

C++ Multi-Dimensional Arrays

Multi-Dimensional Arrays

A multi-dimensional array is an array of arrays.

To declare a multi-dimensional array, define the variable type, specify the name of the array followed by square brackets which specify how many elements the main array has, followed by another set of square brackets which indicates how many elements the sub-arrays have:

```
string letters[2][4];
```

As with ordinary arrays, you can insert values with an array literal - a comma-separated list inside curly braces.

In a multi-dimensional array, each element in an array literal is another array literal.

```
string letters[2][4] = {  
    { "A", "B", "C", "D" },
```

```
    { "E", "F", "G", "H" }  
};
```

Each set of square brackets in an array declaration adds another **dimension** to an array. An array like the one above is said to have two dimensions.

Arrays can have any number of dimensions. The more dimensions an array has, the more complex the code becomes. The following array has three dimensions:

```
string letters[2][2][2] = {  
    {  
        { "A", "B" },  
        { "C", "D" }  
    },  
    {  
        { "E", "F" },  
        { "G", "H" }  
    }  
};
```

Access the Elements of a Multi-Dimensional Array

To access an element of a multi-dimensional array, specify an index number in each of the array's dimensions.

This statement accesses the value of the element in the **first row (0)** and **third column (2)** of the **letters** array.

Example

```
string letters[2][4] = {  
    { "A", "B", "C", "D" },  
    { "E", "F", "G", "H" }  
};
```

```
cout << letters[0][2]; // Outputs "C"
```

Try it Yourself »

Remember that: Array indexes start with 0: [0] is the first element. [1] is the second element, etc.

Change Elements in a Multi-Dimensional Array

To change the value of an element, refer to the index number of the element in each of the dimensions:

Example

```
string letters[2][4] = {  
    { "A", "B", "C", "D" },  
    { "E", "F", "G", "H" }  
};  
letters[0][0] = "Z";
```

```
cout << letters[0][0]; // Now outputs "Z" instead of "A"
```

Try it Yourself »

ADVERTISEMENT

Loop Through a Multi-Dimensional Array

To loop through a multi-dimensional array, you need one loop for each of the array's dimensions.

The following example outputs all elements in the **letters** array:

Example

```
string letters[2][4] = {  
    { "A", "B", "C", "D" },  
    { "E", "F", "G", "H" }  
};
```

```
for (int i = 0; i < 2; i++) {  
    for (int j = 0; j < 4; j++) {  
        cout << letters[i][j] << "\n";  
    }  
}
```

Try it Yourself »

This example shows how to loop through a three-dimensional array:

Example

```
string letters[2][2][2] = {  
    {  
        { "A", "B" },  
        { "C", "D" }  
    },  
    {  
        { "E", "F" },  
        { "G", "H" }  
    }  
}
```

```
    { "G", "H" }  
  }  
};  
  
for (int i = 0; i < 2; i++) {  
  for (int j = 0; j < 2; j++) {  
    for (int k = 0; k < 2; k++) {  
      cout << letters[i][j][k] << "\n";  
    }  
  }  
}
```

Try it Yourself »

Why Multi-Dimensional Arrays?

Multi-dimensional arrays are great at representing grids. This example shows a practical use for them. In the

following example we use a multi-dimensional array to represent a small game of Battleship:

Example

```
// We put "1" to indicate there is a ship.
```

```
bool ships[4][4] = {  
    { 0, 1, 1, 0 },  
    { 0, 0, 0, 0 },  
    { 0, 0, 1, 0 },  
    { 0, 0, 1, 0 }  
};
```

```
// Keep track of how many hits the player has and how many  
turns they have played in these variables
```

```
int hits = 0;
```

```
int numberOfTurns = 0;
```

```
// Allow the player to keep going until they have hit all four  
ships
```

```
while (hits < 4) {
```

```
int row, column;
```

```
cout << "Selecting coordinates\n";
```

```
// Ask the player for a row
```

```
cout << "Choose a row number between 0 and 3: ";
```

```
cin >> row;
```

```
// Ask the player for a column
```

```
cout << "Choose a column number between 0 and 3: ";
```

```
cin >> column;
```

```
// Check if a ship exists in those coordinates
```

```
if (ships[row][column]) {
```

```
    // If the player hit a ship, remove it by setting the value to  
    zero.
```

```
    ships[row][column] = 0;
```

```
// Increase the hit counter
```

```
hits++;
```

```
__ // Tell the player that they have hit a ship and how many  
ships are left
```

```
__ cout << "Hit! " << (4-hits) << " left.\n\n";
```

```
_ } else {
```

```
__ // Tell the player that they missed
```

```
__ cout << "Miss\n\n";
```

```
_ }
```

```
__ // Count how many turns the player has taken
```

```
_ numberOfTurns++;
```

```
}
```

```
cout << "Victory!\n";
```

```
cout << "You won in " << numberOfTurns << " turns";
```