Arrays intro (& strings)
Ch 7

Why science teachers are not asked to monitor recess.
Highlights

- arrays

```cpp
int x[4];
x[0] = 1;
```

- string functions

```cpp
string x = "hello there!";
cout << x.substr(x.find('t'));
```
string

We have been using strings to store words or sentences for a while now.

However, when we type “string x” it does not turn blue, as it is not a fundamental type (like char).

Strings are basically a grouping of multiple chars together in a single variable.
String greeting = “Hello”;

The position of a character is called its index.

Note that the index starts from zero, not one (this is just to make your life miserable)
String greeting = “Hello”;

```
Hello
0 1 2 3 4
```

greeting.length();

returns value 5 (int)

Tells how many characters are in the variable
String concatenation does not automatically add a space (see: stringConcatenation.cpp)
strings

There are also some other useful functions (see book or google for a full list)

Some of the more useful ones are:
  .at(int index): character at the index
  .find(): finds first character or string
  .substr(int start): pulls out part of the original string

(see: string.cpp)
Arrays

Arrays are convenient ways to store similar data types (like multiple chars for a string).

Arrays are indexed starting from 0, so index 0 is the first element, index 1 is the second element ...

Unlike strings, you can make an array of whatever type you want (any type!)
Arrays - declaration

When making an array, you need both a type and a length.

The format for making an array is below:

```c
int x[5]; // 5 ints
```

- Type of things in array
- Variable name
- Length of array
Arrays - elements

To access an element of an array, use the variable name followed by the index in [ ]

\[ x[1] = 2; \]

(See: simpleArray.cpp)
Arrays

Note that the number in the [ ] is inconsistent:

1. First time (declaration): this is the length

2. All other times: this is the index of a single value inside the array

If you want to indicate a whole array, just use the variable name without any [ ] (more on this later)
There are actually two types of “strings” (multiple characters) in C++

A **C-String** is a char array, and this is what you get when you put quotes around words

```cpp
cout << "HI!\n"; // C-String
```

A **string** (the thing you #include) is a more complicated type called a **class** (few weeks)
C-Strings and strings

It is fairly easy to convert between C-Strings and strings:

```cpp
char cString[] = "move zig";
string IMAString = cString;
cout << IMAString.c_str() << endl;
// above converts it back to C-String
```

You can also convert between numbers and strings:

```cpp
char number1[20];
string number2;
cin >> number1 >> number2;
cout << "sum is: " << (atof(number1) + stod(number2)) << endl;
```

(see: stringConversion.cpp)
C-Strings and strings

C-Strings are basically strings without the added functions

```c
char word[] = {'o', 'm', 'g', '\0'};
```

You should end C-Strings with **null character**, as this tells cout when to stop displaying

This means you can initialize char arrays with quotes (**BUT NOT OTHER ARRAYS**) (see: cstring.cpp)