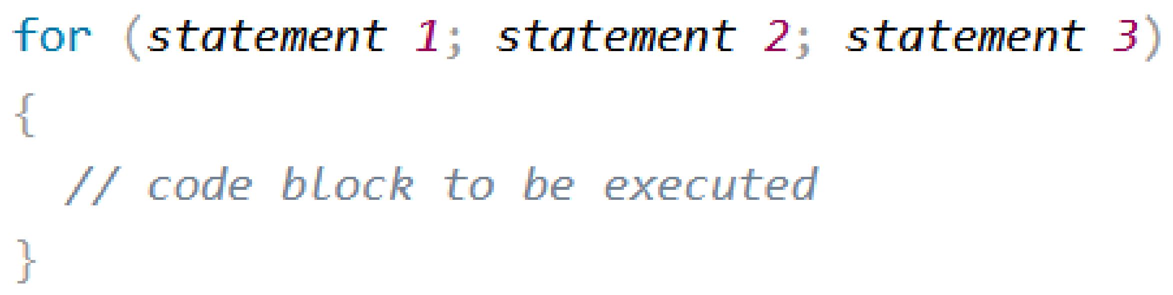
Advanced Computer Programming TICT2134

C# For Loop

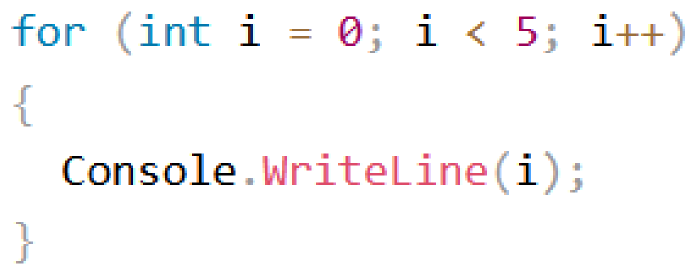
- **Statement 1** is executed (one time) before the execution of the code block.
- **Statement 2** defines the condition for executing the code block.
- **Statement 3** is executed (every time) after the code block has been executed.



```
for (statement 1; statement 2; statement 3)
{
  // code block to be executed
}
```

C# For Loop (Cont.)

- The example below will print the numbers 0 to 4:



```
for (int i = 0; i < 5; i++)
{
  Console.WriteLine(i);
}
```

C# Nested Loops

- It is also possible to place a loop inside another loop. This is called a nested loop.

```
// Outer loop
for (int i = 1; i <= 2; ++i)
{
  Console.WriteLine("Outer: " + i);  // Executes 2 times

  // Inner loop
  for (int j = 1; j <= 3; j++)
  {
    Console.WriteLine(" Inner: " + j); // Executes 6 times (2 * 3)
  }
}
```

C# Foreach Loop

- There is also a foreach loop, which is used exclusively to loop through elements in an array:

```
foreach (type variableName in arrayName)
{
  // code block to be executed
}
```

- Example

```
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
foreach (string i in cars)
{
  Console.WriteLine(i);
}
```

C# Arrays

- To declare an array, define the variable type with square brackets:

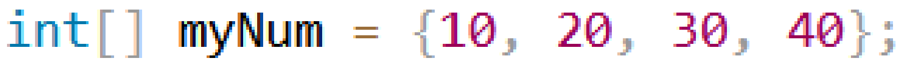
```
string[] cars;
```

- To insert values to it, we can use an array literal - place the values in a comma-separated list, inside curly braces:

```
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
```

C# Arrays (Cont.)

- To create an array of integers, you could write:



```
int[] myNum = {10, 20, 30, 40};
```

Access the Elements of an Array

- You access an array element by referring to the index number. This statement accesses the value of the first element in cars:

```
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
Console.WriteLine(cars[0]);
```

- To change the value of a specific element, refer to the index number:

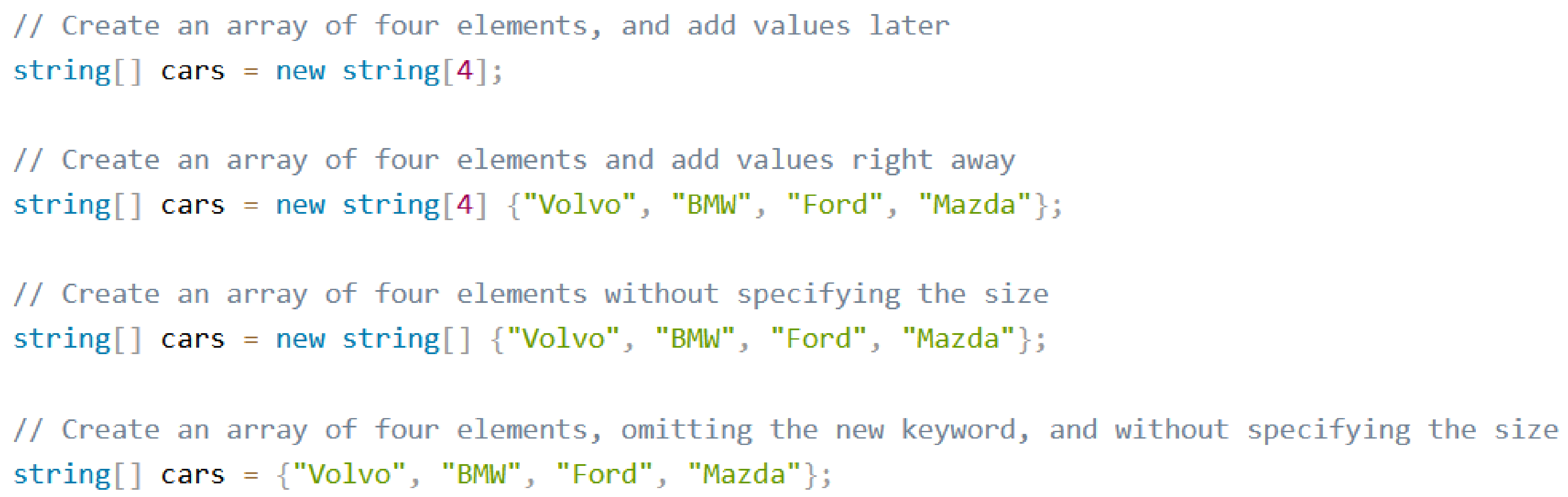
```
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
cars[0] = "Opel";
Console.WriteLine(cars[0]);
// Now outputs Opel instead of Volvo
```

Array Length

- To find out how many elements an array has, use the Length property:

```
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
Console.WriteLine(cars.Length);
// Outputs 4
```

Other Ways to Create an Array

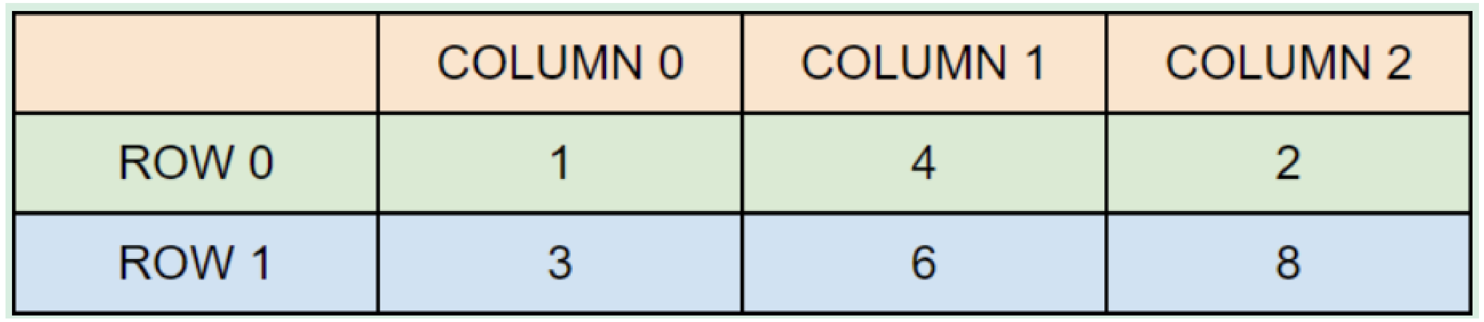


```
// Create an array of four elements, and add values later
string[] cars = new string[4];

// Create an array of four elements and add values right away
string[] cars = new string[4] {"Volvo", "BMW", "Ford", "Mazda"};

// Create an array of four elements without specifying the size
string[] cars = new string[] {"Volvo", "BMW", "Ford", "Mazda"};

// Create an array of four elements, omitting the new keyword, and without specifying the size
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
```

C# Multidimensional Arrays

- To create a 2D array, add each array within its own set of curly braces, and insert a comma (,) inside the square brackets:

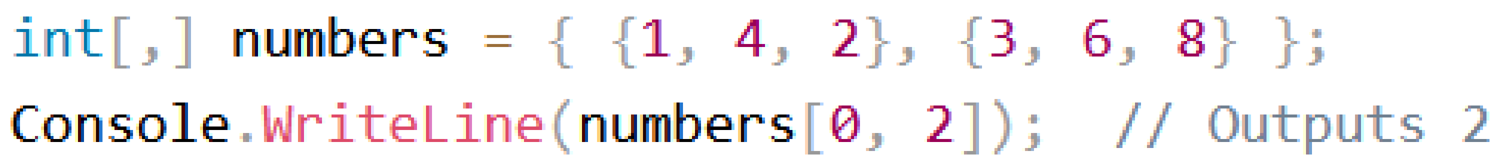
```
int[,] numbers = { {1, 4, 2}, {3, 6, 8} };
```



	COLUMN 0	COLUMN 1	COLUMN 2
ROW 0	1	4	2
ROW 1	3	6	8

Access Elements of a 2D Array

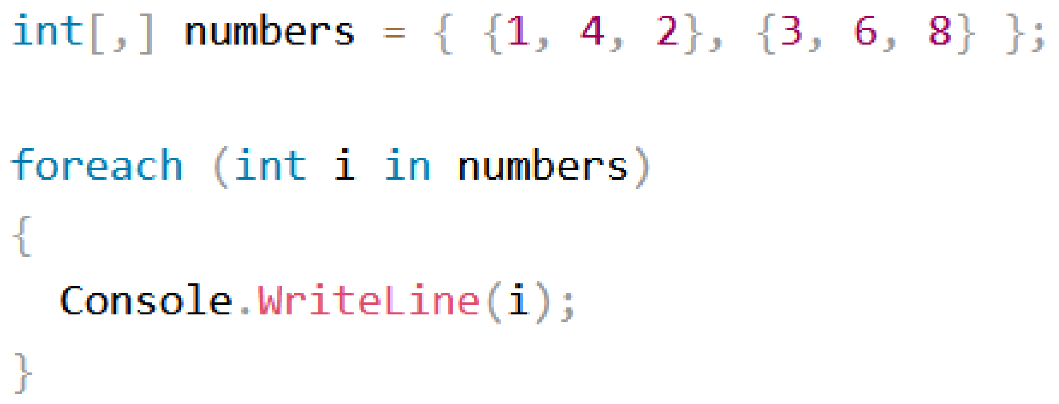
This statement accesses the value of the element in the first row (0) and third column (2) of the numbers array:



```
int[,] numbers = { {1, 4, 2}, {3, 6, 8} };
Console.WriteLine(numbers[0, 2]);  // Outputs 2
```

Loop Through a 2D Array

You can easily loop through the elements of a two-dimensional array with a foreach loop:



```
int[,] numbers = { {1, 4, 2}, {3, 6, 8} };

foreach (int i in numbers)
{
  Console.WriteLine(i);
}
```

C# Methods

Create a method inside the Program class:

```
class Program
{
  static void MyMethod()
  {
    // code to be executed
  }
}
```

Call a Method

Inside Main(), call the myMethod() method:

```
static void MyMethod()
{
  Console.WriteLine("I just got executed!");
}

static void Main(string[] args)
{
  MyMethod();
}

// Outputs "I just got executed!"
```