单重继承无虚函数

单重继承

class [类名]: [继承权限] [类名]

class CChinese : public CPerson

```
1
2
  1. 单重继承无虚函数
3
  分析从几个方面入手:
4
  1. 查看对象大小
5
  2. 对象内存分布
  3. 成员调用关系
7
8
  */
9
10 #include "stdafx.h"
11
12 class CPerson {
13 public:
       CPerson() {
14
15
           m_p = 0;
       }
16
17
18
       void speak() {
           printf("CPerson::speak(人话)\r\n");
19
       }
20
21
22 public:
23
       int m_p;
24 };
25
26 class CChinese:public CPerson {
   public:
27
       CChinese() {
28
29
           m_c = 1;
30
       }
31
32
       void speak() {
           printf("CChinese::speak(汉语)\r\n");
33
```

```
34
       }
35
36 public:
37
       int m_c;
38 };
39
40
   int main()
   {
41
       // 对象大小
42
       int nPerSize = sizeof(CPerson); //4
43
       int nChsSize = sizeof(CChinese); //8
44
45
46
       // 对象内存分布
47
       CPerson per;
48
       CChinese chs;
49
       // 调用关系
50
       per.speak(); // 直接调用
51
       chs.speak(); // 直接调用
52
53
       // 向上转型
54
       // 1 对象赋值
55
       per = chs;
56
       per.speak(); // 直接调用
57
58
       // 2 指针赋值
59
60
       CPerson* pPer = NULL;
       pPer = &chs;
61
       pPer->speak(); // 直接调用
62
63
       // 3 引用赋值
64
65
       CPerson &rPer = per;
66
       rPer = chs;
       rPer.speak(); // 直接调用
67
68
69
       return 0;
70 }
```

```
1 output:
2 CPerson::speak(人话)
```

```
CChinese::speak(汉语)
CPerson::speak(人话)
CPerson::speak(人话)
CPerson::speak(人话)
```

对象大小

CPerson 对象中有 int 型成员变量 m_p , 会在内存中分配 4bytes 内存空间。
CChinese 对象以 public 权限继承 CPerson 对象, 此时 CChinese 对象中有两个成员变量。

- 一个是从 CPerson 对象继承的 int 型成员变量 m p (基类成员变量),
- 一个是自身 CChinese 对象的 int 型成员变量 m c (自身成员变量)。

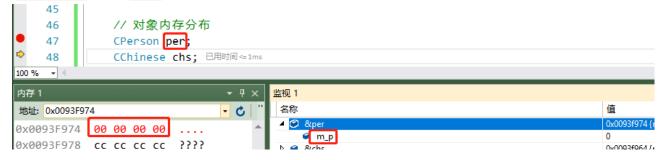
```
int nPerSize = sizeof(CPerson); //4
int nChsSize = sizeof(CChinese); //8
```

通过验证得知对应的对象大小如上例代码所示。

对象内存分布

了解了对象的大小,还需要观察对象在内存的分布。

CPerson 对象 per 的内存分布:



CChinese 对象 chs 的内存分布:

```
CChinese chs;
    49
               // 调用关系
    50
100 %
                                         监视 1
                                 - 0
地址: 0x0093F964
                                                                                              0x0093f974 {m_p=0 }
0x0093F964 00 00 00 00
                                             m_p
0x0093F968 01 00 00 00
0x0093F96C cc cc cc
                                                                                              {m_p=
                                              🐴 m_c
0x0093F970 cc cc cc cc ?????
```

CChinese 的自身成员变量 m_c , 排列在基类 CPerson 对象成员变量的后面。 也可以简单的理解为, 按照 就近原则 入栈。

CChinese 自身成员变量按顺序依次入栈,然后将继承来的基类成员变量依次入栈到自身对象内存分布中。

拿本例来说(小端)入栈:

先入栈 CChinese 对象自身成员变量的 m_c , 再入栈基类 CPerson 对象的成员变量 m_p , 那观察到的顺序就是上图的内存结构了。

但是,这条总结并不一定适用所有情况。

成员调用关系

见代码注释部分