Looping/Repetition/Iteration – Chapter 5

# Demonstration of Questions 7-11 is due at the beginning of next class

Looping – this is the third control structure we have worked with in this course. Each control structure brings some advantage to the programmer. This one allows the coder to do more and at the same time write less code. This is possible by processing a block of code statements more than once. C# provides 4 looping constructs.

The do-while is the most intuitive of the looping constructs. The syntax of the do-while structure is:

int counter = 0;  
do  
{  
 Console.WriteLine(counter);  
 counter++;  
}while(counter < 5);

initializer  
do  
{  
 statement 1;  
 statement 2;  
}while(«boolean expression»);

Lab Exercise on Looping Using a do-while loop write the following programs:

1. To display five asterisks on a single line. Each asterisk must be separated by a space. You must use a do-while loop to solve this program
2. To display the numbers 1 … 5 on separate lines
3. To display the numbers 1 … 20 on separate lines
4. To display the numbers 1 … 20 on separate lines with an asterisk after the multiples of 5. (multiples besides asterisk)
5. To display the numbers 10 … 40 on separate lines with an asterisk before the multiples of 7. (asterisk besides multiples)
6. To display numbers 20 … 60 on separate lines skipping the multiple of 3. You must print a blank line for each multiple.
7. To repeatedly prompt for a number and sum it. When the sum just exceeds 100, stop the prompting and display the sum at the end. You must not display the sum while the user in typing in numbers. (You do not need a counter but you will need some way of terminating the loop)
8. Same as the previous question but additionally displays the average of the numbers at the end of the loop. You will need a counter but not as a loop terminator.
9. To display the sum of all the multiples of 3 between 1000000 and 2000000.   
   (Answer = 499, 999, 500, 000). If the sum might be larger than 2billion then the type of sum should be a long and not an int. [Hint: like the **C** (currency) and the **F** (decimal) format specifiers, there is also a **N** specifier]
10. A conversion table of Celsius to Fahrenheit temperature. The table must start with 0 Celsius and end at 100 Celsius with increments of 10. (Fahrenheit = Celsius \* 9/5 + 32)
11. A machine purchased for $28,000 is depreciated at a rate of $4,000 a year for seven years. Write and run a C# program that computes and displays a depreciation table for seven years, The table should have the form:

END-OF-YEAR ACCUMULATED  
YEAR DEPRECIATION VALUE DEPRECIATION  
---- ------------ ------------ ------------  
 1 4000 24000 4000  
 2 4000 20000 8000  
 3 4000 16000 12000  
 4 4000 12000 16000  
 5 4000 8000 20000  
 6 4000 4000 24000  
 7 4000 0 28000

### More on loops – the continue and break keywords

The do-while is a post-test loop, this means that the body of the loop is processed and then the condition is checked. This implies that the body is guaranteed to be processed at least once.

The other loop is the pre-test loop, this means that the condition is checked before the body is processed. This implies that the body might not be processed at all!

#### The continue keyword

When the continue keyword is seen, the rest of the statements in the loop body are ignored and processing carry on from the start of the loop.

#### The break keyword

When the break keyword is seen, the rest of the statements in the loop body are ignored and the loop is terminated

int a = 1;  
do  
{  
 Console.Write(a++);  
 if(a % 3 == 0)  
 break;  
 Console.WriteLine()  
*}while(a < 8);*

*//outputs  
123*

int a = 1;  
do  
{  
 Console.Write(a++);  
 if(a % 3 == 0)  
 continue;  
 Console.WriteLine()  
}while(a < 8);

//outputs  
123  
456  
78

Nested Loops

Write programs to output the following patterns. You must not hard-code the size of the figure. You should be able to output different sizes by changing the value of one or two variables

\*\*\*\*\*\*\*\*  
\*\*\*\*\*\*\*\*  
\*\*\*\*\*\*\*\*

\*  
 \*  
 \*  
 \*  
 \*  
 \*  
 \*

\*  
 \*  
 \*  
 \*  
 \*  
 \*  
\*

\* \*  
 \* \*  
 \* \*  
 \* \*  
 \*  
 \* \*  
 \* \*  
 \* \*  
\* \*

\*\*\*\*\*\*\*  
\* \*\*\*\*\*\*  
\*\* \*\*\*\*\*  
\*\*\* \*\*\*\*  
\*\*\*\* \*\*\*  
\*\*\*\*\* \*\*  
\*\*\*\*\*\* \*  
\*\*\*\*\*\*\*

\*\*\*\*\*\*  
\*\*\*\*\* \*  
\*\*\*\* \*\*  
\*\*\* \*\*\*  
\*\* \*\*\*\*  
\* \*\*\*\*\*  
 \*\*\*\*\*\*

\*\*\*\*\*\*\*\*  
\* \*  
\* \*  
\* \*  
\* \*  
\* \*  
\* \*  
\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*  
 \*\*\*\*\*\*\*\*  
 \*\*\*\*\*\*\*\*  
 \*\*\*\*\*\*\*\*  
 \*\*\*\*\*\*\*\*

\*  
 \*\*\*  
 \*\*\*\*\*  
 \*\*\*\*\*\*\*  
 \*\*\*\*\*\*\*\*\*  
 \*\*\*\*\*\*\*\*\*\*\*  
 \*\*\*\*\*\*\*\*\*\*\*\*\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*  
 \* \*  
 \* \*  
 \* \*  
 \* \*  
 \*

\*  
 \*\*\*  
 \*\*\*\*\*  
 \*\*\*\*\*\*\*  
\*\*\*\*\*\*\*\*\*  
 \*\*\*\*\*\*\*  
 \*\*\*\*\*  
 \*\*\*  
 \*