Two-Dimensional Arrays

What is a 2D Array?

A two-dimensional array is:

- essentially a table of data (with rows and columns)
- an array of arrays!
- very common in programming

How to use them:

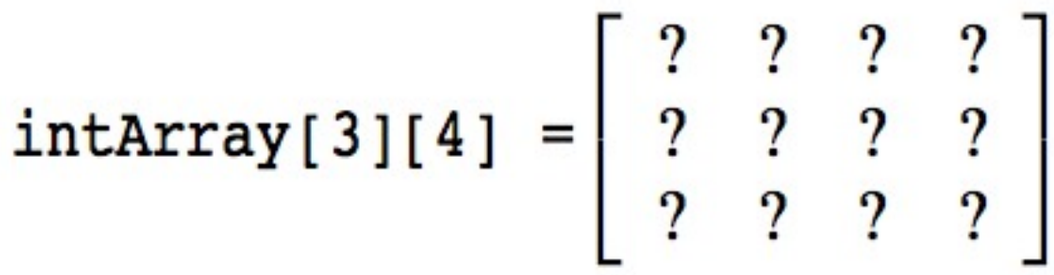
- simply use two pairs of square brackets!

Declaring a 2D Array

Example:

```
// constants specifying the dimensions of the array
const int ROWS = 3, COLS = 4;

// declares an 3x4 array of integers
int intArray[ROWS][COLS];
```



Declaring a 2D Array

Example:

```
// constants specifying the dimensions of the array
const int ROWS = 3, COLS = 4;

// declares an 3x4 array of integers
int intArray[ROWS][COLS] = {};
```

$$\texttt{intArray[3][4]} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Declaring a 2D Array

Example:

```
// declares a 3x4 array of integers w/ initial values
const int ROWS = 3, COLS = 4;
int intArray[ROWS][COLS] = {
    { 1,  2,  3,  4},
    {-2,  1,  2,  3},
    {-3, -2,  1,  2}
};
```

$$\texttt{intArray[3][4]} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ -2 & 1 & 2 & 3 \\ -3 & -2 & 1 & 2 \end{bmatrix}$$

Declaring a 2D Array

Example:

```
// declares a 3x4 array of integers
const int ROWS = 3, COLS = 4;
int intArray[][COLS] = { // implicit 1st dimension OK
    { 1,  2,  3,  4},
    {-2,  1,  2,  3},
    {-3, -2,  1,  2}
};
```

$$\texttt{intArray[3][4]} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ -2 & 1 & 2 & 3 \\ -3 & -2 & 1 & 2 \end{bmatrix}$$

Declaring a 2D Array

Example:

```
// you MUST specify all dimensions but the first!
const int ROWS = 3, COLS = 4;
int intArray[][] = { // will not work
        { 1,  2,  3,  4},
        {-2,  1,  2,  3},
        {-3, -2,  1,  2}
};
```

Only the first dimension of a multidimensional array can be implicitly determined by the compiler!

Declaring a 2D Array

Example:

```
// declare a 5x3 array
// initialize some values
double dblArray[5][3] = {
    { 1, 2, 3 },
    { 1, 2 },
    { 0 },
    { },
    // no values for 5th row
};
```

$$\texttt{dblArray[5][3]} = \begin{bmatrix} 1.0 & 2.0 & 3.0 \\ 1.0 & 2.0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Accessing / Modifying a 2D Array

Use 2 pairs of square brackets!

```
// first column of first row
cout << intArray[0][0]; // 1
// last column of last row
cout << intArray[2][3]; // 2
// changes 3rd column of 3rd row from 1 to 3
intArray[2][2] = intArray[0][2];
```

$$\texttt{intArray[3][4]} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ -2 & 1 & 2 & 3 \\ -3 & -2 & 1 & 2 \end{bmatrix}$$

2D Array Practice

Write a single C++ statement that declares:

- a `3x3` array of integers
- a `2x4` array of doubles where each element contains the value of its column index
- a `3x2` array of doubles where each element contains its row index and the number of rows is implicitly declared
- a `SIZExSIZE` array of doubles where each element is initialized to zero.
- a `4x16` array of integers where the first element in each row is initialized to its row index, and all subsequent elements are initialized to zero
- a `3x32` array of booleans with the number of rows implicitly declared and each element initialized to false
- an array of 7 strings containing the seven days of the week (Sunday-Saturday)

An array of strings as 2D?

A string is actually an array of characters…

```
// so this is a 2D array, too!
const string WEEKDAYS[] = {
    “Sunday”,   “Monday”,  “Tuesday”,  “Wednesday”,
    “Thursday”, “Friday”,  “Saturday”
};

cout << WEEKDAYS[1];    // prints “Monday”
cout << WEEKDAYS[1][0]; // prints ‘M’
cout << WEEKDAYS[1][5]; // prints ‘y’
```

strings

A string is just a one-dimensional array of char data!

We'll talk about array-like properties of strings next Monday…

- I just wanted to take this opportunity to blow your minds.