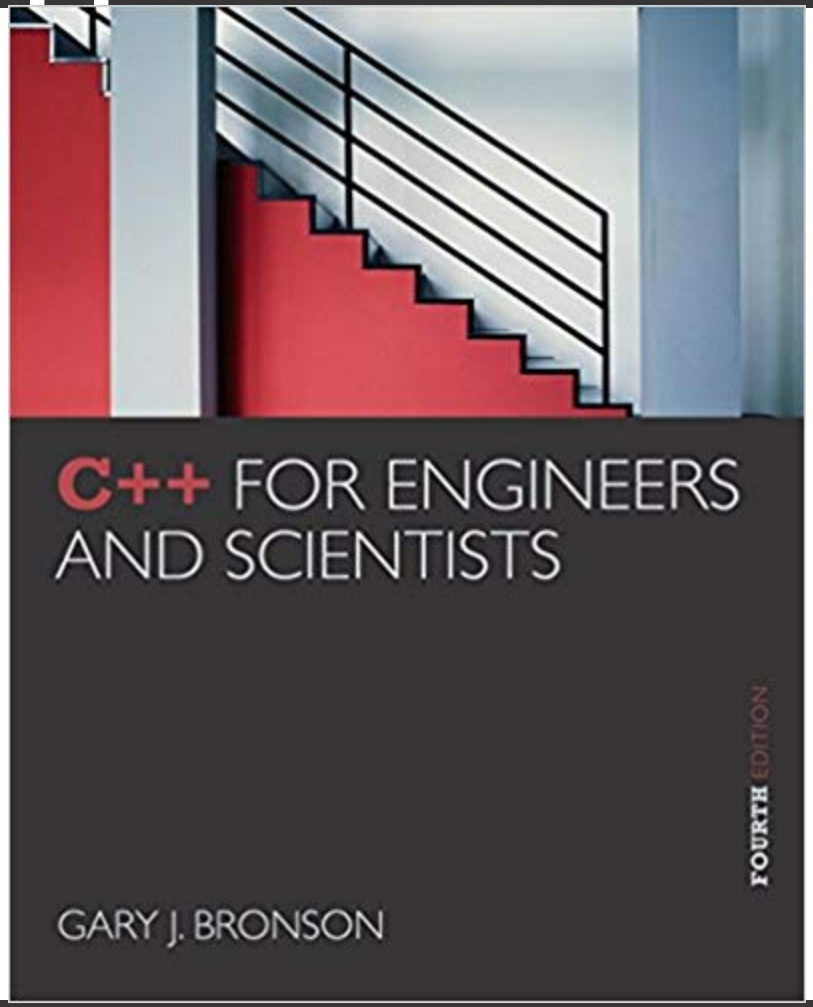
ELEG 1043
Computer Applications in Engineering



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];
```

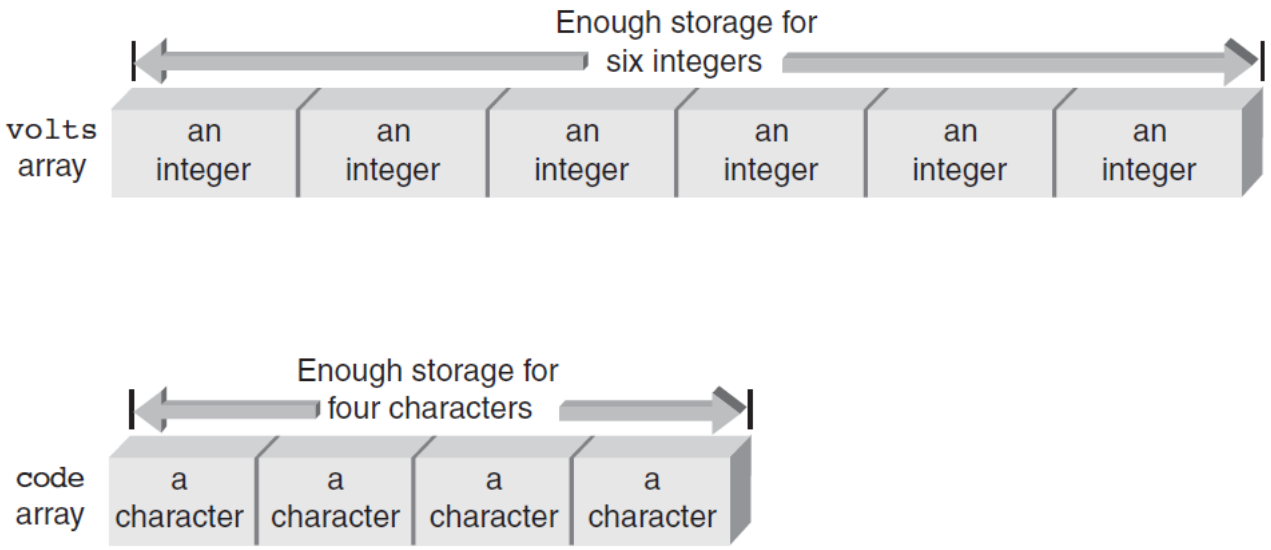


Figure 7.1 The `volts` and `code` arrays in memory

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

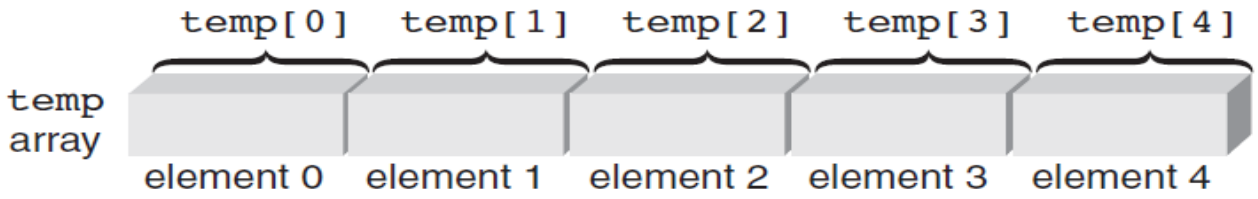


Figure 7.2 Identifying array elements

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";
```

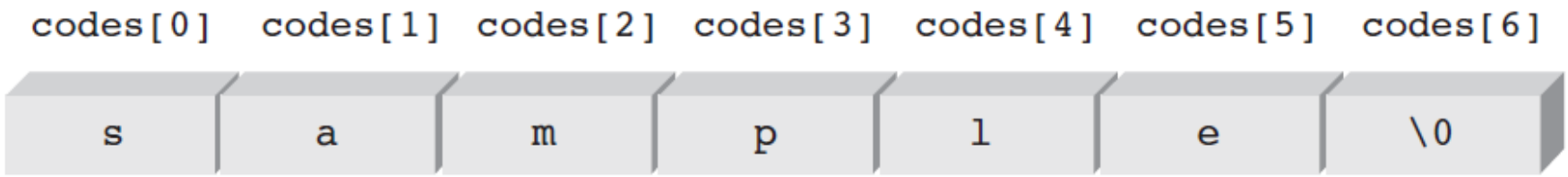


Figure 7.4 Initializing a character array with a string adds a terminating \0 character

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];
```

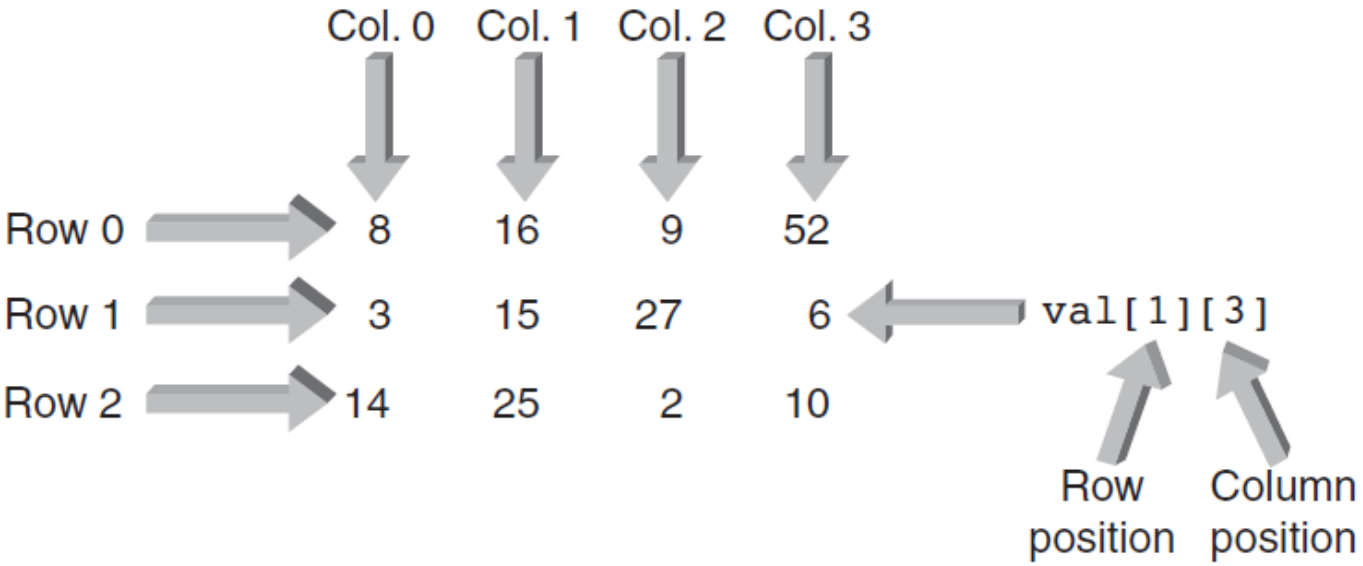


Figure 7.5 Each array element is identified by its row and column position

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

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

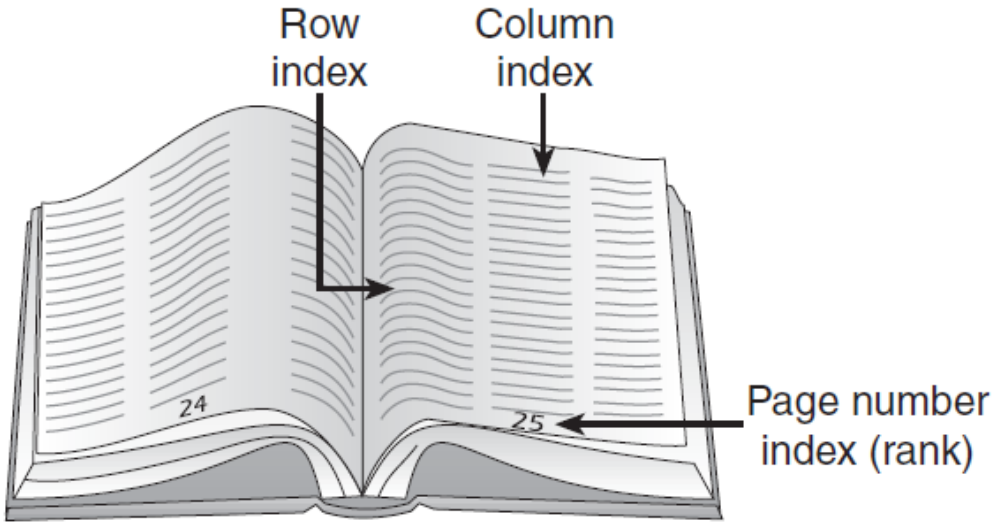


Figure 7.7 Representation of a three-dimensional array

Arrays as Arguments

- An individual array element can be passed as an argument just like any individual variable
- 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

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 = column index value * the size of the element*

Internal Array Element Location Algorithm (continued)

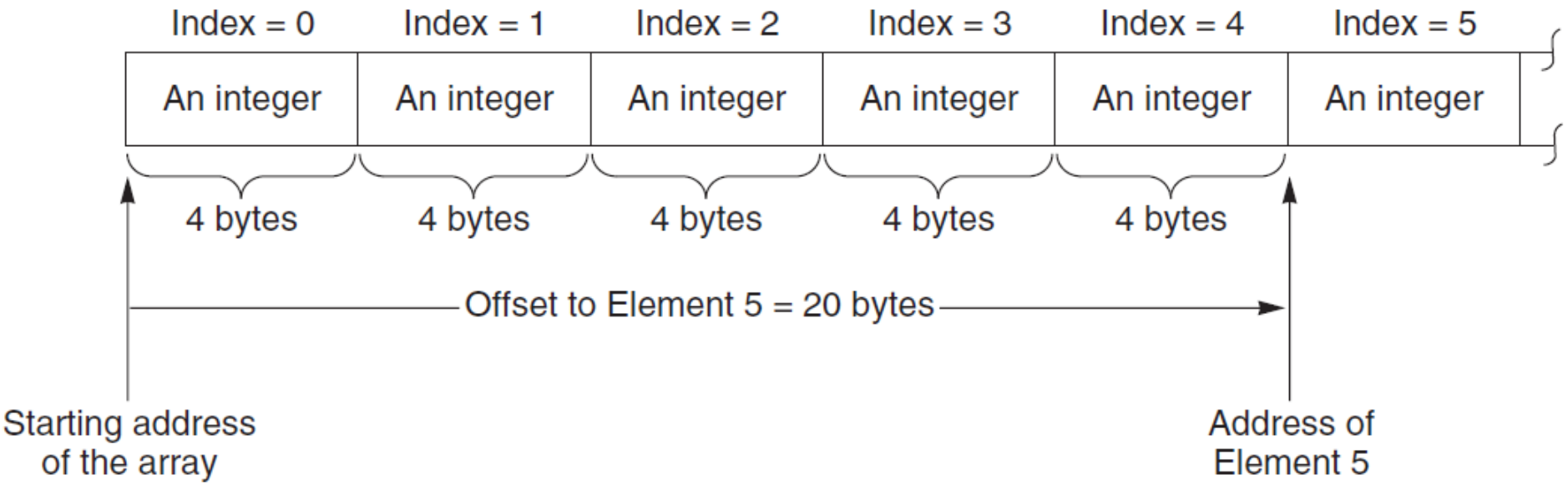


Figure 7.11 The offset to the element with an index value of 5

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 two dimensional arrays:
  - *Offset = column index value * the size of an element + row index value * number of bytes in a complete row*

Arrays as Arguments (continued)

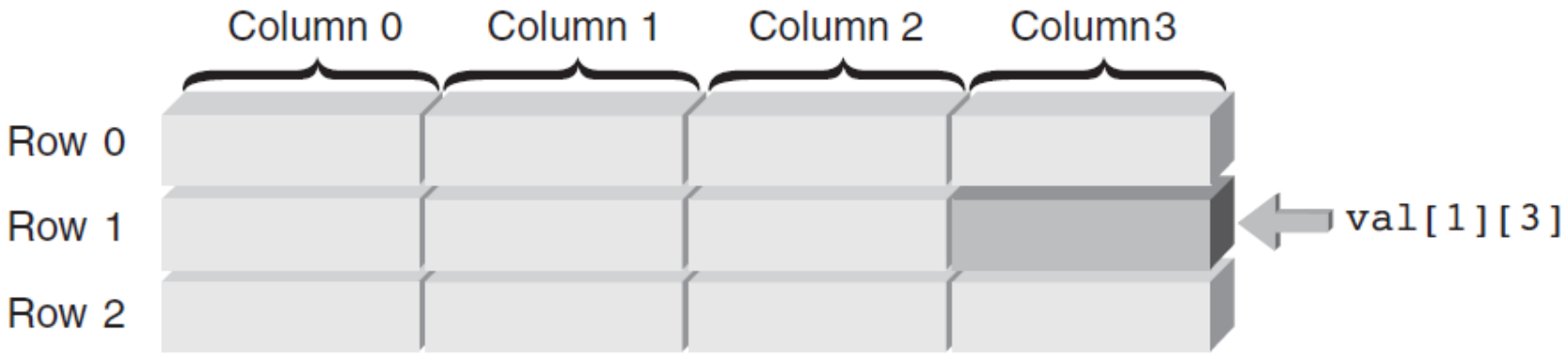
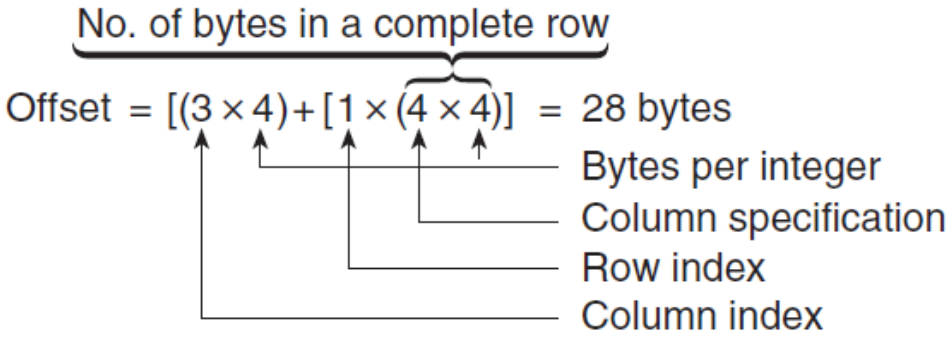


Figure 7.10 Storage of the `val` array



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

Case Study

- Mean value

$$\mu = \frac{\sum_{i=1}^{N} x_i}{N}$$

- Standard Deviation

$$\delta = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N-1}}$$

Mean value

```
double findAvg(int nums[], int numel)
{
     int i;
    double sumnums = 0.0;
    for (i = 0; i < numel; i++)
          sumnums = sumnums + nums[i];
    return (sumnums / numel);
}
```

Standard Deviation

```
double stdDev(int nums[], int numel, double avr)
{
    int i;
    double sumdevs = 0.0;
    for (i = 0; i < numel; i++)
        sumdevs = sumdevs + pow((nums[i] – avr),2);
    return (sqrt(sumdevs/(numel – 1.0)));
}
```

Main Function

```
#include <iosteam>
using namespace std;
int main(){
    const int NUMELS = 10;
    int values[NUMELS] = {98, 82, 67, 54, 78, 83, 95, 76, 68, 63};
    double average, sDev;
    average = findAvg(values, NUMELS); // call the function
    sDev = stdDev(values, NUMELS, average); // call the function
    cout << "The average of the numbers is “<<average << endl;
    cout << "The standard deviation of the numbers is “<<sDev << endl;
    return 0;
}
```

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:

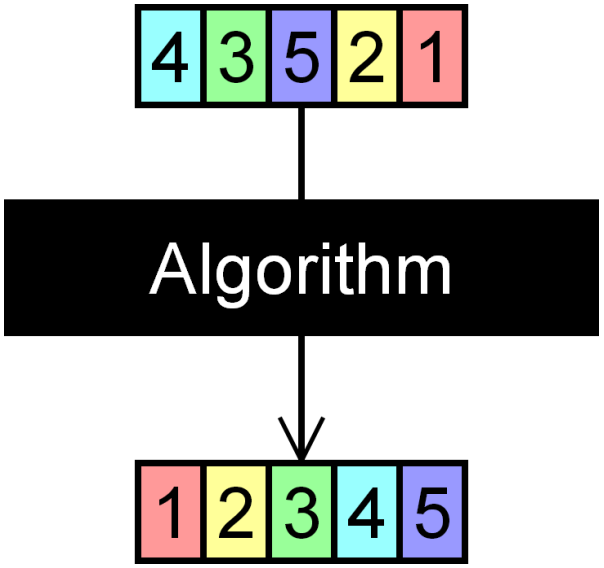
    ```
    vector<dataType> vectorName(elemNum);
    ```

  - To modify a specific element:

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

A Closer Look: Searching & Sorting

- Sorting: Arranging data in ascending or descending order for some purpose



- Searching: Scanning through a list of data to find a particular item

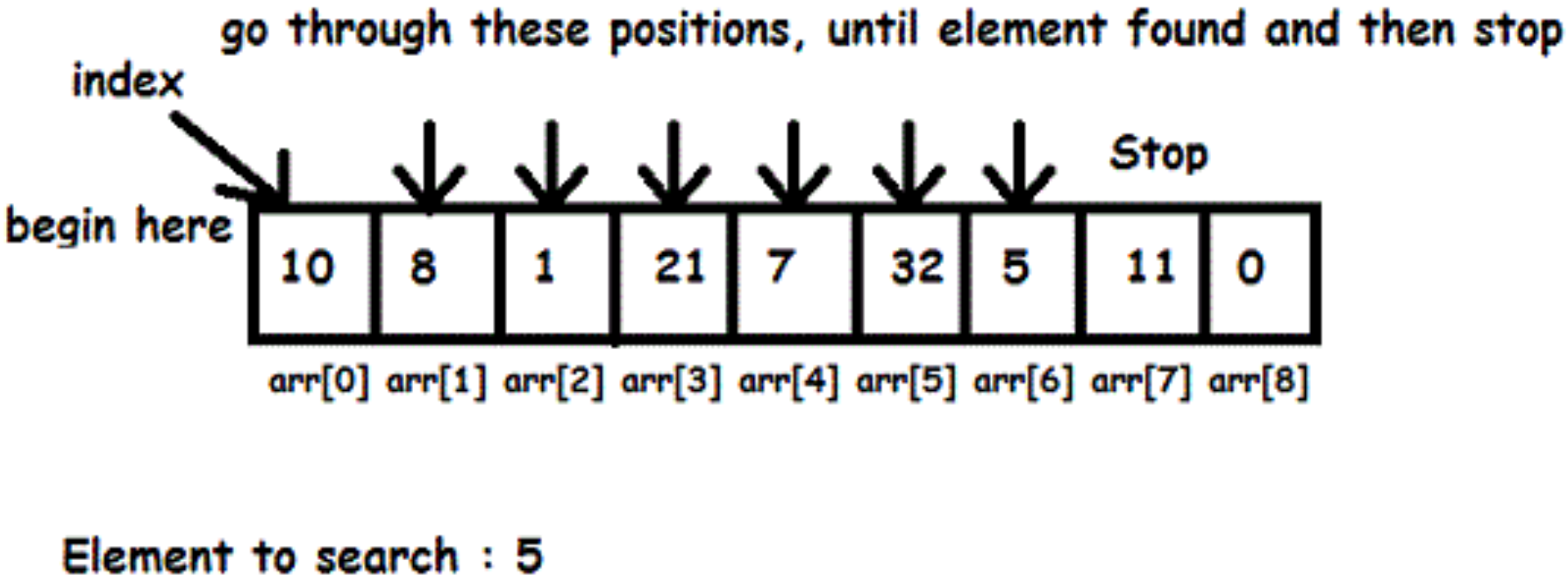
Search Algorithms

- Searches can be faster if the data is in sorted order
- 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

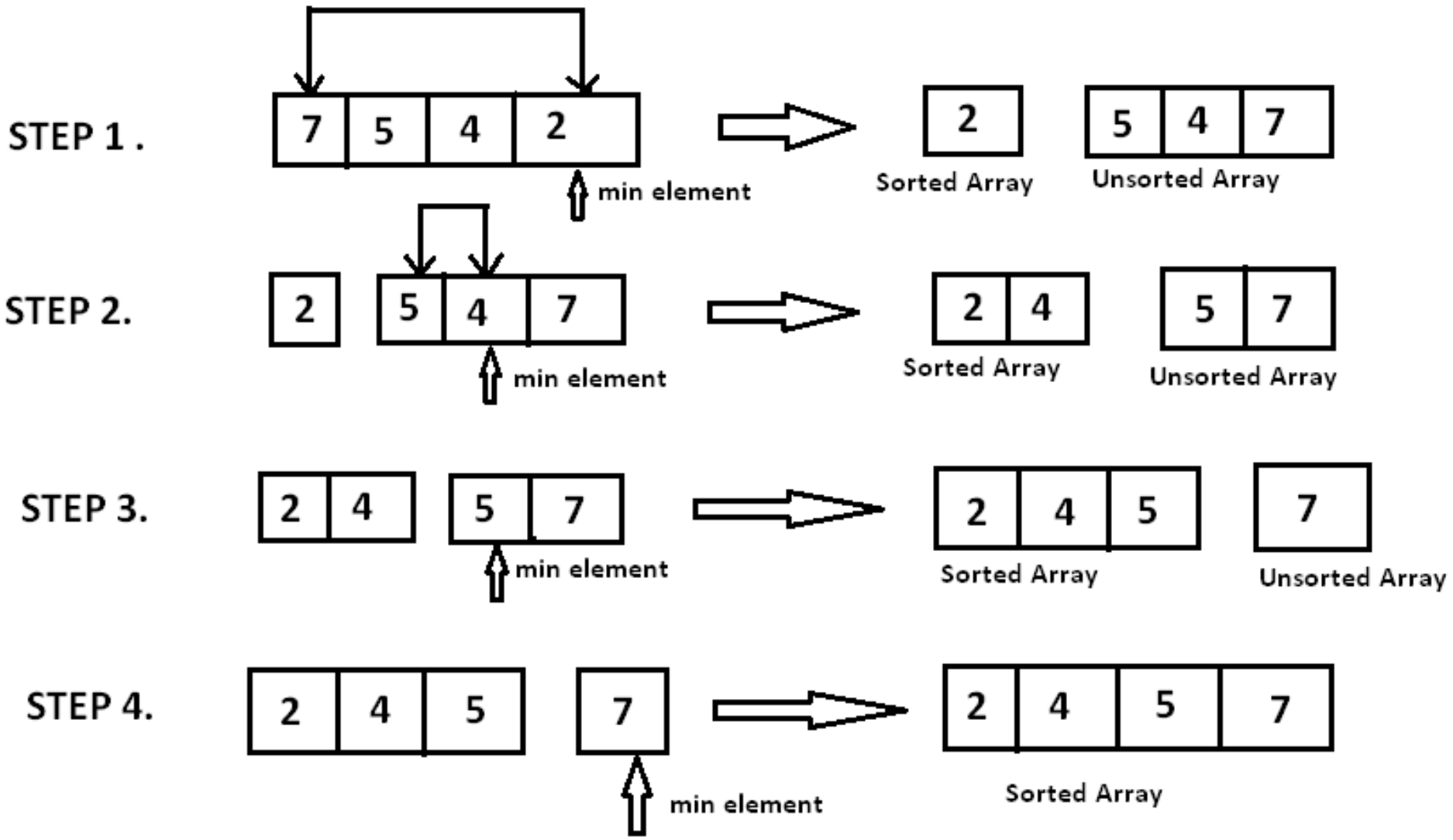
Linear Search (continued)

```
/* Linear Search Function */
int linear_search(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 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)