|  |  |
| --- | --- |
| **编号:** | **1-2** |

****

信息科学与工程学院实验报告

《面向对象程序设计》

**Object-Oriented Programming**

|  |  |
| --- | --- |
| 姓名： | 肖威 |
| 学号： | 201911990116 |
| 班级： | 计师本1901 |
| 时间： | 2020年11月24日 |

**《面向对象程序设计》实验报告**

**基本要求：**请围绕实验目的、实验内容、实验过程、实验结果（附图）、实验总结（重点阐述）五个部分进行撰写。若报告中若涉及源代码内容，请在附录部分提供完整源码及GitHub源码托管地址。报告撰写完毕后请提交PDF格式版本到云班课。

1. **实验目的**

理解类的三种不同关系（组合，依赖，继承）

掌握复合类构造函数、析构函数的定义方法与使用方法

熟练掌握类继承的定义方式（单继承，多继承）

理解三种不同继承方式间的区别（公有，私有，保护）

掌握派生类同名覆盖原理及相应同名冲突解决方法

掌握赋值兼容性基本原理（左公 = 右派）

熟练掌握复杂类的设计方法（三构一析+普函）

1. **实验内容**

【任务1】设计一个基类base，其内含有数据成员（public：int a， protected： int b，private： int c，public：static int count）和函数成员（输出类的数据成员函数print（），统计类对象个数函数 static int statistic（）），然后请采用三种不同的继承方式由base类分别派生出三个子类：derived1，derived2，derived3，请根据上述基类和派生类尝试编程论证下面的三个问题。（可参考课堂演示程序）

派生类全盘接受基类的所有本类成员，其中包括基类的普通公有成员，保护成员和私有成员。

根据继承类数据成员能否在类内或类外被访问的问题，探索分析三种不同继承方式各自的特点（参考课程ppt）。

派生类对象被建立时派生类是如何调用构造函数的，给出构造函数调用的次序，析构函数析构次序，并分析其中规律。

【任务2】定义一个二维空间点类Location，采用数据成员x，y表示该类对象在二维坐标系中的坐标位置，类中函数成员函数move（）可以实现移动该类对象的坐标位置，show（）函数可以输出当前类对象的信息。然后，以Location为基类，派生出三维空间坐标点类Point，接着，再利用三维空间点类Point派生出一个三维空间下的球体类Sphere,定义Point点类和球体类Sphere中各自特有的move函数和show()函数。要求设计并实现上述类，并在主函数中定义各个类的对象，通过各自对象调用上述成员函数。

【任务3】完成本章所有课后思考与练习题目，给出必要的Visual Studio程序执行结果，解释产生相关问题的原因(写在报告内！)。

1. **实验过程**

任务一：第一步：设计一个类Base, 在里面定义数据成员（public：int a， protected： int b，private： int c，public：static int count）和函数成员（输出类的数据成员函数print（），统计类对象个数函数 static int statistic（））。

第二步：设计三个不同的子类，分别用三种不同的继承方式去继承父类Base，分别在四个类里面定义构造函数和析构函数，并且当初始化对象和析构对象是能够打印出来。

第三部，在主函数里面输出四个类的大小，总结规律，在主函数里面定义不同的对象，派生类对象被建立时派生类是如何调用构造函数的，给出构造函数调用的次序，析构函数析构次序，并分析其中规律。

任务二：第一步设计一个二维空间点类Location，并在类里面设计两个普通函数。、

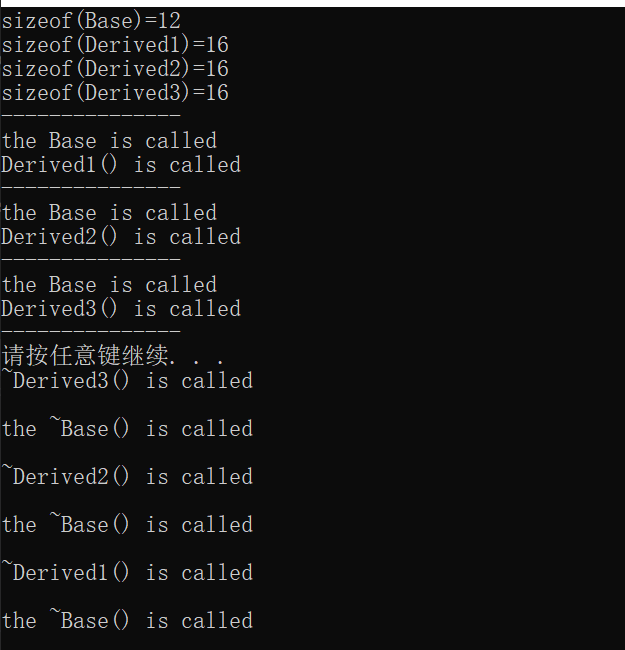
第二步 通过类的派生可以派生出空间点类Point，利用空间点类Point派生出空间球体类Sphere。

第三步 在主函数里面定义各个类的对象，通过各个对象调用上述的成员函数。

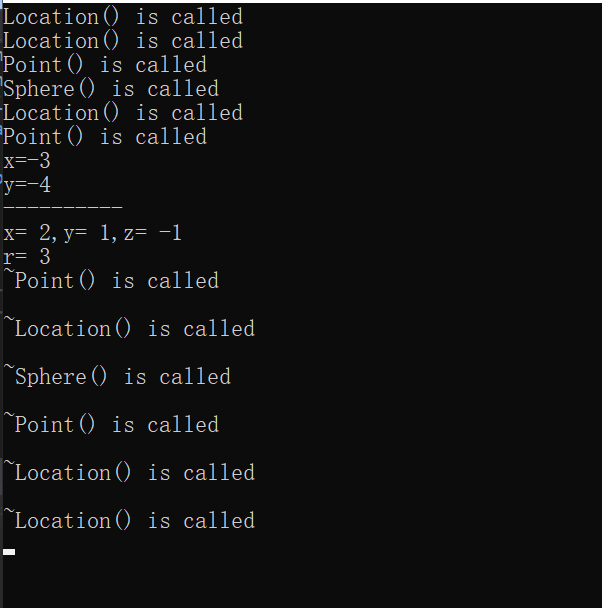
任务三：对象成员初始值妍在初始化列表中定义，不可以放在对象成员中初始化。将A类构造函数改之后不可行。对象成员和基类构造函数必须放在初始化列表中。在类中不可用赋值语句。X=100输出的是子类的。输出的不可能是父类的，因为没有用virtual.

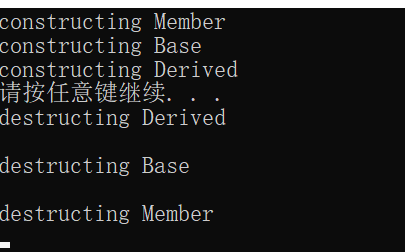
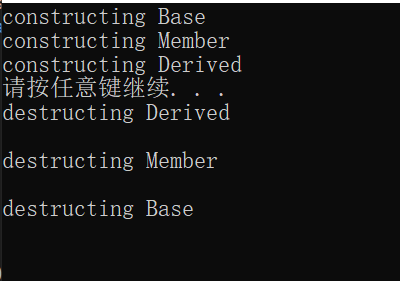
1. **实验结果**

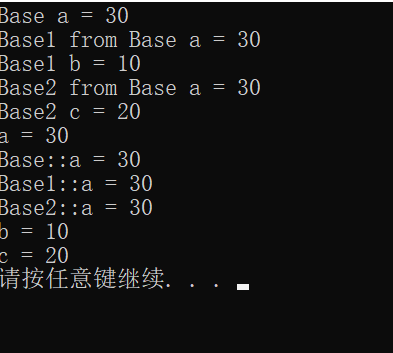
任务一



任务二：







1. **实验总结**

通过建立父类子类理解了类的三种不同关系（组合，依赖，继承），掌握复合类构造函数、析构函数的定义方法与使用方法，熟练掌握了类继承的定义方式（单继承，多继承），理解三种不同继承方式间的区别（公有，私有，保护），掌握派生类同名覆盖原理及相应同名冲突解决方法，掌握赋值兼容性基本原理（左公 = 右派），熟练掌握复杂类的设计方法（三构一析+普函）收获很多

* **附录：程序源码（建议基于Highlight软件导入）**

任务一

1 **#include<iostream>**

2 **#include<cstdlib>**

3 **using namespace** std**;**

4 class Base

5 **{**

6 public**:**

7 **int** a**;**

8 **int** print**();**

9 **static int** statistic**();**

10 Base**(int** \_a **=** **0,int** \_b **=** **0,int** \_c **=** **0)**

11 **{**

12 **this->**a **=** \_a**;**

13 **this->**b **=** \_b**;**

14 **this->**c **=**\_c**;**

15 cout **<<** "the Base is called" **<<** endl**;**

16

17

18 **}**

19 **~**Base**()**

20 **{**

21 cout **<<** "the ~Base() is called " **<<** endl**;**

22 getchar**();**

23 **}**

24 protected**:**

25 **int** b**;**

26 private**:**

27 **int** c**;**

28 **static int** count**;**

29

30 **};**

31 **int** Base**::**count **=** **0;**

32 class Derived1**:**public Base

33 **{**

34 public**:**

35 **int** a**;**

36 Derived1**()**

37 **{**

38 cout **<<** "Derived1() is called " **<<** endl**;**

39

40 **}**

41 **~**Derived1**()**

42 **{**

43 cout **<<** "~Derived1() is called" **<<** endl**;**

44 getchar**();**

45 **}**

46 **};**

47 class Derived2 **:**public Base

48 **{**

49 public**:**

50 **int** a**;**

51 Derived2**()**

52 **{**

53 cout **<<** "Derived2() is called " **<<** endl**;**

54

55 **}**

56 **~**Derived2**()**

57 **{**

58 cout **<<** "~Derived2() is called" **<<** endl**;**

59 getchar**();**

60 **}**

61 **};**

62 class Derived3 **:**public Base

63 **{**

64 public**:**

65 **int** a**;**

66 Derived3**()**

67 **{**

68 cout **<<** "Derived3() is called " **<<** endl**;**

69

70 **}**

71 **~**Derived3**()**

72 **{**

73 cout **<<** "~Derived3() is called" **<<** endl**;**

74 getchar**();**

75 **}**

76 **};**

77 **int** main**()**

78 **{**

79 cout **<<** "sizeof(Base)=" **<<** **sizeof(**Base**) <<** endl**;**

80 cout **<<** "sizeof(Derived1)=" **<<** **sizeof(**Derived1**) <<** endl**;**

81 cout **<<** "sizeof(Derived2)=" **<<** **sizeof(**Derived2**) <<** endl**;**

82 cout **<<** "sizeof(Derived3)=" **<<** **sizeof(**Derived3**) <<** endl**;**

83 cout **<<** "---------------" **<<** endl**;**

84 Derived1 one**;**

85 cout **<<** "---------------" **<<** endl**;**

86

87 Derived2 two**;**

88 cout **<<** "---------------" **<<** endl**;**

89

90 Derived3 three**;**

91 cout **<<** "---------------" **<<** endl**;**

92

93 one**.**a **=** **1;**

94

95 two**.**a **=** **2;**

96 three**.**a **=** **3;**

97 system**(**"pause"**);**

98 **return** **0;**

任务二

1 **#include<iostream>**

2 **#include<cstdlib>**

3 **using namespace** std**;**

4 class Location

5 **{**

6 public**:**

7 **double** x**;**

8 **double** y**;**

9 Location**(int** \_x**=0,int** \_y**=0):**x**(**\_x**),**y**(**\_y**)**

10 **{**

11

12 cout **<<** "Location() is called" **<<** endl**;**

13 **}**

14 **~**Location**()**

15 **{**

16 cout **<<** "~Location() is called " **<<** endl**;**

17 getchar**();**

18 **}**

19 **int** move**()**

20 **{**

21 x **=** x **\* -1;**

22 y **=** y **\* -1;**

23 **return** **0;**

24 **}**

25 **int** show**()**

26 **{**

27 cout **<<** "x=" **<<** x **<<** endl**;**

28 cout **<<** "y=" **<<** y **<<** endl**;**

29 cout **<<** "----------" **<<** endl**;**

30 **return** **0;**

31

32 **}**

33 **};**

34 class Point **:**public Location

35 **{**

36 public**:**

37 **double** x**;**

38 **double** y**;**

39 **double** z**;**

40 Point**(int** \_x **=** **0,** **int** \_y **=** **0,** **int** \_z **=** **0):** x**(**\_x**),** y**(**\_y**),** z**(**\_z**)**

41 **{**

42

43 cout **<<** "Point() is called " **<<** endl**;**

44 **}**

45 **~**Point**()**

46 **{**

47 cout **<<** "~Point() is called " **<<** endl**;**

48 getchar**();**

49 **}**

50 **};**

51 class Sphere **:**public Point

52 **{**

53 public**:**

54 **double** x**;**

55 **double** y**;**

56 **double** z**;**

57 **double** r**;**

58 Sphere**(int** \_x **=** **0,** **int** \_y **=** **0,** **int** \_z **=** **0,int** \_r**=0) :** x**(**\_x**),** y**(**\_y**),** z**(**\_z**),**r**(**\_r**)**

59 **{**

60 cout **<<** "Sphere() is called" **<<** endl**;**

61

62 **}**

63 **~**Sphere**()**

64 **{**

65 cout **<<** "~Sphere() is called" **<<** endl**;**

66 getchar**();**

67 **}**

68 **int** move**()**

69 **{**

70 x **=** x **\* -1** **+** **5;**

71 y **=** y **\* -1** **+** **5;**

72 z **=** z **\* -1** **+** **5;**

73

74 **return** **0;**

75

76 **}**

77 **int** show**()**

78 **{**

79 cout **<<** "x= " **<<** x **;**

80 cout **<<** ",y= " **<<**y **;**

81 cout **<<** ",z= " **<<** z **<<** endl**;**

82 cout **<<** "r= " **<<**r **<<** endl**;**

83

84 **return** **0;**

85

86 **}**

87 **};**

88 **int** main**()**

89 **{**

90

91 Location one**(3,4);**

92

93 Sphere two**(3,** **4,** **6,3);**

94 Point three**;**

95 one**.**move**();**

96 one**.**show**();**

97 two**.**move**();**

98 two**.**show**();**

99

100

101 **}**