

示例代码:

```
1  // 5_2 菱形继承无虚继承有虚函数.cpp
2  //
3
4  #include "stdafx.h"
5  #include <iostream>
6  using namespace std;
7
8  class A {
9  public:
10     A() {
11         _a = 1;
12     };
13
14     virtual void funA() {
15         printf("A::funA()\r\n");
16     }
17
18 public:
19     int _a;
20 };
21
22 class B : public A {
23 public:
24     B() {
25         _b = 2;
26     }
27
28     virtual void funB() {
29         printf("B::funB()\r\n");
30     }
31 public:
32     int _b;
33 };
34
35 class C : public A {
36 public:
37     C() {
38         _c = 3;
```

```
39     }
40
41     virtual void funC() {
42         printf("C::funC()\r\n");
43     }
44 public:
45     int _c;
46 };
47
48 class D :public B, public C {
49 public:
50     D() {
51         _d = 4;
52     }
53
54     virtual void funA() {
55         printf("D::funA()\r\n");
56     }
57
58     virtual void funB() {
59         printf("D::funB()\r\n");
60     }
61
62     virtual void funC() {
63         printf("D::funC()\r\n");
64     }
65
66 public:
67     int _d;
68 };
69
70 int main()
71 {
72     A a;
73     B b;
74     C c;
75     D d;
76
77     d.B::_a = 10; // 通过指定是哪个类中的_a来消除二义性的目的
78     cout << d.B::_a << endl;
79
80     // 函数调用
```

```
81 // 将派生类对象赋值给基类对象
82 // a->b a->c
83 a = b;
84 a.funA(); // A::funA()
85 a = c;
86 a.funA(); // A::funA()
87 // b->d c->d
88 b = d;
89 b.funA(); // A::funA()
90 b.funB(); // B::funB()
91 c = d;
92 c.funA(); // A::funA()
93 c.funC(); // C::funC()
94 printf("\r\n");
95
96 // 将派生类指针赋值给基类指针
97 // a->b a->c
98 A* pa = &b;
99 pa->funA(); // A::funA()
100 A* pa2 = &c;
101 pa2->funA(); // A::funA()
102 // b->d c->d
103 B* pb = &d;
104 pb->funA(); // D::funA()
105 pb->funB(); // D::funB()
106 B* pb2 = &d;
107 pb2->funA(); // D::funA()
108 pb2->funB(); // D::funB()
109 printf("\r\n");
110
111 // 将派生类引用赋值给基类引用
112 // a->b a->c
113 A &ra = b;
114 ra.funA(); // A::funA()
115 A &ra2 = c;
116 ra2.funA(); // A::funA()
117 // b->d c->d
118 B &rb = d;
119 rb.funA(); // D::funA()
120 rb.funB(); // D::funB()
121 B &rb2 = d;
122 rb2.funA(); // D::funA()
```

```
123     rb2.funB(); // D::funB()
124
125     return 0;
126 }
127 /*
128 output:
129
130 10
131 A::funA()
132 A::funA()
133 A::funA()
134 B::funB()
135 A::funA()
136 C::funC()
137
138 A::funA()
139 A::funA()
140 D::funA()
141 D::funB()
142 D::funA()
143 D::funB()
144
145 A::funA()
146 A::funA()
147 D::funA()
148 D::funB()
149 D::funA()
150 D::funB()
151 */
```

对象内存分析

A a

内存 2

地址: 0x00F07B34

0x00F07B34	f9 11 f0 00	?.?
0x00F07B38	0d 0a 00 00
0x00F07B3C	41 3a 3a 66	A::f
0x00F07B40	75 6e 41 28	unA(
0x00F07B44	29 0d 0a 00)...
0x00F07B48	00 00 00 00

虚函数表

内存 1

地址: 0x00ECFC74

x00ECFC74	34 7b f0 00	4{?. 虚表指针
x00ECFC78	01 00 00 00 A
x00ECFC7C	cc cc cc cc	????
x00ECFC80	4d 5e 5d 3e	M^]>
x00ECFC84	98 fc ec 00	???.
x00ECFC88	9e 26 f0 00	?&?.
x00ECFC8C	a1 aa aa aa

监视 1

名称	值
&d	0x00ecfc28 {_d=4 }
&a	0x00ecfc74 {_a=1 }
&b	0x00ecfc60 {_b=2 }
&c	0x00ecfc4c {_c=3 }

B b

内存 2

地址: 0x00F07B50

0x00F07B50	f9 11 f0 00	?.?
0x00F07B54	27 11 f0 00	'?.'
0x00F07B58	00 00 00 00
0x00F07B5C	42 3a 3a 66	B::f
0x00F07B60	75 6e 42 28	unB(
0x00F07B64	29 0d 0a 00)...

虚函数表

内存 1

地址: 0x00ECFC60

:00ECFC60	50 7b f0 00	P{?. 虚函数表指针
:00ECFC64	01 00 00 00	... A
:00ECFC68	02 00 00 00	... B
:00ECFC6C	cc cc cc cc	????
:00ECFC70	cc cc cc cc	????
:00ECFC74	34 7b f0 00	4{?.
:00ECFC78	01 00 00 00
:00ECFC7C	cc cc cc cc	????

监视 1

名称	值
&d	0x00ecfc28 {_d=4 }
&a	0x00ecfc74 {_a=1 }
&b	0x00ecfc60 {_b=2 }
&c	0x00ecfc4c {_c=3 }

C c

