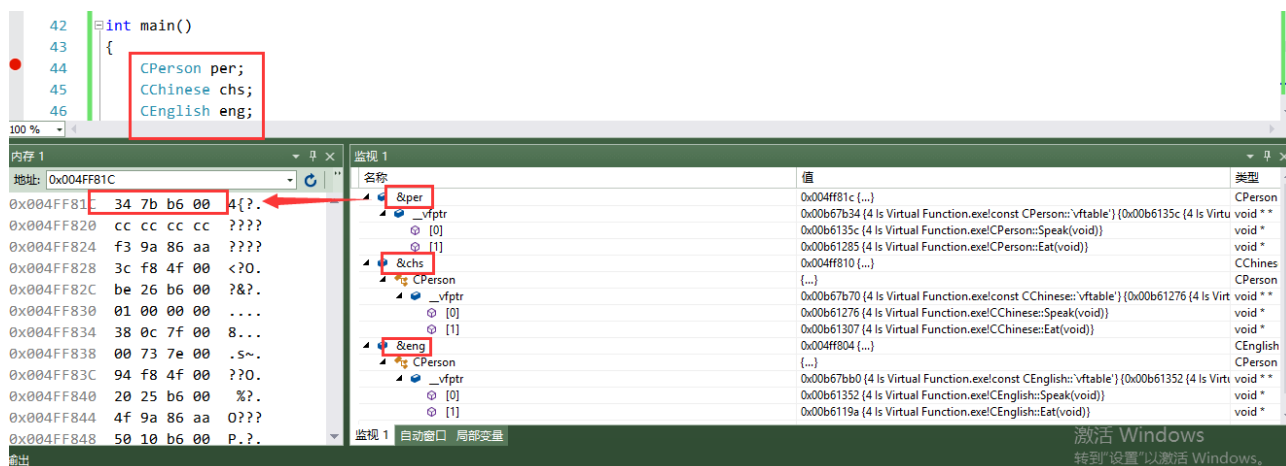


virtual 示例

```
1  /*
2  4.编写例子，结合内存的实际布局来说明虚函数的原理。
3  */
4
5  #include "stdafx.h"
6  #include <iostream>
7  using namespace std;
8
9  //面向对象：多态-虚函数
10
11 class CPerson {
12 public:
13     virtual void Speak() {
14         printf("CPerson::Speak()\r\n");
15     }
16
17     virtual void Eat() {
18         printf("CPerson::Eat()\r\n");
19     }
20 };
21
22 class CChinese :public CPerson {
23 public:
24     void Speak() {
25         printf("CChinese::Speak()\r\n");
26     }
27     void Eat() {
28         printf("CChinese::Eat()\r\n");
29     }
30 };
31
32 class CEnglish :public CPerson {
33 public:
34     void Speak() {
35         printf("CEnglish::Speak()\r\n");
36     }
37     void Eat() {
```

```
38         printf("CEnglish::Eat()\r\n");
39     }
40 };
41
42 int main()
43 {
44     CPerson per;
45     CChinese chs;
46     CEnglish eng;
47
48     int nPersonSize = sizeof(CPerson);
49     cout << nPersonSize << endl;
50     int nChsSize = sizeof(CChinese);
51     cout << nChsSize << endl;
52     int nEngSize = sizeof(CEnglish);
53     cout << nEngSize << endl;
54
55     CPerson* ary[2] = { &chs, &eng };
56     for (int i = 0; i < 2; i++) {
57         ary[i]->Speak();
58     }
59
60     return 0;
61 }
62
63 /*
64 output:
65 4
66 4
67 4
68 CChinese::Speak()
69 CEnglish::Speak()
70 */
71 /*
72 output:
73 4
74 4
75 4
76 CChinese::Speak()
77 CEnglish::Speak()
78 */
```

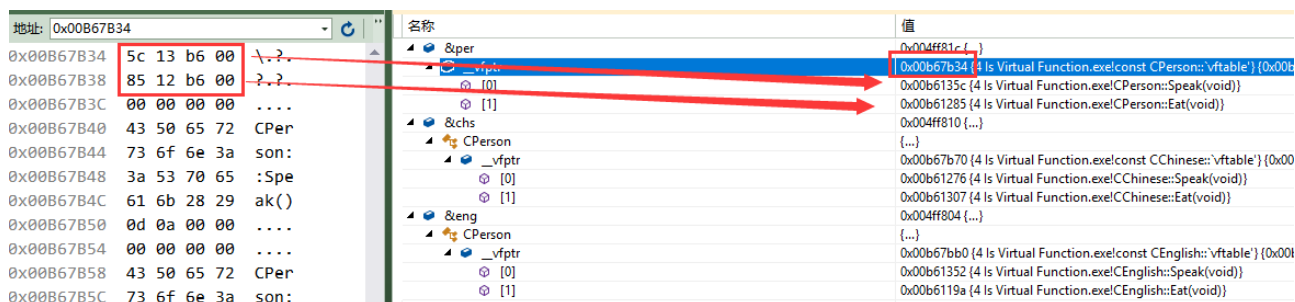
virtual 原理分析



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`