

Row Major Order

While conceptually the array `a2d` contains rows and columns, physically the elements are stored linearly, with the elements of each row following the elements of the preceding row in memory.

The array `a2d` actually contains **two elements** (not 6!). Each is a one-dimensional `int` array of size `3`. This is how the compiler sees the declaration:

	a2d[0]			a2d[1]		
a2d =	5	19	3	22	-8	10
	[0][0]	[0][1]	[0][2]	[1][0]	[1][1]	[1][2]

That means, instead of using a **2D** array, we **could** store the same elements in a **1D** array, here named `a`, like this:

	[0]	[1]	[2]	[3]	[4]	[5]
a =	5	19	3	22	-8	10

To treat this **1D** array as a **2D** array, (as you've done with all of your image projects in this class), you need to recall the formula for **array access expressions**:

```
a[offset] = *(a + offset)
```

You can convert this to where a **2D array offset expression** like this:

```
a[row, col] = *(a + (row * row - width + col))
```

Notice how **similar** this is to **1D** pointer-address arithmetic; the only new addition is the expression `row * row-width` to the calculation.



This course content is offered under a CC Attribution Non-Commercial license. Content in this course can be considered under this license unless otherwise noted.