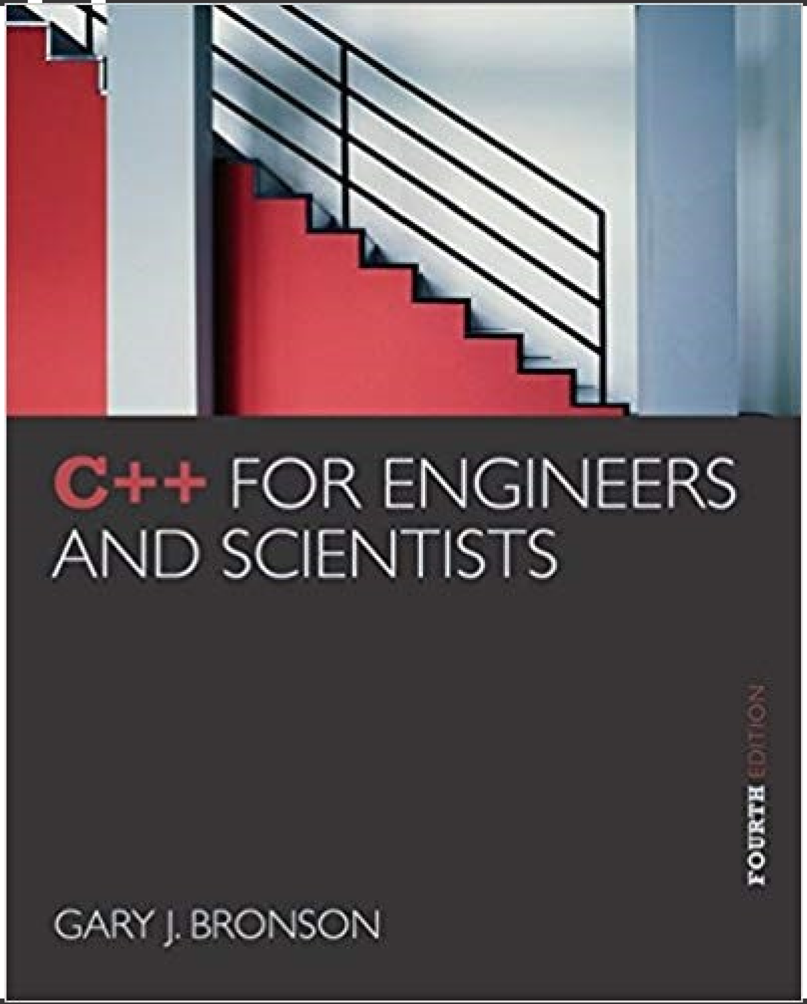
ELEG 1043
Computer Applications in Engineering



Chapter 7: Arrays

C++ FOR ENGINEERS AND SCIENTISTS

Acknowledgement

- Some of the slides or images are from various sources. The copyright of those materials belongs to their original owners.

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:

```
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)

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

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

const int SIZE = 100;
double amount[SIZE];
```

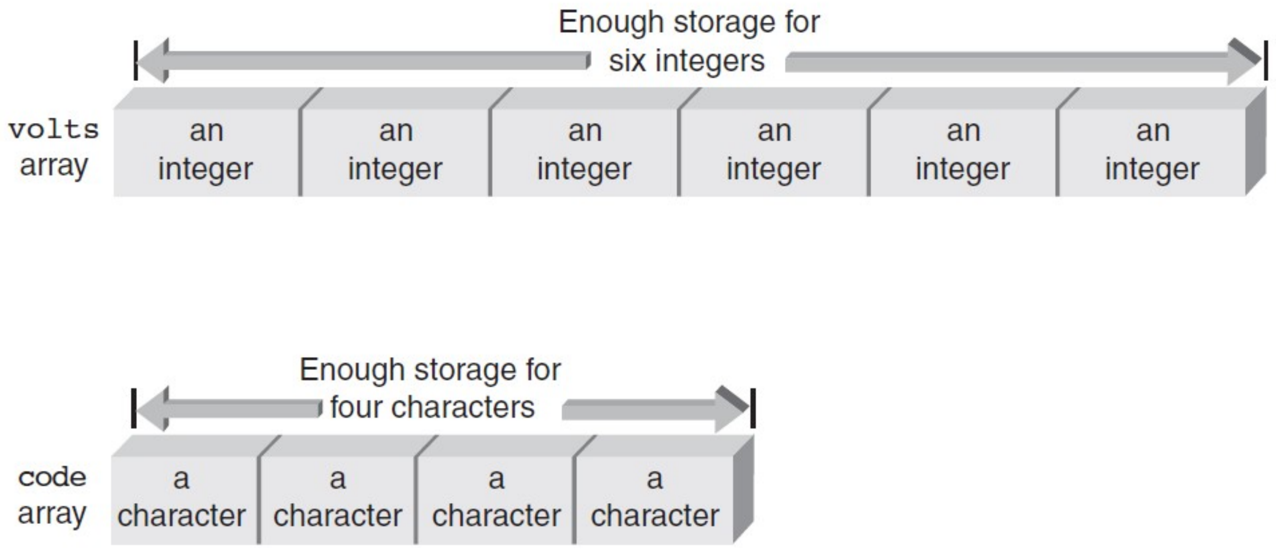


Figure 7.1 The `volts` and `code` arrays in memory

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

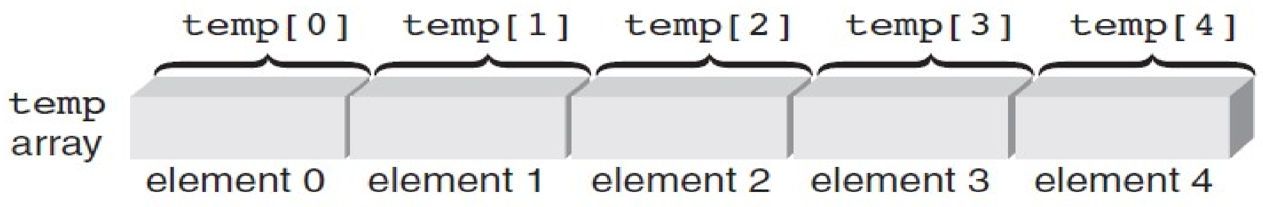


Figure 7.2 Identifying array elements

One-Dimensional Arrays (continued)

- Index represents the offset from the start of the array
- Element is also called **indexed variable** or **subscripted variable**
- 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:

```
int sum = 0;
int temp[5] = {1,2,3,4,5};
for (int 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

Array Initialization

- Array elements can be initialized in the array declaration statement
- Example:

```
int temp[5] = {98, 87, 92, 79, 85};
```

- Initialization:
  - Can span multiple lines, because white space is ignored
  - Starts with array element 0
- If initializing in the declaration, the size may be omitted: `int temp[] = {98, 87, 92, 79, 85};`

Array Initialization (continued)

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

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

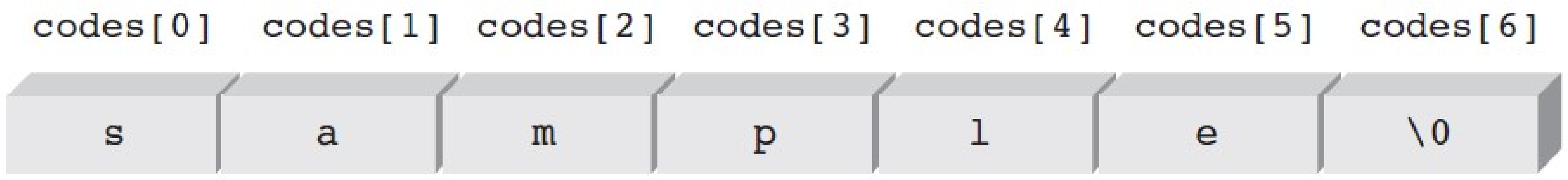


Figure 7.4 Initializing a character array with a string adds a terminating \0 character

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 (cont'd)

- Example:

```
int val[1][3];
```

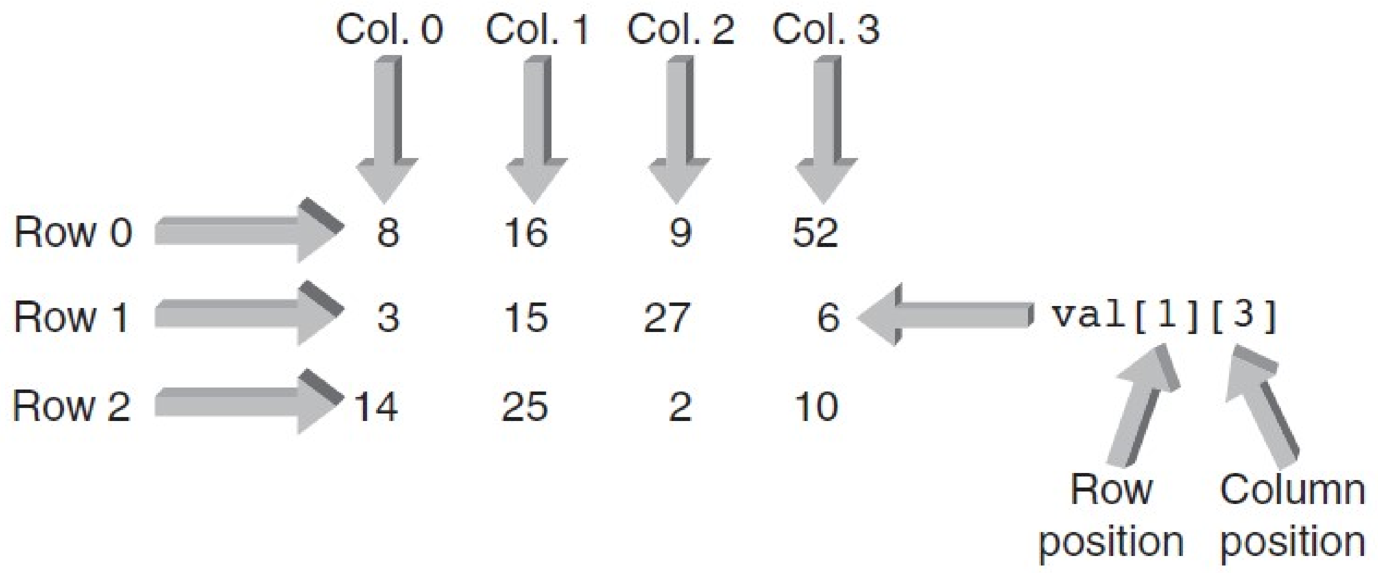


Figure 7.5 Each array element is identified by its row and column position

Declaring and Processing Two-Dimensional Arrays (cont'd)

- 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
- Nested **`for`** loops are used to process two-dimensional arrays
  - Outer loop controls the rows
  - Inner loop controls the columns