Two Dimensional Arrays

Declaration of 2D-Arrays

2d arrays in C are somewhat similar to those in Java. To declare a 10 by 20 2d-array of integers, you would code:

```
int myArray[10][20];  /*An int array named myArray with 10 rows, 20
columns */
```

To access the (i,j) member of an array, you would code:

```
myArray[i][j] = 10;
```

It's important to realize that you need to keep track of how big your array is yourself. In general, there are no bounds checks in C,
and no way to determine the size of your array.

Passing 2D-Arrays to Functions

There are several ways to pass an array to a function, but by the end of this, you'll see that it's easier not to do it, especially if many routines require access to the array.

The easiest way to demonstrate this is by an example. Let's have a function that zero's out all members of an 2-d array of ints with numRows rows, and numCols columns.

```
Your first attempt might be to write the function header as:

void zeroArray(int m[][], int numRows, int numCols)
{
      ...
}
```

Unfortunately the C compiler will complain as you need to specify the # of columns (at the minimum), so you could say (and there are other ways also):

```
void zeroArray(int m[][20], int numRows, int numCols)
{
      ...
}

or


void zeroArray(int m[10][20], int numRows, int numCols)
{
      ...
}
```

In each of these last 2 cases, the address of the array is passed to the function, and therefore the contents of the array can be modified in the functions.

File Variables

As you can see, it becomes quite tedious and confusing to pass arrays to every function that requires it. Instead, you can create your array as a file variable. Then every function within that file that requires that array will have access to it. The key point is that only functions in the same file as where the variable resides can access it. Here is an example:

```
#define ROWS 10
#define COLS 20

int myArray[ROWS][COLS];  /*an array accessible only to functions within
this file.*/

...

void fillArray()
{
      int r;
      int c;

      for (r = 0; r < ROWS; r++)
      {
            for (c = 0; c < COLS; c++)
            {
                  myArray[r][c] = r*c + c; /*accessing the array */
            }
      }
}

....
```

What we have done here is similar to private instance and class variables that are declared inside a Java class.

In Java, private variables declared inside a class are only accessible to other members of that class. In C, a file variable is accessible only to functions within the file the variable is declared in (more accurate details will follow when we discuss multi-file compilation).

As a final example, consider the following snippet of Java code:

```
class Chess
{
      private static char board[][];
      …
      void init()
      {
            board = new char[8][8];
            …
      }

      void move()
      {
            board[0][2]='X';
      }
      …
}
```

The corresponding C snippet is:

```
char board[8][8];

void move(…)
{
      board[0][2] = 'X';
}
…
```