Two Dimensional Array

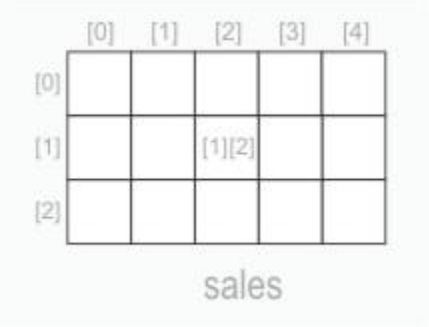
It is a collection of data elements of same data type arranged in rows and columns (that is, in two dimensions).

Declaration of Two-Dimensional Array

Type arrayName[numberOfRows][numberOfColumn];

For example,

```
int Sales[3][5];
```



Initialization of Two-Dimensional Array

An two-dimensional array can be initialized along with declaration. For two-dimensional array initialization, elements of each row are enclosed within curly braces and separated by commas. All rows are enclosed within curly braces.

```
int A[4][3] = {{22, 23, 10},
               {15, 25, 13},
               {20, 74, 67},
               {11, 18, 14}};
```

Referring to Array Elements

To access the elements of a two-dimensional array, we need a pair of indices: one forthe row position and one for the column position. The format is as simple as:
name[rowIndex][columnIndex].

Examples:

```
cout << A[1][2];       //print an array element
A[1][2] = 13;          // assign value to an array element
cin >> A[1][2];        //input element
```

Using Loop to input an Two-Dimensional Array from user

```
int mat[3][5], row, col ;
for (row = 0; row < 3; row++)
  for (col = 0; col < 5; col++)
    cin >> mat[row][col];
```

Arrays as Parameters

Two-dimensional arrays can be passed as parameters to a function, and they are passed by reference. When declaring a two-dimensional array as a formal parameter, we can omit the size of the first dimension, but not the second; that is, we must specify the number of columns. For example:

```
   void print(int A[][3], int N, int M)
```

In order to pass to this function an array declared as:

```
   int arr[4][3];
```

we need to write a call like this:

```
   print(arr);
```

Here is a complete example:

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

void print(int A[][3], int N, int M)
{
  for (R = 0; R < N; R++)
    for (C = 0; C < M; C++)
       cout << A[R][C];
}

int main ()
{
  int arr[4][3] ={{12, 29, 11},
                  {25, 25, 13},
                  {24, 64, 67},
                  {11, 18, 14}};
  print(arr,4,3);
  return 0;
}
```

Function to read the array A

```
void Read(int A[][20], int N, int M)
{
  for(int R = 0; R < N; R++)
    for(int C = 0; C < M; C++)
    {
      cout << "(R<<','<<")?";
      cin >> A[R][C];
     }
}
```

Function to display content of a two dimensional array A

```
void Display(int A[][20], int N, int M)
{
  for(int R = 0;R < N; R++)
  {
     for(int C = 0; C < M; C++)
        cout << setw(10) << A[R][C];
     cout << endl;
   }
}
```

Function to find the sum of two dimensional arrays A and B

```
void Addition(int A[][20], int B[][20], int N, int M)
{
  for(int R = 0; R < N; R++)
    for(int C = 0;C < M; C++)
      C[R][C] = A[R][C] + B[R][C];
}
```

Function to multiply two dimensional arrays A and B of order NxL and LxM

```
void Multiply(int A[][20], int B[][20], int C[][20],int N, int L,
int M)
{
  for(int R = 0; R < N; R++)
   for(int C = 0; C < M; C++)
   {
      C[R][C] = 0;
      for(int T = 0; T < L; T++)
        C[R][C] += A[R][T] * B[T][C];
    }
}
```

Function to find & display sum of rows & sum of cols. of array A

```
void SumRowCol(int A[][20], int N, int M)
{
  for(int R = 0; R < N; R++)
  {
     int SumR = 0;
     for(int C = 0; C < M; C++)
       SumR += A[R][C];
     cout << "Row("<<R<<")=" << SumR << endl;
   }
  for(int R = 0;R < N; R++)
  {
    int SumR = 0;
    for(int C = 0; C < M; C++)
      SumR += A[R][C];
    cout << "Row("<<R<<")=" << SumR << endl;
   }
}
```

Function to find sum of diagonal elements of a square matrix A

```
void Diagonal(int A[][20], int N, int &Rdiag, int &LDiag)
{
  for(int I = 0, Rdiag = 0; I < N; I++)
    Rdiag += A[I][I];
  for(int I = 0, Ldiag = 0; I < N; I++)
    Ldiag += A[N-I-1][I];
}
```

Function to find out transpose of a two dimensional array A

```
void Transpose(int A[][20], int B[][20], int N, int M)
{
  for(int R = 0; R < N; R++)
    for(int C = 0;C < M; C++)
       B[R][C] = A[C][R];
}
```