Lecture 2.2

Arrays (cont.)

How to initialize an array? (cont)

- An array can be initialized with an explicit **initialization list** in its definition.

Example:

- int myArray[5] = { 6, 7, 888, 987, 0 };

How to initialize an array?

Example

- int myArray[10];
- int i;

can do:

- for (i =0; i < 10; i++)
  myArray [i] = 0; // execution time

or:

- int myArray[10] = { 0 }; //compile time,
  //better
  //elements are initialized implicitly

How to initialize an Array?

Example

- int myArray [3] = { 3, 4, 5 };

// make sure to provide exact number

or

- int myArray [ ] = {3 ,4, 5 };

// creates an three-element array

Initializing arrays of characters

Example

- char charArray [6] = { ‘h’,’e’,’l’,’l’,’o’, ‘\0’};

or

- char charArray1[ ] = {‘h’,’e’,’l’,’l’,’o’, ‘\0’};

or

- char charArray2[ ] = “hello”;

// size is determined by the compiler

Special character at the end of the string is known as ‘null character’ which is used to signify end of the string.

Initializing arrays of characters (cont)

Example

- char exOne[ ] = “bula”;
- int j;
- for (j = 0; j < 5; j++)
  cout << exOne[j] ;

- **This will print**
- bula

Multidimensional arrays

- Commonly used to represent tables/matrices

Examples

- float temperatures[12][31];
- /* Used to store temperature data for a year */
- char daysofweek[7][10];

Accessing multidimensional arrays

- The common way is to use nested for loops.

```
#define MAXI 50;
#define MAXJ 75;
int i , j ;
float values[ MAXI ][ MAXJ ];
for (i = 0; i < MAXI; i++) {
      for (j = 0; j < MAXJ; j++) {
            values[ i ] [ j ] = whatever; }
}
```

Accessing multidimensional arrays (cont.)

- Notice the expression
  values[i] [j] = whatever;
  this is the correct form to reference a multidimensional array,
- The expression
  values [i, j] would be **wrong**!

Common errors for arrays

- The most common cause of writing/reading to invalid array indexes are errors in loop limits.

```
int i;
float b[10];
for (i < 0 ; i <= 10; i++) {
   b[i] = 3.14 * i * i;
}
```

- This loop should use "<" rather than "<="

Initializing multidimensional arrays

- Example

```
int a[2][3] = { {1,2,3} , {4, 5, 6} },
b[2][3] = { 1,2,3,4,5},
c[2][3] = { {1,2 }, {4} };
```

- Note: If there are not enough initializers for a given row, the remaining elements of that row are initialized to 0.

Initializing multidimensional arrays

- Row 0 a [0][0] a[0][1] a[0][2]
- Row 1 a [1][0] a[1][1] a[1][2]
- //printing the array with the following code

```
for (int i = 0; i< 2 ; i++){
    for (into j = 0; j<3 ; j++)
      cout << a[i][j]<< ' ';
}
```

Initializing multidimensional arrays

- Values for array a
  1 2 3
  4 5 6
- Values for array b
  1 2 3
  4 5 0
- Values for array c
  1 2 0
  4 0 0

Passing multidimensional arrays to functions

- When passing array parameters, the size of the first subscript of a multidimensional array is not required, but all subsequent subscript sizes are required.

Example:

- Function Declaration:

```
void printArray(int array[][3]);
```

- Function Call:

```
int myarray[4][3];
printArray(myarray);
```