Data Structures – Lesson 2

B.Sc. 3rd Semester, Paper C5

Paulami Basu Ray
Assistant Professor
Department of Computer Science & Applications
Prabhat Kumar College, Contai

Multidimensional Array

- We can define multidimensional arrays in simple words as array of arrays. Data in multidimensional arrays are stored in tabular form (in row major order).
- Examples:

2D array: int twoD[10][20];

3D array: int threeD[10][20][30];

Size of multidimensional arrays

Total number of elements that can be stored in a multidimensional array can be calculated by multiplying the size of all the dimensions.
For example:
The array **int x[10][20]** can store total (10*20) = 200 elements.
Similarly array **int x[5][10][20]** can store total (5*10*20) = 1000 elements.

Two-Dimensional Array:

It is the simplest form of a multidimensional array.

Syntax:

data_type array_name[x][y];

data_type=type of data to be stored

Two-Dimensional Array

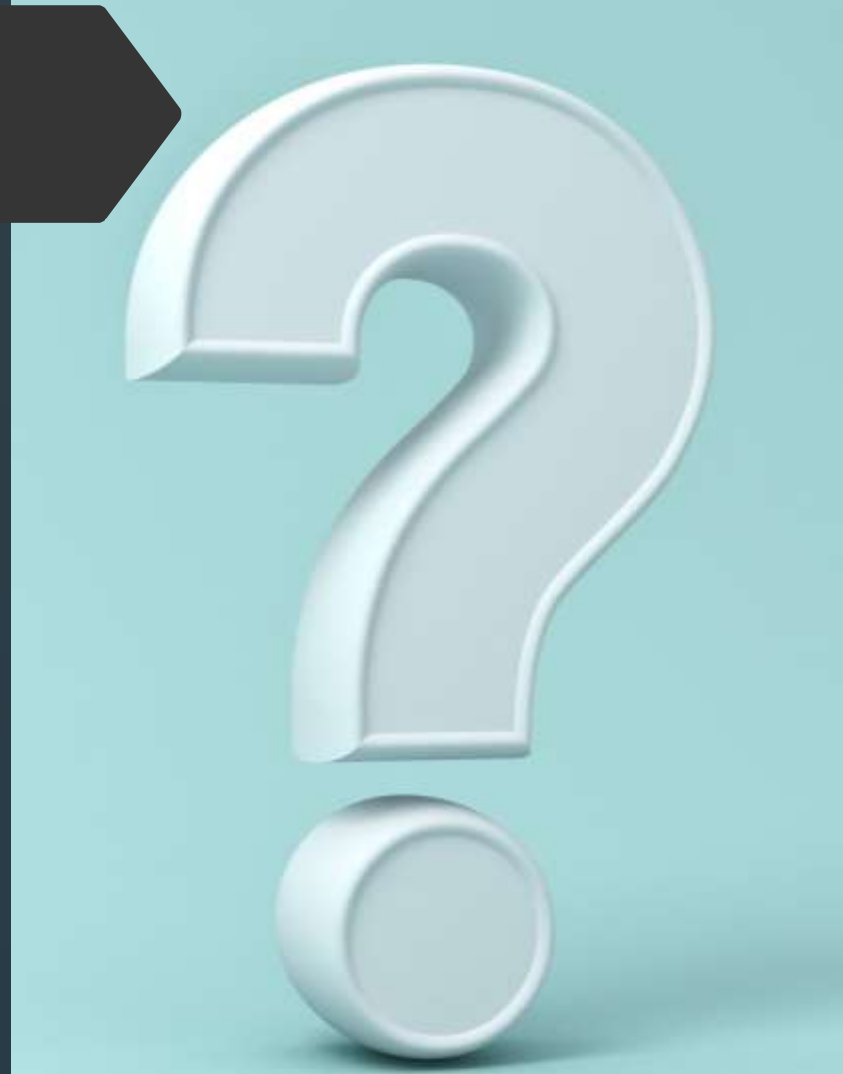
	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

C++ Program to print the elements of a 2D array

```
#include<iostream>
using namespace std;

int main()
{
   // an array with 3 rows and 2 columns.
   int x[3][2] = {{0,1}, {2,3}, {4,5}};

   // output each array element's value
   for (int i = 0; i < 3; i++)
   {
      for (int j = 0; j < 2; j++)
      {
         cout << "Element at x[" << i
              << "][" << j << "]: ";
         cout << x[i][j]<<endl;
      }
   }

   return 0;
}
```

Output

Element at x[0][0]: 0
Element at x[0][1]: 1
Element at x[1][0]: 2
Element at x[1][1]: 3
Element at x[2][0]: 4
Element at x[2][1]: 5



Sparse Matrix

- A matrix is a two-dimensional data object made of m rows and n columns, therefore having total m x n values. If most (atleast 2/3rd)of the elements of the matrix have **0 value**, then it is called a sparse matrix.
- **Why to use Sparse Matrix instead of simple matrix ?**
- **Storage:** There are lesser non-zero elements than zeros and thus lesser memory can be used to store only those elements.
- **Computing time:** Computing time can be saved by logically designing a data structure traversing only non-zero elements.

Representation of Sparse Matrix

- 2D array is used to represent a sparse matrix in which there are three rows named as
- **Row:** Index of row, where non-zero element is located
- **Column:** Index of column, where non-zero element is located
- **Value:** Value of the non zero element located at index – (row,column)

Representation of Sparse Matrix

$$\begin{bmatrix} 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 5 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 6 & 0 & 0 \end{bmatrix}$$

Row	0	0	1	1	3	3
Column	2	4	2	3	1	2
Value	3	4	5	7	2	6

C++ Program for sparse matrix representation

```
#include<stdio.h>

int main()
{
    // Assume 4x5 sparse matrix
    int sparseMatrix[4][5] =
    {
        {0 , 0 , 3 , 0 , 4 },
        {0 , 0 , 5 , 7 , 0 },
        {0 , 0 , 0 , 0 , 0 },
        {0 , 2 , 6 , 0 , 0 }
    };

    int size = 0;
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 5; j++)
            if (sparseMatrix[i][j] != 0)
                size++;

    // number of columns in compactMatrix (size) must be
    // equal to number of non - zero elements in
    // sparseMatrix
    int compactMatrix[3][size];
            }
```

```
// Making of new matrix

    int k = 0;
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 5; j++)
            if (sparseMatrix[i][j] != 0)
            {
                compactMatrix[0][k] = i;
                compactMatrix[1][k] = j;
                compactMatrix[2][k] = sparseMatrix[i][j];
                k++;
            }

  for (int i=0; i<3; i++)
    {
        for (int j=0; j<size; j++)
            printf("%d ", compactMatrix[i][j]);

        printf("\n");
    }
    return 0;
}
```

0 0 1 1 3 3

2 4 2 3 1 2

3 4 5 7 2 6

Output