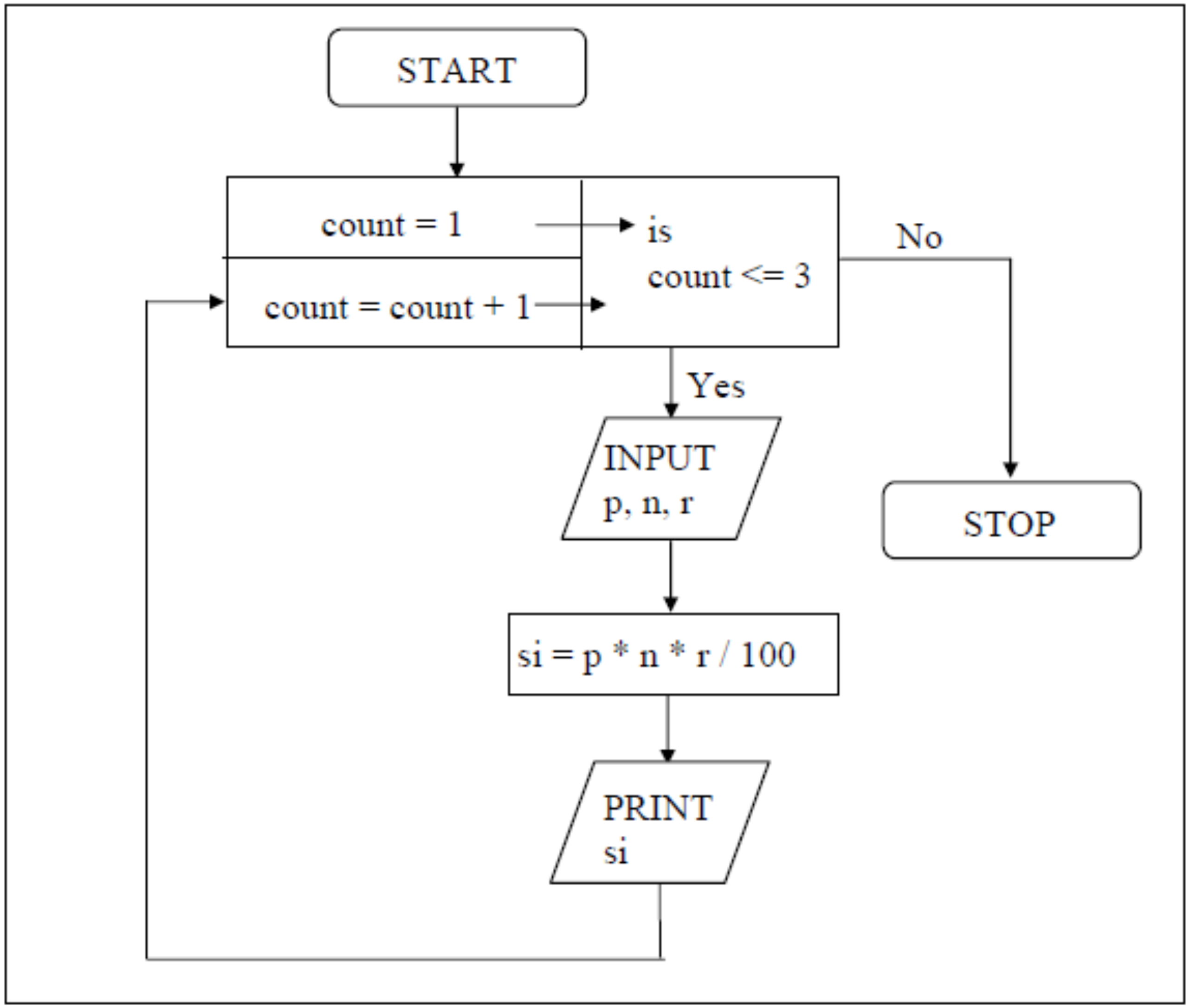
```
for ( initialise counter ; test counter ; increment counter )
{
    do this ;
    and this ;
    and this;
}
```


initialization
condition
FALSE
TRUE
loop statements
modification
out of loop statements

➢Let us now write down the simple interest program using for.

```
main( )
{
int p, n, count ;
float r, si ;
for ( count=1; count <= 3, count=count+1)
{
        printf ( "\nEnter values of p, n and r " ) ;
        scanf (“%d %d %f", &p, &n, &r ) ;
        si = n * r / 100 ;
        printf ( "Simple interest = Rs. %f", si ) ;
}
}
```

```
main( )
{
int p, n, count ;
float r, si ;
count =1;
while ( count <= 3 )
{
printf ( "\nEnter values of p, n and r " ) ;
scanf (“%d %d %f", &p, &n, &r ) ;
si = n * r / 100 ;
printf ( "Simple interest = Rs. %f", si ) ;
count = count + 1;
}
}
```


START
count = 1
count = count + 1
is count <= 3
No
Yes
INPUT p, n, r
STOP
si = p * n * r / 100
PRINT si

How for statement gets executed

- When the for statement is executed for the first time, the value of count is set to an initial value 1.
- Now the condition count <= 3 is tested. Since count is 1 the condition is satisfied and the body of the loop is executed for the first time.
- Upon reaching the closing brace of for, control is sent back to the for statement, where the value of count gets incremented by 1.
- Again the test is performed to check whether the new value of count exceeds 3.
- If the value of count is still within the range 1 to 3, the statements within the braces of for are executed again.
- The body of the for loop continues to get executed till count doesn't exceed the final value 3.
- When count reaches the value 4 the control exits from the loop and is transferred to the statement (if any) immediately after the body of for.

- It is important to note that the initialization, testing and incrementation part of a for loop can be replaced by any valid expression.
- Thus the following for loops are perfectly ok.
- for (i = 10 ; i ; i --)
  - printf ("%d", i) ;
- for (i < 4 ; j = 5 ; j = 0)
  - printf ("%d", i) ;
- for (i = 1; i <=10 ; printf ("%d",i++) ;
- for (scanf ("%d", &i) ; i <= 10 ; i++)
  - printf ("%d", i) ;

- Let us now write down the program to print numbers from 1 to 10 in different ways.

- This time we would use a for loop instead of a while loop.

Way 1

```
main( )
{
    int i ;
    for ( i = 1 ; i <= 10 ; i = i + 1)
    printf ( "%d\n", i ) ;
}
```

- Note that the initialisation, testing and incrementation of loop counter is done in the for statement itself. Instead of i = i + 1, the statements i++ or i += 1 can also be used.

- Since there is only one statement in the body of the for loop, the pair of braces have been dropped. As with the while, the default scope of for is the immediately next statement after for.

Way 2

```
main( )
{
    int i ;
    for ( i = 1 ; i <= 10 ;)
    printf ( "%d\n", i ) ;
    i=i+1;
}
```

- Here, the incrementation is done within the body of the for loop and not in the for statement. Note that in spite of this the semicolon after the condition is necessary.

Way 3

```
main( )
{
    int i =1;
    for ( ; i <= 10 ; i = i + 1)
    printf ( "%d\n", i ) ;
}
```

- Here the initialisation is done in the declaration statement itself, but still the semicolon before the condition is necessary.

Way 4

```
main( )
{
    int i =1;
    for ( ; i <= 10 ;)
    printf ( "%d\n", i ) ;
    i=i+1;
}
```

- Here, neither the initialisation, nor the incrementation is done in the for statement, but still the two semicolons are necessary.

Way 5

```
main( )
{
    int i ;
    for ( i = 0; i++<10 ; )
    printf ( "%d\n", i ) ;
}
```

- Here, the comparison as well as the incrementation is done through the same statement, i++ < 10.
- Since the ++ operator comes after i firstly comparison is done, followed by incrementation. Note that it is necessary to initialize i to 0.

Way 6

```
main( )
{
    int i ;
    for ( i = 0; ++i<10 ; )
    printf ( "%d\n", i ) ;
}
```

- Here, both, the comparison and the incrementation is done through the same statement, ++i <= 10. Since ++ precedes i firstly incrementation is done, followed by comparison. Note that it is necessary to initialize i to 0.

Nesting of Loops

- The way if statements can be nested, similarly whiles and fors can also be nested. To understand how nested loops work, look at the program given below:

```
/* Demonstration of nested loops */
main( )
{
    int r, c, sum ;
    for ( r = 1 ; r <= 3 ; r++ ) /* outer loop */
    {
        for ( c = 1 ; c <= 2 ; c++ ) /* inner loop */
        {
            sum = r + c ;
            printf ( "r = %d c = %d sum = %d\n", r, c, sum ) ;
        }
    }
}
```

- When you run this program you will get the following output:

```
r = 1 c = 1 sum = 2

r = 1 c = 2 sum = 3

r = 2 c = 1 sum = 3

r = 2 c = 2 sum = 4

r = 3 c = 1 sum = 4

r = 3 c = 2 sum = 5
```

- Here, for each value of r the inner loop is cycled through twice, with the variable c taking values from 1 to 2. The inner loop terminates when the value of c exceeds 2, and the outer loop terminates when the value of r exceeds 3.

Multiple Initialisations in the for Loop

- The initialisation expression of the for loop can contain more than one statement separated by a comma.
- For example,
  - for (i = 1, j = 2 ; j <= 10 ; j++)
- Multiple statements can also be used in the incrementation expression of for loop; i.e., you can increment (or decrement) two or more variables at the same time.

- However, only one expression is allowed in the test expression. This expression may contain several conditions linked together using logical operators.

- Use of multiple statements in the initialisation expression also demonstrates why semicolons are used to separate the three expressions in the for loop.

- If commas had been used, they could not also have been used to separate multiple statements in the initialisation expression, without confusing the compiler.

break statement

- We often come across situations where we want to jump out of a loop instantly, without waiting to get back to the conditional test.
- The keyword **break** allows us to do this. When break is encountered inside any loop, control automatically passes to the first statement after the loop.
- A break is usually associated with an if.

https://ideone.com/610SyM

https://ideone.com/gp8653

Continue Statement

- In some programming situations we want to take the control to the beginning of the loop, by passing the statements inside the loop, which have not yet been executed.

- The keyword continue allows us to do this.

- When continue is encountered inside any loop, control automatically passes to the beginning of the loop.

https://ideone.com/bc7nwj

Guess the output

- What would be the output of the following programs:

```
main( )
{
        int i = 0 ;
        for ( ; i ; )
                printf ( "\nHere is some mail for you" ) ;
}
```

Guess the output

```
main( )
{
int i ;
for ( i = 1 ; i <= 5 ; printf ( "\n%d", i ) ) ;
 i++ ;
}
```

Guess the output

```
main( )
{
    int i = 1, j = 1 ;
    for ( ; ; )
    {
        if ( i > 5 )
          break ;
        else
          j += i ;
        printf ( "\n%d", j ) ;
        i += j ;
    }
}
```

Exercise

According to a study, the approximate level of intelligence of a person can be calculated using the following formula:

$$i = 2 + (y + 0.5 x)$$

Write a program, which will produce a table of values of i, y and x, where y varies from 1 to 6, and, for each value of y, x varies from 5.5 to 12.5 in steps of 0.5.

https://ideone.com/grLnFx

Exercise

- Write a program to print the multiplication table of the number entered by the user. The table should get displayed in the following form.

```
29 * 1 = 29
29 * 2 = 58
…
```