Creating a Two-Dimensional Array

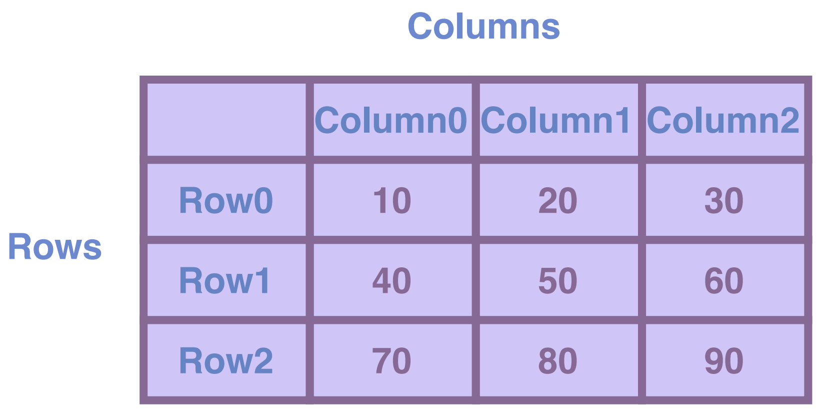
Learn about the implementation of the two-dimensional array.

We'll cover the following

Two-dimensional arrays#

A **two-dimensional array** is an array of arrays.

Two-dimensional arrays represent a matrix. We can access the element in a two-dimensional array by the row and column index. Both the row and column index start at **0**.



Columns
Rows

	Column0	Column1	Column2
Row0	10	20	30
Row1	40	50	60
Row2	70	80	90

2D array

Declaration#

The general syntax for declaring a two-dimensional array is:

DataType ArrayName [RowSize] [ColumnSize];

In the 2D array declaration, we specify the data type of an array followed by an array name, which is further followed by the row index and column index in square brackets.

See the program given below!

```
#include <iostream>

using namespace std;

int main() {

  int Student[10][5];

}
```

We have declared a two-dimensional array `Student[10][5]` that can hold **10** arrays of `Student[5]`. Each `Student[5]` array can store **5** integer values.

The code given above reserves space for **10*5** = **50** elements of type `int` consecutively in memory. Since the element is of type `int`, the compiler reserves **4 bytes** for each element, and in total, it reserves **50*4** = **200 bytes** with the name `Student`.

Array initialization#

We can assign a value to the array elements in a 2D array by accessing its row and column index.

ArrayName [RowIndex] [ColumnIndex] = Value ;

See the code given below!

```cpp
#include <iostream>

using namespace std;

int main() {

  int Student[2][2];

  Student[0][0] = 100;
  Student[0][1] = 134;

  Student[1][0] = 34;
  Student[1][1] = 189;

}
```

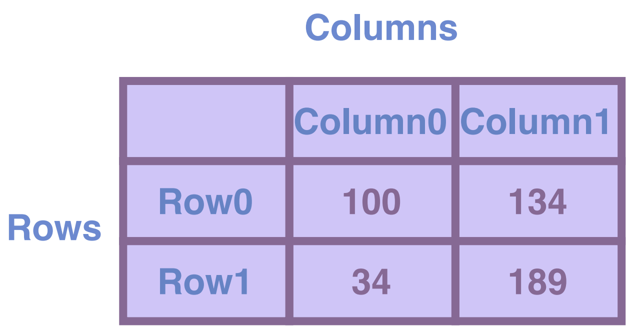
The code above initializes a two-dimensional array that stores:

100 at row index `0` and column index `0`.

134 at row index `0` and column index `1`.

34 at row index `1` and column index `0`.

189 at row index `1` and column index `1`.



		Column0	Column1
Rows	Row0	100	134
	Row1	34	189

2D array

Array initialization in the declaration step#

We can assign values to the 2D array in the declaration step.

DataType ArrayName [ ][ ] = { {value1....,N}.....{value1....,N} } ;

See the code given below!

```
#include <iostream>

using namespace std;

int main() {

  int Student[][3] = {{100, 134, 234}, {34, 189, 221}, {109, 139, 56}};

}
```

ℹ️ If we initialize an array with elements fewer than its total size, it automatically initializes the remaining elements with their default values.

ℹ️ When *initializing* a 2-D array, specifying the first dimension is optional. The compiler will infer the number of rows from the statement. In the above program, changing `Student[3][3]` to `Student[][3]` is fine, but either `Student[][]` or `Student[3][]` isn't valid.

ℹ️ If we aren't initializing a 2-D array, all of its dimensions must be specified.

That is all about creating a two-dimensional array in C++. In the next lesson, we learn how to access and update elements stored in two-dimensional arrays.

← Back
Arrays and Functions

Next →
Accessing Two-Dimensional Arrays

✓ Mark as Completed

Report an Issue