

示例代码:

```
1  // 5_3 菱形继承有虚继承有虚函数.cpp : 定义控制台应用程序的入口点。
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 : virtual 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 : virtual 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 : virtual public B, virtual 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

B b

C c

D d

内存 1

地址: 0x0076F758

0x0076F758	d8 7b 39 01	?{9.	d	偏基类, 指向偏移对象 a 地址的指针, 表中存储的内容是偏移量 64 - 58 = 0c
0x0076F75C	04 00 00 00	d 成员变量	
0x0076F760	00 00 00 00	a	指向 a 对象虚函数表地址的指针
0x0076F764	b8 7b 39 01	?{9.	a 成员变量	
0x0076F768	01 00 00 00	b	指向 b 对象虚函数表地址的指针
0x0076F76C	00 00 00 00	b 成员变量	
0x0076F770	c4 7b 39 01	?{9.	c	指向 c 对象虚函数表地址的指针
0x0076F774	ec 7b 39 01	?{9.	偏基类, 指向偏移对象 b 地址的指针, 表中存储的内容是偏移量 74 + fffffff0 = 64	
0x0076F778	02 00 00 00	c 成员变量	
0x0076F77C	00 00 00 00	D	
0x0076F780	d0 7b 39 01	?{9.	偏基类, 指向偏移对象 c 地址的指针, 表中存储的内容是偏移量 84 + fffffff0 = 64	
0x0076F784	f8 7b 39 01	?{9.		
0x0076F788	03 00 00 00		
0x0076F78C	cc cc cc cc	????		
0x0076F790	cc cc cc cc	????		
0x0076F794	84 7b 39 01	?{9.		
0x0076F798	98 7b 39 01	?{9.		
0x0076F79C	03 00 00 00		

监视 1

名称

值

名称	值
0x0076F758	0x0076F758
0x0076F75C	0x0076F75C
0x0076F760	0x0076F760
0x0076F764	0x0076F764
0x0076F768	0x0076F768
0x0076F76C	0x0076F76C
0x0076F770	0x0076F770
0x0076F774	0x0076F774
0x0076F778	0x0076F778
0x0076F77C	0x0076F77C
0x0076F780	0x0076F780
0x0076F784	0x0076F784
0x0076F788	0x0076F788
0x0076F78C	0x0076F78C
0x0076F790	0x0076F790
0x0076F794	0x0076F794
0x0076F798	0x0076F798
0x0076F79C	0x0076F79C

函数调用

参考虚函数调用。