|  |
| --- |
| **CS1400**  **Lab #21**  **Using Arrays**  **Version 1.0**  **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**  http://debryro.tc.uvu.edu/1400/labs/lab21/array.pngAn array is simply a way of organizing a list or table of data in the computer's memory, so that it can be efficiently processed using a single name. In most programming languages, all of the elements of an array must be of the same kind of data, for example all integers, all doubles, all strings, etc.. Figure 1 illustrates an array of integer values. An important property of arrays is that elements of the array are stored consecutively, one after the other in memory. This makes it very fast to access data in an array.  Figure int 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 1 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 (index zero) 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 first element of 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 consecutive 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 idx = 0; idx < numbers.Length; idx++)       {          Console.WriteLine(numbers[ idx ]);       }    }**  **The Length Property**  Take careful note of the line in the code above that reads      for (int idx = 0; idx < numbers.Length; idx++)  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 in the following manner:  * Uses a loop to fill the array by   + Prompting the user for a value   + Storing the value in the array   + Terminates user input when a -1 is entered by the user * Passes the array to a method that: * Creates a new array and adds 5 to each element. * adds up all of the elements in the array passed into the method and returns the sum, via an out parameter. * Returns the reference to the new array in the ***eax Register***. * NOTE: Your method should work for an array of any size. * Outputs the returned sum and the returned array.   **File(s) to Submit:**  Place your complete project folder in a zip file and name the zip file Lab\_21\_your-initials\_V1.0.zip. For example, I would name my file Lab\_21\_DAF\_V1.0.zip. Submit this assignment as Lab #21 on Canvas. |

|  |  |  |
| --- | --- | --- |
| # | **Grading Checklist** | C(correct)  X(incorrect) |
|  | Program |  |
| 1 | Meets & works to specifications | 6 points |
| 2 | Error Free, elegant & efficient | 4 points |
| 3 | Pseudo-Code | -3 points |
| 4 | Style Guidelines | -2 points |
| 6 | Source Files(s) & Formatting | -2 points |
| 7 | Project Prolog | -1 points |
| 8 | Function Prologs | -1 points |
| 9 | Zip Filename | -1 points |
| 10 | Lab & Project Names | -1 points |
| 11 | Zip File is invalid or will not unzip | Lab = 0 pts |
|  | Total Points | 10 | 0-9 |