Copyright © 2015 by The Learning House.

All rights reserved. No part of these materials may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of The Learning House. For permission requests, write to The Learning House, addressed “Attention: Permissions Coordinator,” at the address below.

The Learning House

427 S. 4th Street #300

Louisville KY 40202

Arrays in C#

.NET Cohort

Coding Bootcamp

SOFTWARE GUILD

Definitions

- An array is a set of uniform data elements represented by a variable
- Elements are individual data items. All elements must be of the same type.
- Rank/dimensions are the number of positions available.
- Dimension Length is the number of positions in a direction.
- Array Length is the total number of elements across all dimensions.

SOFTWARE GUILD

Array Types

- C# supports two types of arrays:
  - One-dimensional arrays are a single line, or vector, of elements.
  - Multi-dimensional arrays are composed such that each position in the primary vector is also an array, or subarray. Positions in subarrays can also be subarrays.
    - Multi-dimensional arrays can be rectangular (all subarrays the same length) or jagged (subarrays are different lengths)
- Arrays are reference types whose elements are automatically initialized.

SOFTWARE GUILD

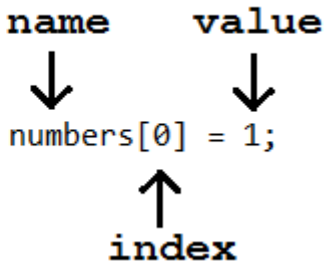
Important Things to Know

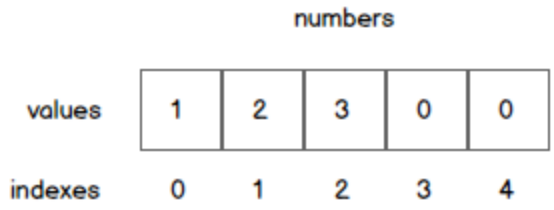
- After an array is created, its size is fixed (some languages allow for dynamic arrays, but C# does not).
- Array positions are referred to as *indexes*. Indexes start at 0.
- Arrays are class objects, and must be instantiated before you use them.

SOFTWARE GUILD

Creating and Populating a One-Dimensional Array

```
int[] numbers = new int[5];

numbers[0] = 1;
numbers[1] = 2;
numbers[2] = 3;
```





SOFTWARE GUILD

One-Dimensional and Rectangular Arrays

- To declare a one-dimensional or rectangular array use square brackets after the type
  - Ex: int[] myIntArray; // array of ints
- Rank Specifiers are commas between the brackets. They are used to specify the number of dimensions
  - Ex: int[,] arr2; // a 2D array of ints
  - Ex: int[,,] arr3; // a 3D array of ints
- Arrays must be instantiated with the length of each dimension
  - Ex: int[] arr4 = new int[5]; // an int array with 5 positions

SOFTWARE GUILD

Creating and Populating a Two-Dimensional Array

```
int[,] numbers = new int[2,5];

numbers[0,1] = 1;
numbers[0,2] = 2;
numbers[0,3] = 3;
numbers[1,0] = 5;
numbers[1,2] = 6;
numbers[1,4] = 7;
```

numbers

1	5	6	0	0	7
0	1	2	3	0	0
indexes	0	1	2	3	4

SOFTWARE GUILD

Accessing Array Elements

To access an individual element for reading or assigning values, refer to it by index.

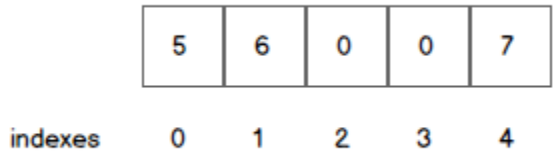
```
int[] intArr1 = new int[15];     // Declare a 1D array of 15 elements
intArr1[2] = 10;                 // Write to element 2 of the array
int var1 = intArr1[2];           // Read from element 2 of the array

int[,] intArr2 = new int[5,10]; // Declare a 2D array
intArr2[2, 3] = 7;               // Write to position 2 dimension 3
var1 = intArr2[2, 3];            // Read from position 2 dimension 3
```

SOFTWARE GUILD

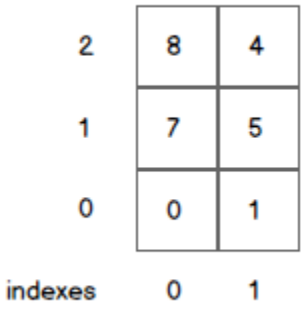
Explicit Initialization

Pre-populates the values during initialization

```
int[] numbers = new int[]
{
   5,6,0,0,7
};
```



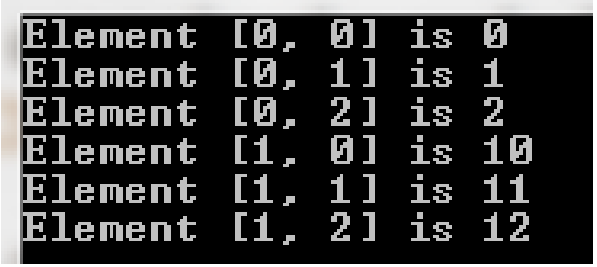

```
int[,] numbers = new int[,]
{
    {0,1}, {7,5}, {8,4}
};
```



SOFTWARE GUILD

Iterating an Array

```
//Declare, create, and initialize an implicitly typed array.
int[,] arr = new int[,] {{ 0, 1, 2}, {10, 11, 12}};

// Print the values.
for( int i = 0; i < 2; i++)
    for(int j = 0; j < 3; j++)
        Console.WriteLine("Element [{0}, {1}] is {2}", i, j, arr[i,j]);
```



SOFTWARE GUILD

Jagged Arrays

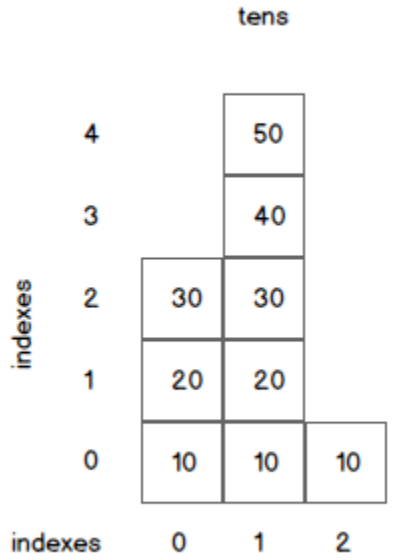
Jagged arrays have subarrays of different lengths.

Jagged arrays are declared using square brackets [] instead of an absolute dimension.

For example: “tens is an array of three arrays of ints”

Notice: in memory, there are four array objects. One is for the container and three for the subarrays.

```
int[][] tens = new int[3][];

tens[0] = new int[] { 10, 20, 30 };
tens[1] = new int[] { 10, 20, 30, 40, 50 };
tens[2] = new int[] { 10 };
```



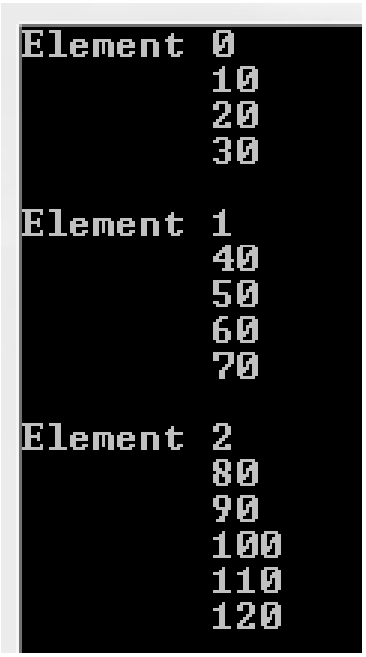
Now, it's Time to Cover foreach

- foreach is an awesome loop that executes a block of code once for each element of an array or other enumerable (lists, etc).
- The syntax is:
  - foreach(type variable in sourcevariable)

SOFTWARE GUILD

Using foreach to Output a Jagged Array

```
int[][] Arr = new int[3][];                     // 1. Instantiate top level.

Arr[0] = new int[] { 10, 20, 30 };              // 2. Instantiate subarray.
Arr[1] = new int[] { 40, 50, 60, 70 };          // 3. Instantiate subarray.
Arr[2] = new int[] { 80, 90, 100, 110, 120 };   // 4. Instantiate subarray.

int outercount = 0;

foreach(int[] element in Arr)
{
    Console.WriteLine("Element {0}", outercount);

    foreach (int subelement in element)
    {
        Console.WriteLine("\t{0}", subelement);
    }

    Console.WriteLine();
    outercount += 1;
}
```



SOFTWARE GUILD

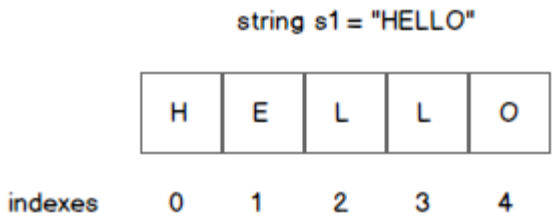
Useful Array Members

Member	Type	Lifetime	Usage
Length	Property	Instance	Gets the number of elements in **all dimensions** of an array
Rank	Property	Instance	Gets the total dimensions in an array
GetLength(n)	Method	Instance	Gets the number of elements in a specified dimension (n)
Clear()	Method	Static	Resets a range of elements to the default type (0 for numeric, null for reference types)
Sort()	Method	Static	Sorts all elements in a one-dimensional array
Reverse()	Method	Static	Reverses the order of the elements

SOFTWARE GUILD

Remember I Mentioned Strings Are Arrays?

```
string s1 = "This is a string of characters";

foreach (char c in s1)
{
    Console.WriteLine(c);
}

Console.WriteLine("The character at position 3 is {0}", s1[3]);
Console.WriteLine("The length of s1 is {0}", s1.Length);
```



SOFTWARE GUILD

Strings and arrays

DEMO

Write some code to read a string from the console and print it back in reverse order.

LAB EXERCISE

Write some code to roll a die 100 times and count the number of times each number comes up.

LAB EXERCISE