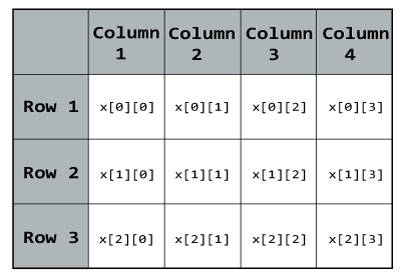
C++ Multidimensional Arrays

In C++, you can create an [array](https://www.programiz.com/cpp-programming/arrays) of an array known as multi-dimensional array. For example:

int x[3][4];

Here, x is a two dimensional array. It can hold a maximum of 12 elements.

You can think this array as table with 3 rows and each row has 4 columns as shown below.



Three dimensional array also works in a similar way. For example:

float x[2][4][3];

This array x can hold a maximum of 24 elements. You can think this example as: Each of the 2 elements can hold 4 elements, which makes 8 elements and each of those 8 elements can hold 3 elements. Hence, total number of elements this array can hold is 24.

## Multidimensional Array Initialisation

Better way to initialise this array with same array elements as above.

int test[2][3] = { {2, 4, 5}, {9, 0 0}};

### Initialisation of three dimensional array

Better way to initialise this array with same elements as above.

int test[2][3][4] = {

{ {3, 4, 2, 3}, {0, -3, 9, 11}, {23, 12, 23, 2} },

{ {13, 4, 56, 3}, {5, 9, 3, 5}, {3, 1, 4, 9} }

};

## Example 1: Two Dimensional Array

**C++ Program to display all elements of an initialised two dimensional array.**

#include <iostream>

using namespace std;

int main()

{

int test[3][2] =

{

{2, -5},

{4, 0},

{9, 1}

};

// Accessing two dimensional array using

// nested for loops

for(int i = 0; i < 3; ++i)

{

for(int j = 0; j < 2; ++j)

{

cout<< "test[" << i << "][" << j << "] = " << test[i][j] << endl;

}

}

return 0;

}

**Output**

test[0][0] = 2

test[0][1] = -5

test[1][0] = 4

test[1][1] = 0

test[2][0] = 9

test[2][1] = 1

### Example: Three Dimensional Array

**C++ Program to Store value entered by user in three dimensional array and display it.**

#include <iostream>

using namespace std;

int main()

{

// This array can store upto 12 elements (2x3x2)

int test[2][3][2];

cout << "Enter 12 values: \n";

// Inserting the values into the test array

// using 3 nested for loops.

for(int i = 0; i < 2; ++i)

{

for (int j = 0; j < 3; ++j)

{

for(int k = 0; k < 2; ++k )

{

cin >> test[i][j][k];

}

}

}

cout<<"\nDisplaying Value stored:"<<endl;

// Displaying the values with proper index.

for(int i = 0; i < 2; ++i)

{

for (int j = 0; j < 3; ++j)

{

for(int k = 0; k < 2; ++k)

{

cout << "test[" << i << "][" << j << "][" << k << "] = " << test[i][j][k] << endl;

}

}

}

return 0;

}

**Output**

Enter 12 values:

1

2

3

4

5

6

7

8

9

10

11

12

Displaying Value stored:

test[0][0][0] = 1

test[0][0][1] = 2

test[0][1][0] = 3

test[0][1][1] = 4

test[0][2][0] = 5

test[0][2][1] = 6

test[1][0][0] = 7

test[1][0][1] = 8

test[1][1][0] = 9

test[1][1][1] = 10

test[1][2][0] = 11

test[1][2][1] = 12

As the number of dimension increases, the complexity also increases tremendously although the concept is quite similar.