UNIVERSITY OF MASSACHUSETTS
DARTMOUTH

ECE160: Foundations of Computer Engineering I

Lecture #22 – Array (2)

Instructor: Dr. Liudong Xing
SENG-213C, lxing@umassd.edu
ECE Dept.

Administrative Issues (4/3)

- Lab#10 starts today and due 5pm, Wednesday, April 5.

- Today's topics
  - Arrays (Finish L#21;Then L#22)

Review of Lectures #21

- Arrays allow to represent a group of elements of the same type as a single unit.
- Like variables, before use, an array has to be defined and declared

```
int myarray[5];
```

- Three ways to initialize an array
  - At the definition time

```
int myarray[5]={1,2,10,15,0};
```

  - Inputting values from the keyboard
  - Assigning values

```
for(int i=0; i< 5; i++)
{
        scanf("%d", &myarray[i]);
}
```

```
for(int i=0; i< 5; i++)
{
        myarray[i]=i*2+1;
}
```

Outline

- Exchanging values in arrays
- Multidimensional arrays

Exchanging Values

- When you exchange variables, you swap the values of elements without knowing what is in them
- Example: assume

int num_array[6];

  - How to swap num_array[2] and num_array[4]?

```
num_array[2]=num_array[4];
num_array[4]=num_array[2];
```

The original value of num_array[2] is destroyed before we can move it!

Exchanging Values (Cont'd)

- **Solution:** to use a temporary variable to hold the value of num_array[2] before the moving

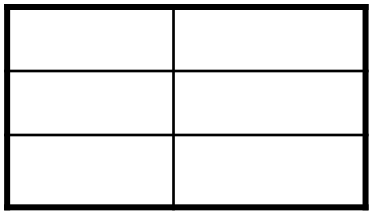
```
temp = num_array[2];
num_array[2] = num_array[4];
num_array[4] = temp;
```

Agenda

- Exchanging values in arrays
- Multidimensional arrays

Concepts

- The arrays we discussed so far are 1-D arrays
  - Data are organized linearly in one direction
- Many applications require data to be stored in more than one dimension
- Example: tables (columns and rows)→ 2-D arrays



rows
(1st dimension)

columns
(2nd dimension)

Declaration and Definition

int myarray[3][2];

- The first number specifies the rows of the array
- The second number specifies the columns of the array.

- Indexing:
  - The first index for both row and column is ZERO!
  - myarray[0][1]: element in the first row, second column
  - myarray[1][0]: element in the second row, first column

Declaration and Definition (Syntax)

```
element_type   array_name [num _of_rows][num_of_columns];
```

- element_type specifies the type of the array's elements, e.g., int, float, double **(cannot be void type!)**
- array_name: the name of the array
  - Array names are C identifiers
- number_of_elements: the size of the array →
  num_of_rows * num_of_columns

Initialization

- Three ways to initialize a multi-D array
  - **At the definition time**
  - Inputting values from the keyboard
  - Assigning values

Initialization (Way#1)

```
int myarray [3][2] =
{
    {0,10},     /*1st row*/
    {10,15},    /*2nd row*/
    {1,2}       /*3rd row*/
};
```

- Nest the initializers in braces to show the exact nature of the array
- Use commas between elements in each row and commas between rows

```
int myarray [3][2] = {0,10, 10,15, 1,2};
```

- Put all the initializers in braces
- Use commas between elements

Initialization (Way #1: Cont'd)

- 1-D arrays (L#21 review)
  - int myarray[] = {1,2,10,15,0};
    - If an array size is omitted from the definition with an initializer list, the size of the array will be the number of elements in the initializer list
    - The array is automatically declared to have a size of 5

- Multi-D arrays
  - Only the size of the first dimension can be omitted, all others must be specified!

```
int myarray [][2] =
            {
        {0,10},
        {10,15},
        {1,2}
    };
```

Initialization (Way #1: Cont'd)

- To initialize an array to all 0s, supply just the first 0:

```
int myarray[3][2] = {0};
```

Exercise (1)

- Based on the following statement:

  int a[3][1]={1,2,3}, b[6], c[3][2], d[2][3];

  determine whether each of the following statements is true or false?

  a) ___Array c[3][2] contains 3+2 = 5 elements

  b) ___The 1-D array b[6] can be used to store all data saved in the 2-D array d[2][3]

  c) ___Array c[3][2] contains six elements: c[0], c[1], c[2], c[3],c[4], c[5]

  d) ___Array c[3][2] contains 3 1-D sub-arrays while array d[2][3] contains 2 1-D sub-arrays

  e) ___Array d[2][3] contains six elements: d[1][1], d[1][2], d[1][3], d[2][1], d[2][2], d[2][3]

Exercise (2)

- Find error(s), if any, in the following statements:
  - int a[2][0];
  - float city(23)(25), town(12)(21);
  - int a[2,3]=(23, 34, 45, 56);
  - double main[4][1]= {11, 22, 33, 44};

3-D Arrays

int myarray[2][3][2]=

{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};

- The first number is the number of planes; each plane consists of rows and columns (2-D array)
- The second number is the number of rows
- The third number is the number of columns
- Order:

  [0][0][0], [0][0][1],
  [0][1][0], [0][1][1],
  [0][2][0], [0][2][1],
  [1][0][0], [1][0][1],
  [1][1][0], [1][1][1],
  [1][2][0], [1][2][1]

```
int myarray[2][3][2]=
    {
        { /*plane 0*/
                {1, 2},
                { 3, 4},
                { 5, 6}
        },
        {/*plane 1*/
                {7, 8},
                {9, 10},
                {11, 12}
        }
    };
```

Exercise (3)

- 3-D array:

```
int myarray[2][3][2]={1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
```

- What are the values of
  - myarray[1][1][1]?
  - myarray[0][2][1]?
  - myarray[2][1][0]?

```
int myarray[2][3][2]=
    {
        { /*plane 0*/
            {1, 2},
            { 3, 4},
            { 5, 6}
        },
        {/*plane 1*/
            {7, 8},
            {9, 10},
            {11, 12}
        }
    };
```

3-D Arrays

- To initialize an array to all 0s, supply just the first 0:

```
int myarray[2][3][2] = {0};
```

Initialization (Agenda)

- Three ways to initialize a multi-D array
  - At the definition time
  - **Inputting values from the keyboard**
  - **Assigning values**

Array Initialization (Way #2): Inputting Values into the Array

- Usually done in nested for loops
- Example: assume

int myarray[3][2];

```
for(int i=0; i< 3; i++)    /*i is row index*/
        for (int j=0; j<2; j++)    /*j is col index*/
            scanf("%d", &myarray[i][j]);
```

Array Initialization (Way #3): Value Assignment

- Usually done in nested for loops
- Example: assume

int myarray[3][2];

```
for(int i=0; i< 3; i++) /*i is row index*/
        for (int j=0; j<2; j++) /*j is col index*/
        myarray[i][j]=2*i+j;
```

- Assign values to individual elements:

```
myarray[2][1] = 37;
```

Exercise (4)

- Read a 2X3 array from the keyboard. Then print out the elements of the second row on the screen.

```
#include <stdio.h>
void main(void)
{
    int num_array[2][3], i, j;

    printf("Read the array from the keyboard:\n");

    printf("please print out the second row:\n");

}
```

Summary of Lectures #22

- How to exchange contents of two array elements?
  - Through a temporary variable
- Multi-dimensional arrays
  - Before use, a multi-D array has to be defined and declared
  - Three ways to initialize a multi-D array
    - At the definition time
    - Inputting values from the keyboard
    - Assigning values

Things To Do

- Review lectures #21 and #22

Next Topics

- Arrays and functions