**Introducing Arrays**

An array is a set of objects that are grouped together and managed as a unit. You can think of an array as a sequence of elements, all of which are the same type. You can build simple arrays that have one dimension (a list), two dimensions (a table), three dimensions (a cube), and so on. Arrays in Visual C# have the following features:

* Every element in the array contains a value.
* Arrays are zero-indexed, that is, the first item in the array is element 0.
* The size of an array is the total number of elements that it can contain.
* Arrays can be single-dimensional, multidimensional, or jagged.
* The rank of an array is the number of dimensions in the array.

Arrays of a particular type can only hold elements of that type. If you need to manipulate a set of unlike objects or value types, consider using one of the collection types that are defined in the System.Collections namespace.

**Creating and Using Single Dimension Arrays**  
  
When you declare an array, you specify the type of data that it contains and a name for the array. Declaring an array brings the array into scope, but does not actually allocate any memory for it. The CLR physically creates the array when you use the new keyword. At this point, you should specify the size of the array.

To declare a single-dimensional array, you specify the type of elements in the array and use brackets, [] to indicate that a variable is an array. Later, you specify the size of the array when you allocate memory for the array by using the new keyword. The size of an array can be any integer expression. The following code example shows how to create a single-dimensional array of integers with elements zero through nine.  
  
int[] arrayName = new int[10];

You can also choose to create an array and initialize it with values at the same time as in the following example that declares and integer array and assigns values to it.  The compiler know how large to make the array by the number of values in the curly braces:

int[] arrayName = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

**Accessing Data in an Array**

You can access data in an array in several ways, such as by specifying the index of a specific element that you require or by iterating through the entire array and returning each element in sequence.

The following code example uses an index to access the element at index two.

//Accessing Data by Index  
int[] oldNumbers = { 1, 2, 3, 4, 5 };  
//number will contain the value 3  
int number = oldNumbers[2];

*Note: Arrays are zero-indexed, so the first element in any dimension in an array is at index zero. The last element in a dimension is at index N-1, where N is the size of the dimension. If you attempt to access an element outside this range, the CLR throws an IndexOutOfRangeException exception.*

You can iterate through an array by using a *for* loop. You can use the *Length* property of the array to determine when to stop the loop.

The following code example shows how to use a for loop to iterate through an array.

//Iterating Over an Array  
int[] oldNumbers = { 1, 2, 3, 4, 5 };  
for (int i = 0; i < oldNumbers.Length; i++)  
{  
    int number = oldNumbers[i];  
    ...  
}

**Multidimensional arrays**

An array can have more than one dimension. The number of dimensions corresponds to the number of indexes that are used to identify an individual element in the array. You can specify up to 32 dimensions, but you will rarely need more than three. You declare a multidimensional array variable just as you declare a single-dimensional array, but you separate the dimensions by using commas. The following code example shows how to create an array of integers with three dimensions.  
  
// Create an array that is 10 long(rows) by 10 wide(columns)  
int[ , ] arrayName = new int[10,10];

In order to access elements in a multidimensional array, you must include all indices as in the example code here.

// Access the element in the first row and first column  
int value = arrayName[0,0]

//Access the element in the first row and second column  
int value2 = arrayName[0, 1];

//Access the element in the second row and first column  
int value2 = arrayName[1, 0];

**Jagged arrays**

A jagged array is simply an array of arrays, and the size of each array can vary. Jagged arrays are useful for modeling sparse data structures where you might not always want to allocate memory for every item if it is not going to be used. The following code example shows how to declare and initialize a jagged array. Note that you must specify the size of the first array, but you must not specify the size of the arrays that are contained within this array. You allocate memory to each array within a jagged array separately, by using the new keyword.  
  
int[][] jaggedArray = new int[10][];  
jaggedArray[0] = new Type[5]; // Can specify different sizes.  
jaggedArray[1] = new Type[7];  
...  
jaggedArray[9] = new Type[21];