FUNDAMENTALS OF PROGRAMMING WITH C#

LOOP CONSTRUCTS

Liu Fan
isslf@nus.edu.sg

Total:24

Objectives

- Write program that uses loop construct for repetition

Topics

- *While* loop Construct
- *Do…While* loop Construct
- *For* Constructs
  - Increment
  - Decrement
  - Step
- *Foreach* Constructs
- Exercise / Case Examples

While loop - Syntax

- The while loop is used to provide pre-check loops without counters. The following are the syntax:

```
while (Condition)
{
    These statements are repeatedly executed when the condition
    remains True.  Loop is exited at the head of the block after
    the condition becomes False.
    If the condition is False at the start, the block of statement
    would not be executed even once.
}
```

Example of a *While loop*

Do till **n** reaches one thousand?

```
int n, b;

b = 2;
n = 1;
while (n < 1000)
{
    n = b * n;
    b = b * b;
}
```

Example of a *While loop*

While we type anything but “Quit”, keep repeating the loop.

Program:

```
public static void Main()
{
    string a = "None";
    Console.WriteLine("Loop starts…");
    while (a != "Quit")
    {
        Console.WriteLine(“Type a Name: ”);
        a = Console.ReadLine();
        Console.WriteLine("You wrote:  " + a);
    }
        Console.WriteLine("Loop Over");
}
```

Output:

```
Loop starts…
Type a Name: Venkat
You wrote: Venkat
Type a Name: Quite ok
You wrote: Quite ok
Type a Name: Quit
You wrote: Quit
Loop Over
```

Do…*While* loop - Syntax

- The following are the syntax:

```
do
{
    These statements are repeatedly executed when the condition
    remains True.  Loop is exited at the end of the block after
    the condition becomes False.
    The block is executed at least once even if the condition
    was False at the start, since the condition is evaluated only
    at the end of the block.
}
while (Condition);
```

Example of a *Do…While* loop

While we type anything but “Quit”, keep repeating the loop.

Program:

```
public static void Main()
{
   string a;
   Console.WriteLine("Loop starts…");
   do
   {
       Console.WriteLine(“Type a Name: ”);
       a = Console.ReadLine();
       Console.WriteLine("You wrote:  " + a);
   } while (a != "Quit");
   Console.WriteLine("Loop Over");
}
```

Output:

```
Loop starts…
Type a Name: Venkat
You wrote: Venkat
Type a Name: VenQuit
You wrote: VenQuit
Type a Name: Quit
You wrote: Quit
Loop Over
```

For Statement Structure

- The For statement is a loop construct providing inherent structural facility (or placeholder) for initialising variables, iteration statements and terminal condition.
- Syntax is:

```
for (initializers; expression; iterators)
        statement( or block of statements)
```

where:

initializers

A comma separated list of expressions or assignment statements to initialise the loop counters.

expression

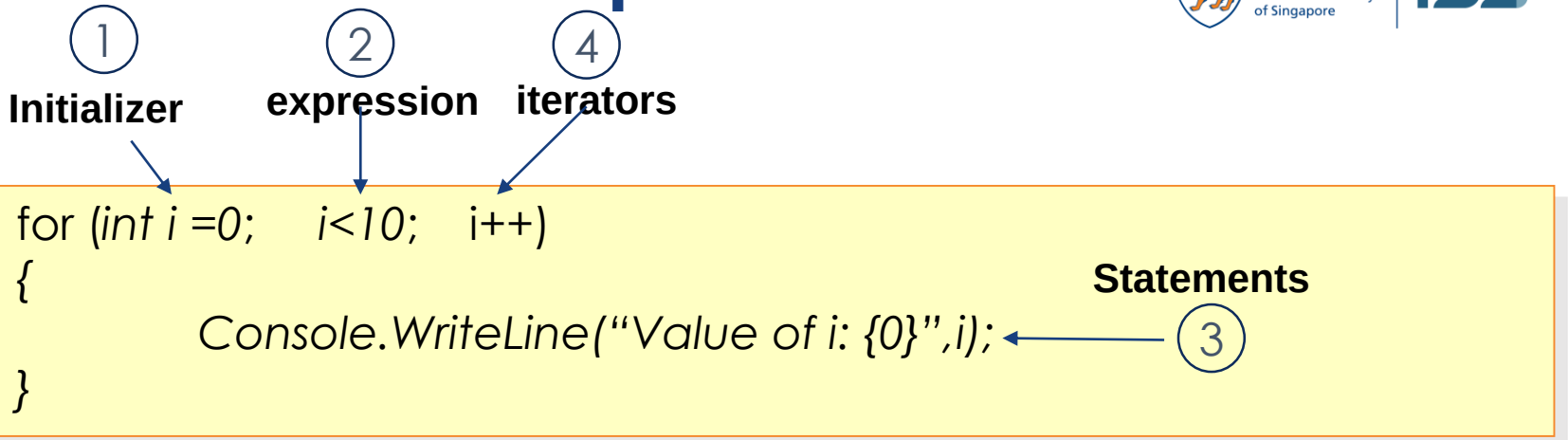
A expression (boolean/relational) that results in 'true' or 'false'. The expression is used to test the loop-termination criteria. So long as this expression is 'true' loop would be executed.

iterators

Expression statement(s) to increment or decrement the loop counters - multiple statements if used are comma separated.

All sections are optional.The body of the loop is either a statement or a block of statements.

For Statement Implementation



1 In **initializer**, the variable will be declared and initialized. The **initializer** will be executed only once at the starting of the for loop. Here *int i=0* is the initializer.

2 After completion of **initializer**, the **expression** will be evaluated. Here condition is a Boolean expression and it will return either **true** or **false**. Here, *i<10* is the expression.

3 In case, if **expression** is evaluated to **true**:

- The statements inside of **for** loop will be executed
- After that, the **iterator** will be executed and it will increase or decrease the initialized variable value based on our requirements. Here *i++* is the iterator.
- After changing the variable value, again the **expression** will be evaluated and execute the statements within the loop. Here is the print statement: *Console.WriteLine("Value of i: {0}",i);*
- This process will continue until the **expression** is evaluated to **false**.

4 If **expression** is evaluated to **false**, then the **for** loop execution will be stopped and control will come out of the loop.

For Statement Normal Use

- The For statement allows us to repeat a block of statements a certain number of times which is predetermined.
- Using the For statement we can perform a task repeatedly and increment / decrement the counter.
- Syntax is:

```
for ( count_variable = starting value; Terminal_Condition; increment statement )
{
   These statements are to be repeated 'n' number of times  where 'n' is determined by
   the initial value, terminal condition and the increment statement.
   Generally the Terminal_condition would contain a relational expression containing
   count_variable and increment statement  would alter the count_variable
   either increasing or decreasing its value by fixed number.
}
```

For Statement: Example

- The most straight forward usage:

Program:

```
int c;
for ( c = 1; c <= 5; c++ )
{
   Console.WriteLine(c);
}
Console.WriteLine("After the loop c has the value : {0} ", c);
```

Output:

```
1
2
3
4
5
After the loop c has the value: 6
```

For Loop Disciplines

- You can use a decrement operator (--) to indicate a decrement.
  - In this case, ensure that the start value is higher than the end value and the condition is correctly specified.
  - Usually the terminal condition statement reverses direction of test.
- You can work with non-integers - although rare for both the initial and terminal values as well as the step increment.
- **Never** change the value of the counter variable within a For loop (unless you are 100% sure). Unexpected (usually adverse) results occur.
- You can jump out of a for loop (to any other point) with the aid of goto statement. If structured discipline is required and you wish to avoid the use of goto, then you should use the do…while loop with explicit or use a complex condition in the for loop.
- **Never** jump into a for loop! This is stupid as counter value becomes indeterminate. Generally avoid use of goto.
- You can use break keyword to stop the execution of For loop statement based on our requirements.

For Loop Constructs

- This will present 10 lines with the word “Hello” in each.

```
for(i = 1; i <= 10; i++)
    Console.WriteLine( “Hello”);
```

- This will print numbers 1 through 10, one per line.

```
for(i = 1; i <= 10; i++)
    Console.WriteLine( i );
```

For Loop - a boon for mathematics

In this example, we will sum all the integers from 1 to k.

$$\sum_{j=1}^{k} j$$

```
int j, k, m;
k = 5;
m = 0;
for ( j = 1; j <= k; j++ )
{
   m = m + j;
}
Console.WriteLine(m);
```

Example of a *For* Loop using STEP

- This program prints all even numbers up to 10.

```
int j, k, m;
m = 2;
k = 10;
j = 0;
for (j = m; j <= k; j = j + m)
{
   Console.WriteLine( “ {0}  is an even number”, j);
}
```

Nested *For* Loops

- Like If statements, For loops can be nested.
- The innermost loop will be executed a number of times equal to the number of iterations of the outer loop *times* the number of the inner loop.

Program:

```
for ( j = 1; j <= 3; j++ )
{
    Console.WriteLine(“** {0} **”, j);
    for ( i=1; i <=2; i++ )
    {
        Console.WriteLine("{0} \t {1}", j, i);
    }
}
```

Output:

```
** 1 **
1       1
1       2
** 2 **
2       1
2       2
** 3 **
3       1
3       2
```

Interacting *For* Loops

- Interactions between the interaction variables can create many effects.

Program:

```
for ( j = 1; j <= 3; j++ )
{
    Console.WriteLine(“** {0} **”, j);
    for(i=1; i <=j; i++)
    {
        Console.WriteLine("{0} \t {1}", j, i);
    }
}
```

Output:

```
** 1 **
1       1
** 2 **
2       1
2       2
** 3 **
3       1
3       2
3       3
```

For Loop with Break Statement

- Following is the example of stop the execution of “for loop” by using break.

Program:

```
for ( i = 0; i <= 4; i++ )
{
    Console.WriteLine(“Value of i: {0}”,i);
    if (i==3)
    {
        break;
    }
}
Console.WriteLine(“Exit”)
```

Output:

```
Value of i: 0
Value of i: 1
Value of i: 2
Value of i: 3
Exit
```

For Loop without initializers and iterators

- It is not necessary to put the *initializers*, *expressions* and iterators into brackets. You can initialize a variable before the “for loop”, and the expressions and iterators can be defined inside the “for loop”.

Program:

```
int i = 0;
for (;;)
{
   if (i < 5)
   {
      Console.WriteLine("Value of i: {0}", i);
      i++;
   }
   else
      break;
}
```

Output:

```
Value of i: 0
Value of i: 1
Value of i: 2
Value of i: 3
Value of i: 4
```

Infinite *For* Loop

In case, if the expression in for loop always returns true, then the “for loop” will be an infinite and runs forever. Even if we miss the expression in “for loop” automatically that loop will become an infinite loop

Program:

```
for (int i = 5; i > 4; i++ )
{
    Console.WriteLine(“Value of i: {0}”,i);
}
```

Some Mathematical Notation using *For* loop

- Remember that computer languages were first invented to simplify mathematical calculations.
- The For loop is similar to the mathematical terms pi and sigma.
- Write program segments for the following and test it out.

$$\sum_{j=1}^{k} j \qquad \prod_{i=1}^{5} i \qquad \sum_{i=1}^{10}\sum_{j=1}^{10} ij$$

Foreach Construct

- C# provides an easy to use and more readable alternative to “for loop”, the “foreach loop” when working with arrays and collections to iterate through the items of arrays/collections. The “foreach loop” iterates through each item, hence called “foreach loop”.

```
foreach (element in iterable-item)
        statement( or block of statements)
```

Program:

```
int[] myarray = {1,2,3,4,5};
foreach(int a in myarray)
{
  Console.WriteLine(a);
}
```

Summary

- *while* loop check for condition before executing the block
  - Used for possible repetition of 0 to many times
- *do … while* loop check for condition after executing the block
  - Repeat at least once
- *for* loop normally used when the number of repetition is fixed (based on a variable or something) to make the code more readable
  - Although you can convert any while loop statement into a for loop