Arrays

Chapter 6

https://csharp.christiannagel.com

Simple Arrays

```
int[] myArray;  // array declaration
int[] myArray = new int[4]; // allocate array

// initialize array
int[] myArray1 = new int[4] {4, 7, 11, 2};
int[] myArray2 = new int[] {4, 7, 11, 2};
int[] myArray3 = {4, 7, 11, 2};
```

```
int v1 = myArray1[0];
MyArray[3] = 44;
```

Multidimensional Arrays

- Two-dimensional

```
int[,] twodim =
{
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9}
};
```

- Three-dimensional

```
int[,,] threedim =
{
  { { 1, 2 }, { 3, 4 } },
  { { 5, 6 }, { 7, 8 } },
  { { 9, 10 }, { 11, 12 } }
};
```

Jagged Arrays

- Not rectangular

```
int[][] jagged =
{
  new[] { 1, 2 },
  new[] { 3, 4, 5, 6, 7, 8 },
  new[] { 9, 10, 11 }
};
```

Array Class

Array class is used behind the scenes using arrays

Create arrays:

- Array.CreateInstance

Create shallow copy

- Array.Clone (ICloneable)

Sorting

- IComparable<T>
- IComparer<T>
- Delegate

Enumerators

foreach uses enumerators

IEnumerator<T>, IEnumerable<T> Interfaces

Yield Statement

- Create Enumerator

```
class HelloCollection
{
  public IEnumerator<string> GetEnumerator()
  {
    yield return "Hello";
    yield return "World";
  }
}
```

Span with Arrays

- Access continuous managed or unmanaged memory
- Span<T>

```
Span<int> IntroSpans()
{
  int[] arr1 = { 1, 4, 5, 11, 13, 18 };
  Span<int> span1 = new(arr1);
  span1[1] = 11;
  Console.WriteLine($"arr1[1] is changed via span1[1]: {arr1[1]}");
  return span1;
}v
```

Slices

- Slice of a Span

```
int[] arr2 = { 3, 5, 7, 9, 11, 13, 15 };
Span<int> span2 = new(arr2);
Span<int> span3 = new(arr2, start: 3, length: 3);
Span<int> span4 = span1.Slice(start: 2, length: 4);
```

Indices and Ranges

- Index struct
- Hat Operator ^
- Range struct
- Range Operator ..

Indices and Ranges with Custom Collections

Index

- Count or Length
- Indexer

Range

- Slice Method with int start, int length Parameters

Summary

- Arrays
- Multidimensional Arrays
- Jagged Arrays
- Enumerators
- yield Statement
- Span
- Indices and Ranges