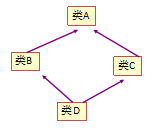
重复继承是指一个派生类多次继承同一个

基类，C++中允许出现重复继承。



**解决继承的重复问题**

解决继承的重复问题有两种方法：

   —使用作用域分辨符来唯一标识并分别访问他们；

   —将直接基类的共同基类设置为虚基类

        》从不同的路径继承过来的该类成员在内存中

            只拥有一个复制，这样就解决了同名成员的

            唯一标识问题

#include<iostream>

using namespace std;

class A

{

public:

int x;

A(int a){x=a;}

};

class B:public A

{

public:

int y;

B(int a, int b):A(b){y=a;}

};

class C:public A

{

public:

int z;

C(int a, int b):A(b){z=a;}

};

class D:public B, public C

{

public:

int m;

D(int a, int b, int c, int d, int e):B(a,b),C(c,d){m=e;}

void disp()

{

cout<<"x="<<B::x<<", y="<<y<<endl;

cout<<"x="<<C::x<<", z="<<z<<endl;

cout<<"m="<<m<<endl;

}

};

int main()

{

D d1(1,2,3,4,5);

d1.disp();

return 0;

}

上面代码可以看出是重复继承了A基类，如果直接引用A基类中的x就会报错，这是因为编译器不知道你指的到底是从C那里继承过来的A对象呢还是从B那里继承过来的A对象。

于是虚基类诞生了，将B和C的继承方式改为虚继承，那么D访问自己从A那里继承过来的成员就不会有二义性问题了，也就是将D对象里的A对象统一为一个了。

#include<iostream>

using namespace std;

class A

{

public:

int x;

A(int a=0){x=a;}

};

class B:virtual public A//由公共基类A虚拟派生出类B

{

public:

int y;

B(int a, int b):A(b){y=a;}

};

class C:virtual public A//由公共基类A虚拟派生出类C

{

public:

int z;

C(int a, int b):A(b){z=a;}

};

class D:public B, public C//由基类B,C派生出类D

{

public:

int m;

D(int a, int b, int c, int d, int e):B(a, b),C(c, d){m=e;}

void disp(){

cout<<"x="<<x<<", y="<<y<<endl;

cout<<"x="<<x<<", z="<<z<<endl;

cout<<"m="<<m<<endl;

}

};

int main()

{

D d1(1,2,3,4,5);

d1.disp();

d1.x=4;

d1.disp();

return 0;

}

这样就不会报错了！！！！