- **Data structures** are formats designed to store larger amounts of information in an organized fashion.  
- An **array** is a data structure used in C# to store a sequential collection of elements.  
- Order data in a specific, linear sequence  
- When defining arrays, need to define the type as well. Hold data types that are all the same

A screen shot of a computer

Description automatically generated A number of letters and symbols

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

- new signifies that we are instantiating a new array from the array class  
- If we decide to declare an array and then initialize it later, rather than in one line like above, the new keyword *must* be used:

A computer screen shot of a program code

Description automatically generated

**Array Length:**

- Can find out how many items an array contains by using the .Length method  
- Using the .Length property will return the number of items in an array and 0 if the array is empty.

A computer screen shot of a number

Description automatically generated

A screenshot of a computer

Description automatically generated

**Accessing Array Items:**

- Arrays order items so that they’re in a specific sequence, which makes it easy to access a specific item.   
- Each value has a position in the array, which is known as its **index**.  
- Arrays start at index value 0 and increment by 1 for each subsequent value

A diagram of a data flow

Description automatically generated with medium confidence

- To access a value from a list, we write out the name of the array, followed by brackets []. Within the brackets, we specify the index number of the value we want:

A computer screen shot of a number

Description automatically generated A screenshot of a computer code

Description automatically generated

**Editing Arrays:**

- Once we create an array, the size of that array is fixed. However, it’s possible to change the values it contains  
- Can initialize an array and fill in values later on

A computer code with white and orange text

Description automatically generated

- When we create an array with a known length but no known values, it uses the **default type value** (0 for int, null for string).

A computer screen shot of numbers

Description automatically generated

- Can swap out the values of existing arrays

**Built-In Methods:**

- Several built-in methods for use with arrays

**Sort:**

- The built-in method Array.Sort(),sorts an array either via ascending order (int) or alphabetical order (string)  
- Cannot assign it to a variable as it reverse the existing list in place and does not return a value

A computer code with white and orange text

Description automatically generated

**IndexOf:**

- The Array.IndexOf() method takes a value and returns its index within the array.  
- Works best when you have a specific value and need to know where it’s located in the array (or if it even exists!).  
-  Typically takes 2 parameters: the first is the array and the second is the value whose index we’re looking for.

A computer screen with text

Description automatically generated

- If the value appears more than once in an array, it returns only the index of the first occurrence within the specified range. If it does not find the value at all, it returns -1

**Reverse:**

- The Array.Reverse() method returns the array with the original elements in reverse order.  
- Cannot assign it to a variable as it reverse the existing list in place and does not return a value

A computer code with numbers and symbols

Description automatically generated