Multiple Inheritance

---

none-virtual base none-polymorphism:

```
struct B1{ void f(); };
struct B2{ void f(); };
struct Derived: B1, B2{ public: /* ... */ };
int main()
{
    Derived* pd = new Derived;
    B1* pB1 = pd; B2* pB2 = pd; //up-cast to different base parts, base-ptr
val varies
    pB1->f(); pB2->f(); //calls corresponding base versions
    pd->B1::f(); pd->B2::f(); //ambiguous bases func-names must be
qualified
    auto pd1 = static_cast<Derived*>(pB1);//static_cast down cast ok
    auto pd2 = static_cast<Derived*>(pB2);//static_cast down cast ok
    // auto pB2_cast = static_cast<B2*>(pB1);//error casting from one base
to another
}
    /*
        Layout:
        class B1        size(1):
            +---
            +---
        class B2        size(1):
            +---
            +---
        class Derived   size(1):
            +---
        0       | +--- (base class B1)
            | +---
        1       | +--- (base class B2)
            | +---
            +---
    */
}
```

Up/down casting between B1/B2 and Derived is OK(Addr adjustment got from Derived Layout)

---

none-virtual base polymorphism:

```
struct B1{ virtual void f(); }; struct B2{ virtual void f(); };
struct Derived: B1, B2{ public: void f(); };
int main(){
    Derived* pd = new Derived; B1* pB1 = pd; B2* pB2 = pd;
    pB1->f(); pB2->f(); //calls corresponding base versions
    dynamic_cast<B1*>(pB2); //can cross-cast through dynamic cast
```

```
    pd->f(); //ok, Derived::f overwrites f in two vftable
}

/*layout
class B1        size(8):
        +---
 0      | {vfptr}
        +---
B1::$vftable@:
        | &B1_meta
        |  0
 0      | &B1::f
B1::f this adjustor: 0

class B2        size(8):
        +---
 0      | {vfptr}
        +---
B2::$vftable@:
        | &B2_meta
        |  0
 0      | &B2::f
B2::f this adjustor: 0

class Derived   size(16):
        +---
 0      | +--- (base class B1)
 0      | | {vfptr}
        | +---
 8      | +--- (base class B2)
 8      | | {vfptr}
        | +---
        +---

Derived::$vftable@B1@:
        | &Derived_meta
        |  0
 0      | &Derived::f
Derived::$vftable@B2@:
        | -8
 0      | &thunk: this-=8; goto Derived::f
Derived::f this adjustor: 0

*/
```

As shown in the layout, Class Derived has two derived virtual tables in which methods are searched, each has one defination of void f(). They are all overwritten by Derived::f. If Derived::f does not exist, pd->f() would be ambiguous.

---

virtual base classes

```
struct B1{ virtual void f(); }; struct B2{ virtual void f(); };
struct Derived: virtual B1, virtual B2{ public: void f(); };
/* Layout
class B1        size(8):
        +---
 0      | {vfptr}
        +---
B1::$vftable@:
        | &B1_meta
        |  0
 0      | &B1::f
B1::f this adjustor: 0

class B2        size(8):
        +---
 0      | {vfptr}
        +---
B2::$vftable@:
        | &B2_meta
        |  0
 0      | &B2::f
B2::f this adjustor: 0

class Derived   size(24):
        +---
 0      | {vbptr}
        +---
        +--- (virtual base B1)
 8      | {vfptr}
        +---
        +--- (virtual base B2)
16      | {vfptr}
        +---

Derived::$vbtable@:
 0      | 0
 1      | 8 (Derivedd(Derived+0)B1)
 2      | 16 (Derivedd(Derived+0)B2)
Derived::$vftable@B1@:
        | -8
 0      | &Derived::f
Derived::$vftable@B2@:
        | -16
 0      | &thunk: this-=8; goto Derived::f

Derived::f this adjustor: 8

vbi:       class  offset o.vbptr  o.vbte fVtorDisp
              B1       8       0       4 0
              B2      16       0       8 0
*/
```

The derived class has a vbtable holding all the virtual bases. We still have two vftables.

```
struct VBase{
    int i;
    virtual void f();
    virtual void g();
    virtual void h();
};
struct B1: virtual VBase{ virtual void f(); virtual void h();};
struct B2: virtual VBase { virtual void g(); virtual void h();};
struct Derived: B1,  B2{ void h(); };

/* layout
class VBase     size(16):
        +---
 0      | {vfptr}
 8      | i
        | <alignment member> (size=4)
        +---
VBase::$vftable@:
        | &VBase_meta
        |  0
 0      | &VBase::f
 1      | &VBase::g
 2      | &VBase::h
VBase::f this adjustor: 0
VBase::g this adjustor: 0
VBase::h this adjustor: 0

class B1        size(24):
        +---
 0      | {vbptr}
        +---
        +--- (virtual base VBase)
 8      | {vfptr}
16      | i
        | <alignment member> (size=4)
        +---
B1::$vbtable@:
 0      | 0
 1      | 8 (B1d(B1+0)VBase)
B1::$vftable@:
        | -8
 0      | &B1::f
 1      | &VBase::g
 2      | &B1::h
B1::f this adjustor: 8
B1::h this adjustor: 8
vbi:       class  offset o.vbptr  o.vbte fVtorDisp
           VBase       8       0       4 0

class B2        size(24):
```

```
        +---
 0      | {vbptr}
        +---
        +--- (virtual base VBase)
 8      | {vfptr}
16      | i
        | <alignment member> (size=4)
        +---
B2::$vbtable@:
 0      | 0
 1      | 8 (B2d(B2+0)VBase)
B2::$vftable@:
        | -8
 0      | &VBase::f
 1      | &B2::g
 2      | &B2::h
B2::g this adjustor: 8
B2::h this adjustor: 8
vbi:       class  offset o.vbptr  o.vbte fVtorDisp
           VBase       8       0       4 0

class Derived   size(32):
        +---
 0      | +--- (base class B1)
 0      | | {vbptr}
        | +---
 8      | +--- (base class B2)
 8      | | {vbptr}
        | +---
        +---
        +--- (virtual base VBase)
16      | {vfptr}
24      | i
        | <alignment member> (size=4)
        +---
Derived::$vbtable@B1@:
 0      | 0
 1      | 16 (Derivedd(B1+0)VBase)
Derived::$vbtable@B2@:
 0      | 0
 1      | 8 (Derivedd(B2+0)VBase)
Derived::$vftable@:
        | -16
 0      | &thunk: this-=8; goto B1::f
 1      | &B2::g
 2      | &Derived::h
Derived::h this adjustor: 16
vbi:       class  offset o.vbptr  o.vbte fVtorDisp
           VBase      16       0       4 0
*/
```

In this case, the vbptr of B1, B2 and Deriveds are all pointing to the same place. Even though class B1 and B2 does have VBase contained, as well as the vbptr, in Derived, the parts of B1 and B2 do not contain VBase any more(their vbptr points to it), and the VBase(together with its vfptr) is "bubbled up" to the Derived class. In this implementation, the vftable of a class does not contain functions in the base class even if the function is overwriten in derived class.