北 京 林 业 大 学

2019 学年—2020 学年第 1 学期 面向对象程序设计 B

实验指导书

专业名称： 地信 实验学时： 2 任课教师： 卢昊

实验题目：实验三 继承和多态

实验环境：Visual Studio

实验目的

1. 掌握利用单继承的方式定义派生类的方法
2. 掌握利用多继承的方式定义派生类的方法
3. 理解在各种继承方式下构造函数和析构函数的执行顺序
4. 掌握公有继承、私有继承和保护继承对基类成员的访问机制
5. 学会虚函数、纯虚函数和抽象类的定义方法
6. 掌握虚函数和基类指针实现多态性的方法

实验要求

1. 掌握 Visual Studio 开发工具的基本用法；
2. 按要求创建、命名和管理解决方案、工程目录；
3. 使用 C++程序写法，而非 C 语言写法；
4. 将程序测试运行通过后再整理提交，严格按照命名要求命名；
5. 实验完成后认真撰写实验报告。

实验内容和步骤

注意：本次实验只需创建 **1 个解决方案**，名称为“**你的学号_s3**”，如“180XXXXXX_s3”。实验的所有习题均**在该解决方案下创建独立的工程**。创建方法请参看实验一指导书，规范命名和存放目录，本指导书不再赘述。

重要：所有工程在创建时均设置为“空项目”，避免 VS 自动创建代码。

练习 1：单继承

“车辆”是指一类交通工具的统称，而“轿车”和“货车”则是两种具体的车辆，利用面向对象程序设计中的继承特性，可以方便地实现这种逻辑关系的表达。试按要求完成下面的任务。

步骤 1：在解决方案中创建工程，命名为“学号_Single_Inheritance”，再添加**同名的 cpp** 文件。

步骤 2：在 cpp 文件中完成以下类的设计实现。

添加**基类 Vehicle** 的定义：

```
class Vehicle
{
protected:
    int n_wheels;          //保存轮子数
    double weight;         //保存车重
public:
    Vehicle(int n, double w); //对轮子数和车重进行初始化
};
```

在该 cpp 文件中写出 Vehicle 类的实现部分，并且**公有派生**出两个派生类：**Car 类**和 **Truck 类**，这两个类的定义如下：

```
class Car : public Vehicle
{
protected:
```

```
    int capacity;       //保存载客数
public:
    Car (int c, int n, double w);    //c 是载客数，n 是车轮数，w 是车重
    void print();       //输出车轮数、车重、载客数
};
class Truck : public Vehicle
{
protected:
    double load;        //保存载重量
public:
    Truck (double l, int n, double w);  //l 是载重量，n 是车轮数，w 是车重
    void print();                       //输出车轮数、车重、载重量
};
```

在该 cpp 文件中写出这两个类的实现部分。要求程序运行时可输出构造和析构信息。

步骤 3：编写 main 函数，内部声明 Car 类对象和 Truck 类对象，分别调用每个对象的类成员函数测试运行。

思考：

- 如果将 Vehicle 类的成员变量改成 private 或 public 类型，会有什么变化？
- 派生类 Car 和 Truck 分别从基类继承了哪些成员（包括变量和函数）？
- 基类和派生类的构造函数和析构函数调用的顺序是什么关系？

尝试在程序中做出相应修改看你的答案是否正确。

练习 2：多继承

“圆”是很多物体具有的基本形状，圆的基本属性是“半径”；“桌子”是一种具体的家具，但是其形状有很多种，不同形状的桌子都有基本的属性“高度”。

“圆桌”是一种具有圆特征的桌子，兼具圆和桌子的属性，这种逻辑关系可以使用面向对象程序设计中的多继承方式来表达。按要求完成下面的任务。

步骤 1：在解决方案中创建工程，命名为“学号_Multiple_Inheritance”，再

添加**同名的 cpp** 文件。

步骤 2：在 cpp 文件中完成以下类的设计实现。

定义圆类 Circle（在同一个源文件中，下同），包括以下成员：

protected 部分：double radius;　　//保存半径

public 部分：　构造函数 Circle( ); //在其中初始化 radius

　　getArea( );　　//得到圆面积

　　setRadius( );　　//对 radius 进行设定

注意：上述函数声明未包括参数列表，请按需要添加函数参数列表，下同。

定义桌子类 Table，包括以下成员：

protected 部分：　double height;　　//保存高度

public 部分：　构造函数 Table( );　//在其中初始化 height

　　getHeight( );　　//得到高度 height

利用**公有继承**方式，定义 Circle 类和 Table 类的派生类 RoundTable，包括以下成员：

protected 部分：　string color;　　//保存颜色

public 部分：　构造函数 RoundTable( );//初始化派生类和基类的成员量

　　print( ); //显示圆桌的高度、面积和颜色

在该 cpp 文件中写出这些类的实现部分。要求程序运行时可输出构造和析构信息。

步骤 3：编写 main 函数，内部声明 RoundTable 类对象，分别调用对象的每个类成员函数测试运行。

思考：

- 如果将 Circle 类和 Table 类的成员改成 private 或 public 类型，会有什么变化？
- 派生类分别从两个基类继承了哪些成员（包括变量和函数）？
- 基类和派生类的构造函数和析构函数调用的顺序是什么关系？两个基类的构造和析构顺序是怎样的？

尝试在程序中做出相应修改看你的答案是否正确。

练习 3：虚函数和多态性

银行账户主要分为个人账户和企业账户两种。在相同的业务流程中，不同类型的账户可能进行不同的操作。利用继承、虚函数和多态特性完成以下程序设计。

步骤 1：在解决方案中创建工程，命名为“你的学号_Bank”，再添加**同名的 cpp** 文件。

步骤 2：在 cpp 文件中完成以下类的设计实现。

定义基类 BankAccount，表示银行客户：

protected 部分：　int　accountNum　表示账号号码

　　double　balance　表示帐户现在的余额

public 部分：　构造函数

虚函数 printAccountInfo( )　输出成员变量的值（应该输出哪些成员？）

纯虚函数 compute( )　计算并输出年末帐户余额

步骤 3：利用**公有继承**方式，定义 BankAccount 类的派生类 PersonalAccount，表示个人帐户，包括：

private 部分：　字符数组 pwd[7]，保存 6 位用户密码，最后一位为 0

public 部分：　构造函数

　　printAccountInfo( )　输出 **3 个**成员变量的值

　　compute( )　计算并输出年末帐户余额，计算式为 balance+balance*0.1

步骤 4：利用**公有继承**方式，定义 BankAccount 类的派生类 EnterpriseAccount，表示企业帐户，包括：

private 部分：　字符数组 taxNum[10]保存企业纳税人识别号

public 部分：　构造函数

　　printAccountInfo( )　输出 **3 个**成员变量的值

　　compute( )　计算并输出年末帐户余额，计算式为 balance+balance*0.2

步骤 5：在相同 cpp 文件中编写主函数 main，定义基类和派生类对象，测试类的定义。主函数和运行结果如下：

主函数：

```
void main()
{
    // 密码 123456，账户号码 170001，余额 1000.0
    PersonalAccount pa("123456",170001,1000.0);
    pa.printAccountInfo ();
    pa.compute();

    // 纳税人识别号 bjfu123，账户号码 200001，余额 1000000.0
    EnterpriseAccount ea("bjfu123",200001,1000000.0);
    ea.printAccountInfo();
    ea.compute();

    BankAccount* pba; //声明基类指针

    pba=&pa;
    pba->printAccountInfo();
    pba->compute();

    pba=&ea;
    pba->printAccountInfo();
    pba->compute();
}
```

运行结果：

```
个人帐户
账户号码：170001
账户当前余额：1000
密码：123456
年末账户余额：1100

企业账户
账户号码：200001
账户当前余额：1e+006
企业纳税人识别号：bjfu123
年末账户余额：1.2e+006

个人帐户
账户号码：170001
账户当前余额：1100
密码：123456
年末账户余额：1210

企业账户
账户号码：200001
账户当前余额：1.2e+006
企业纳税人识别号：bjfu123
年末账户余额：1.44e+006
```

思考：

- 该程序中，什么函数的运行结果体现了多态性？
- 哪些操作、用法保证了多态性的实现？尝试将它（们）去掉观察结果。

尝试在程序中做出相应修改看你的答案是否正确。

练习 4：抽象类

编写程序，计算圆柱体、正方体和长方体的表面积和体积。要求圆柱体类、正方体类和长方体类都由同一个**抽象类**派生，并且这三个类的计算表面积的函数和计算体积的**函数原型**完全一样（提示：可以定义一个抽象类，其中包括分别求表面积和体积的纯虚函数，然后在此类基础上，派生出圆柱体类、正方体类和长方体类）。

步骤 1：在解决方案中添加新的工程，命名为“你的学号_Shape”，再添加**同名的 cpp** 文件。

步骤 2：在 cpp 文件中完成以下类的设计实现。

定义抽象基类 **Shape**，表示图形，包含：

public 部分：　　纯虚函数 area( )　　//求表面积函数

　　纯虚函数 volume( ) //求体积函数

其余的数据成员和成员函数自定。

步骤 3：利用**公有继承**方式，（在同一个源文件中，下同）定义 Shape 类的派生类 Cylinder，表示圆柱体，包含：

public 部分：　　重定义 area( )　//重定义即 override，或称重写、覆盖

　　重定义 volume( )

其余的数据成员和成员函数自定。

步骤 4：利用**公有继承**方式，定义 Shape 类的派生类 Cube，表示正方体，包含

public 部分：　　重定义 area( )

　　重定义 volume( )

其余的数据成员和成员函数自定。

步骤 5：利用**公有继承**方式，定义 Shape 类的派生类 Cuboid，表示长方体，

包含

public 部分：　　重定义 area( )

　　重定义 volume( )

其余的数据成员和成员函数自定。

步骤 6：编写 main 函数进行测试，要求使用**基类的指针调用不同派生类对象**的求表面积函数和求体积函数，输出圆柱体、正方体和长方体的表面积和体积，测试数据如下：

圆柱体：（半径：5；高：10）； 正方体：（边长：10）； 长方体：（长：10；宽：8；高：5）。

思考：

- 该程序中，什么函数的运行结果体现了多态性？
- 哪些操作、用法保证了多态性的实现？尝试将它（们）去掉观察结果。

尝试在程序中做出相应修改看你的答案是否正确。

练习 5：利用多态性设计重写程序

常见的图像格式有很多，如 JPG、PNG、TIFF、BITMAP 等，它们均具有这些基本属性和基本操作，但是，对于不同图像格式，相同操作必须以不同的方式实现。思考多态性的用法，以 Bitmap 格式为例，将实验二练习 5 的 C++程序以多态性的方式重写，并测试运行，达到与原程序同样的结果。

步骤 1：在解决方案中添加新的工程，命名为“你的学号_polymorphism”，再添加同名的 cpp 文件。

步骤 2：设计相关的类，完成程序功能，**可以全部写在一个 cpp 文件中**。

提示：设计一个抽象基类 CImage，表示一般的图像，该类包含图像的基本属性（width，height 等），将图像的基本操作（读取、输出等）封装为虚函数或纯虚函数。Bitmap 是一种简单的图像格式，可由 CImage 类派生而来，避免重写通用的成员。

步骤 3：测试运行。可参考下面的 main 函数写法。

```
int main()
{
    string raw_image = "d:\\bjfu_gray.bmp";

    // 声明一个基类指针
    Image* pImg = 0;

    // 假设程序可以处理多种图像格式，可根据后缀名判断类型，创建不同的派生类对象
    // 获取文件后缀
    string ext = raw_image.substr(raw_image.find_last_of('.') + 1);

    // 创建具体的图像类对象
    if (ext == "bmp")//注意大小写
    {
        pImg = new Bitmap;
    }
    else if (ext == "jpg" || ext == "jpeg")
    {
        // 实际上我们并未实现 JPG 类来支持 JPG 格式图像
        // 但可以先把结构搭建好，待以后实现
        //pImg = new Jpg;
    }
    else
    {
        cout << "不支持格式：" << ext << "，异常退出..." << endl;
        return -1;
    }

    // 开始利用基类指针进行处理
    // 打开文件
    pImg->read_image(raw_image);

    // 输出图像信息
    pImg->print_info();

    // 最后的清理工作，注意与虚析构函数一起使用
    // 因 pImg 是基类指针，若不声明为虚析构函数，则派生类析构函数不会被调用
    delete pImg;

    return 0;
}
```

步骤 4：检查发际线。

思考：

- 基类什么函数应声明为虚函数/纯虚函数？两种函数类型的区别是什么？
- 如果再从 CImage 类派生出 JPG 格式的类，哪些函数是需要重写（override）的？
- 若小组共同开发，一人负责程序框架（main），一人设计抽象基类，一人

负责 Bitmap 支持，一人负责 JPG 支持，他们可以如何分工？从中理解面向对象程序设计和多态性的好处。

尝试在程序中做出相应修改看你的答案是否正确。

实验考核

1、实验完毕后上交实验报告，实验报告模板从 FTP 中下载，实验报告的内容包括实验目的、实验内容和结果分析，实验报告一律写成 Word 格式文档（**.doc/.docx**）。

2、将调试好的程序源代码和程序的输入输出数据的情况附到实验报告中，并对程序的数据结果进行分析，说明基本运行机理。

3、实验报告重点总结你的心得体会，遇到了什么问题，怎么解决的，对以后有什么帮助。

4、上传电子版实验报告和源程序（删除 Debug、Release 目录和其他无用中间文件，仅保留解决方案、工程等文件和程序代码文件）到 FTP 服务器 homework 目录下对应课程、班级目录中。多个文件请压缩打包，请注意文件命名格式。

5、文件打包命名为“**你的学号_姓名_s3**”。

6、鼓励提问、交流、讨论，严禁相互复制、抄袭，违者按 0 分处理。