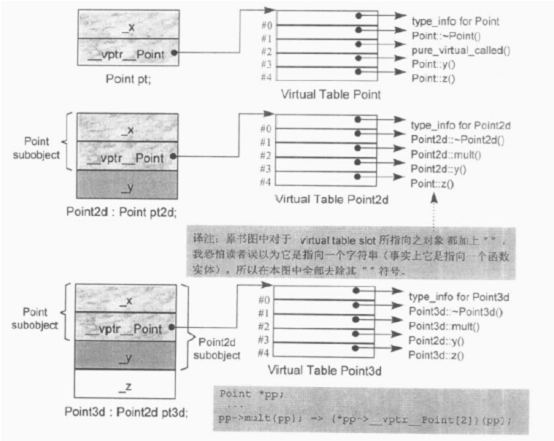
1. Member function type
2. Static:
3. can only access static member data
4. can not be const, volatile, virtual fuction
5. can access directly without objects.
6. The type of function’s address is a non-member function pointer.
7. Non-static
8. Virtual: can be effective through pointer or reference. (\*ptr->vptr[num])();, num is the function in virtual table slot number.

Non-member function has the same effective with member function.

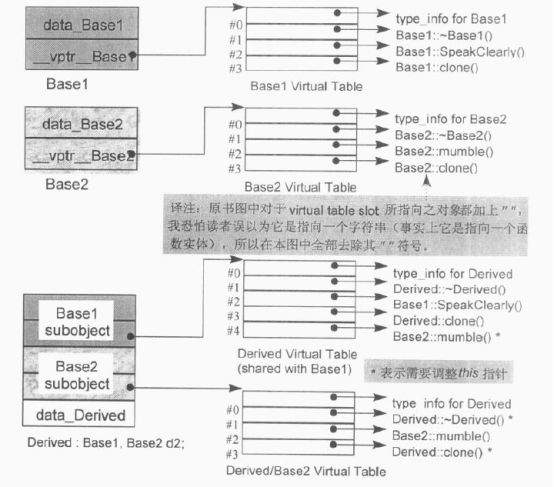
1. Virtual function
2. Single inheritance
3. A pointer named vptr was made out in the object by compiler. Vptr point to vitual table which contains all the active virtual function of the class. Each class has only one virtual table.
4. Each virtual function is assign to the slot of the virtual table.



How about the virtual function comes out when a new class derived from a base class.

1. A virtual function inherits from base class don’t change it. It will copy the function address to the slot of virtual table of derived class.
2. A virtual function inherits from the base class but derived class has its own version. It will copy the own function address to the slot of virtual table of derived class.
3. A virtual function add by derived class, the size of the virtual table of derived class will increase one slot, then the address of the function will put into the slot.
4. Multiple inheritance

A derived class has n virtual table, n is the upper level base class number. All the virtual function in the derived class will be put into Base1 virtual table of the class. (such as destructor and clone)



Class Derived : public base1, public base2

1. Virtual destructor

Base2 \*b2 = new Drived;

Delete b2; /\*adjust b2 the sizeof(Base1) backward to base1/derived\*/

1. Base2 virtual function func, Base1 no and Derive do not change it

Base1/Derive \*b = new Drived;

p->func; /\*adjust b the sizeof(Base1) forward to base2\*/

1. Clone virtual function: virtual Base1/Base2/Derived \*clone(); Derived has its own version, so it should be put into the Base1 virtual table clone.

Base2 \*b2 = new Derived;

Base1 \*new = b->clone;/\*adjust b2 the sizeof(Base1) backward to base1/derived \*/

1. pointer to Function Member: returntype (className::name)(list);

Class T { Public: int func(int)}/\*should be public\*/

int (T::\*p)(int) =&T::func;

If func is not a virtual function it means a memory address or an int number for slot in virtual table。

int (T::\*p)(int) =&T::func;

T t; t.(\*p) means t.func

T \*tn = &t; tn->(\*p) means tn->func;