C# Arrays

An array is a set of variables, all of the same type. It enables the set to be manipulated together, often within a 'for' loop. There are also some built in array methods such as sorting the members of the array or finding how many members are contained within the array.

Without arrays variables need to be declared one at a time, values assigned and comparisins such as if used to work with the set.

```
int first, second, third, fourth;
first = 10;
second = 20;
```

An array is set up with a name and data type. This code creates a set of 12 int variables. Arrays are 0 based, the first member is 0 and the last is 11.

```
int[] numSet = new int[12];
```

The length of the array can be set in code.

```
String[] testArray;
Console.WriteLine("Enter an array length");
String input = Console.ReadLine();
int len = Int16.Parse(input);
testArray = new String[len];
for (int x = 0; x < testArray.Length; x++)
      Console.WriteLine(testArray[x] + " is empty");
```

The array length can also be changed in code but any data held within that array will be lost.

```
String[] testArray;
testArray = new String[8];
for (int x = 0; x < testArray.Length; x++)
        testArray[x] = "value of " + x;
testArray = new String[4];
for (int x = 0; x < testArray.Length; x++)
        Console.WriteLine(testArray[x] + " is now");
```

The individual members of the array can be assigned a value by specifying the position number within [ ]s.

```
numSet[4] = 999;
Console.WriteLine(numSet[4]);
```

This line will throw an error because there is no numSet[12]. The 12th member of numSet is numSet[11].

```
Console.WriteLine(numSet[12]);
```

The easiest way to run through the set is with a 'for' loop. This is similar to other loops but runs through a counter. This 'for' loop runs for 12 times; from 0 to 11. All the values of numSet are output, they are all initially 0.

```
for (int x = 0; x < 12; x++)
{
	Console.WriteLine(numSet[x]);
}
```

A for loop can also be used to assign values to all members of an array

```
for (int x = 0; x < 12; x++)
{
	numSet[x] = x;
	Console.WriteLine(numSet[x]);
}
```

Arrays of other data types than int are assigned data in the usual way but the whole array must have the same data type.

```
String[] wordSet = new String[12];
for (int x = 0; x < 12; x++)
{
	wordSet[x] = "hello";
}
```

An array can be set up and assigned values in a single line:

```
String[] wordSet = { "Bill", "Fred", "Stanley", "Amy", "Bilal" };
int[] numSet = { 1, 4, 34, 2, 1, 8 };
```

These can be accessed as usual with a 'for' loop.

```
for (int x = 0; x < 5; x++)
{
	Console.WriteLine(wordSet[x]);
}
```

The above code can be made more robust because the length of an array can be calculated.

```
int arrayLength = numSet.Length;
```

This method can be accessed within the 'for' loop to prevent it running off the end of the array and throwing an error.

```
for (int x = 0; x < wordSet.Length; x++)
{
      Console.WriteLine(wordSet[x]);
}
```

Array methods

Length is one of a number of methods that can be applied to an array. Not all of these are relevant to every data type that might be held within an array. The following are useful array method examples in C# when dealing with int values.

```
int length = numSet.Length;
double av = numSet.Average();
int biggest = numSet.Max();
int littlest = numSet.Min();
int total = numSet.Sum();
Console.WriteLine(av + ", " + length + ", " + biggest + ", "  +
littlest + ", " + total);
```

The length is a property not a method so is not followed by ( ).

All the above methods (and the property) return some information about the array. They do not manipulate its contents. The Array class has a number of methods that change the elements within the array.

```
Array.Sort(wordSet);
for (int x = 0; x < wordSet.Length; x++)
{
      Console.WriteLine(wordSet[x]);
}
```

Note that these begin with the class name Array not the name of the array concerned, wordSet above.

```
Array.Sort(wordSet);
Array.Reverse(wordSet);
Array.Clear(wordSet,0,2); //clear from position 0 - go along 2
String [] newWords = new String[4];
Array.Copy(wordSet, newWords, 4);//copy 1st 4
for (int x = 0; x < newWords.Length; x++)
{
      Console.WriteLine(newWords[x]);
}
```

Arrays in functions

The key main line `static void setArray(int[] myArray)` of every C# program shows that an array can be passed to a function. This code proves it:

```
static void setArray(int[] myArray)
{
     for(int x=0;x<myArray.Length;x++)
     Console.WriteLine(myArray[x]);
}
static void Main(string[] args)
{
     int [] testArray = {1,3,5,7,9};
     setArray(testArray);
     Console.WriteLine("Back to main");
     Console.ReadLine();
}
```

It also possible to return an array from a function. As the whole array is returned the array that is passed back does not have to be the same length or even the same type as the original.

```
static int [] zeroArray(int[] myArray)
{
      for(int x=0;x<myArray.Length;x++)
          myArray[x] = 0;
return myArray;
}

static void Main(string[] args)
{
     int [] testArray = {1,3,5,7,9};
     int[] newArray = zeroArray(testArray);
     for (int x=0; x< newArray.Length;x++)
                Console.WriteLine(newArray[x]);
     Console.ReadLine();
}
```

This more useful function will convert an array of Strings (as might be input by a user) to an array of int data types (as might be used for maths).

```
static int [] intArray(String[] myArray)
{
       int num = myArray.Length;
       int[] retArray = new int[num];
       for(int x=0;x<myArray.Length;x++)
           retArray[x] =  Int16.Parse (myArray[x]);
       return retArray;
}
```

```
static void Main(string[] args)
{
      String [] testArray = {"1","3","5","7","9"};
      int[] newArray = intArray(testArray);
      for (int x=0; x< newArray.Length;x++)
                Console.WriteLine(newArray[x]);
       Console.ReadLine();
}
```

Exercises

1. Write a for loop that outputs a simple message 10 times
2. Ask the user for a value and run the message for loop that many times.
3. Create an array of type int and place the numbers from 1 to 10 in it. Output the members of the array to the console. Two for loops should be used.
4. Create an array of type int and place the numbers from 2 to 20 in it. Output the members of the array to the console. Two for loops should be used.
5. Ask the user for a number. Create an array of numbers of that length and fill it with the numbers 0 up to the last array position. Output the members of the array to the console.
6. Create an array of at least 10 non sequential numbers. Output the array members plus their total, maximum, minimum and average values.
7. Create an array of at least 10 names. Output the array to the console.
8. Sort the array of names alphabetically, output the original and sorted array to the console.
9. Copy the 1st 6 members of the array of names into another array; sort the original in ascending form and the copy in descending. Output both arrays to the console.
10. Modify the function intArray to check that each member is a valid number before converting to an int. If the value is not a number 0 should be substuted into the array. Using a try catch block to catch the error `FormatException` would be a possible solution