x64逆向：类

类

在x64中，类的识别同x86一样。主要依据依然是构造、析构与虚表

在高版本下，可执行文件中记录了类名的名称粉碎，因为要使用`RTTI`机制

构造和析构识别的特点：

- 构造传入参数`this`指针，并作为返回值返回
- 构造和析构都有填写虚表指针`vtptr`的操作
- 构造是对象在生命期中第一个调用的函数，析构是对象在生命期最后一个调用的函数

虚表

虚表特性：

- 如果一个类至少有一个虚函数，则就有一个指向虚表的指针`vtptr`
- 不同的类虚表不同，相同的类对象共享一个虚表
- 虚表指针`vtptr`存放在对象首地址处
- 虚表地址在全局数据区
- 虚表的每个元素都指向一个类成员函数指针
- 虚表不一定以0结尾
- 虚表的成员函数顺序一般按照类声明的顺序排列（例外情况：函数重载）
- 虚表指针`vtptr`在构造函数中会被初始化
- 虚表指针`vtptr`在析构函数中会被赋值
- 子类的虚表会复制基类的虚表
    - 如果子类虚函数中有覆盖基类虚函数，则使用子类虚函数的地址覆盖对应表项
    - 如果子类有新增虚函数，则将其放在虚表后面

继承

继承关系中构造和析构行为总结：

- 构造函数
    1. 调用虚基类构造（多个按继承顺序调用）
    2. 调用普通基类构造（多个按继承顺序调用）
    3. 调用成员对象构造（多个按定义顺序调用）
    4. 调用子类构造
- 析构函数
    1. 调用子类析构
    2. 调用成员对象析构

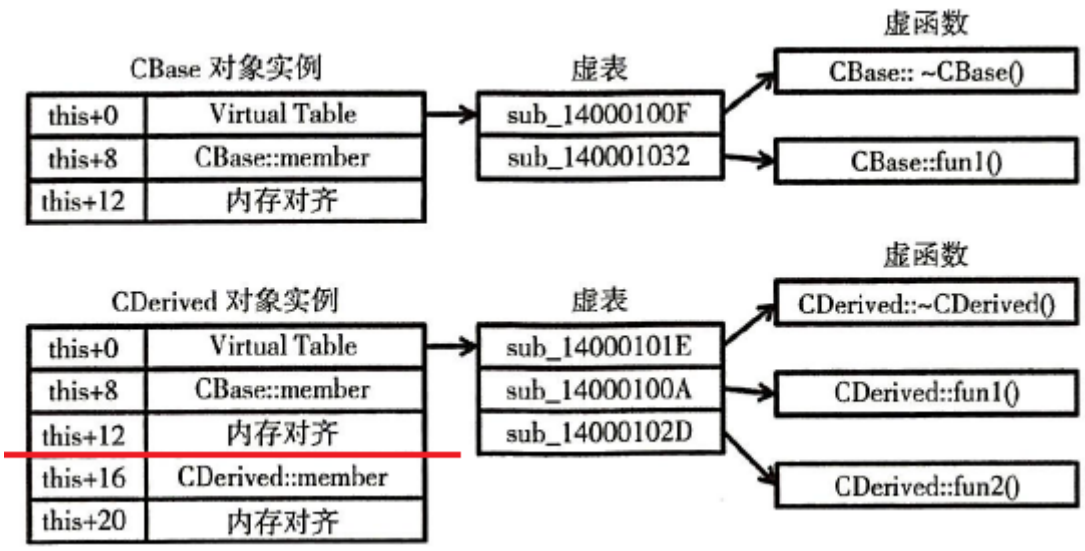
3. 调用普通基类析构
4. 调用虚基类析构

若存在虚表，则在构造基类和子类、析构基类和子类时，都会填写虚表指针`vtptr`指向当前类的虚表

单继承

```
// 基类CBase
class CBase {
public:
    CBase();
    virtual ~CBase();
    virtual void fun1();
private:
    int m_nMember;
};

// 子类CDerived，继承于CBase
class CDerived : public CBase {
public:
    CDerived();
    virtual ~CDerived();
    virtual void fun1() override;   // 重写fun1
    virtual void fun2();
private:
    int m_nMember;
};
```

对象内存分布：

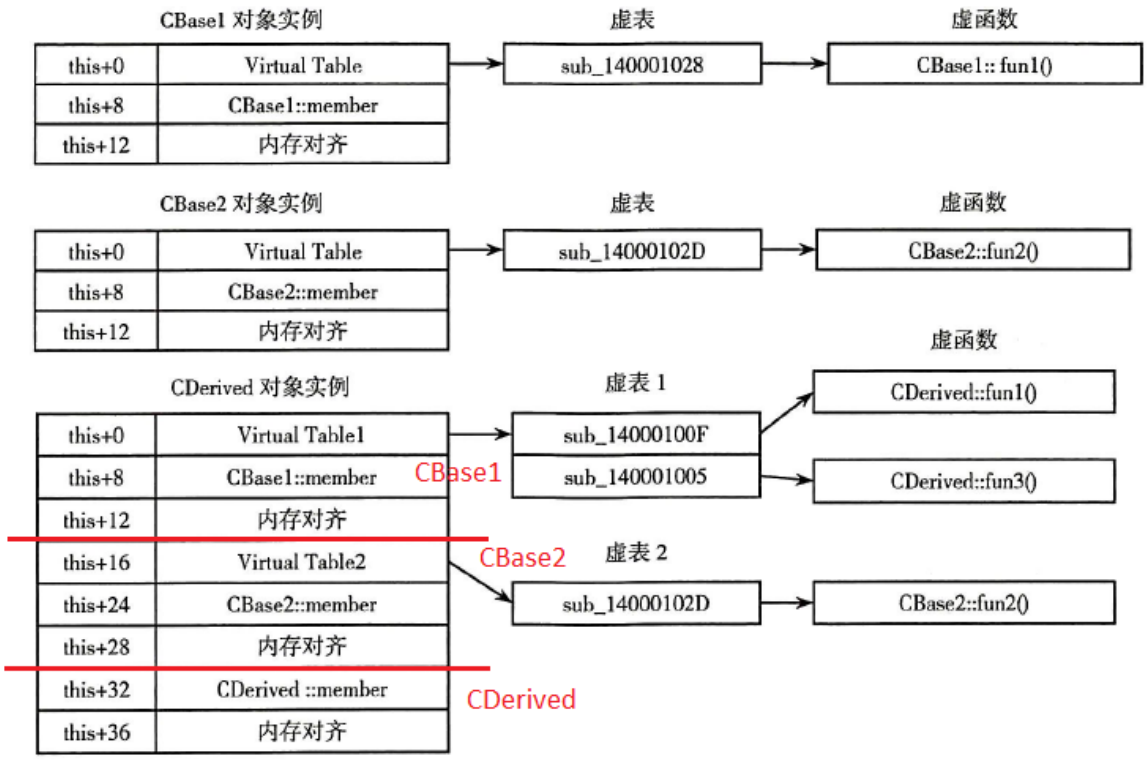


多继承

```
// 基类CBase1
class CBase1 {
public:
    CBase1();
    virtual ~CBase1();
```

```
    virtual void fun1();
private:
    int m_nMember;
};

// 基类CBase2
class CBase2 {
public:
    CBase2();
    virtual ~CBase2();
    virtual void fun2();
private:
    int m_nMember;
};

// 子类CDerived，继承于CBase1、CBase2
class CDerived : public CBase1, public CBase2 {
public:
    CDerived();
    virtual ~CDerived();
    virtual void fun1() override;   // 重写fun1
    virtual void fun3();
private:
    int m_nMember;
};
```

对象内存分布：

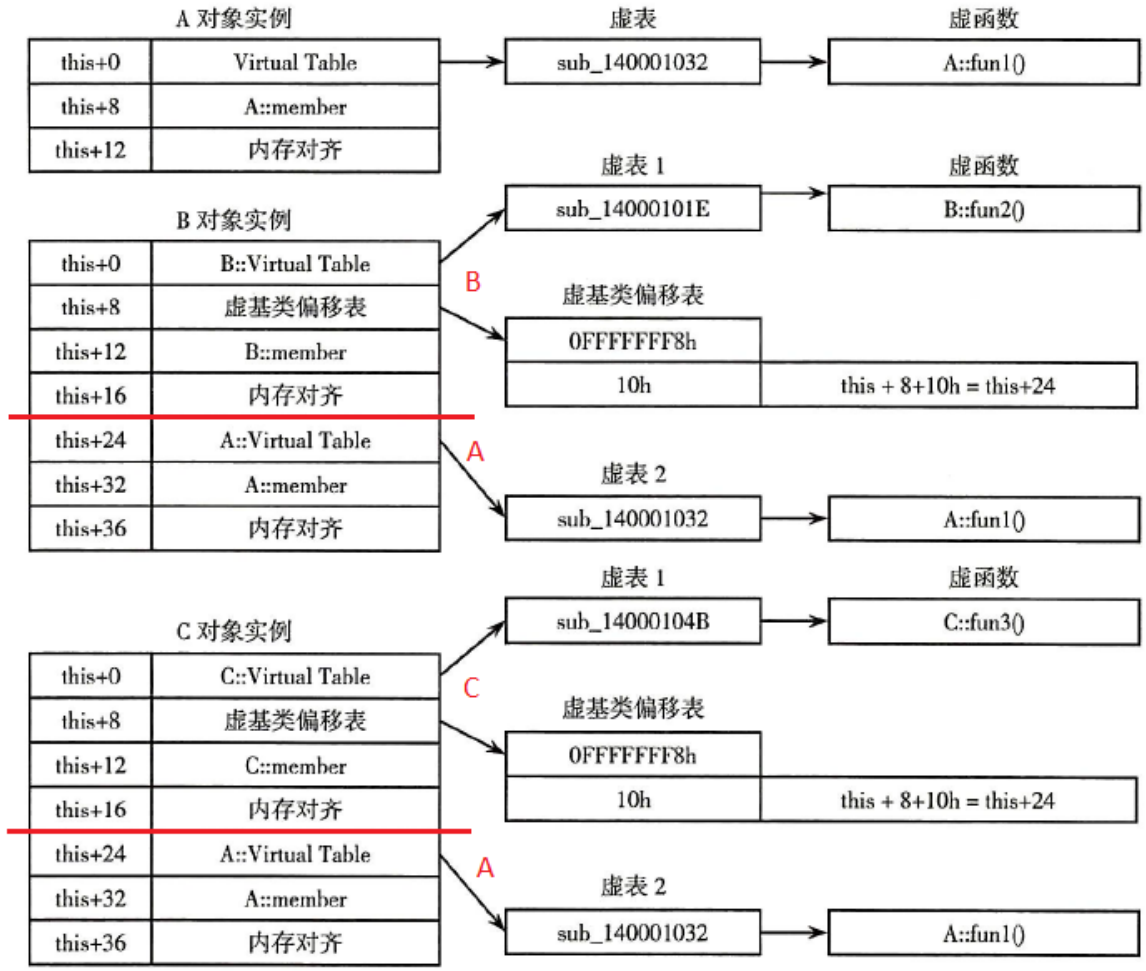


菱形继承

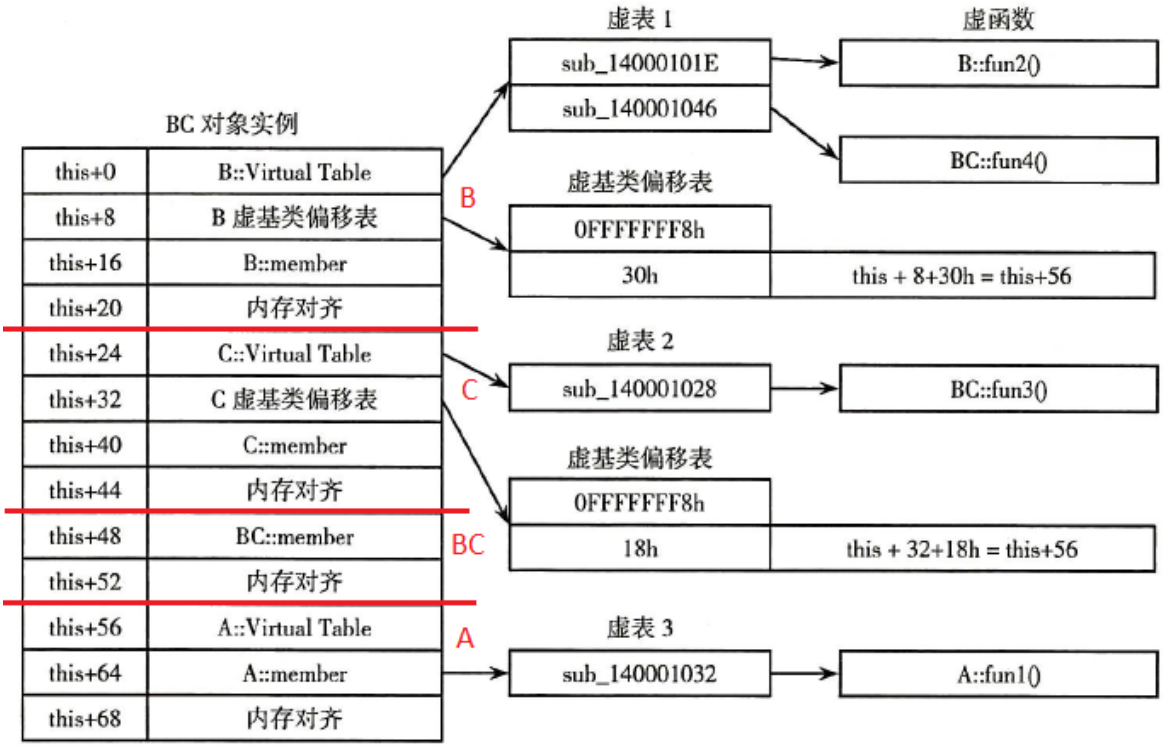
```
// 基类A
class A {
public:
```

```
    A();
    virtual ~A();
    virtual void fun1();

private:
    int m_nMember;
};

// 基类B，虚继承于A
class B : public virtual A {
public:
    B();
    virtual ~B();
    virtual void fun2();

private:
    int m_nMember;
};

// 基类C，虚继承于A
class C : public virtual A {
public:
    C();
    virtual ~C();
    virtual void fun3();

private:
    int m_nMember;
};

// 子类BC，继承于B、C
class BC : public B, public C {
public:
    BC();
    virtual ~BC();
    virtual void fun3() override;   // 重写fun3
    virtual void fun4();

private:
    int m_nMember;
};
```

基类对象内存分布：



子类对象内存分布：



虚继承的识别特点：

- 构造基类的时候会传参，表示是否需要构造虚基类
- 存在虚基类偏移表
  - 第一个成员未发现有明显用途

    - 第二个成员表示此虚基类偏移表对虚基类的偏移量

抽象类

有纯虚函数的类则是抽象类，不能实例化对象。

抽象类的识别特点：

- 抽象类必定有纯虚函数
- 纯虚函数在虚表中会被填写为`_purecall`函数，此函数用于报错并退出程序