ENGG1003 - PASS Session 8

Mitchell Deltoer

Monday	**14-15**	**ES238**
Wednesday	**12-13**	**ES238**
Thursday	**10-11**	**MCLG42**

Multi-dimensional arrays

The simplest form of multidimensional array is the two-dimensional array. In essence a 2D array is a list of one-dimensional arrays. The following is an example of declaring a 2D array of size `[x][y]`:

```
type arrayName [x][y];
```

where type is the array data type and arrayName is a valid C variable name. A 2D array can be considered as a table which will have `x` number of rows and `y` number of columns. A 2D array `a`, which contains three rows and four columns can be shown as follows

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

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

Multidimensional arrays may be initialized by specifying bracketed values for each row. The following initialises an array with 3 rows and each row has 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 line breaks in the above code are just for readability and are not required.

Just like with 1D arrays, an element in a 2D array is accessed by using the array indexes like so

```
int val = a[2][3];   // store the value 11 from the array in val
```

Like with 1D arrays, loops will be needed if you want to access each element of a 2D array. However, an additional nested loop is required for each additional dimension of the array, i.e. one nested loop for a 2D array. The following is an example of printing each value of the 2D array `a`

```
int i, j;
for ( i = 0; i < 3; i++ ) {      // loop for the row index
   for ( j = 0; j < 4; j++ ) {   // loop for the column index
      printf("a[%d][%d] = %d\n", i,j, a[i][j] );
   }
}
```

Practice Programming

Task 1: 2D Array Sum

Write a C program that finds the sum of all of the elements in a 2D array. Use the following initialisation for your initial testing.

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

Task 2: Row Sum

Write a C program that finds the smallest number in each row of a 2D array. The result for each *ith* row should be stored in the *ith* element of a 1D array. Use the same initialisation as previous for initial testing.

Task 3: 2D Matrix Inverse

Note: You won't have seen these words or concepts before but it's not necessary to complete this task.

For a 2×2 matrix (2D array: 2 rows, 2 columns) A defined as

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

the *determinant* of the matrix A can be calculated by

$$\det(A) = ad - bc$$

If the determinant is zero, the matrix is said to be *non-invertible*, i.e. its *inverse* does not exist. If the determinant is non-zero, the inverse of the matrix (A^{-1}) can be found with the formula

$$A^{-1} = \frac{1}{\det(A)} \times \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

1. Write a C program that will calculate the determinant of a 2×2 matrix.

2. Expand on the previous program by calculating the inverse of the 2×2 matrix (only if it exists). If it does not exist, print a warning message to the user.