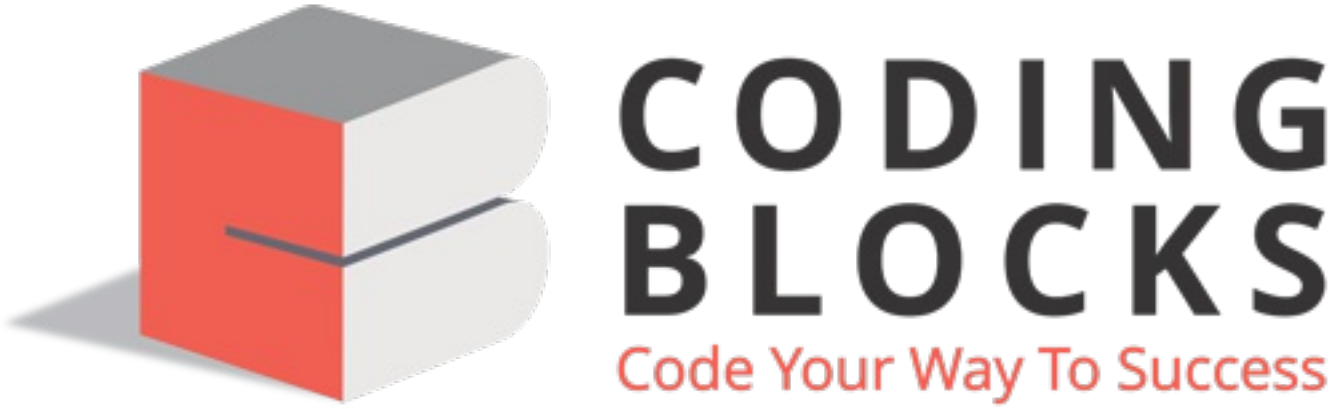

Multidimensional Arrays

Multidimensional arrays means array of arrays. Data in multidimensional arrays are stored in tabular form (in row major order).

Declaration

```
data_type array_name[size1][size2]....[sizeN];
```

data_type: Type of data to be stored in the array.

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.

Two – dimensional Array

- an array of one – dimensional array
- **Declaration** data_type array_name[x][y];
- A two – dimensional array can be seen as a table with ‘x’ rows and ‘y’ columns where the row number ranges from 0 to (x-1) and

column number ranges from 0 to (y-1).

Initializing Two – Dimensional Arrays

There are two ways:-

- `int x[3][4] = {0, 1 ,2 ,3 ,4 , 5 , 6 , 7 , 8 , 9 , 10 , 11};`
  The elements will be filled in the array in the order, first 4 elements from the left in first row, next 4 elements in second row and so on.
- `int x[3][4] = {{0,1,2,3}, {4,5,6,7}, {8,9,10,11}};`
  This type of initialization make use of nested braces. Each set of inner braces represents one row. This is the better method.

Accessing Elements of Two-Dimensional Arrays: Elements in Two-Dimensional arrays are accessed using the row indexes and column indexes.

```
int x[2][1]; //element present in third row and second column
.
```

Example Program: Print all the elements of a Two-Dimensional array.

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

int main()
```

```
{
    // an array with 3 rows and 2 columns.
    int x[3][2] = {{1,2}, {3,4}, {5,6}};

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

    return 0;
}
```

OUTPUT

```
1 2
3 4
5 6
```

Three-Dimensional Array

- First Method `int x[2][3][4] = {0, 1, 2, 3, 4, 5, 6, 7, 8,`

`9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23};`

- Better Method `int x[2][3][4] = { { {0,1,2,3}, {4,5,6,7}, {8,9,10,11} }, { {12,13,14,15}, {16,17,18,19}, {20,21,22,23} } };`

Accessing elements in Three-Dimensional Arrays: The difference is we have to use three loops instead of two loops for one additional dimension in Three-dimensional Arrays.

Example Program: Print all the elements of a Three-Dimensional array.

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

int main()
{
    // initializing the 3-dimensional array
    int x[2][3][2] =
    {
        { {0,1}, {2,3}, {4,5} },
        { {6,7}, {8,9}, {10,11} }
    };

    // output each element's value
```

```
    for (int i = 0; i < 2; ++i)
    {
        for (int j = 0; j < 3; ++j)
        {
            for (int k = 0; k < 2; ++k)
            {
                cout << "Element at x[" << i << "][" << j
                     << "][" << k << "] = " << x[i][j][k]
                     << endl;
            }
        }
    }
    return 0;
}
```

OUTPUT

```
Element at x[0][0][0] = 0
Element at x[0][0][1] = 1
Element at x[0][1][0] = 2
Element at x[0][1][1] = 3
Element at x[0][2][0] = 4
Element at x[0][2][1] = 5
Element at x[1][0][0] = 6
Element at x[1][0][1] = 7
Element at x[1][1][0] = 8
Element at x[1][1][1] = 9
```

```
Element at x[1][2][0] = 10
Element at x[1][2][1] = 11
```

We can create arrays with any number of dimension. However the complexity also increases as the number of dimension increases.