2D Array

2D array can be defined as an array of arrays. The 2D array is organized as matrices which can be represented as the collection of rows and columns.

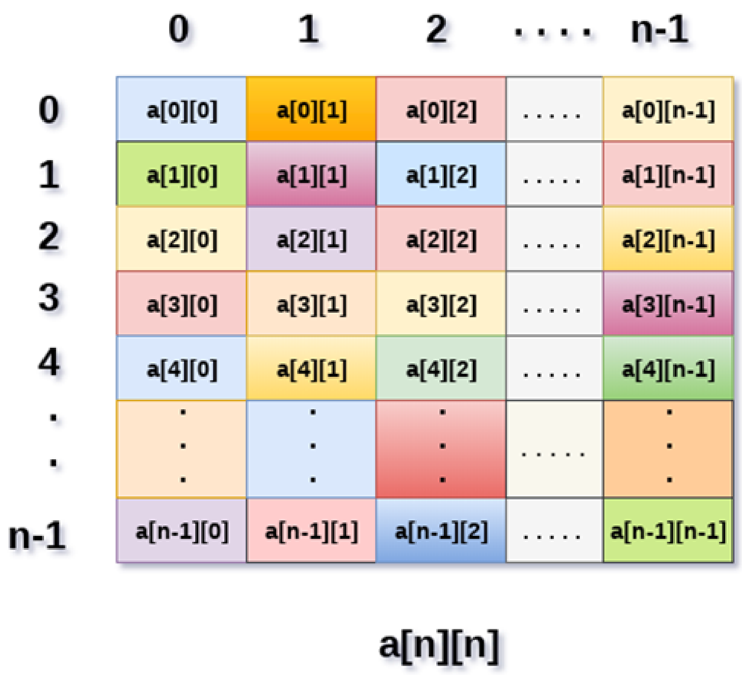
However, 2D arrays are created to implement a relational database look alike data structure. It provides ease of holding bulk of data at once which can be passed to any number of functions wherever required.

How to declare 2D Array

The syntax of declaring two dimensional array is very much similar to that of a one dimensional array, given as follows.

```
int arr[max_rows][max_columns];
```

however, It produces the data structure which looks like following.


0 1 2 .... n-1
0 a[0][0] a[0][1] a[0][2] ..... a[0][n-1]
1 a[1][0] a[1][1] a[1][2] ..... a[1][n-1]
2 a[2][0] a[2][1] a[2][2] ..... a[2][n-1]
3 a[3][0] a[3][1] a[3][2] ..... a[3][n-1]
4 a[4][0] a[4][1] a[4][2] ..... a[4][n-1]
n-1 a[n-1][0] a[n-1][1] a[n-1][2] ..... a[n-1][n-1]
a[n][n]

Above image shows the two dimensional array, the elements are organized in the form of rows and columns. First element of the first row is represented by a[0][0] where the number shown in the first index is the number of that row while the number shown in the second index is the number of the column.

How do we access data in a 2D array

Due to the fact that the elements of 2D arrays can be random accessed. Similar to one dimensional arrays, we can access the individual cells in a 2D array by using the indices of the cells. There are two indices attached to a particular cell, one is its row number while the other is its column number.

However, we can store the value stored in any particular cell of a 2D array to some variable x by using the following syntax.

```
int x = a[i][j];
```

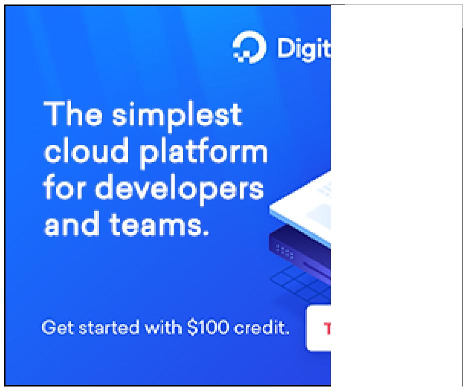
where i and j is the row and column number of the cell respectively.

We can assign each cell of a 2D array to 0 by using the following code:

```
for ( int i=0; i<n ;i++)
{
    for (int j=0; j<n; j++)
    {
        a[i][j] = 0;
    }
}
```

Initializing 2D Arrays

We know that, when we declare and initialize one dimensional array in C programming simultaneously, we don't need to specify the size of the array. However this will not work with 2D arrays. We will have to define at least the second dimension of the array.


Digit
The simplest cloud platform for developers and teams.
Get started with $100 credit.

The syntax to declare and initialize the 2D array is given as follows.

```
int arr[2][2] = {0,1,2,3};
```

The number of elements that can be present in a 2D array will always be equal to (**number of rows * number of columns**).

Example : Storing User's data into a 2D array and printing it.

C Example :

```
#include <stdio.h>
void main ()
{
    int arr[3][3],i,j;
    for (i=0;i<3;i++)
    {
        for (j=0;j<3;j++)
        {
            printf("Enter a[%d][%d]: ",i,j);
            scanf("%d",&arr[i][j]);
        }
    }
    printf("\n printing the elements ....\n");
    for(i=0;i<3;i++)
    {
        printf("\n");
        for (j=0;j<3;j++)
        {
            printf("%d\t",arr[i][j]);
        }
    }
}
```

Java Example

```
import java.util.Scanner;
publicclass TwoDArray {
publicstaticvoid main(String[] args) {
    int[][] arr = newint[3][3];
    Scanner sc = new Scanner(System.in);
    for (inti =0;i<3;i++)
    {
        for(intj=0;j<3;j++)
        {
            System.out.print("Enter Element");
            arr[i][j]=sc.nextInt();
            System.out.println();
        }
    }
    System.out.println("Printing Elements...");
    for(inti=0;i<3;i++)
    {
        System.out.println();
        for(intj=0;j<3;j++)
        {
            System.out.print(arr[i][j]+"\t");
        }
    }
}
}
```

C# Example

```
using System;
```

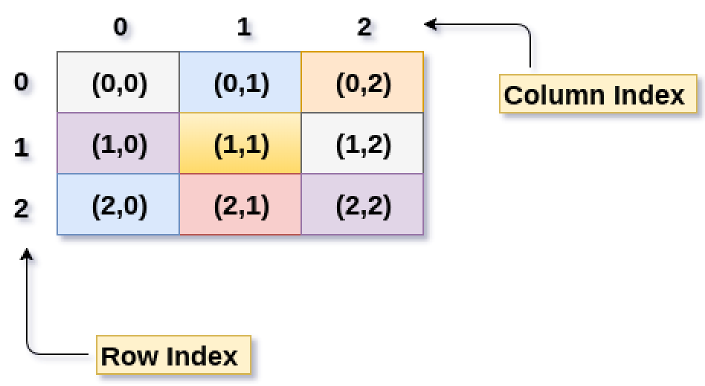
```
public class Program
{
    public static void Main()
    {
        int[,] arr = new int[3,3];
        for (int i=0;i<3;i++)
        {
            for (int j=0;j<3;j++)
            {
                Console.WriteLine("Enter Element");
                arr[i,j]= Convert.ToInt32(Console.ReadLine());
            }
        }
        Console.WriteLine("Printing Elements...");
        for (int i=0;i<3;i++)
        {
            Console.WriteLine();
            for (int j=0;j<3;j++)
            {
                Console.Write(arr[i,j]+" ");
            }
        }
    }
}
```

Mapping 2D array to 1D array

When it comes to map a 2 dimensional array, most of us might think that why this mapping is required. However, 2 D arrays exists from the user point of view. 2D arrays are created to implement a relational database table lookalike data structure, in computer memory, the storage technique for 2D array is similar to that of an one dimensional array.

The size of a two dimensional array is equal to the multiplication of number of rows and the number of columns present in the array. We do need to map two dimensional array to the one dimensional array in order to store them in the memory.

A 3 X 3 two dimensional array is shown in the following image. However, this array needs to be mapped to a one dimensional array in order to store it into the memory.



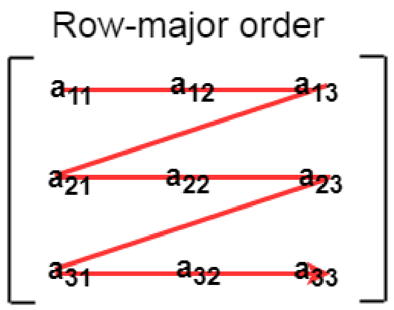
There are two main techniques of storing 2D array elements into memory

1. Row Major ordering

In row major ordering, all the rows of the 2D array are stored into the memory contiguously. Considering the array shown in the above image, its memory allocation according to row major order is shown as follows.

(0,0)	**(0,1)**	**(0,2)**	**(1,0)**	**(1,1)**	**(1,2)**	**(2,0)**	**(2,1)**	**(2,2)**

first, the 1st row of the array is stored into the memory completely, then the 2nd row of the array is stored into the memory completely and so on till the last row.

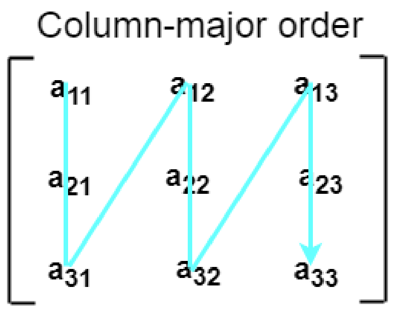


2. Column Major ordering

According to the column major ordering, all the columns of the 2D array are stored into the memory contiguously. The memory allocation of the array which is shown in in the above image is given as follows.

(0,0)	**(1,0)**	**(2,0)**	**(0,1)**	**(1,1)**	**(2,1)**	**(0,2)**	**(1,2)**	**(2,2)**

first, the 1st column of the array is stored into the memory completely, then the 2nd row of the array is stored into the memory completely and so on till the last column of the array.



Calculating the Address of the random element of a 2D array

Due to the fact that, there are two different techniques of storing the two dimensional array into the memory, there are two different formulas to calculate the address of a random element of the 2D array.

By Row Major Order

If array is declared by a[m][n] where m is the number of rows while n is the number of columns, then address of an element a[i][j] of the array stored in row major order is calculated as,

```
Address(a[i][j]) = B. A. + (i * n + j) * size
```

where, B. A. is the base address or the address of the first element of the array a[0][0] .

Example :

```
a[10...30, 55...75], base address of the array (BA) = 0, size of an element = 4 bytes .
Find the location of a[15][68].

Address(a[15][68]) = 0 +
((15 ? 10) x (68 ? 55 + 1) + (68 ? 55)) x 4

= (5 x 14 + 13) x 4
= 83 x 4
= 332 answer
```

By Column major order

If array is declared by a[m][n] where m is the number of rows while n is the number of columns, then address of an element a[i][j] of the array stored in row major order is calculated as,

```
Address(a[i][j]) = ((j*m)+i)*Size + BA
```

where BA is the base address of the array.

Example:

```
A [-5 ... +20][20 ... 70], BA = 1020, Size of element = 8 bytes. Find the location of a[0]
[30].

Address [A[0][30]) = ((30-20) x 24 + 5)  x 8 + 1020   =  245 x 8 + 1020 = 2980 bytes
```

← prev next →

Please Share

Learn Latest Tutorials

Notifications Powered By Feedify

AWS | D. Math. | React Native | TypeScript

COA | Tkinter

B.Tech / MCA

DBMS | DS | DAA | OS

C. Network | Compiler D. | COA | Web Tech.

Cyber Sec. | C | C++ | Java

.Net | Python | Programs | Control S.