CS118 – Programming Fundamentals

Lecture # 25
Monday, November 25, 2019
FALL 2019
FAST – NUCES, Faisalabad Campus

Zain Iqbal

Multidimensional Arrays

- Multidimensional array: collection of a fixed number of elements (called components) arranged in n dimensions (**n >= 1**)
- Also called an n-dimensional array
- Declaration syntax:

```
dataType arrayName[intExp1][intExp2] ... [intExpn];
```

- To access a component:

```
arrayName[indexExp1][indexExp2] ... [indexExpn]
```

Example

For example
double carDealers[10][5][5];

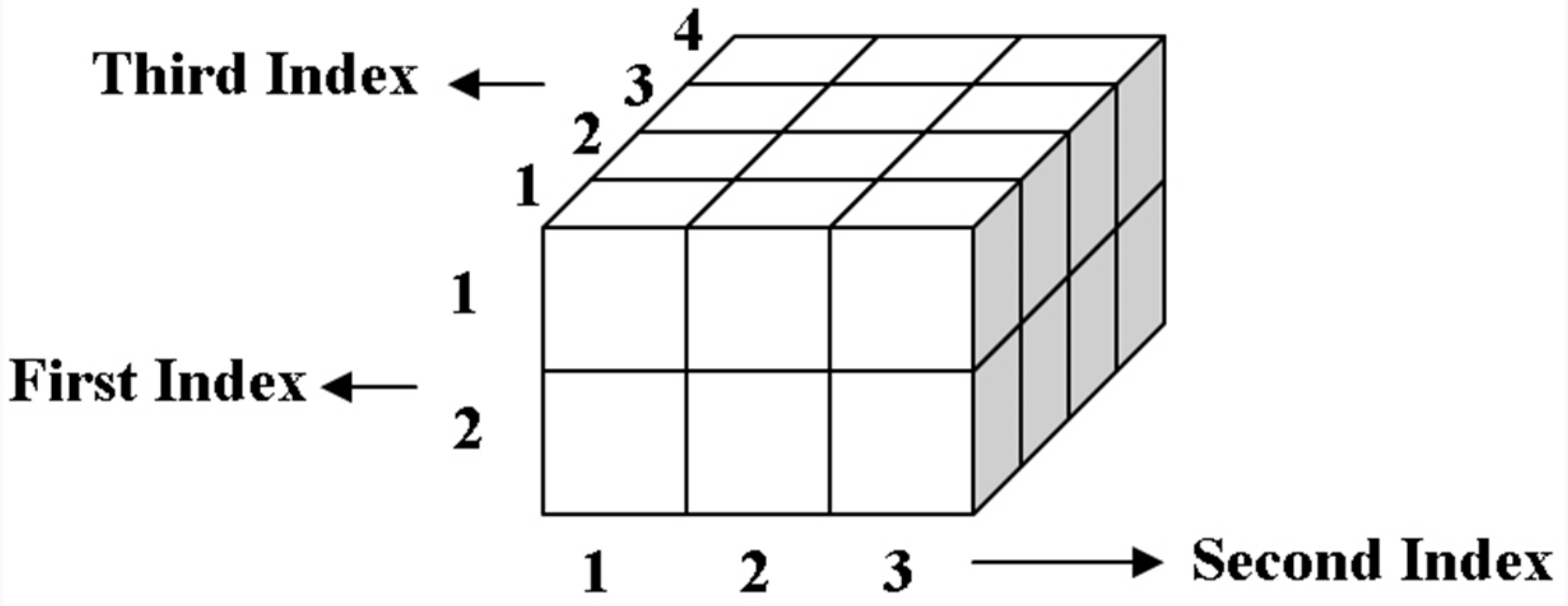
- The base address of the array carDealers is the address of the first array component—that is, the address of carDealers[0][0][0]
- The total number of components in the array carDealers is 10 * 5 * 5 = 250

- carDealer[5][3][2] = 15009.65; // sets the value of carDealer[5][3][2] to 15009.65

```
 for(int i=0 ; i<10 ; i++)
         for(int j=0 ; j<5 ; j++)
                 for(int k=0 ; k<5 ; k++)
                         carDealer[i][j][k] = 0.0 ;
```

Initialize all of the elements in array to 0.0

Example



Three-dimensional array with twenty four elements

Multidimensional Arrays (cont'd.)

- When declaring a multidimensional array as a formal parameter in a function
  - Can omit size of first dimension but not other dimensions
- As parameters, multidimensional arrays are passed by reference only
- A function cannot return a value of the type array
- There is no check if the array indices are within bounds

String Datatype

Basic Functions

The string Type

- To use the data type string, the program must include the header file string

  #include <string>

- The statement

  string name = "William Jacob";

  declares name to be string variable and also initializes name to "William Jacob".

- The position of the first character, 'W', in name is 0, the position of the second character, 'i', is 1, and so on
- The variable name is capable of storing (just about) any size string

String basic functions

- Binary operator + (to allow the string concatenation operation), and the array index (subscript) operator [], have been defined for the data type string
- Suppose we have the following declarations

  string str1, str2, str3;

- The statement

  str1 = "Hello There“ ;

  stores the string "Hello There" in str1.

- The statement

  str2 = str1;

  copies the value of str1 into str2.

String basic functions

- If **str1 = "Sunny"**, the statement

  str2 = str1 + " Day";

  stores the string "Sunny Day" into str2.

- If **str1 = "Hello“** and **str2 = "There“** then

  str3 = str1 + " " + str2;

  stores "Hello There" into **str3**

- This statement is equivalent to the statement

  str3 = str1 + ' ' + str2;

String basic functions

- The statement

  str1 = str1 + "Mickey“ ;

  updates the value of str1 by appending the string "Mickey" to its old value

- If **str1 = "Hello there"**, the statement

  str1[6] = 'T‘ ;

  replaces the character t with the character T.

The length Function

- The **length** function returns the number of characters currently in the string
- The value returned is an unsigned integer
- The syntax to call the length function is:

  strVar.length()

  where strVar is variable of the type string

- The function length has no arguments

String datatype

- Consider the following statements:

```
string firstName ;
string name ;
string str ;

firstName = “Elizabeth";
name = firstName + " Taylor";
str = "It is sunny outside.";
```

Statement	Effect
cout<<firstName.length()<<endl;	Outputs 9
cout<<name.length()<<endl;	Outputs 16
cout<<str.length()<<endl;	Outputs 20

The size Function

- The function size is same as the function length
- Both these functions return the same value
- The syntax to call the function size is:

  strVar.size()

  where **strVar** is variable of the type string.
- As in the case of the function length, the function size has no arguments

The find **Function**

- The find function searches a string to find the first occurrence of a particular substring and returns an unsigned integer value (of type **string::size_type**) giving the result of the search
- The syntax to call the function find is:
  strVar.find(strExp)
- Where strVar is a string variable and strExp is a string expression evaluating to a string
  - The string expression, strExp, can also be a character
- If the search is successful, the function find returns the position in strVar where the match begins
- For the search to be successful, the match must be exact
- If the search is unsuccessful, the function returns the special value **string::npos** (“**not a position within the string**”).

String datatype

- The following are valid calls to the function find

```
str1.find(str2)
str1.find("the")
str1.find('a')
str1.find(str2+"xyz")
str1.find(str2+'b')
```

String datatype

```
string   sentence;
string   str;
string::size_type position;

sentence = "It is cloudy and warm.";
str = "cloudy";
```

Statement	**Effect**
`cout<<sentence.find("is")<<endl;`	Outputs 3
`cout<<sentence.find("and")<<endl;`	Outputs 13
`cout<<sentence.find('s')<<endl;`	Outputs 4
`cout<<sentence.find(str)<<endl;`	Outputs 6
`cout<<sentence.find("the")<<endl;`	Outputs the value of string::nops
`position = sentence.find("warm");`	Assigns 17 to position

The substr **Function**

- The **substr** function returns a particular substring of a string
- The syntax to call the function **substr** is:

  strVar.substr(expr1,expr2)

  where expr1 and expr2 are expressions evaluating to unsigned integers.

- The expression expr1 specifies a position within the string (starting position of the substring). The expression expr2 specifies the length of the substring to be returned.

string sentence;
string str;

sentence = "It is cloudy and warm.";

Statement	**Effect**
cout<<sentence.substr(0,5) << endl ;	Outputs: It is
cout<<sentence.substr(6,6) << endl ;	Outputs: cloudy
cout<<sentence.substr(6,16) << endl ;	Outputs: cloudy and warm.
cout<<sentence.substr(3,6) << endl ;	Outputs: is clo
str = sentence.substr(0,8);	str = "It is cl"
str = sentence.substr(2,10);	str = " is cloudy"

The Function swap

- The function **swap** is used to swap—that is, interchange—the contents of two string variables
- The syntax to use the function swap is

  strVar1.swap(strVar2);

  where **strVar1** and **strVar2** are string variables.

- Suppose you have the following statements:

```
string str1 = "Warm";
string str2 = "Cold";
```

- After the following statement executes, the value of **str1** is **"Cold"** and the value of **str2** is **"Warm"**.

```
str1.swap(str2);
```

Questions