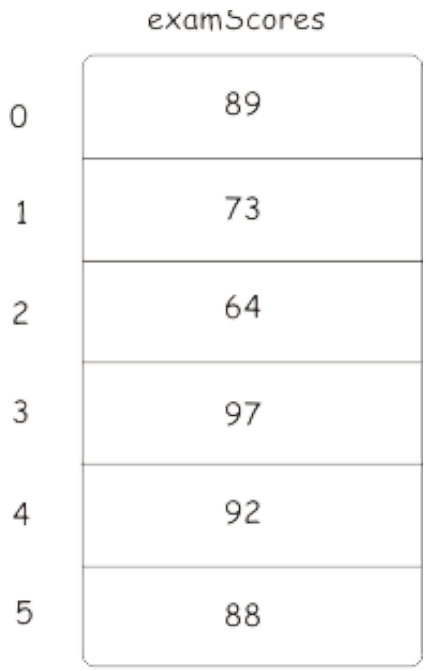
CS 1400 Lab #23: Using Arrays

Objectives:

The objective of this lab is to help you better understand how to use arrays in your programs. After completing this lab, you should be able to write a simple program that uses arrays

Arrays



An array is simply a way of organizing a list or table of data in the computer's memory so that it is efficient to process the data. In most programming languages, all of the elements of an array must be of the same kind of data, for example all integers or all doubles. The figure to the left illustrates an array of integer values. An important property of arrays is that elements of the array are stored one after the other in memory. This makes it very fast to access data in an array. Each element of the array has an *index* which indicates its position in the array. The first element of an array is always at index zero. So, in the figure to the left, we have an array of six elements, with indices zero through five. To access any element of the array we use the name of the array, followed by its index inside a set of square brackets. Thus, for example, the third element of this array can be accessed by the term

```
examScores[2]
```

To store a value in the third element of this array we would write something like this:

```
examScores[2] = 64;
```

And to get the value store in that element of the array we would write something like

```
int oneScore = examScores[2];
```

Declaring an Array

To declare an array, we have to declare the type of data that will be stored in the array as well as the size of the array. The array declaration for our examScores example would look something like

```
const int SIZE = 6;
int[ ] examScores = new int[SIZE];
```

The variable *examScores* is a reference variable. That is, it is not the array itself, but rather it is a reference to the array. The array was created on the heap by the *new* operator. In C#, arrays are objects.

Processing Arrays with Loops

The most common way of processing the data that is in an array is with a loop. Consider the following loop, which will display all of the values in the *examScores* array:

```
const int SIZE = 6;
 .  .  .
for (int i = 0; i < SIZE; i++)
{
   Console.WriteLine(examScores[ i ]);
}
```

Writing Methods That Take Arrays as Parameters

Recall that an array is a sequential list of values, and that the name of the array is really a reference to the array object. Often times we want to write methods that operate on arrays, and so the array has to be passed to the method as a parameter. We indicate that a parameter is an array be using the square brackets [ ] when we write down the data type of the parameter. For example, consider this method that outputs all of the elements of an array:

```
static void PrintArray(int[ ] numbers)
{
   for (int i = 0; i < numbers.Length; i++)
   {
      Console.WriteLine(numbers[ i ]);
   }
}
```

The Length Property

Take careful note of the line in the code above that reads

```
for (int i = 0; i < numbers.Length; i++)
```

Array objects have a property called *Length* that returns the size of the array. When using a loop to process an array, we need to know how big the array is. The *Length* property conveniently provides this for us.

The foreach Loop

C# has a unique looping construct called a *foreach* loop. A foreach loop can be used to access all of the elements of an array in sequence. Note that we can only use a foreach loop when reading the elements of an array. You cannot use a foreach loop to change the elements of an array. Writing the printArray method above using a foreach loop we have

```
static void PrintArray(int[ ] numbers)
{
   foreach(int value in numbers)
   {
      Console.WriteLine(value);
   }
}
```

Programming Exercise

For this assignment, write a short program that does the following:

1. Creates an integer array of ten elements.
2. Uses a loop to fill the array by
    - Prompting the user for a value
    - Storing the value in the array
3. Passes the array to a method that adds up all of the elements in the array and returns the sum. Your method should work for an array of any size.
4. Outputs the sum.

File(s) to Submit:

Place your complete project folder in a zip file and name the zip file lab_23_your-initials_V1.0.zip. For example, I would name my file lab_23_RKD_V1.0.zip. Submit this assignment as Lab #23 on Canvas.

Grading Guidelines

Description	**Points possible**
Assignment meets grading guidelines: o Source code files contain a declaration that you did not copy any code, except that provided. o Assignment has been properly submitted to Canvas	2

o Code meets style guidelines o Code contains a Console.ReadLine( ) statement at the end	
Program contains a Sum method that takes an array as a paremeter.	2
Program executes correctly and meets the specification.	1
Total	5