CS118 – Programming Fundamentals

Lecture # 21
Monday, November 11, 2019
FALL 2019
FAST – NUCES, Faisalabad Campus

Zain Iqbal

Objectives

In this chapter, you will:

- Learn about arrays
- Explore how to declare and manipulate data into arrays
- Learn about "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

Introduction

- 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

Example


```
//Program that takes five numbers print their average
//and the numbers again
#include<iostream>
using namespace std;
int main()
{
    int n1, n2, n3, n4, n5;
    double average;
    cout << "Enter five integers : " ;
    cin >> n1 >> n2 >> n3 >> n4 >> n5 ;

    average = (n1 + n2 + n3 + n4 + n5) / 5.0 ;

    cout << "The average of the given numbers = " << average ;
    cout << "\nand the numbers are n1 = " << n1 << " n2 = " << n2
        << " n3 = " << n3 << " n4 = " << n4
        << " n5 = " << n5 << endl ;
    return 0;
}
```

Example

- Five variables must be declared because the numbers are to be printed later
- All variables are of type int—that is, of the same data type
- The way in which these variables are declared indicates that the variables to store these numbers all have the same name—except the last character, which is a number

Arrays

- **Array:** A collection of a fixed number of components where 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:

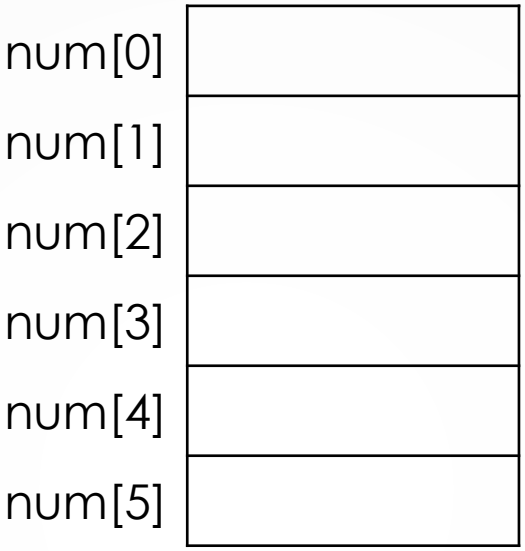
```
dataType arrayName[intExp];
```

  - **intExp** evaluates to a positive integer

Arrays

Example:

```
int num[6];
```



	num[0]	num[1]	num[2]	num[3]	num[4]	num[5]
num						

	[0]	[1]	[2]	[3]	[4]	[5]
num						

Defining Arrays

- When defining arrays, specify
  - Name
  - Type of array
  - Number of elements

    arrayType arrayName[numberOfElements];
  - Examples:

    int c[10];

    float myArray[3284];
- Defining multiple arrays of same type
  - Format similar to regular variables
  - Example:

    int b[100], x[27];

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 (cont'd.)

int list[8];

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
list								

list[5] = 75;

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
list						75		

Accessing Array Components (cont'd.)

list[3] = 20 ;

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
list				20				

list[6]=100;

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
list				20			100	

list[2]= list[3] + list[6];

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
list			120	20			100	

Accessing Array Components (cont'd.)

Suppose `i` is an `int` variable. Then, the assignment statement:

```
list[3] = 63;
```

is equivalent to the assignment statements:

```
i = 3;
list[i] = 63;
```

If `i` is `4`, then the assignment statement:

```
list[2 * i - 3] = 58;
```

stores `58` in `list[5]` because `2 * i - 3` evaluates to `5`. The index expression is evaluated first, giving the position of the component in the array.

Accessing Array Components (cont'd.)

Array can also be declared as

```
const int SIZE_OF_ARRAY = 20;
int array[SIZE_OF_ARRAY] ;
```

First declare a named constant and then use it to declare an array of this specific size.

When an array is declared its size must be known. You cannot do this:

```
int arr_size;
cout << "Enter size of array ";
cin >> arr_size;

int arr[arr_size];
```

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 (cont'd.)

- 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
```

Processing One-Dimensional Arrays (cont'd.)


```
double scores[10];
int index;
double largest, sum, average;
```

Initializing an array

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

Reading data into array

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

Printing the array

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

Processing One-Dimensional Arrays (cont'd.)

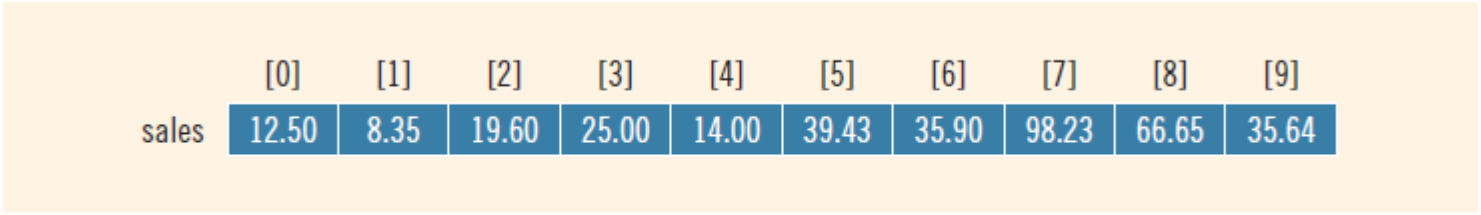
Finding sum and average of an array

```
sum = 0.0;
for (index = 0 ; index < 10 ; ++index)
    sum = sum + scores[index];
average = sum / 10;
```

Largest element in the array

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

Processing One-Dimensional Arrays



	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
sales	12.50	8.35	19.60	25.00	14.00	39.43	35.90	98.23	66.65	35.64

FIGURE 9-6 Array `sales`

index	maxIndex	sales [maxIndex]	sales [index]	sales[maxIndex] < sales[index]
1	0	12.50	8.35	12.50 < 8.35 is false
2	0	12.50	19.60	12.50 < 19.60 is true; maxIndex = 2
3	2	19.60	25.00	19.60 < 25.00 is true; maxIndex = 3
4	3	25.00	14.00	25.00 < 14.00 is false
5	3	25.00	39.43	25.00 < 39.43 is true; maxIndex = 5
6	5	39.43	35.90	39.43 < 35.90 is false
7	5	39.43	98.23	39.43 < 98.23 is true; maxIndex = 7
8	7	98.23	66.65	98.23 < 66.65 is false
9	7	98.23	35.64	98.23 < 35.64 is false

After the `for` loop executes, `maxIndex = 7`, giving the index of the largest element in the array `sales`. Thus, `largestSale = sales[maxIndex] = 98.23`.

Questions