

派生类

调用顺序

```
1 // 3 A B C extends.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 CTop
9 {
10 private:
11     int m_a;
12
13 public:
14     CTop(int a);
15     ~CTop();
16 };
17
18 class CMiddle:public CTop
19 {
20 private:
21     int m_b;
22
23 public:
24     CMiddle(int a, int b);
25     ~CMiddle();
26 };
27
28 class CBottom:public CMiddle
29 {
30 private:
31     int m_c;
32
33 public:
34     CBottom(int a, int b, int c);
35     ~CBottom();
```

```
36 };
37
38 CTop::CTop(int a = 0)
39 {
40     this->m_a = a;
41     cout << "a.." << m_a << endl;
42     cout << "CTop constructed\r\n";
43 }
44
45 CTop::~CTop()
46 {
47     this->m_a = 0;
48     cout << "a.." << m_a << endl;
49     cout << "~CTop deconstructed\r\n";
50 }
51
52 CMiddle::CMiddle(int a = 0, int b = 0) :CTop(a) //初始化成员列表,
    执行CTop类中的构造函数
53 {
54     this->m_b = b;
55     cout << "b.." << m_b << endl;
56     cout << "CMiddle constructed\r\n";
57 }
58
59 CMiddle::~CMiddle()
60 {
61     this->m_b = 0;
62     cout << "b.." << m_b << endl;
63     cout << "~CMiddle constructed\r\n";
64 }
65
66 CBottom::CBottom(int a, int b, int c):CMiddle(a,b)//初始化成员列
    表, 执行CMiddle类中的构造函数
67 {
68     this->m_c = c;
69     cout << "c.." << m_c << endl;
70     cout << "CBottom constructed\r\n";
71 }
72
73 CBottom::~CBottom()
74 {
75     this->m_c = 0;
```

```

76     cout << "c.." << m_c << endl;
77     cout << "~CBottom constructed\r\n";
78 }
79
80
81 int main()
82 {
83     CBottom bottom(1, 2, 3);
84
85     return 0;
86 }
87 /*
88 output:
89
90 a..1
91 CTop constructed
92 b..2
93 CMiddle constructed
94 c..3
95 CBottom constructed
96 c..0
97 ~CBottom constructed
98 b..0
99 ~CMiddle constructed
100 a..0
101 ~CTop deconstructed
102 */
103
104 /*
105 创建类对象时，应先调用其构造函数。
106 但是如果这个类有成员对象，则要先执行成员对象自己所属类的构造函数，
107 当全部成员对象都执行了自身类的构造函数后，再执行当前类的构造函数。
108
109 调用顺序，
110 构造函数
111 CTop -> CMiddle -> CBottom
112
113 析构函数
114 CBottom -> CMiddle -> CTop
115 */

```

参考资料

- 派生类及调用顺序

<https://blog.csdn.net/rhzwani23/article/details/2105205>