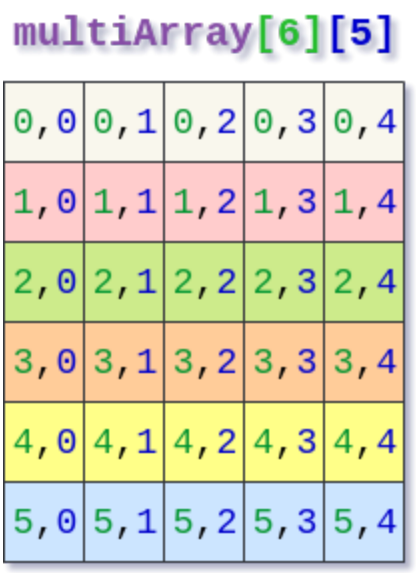
Two-Dimensional Array

A two-dimensional array (2D array) has the same properties as an array except there are rows and columns versus solely one row. This can be thought of as cells in a spreadsheet. With the addition of an extra dimension comes a second square bracket on the end of your variable. For example: `int multiArray[6][5];` The first square bracket is the number of rows and the second square bracket is the number of columns. The picture on the right shows what indexes are used to access that particular element.



There are two ways to initialize a 2D array. One way is via a nested **for** loop, such as:

```
for(int i = 0; i < 6; i++)
    for(int j = 0; j < 5; j++)
        multiArray[i][j] = i * 5 + j;
```

Here, **multiArray** is initialized with cell 0,0 housing the value of 0, cell 0,1 with the value 1, and this continues until cell 5,4 which houses the value of 29. This makes 30 values in total, which is what we want, because 6 rows * 5 columns = 30. If you want to initialize your two-dimensional array at the same time you declare it, you would do this:

```
int multiArray[3][3] = { {1,2,3} , {4,5,6} , {4,7,9} };
```

A two-dimensional array is really an array of arrays. Therefore, you need two outer curly brackets and subarrays inside those curly brackets. The first set of the nested arrays would be row zero and columns zero, one, and two. The second set of nested arrays would be row one and columns zero, one, and two and so forth.

In order to pass a two-dimensional array into a function, an explicit size is needed in the rightmost bracket (the column bracket). If you were to create a function to pass in **multiArray**, it would look like:

```
void function(int multiArray[][5]); //Function definition
```

The function invocation would solely pass in the array variable just like any other argument you've seen so far.

You can treat a 1D array as a 2D array with the following formula:

```
int array[row*column]; //Row * Column for array size
for(int i = 0; i < row; ++i) //Nested for loop to put in values
    for(int j = 0; j < column; ++j) //i * column + j index will give
        cin >> array[i*column+j] //you each element in sequence from
                                 //top-left to bottom-right
```