Bonus Day 3 Working with C++ Classes and Objects

working with complex data in C++

can add functions as members to structures

to call use member operator, for struct time with member function print time:

```
struct time {
    int hours;
    ...

    void print_time();
}
```

```
time.print_time();
```

to create its prototype:

```
void time::print_time(void){ function }
```

Using classes

just like structure:

```
class time {
    int variables...
};
```

in C++ and object is simply a declared data item created by using a class

instantiating- when create an instance of a class

determine which routines have access to data by using three additional keywords:

```
public
private
protected
```

by default classes are **private**- meaning data members and member functions are only accessible to themselves

structures are default public

public- any external source within program can access

usually member data is kept private or protected for inherited classes, and member functions are public and set up to modify private member data

```
class time {
    private:
        int hours;

    public:
        void add_hour(void);
};
```

allows you to **encapsulate** programs functionality

allows you to change data members without having to change all the programs that use your class

use classes instead of structures if member functions are going to be used

Constructors and Destructors

constructor- specialized member function- included with class. same name as class- used to create classes- modify definition to initialize data members

destructor- same name as class but has ~ in front. destroys object. destroyed when goes out of scope or program ends

very valuable to overload class's constructor allows for more dynamic input

Using Classes as data members

class inception

access same way as nested structure:

```
line1.start.x
```

uses example of class to record x and y of point nested in a line class that uses 2 point classes to define the line

Inheriting in C++

inheritance capability to create new classes by building upon existing classes

base class- class that is inherited from by another class

subclass- class that inherits from another class

to declare:

```
class subclass: public base class {
    protected:
        long subclassvariables;
    public:
        void sublcassfunction();
}
```

actually creating base class THEN creating subclass

for desctuctors- first subclass is called then base class is called