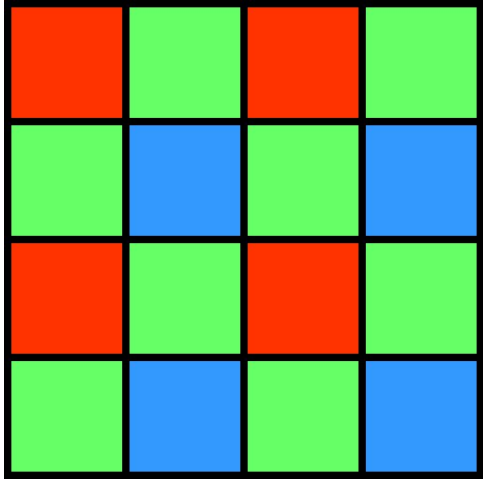
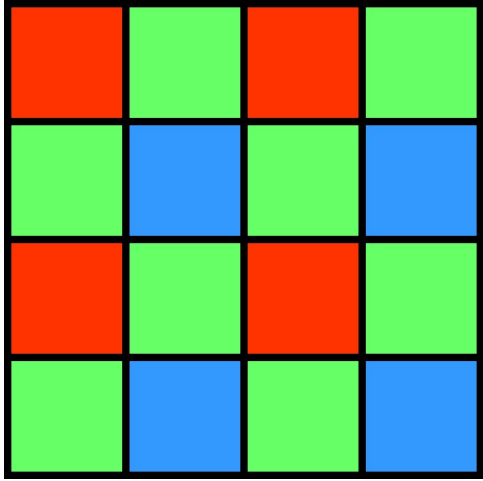
x-D Arrays

Array Review



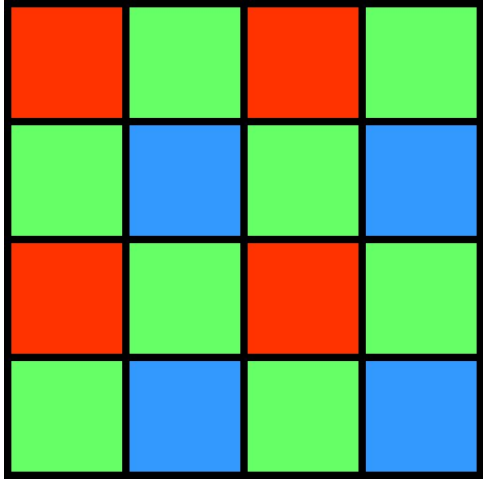
- Arrays are a collection of elements
  - Those elements can be Ints, Chars, Floats, whatever…
  - Each element is an individual instance of that type of variable

Elements of Arrays



- So far, we've only seen *primitive data types* as elements
- *Primitive Data Type:* Only take up one segment of memory at a time
  - Int: One 32-bit segment
  - Char: One 8-bit segment
  - Boolean: One 1-bit segment

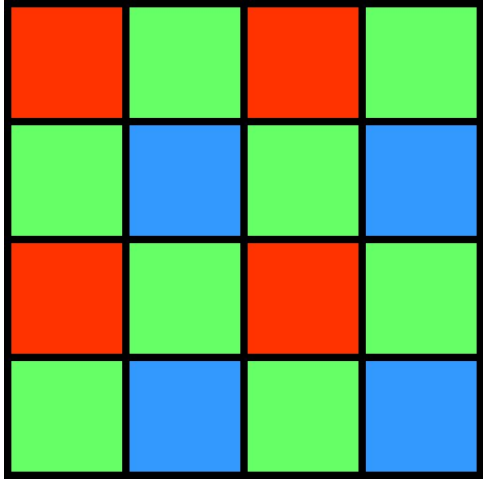
Elements of Arrays



- Arrays build up these primitive data types into *composite data types*

- *Composite data types:* Are built up from primitive data types
  - Ex: Strings are a composite data type of chars

Elements of Arrays



- What if: composite data types can be used as elements within composite data types?

- Ex: An array made up of strings
  - [“Hello”, “world”, “you”, “look”, “good”]

String Arrays

- char sentence = { “Hello”, “world”, “you”, “look”, “good” }

- sentence[0] = “Hello”
- sentence[1] = “world”
- sentence[2] = “you”
- sentence[3] = “look”
- sentence[4] = “good”

String Arrays

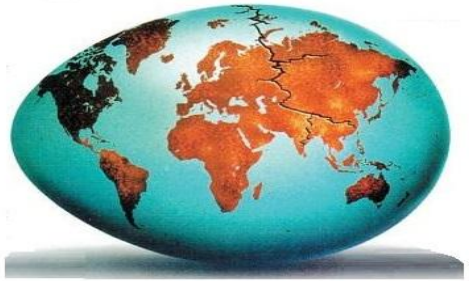
- char sentence = { “Hello”, “world”, “you”, “look”, “good” }

- sentence[0] = [‘H’, ‘e’, ‘l’, ‘l’, ‘o’]
- sentence[1] = [‘w’, ‘o’, ‘r’, ‘l’, ‘d’]
- sentence[2] = [‘y’, ‘o’, ‘u’]
- sentence[3] = [‘l’, ‘o’, ‘o’, ‘k’]
- sentence[4] = [‘g’, ‘o’, ‘o’, ‘d’]

String Arrays

- char sentence = { “Hello”, “world”, “you”, “look”, “good” }
- sentence[o] = [‘H’, ‘e’, ‘l’, ‘l’, ‘o’]

- How can I change the characters in the array??
  - Example: Change “Hello” to “H**a**llo”



String Arrays

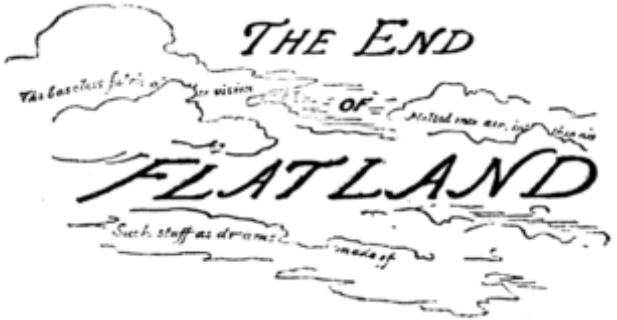
- char sentence = { “Hello”, “world”, “you”, “look”, “good” }
- sentence[0] = [‘H’, ‘e’, ‘l’, ‘l’, ‘o’]

- How can I change the characters in the array??
  - Use **two** indices
  - sentence[0][1] ↠ ‘e’
  - sentence[0][1] = ‘a’ ↠ Will change *e* to *a*

2-D Arrays

- These data types are called 2-D arrays
  - Can be used with ANY kind of primitive data type (int, char, float, bool, etc...)

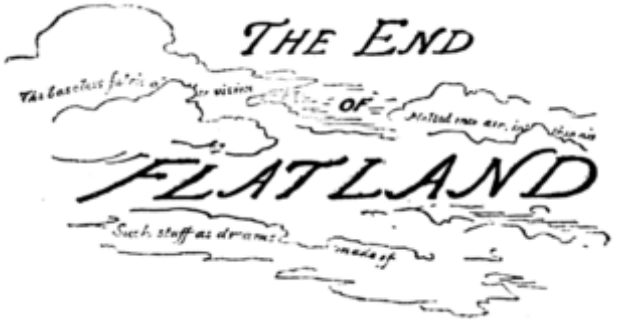
2-D Arrays



- Can be declared using 2D notation
  - **type Test[x][y]**

  - Type ➳ variable within the 2D array (int, char, etc...)
  - x ➳ Number of inner arrays
  - y ➳ Number of elements within the inner array

2-D Arrays



- Can be declared using 2D notation in one of two ways...

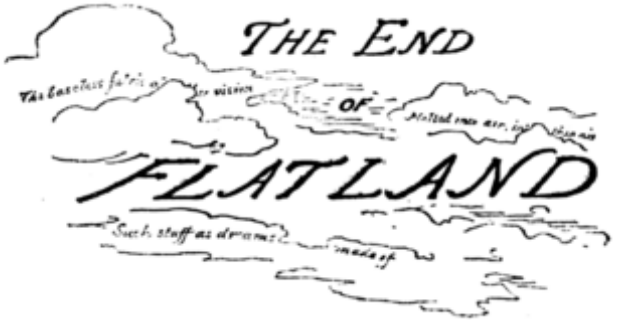
```
//Easier way

int test[3][2] = { {1, 2}, {3, 4}, {5, 6}};

//More confusing way

int test[3][2] = { 1, 2, 3, 4, 5, 6};
```

Printing 2D arrays



- Need a double for loop
  - Outer loop: Goes through the large array
  - Inner loop: Goes through the individual elements of the smaller arrays

```
int test[3][2];
for (i = 0; i < 3; i++) {
      for (j = 0; j < 2; j++) {
            printf("%d", test[i][j]);
      }
}
```

Printing 2D arrays

- Need a double for loop
  - Outer loop: Goes through the large array
  - Inner loop: Goes through the individual elements of the smaller arrays

```
int test[3][2];
for (i = 0; i < 3; i++) {
      for (j = 0; j < 2; j++) {
            printf("%d", test[i][j]);
      } printf("\n");
}
```

Populating 2D Arrays

- Same idea as populating 1D arrays

- Idea: Want to make an array of ‘*’s

```
size1 = 4; size2 = 5;
for (i = 0; i < size1; i++) {
    for (j = 0; j < size2; j++) {
        test[i][j] = ‘*’;
    }
}
```

xD Arrays

- Arrays can be more than 2D…
- …they can have as many dimensions as the programmer wants (hence **x**D)

- Ex: 5D Array
  - [[[[[‘hi’]]]]];