Objectives

In this chapter, you will learn about:

- One-dimensional arrays review
- Declaring and processing two-dimensional arrays
- Large dimension array
- Example
One-Dimensional Arrays

- **One-dimensional array**: A list of related values with the same data type, stored using a single group name (called the **array name**)
  - Syntax:
    ```
    dataType arrayName[number-of-items]
    ```
  - By convention, the number of items is first declared as a constant, and the constant is used in the array declaration
One-Dimensional Arrays (continued)

- **Examples:**

  ```cpp
  const int NUMELS = 6;
  int volts[NUMELS];
  const int ARRAYSIZE = 4;
  char code[ARRAYSIZE];
  ```

*Figure 7.1 The `volts` and `code` arrays in memory*
Declaring and Processing Two-Dimensional Arrays

- **Two-dimensional array**: Has both rows and columns
  - Also called a **table**
- Both dimensions must be specified in the array declaration
  - Row is specified first, then column
- Both dimensions must be specified when referencing an array element
Declaring and Processing Two-Dimensional Arrays (continued)

- Example:
  ```c
  int val[1][3];
  ```

Figure 7.5 Each array element is identified by its row and column position
Declaring and Processing Two-Dimensional Arrays (continued)

- Two-dimensional arrays can be initialized in the declaration by listing values within braces, separated by commas.
- Braces can be used to distinguish rows, but are not required.
Declaring and Processing Two-Dimensional Arrays (continued)

- Nested `for` loops are used to process two-dimensional arrays
  - Outer loop controls the rows
  - Inner loop controls the columns
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    const int NUMROWS = 3;
    const int NUMCOLS = 4;
    int i, j;
    int val[NUMROWS][NUMCOLS] = {
        {8,16,9,52},
        {3,15,27,6},
        {14,25,2,10}
    };

    // multiply each element by 10 and display it
    cout << "Display of multiplied elements";
    for (i = 0; i < NUMROWS; i++)
    {
        cout << endl;   // start each row on a new line
        for (j = 0; j < NUMCOLS; j++)
        {
            val[i][j] = val[i][j] * 10;
            cout << setw(5) << val[i][j];
        }   // end of inner loop
    }   // end of outer loop
    cout << endl;
    return 0;
}
Two-Dimensional Array Example

5x5 Bingo Card

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>23</td>
<td>32</td>
<td>48</td>
<td>75</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>43</td>
<td>50</td>
<td>68</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>41</td>
<td>47</td>
<td>62</td>
</tr>
<tr>
<td>7</td>
<td>18</td>
<td>35</td>
<td>52</td>
<td>72</td>
</tr>
<tr>
<td>11</td>
<td>16</td>
<td>42</td>
<td>46</td>
<td>63</td>
</tr>
</tbody>
</table>

Column 1: 1 - 15
Column 2: 16 - 30
Column 3: 31 - 45
Column 4: 46 - 60
Column 5: 61 - 75

Random number generator
#include <cstdlib>
#include <ctime>
Larger Dimensional Arrays

- Arrays with more than two dimensions can be created, but are not commonly used
- Think of a three-dimensional array as a book of data tables

Figure 7.7 Representation of a three-dimensional array
Common Programming Errors

- Failing to declare the array
- Using a subscript that references a non-existent array element (out of bounds)
- Failing to use a counter value in a loop that is large enough to cycle through all array elements
- Failing to initialize the array
Summary

• Single dimension array is a data structure that stores a list of values having the same data type
• Array elements are stored in contiguous memory locations, and referenced by array name and index position
• Two-dimensional array has rows and columns
• Arrays may be initialized when they are declared