MODULE-III

ARRAYS

INTRODUCTION

A group of homogeneous elements of a specific data type is known as an *array*, one of the simplest data structures.

Arrays hold a sequence of data elements, usually of the same size and same data type placed in contiguous memory locations that can be individually referenced. Hence arrays are essentially a way to store many values under the same name.

Individual elements are accessed by their position in the array. The position is given by an index, which is also called a *subscript*.

The index usually uses a consecutive range of integers. Some arrays are multi-dimensional, but generally, one- and two-dimensional arrays are the most common.

You can consider using arrays when you work with a collection of similar data and when you perform repetitive computations.

Arrays are ideal for storing data you collect from waveforms, graphs, data generated in loops or data collected from multiple sensors.

ARRAYS IN LABVIEW

In LabVIEW arrays group data elements of the same type. They are analogous to arrays in traditional languages.

An array consists of elements and dimensions. Elements are the data that make up an array. A dimension is the length, height or depth of an array.

An array can have one or more dimensions and as many as (231) – 1 element per dimension, memory permitting.

You can build arrays of numeric, Boolean, path, string and cluster data types. You cannot create arrays of arrays. However, you can use a multidimensional array or an array of clusters where each cluster contains one or more arrays. Also, you cannot create an array of subpanel controls, tab controls, .NET controls, ActiveX controls, charts or multiplot XY graphs.

Array elements are ordered. To locate a particular element in an array requires one index per dimension. For example, if you want to locate a particular element in a two-dimensional array, you need both row index and column index.

In LabVIEW, indexes let us navigate through an array and retrieve elements, rows and columns from an array on the block diagram. The index is zero-based, which means it is in the range 0 to n – 1, where n is the number of elements in the array with the first element at index 0 and the last one at index (n – 1). For example, if you create an array of 10 elements, the index ranges from 0 to 9. The fourth element has an index of 3

CREATING ONE-DIMENSIONAL ARRAY CONTROLS, INDICATORS AND CONSTANTS

Create an array control or indicator on the front panel by placing an array shell on the front panel as shown in Figure 5.1(a), and dragging a data object or element, which can be a numeric, Boolean, string, path, refnum, or cluster control or indicator, into the array shell. The array shell automatically resizes to accommodate the new object.

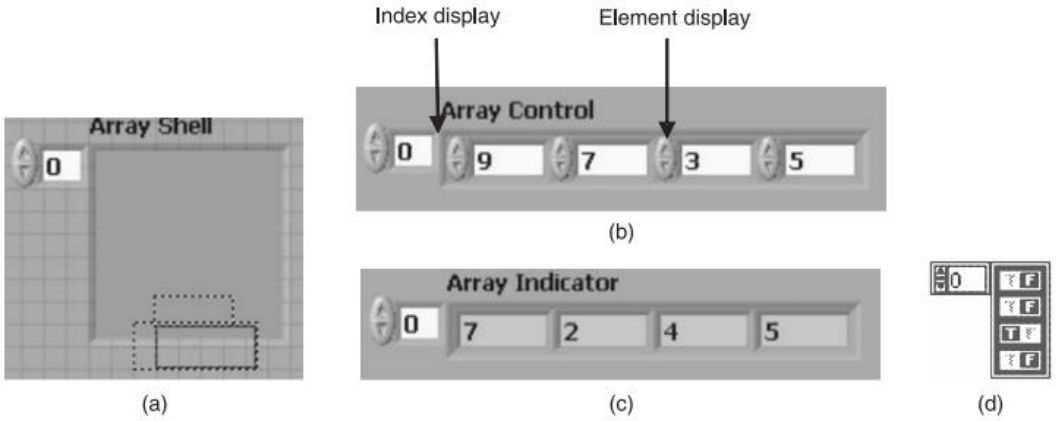


Figure 5.1 (a) Array shell placed on the front panel, (b) Array of numeric controls, (c) Array of numeric indicators and (d) Array of Boolean constants.

The array shell can be selected from Controls>>Modern>>Arrays, Matrix & Clusters palette.

The array elements must be controls orindicators. You must insert an object in the array shell before you use the array on the block diagram. Otherwise, the array terminal appears black with an empty bracket.

After placing an element in the array shell, you can expand the array either horizontally or vertically to see more number of elements. Once a data type is assigned to the array shell, the block diagram takes the color and lettering (in [ ] brackets) of the data type. For example, if the data type is a numeric indicator, the color will be orange with [DBL] written inside the terminal. Arrays can be identified easily by their thicker (or double) wires.

Figure 5.1(b) shows an array of numeric controls and the array has four elements. Figure 5.1(c) shows an array of numeric indicators.

The index ranges from 0 to 3. The first element in the array (9) is at index 0, the second element (7) is at index 1, the third element (3) is at index 2 and the fourth element (5) is at index 3.

In an array the element selected in the index display always refers to the element shown in the upper-left corner of the element display. In Figure 5.2, the element (9) at index 0 is not shown in the array, because index 1 is selected in the index display. If you want to see the element 9 again, the index value in the index display should be changed to 0.

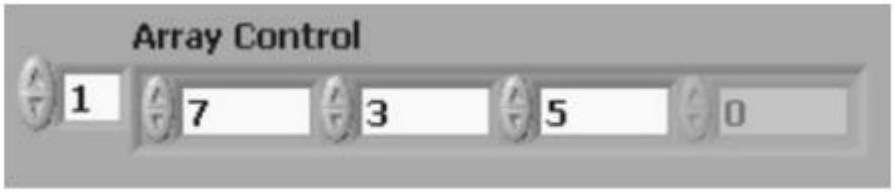


Figure 5.2 Array of numeric control with index 1 selected in the index display.

You can create an array constant on the block diagram by combining an array with a valid constant, which can be a numeric, Boolean, string, path, refnum, or cluster constant. The element cannot be another array. Following are the steps for creating an array constant in the block diagram.

1. Select an array constant from Functions>>Programming>>Arrays palette; place the array shell on the block diagram. The array shell appears, with an index display on the left, an empty element display on the right, and an optional label.
2. Place a constant in the array shell. For example, a numeric constant in the array shell.
3. The array shell automatically resizes to accommodate the object you place in the array shell. When an object is placed in the array shell, the data type of the array constant is defined.
4. An alternative method of creating array constant is to copy or drag an existing array on the front panel to the block diagram to create a constant of the same data type.

You can use an array constant to store constant data or as a basis for comparison with another array. Array constants are also useful for passing data into a subVI. Figure 5.1(d) shows an array of Boolean constants.

CREATING TWO DIMENSIONAL ARRAYS

A two-dimensional array is analogous to a spreadsheet or table.

A two-dimensional array stores elements in a grid. It requires a column index and a row index to locate an element, both of which are zero-based.

Two-dimensional arrays are very commonly used in data acquisition applications. For example, when waveforms from several channels are read from a data acquisition (DAQ) board, the data is stored in a two-dimensional array where each column in the array corresponds to data from one channel.

To create a two-dimensional array on the front panel, right-click the index display of the array and select Add Dimension from the shortcut menu. You also can use the Positioning tool to resize the index display to have one more dimension. Figure 5.3 shows a two-dimensional array

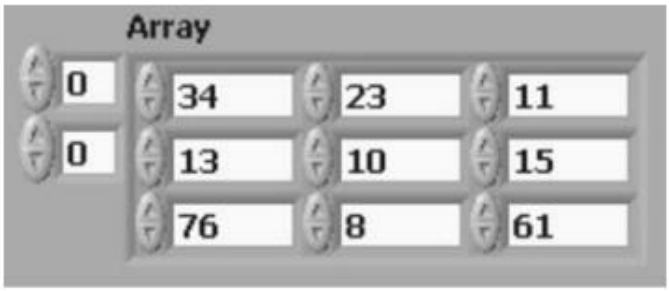


Figure 5.3 Two-dimensional array.

CREATING MULTI DIMENSIONAL ARRAYS

To create a multidimensional array on the front panel, right-click the index display of the array and select Add Dimension from the shortcut menu. You also can use the Positioning tool to resize the index display until you have as many dimensions as you want.

To delete dimensions one at a time, right-click the index display and select Remove Dimension from the shortcut menu.

You can also resize the index display to delete dimensions.

Use the Positioning tool to resize the array to show more than one row or column at a time.

To display a particular element on the front panel, either type the index number in the index display or use the arrows on the index display to navigate to that number.

You can also use the scroll bars of an array to navigate to a particular element. Right-click the array and select Visible Items»Vertical Scrollbar or Visible Items»Horizontal Scrollbar from the shortcut menu to display scroll bars for the array as shown in Figure 5.4.

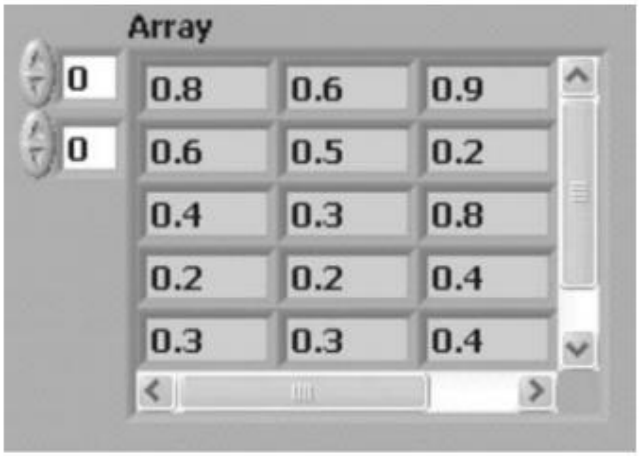


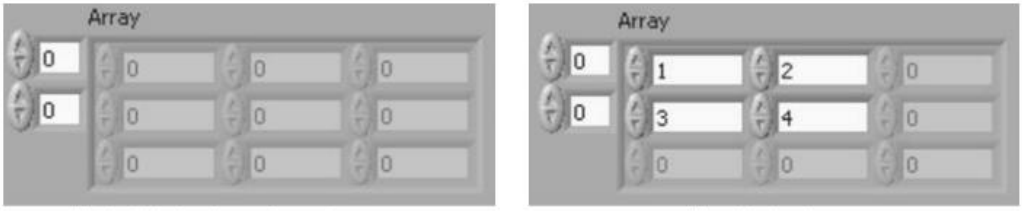
Figure 5.4 Array with horizontal and vertical scroll bars.

INITIALIZING ARRAY

You can initialize an array, or leave it uninitialized.

When an array is initialized, you can define the number of elements in each dimension and the contents of each element. An uninitialized array has a dimension but no elements.

Figure 5.5(a) shows an uninitialized two-dimensional array control with all the elements are dimmed indicating that the array is uninitialized. Figure 5.5(b) shows an array of two rows and two columns.



(a) Uninitialized two-dimensional array (b) Initialized array

Figure 5.5

DELETING ELEMENTS, ROWS, COLUMNS AND PAGES WITHIN ARRAYS

You can delete an element within a one-dimensional array and a row or column within a two-dimensional array.

To delete an element in a one-dimensional array, right-click the array element on the front panel and select Data Operations»Delete Element.

To delete a row or column in a two-dimensional array, right-click the array row or column on the front panel and select Data Operations»Delete Row or Delete Column.

You can also programmatically delete elements, rows, columns and pages within arrays using the Delete From Array function. You can delete an element, row, column or page within an array

programmatically. What you can delete depends on how many dimensions the array has. For example, you can delete a row or a column from an array of two or more dimensions.

You can delete a page from an array of three or more dimensions.

Complete the following steps to delete elements, rows, columns or pages in an array.

Step 1: Place the Delete From Array function on the block diagram.

Step 2: Wire an array of any dimension to the n-dim array input of the Delete From Array function. The function automatically resizes based on the dimensions of the array.

Step 3: Determine which operation you want to perform from Table 5.1 and complete the associated steps. The index input specifies from which element, row, column, or page you want to start deleting, with 0 being the first. The length input specifies the number of elements, rows, columns, or pages you want to delete.

Step 4: Run the VI

If the dimension size of the array you insert is smaller than the dimension size of the array you wired to the n-dim array input of the Insert into Array function, the function pads the array you insert with default data, such as zeros in the case of an array of numeric values.

If the dimension size of the array you insert is larger than the dimension size of the array you wired to the n-dim array input of the Insert into Array function, the function crops the array from the end.

TABLE 5.2 Operations for inserting elements, rows, columns and pages

Array wired to n-dimension array	Inserting	Complete these steps
One-dimensional array	Element	Wire a value 0–*n* to **index**. Wire a value to **new element/subarray**.
	One-dimensional array to row or column	Wire a value 0-*n* to **index**. Wire a one dimensional array to **new element/ subarray**.
Two-dimensional array	Row	Wire a value 0-*n* to **index (row)**. Wire a one dimensional array to **new element subarray**.
	Column	Wire a value 0–*n* to **index (column)**. Wire a one dimensional array to **new element/ subarray**.
	Two-dimensional array (rows and columns)	Wire a value 0–*n* to **index (row)** or **index (column)**. Wire a two dimensional array to **new element/ sub array**.
Three-dimensional-*n*D array	Page	Wire a value 0–*n* to **index (page)**. Wire a two dimensional array to **new element/ sub array**.
	Three dimensional array (pages)	Wire a value 0–*n* to **index (page)**. Wire a three dimensional array to **new element/sub array**.

REPLACING ELEMENTS, ROWS, COLUMNS, AND PAGES WITHIN ARRAYS

You can replace an element, row, column or page in an array. What you can replace depends on how many dimensions the array has. For example, in an array of two or more dimensions, you can replace a row or a column with a one-dimensional array.

In an array of three or more dimensions, you can replace a page with a two-dimensional array. Replacing is done using the Replace Array Subset function.

Complete the following steps to replace elements, rows, columns or pages in an array.

Step 1: Place the Replace Array Subset function on the block diagram.

Step 2: Wire an array of any dimension to the n-dimension array input of the Replace Array Subset function. The function automatically resizes based on the dimensions of the array.

Step 3: Determine which operation you want to perform from Table 5.3 and complete the associated steps. The index input specifies which element, row, column or page to replace, with 0 being the first. The new element/subarray input specifies the value you want to replace an element, or the array you want to replace a row, column or page.

Step 4: Resize the Replace Array Subset function to replace another element, row, column or page within an array and repeat steps 2 and 3.

Step 5: Run the VI.

TABLE 5.3 Operations for replacing elements, rows, columns and pages

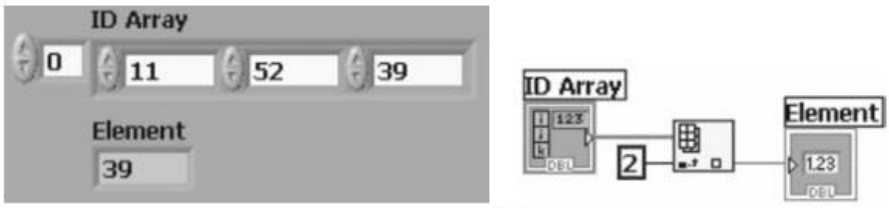
Array wired to *n*-dimension array	Replacing	Complete these steps
One-dimensional array	Element	Wire a value 0–*n* to **index**. Wire a value to **new element/subarray**.
Two-dimensional array	Row	Wire a value 0–*n* to **index (row)**. Wire a one dimensional array to **new element/subarray**.
	Column	Wire a value 0–*n* to **index (column)**. Wire a one dimensional array to **new element/subarray**.
	Element	Wire a value 0–*n* to **index (row)**. Wire a value 0–*n* to **index (column)**. Wire a value to **new element/subarray**.
Three-dimensional-*n*D array	Page	Wire a value 0–*n* to **index (page)**. Wire a two dimensional array to **new element/subarray**.
	Row	Wire a value 0–*n* to **index (page)**. Wire a value 0–*n* to **index (row)**. Wire a one dimensional array to **new element/subarray**.
	Column	Wire a value 0–*n* to **index (page)**. Wire a value 0–*n* to **index (column)**. Wire a one dimensional array to **new element/subarray**.
	Element	Wire a value 0–*n* to **index (page)**. Wire a value 0–*n* to **index (row)**. Wire a value 0–*n* to **index (column)**. Wire a value to **new element/subarray**.

ARRAY FUNCTIONS

Array functions are used to create and manipulate arrays. You can perform common array operations such as extracting individual data elements from an array, inserting, deleting, or replacing data elements in an array or splitting arrays using array functions.

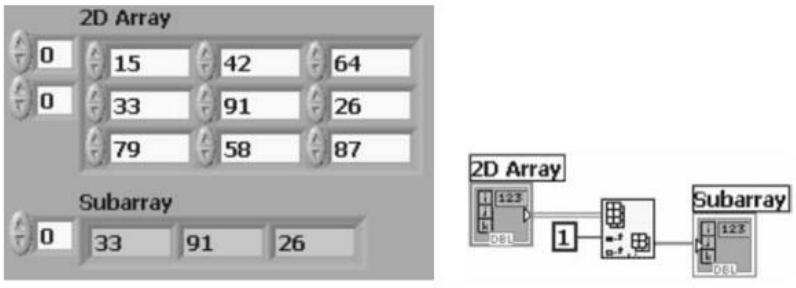
Array functions including Index Array, Replace Array Subset, Insert into Array, Delete from Array, and Array Subset automatically resize to match the dimensions of the input array you wire.

For example, if you wire a one-dimensional array to one of these functions, the function shows a single index input. If you wire a two-dimensional array to the same function, it shows two index inputs—one for the row index and one for the column index.

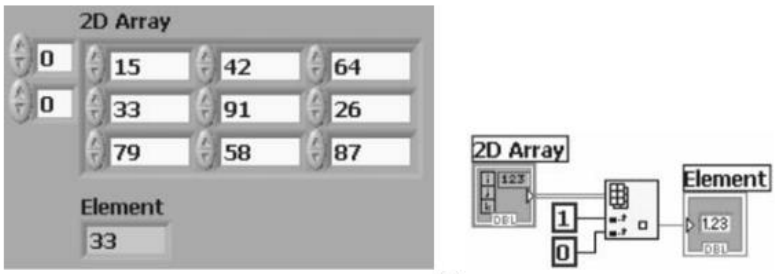


(a)

Figure 5.6 *(Contd.)*



(b)



(c)

Figure 5.6 (a) Index array function with one-dimensional array as input, (b) index array function with two-dimensional array as input and with one index (row index) and (c) index array function with two-dimensional array as input and with two indices.

In the above example shown in Figure 5.6, the Index Array function is used.

In Figure 5.6(a), input to the Index Array function is a one-dimensional array. By providing the index value in the output, you get the array element corresponding to the index value.

When connecting a two-dimensional array as input, the Index Array function automatically resizes to get two index inputs, one for row index and other for column index.

In Figure 5.6(b), input to the Index Array function is a two-dimensional array.

The row index is provided. At the output, you get a one-dimensional array which is a row of the two-dimensional input array. The row output is based on the index input provided. I

n Figure 5.6(c), input to the Index Array function is again a two-dimensional array. Here both row index and column index are given. The output is an element which is based on the row and column indices

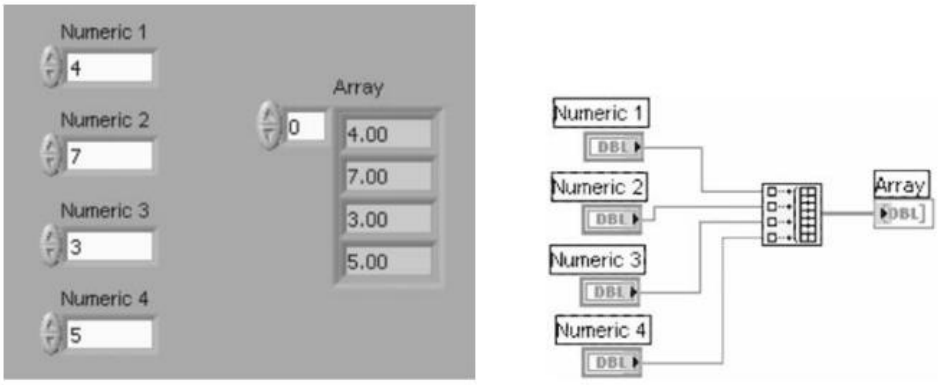
provided. You can access more than one element, or subarray (row, column, or page) with these functions by using the Positioning tool to manually resize the function. When you expand one of these functions, the functions expand in increments determined by the dimensions of the array wired to the function.

If you wire a one-dimensional array to one of these functions, the function expands by a single index input. If you wire a two-dimensional array to the same function, the function expands by two index inputs—one for the row and one for the column.

The index inputs you wire determine the shape of the subarray you want to access or modify. For example, if the input to an Index Array function is a two-dimensional array and you wire only the row input, you extract a complete row (one-dimensional) of the array.

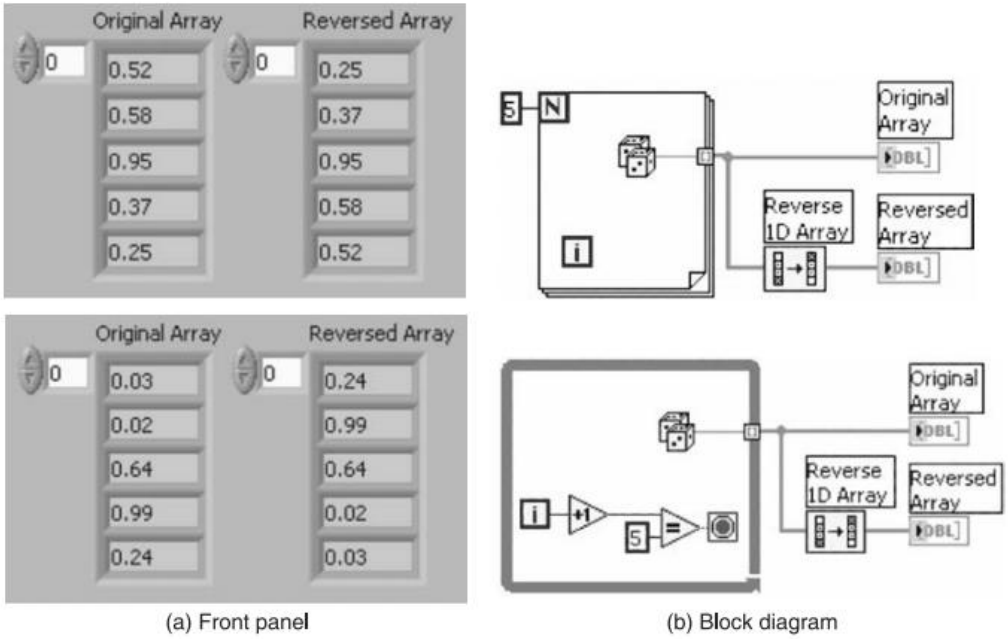
If you wire only the column input, you extract a complete column (one-dimensional) of the array. If you wire the row input and the column input, you extract a single element of the array. Each input group is independent and can access any portion of any dimension of the array

Problem 1 Create a one-dimensional (1D) numeric array using the *Build Array* function which gets array elements from numeric controls.



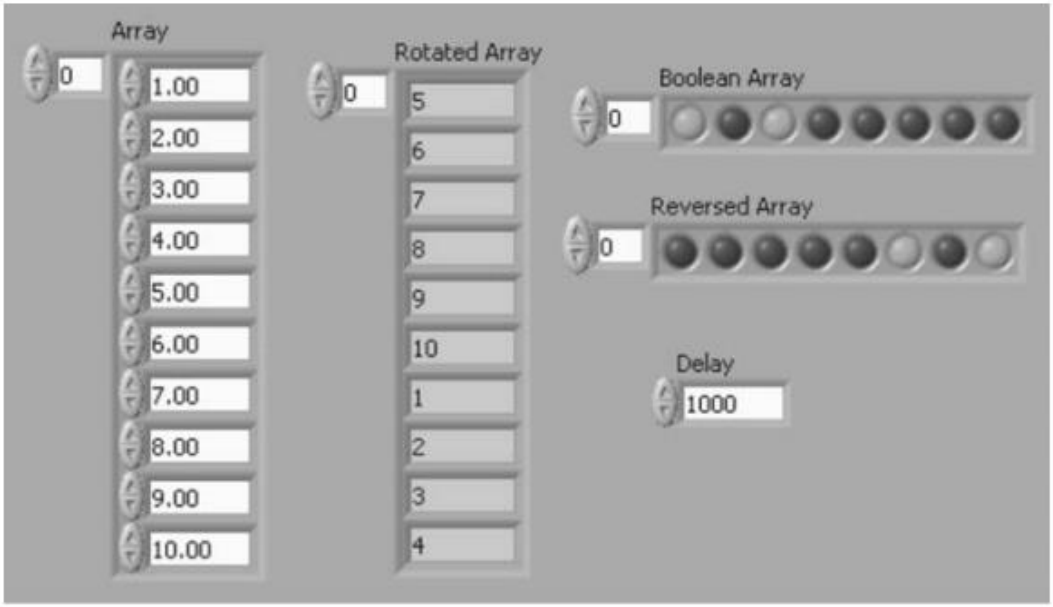
(a) Front panel (b) Block diagram

Problem 2 Create a 1D numeric array from loops (For and While) using random numbers and obtain the reverse of the array.

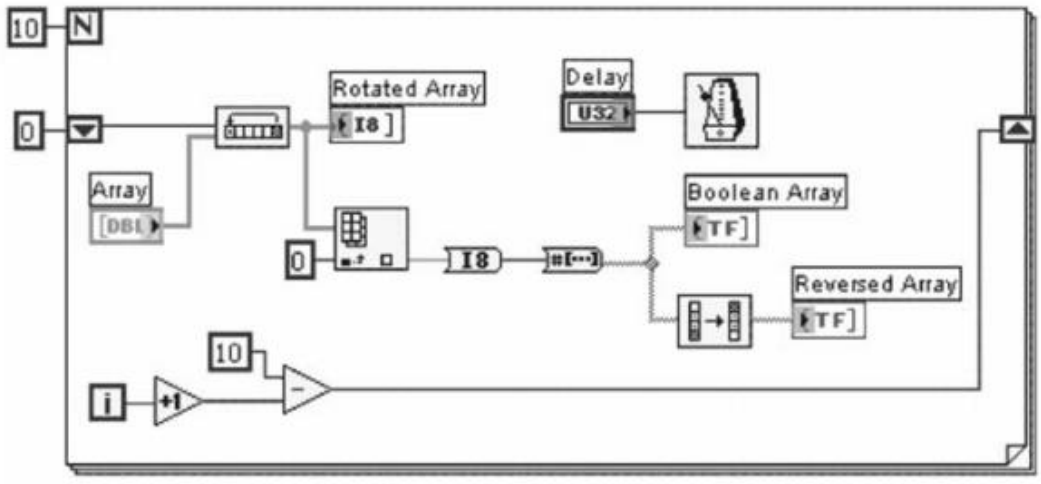


(a) Front panel (b) Block diagram

Problem 3 **Create a 1D numeric array which consists of ten elements and rotate it ten times. For each rotation display the equivalent binary number of the first array element in the form of a Boolean array. Also display the reversed Boolean array. Provide delay to view the rotation.**

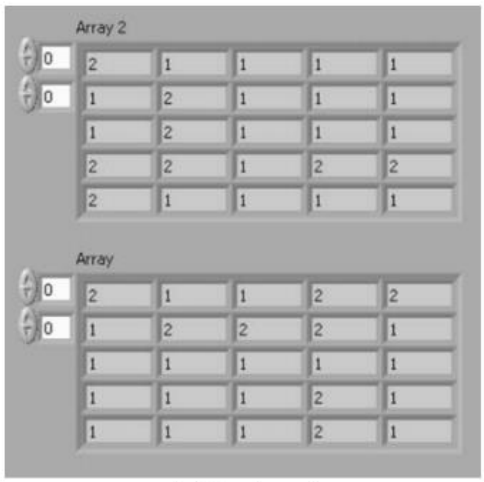


(a) Front panel

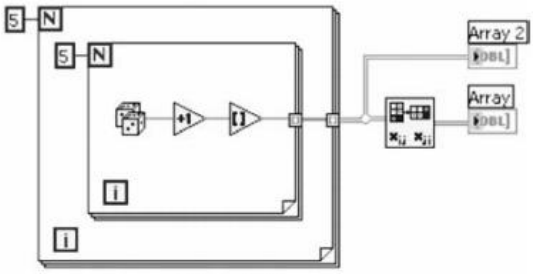


(b) Block diagram

Problem 4 Create a 2D numeric array (5 ¥ 5) containing random numbers and find its transpose

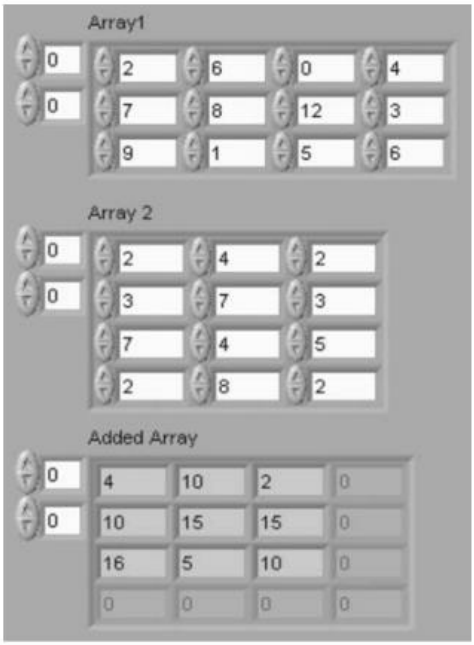


(a) Front panel

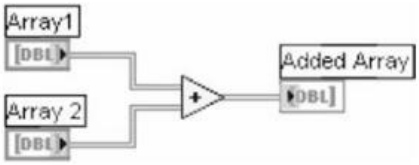


(b) Block diagram

Problem 5 Create two 2D numeric arrays and add them. Change the number of rows and number of columns of each array and see the result.



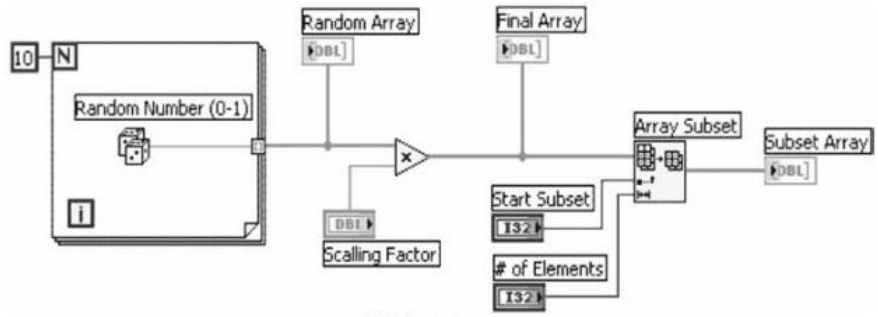
(a) Front panel



(b) Block diagram

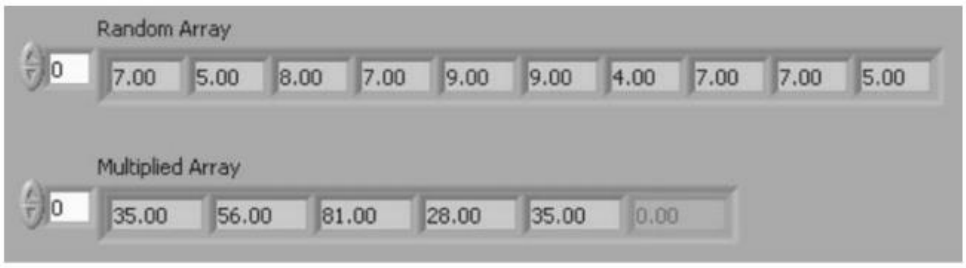
Problem 6 Build a VI that generates a 1D array. Multiply the array elements by a scaling factor and find the resultant array. Create a subset array from the resultant array

(a) Front panel

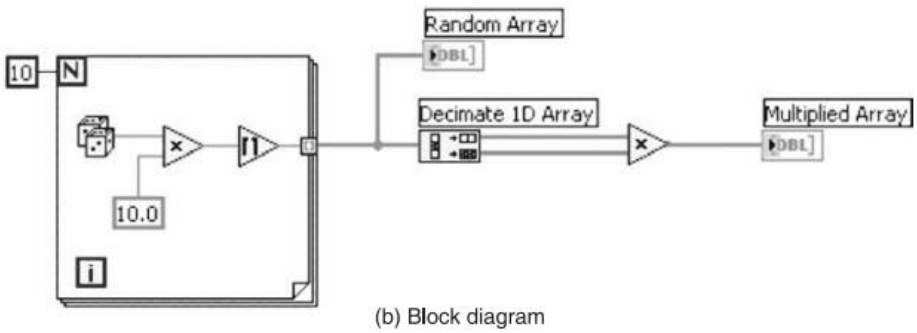


(b) Block diagram

Problem 7 Build a VI that generates a 1D array and then multiplies pairs of elements together and outputs the resulting array. For example, the input array with values 1, 23, 10, 5, 7, 11 will result in the output array 23, 50, 77.

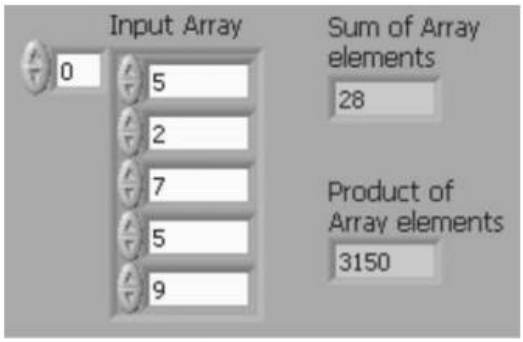


(a) Front panel

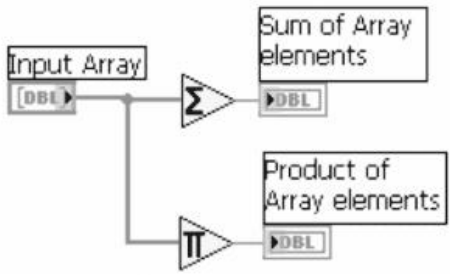


(b) Block diagram

Problem 8 **Build a VI to find the sum and product of array elements**



(a) Front panel



(b) Block diagram