

计算机学院 高级语言程序设计 课程实验报告

实验题目：继承与派生（二）		学号：202200400053
日期：2024-04-25	班级：2202	姓名：王宇涵
Email： 1941497679@qq.com		
<p>实验目的：</p> <ol style="list-style-type: none">1. 掌握继承中同名隐藏问题的解决方案，虚基类。（覆盖 override 与隐藏 hide）2. 认识派生类对象的内存布局3. 掌握继承中类型转换应用		
<p>实验软件和硬件环境：</p> <p>软件环境：VSCODE + DEV-C++</p> <p>硬件环境：Legion Y7000P</p>		
<p>实验步骤与内容：</p> <p>参照《C++语言程序设计》学生用书，完成以下实验：</p> <ol style="list-style-type: none">1. 运行第7章实验7，实验任务（3），虚基类。 lab7_3 <pre>1 #include <iostream> 2 using namespace std; 3 4 class Vehicle 5 { 6 public: 7 int MaxSpeed; 8 int Weight; 9 void Run(); 10 void Stop(); 11 }; 12 13 class Bicycle : virtual public Vehicle 14 { 15 public: 16 int Height; 17 }; 18 19 class Motorcar : virtual public Vehicle 20 { 21 public: 22 int SeatNum; 23 }; 24 25 class Motorcycle : public Bicycle, public Motorcar 26 { 27 }; 28 29 int main() 30 { 31 return 0; 32 } 33</pre> <p>问题 输出 调试控制台 终端 端口</p> <p>PS D:\BaiduSyncdisk\CLASSES\C++\exp\exp7\实验七代码> cd 'd:\BaiduSyncdisk\CLASSES\C++\exp\exp7\实验七代码\output'</p> <p>PS D:\BaiduSyncdisk\CLASSES\C++\exp\exp7\实验七代码\output> & .\lab7_3.exe</p>		

2. 运行第 7 章实验 7，实验任务（4），多重继承。lab7_4

如果用 visual studio 高版本对 char*安全性的警示，可以替换成 string。或者在项目属性中，将 C++ 语言的“符合模式”改为否，就可以了。它默认“是”。

输出

```
TA name: Li Chao
No.: 011401
Department: CST
Class No.: cst61
Subject: computer science
Advisor: Zheng Li
```

其中 teacher 和 student 类继承虚基类 people，graduate 继承类 student，TA 多重继承继承类 graduate 和 teacher。

3. 运行第 7 章习题 7-13，探索派生类对象的内存布局（普通多继承中的成员，虚基类多继承中的成员）。

例 7-6

```
Base1's var pointer: 0x62fe14
Base2's var pointer: 0x62fe18
```

发现按照 base1 和 base2 的基类成员 var 按继承顺序在前后。

例 7-8

```
derived pointer value: 0x62fde0
Base0's pointer value: 0x62fe00
Base1's pointer value: 0x62fde0
Base2's pointer value: 0x62fdf0
Base1's var1 pointer: 0x62fde8
Base2's var2 pointer: 0x62fdf8
Derived's var pointer: 0x62fdfc
Base1's Base0's var0 pointer: 0x62fe00
Base2's Base0's var0 pointer: 0x62fe00
```

发现从低到高的地址顺序为

Derived; base1

Base1 var1

Base2

Base2 var2

Derived var

Base0; base1::base0::var0; base2::base0::var0

纠错：第 7 章习题 7-14 中的解释有错（第一句话后半段与前半段矛盾）。

虚基类的指针无法显式<static_cast>转换为子类的指针。

4. 运行第 7 章习题 7-15，对象指针的转换。（struct 中的可访问性和继承方式都是 public）

借助空指针，转换，内容错。

```
Base2* pb = static_cast<Base2*>(pd); //这样显式转换也可以
```

```
delete pd;
```

输出结果:

```
2 2
```

以 base2 类型的指针(父)指向 derived 类型的对象(子), 可以直接访问 y 变量并输出.

5. 探索继承中的同名隐藏问题。

练习第七章 PPT, 例 7-6, 尝试做如下修改:

(1) 在 Base1 中增加 fun(int a)函数;

(2) 在 Base2 增加 fun(int a,int b)函数;

(3) 在 main()函数中增加如下语句:

```
d.fun(1);
```

```
d.fun(2,3);
```

测试是否可以通过编译? (即 Derived 类中的 fun()是否同名隐藏了父类形参不同的同名函数?)

```
class Base1 {    //定义基类Base1
public:
    int var;
    void fun() { cout << "Member of Base1" << endl; }
    void fun(int a) {}
};

class Base2 {    //定义基类Base2
public:
    int var;
    void fun() { cout << "Member of Base2" << endl; }
    void fun(int a, int b) {}
};
```

```
p->Bas
p->Bas  函数调用中的参数太多 C/C++(140)
        查看问题 (Alt+F8)  快速修复... (Ctrl+.)

d.fun(1);
d.fun(1, 2);
return 0;
```

答 : Derived 类中的 fun()函数成员同名隐藏了父类形参不同的同名函数.

6. (1) 在第七章 PPT, P49-50 例 7_7 的 main()函数中增加输出语句 (如下), 问有几个 var0?

如果再增加输出语句 `cout<<d.var0<<" "<<d.Base0::var0<<endl`; 会怎样? 为什么?

```
int main() { //程序主函数
```

```
    Derived d; //定义 Derived 类对象 d
```

```
    d.Base1::var0 = 2; //使用直接基类
```

```
    d.Base1::fun0();
```

```
    d.Base2::var0 = 3; //使用直接基类
```

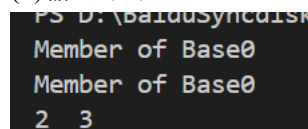
```
    d.Base2::fun0();
```

```
    cout<<d.Base1::var0<<" "<<d.Base2::var0<<endl; //有几个 var0?
```

```
    return 0;
```

```
}
```

(1)输出结果



```
PS D:\baibao\syncdisk>
Member of Base0
Member of Base0
2 3
```

有 2 个 var0

(2)如果再增加输出语句



```
Derived d; //定义 Derived 类对象 d

d.Base2::var0 = 3; //使用直接基类

cout<<d.Base1::var0<<" "<<d.Base2::var0<<endl;

cout<<d.var0<<" "<<d.Base0::var0<<endl;
```

"Derived::var0" 不明确 C/C++(266)

int Base0::var0

查看问题 (Alt+F8) 快速修复... (Ctrl+.)

显示意义不明确, 编译错误.

(2) 练习第七章 PPT, P57-58 例, 将 main 函数改为如下, 解释运行结果。

```
int main() { //程序主函数
```

```
    Derived d(1); //定义 Derived 类对象 d
```

```
    cout<<d.var0<<" "<<d.Base1::var0<<" "<<d.Base2::var0<<endl;
```

```
    d.var0 = 2; //直接访问虚基类的数据成员
```

```
    d.fun0(); //直接访问虚基类的函数成员
```

```
    cout<<d.var0<<" "<<d.Base1::var0<<" "<<d.Base2::var0<<endl;
```

```
    d.Base1::var0=1;
```

```
    d.Base2::var0=3;
```

```
cout<<d.var0<<" "<<d.Base1::var0<<" "<<d.Base2::var0<<endl;
return 0;

}
```

运行结果:

```
PS D:\BaiduSyncdisk\CLASSES\C++\exp\exp7\实验七代码\output> & .\p57-58.exe
1 1 1
Member of Base0
2 2 2
3 3 3
```

解释：

先创建了 derived 类型对象 d, 并用参数 1 初始化 base0, base1, base2 的 val0, 因此输出 1, 1, 1

将 val0 改为 2, 而 d, base1, base2 共用 val0, 因此输出 2, 2, 2

先将 val0 改为 1, 再把 val0 改为 3, d, base1, base2 共用 val0, 因此输出 3, 3, 3

7. 练习课本 7.6 中实例, 用高斯消去法解线性方程组, 用继承实现例 7-9。

输出结果

```
PS D:\BaiduSyncdisk\CLASSES\C++\exp\exp7\实验七代码\output> .\p57-58.exe
The Line equation is:
0.2368 0.2471 0.2568 1.2671      1.8471
0.1968 0.2071 1.2168 0.2271      1.7471
0.1581 1.1675 0.1768 0.1871      1.6471
1.1161 0.1254 0.1397 0.149       1.5471
The Result is:
x[0] = 1.04058
x[1] = 0.987051
x[2] = 0.93504
x[3] = 0.881282
```

继承举例:

```
class LinearEqu: public Matrix { //公有派生类LinearEqu定义
```

8. 练习课本 7.7 中, 个人银行账户管理程序, 用继承新增信用账户, 实现例 7-10。

输出结果

```

2008-11-1      #S3755217 created
2008-11-1      #02342342 created
2008-11-1      #C5392394 created
2008-11-5      #S3755217      5000      5000      salary
2008-11-15     #C5392394      -2000     -2000     buy a cell
2008-11-25     #02342342      10000    10000    sell stock 0323
2008-12-1      #C5392394      -16      -2016    interest
2008-12-1      #C5392394      2016     0        repay the credit
2008-12-5      #S3755217      5500     10500    salary
2009-1-1       #S3755217      17.77    10517.8  interest
2009-1-1       #02342342      15.16    10015.2  interest
2009-1-1       #C5392394      -50      -50      annual fee

S3755217      Balance: 10517.8
02342342      Balance: 10015.2
C5392394      Balance: -50      Available credit:9950
Total: 20482.9
PS D:\BaiduSyncdisk\CLASSES\C++\exp\exp7\实验七代码\output>

```

继承新增信用账户

```
class CreditAccount : public Account { //信用账户类
```

结论分析与体会：

本次实验我学习了继承和派生的深层知识，包括派生类对象的内存布局，不同情况下的内存布局，以及基态和派生类之间转换的注意事项等等，并通过实际的案例，包括高斯消元法案例，个人银行账户管理程序案例等等，巩固了继承和派生的知识，提升了自己解决问题的能力。

就实验过程中遇到的问题及解决处理方法，自拟 1—3 道问答题：

1. 继承虚基类和普通继承有什么区别？

答：**虚继承**：当一个类以虚拟方式继承另一个类时，它将共享基类的一个实例，而不是每个派生类都有自己的基类实例。这意味着无论多少次继承自虚基类的类，虚基类中的成员只有一份拷贝。

普通继承：在普通继承中，每个派生类都会拥有基类的一份完整拷贝。这意味着每次派生类被实例化时，都会有自己的基类成员，而不共享基类实例。

2. 派生类对象的内存布局是怎样的？

答：**单继承**：基类数据在前，派生类新增数据在后

多继承：各基类数据按继承顺序在前，派生类新增数据在后

虚继承：需要增加指针，间接访虚基类数