ARRAYS

WHAT IS AN ARRAY?

An array is an object that contains one or more items called elements. All elements of an array must be of the same type.

Arrays have a fixed length or size that indicates the number of elements that it contains. You can code the number of elements as a literal, a constant, or a variable of type int.

CREATING ARRAYS

To create an array you must declare a variable of the correct type and instantiate an array object that the variable refers to.

The elements of an array are referred to by their index. Indexing begins with 0, so an index of 0 refers to the first element; an index of 3 refers to the fourth element.

INITIAL VALUES IN ARRAYS

Depending on the data type for the array, each element will be given an initial value.

- 0 for numeric arrays
- false for booleans
- '\0' (zero) for characters
- null for objects

DECLARING AND INSTANTIATING ARRAYS

```
type[ ] arrayName = new type[length];
```

Or

```
type[ ] arrayName;
arrayName = new type[length];
```

DECLARING AND INSTANTIATING ARRAYS

```
String[ ] titles = new String[3];

double[ ] prices;
prices = new double[4];
```

DECLARING AND INSTANTIATING ARRAYS

```
String[ ] titles = new String[3];//Array of Strings


const int TITLE_COUNT = 100;// Code that uses a constant
String[ ] titles = new String[TITLE_COUNT];
```

DECLARING AND INSTANTIATING ARRAYS

```
//Code that uses a variable
string input = Console.ReadLine();
int titleCount = int.Parse(input);
String[ ] titles = new String[titleCount];
```

ACCESSING ARRAY ELEMENTS

```
arrayName[index];
myArray[2]; // Refer to the third element
```

ASSIGNING VALUES TO ARRAYS

```
type[ ] arrayName = {value1, value2, value3, …};

// or

arrayName = new type[length];
arrayName[0] = value1;
arrayName[1] = value2;
arrayName[2] = value3;
```

USING FOR LOOP WITH ARRAYS

For loops are commonly used to process the elements in an array one at a time by incrementing an index variable. You can use the length field of an array to determined how many elements are defined for the array.

USING FOR LOOP WITH ARRAYS

SYNTAX

```
for (int i = 0; i < arrayName.length; i++)
{
//statements
}
```

USING FOR LOOP WITH ARRAYS

Code that prints an array of prices to the console

```
double[ ] prices = {14.95, 12.95, 11.95, 9.95};
for (int i = 0; i < prices.length; i++)
{
Console.WriteLine(prices[i]);
}
```

USING FOR LOOP WITH ARRAYS

Code that computes the average of the array of prices

```
double sum = 0.0;
for (int i = 0; i < prices.length; i++)
{
sum += prices[i];
}
double average = sum / prices.length;
```

USING FOREACH

The foreach loop lets you process each element of an array without the need to use indexes. The foreach loop simplifies the code required to loop through arrays.

FOREACH SYNTAX

```
foreach (type variableName in arrayName)
{
//use the variableName is access elements
}
```

USING FOREACH

Code that prints an array of prices to the console

```
double[ ] prices = {14.95, 12.95, 11.95, 9.95};
foreach (int element in prices)
{
Console.WriteLine(element);
}
```

USING FOREACH

Code that computes the average of the array of prices

```
double sum = 0.0;
for (int element in prices)
{
sum += element;
}
double average = sum / prices.length;
```

THE ARRAY CLASS

The Array class contains several static methods that you can use to compare, sort, and search arrays.

Examples: Sort, BinarySearch, Clear.

REFERENCE AND COPY ARRAYS

REFERENCE AN ARRAY

You can create a reference to an array by assigning an array variable to an existing array. Both variables will refer to the same array, thus changes to one will be reflected in the other.

REFERENCE AND COPY ARRAYS

COPY AN ARRAY

The easiest way to copy an array is to use the CopyTo method of the array object. When you copy an array the new array must be of the same type as the source array.

HOW TO COPY ARRAYS

```
double[ ] grades = {92.3, 88.0, 95.2, 90.5};
double[ ] percentages=new double [grades.Length];
grades.CopyTo(percentages, 0);// 0 is where the copy starts
percentages[1] = 70.2;
Console.WriteLine("grades[1]=" + grades[1]);
```

TWO-DIMENSIONAL ARRAYS

Two-dimensional arrays use two indexes to store data. Each element in the array is at the intersection of a row and column.

RECTANGULAR ARRAYS

The simplest type of two-dimensional array is a rectangular array, in which each row has the same number of columns.

RECTANGULAR ARRAYS

SYNTAX AND EXAMPLE

```
type[,] ArrayName = new type [RowCount,ColumnCount];
int[,] numbers = new int[3,2]; // Array with 3 rows and 2 columns
```

JAGGED ARRAYS

A jagged array is a two-dimensional array in which the rows contain unequal numbers of columns. When you instantiate a jagged array, you specify the number rows, but not the number of columns.

JAGGED ARRAYS

SYNTAX

```
type[ ] [ ] arrayName = new type[rowCount] [ ];
// Then we need to create each row
```

JAGGED ARRAYS

EXAMPLE

```
// Declare local jagged array with 3 rows.
int[][] ar = new int[2][];
ar[0] = new int[2]; // Create a new array for row 0
ar[1] = new int[5]; // Create a new array for row 0
```

RECAP

WHAT YOU SHOULD KNOW AT THIS POINT:

- What is an array.
- How to declare and instantiate an array.
- How to access and use array elements.
- How to use loops with arrays.
- How to use the Array class.
- How to copy arrays.
- How to create and use multi-dimensional arrays.