Inheritance and Polymorphism

- Inheritance versus generecity (templates)
- Base Classes and Derived Classes.
- Subclassing for Specialisation/Specification.
- Relationships Among Objects in an Inheritance Hierarchy.
- Polymorphism, Virtual Functions and Dynamic Binding.

Inheritance

Form:

`class` *derived-class* : *access base-class*

where:

- *derived-class* is derived from *base-class*.
- access can be:
  - `public`: public (or protected) members of the base class are public (or protected) in the derived class.
  - `private`: public and protected members of the base class become private members of the derived class.
  - or `protected`: public and protected members of the base class become protected members of the derived class.

Note: constructors are not inherited. Each declaration of an object of the derived class causes execution of the base class constructor before the derived class constructor.

- Interface (.h) of derived class:
  - Contains declarations for new member functions
  - Also contains declarations for inherited member functions to be changed
- Inherited member functions NOT declared: Automatically inherited unchanged
- Implementation of derived class will:
  - Define new member functions
  - Redefine inherited functions as declared

Redefining vs. Overloading

- Redefining in derived class:
  - SAME parameter list
  - Essentially re-writes same function
- Overloading:
  - Different parameter list
  - Defined new function that takes different parameters
  - Overloaded functions must have different signatures

Multiple Inheritance

- Derived class can have more than one base class!
- Syntax just includes all base classes separated by commas:

```
class derivedMulti :  public base1,
base2 ...
```

- Possibilities for ambiguity are endless! Dangerous undertaking!
- Some believe should never be used
- Certainly should only be used be experienced programmers!