# Functions in a Hierarchy

* The logic that a derived class inherits from its base class is limited to the normal member functions of the base class.
* A Derived class does not by default inherit the constructors, the destructor or the copy assignment operator - that is, the special member functions - of the base class.

#### Shadowing

* A member function of a derived class **shadows** the base class member function with the same identifier.
* The C++ compiler binds a call to the member function of the derived class, if one exists.
* To **access the base class version** of a member function that **a derived class** version has shadowed, we **use scope resolution**.

| Base::identifier(arguments) |
| --- |

* Where **base** identifies the class to which the shadowed function belongs.

**Good Design Tip**

By calling Person::display() withing Student::display() , we hide the hierarchy from the client code. The main() function is hierarchy agnostic.

#### Constructors

* A derived class does not inherit a base class constructor by default.
* That is, if we do not declare a constructor in our definition of the derived class, the compiler inserts an empty no-argument constructor by default.
* The compiler constructs an instance of the derived class in three steps:

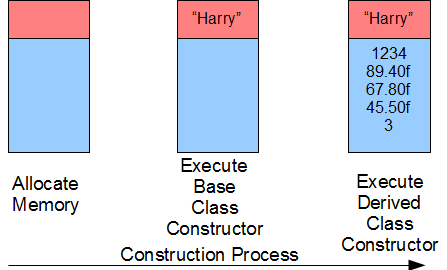
1. Memory Allocation:

* The memory for the entire object (both base and derived parts) is allocated as a single contiguous block.
* This means that the memory for the base class and any additional members introduced by the derived class are allocated together.
* The size of the allocated memory depends on the combined size of the base class and derived class.

1. Base Class Initialization:

* The base class constructor initializes the base class members.

1. Derived Class Initialization:

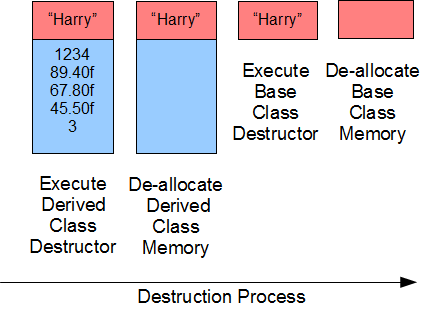
* After the base class is initialized, the derived class constructor is invoked.
* The derived class constructor initializes the derived class members.
* Initialization Order:
* The base class constructor is called first to initialize the base class part of the object.
* Then, the derived class constructor is called to initialize the derived class part.
* The order of constructor calls is always from base to derived.

#### Passing Arguments to a Base Class Constructor

* Each constructor of a derived class, other than the no-argument constructor, receives in its parameters all of the values passed by the client.
* Each constructor forwards the value for the base class part of the object to the base class constructor.
* The base class constructor uses the values received to build the base class part of the object.
* The derived class constructor used the values received to complete building the derived class part of the object.
* Refer to example on course site

#### Destructor

* A derived class does not inherit the destructor of its base class.
* Destructors execute in opposite order to the order of their object’s construction.
* That is, the derived class destructor always executes before the base class destructor



#### Summary

* A member function of a derived class shadows an identically named member function of a base class
* A derived class does not inherit the destructor, assignment operators or helper functions of a base class
* A derived class does not by default inherit the constructor of a base class, but we may add syntax to allow inheritance where the derived class constructor does not contain logic to set its instance variables
* Constructors in an inheritance hierarchy execute in order from the base class to the derived class
* Destructors in an inheritance hierarchy execute in order from the derived class to the base class