Nested for Loop in C#

A **nested for loop** is when a for loop is placed inside another for loop. This structure is useful when dealing with multidimensional problems, such as working with matrices, creating patterns, and performing repetitive calculations.

```
for (initialization; condition; increment/decrement)
{
  for (initialization; condition; increment/decrement)
  {
    // Inner loop statements
  }
  // Outer loop statements
}
```

- The **outer loop** runs first.
- For each iteration of the **outer loop**, the **inner loop** runs completely.
- The process continues until the **outer loop** condition fails.

Example 1: Printing a Multiplication Table

```
for (int i = 1; i <= 5; i++) // Outer loop for rows
  {
    for (int j = 1; j <= 5; j++) // Inner loop for columns
    {
      Console.Write((i * j) + "\t"); // Print product
    }
    Console.WriteLine(); // New line after each row
  }
```

Output:

1 2 3 4 5

2 4 6 8 10

3 6 9 12 15

4 8 12 16 20

5 10 15 20 25

Example 2: Printing a Right-Angled Triangle

```
for (int i = 1; i <= 5; i++) // Controls rows
    {
       for (int j = 1; j <= i; j++) // Controls columns
       {
          Console.Write("* ");
       }
       Console.WriteLine(); // Move to next line
    }
```

Output:

```
*
* *
* * *
* * * *
* * * * *
```

Explanation:

- The outer loop determines the number of rows (i)
- The inner loop prints * according to the row number.
- Each row has i stars.

Example 4: Printing a Pyramid Pattern

```
int n = 5; // Number of rows
    for (int i = 1; i <= n; i++) // Controls rows
   {
       for (int j = 1; j <= n - i; j++) // Prints spaces
       {    Console.Write(" ");   }
       for (int k = 1; k <= 2 * i - 1; k++) // Prints stars
       {       Console.Write("*");   }
       Console.WriteLine(); // Move to the next line
   }
```

Output:

```
    *
   ***
  *****
 *******
*********
```

Key Takeaways

1. **Execution Order**:
    - The inner loop completes all its iterations for each outer loop iteration.
2. **Use Cases**:
    - Matrix operations
    - Printing patterns
    - Working with multi-dimensional arrays
3. **Performance Consideration**:
    - Nested loops increase time complexity.
    - Avoid unnecessary nesting to optimize performance.