CSc 10200
Introduction to Computing

Lecture 12
Edgardo Molina
Fall 2013 – City College of New York
Objectives

In this chapter, you will learn about:

• One-dimensional arrays
• Array initialization
• Declaring and processing two-dimensional arrays
• Arrays as arguments
• Statistical analysis
Objectives (continued)

• The Standard Template Library (STL)
• Searching and sorting
• Common programming errors
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:
    ```cpp
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:

```c
const int NUMELS = 6;
int volts[NUMELS];
```

```c
const int ARRAYSIZE = 4;
char code[ARRAYSIZE];
```

*Figure 7.1 The volts and code arrays in memory*
One-Dimensional Arrays (continued)

• **Element**: An item in the array
  – Array storage of elements is contiguous
• **Index** (or **subscript**) of an element: The position of the element within the array
  – Indexes are zero-relative
• To reference an element, use the array name and the index of the element

![Diagram of array elements](image)

**Figure 7.2** Identifying array elements
One-Dimensional Arrays (continued)

• Index represents the offset from the start of the array
• Element is also called indexed variable or subscripted variable
• Subscripted variable can be used anywhere that a variable can be used
• Expressions can be used within the brackets if the value of the expression
  – Yields an integer value
  – Is within the valid range of subscripts
One-Dimensional Arrays (continued)

• All of the elements of an array can be processed by using a loop
• The loop counter is used as the array index to specify the element
• Example:

```cpp
sum = 0;
for (i=0; i<5; i++)
    sum = sum + temp[i];
```
Input and Output of Array Values

- Array elements can be assigned values interactively using a `cin` stream object
- Out of range array indexes are not checked at compile-time
  - May produce run-time errors
  - May overwrite a value in the referenced memory location and cause other errors
- Array elements can be displayed using the `cout` stream object
Input and Output of Array Values (continued)

Program 7.1

```cpp
#include <iostream>
using namespace std;

int main()
{
    const int MAXTEMPS = 5;
    int i, temp[MAXTEMPS];

    for (i = 0; i < MAXTEMPS; i++) // Enter the temperatures
    {
        cout << "Enter a temperature: ";
        cin >> temp[i];
    }

    cout << endl;

    for (i = 0; i < MAXTEMPS; i++) // Print the temperatures
        cout << "temperature " << i << " is " << temp[i] << endl;

    return 0;
}
```
#include <iostream>
using namespace std;

int main()
{
    const int MAXTEMPS = 5;
    int i, temp[MAXTEMPS], total = 0;

    for (i = 0; i < MAXTEMPS; i++) // Enter the temperatures
    {
        cout << "Enter a temperature: ";
        cin >> temp[i];
    }

    cout << "\nThe total of the temperatures";

    for (i = 0; i < MAXTEMPS; i++) // Display and total the temperatures
    {
        cout << " " << temp[i];
        total = total + temp[i];
    }

    cout << " is " << total << endl;

    return 0;
}
Array Initialization

- Array elements can be initialized in the array declaration statement

- Example:
  ```cpp
  ```

- Initialization:
  - Can span multiple lines, because white space is ignored in C++
  - Starts with array element 0 if an insufficient number of values is specified

- If initializing in the declaration, the size may be omitted
Array Initialization (continued)

- **char** array will contain an extra null character at the end of the string
- Example:

  ```
  char codes[] = "sample";
  ```

![Diagram showing char array initialization with a string]

**Figure 7.4** Initializing a character array with a string adds a terminating \0 character
Array Initialization (continued)

Program 7.3

```cpp
#include <iostream>
using namespace std;

int main()
{
    const int MAXELS = 5;

    int i, max, nums[MAXELS] = {2, 18, 1, 27, 16};

    max = nums[0];

    for (i = 1; i < MAXELS; i++)
        if (max < nums[i])
            max = nums[i];

    cout << "The maximum value is " << max << endl;

    return 0;
}
```
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:
  
  ```cpp
  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 << "\nDisplay 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;
}
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