**实验8：继承、多态和异常处理**

**姓名\_\_金宣成\_\_\_班级\_\_计科03\_\_\_\_学号\_202203151310\_**

* **请阅读此说明：实验8满分100分。做完实验后请按要求将代码和截图贴入该文档。然后将此文档、源代码文件（.hpp, .cpp）打包上传到学习通。**

**实验目的：熟悉并掌握继承和动态多态的概念。能够利用基类指针或者基类的引用结合虚函数实现动态多态；理解抽象类；**

**实验要求：按照每个类两个文件的方式（一个头文件，一个源文件）组织工程内的代码。**

**实验内容：**

1. **多重继承和虚基类（重现课堂例程）**

**理解多重继承模型下派生类对象的构造和析构的工作顺序；理解虚基类继承模型下派生类对象的构造和析构的工作顺序。**

**1、多态-虚函数和抽象类：重现ppt 中（详见附件）的继承层次: Point/Circle/Cylinder ,**

**建立下列主函数测试这个继承模型。**

**int main()**

**{**

**Point point(3.5,6.4), \*p;**

**Circle circle(4,5,6), \*cir;**

**Cylinder cylinder(5,6,8), \*cyl;**

**cout<<point;**

**cout<<circle;**

**cout<<cylinder;**

**p=&point;**

**cir=&circle;**

**cyl=&cylinder;**

**cout<<(\*p);**

**cout<<(\*cir);**

**cout<<(\*cyl);**

**//+++++++++++++++++++**

**cout<<cir->area()<<endl;**

**cir=&cyclinder;**

**cout<<cir->area()<<endl;**

**//+++++++++++++++++++**

**return 0;**

**}**

1. 整合程序，使主函数可以运行；运行结果是什么？
2. 为Circle类的area()成员声明添加virtual, 再运行主函数，运行结果是什么？
3. 为Point类添加area()成员:

float area() const { return 0;}

为主函数添加下列代码，可以执行

p=&circle;

cout<<p->area()<<endl;

p=&cylinder;

cout<<p->area()<<endl;

point &p1=circle;

cout<<p.area()<<endl;

point &p2=cylinder;

cout<<p2.area()<<endl;

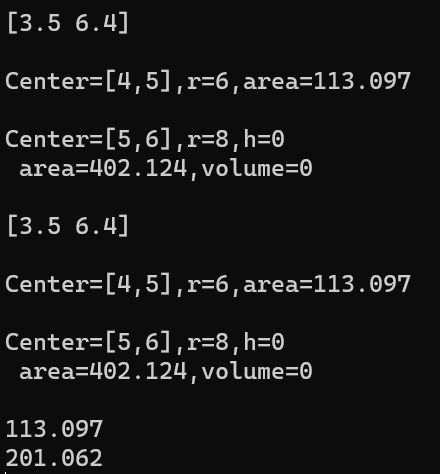
4) 在Point类添加的area()成员前加virtual

再运行主函数，运行结果是什么？

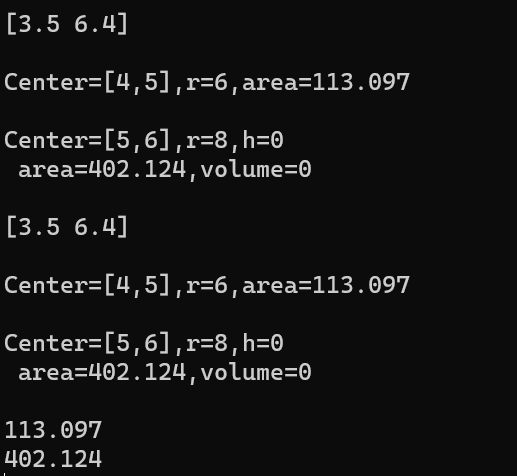
5）在前面设计的继承模型基础上，设计基于Point的其他子类：矩形Rect、三角形Triangle；设计基于Circle的子类球体Sphere、基于矩形类的立方体类Cube。提交设计完毕的类声明及定义附件。

**实验提交：**

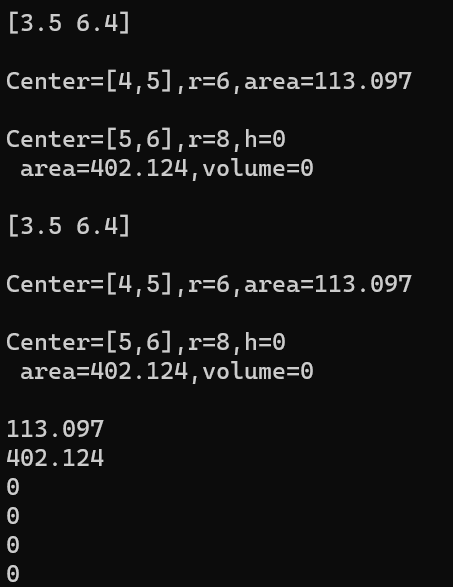
1. **运行结果截屏**

****

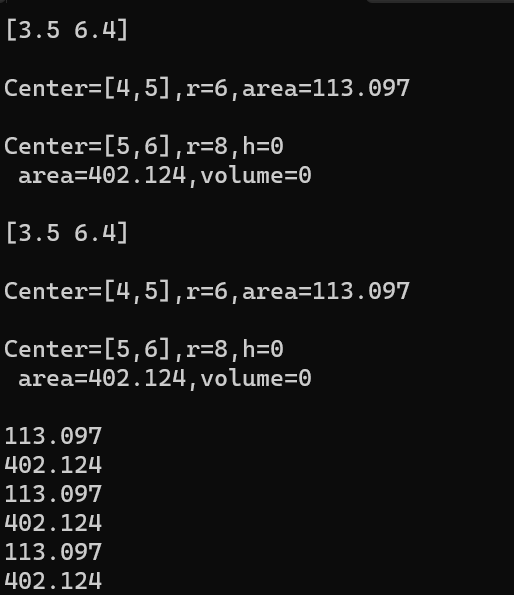
1. **运行结果截屏**

****

1. **运行结果截屏**

****

1. **运行结果截屏**

****

1. **各个补充类型的定义：**

* **矩形类Rect：**

#ifndef ADDED\_RECT\_H

#define ADDED\_RECT\_H

#include "point.hpp"

class Rect:public Point

{

    public:

        Rect(float = 0,float = 0,float = 0,float =0);//构造函数

        void setLength(float);//设置矩形长

        float getLength() const;//读取矩形长

        void setWidth(float);//设置矩形宽

        float getWidth() const;//读取矩形宽

        float area() const;//计算矩形面积

    protected:

        float length;

        float width;

};

#endif

* **三角形类Triangle：**

#ifndef ADDED\_TRIANGLE\_H

#define ADDED\_TRIANGLE\_H

#include "point.hpp"

class Triangle

{

public:

    Triangle(float = 0, float = 0, float = 0, float = 0, float = 0, float = 0); //构造函数

    void setpoint1(float, float);                                               //设置三角形第一个点

    void setpoint2(float, float);                                               //设置三角形第二个点

    void setpoint3(float, float);                                               //设置三角形第三个点

    float getpoint1x() const;                                                   //读取三角形第一个点的x坐标

    float getpoint1y() const;                                                   //读取三角形第一个点的y坐标

    float getpoint2x() const;                                                   //读取三角形第二个点的x坐标

    float getpoint2y() const;                                                   //读取三角形第二个点的y坐标

    float getpoint3x() const;                                                   //读取三角形第三个点的x坐标

    float getpoint3y() const;                                                   //读取三角形第三个点的y坐标

    float getdistance12() const;                                                //读取三角形第一条边的长度

    float getdistance13() const;                                                //读取三角形第二条边的长度

    float getdistance23() const;                                                //读取三角形第三条边的长度

    float area() const;                                                         //计算三角形面积

    float volume() const;                                                       //计算三角形体积

protected:

    Point point1;

    Point point2;

    Point point3;

};

#endif

* **球体类Sphere：**

#ifndef ADDED\_SPHERE\_H

#define ADDED\_SPHERE\_H

#include "circle.hpp"

class Sphere:public Circle{

    public:

        Sphere(float = 0,float = 0,float = 0,float = 0);//构造函数

        void setHeight(float);//设置球体高

        float getHeight() const;//读取球体高

        float area() const;//计算球体表面积

        float volume() const;//计算球体体积

    protected:

        float height;

};

#endif

* **立方体类Cube：**

#ifndef ADDED\_CUBE\_H

#define ADDED\_CUBE\_H

#include "added\_Rect.hpp"

class Cube : public Rect

{

    public:

        Cube(float = 0,float = 0,float = 0,float = 0,float = 0);//构造函数

        void setHeight(float);//设置立方体高

        float getHeight() const;//读取立方体高

        float area() const;//计算立方体表面积

        float volume() const;//计算立方体体积

    protected:

        float height;

};

#endif

6)体验使用基类引用体现多态的应用场合。

**Step1：**为Point类，Circle类，Cylinder类设计同名成员函数display()，原型为：

void display(ostream& out) const;

该接口完成不同类的数据信息的输出。不同类的display实现参考如下：

void Point::display(ostream& out) const{  
 cout<<”Point:”<<x<<”,”<<y<<endl;

}

void Circle::display(ostream& out)const{

cout<<”Circle:”<<x<<”,”<<y<<”; radius:”<<radius<<”; area:”<<area()<<endl;

}

void Cylinder::display(ostream& out) const{

cout<<”Cylinder:”<<x<<”,”<<y<<”;radius:”<<radius<<”;area:”<<area()

<<”;volumn:”<<volumn()<<endl;

}

**Step2：**并在基类Point中限定display为虚函数。

**Step3：**修改基类Point的输出流重载函数为：

ostream& operator<<(ostream& out, const Point& p){

p.display(out);

return out;

}

**Step4：**注释/删除 类内友元声明；注释/删除Circle，Cylinder的输出流重载函数。

**Step5：**运行测试程序，特别注意观察：

**cout<<point;**

**cout<<circle;**

**cout<<cylinder;**

**这样的输出流重载的使用，和原来有什么不同？体会基类引用的多态应用场合。**

7）在主函数中增加以下代码，观察输出，比较rp1,rp2工作的差异；以及和在6）中step3定义的输出流运算符重载中形参p使用的差别。

**Point &rp1=point;**

**rp1=circle;**

**rp1.display();**

**point.display();**

**Point &rp2=circle;**

**rp2.display();**

8）增加一个抽象类Shape

class shape{

public:

virtual float area()=0;

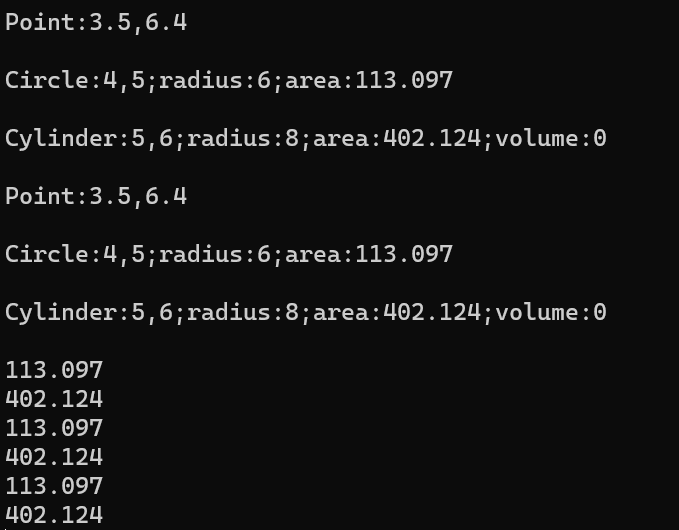
};

让Point成为Shape的子类。看看对主函数有什么影响？

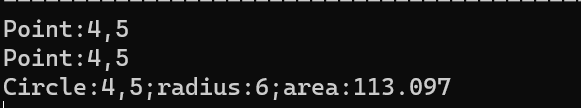
将Point中的area（）成员删除，看看有什么影响？

**实验提交：**

**1)运行结果截屏**

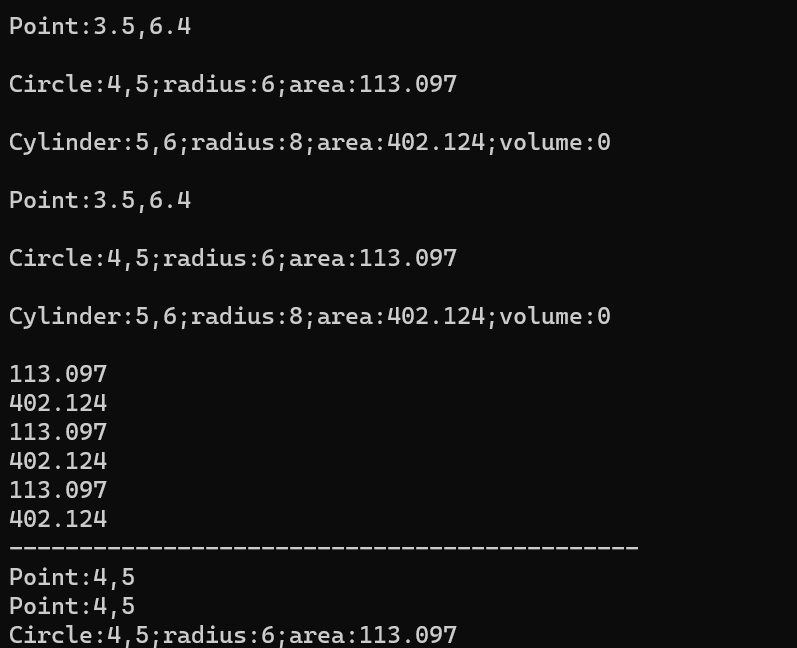
****

**2)运行结果截屏**

****

**总结：引用先作为哪个类型的别名matters**

**3)运行结果截屏**



现象：删除了point中的area()之后，直接继承得来的虚函数没有重定义的话，是无法初始化实例的。

**2、虚析构:见实验内容2**

使用不同的main函数，观察输出结果。掌握虚成员函数，和虚析构函数。理解虚析构和一般的成员函数的不同。掌握虚析构的使用场景。

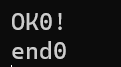
**3、异常处理（try、throw、catch）**

**3-1:**

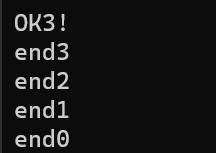
1. 例程example1.cpp，输出结果是什么？请运行程序验证你的想法。理解throw的数据类型和catch捕获异常类型之间的关系。然后分别做 2) 3) 4)。
2. 将f3()中的catch(float)改为catch(double)，思考输出结果是什么？请运行程序验证你的想法。
3. 将f2()中的catch(int)改为catch(double)，思考输出结果是什么？请运行程序验证你的想法。
4. 将f1()中的catch(char)改为catch(double)，思考输出结果是什么？请运行程序验证你的想法。

**实验提交：**

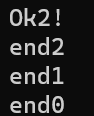
**1)运行结果截屏**

****

**2)运行结果截屏**

****

**3)运行结果截屏**

****

**4)运行结果截屏**

****

**3-2:**

1. 例程example2.cpp，输出结果是什么？请运行程序验证你的想法。
2. 使用下列语句 ：

Student stud1(1101, "Tan"); //建立对象stud1

Student stud2(0, "Li"); //建立对象stud2

fun1(stud1);

fun1(stud2);

替换主函数中fun0的调用，输出结果是什么？请运行程序验证你的想法。

1. 使用下列语句 ：

Student stud1(1101, "Tan"); //建立对象stud1

Student stud2(0, "Li"); //建立对象stud2

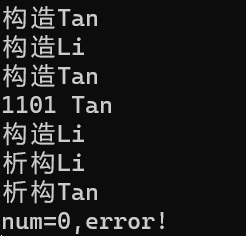
fun2(stud1);

fun2(stud2);

替换主函数中 fun0的调用，输出结果是什么？请运行程序验证你的想法。

**实验提交：**

**1)运行结果截屏**

****

**2)运行结果截屏**

****

1. **运行结果截屏**

****

**\*4、选做内容：名空间（namespace）**

组装example3中的文件，观察错误信息。使用namespace 去除名冲突。理解namesapce 的含义和用法。给出修改后的程序。

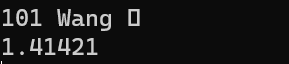
实验提交：

* 含有的错误信息：

redefinition of 'class Student'

* 修改后的程序及运行截图：





Header1.hpp

// header1.h (头文件1，设其文件名为example2.h)

#include <string>

#include <cmath>

using namespace std;

namespace header1{

    class Student //声明Student类

    {

    public:

        Student(int n, string nam, char s)

        {

            num = n;

            name = nam;

            sex = s;

        }

        void get\_data();

    private:

        int num;

        string name;

        char sex;

    };

    void Student::get\_data() //成员函数定义

    {

        cout << num << "" << name << "" << sex << endl;

    }

    double fun(double a, double b) //定义全局函数(即外部函数)

    {

        return sqrt(a + b);

    }

}

Header2.hpp

// header2.h (头文件2，设其文件名为example2\_header2.h)

#include <string>

#include <cmath>

using namespace std;

namespace header2{

    class Student //声明Student类

    {

    public:

        Student(int n, string nam, char s) //参数与header1中的student不同

        {

            num = n;

            name = nam;

            sex = s;

        }

        void get\_data();

    private:

        int num;

        string name;

        char sex; //此项与header1不同

    };

    void Student::get\_data() //成员函数定义

    {

        cout << num << " " << name << " " << sex << endl;

    }

    double fun(double a, double b) //定义全局函数

    {

        return sqrt(a - b);

    } //返回值与header1中的fun函数不同

    //头文件2中可能还有其他内容

}

Test.cpp

//在main函数所在的文件中包含头文件header1.h:

#include <iostream>

#include "example3\_header2.h"

#include "example3\_header1.h"

//注意要用双引号，因为文件一般是放在用户目录中的

using namespace std;

int main()

{

    header2::Student stud1(101, "Wang", 18); //定义类对象stud1

    stud1.get\_data();

    cout << header2::fun(5, 3) << endl;

    return 0;

}