```
#include<iostream>
using namespace std;
class base
{
	protected:
		int base_number;
	public:
		base(int=0);
		base(const base&);
		~base();
		int get_base_number()const;
		base& operator=(const base&);
};
inline base::base(int x) : base_number(x)
{
	cout << "In base constructor" << endl;
}
inline base::base(const base& b) : base_number(b.base_number)
{
	cout << "In base copy constructor" << endl;
}
inline base::~base()
{
	cout << "In base destructor" << endl;
}
inline int base::get_base_number()const
{
	return base_number;
}
inline base& base::operator=(const base& b)
{
	cout << "base operator =" << endl;
	if(&b != this)
		base_number = b.base_number;
	return *this;
}
class derived : public base
{
		int derived_number;
	public:
		derived(int = 0, int = 0);
		derived(const derived&);
		~derived();
		int get_derived_number()const;
		derived& operator=(const derived&);
};
```

```
inline derived::derived(int a, int b) : base(a), derived_number(b)
{
	cout << "In derived constructor" << endl;
}
inline derived::derived(const derived& d) : base(d),derived_number(d.derived_number)
{
	cout << "In derived copy constructor" << endl;
}
inline derived::~derived()
{
	cout << "In derived destructor" << endl;
}
inline int derived::get_derived_number()const
{
	return derived_number;
}
inline derived& derived::operator=(const derived& d)
{
	cout << "derived operator=" << endl;
	if(&d != this)
	{
		base::operator=(d);
		derived_number = d.derived_number;
	}
	return *this;
}
int main()
{
	derived d1(1,2);
	derived d2;
	d2 = d1;
	cout << "d2 = (" << d2.get_base_number()  << "," << d2.get_derived_number() << ")" << endl;

	return 0;
}
```