[0440] C# Arrays

Arrays are the simplest data structure in programming. For several decades, arrays were all programmers had for storing data. Those were dark days.

Like all data structures, arrays have Pro's and Con's. The biggest Pro for arrays is that they are VERY MEMORY EFFICIENT. Arrays have zero "overhead" meaning that there is no additional memory needed other than the actual data being saved. No other data structure can say that. That is why arrays are still very popular in low-memory-footprint environments.

Arrays are also very quick to get data out of. To read an old-school array element, you just multiply the index of the element you want by the size of each element, add that result to the start of the array and then get your data from the resulting location. Easy peazy! C and C++ still do it like this. C# does it slightly differently to help with memory management, but the concept is generally the same.

While an array is very memory efficient and very quick for data retrieval, it is also very slow and memory inefficient if you need to insert new data elements into its middle. (Do you know why?)

To declare an array you use this syntax:

```
<type>[] myArrayName = new <type>[##];
```

where <type> is the data type of the elements in the array
is the maximum number of elements you can store in the array

You can initial an array with a set of literal constants like this:

```
int[] myInts = { 1, 2, 3, 4, 5 };
```

Note: Initializing arrays like this is common in school assignments, uncommon in the real-world.

C# Arrays

In C#, arrays are automatically converted into .NET's underlying System.Array Base Class which means C# arrays automatically have a bunch of really useful utility methods including Array.*Sort(), Array.CopyTo()* and the *Length* property (the most used).