实验8：继承多态和异常处理

姓名 陈王子

班级 数据科学与大数据技术(分析方向)2101班

学号 202103150503

* 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;

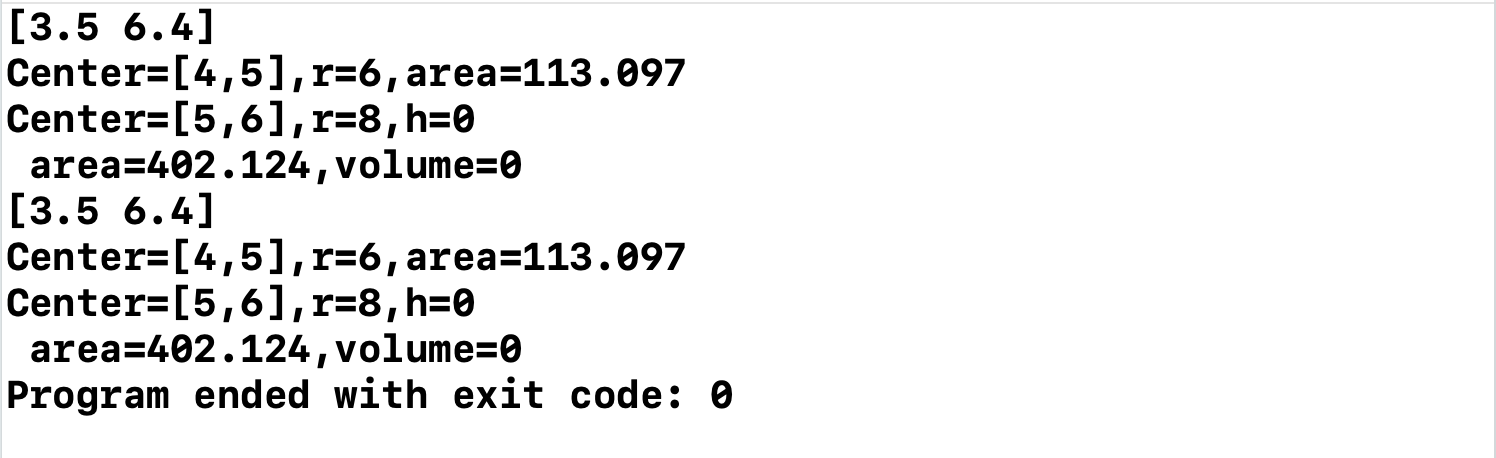
4) 在Point类添加的area()成员前加virtual

再运行主函数，运行结果是什么？

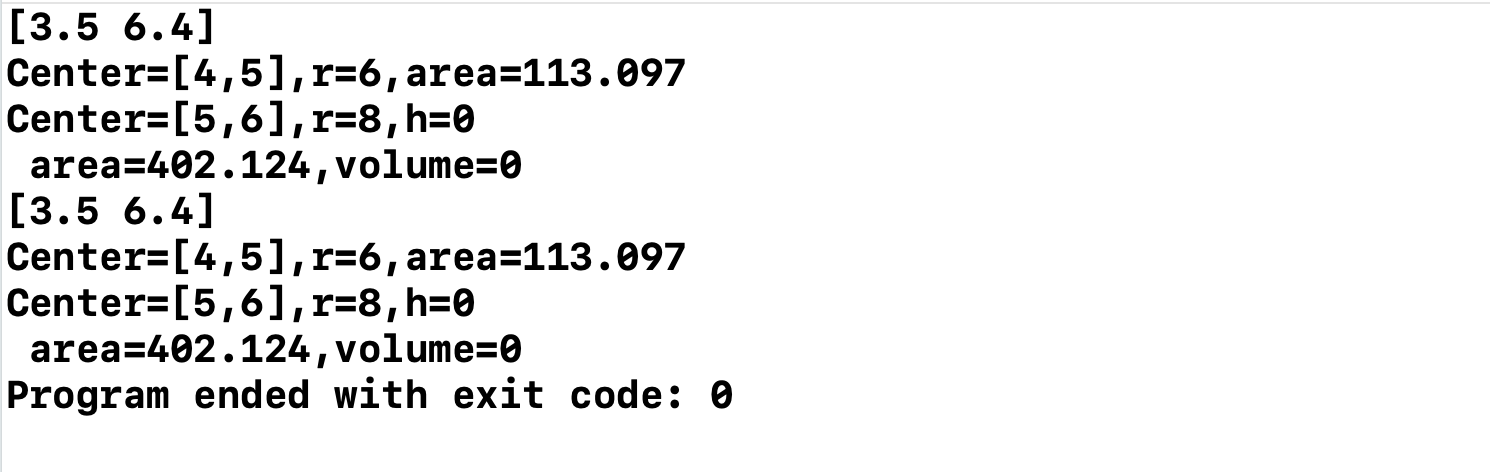
5）在前面设计的继承模型基础上，设计基于Point的其他子类：矩形Rect、三角形Triangle；设计基于Circle的子类球体Sphere、基于矩形类的立方体类Cube。提交设计完毕的类声明及定义附件。

实验提交：

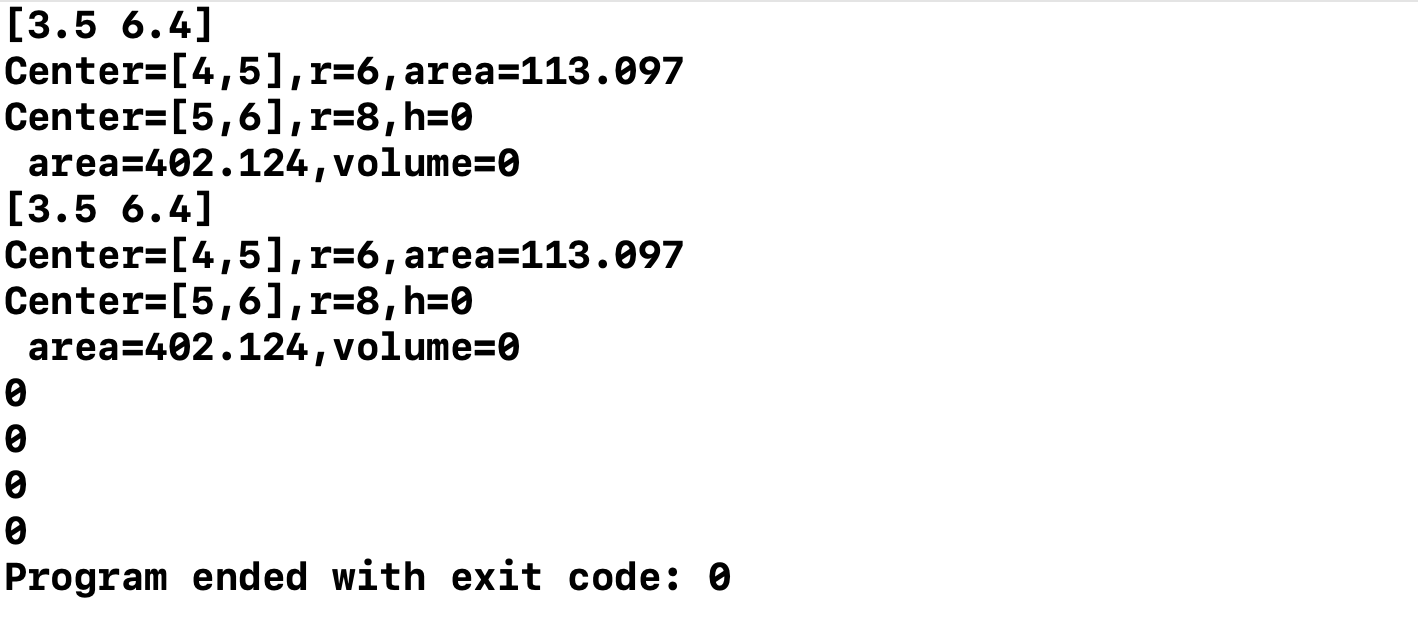
1. 运行结果截屏



1. 运行结果截屏



1. 运行结果截屏



#ifndef RECT\_H    // 防止头文件重复包含

#define RECT\_H

#include <iostream>

#include "point.h"

class Rect : public Point {

public:

    Rect(float x = 0, float y = 0, float length = 0, float width = 0); // 构造函数

    void setLength(float);  // 设置长

    void setWidth(float);   // 设置宽

    float getLength() const;// 读取长

    float getWidth() const; // 读取宽

    virtual float area() const override; // 计算矩形面积

    friend std::ostream& operator<<(std::ostream&, const Rect&);

protected:

    float length, width;

};

#endif // RECT\_H

#include "rect.h"

Rect::Rect(float x, float y, float length, float width) : Point(x, y), length(length), width(width) {}

void Rect::setLength(float length){

    this->length = length;

}

void Rect::setWidth(float width){

    this->width = width;

}

float Rect::getLength() const{

    return length;

}

float Rect::getWidth() const{

    return width;

}

float Rect::area() const{

    return length \* width;

}

std::ostream& operator<<(std::ostream& output, const Rect & R){

    output << "Center=[" << R.x << "," << R.y << "], length=" << R.length << ", width=" << R.width << ", area=" << R.area() << std::endl;

    return output;

}

#ifndef TRIANGLE\_H

#define TRIANGLE\_H

#include <iostream>

#include <cmath>

#include "point.h"

class Triangle : public Point {

public:

    Triangle(float x = 0, float y = 0, float x2 = 0, float y2 = 0, float x3 = 0, float y3 = 0); // 构造函数

    void setX2Y2(float x, float y); // 设置(x2,y2)点

    void setX3Y3(float x, float y); // 设置(x3,y3)点

    float getX2() const; // 读取x2

    float getY2() const; // 读取y2

    float getX3() const; // 读取x3

    float getY3() const; // 读取y3

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

    friend std::ostream& operator<<(std::ostream&, const Triangle&);

protected:

    float x2, y2, x3, y3;

};

#endif // TRIANGLE\_H

#include "triangle.h"

Triangle::Triangle(float x, float y, float x2, float y2, float x3, float y3) : Point(x, y), x2(x2), y2(y2), x3(x3), y3(y3) {}

void Triangle::setX2Y2(float x, float y){

    x2 = x;

    y2 = y;

}

void Triangle::setX3Y3(float x, float y){

    x3 = x;

    y3 = y;

}

float Triangle::getX2() const{

    return x2;

}

float Triangle::getY2() const{

    return y2;

}

float Triangle::getX3() const{

    return x3;

}

float Triangle::getY3() const{

    return y3;

}

float Triangle::area () const{

    float a = sqrt(pow(x3 - x2, 2) + pow(y3 - y2, 2));

    float b = sqrt(pow(x2 - getX(), 2) + pow(y2 - getY(), 2));

    float c = sqrt(pow(x3 - getX(), 2) + pow(y3 - getY(), 2));

    float p = (a + b + c) / 2;

    return sqrt(p \* (p - a) \* (p - b) \* (p - c));

}

std::ostream& operator<<(std::ostream& output,const Triangle &T){

    output << "(x,y)=[" << T.getX() << "," << T.getY() << "]" << std::endl;

    output << "(x2,y2)=[" << T.getX2() << "," << T.getY2() << "]" << std::endl;

    output << "(x3,y3)=[" << T.getX3() << "," << T.getY3() << "]" << std::endl;

    output << "area=" << T.area() << std::endl;

    return output;

}

#ifndef Sphere\_hpp

#define Sphere\_hpp

#include <iostream>

#include "circle.hpp"

class Sphere : public Circle {

public:

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

    void setR(float); // 设置圆半径

    float getR() const; // 读取圆半径

    float area() const override; // 计算球面积

    float volume() const; // 计算球体积

    friend std::ostream& operator<<(std::ostream&, const Sphere&); // 重载输出运算符

};

#endif /\* Sphere\_hpp \*/

#include "Sphere.hpp"

Sphere::Sphere(float a, float b, float r) : Circle(a, b, r) {}

void Sphere::setR(float r){

    radius = r;

}

float Sphere::getR() const{

    return radius;

}

float Sphere::area() const{

    return 4 \* 3.14159 \* radius \* radius;

}

float Sphere::volume() const{

    return (4 / 3) \* 3.14159 \* radius \* radius \* radius;

}

std::ostream& operator<<(std::ostream& output, const Sphere& S){

    output << "Center=[" << S.getX() << "," << S.getY() << "],r=" << S.radius << std::endl;

    output << "area=" << S.area() << ",volume=" << S.volume() << std::endl;

    return output;

}

#ifndef Cube\_hpp

#define Cube\_hpp

#include <stdio.h>

#include "rect.hpp"

class Cube:public Rect{

public:

    Cube(float x=0,float y=0,float chang=0,float kuan=0,float height=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

    friend ostream& operator<<(ostream&,const Cube&);//重载输出运算符

protected:

    float height; //立方体高

};

#endif /\* Cube\_hpp \*/

#include "Cube.hpp"

Cube::Cube(float a,float b,float ch,float k,float h)//构造函数

{

    x = a;

    y = b;

    chang = ch;

    kuan = k;

    height = h;

}

void Cube::setHeight(float h)//设置立方体高

{

    height = h;

}

float Cube::getHeight( ) const //读取立方体高

{

    return height;

}

float Cube::area( ) const //计算立方体表面积

{

    return 2\*(chang\*kuan)+2\*(chang\*height)+2\*(kuan\*height);

}

float Cube::volume( ) const //计算立方体体积

{

    return Rect::area()\*height;

}

ostream& operator<<(ostream& output,const Cube& cu)//重载输出运算符

{

    output<<"chang="<<cu.chang<<",kuan="<<cu.kuan<<",height="<<cu.height<<endl;

    output<<"area="<<cu.area()<<",volume="<<cu.volume()<<endl;

    return output;

}

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：

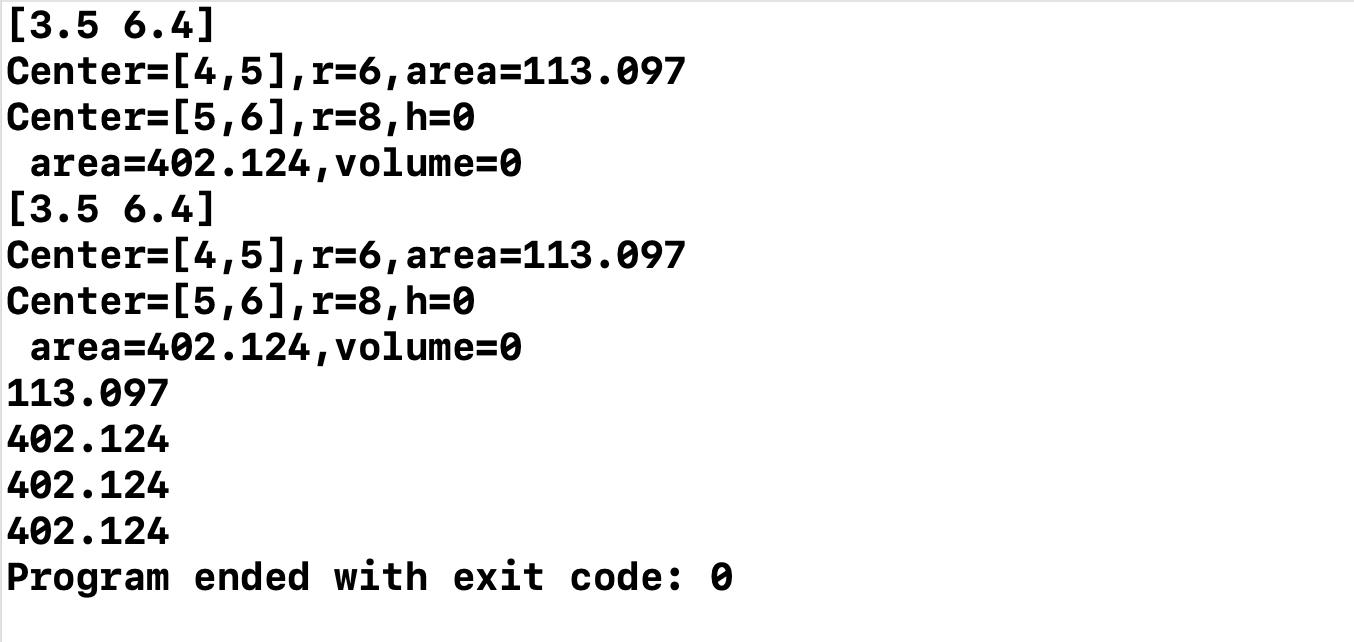
ostream& operator<<(ostream& out, const Point& p){

p.display(out);

return out;

}

1. 运行结果截屏



1. 各个补充类型的定义：

* 矩形类Rect：

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;

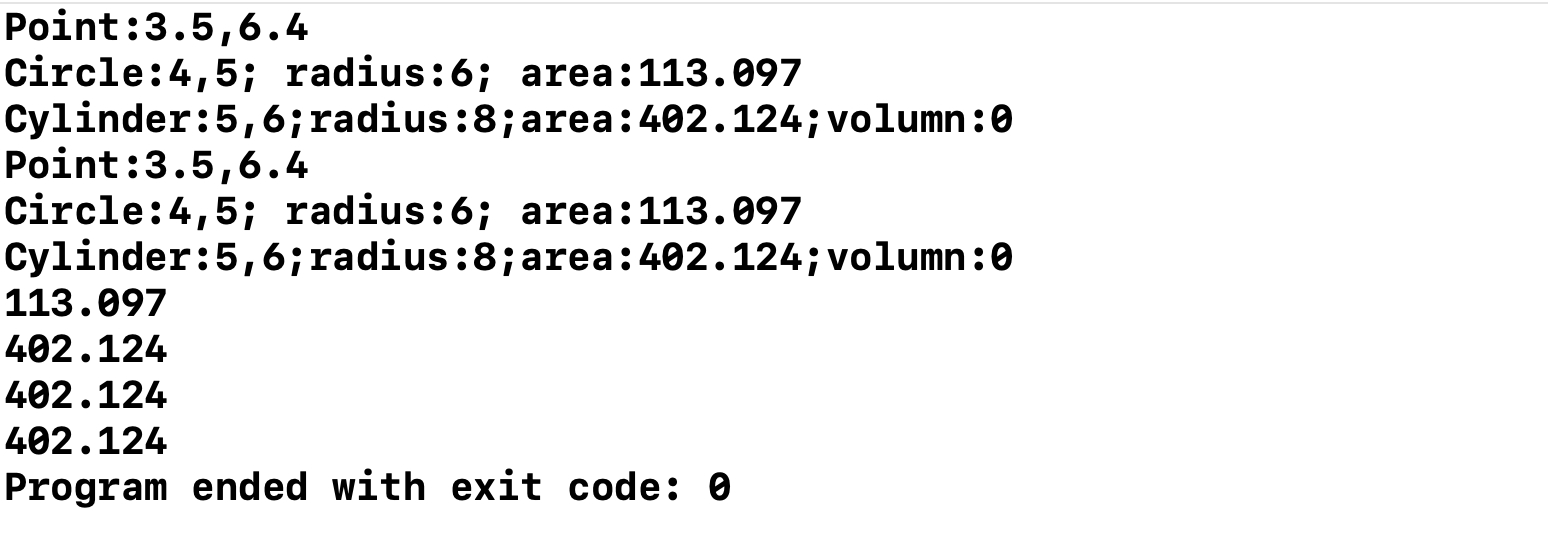
};

让Point成为Shape的子类。看看对主函数有什么影响？

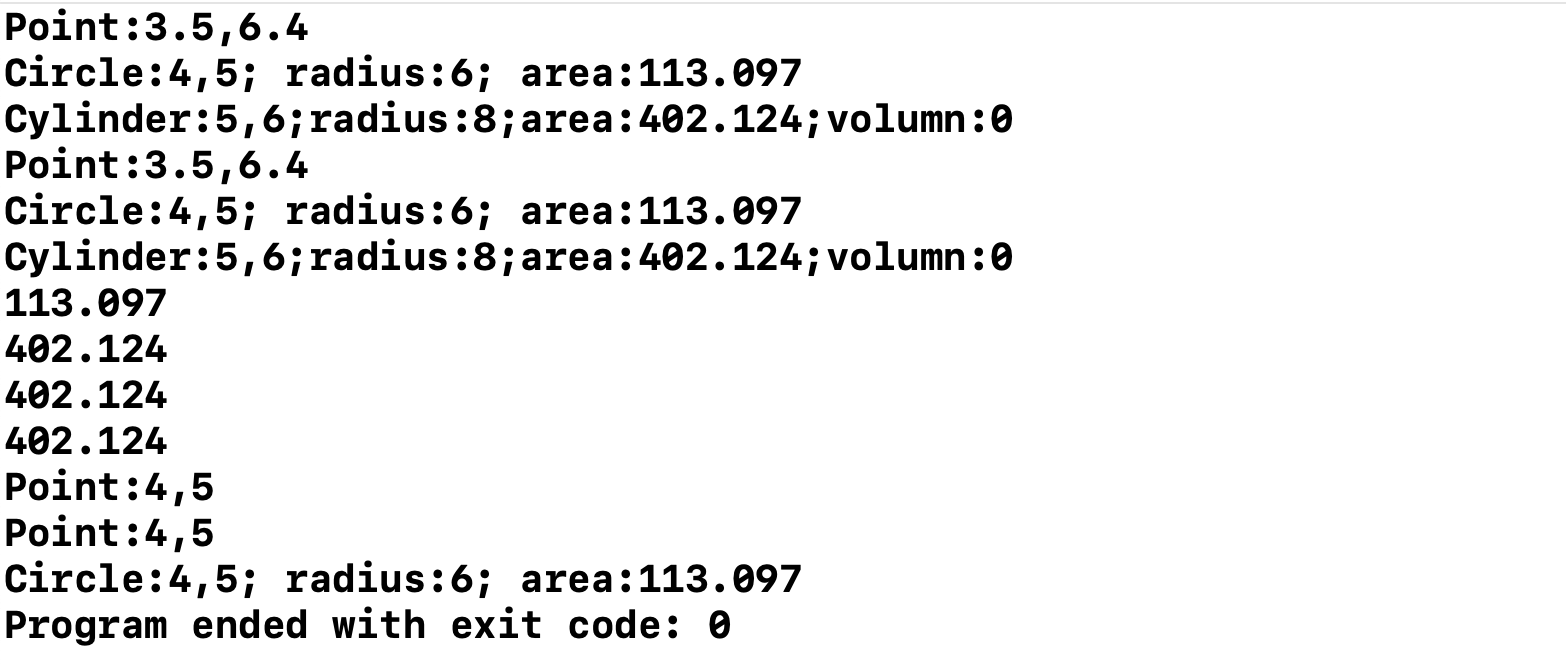
将Point中的area（）成员删除，看看有什么影响？

实验提交：

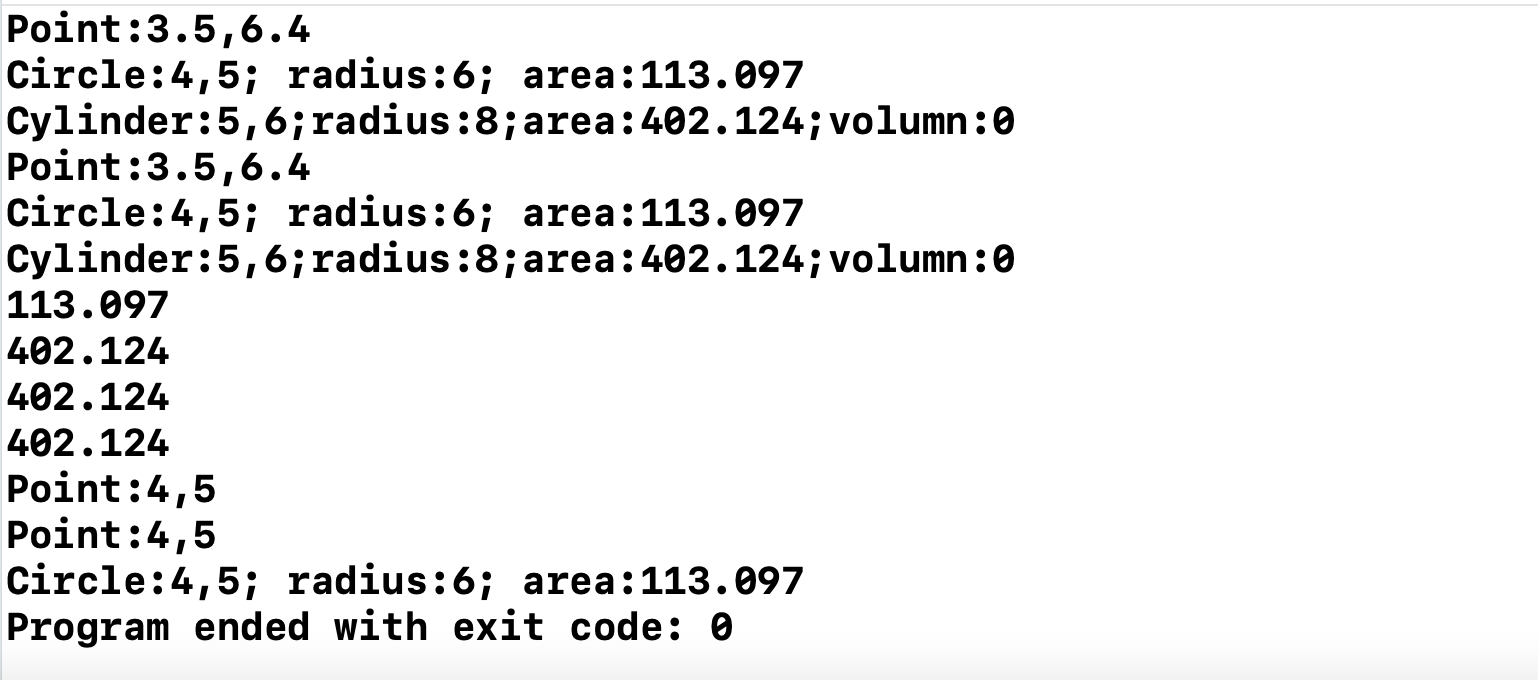
1)运行结果截屏



2)运行结果截屏



3)运行结果截屏



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:

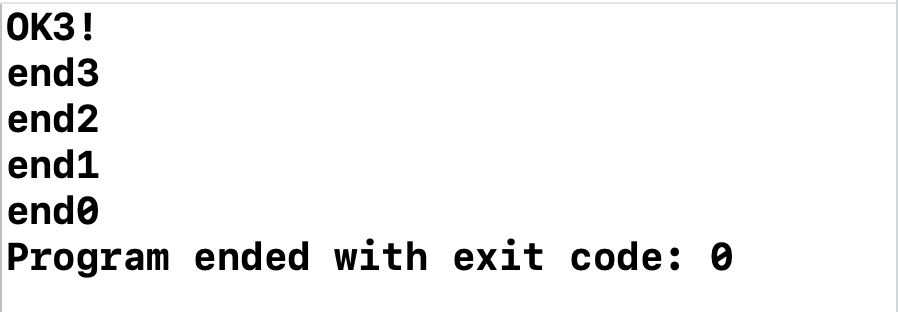
1. 例程example1.cpp，输出结果是什么？请运行程序验证你的想法。理解throw的数据类型和catch捕获异常类型之间的关系。然后分别做 2) 3) 4)。
2. 将f3()中的catch(float)改为catch(double)，思考输出结果是什么？请运行程序验证你的想法。
3. 将f2()中的catch(int)改为catch(double)，思考输出结果是什么？请运行程序验证你的想法。
4. 将f1()中的catch(char)改为catch(double)，思考输出结果是什么？请运行程序验证你的想法。

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

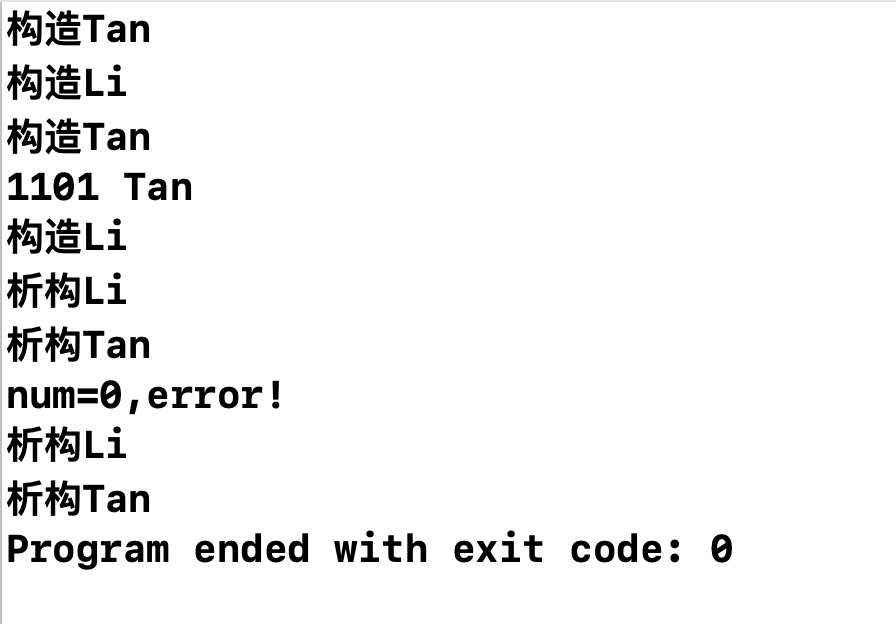
fun2(stud1);

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



3)运行结果截屏

