编号：1-1

山东师范大学
SHANDONG NORMAL UNIVERSITY

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：	熊淑贤
学号：	201911010517
班级：	计本 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。

一、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）

二、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，private：static int count）和函数成员（输出类的数据成员函数 print（），统计类对象创建个数的函数 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）

①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。

②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）。

③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！)。

三、实验过程

【任务 1】

(1)反复翻看理解书中代码、老师课堂演示及上传到 GitHub 中的代码，设计基类 base 和三种方式派生出的子类 derived1， derived2， derived3.

(2)通过程序论证问题：

①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。

②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。

public 继承特点为：不改变基类成员属性；类内可以访问基类中的公有、保护成员，类外可以直接访问基类中的公有成员。

protected 继承特点为：基类的公有成员与保护成员变成派生类中的保护成员，基类中的私有成员变成派生类中的私有成员；类内可以访问基类中的公有、保护成员，类外不可直接访问基类成员。

private 继承特点为：基类中的成员皆变成派生类中的私有成员；类内可以访问基类中的公有、保护成员，类外不可直接访问基类成员。

③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。

派生类对象被建立时构造函数调用的次序为：基类 > 派生类内成员对象 > 派生类自己。

析构函数析构次序：派生类自己 > 派生类内成员对象 > 基类。

其中规律：析构函数析构次序与构造函数调用次序完全相反。

【任务 2】

(1)为了解决派生类内不能直接访问基类 private 数据成员的问题，将坐标 x， y， z 设成了 protected 属性，便于在派生类中直接修改相关变量

(2)定义了两类 move(a，b，c) 函数，一类直接将坐标 (x，y，z) 移动到 (a，b，c)，一类将 (x，y，z) 移动到 (x+a，y+b，z+c)

【任务 3】

(1)写出书上代码　(2)跑代码

结果及解释：

4-1：

```
construting A
construting B
The end.
destructing B
destructing A
```

类的组合关系中，建立对象时，先调用对象成员构造函数，再调用对象自身构造函数

个人答案错误：将 The end. 写在析构函数之后，其实应该写在构造函数调用结束之后，析构函数调用之前，因为析构函数是在对象生命周期结束、被销毁时调用的

4-2

```
constructing A, n = 10 //类的组合，先按定义顺序调用对象成员a 的构造函数
constructing A, n = 5 //按定义顺序接着调用对象成员b的构造函数，b又先调用b中的对象成员a的构造函数，再调用b自身的构函
constructing B, n = 6 //调用b的构函
constructing C, n = 7 //调用对象c自身构函
destructing C, n = 7 //析构顺序和构造顺序完全相反
destructing B, n = 6
destructing A, n = 5
destructing A, n = 10
```

4-3

```
A
B
AC //C 继承A，先调用基类构函，再调用自身
ABD //D 继承A， B， 按继承顺序调用基类构函，先调A，再B， 再派生类D构函
ACBABDE //E直接继承B，virtual 继承C，类内定义对象成员D d,先构造虚基类C，输出“AC”,再构造直接基类B，输出‘B’，接着构造对象成员d,输出“ABD", 最后构造自身
ACABDBABDEACACACBABDEF //F virtual继承C，直接继承D、E，类中定义对象成员C c, d; E e; ①先构造虚基类C，再构造直接基类D，输出“ACABD”②接着构造直接基类E,因为C是虚基类，所以其构造函数只调用一次，又因为先构造虚基类，所以继承了C的直接基类E不再调用C的构函，不再输出“ACBABDE”而是输出 “BABDE”③构造对象成员c,d ,输出“ACAC”，④构造对象成员e,输出“ACBABDE”⑤调用对象f自身构函，输出'F'
```

4-4

```
X::X() constructor executing
Y::Y() constructor executing
Z::Z() constructor executing
Z::~Z() destructor executing
Y::~Y() destructor executing
X::~X() destructor executing
```

4-5

```
d.A::n = 10, d.B::n = 10, d.C::n = 10, d.D::n = 10
d.A::n = 20, d.B::n = 20, d.C::n = 20, d.D::n = 20
d.A::n = 30, d.B::n = 30, d.C::n = 30, d.D::n = 30
d.A::n = 40, d.B::n = 40, d.C::n = 40, d.D::n = 40
```

此题定义了虚基类 A ,所以公共基类 A 的成员在各派生类中都只有一个拷贝，通过任何一个类作用域来改变成员 n 的值时，所有列中访问的都是同一个值

4-6

```
X::f() executing
X::f() executing  //复制兼容，Y中的f()算派生类新增成员，不可访问，所以调用的是基类X中的f()
Y::f() executing  //同名覆盖，调用Y中的f()
```

四、实验结果

【任务 1】

```
constructing Base, a = 4, b = 5, c = 6
constructing Derived1, a = 4, b = 5, c = 6, d1 = 7
count = 2

constructing Base, a = 8, b = 9, c = 10
constructing Derived2, a = 8, b = 9, c = 10, d2 = 11
count = 3

constructing Base, a = 12, b = 13, c = 14
constructing Derived3, a = 12, b = 13, c = 14, d3 = 15
count = 4

x.a = 4, x.b = 5, x.d1 = 7

y.a = 8

y.c = 10

destructing Derived3, a = 12, b = 13, c = 14, d3 = 15
destructing Base, a = 12, b = 13, c = 14
destructing Derived2, a = 8, b = 9, c = 10, d2 = 11
destructing Base, a = 8, b = 9, c = 10
destructing Derived1, a = 4, b = 5, c = 6, d1 = 7
destructing Base, a = 4, b = 5, c = 6
destructing Base, a = 1, b = 2, c = 3

C:\Users\elegant\source\repos\Project1\Debug\Project1.exe (进程 6904)
按任意键关闭此窗口...
```

【任务 2】

```
a.x = 3, a.y = 4
after move(1, 2), the location is : (x, y) = (1, 2)

after move_add(1, 2), the location is : (x, y) = (2, 4)

b.x = 5, b.y = 6, b.z = 7
after move(3, 4, 5), the point's location is : (x, y, z) = (3, 4, 5)

after move_add(5, 6, 7), the point's location is : (x, y, z) = (8, 10, 12)

c.x = 8, c.y = 9, c.z = 10, c.r = 11
after move(1, 2, 3), the sphere's location is : (x, y, z) = (1, 2, 3)
The radius is r = 11
the area is S = 1520.53
the volume is V = 4181.46

after move_add(3, 4, 5), the sphere's location is : (x, y, z) = (4, 6, 8)
The radius is r = 11
the area is S = 1520.53
the volume is V = 4181.46

after modify_r(3), the sphere's location is : (x, y, z) = (4, 6, 8)
The radius is r = 3
the area is S = 113.097
the volume is V = 84.823

after add_r(6), the sphere's location is : (x, y, z) = (4, 6, 8)
The radius is r = 9
the area is S = 1017.88
the volume is V = 2290.22


C:\Users\elegant\source\repos\Project1\Debug\Project1.exe (进程 20124)已退出，返回
按任意键关闭此窗口...
```

【任务 3】

```
constructing A
constructing B
The end.
destructing B
destructing A
```

```
constructing A, n = 10
constructing A, n = 5
constructing B, n = 6
constructing C, n = 7
destructing C, n = 7
destructing B, n = 6
destructing A, n = 5
destructing A, n = 10
```

```
A
B
AC
ABD
ACBABDE
ACABDBABDEACACACBABDEF
```

```
X::X() constructor executing
Y::Y() constructor executing
Z::Z() constructor executing
Z::~Z() destructor executing
Y::~Y() destructor executing
X::~X() destructor executing
```

```
d.A::n = 10, d.B::n = 10, d.C::n = 10, d.D::n = 10
d.A::n = 20, d.B::n = 20, d.C::n = 20, d.D::n = 20
d.A::n = 30, d.B::n = 30, d.C::n = 30, d.D::n = 30
d.A::n = 40, d.B::n = 40, d.C::n = 40, d.D::n = 40

C:\Users\elegant\source\repos\Project1\Debug\Project1.exe
按任意键关闭此窗口...
```

```
X::f() executing
X::f() executing
Y::f() executing

C:\Users\elegant\source\repos\Pro
按任意键关闭此窗口...
```

五、实验总结

程序实践是是重要教学环节之一。我们需要通过程序实践，增强工程化意识,提高 C/C++语言实践能力。强化计算机应用技能，从而巩固和充实所学的理论知识，加深对相关内容的理解，拓宽知识面，培养的创新精神和实践能力。

通过这次实验，我理解了类的三种不同关系：组合，依赖，继承；掌握了复合类构造函数、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（公有，私有，保护）间的区别；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）；熟练掌握了复杂类的设计方法（三构一析+普函）

■ 附录：程序源码（建议基于 Highlight 软件导入）

GitHub 源码托管地址：

https://github.com/Shuxian-X/the-first/tree/master/%E7%AC%AC%E5%9B%9B%E7%AB%A0%E7%AC%AC%E4%B8%80%E6%AC%A1%E4%BD%9C%E4%B8%9A

【任务 1】

```
01 #include <iostream>
02
03 using namespace std;
04
05 class Base
06 {
07     static int cnt;
08 public:
09     int a;
10 protected:
11     int b;
12 private:
13     int c;
14
15 public:
16     Base()
17     {
18         ++cnt;
19         cout << "constructing Base, a = " << a << ", b = " << b << ", c = " << c << endl;
20     }
21     Base(int x, int y, int z)
22     {
23         ++cnt;
24         a = x;
25         b = y;
26         c = z;
27         cout << "constructing Base, a = " << a << ", b = " << b << ", c = " << c << endl;
28     }
29     ~Base()
30     {
31         --cnt;
32         cout << "destructing Base, a = " << a << ", b = " << b << ", c = " << c << endl;
33     }
34
35     int get_a()
```

```
36      {
37          return a;
38      }
39      int get_b()
40      {
41          return b;
42      }
43      int get_c()
44      {
45          return c;
46      }
47      static int statistic();
48  };
49
50  int Base::cnt = 0;
51
52  class Derived1 : public Base
53  {
54  private:
55      int d1;
56  public:
57      Derived1()
58      {
59          // cout << "constructing Derived1, a = " << a << ", b = " << b << ", c = " << c <<
"d1 = " << d1 << endl;
60          // 报错，私有数据 c 虽然被子类继承，但是不可见，即具有不可访问属性
61          cout << "constructing Derived1, a = " << a << ", b = " << b << ", c = " << get_c()
<< ", d1 = " << d1 << endl;
62      }
63      Derived1(int x, int y, int z, int t) : Base(x, y, z)
64      {
65          d1 = t;
66          // cout << "constructing Derived1, a = " << a << ", b = " << b << ", c = " << c <<
"d1 = " << d1 << endl;
67          // 报错，私有数据 c 虽然被子类继承，但是不可见，即具有不可访问属性
68          cout << "constructing Derived1, a = " << a << ", b = " << b << ", c = " << get_c()
<< ", d1 = " << d1 << endl;
69      }
70      ~Derived1()
71      {
72          cout << "destructing Derived1, a = " << a << ", b = " << b << ", c = " << get_c() <<
", d1 = " << d1 << endl;
73      }
74      int get_d1()
75      {
```

```
76          return d1;
77      }
78  };
79
80  class Derived2 : protected Base
81  {
82  private:
83      int d2;
84  public:
85      Derived2()
86      {
87          cout << "constructing Derived2, a = " << a << ", b = " << b << ", c = " << get_c()
<< ", d2 = " << d2 << endl;
88      }
89      Derived2(int x, int y, int z, int t) : Base(x, y, z)
90      {
91          d2 = t;
92          cout << "constructing Derived2, a = " << a << ", b = " << b << ", c = " << get_c()
<< ", d2 = " << d2 << endl;
93      }
94      ~Derived2()
95      {
96          cout << "destructing Derived2, a = " << a << ", b = " << b << ", c = " << get_c() <<
", d2 = " << d2 << endl;
97      }
98      int get_d2()
99      {
100         return d2;
101     }
102     int Get_a()
103     {
104         return a;
105     }
106     int Get_b()
107     {
108         return b;
109     }
110     int Get_c()
111     {
112         return get_c();// 通过派生类公有函数访问基类继承变为保护的成员函数，进而访问私有数据成员
113     }
114 };
115
116 class Derived3 : private Base
117 {
```

```
118 private:
119     int d3;
120 public:
121     Derived3()
122     {
123         cout << "constructing Derived3, a = " << a << ", b = " << b << ", c = " << get_c()
<< ", d3 = " << d3 << endl;
124     }
125     Derived3(int x, int y, int z, int t) : Base(x, y, z)
126     {
127         d3 = t;
128         cout << "constructing Derived3, a = " << a << ", b = " << b << ", c = " << get_c()
<< ", d3 = " << d3 << endl;
129     }
130     ~Derived3()
131     {
132         cout << "destructing Derived3, a = " << a << ", b = " << b << ", c = " << get_c()
<< ", d3 = " << d3 << endl;
133     }
134     int get_d3()
135     {
136         return d3;
137     }
138 };
139
140 int Base::statistic()
141 {
142     return cnt;
143 }
144
145 int main()
146 {
147     Base t(1, 2, 3);
148     cout << "count = " << Base::statistic() << endl << endl;
149     Derived1 x(4, 5, 6, 7);
150     cout << "count = " << Base::statistic() << endl << endl;
151     Derived2 y(8, 9, 10, 11);
152     cout << "count = " << Base::statistic() << endl << endl;
153     Derived3 z(12, 13, 14, 15);
154     cout << "count = " << Base::statistic() << endl << endl;
155
156     //cout << "x.a = " << x.a << ", x.b = " << x.b << "x.d1 = " << x.d1 << endl;
157     // b, d1  不可直接访问
158     cout << "x.a = " << x.a << ", x.b = " << x.get_b() << ", x.d1 = " << x.get_d1() << endl
<< endl;
```

```
159
160     //cout << "y.a = " << y.a << endl;
161     // a 不可直接访问
162     //cout << "y.a = " << y.get_a() << endl;
163     // get_a() 不可直接访问
164     cout << "y.a = " << y.Get_a() << endl << endl;
165
166     cout << "y.c = " << y.Get_c() << endl << endl;
167
168     return 0;
169 }
```

【任务 2】

```
01 #include <iostream>
02
03 using namespace std;
04
05 class Location
06 {
07 protected:
08     double x, y;
09 public:
10     Location(double a, double b)
11     {
12         x = a;
13         y = b;
14     }
15     ~Location()
16     {
17
18     }
19
20     void move(double a, double b) //将坐标移动到 (a, b)
21     {
22         x = a;
23         y = b;
24     }
25     void move_add(double a, double b) //将坐标 (x, y) 分别增加 a, b 个单位，移动到 (x+a, y+b)
26     {
27         x += a;
28         y += b;
29     }
30     double get_x()
31     {
32         return x;
```

```
33      }
34      double get_y()
35      {
36          return y;
37      }
38      void show()
39      {
40          cout << "the location is : (x, y) = " << "(" << x << ", " << y << ")" <<  endl <<
endl;
41      }
42  };
43
44  class Point : public Location
45  {
46  protected:
47      double z;
48  public:
49      Point(double a, double b, double c) : Location(a, b)
50      {
51          z = c;
52      }
53      ~Point()
54      {
55
56      }
57
58      void move(double a, double b, double c)
59      {
60          x = a;
61          y = b;
62          z = c;
63      }
64      void move_add(double a, double b, double c) //将点 (x, y, z) 的3个坐标分别增加 a, b, c 个单
位，移动到 (x+a, y+b, z+c)
65      {
66          x += a;
67          y += b;
68          z += c;
69      }
70      double get_z()
71      {
72          return z;
73      }
74      void show()
75      {
```

```
76          cout << "the point's location is : (x, y, z) = " << "(" << x << ", " << y << ", " << z << ")" << endl << endl;
77      }
78  };
79
80  class Sphere : public Point
81  {
82  protected:
83      const double PI; //圆周率
84      double r, S, V; //半径，表面积，体积
85  public:
86      Sphere(double a, double b, double c, double rr) : Point(a, b, c), PI(3.1415926) //初始化常数据成员
87      {
88          r = rr;
89          S = 4.0*PI*r*r;
90          V = 4 / 3 * PI*r*r*r;
91      }
92      ~Sphere()
93      {
94
95      }
96
97      void modify_r(double rr) //半径修改为 rr
98      {
99          r = rr;
100         S = 4.0*PI*r*r;
101         V = 4 / 3 * PI*r*r*r;
102
103     }
104     void add_r(double rr) //半径增加 rr
105     {
106         r += rr;
107         S = 4.0*PI*r*r;
108         V = 4 / 3 * PI*r*r*r;
109     }
110     double get_r()
111     {
112         return r;
113     }
114     double get_S()
115     {
116         return S;
117     }
118     double get_V()
```

```
119     {
120         return V;
121     }
122     void show()
123     {
124         cout << "the sphere's location is :  (x, y, z) = " << "(" << x << ", " << y << ", "
<< z << ")" << endl;
125         cout << "The radius is r = " << r << "\nthe area is S = " << S << "\nthe volume is
V = " << V << endl << endl;
126     }
127 };
128
129 int main()
130 {
131     Location a(3, 4);
132     //cout << "a.x = " << a.x << ", a.y = " << a.y << endl; //报错，不可访问
133     cout << "a.x = " << a.get_x() << ", a.y = " << a.get_y() << endl;
134     a.move(1, 2);
135     cout << "after move(1, 2), ";
136     a.show();
137     a.move_add(1, 2);
138     cout << "after move_add(1, 2), ";
139     a.show();
140
141     Point b(5, 6, 7);
142     //cout << "b.x = " << b.x << ", b.y = " << b.y << ", b.z = " << b.z << endl;//报错，不可
访问
143     cout << "b.x = " << b.get_x() << ", b.y = " << b.get_y() << ", b.z = " << b.get_z() <<
endl;
144     b.move(3, 4, 5);
145     cout << "after move(3, 4, 5), ";
146     b.show();
147     b.move_add(5, 6, 7);
148     cout << "after move_add(5, 6, 7), ";
149     b.show();
150
151     Sphere c(8, 9, 10, 11);
152     //cout << "c.x = " << c.x << ", c.y = " << c.y << ", c.z = " << c.z << ", c.r = " <<
c.r << endl; //报错，不可访问
153     cout << "c.x = " << c.get_x() << ", c.y = " << c.get_y() << ", c.z = " << c.get_z() <<
", c.r = " << c.get_r() << endl;
154     c.move(1, 2, 3);
155     cout << "after move(1, 2, 3), ";
156     c.show();
157     c.move_add(3, 4, 5);
```

```
158     cout << "after move_add(3, 4, 5), ";
159     c.show();
160     c.modify_r(3);
161     cout << "after modify_r(3), ";
162     c.show();
163     c.add_r(6);
164     cout << "after add_r(6), ";
165     c.show();
166 
167     return 0;
168 }
```

【任务 3】

略，见 github 链接 or 课本 P165