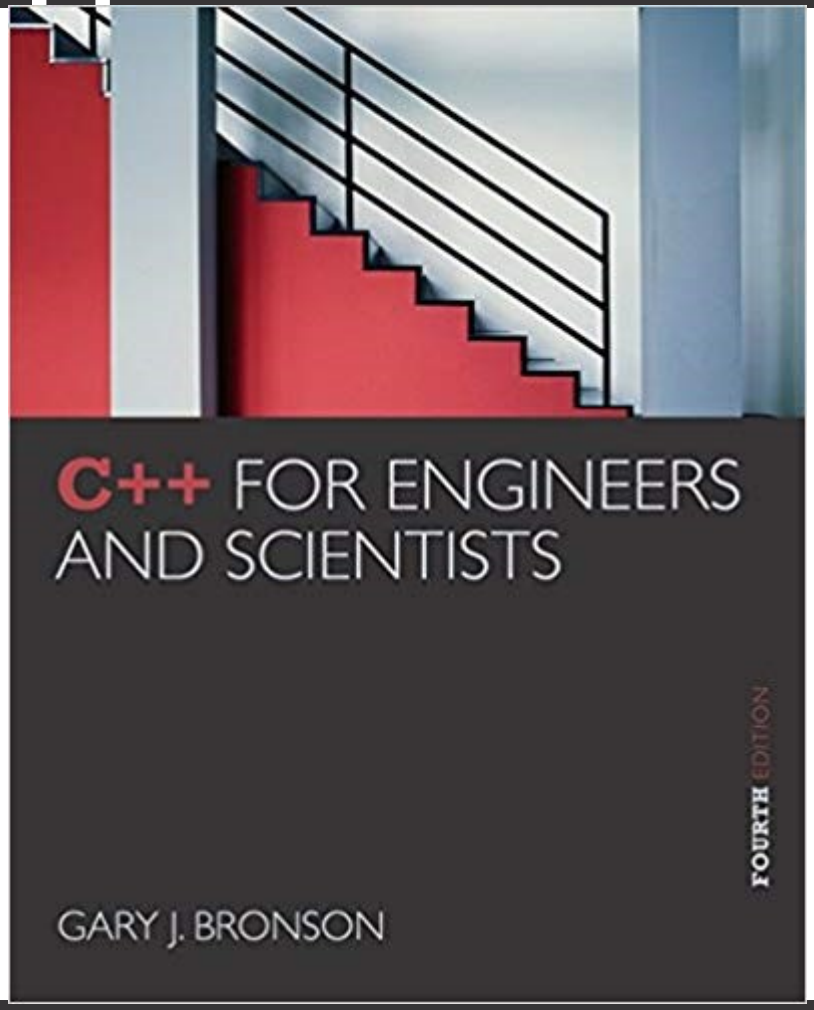
ELEG 1043
Computer Applications in Engineering



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

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

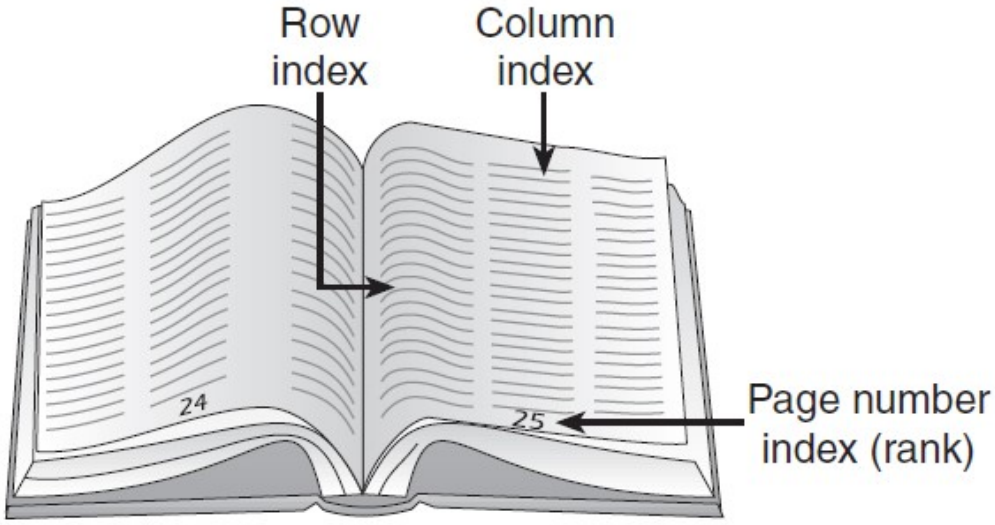


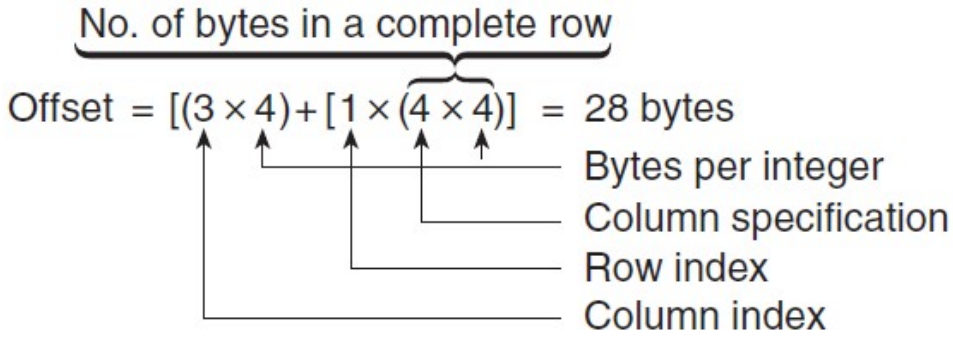
Figure 7.7 Representation of a three-dimensional array

Arrays as Arguments

- An individual array element can be passed as an argument just like any individual variable
- The called function receives a copy of the array element's value
- Passing an entire array to a function causes the function to receive a reference to the array, not a copy of its element values
- The function must be declared with an array as the argument
- Single element of array is obtained by adding an offset to the array's starting location

Arrays as Arguments (continued)

Figure 7.10 Storage of the `val` array



Internal Array Element Location Algorithm

- Each element of an array is obtained by adding an offset to the starting address of the array:
  - *Address of element i = starting array address + the offset*
- Offset for one dimensional arrays:
  - *Offset = i * the size of the element*
- Offset for two dimensional arrays:
  - *Offset = column index value * the size of an element + row index value * number of bytes in a complete row*

Case Study

- Arrays are useful in applications that require multiple passes through the same set of data elements
  - Statistical Analysis
  - Array: *X* = [98, 82, 67, 54, 78, 83, 95, 76, 68, 63]
  - Calculating
    - Mean value
    - Standard Deviation

Standard Template Library

- **Standard Template Library (STL):** Generic set of data structures that can be modified, expanded, and contracted
- **Vector:** Similar to an array
  - Uses a zero-relative index, but automatically expands as needed

The STL (continued)

- STL **`Vector`** class provides many useful methods (functions) for vector manipulation:
  - **`insert(pos, elem)`** : inserts **`elem`** at position **`pos`**
  - **`name.push_back(elem)`** : appends **`elem`** at the end of the vector
  - **`name.size`**: returns the size of the vector

The STL (continued)

- Must include the header files for **`vector`** with the namespace **`std`**
- Syntax:
  - To create and initialize a vector:

    ```
    vector<dataType> vectorName(NumElem);
    ```

  - To modify a specific element:

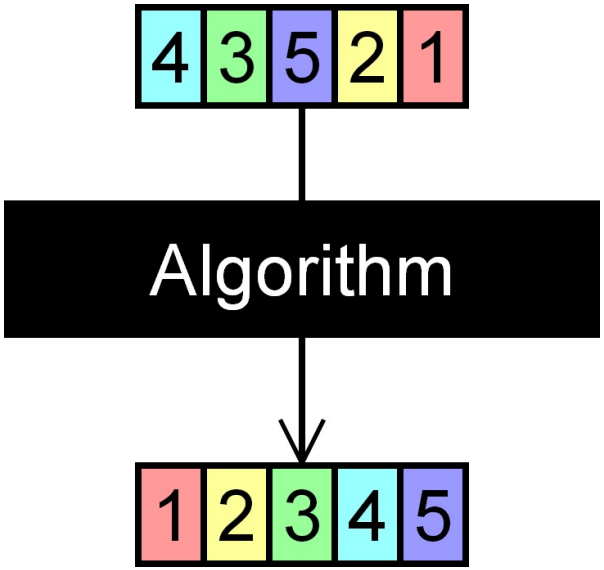
    ```
    vectorName[index] = newValue;
    ```

  - To insert a new element:

    ```
    vectorName.insert(index, newValue);
    ```

A Closer Look: Searching & Sorting

- Searching: Scanning through a list of data to find a particular item
- Sorting: Arranging data in ascending or descending order for some purpose



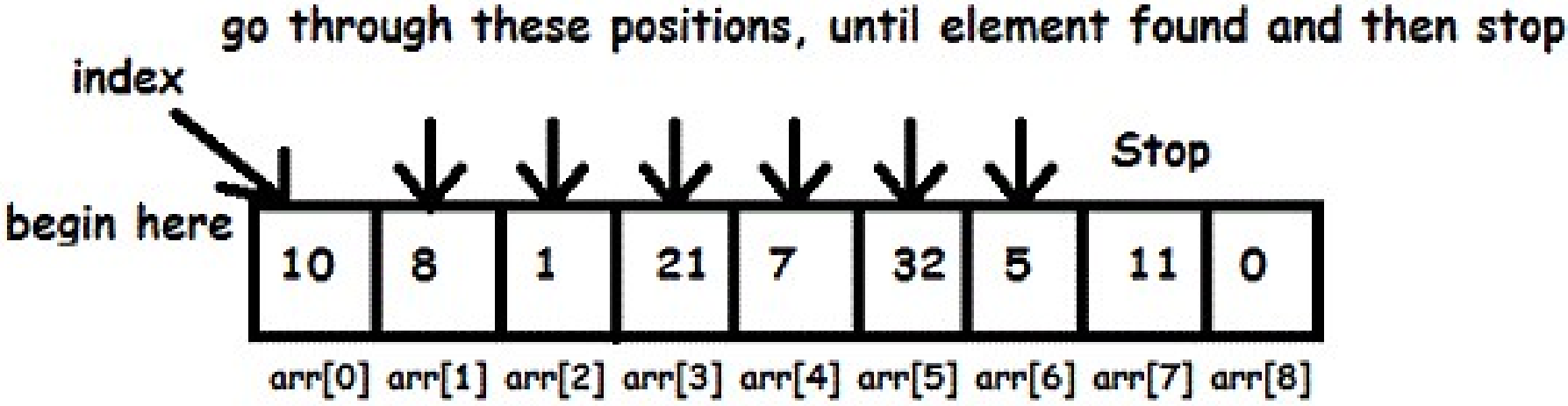
Search Algorithms

- Searches can be faster if the data is in sorted order
- Two common methods for searching:
  - Linear search
  - Binary search
- Linear search is a sequential search
  - Each item is examined in the order it occurs in the list

Linear Search

- Each item in the list is examined in the order in which it occurs
- Not a very efficient method for searching
- Advantage is that the list does not have to be in sorted order

Linear Search (continued)



Element to search : 5

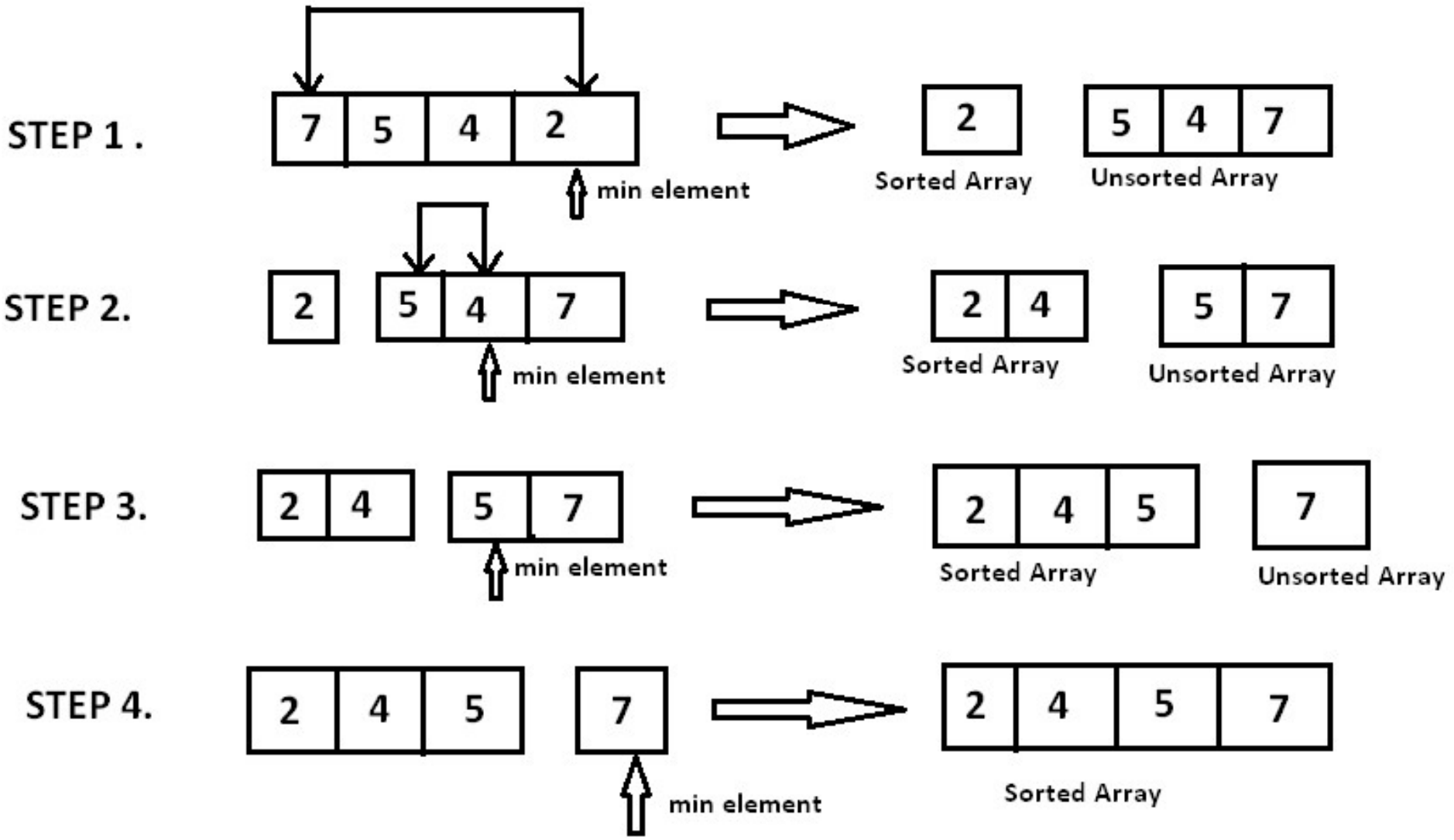
Linear Search (continued)

```
/* Linear Search Function */
int linear_search(std::vector<int> v, int val)
{
    int key = -1;
    for (int i = 0; i < v.size(); i++)
    {
        if (v[i] == val)
        { key = i; break;}
    }
    return key;
}
```

Selection Sort

- Smallest element is found and exchanged with the first element
- Next smallest element is found and exchanged with the second element
- Process continues *n*-1 times, with each pass requiring one less comparison

Selection Sort (continued)



Common Programming Errors

- Failing to declare the array
- Using a subscript (index) 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

- An array is a data structure that stores a list of values having the same data type
  - Array elements: stored in contiguous memory locations; referenced by array name/index position
  - Two-dimensional arrays have rows and columns
  - Arrays may be initialized when they are declared
  - Arrays may be passed to a function by passing the name of the array as the argument
    - Arrays passed as arguments are passed by reference
    - Individual array elements as arguments are passed by value (copy)