More on C++ Classes

Destructors

A destructor is a special method of a class that is executed automatically when an instance of the class is destroyed. Unlike constructors, there can be only be one destructor for each class. The name of the destructor is the tilde character (~) followed by the name of the class. The destructor has no return type and takes no parameters. One common task performed in the destructor is to release resources held by the object. This may be releasing memory that was dynamically allocated, closing open files, closing database connections, closing network connections, etc.

Destructors are normally called in reverse order as the constructors. A global object's constructor is called before any functions, including the main function. The destructor for a global object is called when main ends. The Constructor for an automatic object is called when execution reaches the point where the object is created. Its destructor is called when execution leaves the object's scope. The constructor for a static object is called only once, when execution reaches the point where the object is created. Its destructor is called when the main function ends.

Programming Examples: CreateAndDestroy1.cpp & CreateAndDestroy2.cpp

Returning a Reference to Private Data

One common error is to return a reference to a private data member. Doing this gives direct access to the class's private data. Once a reference to private data has been handed out, the data can be changed, bypassing any data validation the class may have.

Revisiting the **`Time`** class example from earlier in the semester, the pitfalls of returning a reference to private data is illustrated.

In this program the **`setHours`** method returns a reference to the private data **`hours`** and the **`setMinutes`** method returns a pointer to the private data **`minutes`**. The program then uses these references and associated pointers to set the time to invalid values in several different ways.

Programming Examples: Time.h, Time.cpp, and TimeTester.cpp

Copying Objects

One object can be copied to another object using the assignment operator **=**. When this is done a *member wise* assignment is performed. This means that the value of each data member is copied from the source object to the destination object.

Programming Examples: Date.h, Date.cpp, and DateTester.cpp

Note that while copying object like this is acceptable for simple data types, it can cause problems when used with data members that are themselves instances of classes. We will see how to handle this in our next class.