Constructor (7.14)

- Special member function to initialize data members
- It has the same name as the class
- It does not have a return value
- The constructor is called whenever an object of that class is created (instantiated)
- **`Time();`**

Constructor Example

- What would the implementation of the constructor look like?

```
Time::Time()
{
  hour = minute = second = 0;
}
```

Object-Oriented Features

- Information hiding
  - Separate the implementation from the interface
  - Objects are concerned with the interface, for example what functions are available to manipulate the data
  - Objects are not concerned with the implementation. They do not care how the functions do what they do, as long as they do it correctly

Overloaded Constructors (7.15)

- Overloaded constructors are the same as overloaded functions
- We could have multiple constructors in the Time class, each of which accepts a different number of arguments
- The appropriate constructor will be chosen based on the number of arguments used when creating the object
- Create multiple constructors for Time

Default Constructor

- The default constructor is the constructor with no arguments
- If you do not create any constructors in your class, then the default constructor will be created for you
- If you have a constructor that takes arguments, then the default constructor will be created for you
- It is good programming practice to always create a default constructor, why?

Default Arguments (7.15)

- You can set default arguments to constructors
- In the class definition, the constructor prototype will be
  - `Time(int = 0, int = 0, int = 0);`
- The function definition will be

```
Time::Time(int hr, int min, int
  sec)
{
   setTime(hr, min, sec);
}
```

Using Default Arguments

- By having default arguments in the constructor, we can now create objects of the Time class as follows:

```
Time cT1;

Time cT2(9);

Time cT3(9, 25);

Time cT4(9, 25, 30);

Time cT5(45, 90, 72);
```