PROTECTED BASE CLASS INHERITANCE AND INHERITING MULTIPLE BASE CLASSES

Protected Base-Class Inheritance

It is possible to inherit a base class as protected. When this is done, all public and protected members of the base class become protected members of the derived class.

For example,

```cpp
#include <iostream>
using namespace std;

class base {
    protected:
        int i, j; // private to base, but accessible by derived
    public:
        void setij(int a, int b) { i=a; j=b; }
        void showij() { cout << i << " " << j << "\n"; }
    }
    // Inherit base as protected.

class derived : protected base {
    int k;
    public:
        // derived may access base's i and j and setij().
        void setk() { setij(10, 12); k = i*j; }
        // may access showij() here
        void showall() { cout << k << " "; showij(); }
    }
    int main()
    {
        derived ob;
    }
```
// ob.setij(2, 3); // illegal, setij() is
// protected member of derived
ob.setk(); // OK, public member of derived
ob.showall(); // OK, public member of derived
// ob.showij(); // illegal, showij() is protected
// member of derived
return 0;
}
As you can see by reading the comments, even though `setij()` and `showij()` are public members of `base`, they become protected members of `derived` when it is inherited using the `protected` access specifier. This means that they will not be accessible inside `main()`.

**Inheriting Multiple Base Classes**

It is possible for a derived class to inherit two or more base classes. For example, in this short example, `derived` inherits both `base1` and `base2`.

// An example of multiple base classes.
#include <iostream>
using namespace std;
class base1 {
    protected:
    int x;
    public:
    void showx() { cout << x << "\n"; }
};
class base2 {
    protected:
    int y;
    public:
    void showy() { cout << y << "\n"; }
};
// Inherit multiple base classes.
class derived: public base1, public base2 {
public:
void set(int i, int j) { x=i; y=j; }
};
int main()
{
derived ob;
ob.set(10, 20); // provided by derived
ob.showx(); // from base1
ob.showy(); // from base2
return 0;
}

As the example illustrates, to inherit more than one base class, use a comma-separated list. Further, be sure to use an access-specifier for each base inherited.