Classes
March 11, Ch 10.1 - 10.3
Announcements

Tests back

HW4B: Apparently hard
  - Re-uploaded instructions with a hint
Highlights

- classes

```cpp
class blah2 {
    public:
        void foo();
    private:
        std::string word;
};
```
You need the dot to differentiate between two different variables.

You can also think of the dot as possessive in English ( . → 's )

```c
struct date
{
    int day;
    int month;
    int year;
};

int main()
{
    date today;
    date midterm2;
    today.month = 11;
    midterm2.month = 11;
}
```
A **class** is functionally the same as a struct (creates a new data type)

However, the notation is slightly different (contains functions)

```c
struct date
{
    int day;
    int month;
    int year;
};
```

```c
class date
{
    public:
        int day;
        int month;
        int year;
        void print();
};
```
class

You can put const to the right of the function in a class to designate that it will not change any of the member variables.

```cpp
class date
{
public:
    int day;
    int month;
    int year;
    void print() const;
};
```

const means cannot change day, month or year.
To define a class functions, we need to specify the scope using :: (scope resolution)

```cpp
// class "date"'s version of print
void date::print() {
    cout << month << "/" << day << "/" << year;
}

... compared to ...

// not related to "date" class
void print() {
    cout << "Hello!\n";
}
```

(See: date.cpp)
Scope resolution is actually what namespaces are for: `using namespace std;`

Using the above lets us write:

```
cout << "Hi" << endl;
```

... instead of ...

```
std::cout << "Hi" << endl;
```

annoying to rewrite every time
The `::` is very similar to the `.` operator.

`:` is used to specify the location in a general sense (without a specific variable involved).

Example: Put socks on before shoes.

`.` is used to specify the ownership of a variable or function (owner is another variable).

Example: Tie my shoe laces.
class / structs

classes and structs make code much easier to modify in addition to organize

Learning how to write code is practice, this will become natural if you do it a lot

Writing code that can easily be added to is much more difficult
class

class inside of another class? Sure, why not!

(MULTI-CLAS SING)
Because wizards run out of spells

(see: nestedClass.cpp)
Suppose we wanted to make a partially filled array class with the following functions:

- void print()
- double get(int index)
- void add(double element)
- int length()
- int find(double element)

(See: arrayClass.cpp)