Two Dimensional Arrays (or Matrices) can be thought of as tables with rows and columns where the intersection between a row and a column represents an element.

	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]

In C/C++ 2D arrays must have a **constant** size and can be declared with the following:-

Datatype name [NUM_ROWS][NUM_COLUMNS];

```
int         arri    [10][15];
string      arrs    [30][40];
```

2D arrays can also be initialized with **{} syntax**, for example:-

```
int matrix[10][30] = {
      {1, 2, 3, 4, 5},
      {6, 7, 8, 9, 10},
      {11, 12, 13, 14, 15}};
```

2D arrays can be addressed with **name[row][col] syntax :-**

```
x = matrix[2]; // x is a one dimensional array
y = matrix[3][4]; // y is an int variable
```

2D arrays can be iterated over with a 2 nested for loops

To iterate over elements row by row:-

```
for(int i = 0; i < ROWS; i++){
    for(int j = 0; j < COLS; j++){
        matrix[i][j]...
        //rest of code
    }
}
```

To iterate over elements column by column:-

```
for(int j = 0; j < COLS; j++){
    for(int i = 0; i < ROWS; i++){
        matrix[i][j]...
        //... rest of code
    }
}
```

Some examples :-

```
//print matrix row by row
 for(int i = 0; i < ROWS; i++){
    for(int j = 0; j < COLS; j++)
        cout << setw(10) << matrix[i][j];
    cout << endl;
 }
```

Find the largest element in each column:-

```
    int results[COLS];
    for (int j = 0; j < COLS; j++){
        results[j] = matrix[0][j];
        //initially first element
        for(int i = 1; i < ROWS; i++)
            results[j] = max(results[j], matrix[i][j]
);
    }
```

Initialize matrix with random values between 50 and 20:-

```
    srand(time(0));
    for(int i = 0; i < ROWS; i++){
        for(int j = 0; j < COLS; j++){
            matrix[i][j] = rand()%( 50-20 + 1) + 20;
        }
    }
```

Sum all elements in each row:-

```
    int sums[ROWS];
    for(int i = 0; i < ROWS; i++){
        sums[i] = 0;
        for(int j = 0; j < COLS; j++){
            sums[i] += matrix[i][j];
        }
    }
```

Muli Dimensional Arrays are an extension of the 2D array to that allow arrays to have 3, 4, 5, … dimensions.

It's the same basic concept as 2D arrays but with more dimensions.

The size of each dimension should be **constant**.

The syntax is:-

datatype name [DIM1] [DIM2] [DIM3] [DIM4] …

to iterate over a multi dimensional array one needs a for loop for each new dimension to cover the entire array.

For example:-

```
int arr3D [DIM1][DIM2][DIM3];

for(int i = 0; i < DIM1; i++){
    for(int j = 0; j < DIM2; j++){
        for(int k = 0; k < DIM3; k++){
            arr3D[i][j][k]...
            //rest of code
        }
    }
}
```

Passing multidimensional array to a function follows the same rules as passing arrays to functions with few extra rules.

-multi dimensional arrays are **passed by reference**, meaning any changing to it in the array in the function **reflects on the original**.

-one can not return a multidimensional array from a function.

- each array dimension **beyond the first dimension** has to be provided to the function.

- one can use the **const** keyword to forbid changes to an array passed to the function.

for example:-

```
//illegal: can't return an array from function
int[50][30][20] func1(int arr[50][30][20]);

//illegal: third dimension is not provided
void func2(float arr[][20][]);

//legal as each dimension beyond the first is
//provided to the function
void func3(string arr[][20][30]);
```