

```

//菱形继承(钻石继承) right example
class Base
{
public:
    int base_data;
};

class A : virtual public Base
{
public:
    int a_data;
};

class B : virtual public Base
{
public:
    int b_data;
};

class D : public A, public B //多继承
{
public:
    int d_data;
};

void main()
{
    cout<<sizeof(D)<<endl;
    D d;
    d.a_data = 1;
    d.b_data = 2;
    d.d_data = 3;
    d.A::base_data = 5;
    d.B::base_data = 6;
    d.base_data = 0;
    cout<<"sizeof(d) is "<< sizeof(d) << endl; //24
    cout<<&(d.A::base_data)<<endl;
    cout<<&(d.B::base_data)<<endl;
}

```

局部变量

搜索(Ctrl+E) 搜索深度: 3

名称	值	类型
d	(d_data=-858993460)	D
A	(a_data=1)	A
Base	(base_data=-858993460)	Base
a_data	1	int
B	(b_data=2)	B
Base	(base_data=-858993460)	Base
b_data	2	int
Base	(base_data=-858993460)	Base
base_data	-858993460	int
d_data	-858993460	int

自动窗口 局部变量

派生类对象d的大小刚好是24个字节

内存 1

地址: 0x004FFC64

地址	内容	注释
0x004FFC64	48 9b 6c 00 01 00 00 00 80 9b 6c 00 02 00 00 00 03 00 00 00 05 00 00 00 cc cc cc cc a0 fc 4f 00	base_data, a_data, b_data, d_data, vbptr
0x004FFC68	63 30 6c 00 01 00 00 00 00 59 88 00 70 7d 88 00 01 00 00 00 00 59 88 00 70 7d 88 00 fc fc 4f 00	
0x004FFC6C	07 2f 6c 00 10 88 c9 2b ca 13 6c 00 ca 13 6c 00 00 40 24 00 00 00 00 00 00 00 00 00 00 00	
0x004FFC70	00 00	
0x004FFC74	00 00	
0x004FFC78	0c fa 4f 00 39 31 6c 00 1c fa 4f 00 59 63 7b 77 00 40 24 00 00 00 00 00 00 00 00 00 00	
0x004FFD04	00 40 24 00 8f 22 24 c7 00 00 00 00 00 00 00 00 00 40 24 00 00 00 00 00 00 00 00 00	
0x004FFD08	00 00	
0x004FFD0C	00 00	
0x004FFD10	0d 8f e8 77 00	
0x004FFD14	00 00	

vbptr指的是虚基类表指针 (virtual base table pointer)，该指针指向了一个虚基类表 (virtual table)，虚表中记录了虚基类与本类的偏移地址；通过偏移地址，这样就找到了虚基类成员，而虚继承也不用像普通多继承那样维持着公共基类 (虚基类) 的两份同样的拷贝，节省了存储空间。