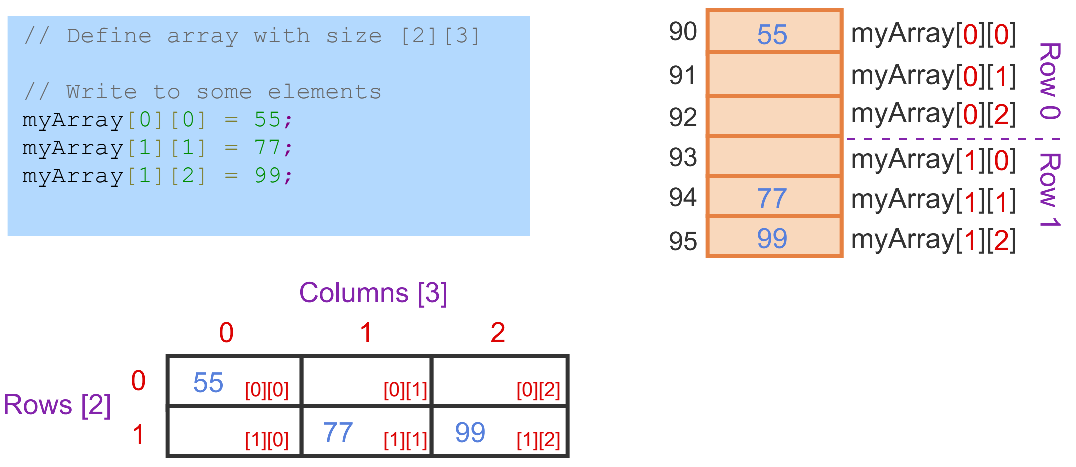
5.12 Two-dimensional arrays

An array can be declared with two dimensions. `int myArray[R][C]` represents a table of int variables with R rows and C columns, so R*C elements total. For example, `int myArray[2][3]` creates a table with 2 rows and 3 columns, for 6 int variables total. Example accesses are `myArray[0][0] = 33;` or `num = myArray[1][2]`.

PARTICIPATION ACTIVITY 5.12.1: Two-dimensional array.

Start 2x speed



```
// Define array with size [2][3]

// Write to some elements
myArray[0][0] = 55;
myArray[1][1] = 77;
myArray[1][2] = 99;
```

Feedback?

Conceptually, a two-dimensional array is a table with rows and columns. The compiler maps two-dimensional array elements to one-dimensional memory, each row following the previous row, known as ***row-major order***.

Figure 5.12.1: Using a two-dimensional array: A driving distance between cities example.

```
#include <iostream>
using namespace std;

/* Direct driving distances between cities, in miles */
/* 0: Boston  1: Chicago  2: Los Angeles */

int main() {
   int cityA;                  // Starting city
   int cityB;                  // Destination city
   int drivingDistances[3][3]; // Driving distances

   // Initialize distances array
   drivingDistances[0][0] = 0;
   drivingDistances[0][1] = 960;  // Boston-Chicago
   drivingDistances[0][2] = 2960; // Boston-Los Angeles
   drivingDistances[1][0] = 960;  // Chicago-Boston
   drivingDistances[1][1] = 0;
   drivingDistances[1][2] = 2011; // Chicago-Los Angeles
   drivingDistances[2][0] = 2960; // Los Angeles-Boston
   drivingDistances[2][1] = 2011; // Los Angeles-Chicago
   drivingDistances[2][2] = 0;

   cout << "0: Boston  1: Chicago  2: Los Angeles" <<
endl;

   cout << "Enter city pair (Ex: 1 2) -- ";
   cin >> cityA;
   cin >> cityB;

   if ((cityA >= 0) && (cityA <= 2) && (cityB >= 0) &&
(cityB <= 2)) {
      cout << "Distance: " << drivingDistances[cityA]
[cityB];
      cout << " miles." << endl;
   }

   return 0;
}
```

```
0: Boston  1: Chicago  2: Los
Angeles
Enter city pair (Ex: 1 2) -- 1
2
Distance: 2011 miles.

...

0: Boston  1: Chicago  2: Los
Angeles
Enter city pair (Ex: 1 2) -- 2
0
Distance: 2960 miles.

...

0: Boston  1: Chicago  2: Los
Angeles
Enter city pair (Ex: 1 2) -- 1
1
Distance: 0 miles.
```

Feedback?

A programmer can initialize a two-dimensional array's elements during declaration using nested braces, as below. Multiple lines make the rows and columns more visible.

Construct 5.12.1: Initializing a two-dimensional array during declaration.

```
// Initializing a 2D array
int numVals[2][3] = { {22, 44, 66}, {97, 98, 99} };

// Use multiple lines to make rows more visible
int numVals[2][3] = {
   {22, 44, 66}, // Row 0
   {97, 98, 99}  // Row 1
};
```

Feedback?

Arrays of three or more dimensions can also be declared, as in `int myArray[2][3][5]`, which declares a total of 2*3*5 or 30 elements. Note the rapid growth in size -- an array declared as `int myArray[100][100][5][3]` would have 100*100*5*3 or 150,000 elements. A programmer should make sure not to unnecessarily occupy available memory with a large array.

PARTICIPATION ACTIVITY 5.12.2: Two-dimensional arrays.

1) Declare a two dimensional array of integers named dataVals with 4 rows and 7 columns.

Check Show answer

2) How many total elements are in an array with 4 rows and 7 columns?

Check Show answer

3) How many elements are in the array declared as: char streetNames[20][50];

Check Show answer

4) Write a statement that assigns 99 into the fifth row, third column of array numVals. Note: the first row/column is at index 0, not 1.

Check Show answer

Feedback?

CHALLENGE ACTIVITY

5.12.1: Find 2D array max and min.

Find the maximum value and minimum value in milesTracker. Assign the maximum value to maxMiles, and the minimum value to minMiles. Sample output for the given program:

```
Min miles: -10
Max miles: 40
```

(Notes)

```
   for (i = 0; i < NUM_ROWS; i++){
      for (j = 0; j < NUM_COLS; j++){
         cin >> value;
         milesTracker[i][j] = value;
      }
   }

   /* Your solution goes here  */
   maxMiles = milesTracker[0][0];
    minMiles = milesTracker[0][0];
   for (i = 0; i < NUM_ROWS; i++){
      for (j = 0; j < NUM_COLS; j++){
         if(milesTracker[i][j]>maxMiles){
            maxMiles = milesTracker[i][j];
         }
         if(milesTracker[i][j]<minMiles){
            minMiles = milesTracker[i][j];
         }
      }
   }

   cout << "Min miles: " << minMiles << endl;
```

Run ✓ All tests passed

✓ Testing with inputs: -10 20 30 40

Your output

```
Min miles: -10
Max miles: 40
```

✓ Testing with inputs: 73 0 50 12

Your output

```
Min miles: 0
Max miles: 73
```

✓ Testing with inputs: -5 -93 -259 -82

```
Min miles: -259
```

Your output

```
Max miles: -5
```

Feedback?