**ITCS 1212L**

**Lab Lessons 9**

# Arrays

**Learning Objectives:**

* **Learning arrays.**

**Overview of the Arrays:**

An array is a series of elements of the same type placed in contiguous memory locations that can be individually referenced by adding an index to a unique identifier.  
  
That means that, for example, five values of type int can be declared as an array without having to declare 5 different variables (each with its own identifier). Instead, using an array, the five int values are stored in contiguous memory locations, and all five can be accessed using the same identifier, with the proper index.  
  
For example, an array containing 5 integer values of type int called foo could be represented as:  
  
http://www.cplusplus.com/doc/tutorial/arrays/arrays1.png   
where each blank panel represents an element of the array. In this case, these are values of type int. These elements are numbered from 0 to 4, being 0 the first and 4 the last; In C++, the first element in an array is always numbered with a zero (not a one), no matter its length.  
  
Like a regular variable, an array must be declared before it is used. A typical declaration for an array in C++ is:  
  
type name [elements];  
  
where type is a valid type (such as int, float...), name is a valid identifier and the elements field (which is always enclosed in square brackets []), specifies the length of the array in terms of the number of elements.  
  
Therefore, the foo array, with five elements of type int, can be declared as:

|  |  |  |
| --- | --- | --- |
|  | int foo [5]; |  |

NOTE: The elements field within square brackets [], representing the number of elements in the array, must be a *constant expression*, since arrays are blocks of static memory whose size must be determined at compile time, before the program runs.

**Two Dimensional Arrays:**

The simplest form of the multidimensional array is the two-dimensional array. A two-dimensional array is, in essence, a list of one-dimensional arrays. To declare a two-dimensional integer array of size x,y, you would write something as follows:

type arrayName [ x ][ y ];

Where **type** can be any valid C++ data type and **arrayName** will be a valid C++ identifier.

A two-dimensional array can be think as a table, which will have x number of rows and y number of columns. A 2-dimensional array **a**, which contains three rows and four columns can be shown as below:



Thus, every element in array a is identified by an element name of the form **a[ i ][ j ]**, where a is the name of the array, and i and j are the subscripts that uniquely identify each element in a.

## Initializing Two-Dimensional Arrays:

2D arrays may be initialized by specifying bracketed values for each row. Following is an array with 3 rows and each row have 4 columns.

int a[3][4] = {

{0, 1, 2, 3} , /\* initializers for row indexed by 0 \*/

{4, 5, 6, 7} , /\* initializers for row indexed by 1 \*/

{8, 9, 10, 11} /\* initializers for row indexed by 2 \*/

};

The nested braces, which indicate the intended row, are optional. The following initialization is equivalent to previous example:

int a[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};

## Accessing Two-Dimensional Array Elements:

An element in 2-dimensional array is accessed by using the subscripts, i.e., row index and column index of the array. For example:

int val = a[2][3];

The above statement will take 4th element from the 3rd row of the array. You can verify it in the above diagram.

#include <iostream>

using namespace std;

int main ()

{

// an array with 5 rows and 2 columns.

int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};

// output each array element's value

for ( int i = 0; i < 5; i++ )

for ( int j = 0; j < 2; j++ )

{

cout << "a[" << i << "][" << j << "]: ";

cout << a[i][j]<< endl;

}

return 0;

}

When the above code is compiled and executed, it produces the following result:

a[0][0]: 0

a[0][1]: 0

a[1][0]: 1

a[1][1]: 2

a[2][0]: 2

a[2][1]: 4

a[3][0]: 3

a[3][1]: 6

a[4][0]: 4

a[4][1]: 8