Structs/classes
Ch 10.1 - 10.3

You have no class.
Highlights

- classes (and structs)

```cpp
class blah2
{
    public:
        void foo();
    private:
        std::string word;
};
```
We also organize these slightly differently:

arrays = numbers to identify (index)

class = use names to identify

```
int x[10];
x[8] = 4;

car x;
x.fuel = 10;
```
A **struct** (structure) is functionally the same as a class (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.

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

const means cannot change day, month or year
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
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 ...

```cpp
// 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 aboves lets us write:
```
cout << "Hi" << endl;
```

... instead of ...
```
std::cout << "Hi" << endl;
```

annoying to rewrite every time
:: VS .

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

specific case
class

class inside of another class? Sure, why not!

(see: nestedClass.cpp)
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.

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