Two Dimensional Array

Declaration:

```
const int MAX_ROW=4;
const int MAX_COL=4;

int  matrix[MAX_ROW][MAX_COL];   // declares a 4x4 two dimensional array ("matrix")
```

Accessing individual elements of the two dimensional array

Row subscript starts with 0, ends with 4
Column subscript starts with 0, ends with 3

```
matrix[2][3] = 23;
matrix[0][0] = matrix[0][1]+matrix[1][0];
```

Initialization

- initialize all elements of the matrix to 0 during declaration

```
int matrix[MAX_ROW][MAX_COL] = {0};
```

- general initialization during declaration

```
int matrix[MAX_ROW][MAX_COL] ={{3, 4, 5, 6},
                               {2, 30, 2, 29},
                               {2, 1, 4, 2},
                               {45, 98, 0, 21}};
```

- **Input values**

```
int row, col;
for (row=0; row<MAX_ROW; row++)
    for (col=0; col<MAX_COL; col++)
        cin >> table[row][col];
```

- **Output values**

```
int row, col;
for (row=0; row<MAX_ROW; row++)
{
    for (col=0; col<MAX_COL; col++)
        cout << setw(5) <<  table[row][col];

    cout << endl;
}
```

Example 1:

```
#include <iostream>
using namespace std;
const int MAX_ROW=6;
const int MAX_COL = 6;

void DisplayTable(int table[][MAX_COL], int numOfRows, int numOfCols);

int main ()
{
        // local Declarations
        int table [MAX_ROW][MAX_COL];
        int row;
        int column;

        // Assign the values of the table
        // The outer for loop loops through the number of rows in the table
        for (row = 0; row < MAX_ROW; row++)
        {
           // The inner for loop loops through the elements/columns of the ith row
           for (column = 0; column < MAX_COL; column++)
           {
               if (row == column)
                    table [row][column] = 0;
               else if (row > column)
                    table [row][column] = -1;
               else
                    table [row][column] = 1;
            }
        }

        // call user defined function to display the content of the table
        DisplayTable(table, MAX_ROW, MAX_COL);

        return 0;
}       // main

// This function displays the content of the table, one row per line
void DisplayTable(int table[][MAX_COL], int numOfRows, int numOfCols)
{
        for (row = 0; row <numOfRows; row++) {
            for (column = 0; column < numOfCols; column++)
                  cout <<  table[row][column] << "  ";
            cout << endl;
        }

        return;
}
```

Practice Problem:

Example 1:
Matrices are essential in Computer Graphics. Some of the basic operations in CG requires the following operations:

1. Declare and initialize an identity matrix, i.e., matrix with 1s on the diagonal, and 0s everywhere else

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2. How to multiply a 4 by 4 matrix to a point defined by a vector of 4

$$\begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 1*1+0*2+0*5+3*1 \\ 0*1+1*2+0*5+4*1 \\ 0*1+0*2+1*5+2*1 \\ 0*1+0*2+0*5+1*1 \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \\ 7 \\ 1 \end{bmatrix}$$

3. How to multiple a 4 by 4 matrix to another 4 by 4 matrix ?

$$\begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 8 & 4 & 0 & 0 \\ 3 & 5 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 1*8+0*3+0*0+3*0 & 1*4+0*5+0*0+3*0 & 1*0+0*0+0*3+3*0 & 1*0+0*0+0*0+3*1 \\ 0*8+1*3+0*0+4*0 & 0*4+1*5+0*0+4*0 & 0*0+1*0+0*3+4*0 & 0*0+1*0+0*0+4*1 \\ 0*8+0*3+1*0+2*0 & 0*4+0*5+1*0+2*0 & 0*0+0*0+1*3+2*0 & 0*0+0*0+1*0+2*1 \\ 0*8+0*3+0+0 & 0*4+0*5+0+1*0 & 0+0+0*3+1*0 & 0+0+0+1 \end{bmatrix}$$

$$= \begin{bmatrix} 8 & 4 & 0 & 3 \\ 3 & 5 & 0 & 4 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Example 2:
A logging operation keeps records of 37 loggers' monthly production for purposes of analysis, using the following array structure:

```
const int NUM_LOGGERS = 37;
int logsCut[NUM_LOGGERS][12];
imt monthlyHigh;
int monthlyTotal;
int yearlyTotal;
int high;
int month;
int bestMonth;
int logger;
int bestLogger;
```

1. The following statement assigns the January log total for logger number 7 to monthlyTotal [True/False]?

```
monthlyTotal = logsCut[7][0];
```

2. The following statements compute the yearly total for logger number 11 [True/False]?

```
yearlyTotal = 0;
for (month = 0; month < NUM_LOGGERS; month++)    {
        yearlyTotal = yearlyTotal + logsCut[month][10];
}
```

3. The following statements find the best logger (most logs cut) in March.[True/False]?

```
monthlyHigh = 0;
for (logger=0; logger < NUM_LOGGERS; logger++) {
        if (logsCut[logger][2] > monthlyHigh)    {
                bestLogger = logger;
                monthlyHigh  = logsCut[logger][2];
        }
}
```

4. The following statements find the logger with the highest monthly production and the logger's best month [True/False]?

```
high = -1;
for (month = 0; month < 12; month++)  {
        for (logger = 0; logger < NUM_LOGGERS; logger++)  {
                if (logsCut[logger][month] > high)  {
                        high = logsCut[logger][month];
                        bestLogger = logger;
                        bestMonth = month;
                }
        }
}
```