C++ Programming: From Problem Analysis to Program Design, Fourth Edition

Arrays and Strings

Objectives

In this chapter, you will:

- Learn about arrays
- Explore how to declare and manipulate data into arrays
- Understand the meaning of “array index out of bounds”
- Become familiar with the restrictions on array processing
- Discover how to pass an array as a parameter to a function

Objectives (continued)

- Discover how to manipulate data in a two-dimensional array

Data Types

- A data type is called simple if variables of that type can store only one value at a time
- A structured data type is one in which each data item is a collection of other data items

Arrays

- Array: a collection of a fixed number of components wherein all of the components have the same data type
- In a one-dimensional array, the components are arranged in a list form
- Syntax for declaring a one-dimensional array:

```
dataType arrayName[intExp];
```

`intExp` evaluates to a positive integer

Arrays (continued)

- Example:

```
int num[5];
```

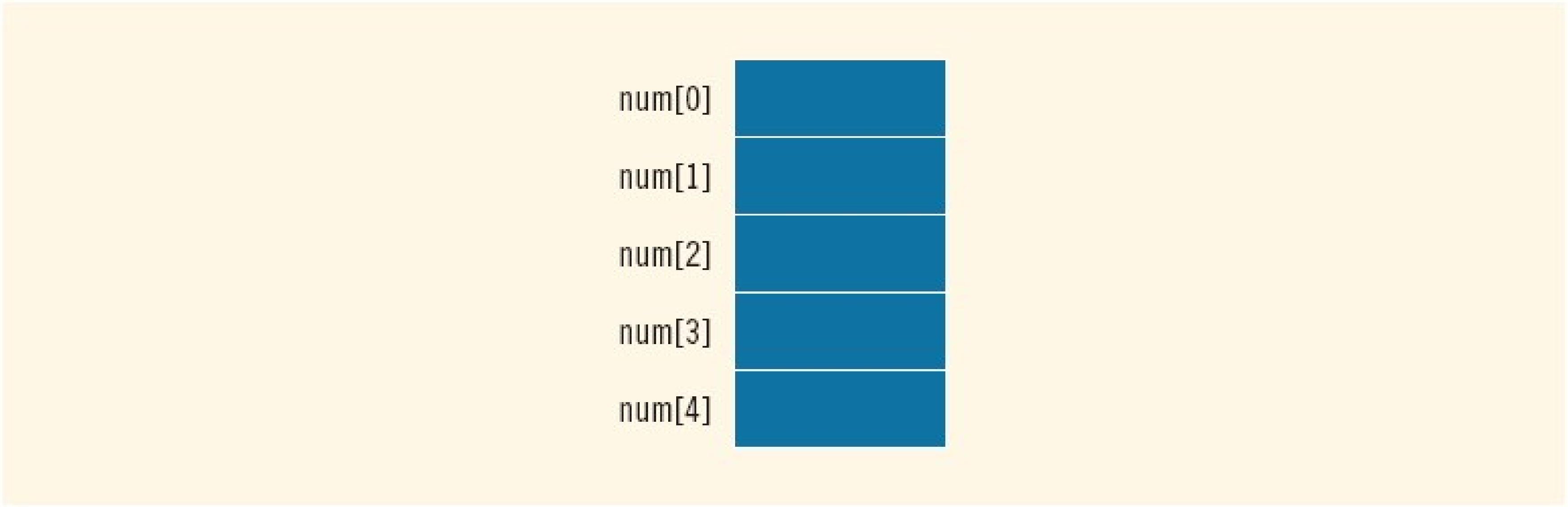


FIGURE 9-1 Array `num`

Accessing Array Components

- General syntax:

```
arrayName[indexExp]
```

  where `indexExp`, called an **index**, is any expression whose value is a nonnegative integer
- Index value specifies the position of the component in the array
- `[]` is the **array subscripting operator**
- The array index always starts at `0`

Accessing Array Components (continued)

```
int list[10];
```

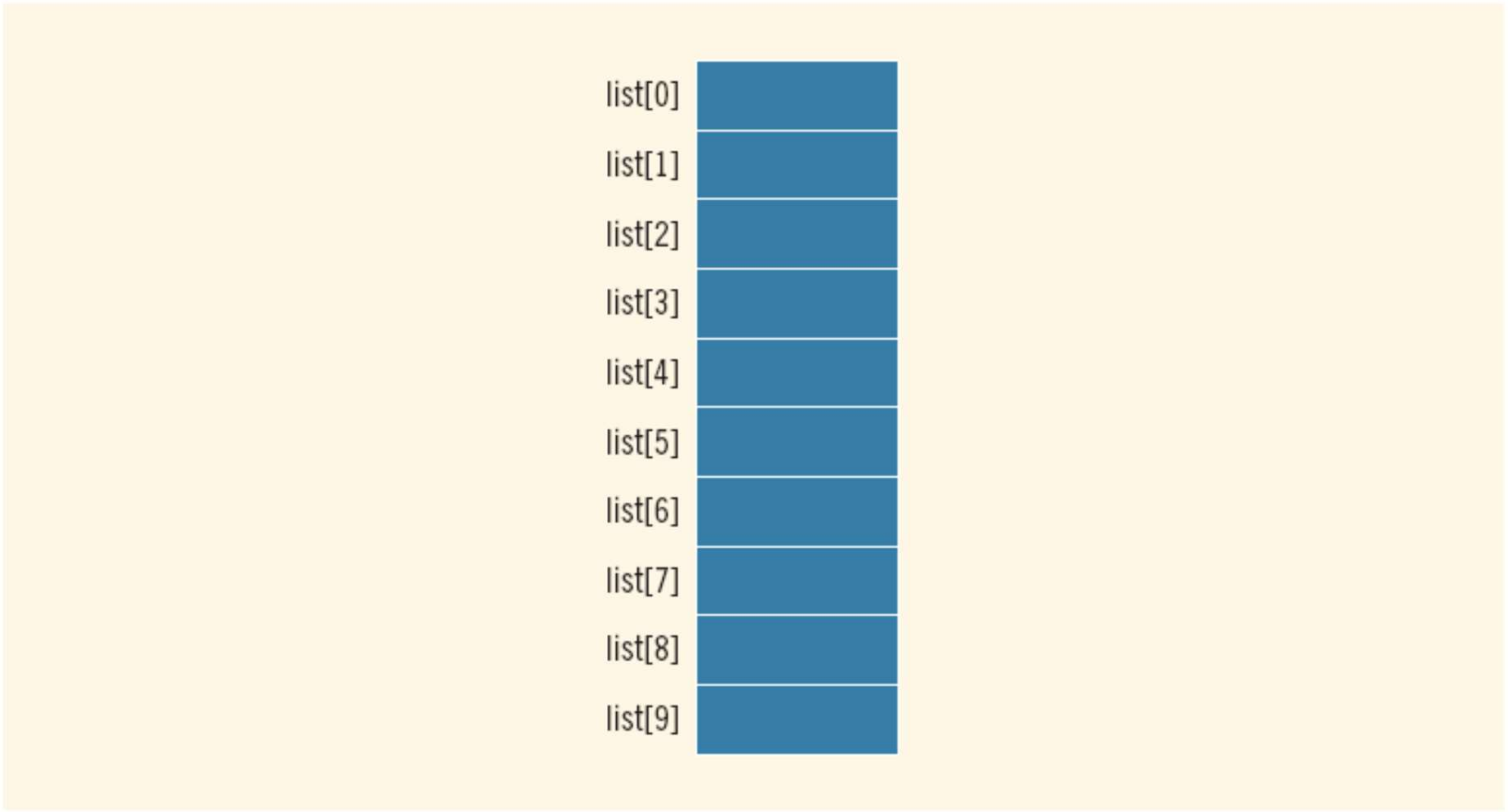


FIGURE 9-2 Array `list`

Accessing Array Components (continued)

```
list[5] = 34;
```

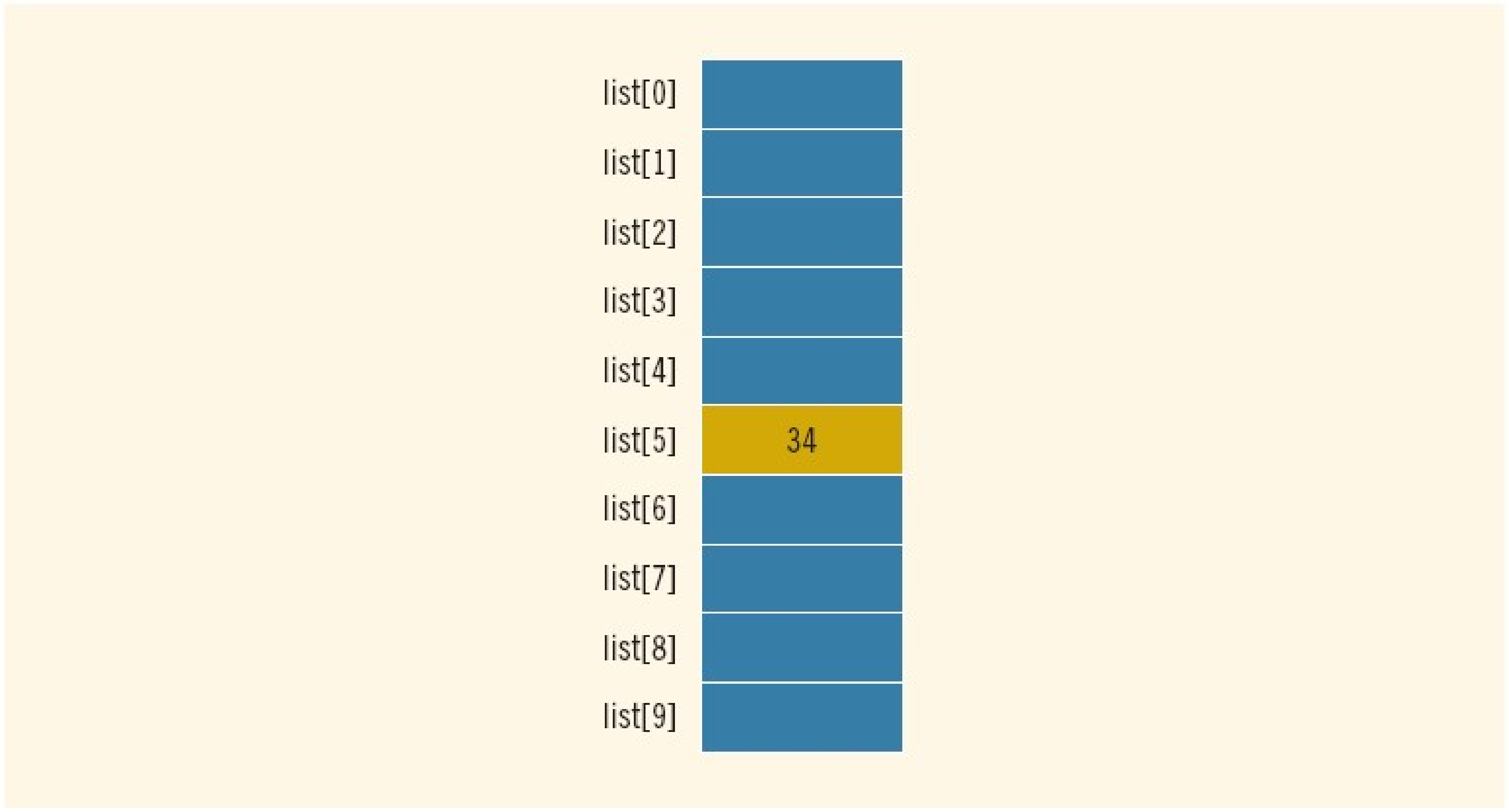


FIGURE 9-3 Array `list` after execution of the statement `list[5]= 34;`

Accessing Array Components (continued)

```
list[3] = 10;
list[6] = 35;
list[5] = list[3] + list[6];
```

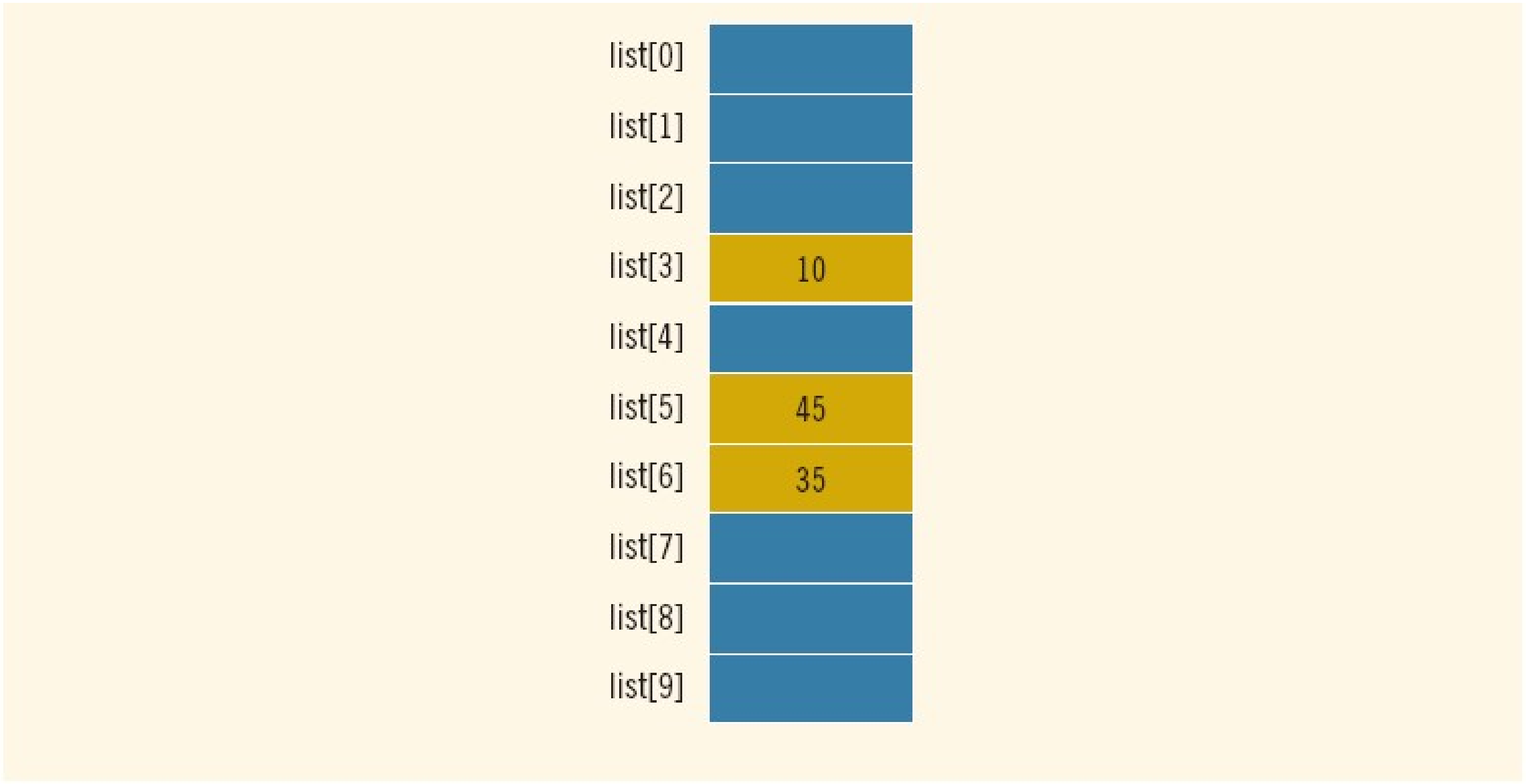
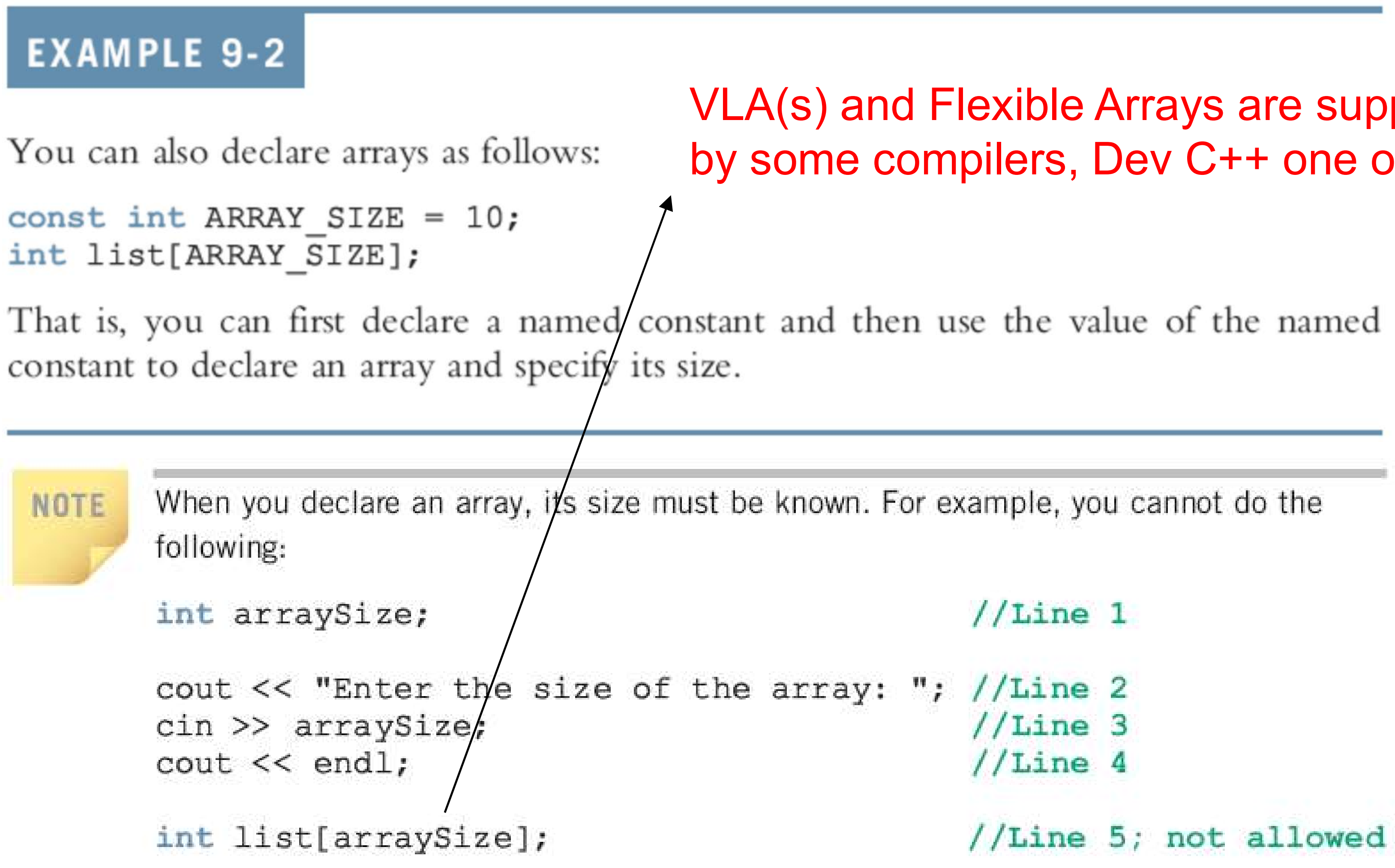


FIGURE 9-4 Array `list` after execution of the statements `list[3]= 10;`, `list[6]= 35;`, and `list[5] = list[3] + list[6];`

Accessing Array Components (continued)

VLA(s) and Flexible Arrays are supported by some compilers, Dev C++ one of them



EXAMPLE 9-2

You can also declare arrays as follows:

```
const int ARRAY_SIZE = 10;
int list[ARRAY_SIZE];
```

That is, you can first declare a named constant and then use the value of the named constant to declare an array and specify its size.

NOTE When you declare an array, its size must be known. For example, you cannot do the following:

```
int arraySize;                             //Line 1

cout << "Enter the size of the array: ";  //Line 2
cin >> arraySize;                          //Line 3
cout << endl;                              //Line 4

int list[arraySize];                       //Line 5; not allowed
```

Processing One-Dimensional Arrays

- Some basic operations performed on a one-dimensional array are:
  - Initializing
  - Inputting data
  - Outputting data stored in an array
  - Finding the largest and/or smallest element
- Each operation requires ability to step through the elements of the array
- Easily accomplished by a loop

Processing One-Dimensional Arrays (continued)

- Consider the declaration

```
int list[100];  //array of size 100
int i;
```

- Using `for` loops to access array elements:

```
    for (i = 0; i < 100; i++) //Line 1
        //process list[i]     //Line 2
```

- Example:

```
    for (i = 0; i < 100; i++) //Line 1
        cin >> list[i];       //Line 2
```

EXAMPLE 9-3

```
double sales[10];
int index;
double largestSale, sum, average;
```

Initializing an array:

```
for (index = 0; index < 10; index++)
    sales[index] = 0.0;
```

Reading data into an array:

```
for (index = 0; index < 10; index++)
    cin >> sales[index];
```

Printing an array:

```
for (index = 0; index < 10; index++)
    cout << sales[index] << " ";
```

Finding the sum and average of an array:

```
sum = 0;
for (index = 0; index < 10; index++)
    sum = sum + sales[index];

average = sum / 10;
```

Largest element in the array:

```
maxIndex = 0;
for (index = 1; index < 10; index++)
    if (sales[maxIndex] < sales[index])
        maxIndex = index;
largestSale = sales[maxIndex];
```

Array Index Out of Bounds

- If we have the statements:

```
    double num[10];
    int i;
```

- The component `num[i]` is valid if `i = 0, 1, 2, 3, 4, 5, 6, 7, 8,` or `9`
- The index of an array is in bounds if the `index >=0` and the `index <= ARRAY_SIZE-1`
  - Otherwise, we say the `index` is out of bounds
- In C++, there is no guard against indices that are out of bounds

Array Initialization During Declaration

- Arrays can be initialized during declaration
  - In this case, it is not necessary to specify the size of the array
    - Size determined by the number of initial values in the braces
- Example:

```
double sales[] = {12.25, 32.50, 16.90, 23, 45.68};
```

Partial Initialization of Arrays During Declaration

- The statement:

  ```
      int list[10] = {0};
  ```

  declares `list` to be an array of `10` components and initializes all of them to zero
- The statement:

  ```
      int list[10] = {8, 5, 12};
  ```

  declares `list` to be an array of `10` components, initializes `list[0]` to `8`, `list[1]` to `5`, `list[2]` to `12` and all other components are initialized to `0`

Partial Initialization of Arrays During Declaration (continued)

- The statement:

  ```
  int list[] = {5, 6, 3};
  ```

  declares `list` to be an array of `3` components and initializes `list[0]` to `5`, `list[1]` to `6`, and `list[2]` to `3`
- The statement:

  ```
  int list[25]= {4, 7};
  ```

  declares an array of `25` components; initializes `list[0]` to `4` and `list[1]` to `7`; all other components are initialized to `0`

Some Restrictions on Array Processing

- Consider the following statements:

```
int myList[5] = {0, 4, 8, 12, 16};  //Line 1
int yourList[5];  //Line 2
```

- C++ does not allow aggregate operations on an array:

```
yourList = myList;  //illegal
```

- Solution:

```
for (int index = 0; index < 5; index ++)
    yourList[index] = myList[index];
```

Some Restrictions on Array Processing (continued)

- The following is illegal too:

```
cin >> yourList; //illegal
```

- Solution:

```
for (int index = 0; index < 5; index ++)
     cin >> yourList[index];
```

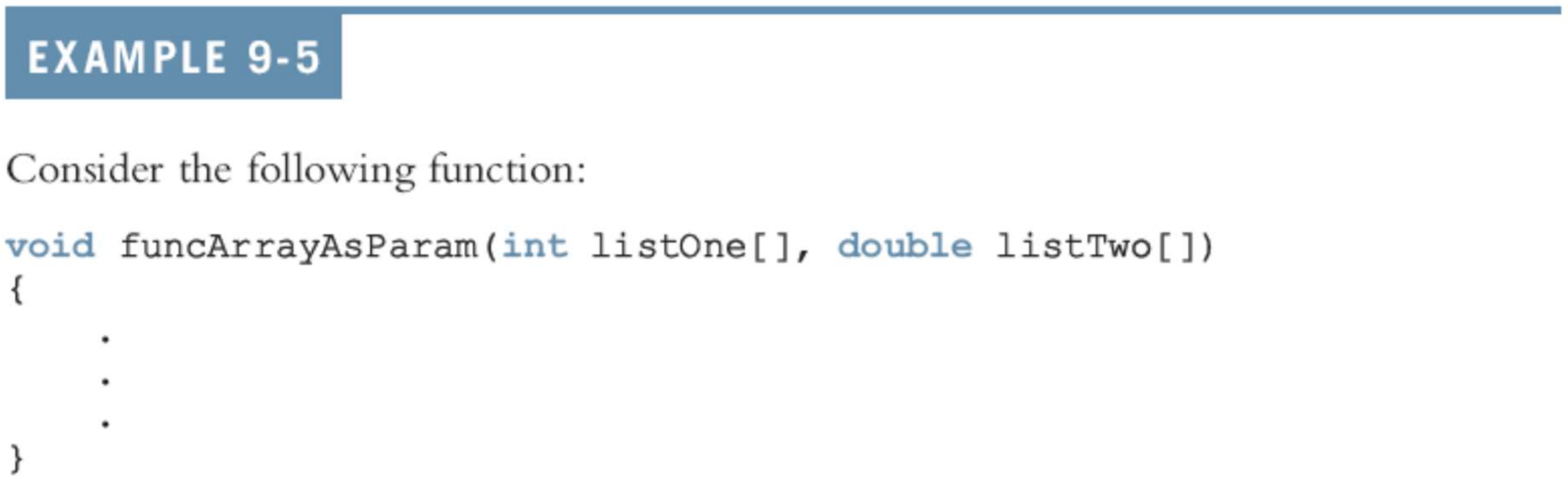
- The following statements are legal, but do not give the desired results:

```
cout << yourList;

if (myList <= yourList)
.
.
```

Arrays as Parameters to Functions

- Arrays are passed by reference only
- The symbol `&` is *not* used when declaring an array as a formal parameter
- The size of the array is usually omitted
  - If provided, it is ignored by the compiler



EXAMPLE 9-5

Consider the following function:

```
void funcArrayAsParam(int listOne[], double listTwo[])
{
      .
      .
      .
}
```

Base Address of an Array and Array in Computer Memory

- The base address of an array is the address, or memory location of the first array component
- If `list` is a one-dimensional array, its base address is the address of `list[0]`
- When we pass an array as a parameter, the base address of the actual array is passed to the formal parameter

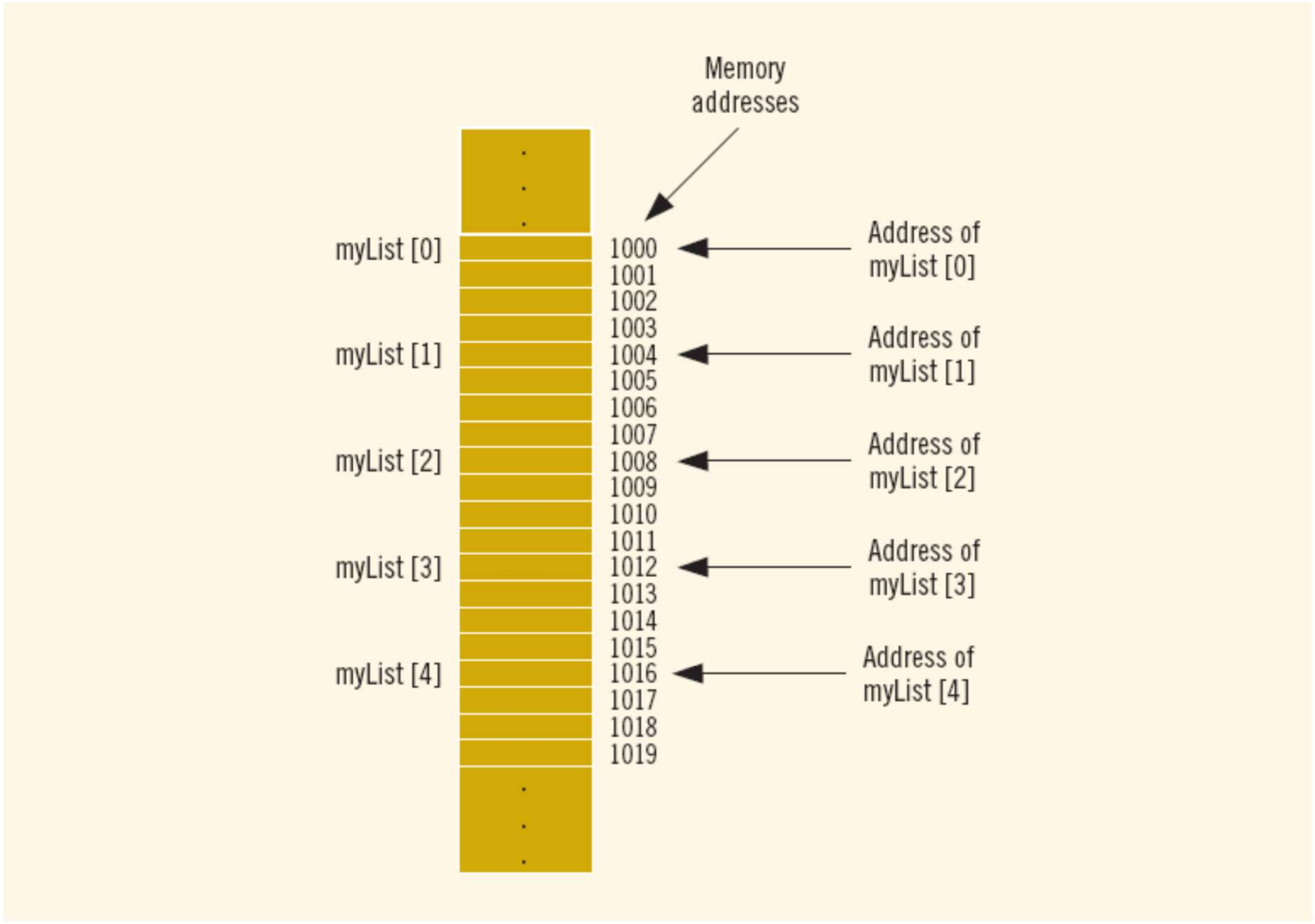


FIGURE 9-6 Array `myList` and the addresses of its components

Functions Cannot Return a Value of the Type Array

- C++ does not allow functions to return a value of the type array

Exercise – 02 Arrays (01)

Q1- Write a program that reads 10 integers in an array, and calculates the Min, Max and Sum of all array elements.

Arrays (01)

Q2 - Write a program that finds a given integer "n" in an array of 10 integers (read the values of array in main). If the value n is found, the program should display found, else display "not found".

Array (01)

- Q-3 Change the previous program so that you have function that tells you if a number is present in array or not

```
bool isFound(int a[ ], int n, int size)
{
 // implement your code here
}
```

Array (01)

- Q4 Write a function that accepts two arrays and displays the common elements of the arrays.

Array (01) - Sorting

- Q-5 Write a program to Sort a given array of integers 10 integers.
- Q-6 Write a C++ program to find the most occurring element in an array of integers
- Q-7 Write a C++ program to find and print all unique elements of a given array of integers