单重虚继承无虚函数

我们用三个类来描述这个问题，
基类，家具类`CFurniture`，
派生类，沙发类`CSofa`和床类`CBed`，

虽然沙发和床确实有一些共性的地方，这里我们仅仅做演示。

代码：

```
// 2. 单重虚继承无虚函数.cpp : 定义控制台应用程序的入口点。
//

#include "stdafx.h"
#include <iostream>
using namespace std;

// 家具类
class CFurniture {
public:
    CFurniture() {
        printf("CFurniture::CFurniture()\r\n");
        m_nFurniture = 3;
        //m_nFurniture2 = 4;
    }

    ~CFurniture() {
        printf("CFurniture::~CFurniture()\r\n");
        m_nFurniture = 0;
    }

    int m_nFurniture;
    //int m_nFurniture2;
};

// 沙发类，虚继承
class CSofa :virtual public CFurniture {
public:
    CSofa() {
        printf("CSofa::CSofa()\r\n");
```

```
        m_nSofa = 1;
    }

    ~CSofa() {
        printf("CSofa::~CSofa()\r\n");
        m_nSofa = 0;
    }

    void sit() {
        printf("CSofa::sit()\r\n");
    }

private:
    int m_nSofa;
};

// 床类，虚继承
class CBed :virtual public CFurniture {
public:
    CBed() {
        printf("CBed::CBed()\r\n");
        m_nBed = 2;
    }

    ~CBed() {
        printf("CBed::~CBed()\r\n");
        m_nBed = 0;
    }

    void sleep() {
        printf("CBed::sleep()\r\n");
    }

private:
    int m_nBed;
};

int main()
{
    // 实例化对象
    CFurniture furniture;
    CSofa sofa;
```

```
    CBed bed;

    // 对象大小
    int nSizeFurniture = sizeof(CFurniture);
    cout << nSizeFurniture << endl; // 4
    int nSizeSofa = sizeof(CSofa);
    cout << nSizeSofa << endl; // 12
    int nSizeBed = sizeof(bed);
    cout << nSizeBed << endl; // 12

    return 0;
}
```

对象实例化

```
    CFurniture furniture;
    CSofa sofa;
    CBed bed;
```

构造，
基类，`CFurniture furniture` 访问自身构造函数`CFurniture()`
派生类，`CSofa sofa`访问自身构造函数，向上查找父类，访问父类的构造函数，执行完毕后，再回到自身访问构造函数。
`CBed bed`同理。

程序执行完毕，析构：
析构对象`CBed ~CBed()`，析构父类对象`CPerson ~CPerson()`，
析构对象`CSofa ~CSofa()`，析构父类对象`CPerson ~CPerson()`，
析构父类对象`CPerson ~CPerson()`

对象大小

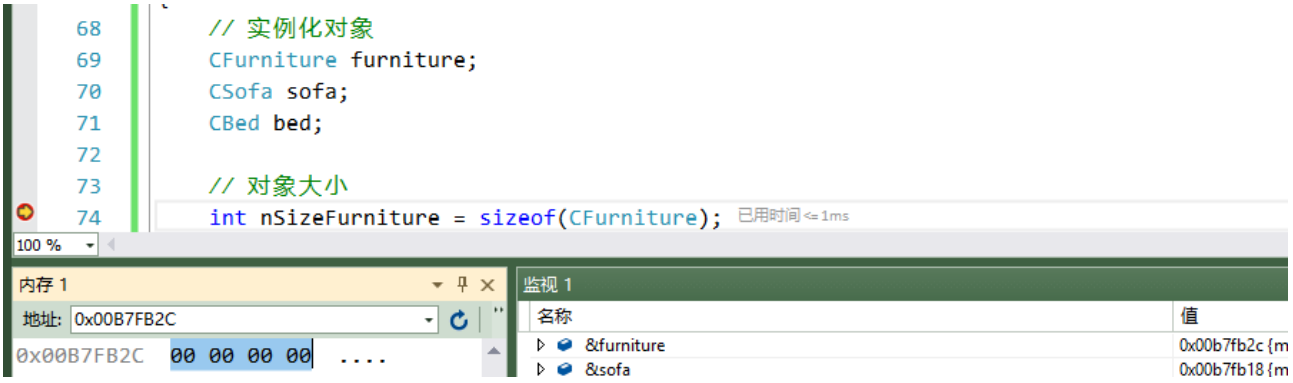
```
// 对象大小
    int nSizeFurniture = sizeof(CFurniture);
    cout << nSizeFurniture << endl; // 4
```

```
    int nSizeSofa = sizeof(CSofa);
    cout << nSizeSofa << endl; // 12
    int nSizeBed = sizeof(bed);
    cout << nSizeBed << endl; // 12
```

基类，`CFurniture`有一个`int`型成员变量`m_nFurniture`，那对象的大小`sizeof(nSizeFurniture)`等于`4`个字节大小。
派生类，
`CSofa`自身有一个`int`型成员变量`m_nSofa`，大小为`4bytes`，
`CSofa`类继承自`CFurniture`类，
所以`CSofa`类还有一个基类的`int`类型的成员变量`m_nFurniture`，
那这里应该是`8bytes`啊，咋就是`12bytes`呢?
`CBed`类同`CSofa`，
让我们来看看`虚继承`是怎么一回事儿。

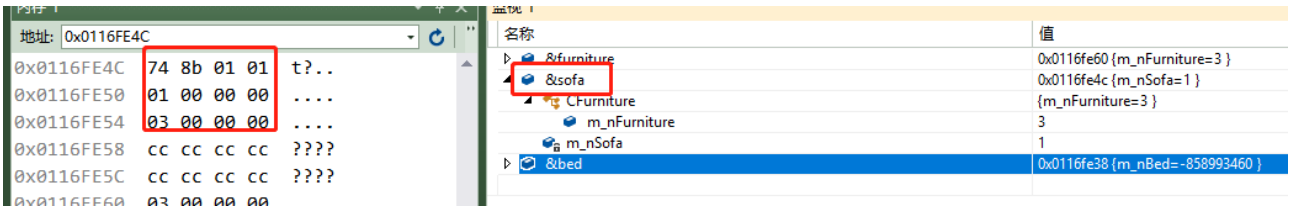
对象内存分布

`CFurniture furniture`对象内存

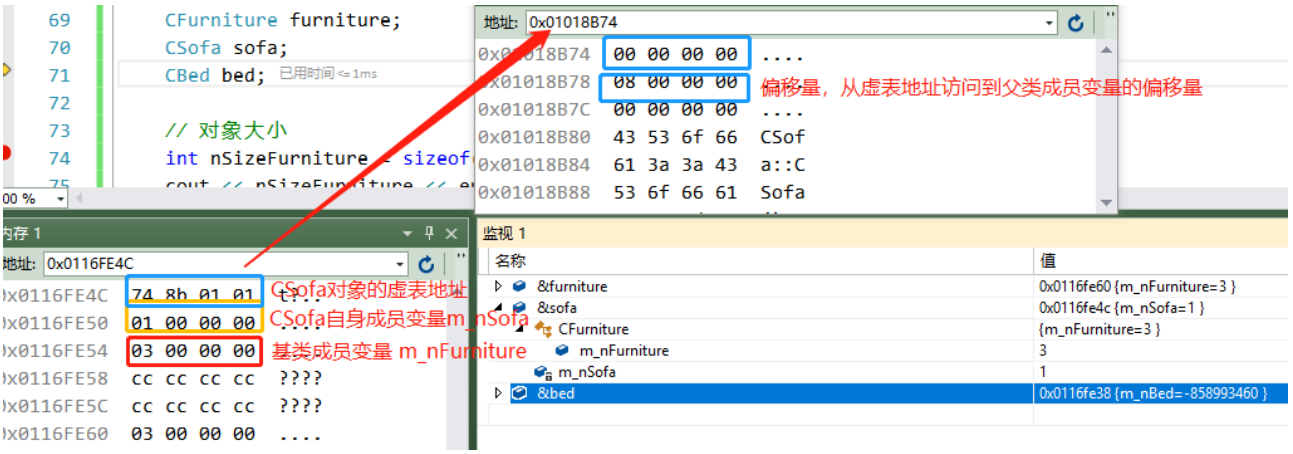


`CFurniture furniture;`
对象的`int`类型的成员变量`m_nFurniture`被初始化为了`0`

`CSofa sofa;`



我们来着重分析下这里面的内存都表示的什么。



看到这里明白了，为什么是 `12bytes` 了吧。

偏移量：

`0x0116FE54` 这个位置是父类成员变量在当前类中的内存位置。

`0x0116FE4C` 是当前类的成员变量在内存的位置。

`0x0116FE54` - `0x0116FE4C` = 8(HEX)，就是 `08 00 00 00` 了。

得到的就是，当前类对象地址到访问父类成员在当前类对象中的成员变量的偏移量。

也就是说，需要偏移多少才可以访问到父类成员变量。

对象组成：

给父类 `CFurniture` 再加一个成员变量 `m_nFurniture2` 来帮助分析。