FULL SAIL UNIVERSITY

C# Arrays

Arrays

An array is a data structure that contains several variables of the same type. Arrays are declared with a type:

```
type[] arrayName;
```

The following examples create single-dimensional, multidimensional, and jagged arrays:

```
class TestArraysClass
{
    static void Main()
    {
        // Declare a single-dimensional array
        int[] array1 = new int[5];

        // Declare and set array element values
        int[] array2 = new int[] { 1, 3, 5, 7, 9 };

        // Alternative syntax
        int[] array3 = { 1, 2, 3, 4, 5, 6 };

        // Declare a two dimensional array
        int[,] multiDimensionalArray1 = new int[2, 3];

        // Declare and set array element values
        int[,] multiDimensionalArray2 = { { 1, 2, 3 }, { 4, 5, 6 } };

        // Declare a jagged array
        int[][] jaggedArray = new int[6][];

        // Set the values of the first array in the jagged array structure
        jaggedArray[0] = new int[4] { 1, 2, 3, 4 };
    }
}
```

Array Overview

An array has the following properties:

- An array can be Single-Dimensional, Multidimensional or Jagged.
- The default value of numeric array elements are set to zero, and reference elements are set to null.
- A jagged array is an array of arrays, and therefore its elements are reference types and are initialized to null.
- Arrays are zero indexed: an array with n elements is indexed from 0 to n-1.

- Array elements can be of any type, including an array type.
- Array types are reference types derived from the abstract base type Array. Since this type implements IEnumerable and IEnumerable<(Of <(T>)>), you can use foreach iteration on all arrays in C#.

Arrays as Objects

In C#, arrays are actually objects, and not just addressable regions of contiguous memory as in C and C++. Array is the abstract base type of all array types. You can use the properties, and other class members, that Array has. An example of this would be using the Length property to get the length of an array. The following code assigns the length of the numbers array, which is 5, to a variable called lengthOfNumbers:

```
int[] numbers = { 1, 2, 3, 4, 5 };
int lengthOfNumbers = numbers.Length;
```

Rank property

This example uses the Rank property to display the number of dimensions of an array.

```
class TestArraysClass
{
    static void Main()
    {
        // Declare and initialize an array:
        int[,] theArray = new int[5, 10];
        System.Console.WriteLine("The array has {0} dimensions.", theArray.Rank);
    }
}
// Output: The array has 2 dimensions.
```

GetLength method

Gets a 32-bit integer that represents the number of elements in the specified dimension of the Array.

```
class TestArraysClass
{
    static void Main()
    {
        // Declare and initialize an array:
        int[,] theArray = new int[5, 10];
        System.Console.WriteLine("Dimension 0 has {0} elements.", theArray.GetLength(0));
        System.Console.WriteLine("Dimension 1 has {0} elements.", theArray.GetLength(1));
    }
}
// Output: Dimension 0 has 5 elements.
// Output: Dimension 1 has 10 elements.
```