Multidimensional Arrays and C Strings

Section 1D

2D Arrays

```
int a[3][4] = { { 0, 1, 2, 3 },
                { 4, 5, 6, 7 },
                { 8, 9, 0, 1 } };
```

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Accessing Elements in 2D Arrays

For an array **a** with NROWS rows and NCOLS columns:

We want to compute or evaluate the elements along each **row**

```
for( row = 0; row < NROWS; row++ )
{
   for( col = 0; col < NCOLS; col++ )
   {
      /*
       * Process a[row][col]
       * rows change slower than
       * columns.
       */
   }
}
```

We want to compute or evaluate the elements along each **column**

```
for( col = 0; col < NCOLS; col++ )
{
   for( row = 0; row < NROWS; row++ )
   {
      /*
       * Process a[row][col]
       * columns change slower than
       * rows.
       */
   }
}
```

Exercise 1

Given the following student grades in a class, write a program that finds and shows the **lowest** and **highest** grade of the class, and the **average** grade per student.

	Exam 1	Exam 2	Exam 3	Exam 4
Amanda	77	68	86	73
Bill	96	87	89	78
Charles	70	90	86	81

Functions and 2D Arrays

Consider we want to write a *function* to print the contents of array **a**:

```
int a[3][4] = { { 0, 1, 2, 3 },
                { 4, 5, 6, 7 },
                { 8, 9, 0, 1 } };
```

A function with a multidimensional array as parameter should have the following signature:

```
void printArray( int b[][4], int nRows, int nColumns );
```

Array name

Leave off number of rows

But indicate number of columns!

Because we can't tell how many rows and columns are there just by looking at the array alone

The function definition would be as follows:

```
void printArray( int b[][4], int nRows, int nColumns )
{
   for( int r = 0; r < nRows; r++ )
   {
      for( int c = 0; c < nColumns; c++ )
         cout << b[r][c] << "\t";

      cout << endl;
   }
}
```

Which would print:

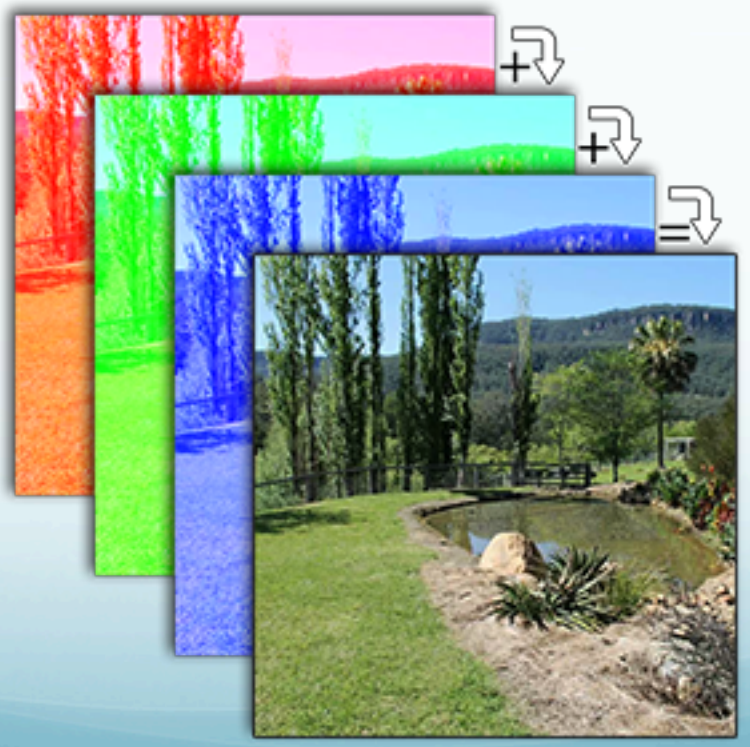
```
0       1       2       3
4       5       6       7
8       9       0       1
```

Exercise 2

- Following the description of exercise 1, write:
  - A function that returns the **lowest** grade of the class.
  - A function that returns the **highest** grade of the class.
  - A function that returns the **average grade** of a student, given **a 1D array** that contains her exam grades.

Multidimensional Arrays

- Not too common, but...
  - **3D** arrays are used to represent **RGB** images.
  - How would you make an RGB image into a **gray-scale** image (e.g. a gray-scale image is one for which R=G=B)?



C Strings

- A C String is an array of characters that finalizes with the **zero byte** '\0':

```
char s[] = {'c','h','a','r','m','a','n','d','e','r','\0'};
char s[11] = "charmander";
char s[] = "charmander";
const int EFFECTIVE = 10;
char s[EFFECTIVE + 1] = "charmander";
```

0	1	2	3	4	5	6	7	8	9	10
'c'	'h'	'a'	'r'	'm'	'a'	'n'	'd'	'e'	'r'	'\0'

C Strings Input and Output

- `char s[10 + 1];`
- **Input**:

```
cin >> s;                       // Reads only one word.
cin.getline( s, 10 + 1 ); // Expects, at most 10
                                // chars. When user presses
                                // enter, a '\0' is appended
                                // to s.
```

- **Output**:

```
cout << s;                      // Works as with any other
                                // basic type.
```

Exercise 3

- Write the following functions:
  - A function that computes the (effective) length of a C string **s**.
  - A function that copies a C string **t** into another C string **s**.

C Strings Functions

- From the `<cstring>` library:

```
char a[21], b[11] = "charmander";
```

Type	Function	Example
Assignment *s* = *t*	`strcpy( char s[],` `  const char t[] )`	`strcpy( a, b )` `// a = "charmander"`
Effective length	`int strlen( const char s[] )`	`strlen( a )` `// 10`
Concatenation *s* += *t*	`strcat( char s[],` `  const char t[] )`	`strcat(a, ", fight!")` `// a = "charmander, fight!"`

Comparison
s == *t*
s < *t*
s > *t*
	`int strcmp( const char s[],`
	`  const char t[] )`
		`strcmp( b, "charmander" )`
		`// 0: they are equal`
		
		`strcmp( b, "charmeleon" )`
		`// < 0: b is smaller`
		
		`strcmp( b, "bulbasaur" )`
		`// > 0: b is larger`

Exercise 4

- Are the following code snippets correct, or do they have errors?

a

```
char s[10];
strcpy( s, “hello” );
cout << s << endl;
```

b

```
char s[] = ‘a’;
cout << s << endl;
```

c

```
int x = strlen( ‘hello’ );
cout << x << endl;
```

d

```
char s[8];
strcpy( s, “Welcome!” );
cout << s << endl;
```

e

```
if( strcmp( string1, string2 ) )
   cout << “The strings are equal “;
```

Arrays of C Strings

- How do we create an array of C strings?
  - The C string is basically an array of characters, then, an array of C strings is an **array of arrays of characters**.
  - An array of C strings is a **matrix**! (OK... a 2D array...)

	0	1	2	3	4	5	6	7	8	9	10
0	'c'	'h'	'a'	'r'	'm'	'a'	'n'	'd'	'e'	'r'	'\0'
1	'b'	'u'	'l'	'b'	'a'	's'	'a'	'u'	'r'	'\0'	
2	'p'	'i'	'k'	'a'	'c'	'h'	'u'	'\0'			
3	's'	'q'	'u'	'i'	'r'	't'	'l'	'e'	'\0'		
4	'm'	'e'	'w'	'\0'							

```
char pokemons[5][10 + 1];
```

Exercise 5

Write a function that given an array of C strings (where each string is at most 10 effective characters), returns how many of them match another target string.

```
int tally( const char a[][10 + 1], int n,
           const char target[] );
```

Materials for Today

- You'll find these slides and solutions to the programming exercises at:

  http://cs.ucla.edu/~langel/cs31/session6