Arrays

An **array** is a set of fixed number of values called *elements* that can be accessed using integer index. **Elements** on an array are of the same data type. Indices in an array starts at 0. An array is used to store multiple values of the same data type at a time without declaring a different variable name for each value. Array elements can be of any data type.

One-Dimensional Arrays

A **one-dimensional array** is an array in which all elements are arranged like a list. In C#, arrays are implemented as objects. To define an array, declare a variable first that refers to an array, followed by creating an instance of the array through the **new** operator. The following general syntax defines a one-dimensional array:

```
data_type[] array_name = new data_type[array_size];
```

The `data_type` refers to the type of data that will be stored in the array, and the `array_size` in square brackets specifies the fixed number of elements can be stored in the array. The new operator creates the array and initializes the array elements to their default values. The following example of one-dimensional array that can store 10 integers and all the array elements are initialized to zero:

```
int[] numbers = new int[10];
```

The following example shows how to assign values to individual array elements by using an index number:

```
numbers[0] = 45;
numbers[4] = 23;
```

The example below shows how to initialize an array upon declaration:

```
double[] grades = { 2.50, 2.75, 1.25, 5.0, 1.50 };
```

In the given example, the declared array `grades` is initialized with five (5) values placed between braces and separated by commas. Array elements can be accessed by using index number. The following example accesses the element of array `grades` by placing the index of the element on the square brackets after the name of the array. It gets the array element of index `1` which is `2.75` then copies its value and stores it on the variable `studentGrade`:

```
double studentGrade = grades[1];
Console.WriteLine(studentGrade); //prints 2.75
```

The **`foreach`** statement in C# is used to process the elements of an array starting with index `0` up to the ending index. This iterates through each item of the array. The following example shows how to use a `foreach` statement to access the all the elements of the array `grades`:

```
foreach (double grade in grades)
{
        Console.Write(grade + " / ");
}
```

Output:

```
2.5 / 2.75 / 1.25 / 5 / 1.5 /
```

In the given example of `foreach` loop, the `in` keyword is used to iterate over the elements of the array. This gets an item from the array on each iteration and stores in on the declared variable—in the given example—`grade`. The number of times the `foreach` loop will execute is equal to the number of elements in the array. In the given example, the `foreach` loop iterates five (5) times.

Two-Dimensional Arrays

In a **two-dimensional array**, the array elements are arranged in rows and columns. The elements in this array are arranged in a tabular form and are therefore stored and accessed by using two (2) indices: one (1) index refers to the row, and the other refers to the column location. The following is the general syntax for defining a two-dimensional array:

```
data_type[,] array_name = new data_type[row_size, col_size];
```

The code `data_type[,]` defines the two-dimensional array being defined. The `row_size` and `col_size` refer to the number

of rows and columns in the array, respectively. The following example defines a two-dimensional array consisting of two (2) rows and four (4) columns:

```
int[,] table = new int[2, 4];
```

The example below shows how to assign values to individual array elements by specifying the row and column numbers:

```
table[0, 1] = 18;
table[1, 3] = 4;
```

The following example shows how to initialize a two-dimensional array upon declaration:

```
int[,] table = {
      { 2, 3 },
      { 12, 5},
      { 3, 8 },
      { 18, 3 }
};
```

The following shows how to access the element of a two-dimensional array:

```
Console.WriteLine(table[3, 0]); //this prints the 18
```

The *ArrayList* Class

The **ArrayList** class in C# is a collection that behaves as a dynamic array where the array size can dynamically increase as required. The ArrayList class is defined in the **System.Collections** namespace. The following is the general syntax for defining an array list. (***Note:*** The namespace System.Collections must be declared first in order to use the ArrayList class.)

```
ArrayList name_of_list = new ArrayList();
```

The following example shows how to create an ArrayList:

```
ArrayList nameList = new ArrayList();
```

The **Add()** method is used to add an element to the list. The following is the example of adding elements on the ArrayList named nameList:

```
nameList.Add("Jack Paul");
nameList.Add("Adrian Castro");
nameList.Add("Peter Cruz");
nameList.Add("Angela Cruz");
```

The elements of an ArrayList can be accessed by specifying an integer index. For example:

```
Console.WriteLine(nameList[2]); //this will print Peter Cruz
```

The **foreach** loop is used to process all the elements of an ArrayList. For example:

```
foreach (string name in nameList)
{
      Console.Write(name + ", ");
}
```

Output:

```
Jack Paul, Adrian Castro, Peter Cruz, Angela Cruz,
```

REFERENCES:

Deitel, P. and Deitel, H. (2015). *Visual C# 2012 how to program* (5th Ed.). USA: Pearson Education, Inc.

Gaddis, T. (2016). *Starting out with visual C#* (4th Ed.). USA: Pearson Education, Inc.

Harwani, B. (2015). *Learning object-oriented programming in C# 5.0*. USA: Cengage Learning PTR.