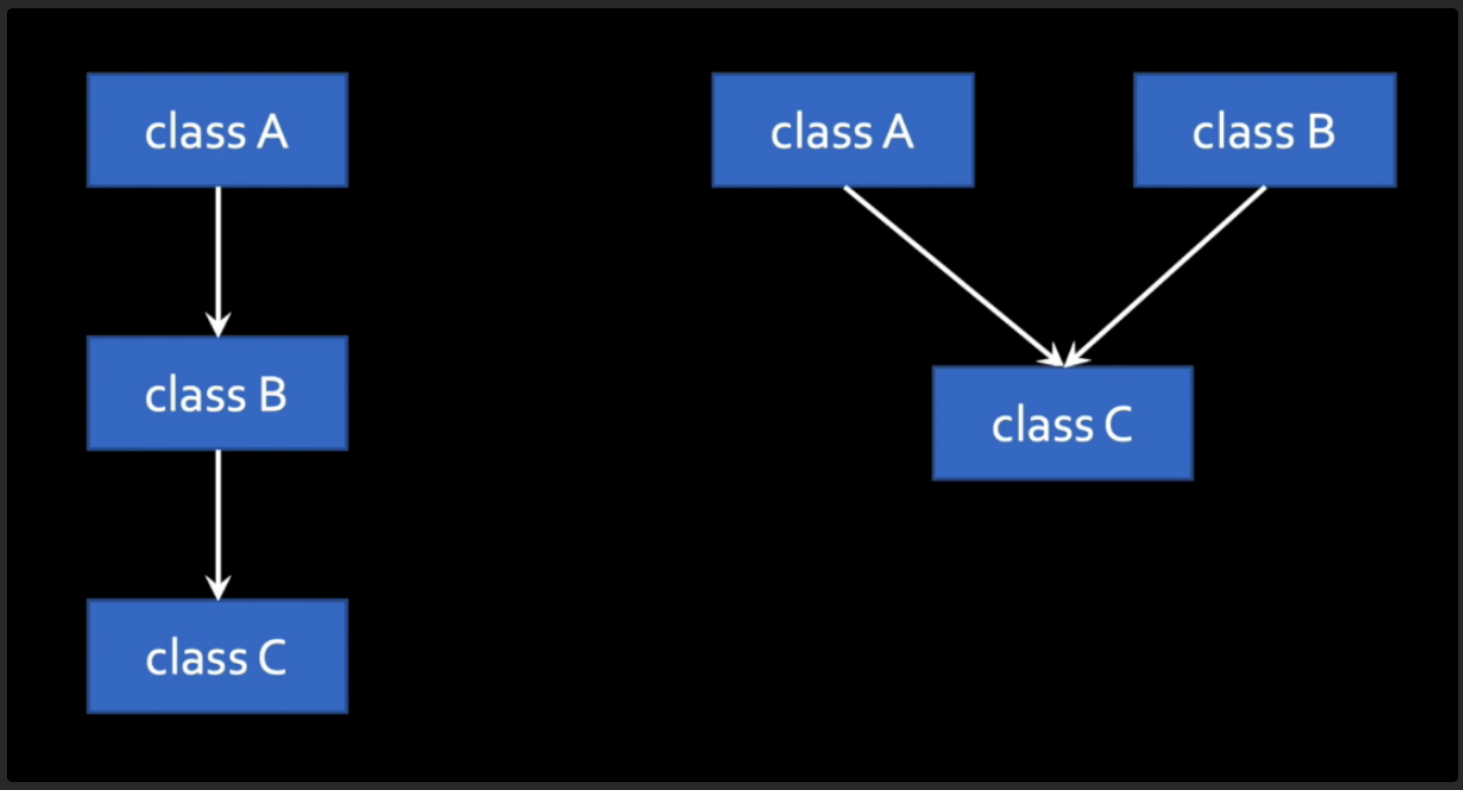
3.2多继承

多继承MultipleInheritance

- 单继承：一个派生类只有一个直接基类。
- 多继承：一个派生类有多个直接基类。



多继承

- 多继承派生类的定义形式如下：

```
class<派生类名>:[<继承方式><基类名1>,[<继承方式>]<基类名2>, ... ,
{
    [<派生类类体>]
}
```

  - 多重继承的派生方式也有private、public和protected三种，各基类的派生方式可以不同。

- 多继承下派生类的构造函数：

```
<派生类名>(<形参声明>):<基类名1>(<参数表1>)<基类名2>(参数表2>),…
{
<派生类构造函数体>
}
```

例子：多继承

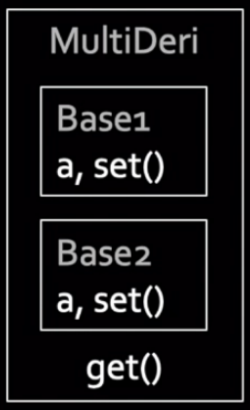
```
#include <bits/stdc++.h>
using namespace std;

class Base1
{
private:
```

```
    int a;
public:
    Base1(int x)
    {
        a = x;
        cout << "Base1 Constructor!" <<endl;
    }
    int geta() {return a;}
};
class Base2
{
private:
    int b;
public:
    Base2(int x)
    {
        b = x;
        cout << "Base2 Constructor!" <<endl;
    }
    int getb() {return b;}
};
//定义顺序决定基类构造函数的调用顺序
class Derived : public Base1, public Base2
{
private:
    int c;
public:
    //基类构造函数的调用顺序与排列顺序无关
    Derived(int x, int y, int z) : Base2(z), Base1(y)
    {
        c = x;
        cout << "Derived Constructor!" <<endl;
    }
    void show() {cout << geta() << ", " << getb() << ", " << c << endl;}
};
int main()
{
    Derived obj(1, 2, 3);
    obj.show();
    return 0;
}
```

虚基类：问题的产生

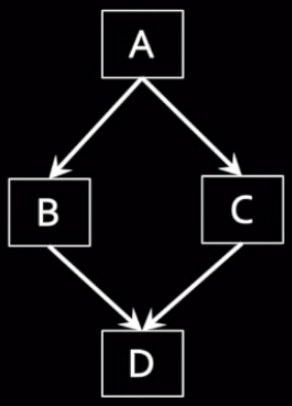
```cpp
class Base1 {
public:
    int a;
    void set(int a) { this->a = a; }
};
class Base2 {
public:
    int a;
    void set(int a) { this-> a  = a; }
};
class MultiDeri : public Base1, public Base2 {
public:
    int get() { return this->a; }
};
```

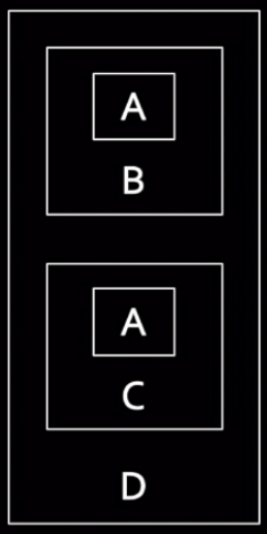


Error:
Member 'a' found in multiple base classes of different types
Member 'set' found in multiple base classes of different types

二义性问题：被继承的基类有同名成员。

```cpp
class A {
public: int a;
};
class B : public A{
public: int b;
};
class C : public A {
public: int c;
};
class D : public B, public C {
public: int d;
};
```





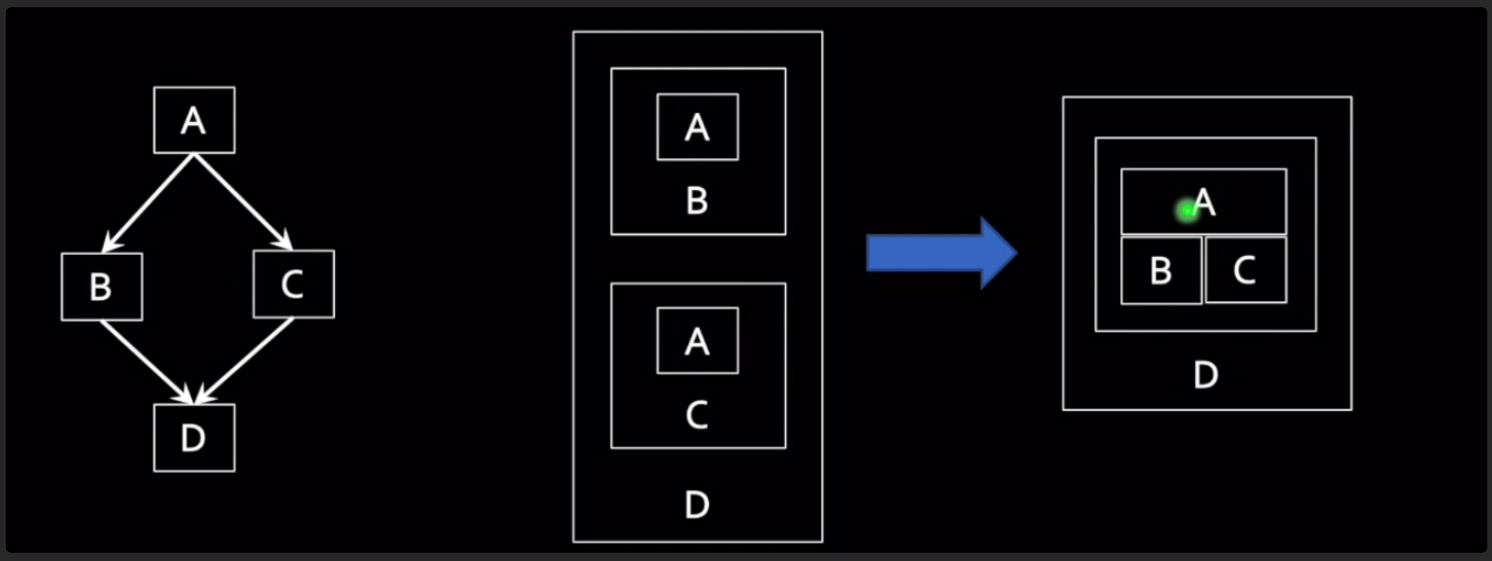
Error:
error C2385: 'D::a' is ambiguous

二义性问题（菱形缺陷）：
父类没有同名成员，但有共同的爷类。

虚基类

- 解决方法：
  1. 作用域限定符，d1.B::a,d1.C::a。

2. 虚基类派生，对同名成员只保留一个副本。虚基类是一种派生方式，不是一种类。



虚基类及其构造函数的调用

- 基类名称前加 **virtual** 即为虚基类。
- 调用虚基类的派生类的构造函数时，首先要调用虚基类的构造函数。
- 在多层次的继承关系中，虚基类的构造函数只能被调用 **一次** 。
- 虚基类的构造函数由 **最派生类** （创建对象的类）调用，最派生类的非虚基类对虚基类构造函数的调用将被忽略。
- 隐式调用时，最派生类自动调用虚基类的默认构造函数。
- 若虚基类没有默认构造函数，虚基类的所有直接或间接派生类都必须显式调用，

例子：虚基类

```
#include <bits/stdc++.h>
using namespace std;

class A
{
private: int a;
public: A (int x) : a(a){cout << "A Constructer: " << a << endl;}
        ~A(){cout << "A Destroyed. " << endl;}
};

class B1 : public virtual A
{
private: int b1;
public: B1 (int a, int b1) : A(a)
    {
        this->b1 = b1;
        cout << "B1 Constructer: " << b1 << endl;
    }
        ~B1(){cout << "B1 Destroyed. " << endl;}
};
class B2 : public virtual A
{
private: int b2;
public: B2 (int a, int b2) : A(a)
```

```
    {
        this->b2 = b2;
        cout << "b2 Constructer: " << b2 << endl;
    }
        ~B2(){cout << "b2 Destroyed. " << endl;}
};

class C : public B1, public B2
{
private: int c;
public:
    C (int a, int b1, int b2, int c) : B1(a, b1), B2(a, b2), A(a)
    {
        this->c = c;
        cout << "C Constructer: " << c << endl;
    }

        ~C(){cout << "C Destroyed. " << endl;}
};

int main()
{
    C c1(1, 2, 3, 4);
    return 0;
}
```