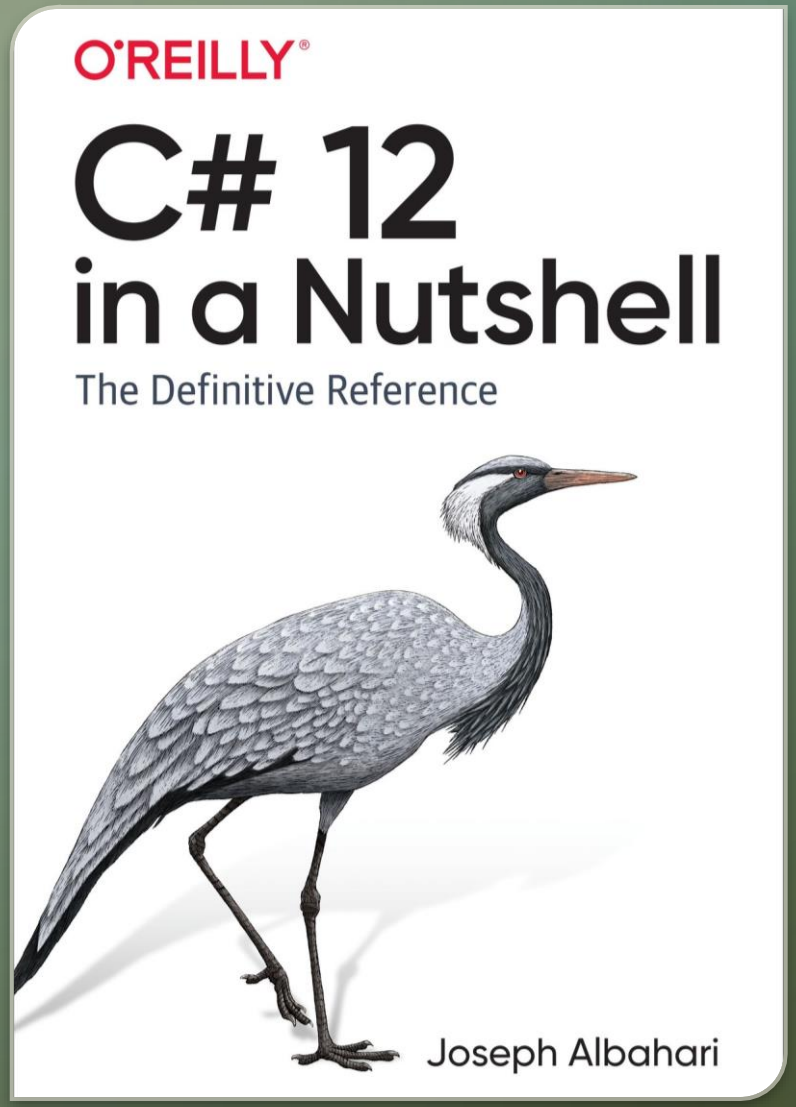
Chapter 7 - Arrays

- Construction and Indexing
- Enumeration
- Length and Rank
- Searching
- Sorting
- Reversing Elements
- Copying
- Converting and Resizing



Construction and Indexing

راحت‌ترین راه برای تعریف آرایه استفاده از ساختارهای زبان C# هست:

```
int[] myArray = { 1, 2, 3 };
int first = myArray[0];
int last = myArray[myArray.Length - 1];
```

راه دیگه برای تعریف آرایه استفاده از دستور Array.CreateInstance هست که هم امکان تعیین نوع اعضای آرایه و هم ابعاد اون رو فراهم میکنه.

متدهای GetValue و SetValue به ما اجازه میده که به آیتمهای یک آرایه دسترسی داشته باشیم.

Construction and Indexing

به راحتی میشه آرایه ها رو به نوع هایی که قابل تبدیل هستند تبدیل کرد:

```
// Create a string array 2 elements in length:
Array a = Array.CreateInstance(typeof(string), 2);
a.SetValue("hi", 0);                              //  → a[0] = "hi";
a.SetValue("there", 1);                           //  → a[1] = "there";
string s = (string)a.GetValue(0);                 //  → s = a[0];

// We can also cast to a C# array as follows:
string[] cSharpArray = (string[])a;
string s2 = cSharpArray[0];
```

مثلا اگه یه کلاس داشته باشیم به اسم Apple که از کلاس Fruit مشتق شده باشه، Apple[] میتونه به Fruit[] تبدیل بشه.

Construction and Indexing

این مثال یه متد رو تعریف میکنه که آیتم اول آرایه (یک بعدی یا چند بعدی) رو برمیگردونه:

```
2 references | - changes | -authors, -changes
void WriteFirstValue(Array a)
{
    Console.Write(a.Rank + "-dimensional; ");

    // The indexers array will automatically initialize to all zeros, so
    // passing it into GetValue or SetValue will get/set the zero-based
    // (i.e., first) element in the array.

    int[] indexers = new int[a.Rank];
    Console.WriteLine("First value is " + a.GetValue(indexers));
}

1 reference | - changes | -authors, -changes
void Demo()
{
    int[] oneD = { 1, 2, 3 };
    int[,] twoD = { { 5, 6 }, { 8, 9 } };

    WriteFirstValue(oneD);   // 1-dimensional; first value is 1
    WriteFirstValue(twoD);   // 2-dimensional; first value is 5
}
```

Enumeration

```
int[] myArray = { 1, 2, 3 };
foreach (int val in myArray)
    Console.WriteLine(val);

Array.ForEach(myArray, (a) => Console.WriteLine(a));
Array.ForEach(new[] { 1, 2, 3 }, Console.WriteLine);
Array.ForEach([1, 2, 3], Console.WriteLine);
```

```
void Array.ForEach<int>(int[] array, Action<int> action)
Performs the specified action on each element of the specified array.
action: The Action<in T> to perform on each element of array.
```

آرایه‌ها به راحتی میتونند با foreach استفاده بشن

همینطور میتونیم از متد استاتیک Array.Foreach استفاده کنیم. این متد دو ورودی دارد. ورودی اول آرایه و ورودی دوم یک Action هست.

(اگه Action رو نمیشناسید پست مربوط به delegate ها رو یه نگاه بندازید)

Length and Rank

متد GetLength و GetLongLength اندازه یک بعد آرایه رو برمیگردونه، پراپرتی‌های Length و LongLength مجموع تعداد آیتم‌های همه ابعاد آرایه رو برمیگردونه.

```
string[,,] s3 = new string[5, 4, 8];
Console.WriteLine($"{nameof(s3.Length)} -> {s3.Length}");                 //160
Console.WriteLine($"{nameof(s3.LongLength)} -> {s3.LongLength}");         //160
Console.WriteLine($"{nameof(s3.GetLength)} -> {s3.GetLength(1)}");        //4
Console.WriteLine($"{nameof(s3.GetLongLength)} -> {s3.GetLongLength(2)}"); //8
Console.WriteLine($"{nameof(s3.GetLowerBound)} -> {s3.GetLowerBound(2)}"); //0
Console.WriteLine($"{nameof(s3.GetUpperBound)} -> {s3.GetUpperBound(2)}"); //7
```

Searching

کلاس Array برای پیدا کردن یک عنصر در یک بعد از آرایه راهکارهای مختلفی رو ارائه میده:

- BinarySearch methods
  برای جستجوی سریع یک عنصر در یک آرایه مرتب
- IndexOf/LastIndex methods
  برای جستجوی یک عنصر در آرایه نامرتب
- Find/FindLast/FindIndex/FindLastIndex/FindAll/Exists/TrueForAll
  برای پیدا کردن یک یا چند عنصر در آرایه با دستورات شرطی

Searching

هیچکدوم از روش‌های جستجو در صورت پیدا نکردن آیتم مدنظر خطا نمیدند، در عوض اگر آیتمی پیدا نکنند مقدار ۱- برمیگردونند.

متد binarySearch سرعت بالایی داره ولی فقط میتونه با آرایه‌های مرتب کار کنه و نیاز داره که هر آیتم بتونه با آیتم قبلی مقایسه بشه. برای همین میتونه یه Icomparer به عنوان ورودیش بگیره. اگه این ورودی رو بهش ندیم با الگوریتم پیش‌فرض خودش جستجو رو انجام میده.

```
Console.WriteLine(Array.BinarySearch([12, 5, 14, 7], 0, 2, 5));
```

Searching

متدهای indexOf و lastIndexOf به ترتیب اولین و آخرین ایندکس آیتم پیدا شده رو برمیگردونه.

```
Console.WriteLine(Array.IndexOf(["a", "b", "5", "c", "d", "5"], "5"));      //2
Console.WriteLine(Array.LastIndexOf(["a", "b", "5", "c", "d", "5"], "5"));  //5
```

برای استفاده از توابعی مثل Find یک متد Predicate<T> باید به عنوان ورودی بهش پاس بدیم.

```
public delegate bool Predicate<T> (T object);
```

```
Console.WriteLine(string.Join(",", Array.FindAll(["a", "b", "5A", "c", "d", "5"], (a) => { return a == "5"; })));  //5,5

#endregion
```

```
string[] Array.FindAll<string>(string[] array, Predicate<string> match)
Retrieves all the elements that match the conditions defined by the specified predicate.
match: The Predicate<in T> that defines the conditions of the elements to search for.
```

Sorting

کلاس Array یک متد built-in داره برای مرتب کردن عناصر خودش.

این متد فقط برای یک بعد آرایه کاربرد داره و هرنوعی که IComparable رو پیاده سازی کرده باشه میتونه سورت کنه.

```
int[] arr = [ 2, 5, 8, 1 ];
Array.Sort(arr);

Console.WriteLine(string.Join(",", arr));  //1,2,5,8

numbers = [3, 2, 1];
string[] words = { "three", "two", "one" };
Array.Sort(numbers, words);

// numbers array is now { 1, 2, 3 }
// words   array is now { "one", "two", "three" }
```

Reversing Elements

با استفاده از متد reverse میتونیم عناصر یک آرایه رو برعکس کنیم:

```
Array.Reverse(arr);
Console.WriteLine(string.Join(",", arr));  //8,5,2,1
```

Copying

برای کپی کردن یک آرایه چهار متد زیر وجود دارد:

Clone, Copy, CopyTo, ConstrainedCopy

```
int[,] int3X3 = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };
int[] int3 = [1, 5, 6];
var clone = int3X3.Clone();

int[] copy3 = new int[3];
int3.CopyTo(copy3, 0);

int[,] copy3X3 = new int[3, 3];
Array.Copy(int3X3, copy3X3, 5);

Array.ConstrainedCopy(int3X3, 0, copy3X3, 3, 5);
```

- هر چهار متد shallow-copy انجام میدهند.

Converting and Resizing

متد ConvertAll میتونه همه آیتمهای یک آرایه رو به یک نوع دیگر تبدیل کند.

```
float[] reals = { 1.3f, 1.5f, 1.8f };
int[] wholes = Array.ConvertAll(reals, r => Convert.ToInt32(r));

// wholes array is { 1, 2, 2 }

Array.Resize(ref wholes, 5);
Console.WriteLine(string.Join(",", wholes));  // { 1, 2, 2, 0, 0 }
```

همینطور با استفاده از متد resize میتونیم اندازه یک آرایه رو تغییر بدیم.