Chapter 21

Arrays

- ❖ Learning how to create arrays in VB2010

21.1 Introduction to Arrays

By definition, an array is a list of variables with the same data type and name. When we work with a single item, we only need to use one variable. However, if we have a list of items, which are of similar type to deal with, we need to declare an array of variables instead of using a variable for each item

For example, if we need to enter one hundred names, it is difficulty to declare 100 different names; this is a waste of time and efforts. Therefore, instead of declaring one hundred different variables, we need to declare only one array. We differentiate each item in the array by using subscript, the index value of each item, for example name(0), name(1),name(2) .....etc. , which will make declaring variables streamline and much systematic.

21.2 Dimension of an Array

An array can be one dimensional or multidimensional. One-dimensional array is like a list of items or a table that consists of one row of items or one column of items. Table 21.1 shows a one-dimensional array.

Student Name	Name(0)	Name(1)	Name(2)	Name(3)	Name(4)	Name(5)

Table 21.1 One-dimensional Array

A two dimensional array is a table of items that make up of rows and columns. The format for a one-dimensional array is ArrayName(x), the format for a two dimensional array is ArrayName(x, y) and a three dimensional array is ArrayName(x, y, z). Normally it is sufficient to use one-dimensional and two-dimensional arrays; you only need to use higher dimensional arrays if you need to deal with problems that are more complex.

21.3 Declaring an Array

We can use Public or Dim statement to declare an array just as the way we declare a single variable. The Public statement declares an array so that it can be used throughout the entire application while the Dim statement declares an array that can be used only in a local procedure.

21.3.1 Declaring One Dimensional Array

The general format to declare a one-dimensional array is as follow:

```
Dim arrayName(subs) as dataType
```

The argument subs indicates the last subscript in the array.

Example 21.1

```
Dim CusName(9) as String
```

declare an array that consists of 10 elements starting from CusName(0) to CusName(9).

Example 21.2

```
Dim Count (100 to 500) as Integer
```

The statement above declares an array that consists of the first element starting from Count (100) and ends at Count (500)

Example 21.3: Creating a Name List

In this program, we want let the user create a name list by entering name into a list box. At runtime, the user will be prompted to enter ten student names. The names entered will appear in a list box. First, start a new project and name it Student Data. Next, insert a list box and a button into the form. Change properties of the controls as follows:

Control	Properties
Form1	Name: StudentList Text: Student List
ListBox	Name: NameList
Control1	Name: BtnAdd Text: Add Name

Table 21.3

Next, click the button and key in the following code:

```
Private Sub BtnAdd_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BtnAdd.Click
    Dim studentName(9) As String
    Dim num As Integer
    For num = 0 To 9
       studentName(num) = Microsoft.VisualBasic.InputBox("Enter a name and
        Click OK", "Names Entry Form", "Enter name here", 100, 200)
       NameList.Items.Add(studentName(num))
    Next

  End Sub
```


When you press F5 and run the program, you will see a popup dialog box where you can enter a name. After you have entered the name and click Ok, the same dialog box will appear again for you to enter the second student name. The process will repeat ten times. The dialog box is as shown in Figure 21.1

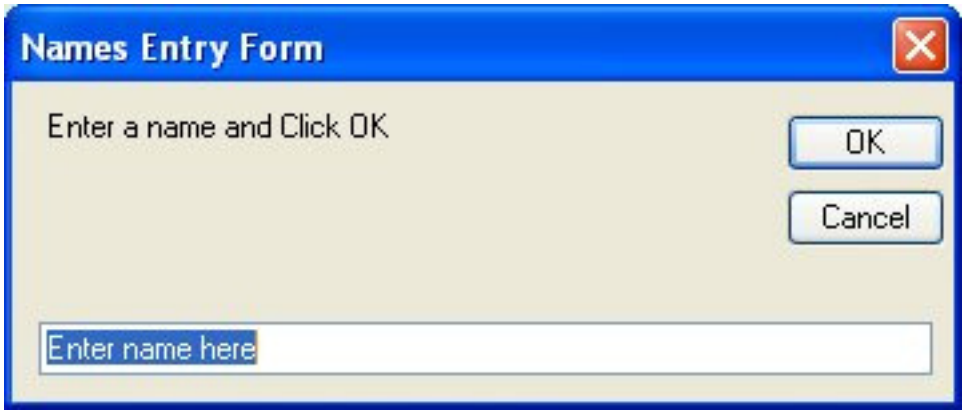

Figure 21.1

After entering ten names, you can see the ten names appear on the list box, as shown in Figure 21.2

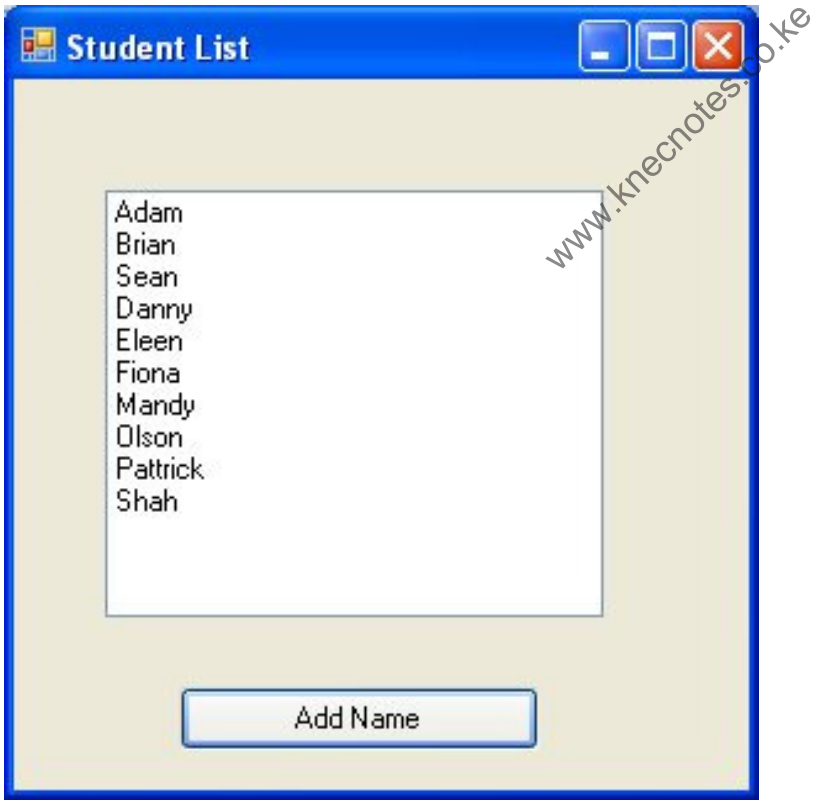

Figure 21.2

21.3.1 Declaring Two Dimensional Array

The general format to declare a two dimensional array is as follow:

```
Dim ArrayName(Sub1,Sub2) as dataType
```

Total number of elements will be (sub1+1)x(sub2+1). For example,

Dim Score(2,3) will produce an array that comprises 3x4=12 elements, as shown in Table 21.2

Score(0,0)	Score(0,1)	Score(0,2)	Score(0,3)
Score(1,0)	Score(1,1)	Score(1,2)	Score(1,3)
Score(2,0)	Score(2,1)	Score(2,2)	Score(2,3)

Table 21.2 Two Dimensional Array

Example 21.4: Managing Students' Examination Scores

In this example, we want to key in the examination marks for five students and four subjects. Since we are handling two variables here, i.e. name and subject, we need to declare a two dimensional array, as follows:

```
Dim score (4, 3) as String
```

The first dimension represents student names and the second dimension represents the subjects. Combining both produces the scores for each student for each subject. For example, the score for the first student for the first subject will be score (0, 0). We can design a program to let the user enter the student names, subject titles as well as the scores. We need to use two nested loops involving the For...Next structure. The first loop gets the students' names and the second loop gets the students' scores for the four subjects. To achieve the purpose, we introduce a one dimensional array StudentName(4) to store the names of the five students. We also introduce a one dimensional array mark(3) to store the mark of every subject for every student .After entering the name of the first student and his scores, we get something like this:

Adam 45 60 56 80

The scores of students in array form are shown in Table 21.3

studentName(0)=Adam	Score(0,0)=45	Score(0,1)=60	Score(0,2)=56	Score(0,3)=80

Table 21.3; Scores for first students

The variable mark are assigned the values of the scores as shown in table 21.4

studentName(0)=Adam	mark(0)=45	mark(1)=60	mark(2)=56	mark(3)=80

Table 21.4: Score in terms of mark

The process repeats until the user has entered all the data. The completed data appear as a two-dimensional array, as shown in terms of scores in Table 21.5 and in terms of marks in Table 21.6.

studentName(0)=Adam	score(0,0)=45	score(0,1)=56	score(0,2)=78	score(0,3)=68
studentName(1)=Brian	score(1,0)=64	score(1,1)=76	score(1,2)=80	score(1,3)=90
studentName(2)=Florence	score(2,0)=87	score(2,1)=80	score(2,2)=90	score(2,3)=100
studentName(3)=Gloria	score(3,0)=45	score(3,1)=54	score(3,2)=34	score(3,3)=48
studentName(4)=Mandy	score(4,0)=56	score(4,1)=87	score(4,2)=68	score(4,3)=66

Table 21.5

studentName(0)=Adam	mark(0)=45	mark(1)=56	mark(2)=78	mark(3)=68
studentName(1)=Brian	mark(0)=64	mark(1)=76	mark(2)=80	mark(3)=90
studentName(2)=Florence	mark(0)=87	mark(1)=80	mark(2)=90	mark(3)=100
studentName(3)=Gloria	mark(0)=45	mark(1)=54	mark(2)=34	mark(3)=48
studentName(4)=Mandy	mark(0)=56	mark(1)=87	mark(2)=68	mark(3)=66

Table 21.6

In this program, we insert a list box and name it NameList. We also introduce a button and name it BtnAdd. Change the form title from Form1 to “Examination Scores”

Now click the button and enter the code. In the code, we declare studentName (4) and mark(3) as one-dimensional array

The code

```
Private Sub BtnAdd_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BtnAdd.Click
  Dim studentName(4) As String
      Dim score(4, 3) As String
      Dim mark(3) As String
      Dim num1, num2 As Integer
      For num1 = 0 To 4
         studentName(num1) = Microsoft.VisualBasic.InputBox("Enter a name and
         Click OK", "Names Entry Form", "Enter name here", 100, 200)

         For num2 = 0 To 3
            score(num1, num2) = Microsoft.VisualBasic.InputBox("Enter score and
           Click OK", "Scores Entry Form", "Enter Score here", 100, 200)
            mark(num2) = score(num1, num2)
         Next
         NameList.Items.Add(studentName(num1) & vbTab & mark(0) & vbTab &
         mark(1) & vbTab & mark(2) & vbTab & mark(3))

      Next
   End Sub

Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Load
     'To Label the the subjects' titles at the top of the list
     NameList.Items.Add("" & vbTab & "English" & vbTab & "Sience" & vbTab &
```

```
        "Math" & vbTab & "Art")
    'To draw a separation line between the subjects' titles and the scores
        NameList.Items.Add("" & vbTab & "----------------------------------------
        ----------")
End Sub
```

The output is shown in Figure 21.3

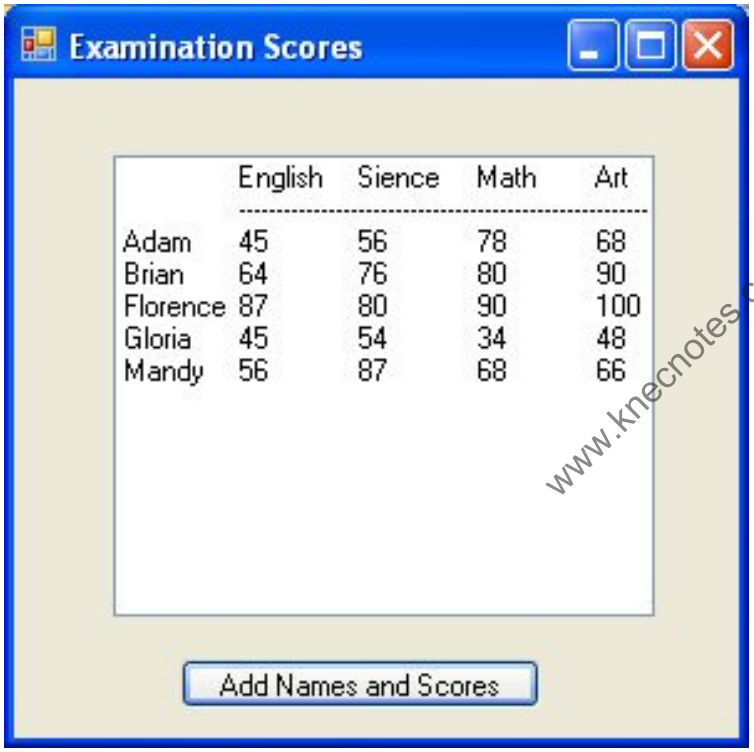

Figure 21.3

The above example has demonstrated the practical usage of arrays. If you wish to add more features to the program, you can modify the code easily, like writing the code to obtain the total mark and average mark.

Summary

- In this section 21.1, you learned that an array is a list of variables with the same data type and name
- In section 21.2, you learned about arrays of different dimensions.
- In section 21.3, you learned how to declare arrays of different dimensions.