Advanced Programming

Inheritance (2)

Topics

- Multiple Inheritance
- Virtual Classes
- Virtual Functions
- Abstract Classes
- Static Variables
- Examples

Compatibility Between Base and Derived Classes

- An object of a derived class can be treated as an object of its base class.
- The reverse is not true.

Passing Derived Class Objects to Functions

- In passing by value a local copy including the base class part (static part) is passed to the function
- In pass by reference the overridden functions are passed

Multiple Inheritance

- Deriving directly from more than one class is usually called multiple inheritance.
- The derived class will have the properties of all base classes

Example

```
class Animal
{
----------
};
class Graphics
{
-----------
};
class Snake: public Animal, public Graphics
{
}
```

Diamond Problem in Multiple Inheritance

- If both of the base classes are derived from the same parent class, then that parent class is repeated in the newly derived class
- Example

  class B:public A

  class C:public A

  class D:public B, public C

Virtual Classes

- If a class is defined as virtual, it appears in the derived classes only once.
- A virtual class must have:
  - No constructor or
  - Constructor with no parameters or,
  - Constructor with parameters having default values

Example

```
class Port
{ -- };
class Region: virtual public Port
{ ---};
class Menu; virtual public port
{---};
class Window: public Region, public Menu
{----};
```

Virtual Functions

- A pointer to a derived class is type-compatible with a pointer to its base class.
- However, only the members of the base class can be accessed through a base-class type pointer

Example

- Define a Polygon class and derive two classes named Triangle and Rectangle. Define pointers to Polygon. Create objects of type Triangle and Rectangle. Call member functions.

Virtual Functions (cont.)

- If the base class has a virtual function the derived classes can re-define it.
- The pointer to the base class will call the right function.
- If the function is not defined as virtual using the pointer to the base class, we always call the base class version of the function

Example

- Modify the Polygon, Triangle, Rectangle classes example. Add a virtual function and call it using base class pointers.

Abstract Classes

- An abstract class is, conceptually, a class that cannot be instantiated and is usually implemented as a class that has one or more pure virtual (abstract) functions.
- A pure virtual function is one which **must be overridden** by any concrete derived class.

Example

```
class AB
{
   public:
   virtual void f() = 0;
};
```

Abstract Classes (cont.)

- Although we cannot instantiate from an abstract class, it is possible to create pointers to it.
- Abstract classes are also useful in defining function parameters.

Static Variables

- When a member of a class is declared as static it means no matter how many objects of the class are created, there is only one copy of the static member.
- Static variables are initialized out of the class.

Example

```
class Box
{
   public:
    static int objectCount;
    Box(double l=2.0, double b=2.0, double h=2.0);
    double Volume() { return length * breadth * height; }
   private:
     double length; // Length of a box
     double breadth; // Breadth of a box
     double height; // Height of a box
};
```

```
Box:: Box(double l, double b, double h)
{
    length = l;
    breadth = b;
    height = h; // Increase every time object is
    created objectCount++;
}
```

```
int Box::objectCount = 0;
int main(void)
{
    Box Box1(3.3, 1.2, 1.5);
    Box Box2(8.5, 6.0, 2.0);
    cout << "Total objects: " << Box::objectCount
 << endl;
  return 0;
}
```