CS 271
Nov 16, 2017

Prototype Using Default Arguments

```
TwoDayPackage( const string &, const string &, const string &, const string &, int, const
string &, const string &, const string &, const string &, int, double = 0.0, double =
0.0, double = 0.0 );
```

In the prototype we can specify default arguments.

Function Definition

In the function definition, there are no default values

```
TwoDayPackage::TwoDayPackage( const string &, const string &, const string &, const
string &, int, const string &, const string &, const string &, const string &, int,
double x , double y, double z) {
```

Prototype: double = 0.0, double = 0.0, double = 0.0

Function definition: double x , double y, double z

When the function is called, if one parameter is omitted, it will be parameter z. If two parameters are omitted, they will be y and z.

Order of constructor/destructor calls and execution:

```
Derived class constructor is called
Base class constructor is called
Base class constructor body executes
Derived class constructor body executes

Derived class destructor is called and executes
Base class destructor is called and executes
```

Which base class constructor is called?

If you don't explicitly call the base class constructor in the header of the derived class, C++ will call the default constructor of the base class.

```
EBook::EBook( string t, float p, string w ) {




}  // this version will call the default
   // constructor of Textbook

EBook::EBook( string t, float p, string w )
   : Textbook( t, p ) {



}  // explicitly calls the Textbook constructor
   // with two parameters
```

Mutator Functions that Return a Reference

```
Package& Package::setSenderName(const string & sname) {

   senderName = sname;
   return *this;

}
```

The pointer "this" must be dereferenced so that the function returns the calling object.