BASE CLASS IN CPP

Base-Class Access Control

When a class inherits another, the members of the base class become members of the derived class.

Class inheritance uses this general form:

```
class derived-class-name : access base-class-name {
// body of class
};
```

The access status of the base-class members inside the derived class is determined by *access*. The base-class access specifier must be either **public**, **private**, or **protected**. If no access specifier is present, the access specifier is **private** by default if the derived class is a **class**. If the derived class is a **struct**, then **public** is the default in the absence of an explicit access specifier. Let's examine the ramifications of using **public** or **private** access. (The **protected** specifier is examined in the next section.) When the access specifier for a base class is **public**, all public members of the base become public members of the derived class, and all protected members of the base become protected members of the derived class. In all cases, the base's private elements remain private to the base and are not accessible by members of the derived class.

For example, as illustrated in this program, objects of type **derived** can directly access the public members of **base**:

```
#include <iostream>
using namespace std;
class base {
int i, j;
public:
void set(int a, int b) { i=a; j=b; }
void show() { cout << i << " " << j << "\n"; }
};
class derived : public base {
int k;
public:
```

```
derived(int x) { k=x; }
void showk() { cout << k << "\n"; }
};
int main()
{
derived ob(3);
ob.set(1, 2); // access member of base
ob.show(); // access member of base
ob.showk(); // uses member of derived class
return 0;
}
```

When the base class is inherited by using the **private** access specifier, all public and protected members of the base class become private members of the derived class.

For example, the following program will not even compile because both **set( )** and **show( )** are now private elements of **derived**:

```
// This program won't compile.
#include <iostream>
using namespace std;
class base {
int i, j;
public:
void set(int a, int b) { i=a; j=b; }
void show() { cout << i << " " << j << "\n";}
};
// Public elements of base are private in derived.
class derived : private base {
int k;
public:
derived(int x) { k=x; }
void showk() { cout << k << "\n"; }
};
```

```
int main()
{
derived ob(3);
ob.set(1, 2); // error, can't access set()
ob.show(); // error, can't access show()
return 0;
}
```

*When a base class' access specifier is **private**, public and protected members of the base become private members of the derived class. This means that they are still accessible bymembers of the derived class but cannot be accessed by parts of your program that are not members of either the base or derived class.*

Source : http://elearningatria.files.wordpress.com/2013/10/cse-iii-object-oriented-programming-with-c-10cs36-notes.pdf