Contents

Advanced Arrays

Value and Reference Types

Here, we will explore the difference between value and reference types. Since arrays are reference types, it is important for you to understand how reference types work.

Let us show why this notion is so critical with an example:

```
int[] arrayA = { 1, 2, 3, 4, 5 };  // Declare a simple
 ↪  array of integers

// I'd like to make a copy of that array. Let me try the
 ↪  following:
int[] arrayCopyWrong = arrayA;

foreach (int i in arrayCopyWrong)
    Console.Write(i + " ");

Console.WriteLine();

// It seems to be working! Except that if we change a
 ↪  value in our copy:
arrayCopyWrong[0] = 6;

// It also changes the value in our original array!
foreach (int i in arrayA)
    Console.Write(i + " ");

Console.WriteLine();
```

Try executing this program yourself to see what happens. The problem is that when we wrote the assignment statement `int[] arrayCopyWrong = arrayA`, we copied the *reference* to the array, but not the array itself. We now have two ways of accessing our array, using `arrayA` or `arrayCopyWrong`, but still only one array.

To correctly copy the array, we need to do something like the following:

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

```
// Create a new array object and assign it to a new
 ↪  reference variable
int[] arrayCopyRight = new int[arrayB.Length];

// Copy each value in the array, one by one:
for(int i = 0 ; i < arrayB.Length; i++)
    arrayCopyRight[i] = arrayB[i];

// If we change a value in our copy:
arrayCopyRight[0] = 6;

// It changes the value only in that copy:
foreach (int i in arrayB)
    Console.Write(i + " ");

Console.WriteLine();

foreach (int i in arrayCopyRight)
    Console.Write(i + " ");

Console.WriteLine();
```

Try executing this program. Can you see the difference?

`Array` is actually a class (documented at https://msdn.microsoft.com/en-us/library/system.array(v=vs.110).aspx), and as such provides several methods. If you have two arrays, `array1` and `array2` containing the same type of values and of size at least `x`, you can copy the first `x` values of `array1` into `array2` using `Array.Copy(array1, array2, x);`. Try using this method with the previous example to create a copy of `arrayB`.