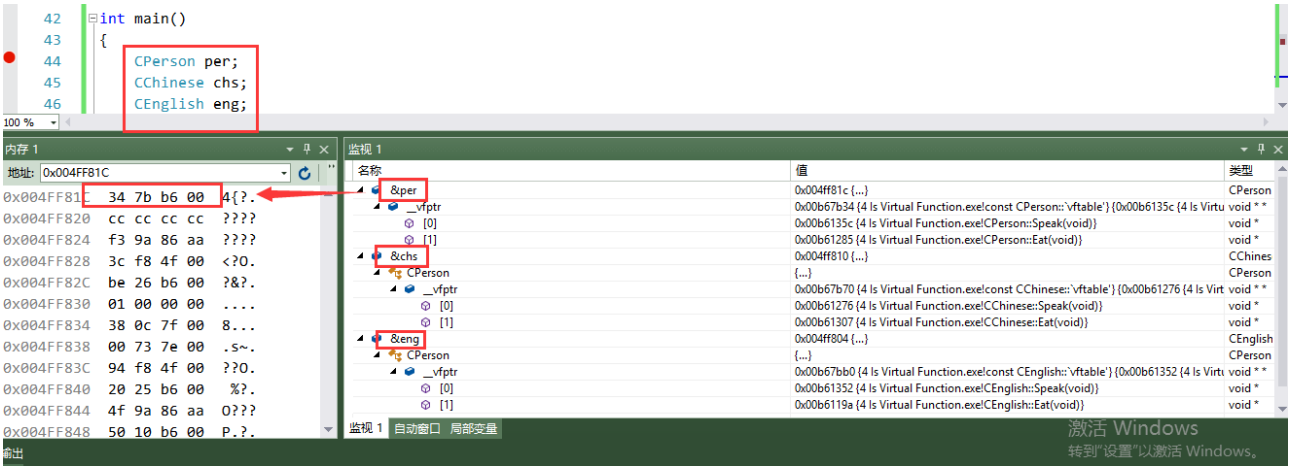
virtual 示例

```
/*
4.编写例子，结合内存的实际布局来说明虚函数的原理。
*/

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

//面向对象：多态-虚函数

class CPerson {
public:
    virtual void Speak() {
        printf("CPerson::Speak()\r\n");
    }

    virtual void Eat() {
        printf("CPerson::Eat()\r\n");
    }
};

class CChinese :public CPerson {
public:
    void Speak() {
        printf("CChinese::Speak()\r\n");
    }
    void Eat() {
        printf("CChinese::Eat()\r\n");
    }
};

class CEnglish :public CPerson {
public:
    void Speak() {
        printf("CEnglish::Speak()\r\n");
    }
    void Eat() {
```

```
        printf("CEnglish::Eat()\r\n");
    }
};

int main()
{
    CPerson per;
    CChinese chs;
    CEnglish eng;

    int nPersonSize = sizeof(CPerson);
    cout << nPersonSize << endl;
    int nChsSize = sizeof(CChinese);
    cout << nChsSize << endl;
    int nEngSize = sizeof(CEnglish);
    cout << nEngSize << endl;

    CPerson* ary[2] = { &chs, &eng };
    for (int i = 0; i < 2; i++) {
        ary[i]->Speak();
    }

    return 0;
}

/*
output:
4
4
4
CChinese::Speak()
CEnglish::Speak()
*/
/*
output:
4
4
4
CChinese::Speak()
CEnglish::Speak()
*/
```

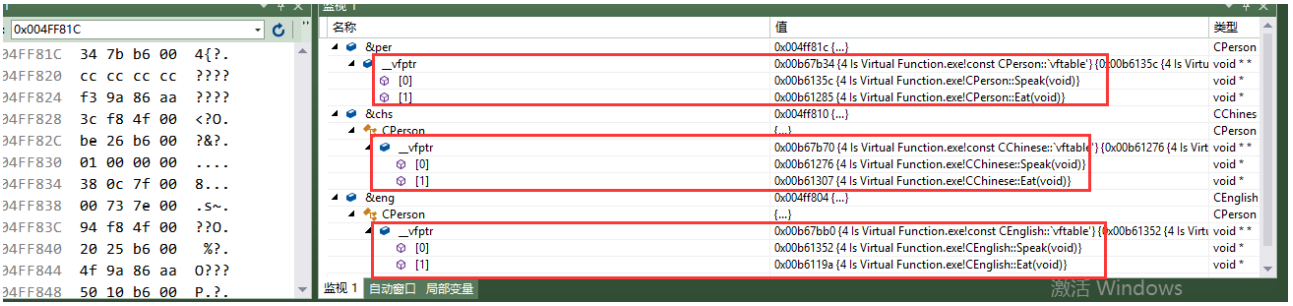
virtual 原理分析



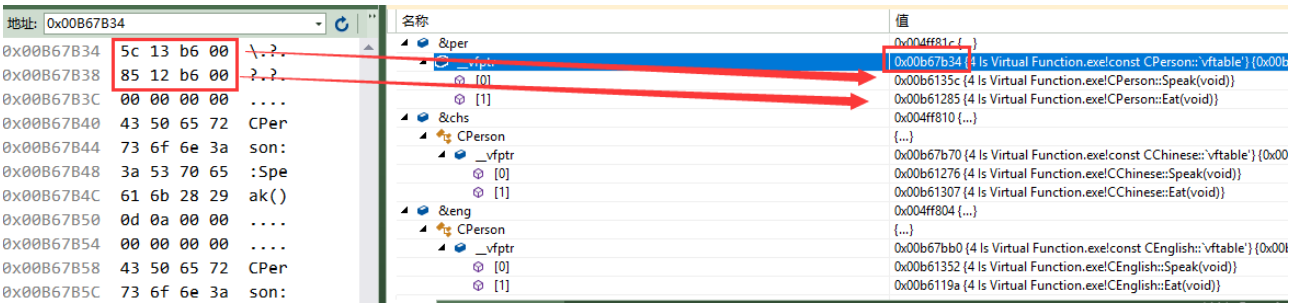
查看内在，观察到，`&per` 对象在内在中记录了一个地址，那这个地址是什么呢？

保存的是类中所有的虚函数在内在中的地址的指针，

那所有的虚函数在一起就需要一张虚拟表来保存这些虚函数的地址，



这些就是虚函数的地址了。



就是对应的这个地址了。

那怎么验证呢？我们来模拟实现虚函数virtual。

代码见课件 `Github CR32/c++/12virtual/Test/TestVirtual/Person.h`