## Two Dimensional Array

The two-dimensional 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 lookalike data structure. It provides ease of holding the bulk of data at once which can be passed to any number of functions wherever required.

## Declaration –

## data\_type array\_name[rows][columns];

## int twodimen[4][3];

Here, 4 is the number of rows, and 3 is the number of columns.

## Initialization -

In the 1D array, we don't need to specify the size of the array if the declaration and initialization are being done simultaneously. However, this will not work with 2D arrays. We will have to define at least the second dimension of the array. The two-dimensional array can be declared and defined in the following way.

**int** arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};

## Accessing-

#include<stdio.h>

**int** main()

{

**int** i=0,j=0;

**int** arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};

//traversing 2D array

**for**(i=0;i<4;i++){

**for**(j=0;j<3;j++){

   printf("arr[%d] [%d] = %d \n",i,j,arr[i][j]);

 }//end of j

}//end of i

**return** 0;

}

## Storing elements in a matrix and printing it

#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]);

        }

    }

}

## Matrix Addition-

## #include <stdio.h>   int main() {    int m, n, c, d, first[10][10], second[10][10], sum[10][10];      printf("Enter the number of rows and columns of matrix\n");    scanf("%d%d", &m, &n);    printf("Enter the elements of first matrix\n");      for (c = 0; c < m; c++)       for (d = 0; d < n; d++)          scanf("%d", &first[c][d]);      printf("Enter the elements of second matrix\n");      for (c = 0; c < m; c++)       for (d = 0 ; d < n; d++)          scanf("%d", &second[c][d]);        printf("Sum of entered matrices:-\n");        for (c = 0; c < m; c++) {       for (d = 0 ; d < n; d++) {          sum[c][d] = first[c][d] + second[c][d];          printf("%d\t", sum[c][d]);       }       printf("\n");    }      return 0; }

## Matrix Multiplication

#include <stdio.h>  
  
int main()  
{  
  int m, n, p, q, c, d, k, sum = 0;  
  int first[10][10], second[10][10], multiply[10][10];  
   **printf("Enter number of rows and columns of first matrix\n");  
  scanf("%d%d", &m, &n);  
  printf("Enter elements of first matrix\n");  
   for (c = 0; c < m; c++)  
    for (d = 0; d < n; d++)  
      scanf("%d", &first[c][d]);**   **printf("Enter number of rows and columns of second matrix\n");  
  scanf("%d%d", &p, &q);**   if (n != p)  
    printf("The multiplication isn't possible.**\n**");  
  else  
  {  
    **printf("Enter elements of second matrix\n");  
     for (c = 0; c < p; c++)  
      for (d = 0; d < q; d++)  
        scanf("%d", &second[c][d]);  
     for (c = 0; c < m; c++)**

**{  
      for (d = 0; d < q; d++)**

**{**

**multiply[c][d]  = 0;  
        for (k = 0; k < p; k++) {  
         multiply[c][d]  = multiply[c][d]  + first[c][k]\*second[k][d];  
        }       
  
      }  
    }**  
     printf("Product of the matrices:**\n**");  
   
    for (c = 0; c < m; c++) {  
      for (d = 0; d < q; d++)  
        printf("%d**\t**", multiply[c][d]);  
   
      printf("**\n**");  
    }  
  }  
   return 0;  
}