

**C++程序设计实验报告**



实验题目 多态性与虚函数

学 号 201806061108

姓 名 胡皓睿

班 级 计实1801

提交日期 2019/6/24

1. 实验目的：

1. 掌握虚函数的定义和使用

2. 掌握多态性的定义与使用

1. 实验内容：
2. 在例12.1程序基础上作一些修改。声明point（点）类，由point类派生出circle（圆）类，再由circle类派生出cylinder（圆柱体）类。将类的声明部分分别作为3个头文件，对它们的成员函数的声明部分分别作为3个源文件（。cpp文件），在主函数中用“#include”命令把他们包含进来，形成一个完整的程序，并上机运行。

2、在例12.3的基础上作以下修改，并作必要的讨论。

（1）把构造函数修改为带参数的函数，在建立对象时初始化。

(2)先不将析构函数声明为virtual，在main函数中另设一个指向circle类对象的指针变量，使它指向grad1。运行程序，分析结果。

(3)不作第（2）点的修改而将析构函数声明为virtual，运行程序，分析结果。

3、编写一个程序，声明抽象基类shape，由它派生出三个派生类：circle（圆形）、rectangle（矩形）、triangle（三角形）。用一个函数printarea分别输出以上三者的面积，三个图形的数据在定义对象时给定。

4、编写一个程序，声明抽象基类shape，由它派生出三个派生类：circle（圆形）、square（正方形）、rectangle（矩形）、trapezoid（梯形）、triangle（三角形）。用虚函数分别就算几种图形，并求他们之和。要求用基类指针数组，使它每一个元素指向一个派生类对象。

三、实验过程

1. 1、在例12.1程序基础上作一些修改。声明point（点）类，由point类派生出circle（圆）类，再由circle类派生出cylinder（圆柱体）类。将类的声明部分分别作为3个头文件，对它们的成员函数的声明部分分别作为3个源文件（。cpp文件），在主函数中用“#include”命令把他们包含进来，形成一个完整的程序，并上机运行。
2. 程序代码

//Point.h

#ifndef POINT\_H

#define POINT\_H

#include <iostream>

using namespace std;

class Point

{

public:

Point(float = 0, float = 0);

void setPoint(float, float);

float getX() const { return x; }

float getY() const { return y; }

friend ostream& operator<<(ostream&, const Point&);

protected:

float x, y;

};

#endif

//Circle.h

#ifndef CIRCLE\_H

#define CIRCLE\_H

#include"point.h"

using namespace std;

class Circle :public Point

{

public:

Circle(float x = 0, float y = 0, float r = 0);

void setRadius(float);

float getRadius() const;

float area() const;

friend ostream& operator<<(ostream&, const Circle&);

protected:

float radius;

};

#endif

//Cylinder.h

#ifndef CYLINDER\_H

#define CYLINDER\_H

#include<iostream>

#include"circle.h"

using namespace std;

class Cylinder :public Circle

{

public:

Cylinder(float x = 0, float y = 0, float r = 0, float h = 0);

void setHeight(float);

float getHeight() const;

float area() const;

float volume() const;

friend ostream& operator<<(ostream&, const Cylinder&);

protected:

float height;

};

#endif

//point.cpp

#include"point.h"

Point::Point(float a, float b)

{

x = a;

y = b;

}

void Point::setPoint(float a, float b)

{

x = a;

y = b;

}

ostream& operator<<(ostream& output, const Point& p)

{

output << "[" << p.x << "," << p.y << "]" << endl;

return output;

}

//circle.cpp

#include"circle.h"

#include<iostream>

using namespace std;

Circle::Circle(float a, float b, float r) :Point(a, b), radius(r) {

}

void Circle::setRadius(float r)

{

radius = r;

}

float Circle::getRadius() const { return radius; }

float Circle::area() const

{

return 3.14159 \* radius \* radius;

}

ostream& operator<<(ostream& output, const Circle& c)

{

output << "Center=[" << c.x << "," << c.y << "],r=" << c.radius << ",area=" << c.area() << endl;

return output;

}

//cylinder.cpp

#include"cylinder.h"

#include<iostream>

Cylinder::Cylinder(float a, float b, float r, float h)

:Circle(a, b, r), height(h) {

}

void Cylinder::setHeight(float h)

{

height = h;

}

float Cylinder::getHeight() const { return height; }

float Cylinder::area() const

{

return 2 \* Circle::area() + 2 \* 3.14159 \* radius \* height;

};

float Cylinder::volume() const

{

return Circle::area() \* height;

};

ostream& operator<<(ostream& output, const Cylinder& cy)

{

output << "Center=[" << cy.x << "," << cy.y << "],r=" << cy.radius << ",h=" << cy.height << "\narea=" << cy.area() << ",volume=" << cy.volume() << endl;

return output;

}

//main.cpp

#include <iostream>

#include"point.h"

#include"circle.h"

#include"cylinder.h"

using namespace std;

int main()

{

Cylinder cy1(3.5, 6.4, 5.2, 10);

cout << "originalcylinder:\nx=" << cy1.getX() << ",y=" << cy1.getY() << ",=" << cy1.getRadius() << ",h=" << cy1.getHeight() << "\narea=" << cy1.area() << ",volume=" << cy1.volume() << endl;

cy1.setHeight(15);

cy1.setRadius(7.5);

cy1.setPoint(5.5, 6.4);

cout << "\nnew cylinder:\n" << cy1;

Point& pRef = cy1;

cout << "\npRef as a point:" << pRef;

Circle& cRef = cy1;

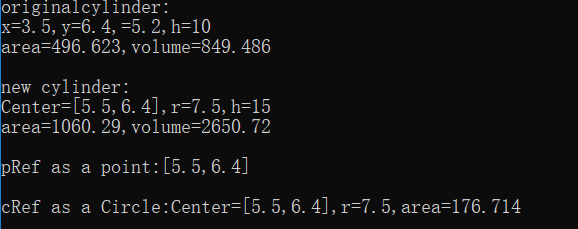
cout << "\ncRef as a Circle:" << cRef;

return 0;

}

（2）运行结果

outtput:



1. 结果分析

需要使用#ifndef来保证源文件只被编译一次

2、 在例12.3的基础上作以下修改，并作必要的讨论。

（1）把构造函数修改为带参数的函数，在建立对象时初始化。

（2）先不将析构函数声明为virtual，在main函数中另设一个指向circle类对象的指针变量，使它指向grad1。运行程序，分析结果。

（3）不作第（2）点的修改而将析构函数声明为virtual，运行程序，分析结果。

（1）程序代码

1、

#include <iostream>

using namespace std;

class point

{

public:

point(int x, int y) :x(x), y(y) {};

point() { cout << "excuting point destructor" << endl; }

private:

int x, y;

};

class circle :public point

{

public:

circle(int x, int y, int r) :point(x, y), r(r) {};

circle() { cout << "excuting circle destructor" << endl; }

private:

int r;

};

int main()

{

point\* p = new circle(1, 1, 3);

delete p;

}

2.

#include <iostream>

using namespace std;

class point

{

public:

point(int x, int y) :x(x), y(y) {};

~point() { cout << "excuting point destructor" << endl; }

private:

int x, y;

};

class circle :public point

{

public:

circle(int x, int y, int r) :point(x, y), r(r) {};

~circle() { cout << "excuting circle destructor" << endl; }

private:

int r;

};

int main()

{

circle\* c = new circle(1, 1, 3);

point\* p = c;

delete p;

}

3、

#include <iostream>

using namespace std;

class point

{

public:

point(int x, int y) :x(x), y(y) {};

virtual ~point() { cout << "excuting point destructor" << endl; }

private:

int x, y;

};

class circle :public point

{

public:

circle(int x, int y, int r) :point(x, y), r(r) {};

virtual ~circle() { cout << "excuting circle destructor" << endl; }

private:

int r;

};

int main()

{

point\* p = new circle(1, 1, 3);

delete p;

}

（2）运行结果

Output：

1.

没有输出

2.



3.



3、编写一个程序，声明抽象基类shape，由它派生出三个派生类：circle（圆形）、rectangle（矩形）、triangle（三角形）。用一个函数printarea分别输出以上三者的面积，三个图形的数据在定义对象时给定。

（1）程序代码

#include <iostream>

using namespace std;

class shape

{

public:

virtual double area() = 0;

};

class circle :public shape {

protected:

double r;

public:

circle(double r) :r(r) {};

virtual double area() { return 3.14 \* r \* r; }

};

class rectangle :public shape

{

protected:

double a, b;

public:

rectangle(double a, double b) :a(a), b(b) {};

virtual double area() { return a \* b; }

};

class triangle :public shape {

protected:

double h, w;

public:

triangle(float h, double w) :h(h), w(w) {};

virtual double area() { return h \* w \* 0.5; }

};

void show(shape& s)

{

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

}

int main()

{

circle c(4.5);

cout << "area:";

show(c);

rectangle r(4.5, 3.8);

cout << "area:";

show(r);

triangle t(4.5, 3.8);

cout << "area:";

show(t);

}

(2)运行结果：



（3）结果分析

虚函数的运用

1. 编写一个程序，声明抽象基类shape，由它派生出三个派生类：circle（圆形）、square（正方形）、rectangle（矩形）、trapezoid（梯形）、triangle（三角形）。用虚函数分别就算几种图形，并求他们之和。要求用基类指针数组，使它每一个元素指向一个派生类对象。
2. 程序代码

#include <iostream>

using namespace std;

class shape

{

public:

virtual double area() = 0;

};

class circle :public shape

{

protected:

double r;

public:

circle(double r) :r(r) {};

virtual double area() { return 3.14 \* r \* r; }

};

class rectangle :public shape

{

protected:

double a, b;

public:

rectangle(double a, double b) :a(a), b(b) {};

virtual double area() { return a \* b; }

};

class square :public shape

{

protected:

double a;

public:

square(double a) :a(a) {};

virtual double area() { return a \* a; }

};

class triangle :public shape

{

protected:

double h, w;

public:

triangle(float h, double w) :h(h), w(w) {};

virtual double area() { return h \* w \* 0.5; }

};

class trapezoid :public shape

{

protected:

double h, sw, lw;

public:

trapezoid(float h, double sw, double lw) :h(h), lw(lw), sw(sw) {};

virtual double area() { return h \* (lw + sw) \* 0.5; }

};

int main()

{

double sum = 0;

shape\* p;

circle c(4.5);

p = &c;

sum += (\*p).area();

cout << "area:" << (\*p).area() << endl;

rectangle r(4.5, 3.8);

p = &r;

sum += (\*p).area();

cout << "area:" << (\*p).area() << endl;

triangle tri(4.5, 3.8);

p = &tri;

sum += (\*p).area();

cout << "area:" << (\*p).area() << endl;

trapezoid tra(3.1, 5.4, 10.8);

p = &tra;

sum += (\*p).area();

cout << "area:" << (\*p).area() << endl;

square s(7.9);

p = &s;

sum += (\*p).area();

cout << "area:" << (\*p).area() << endl;

cout << "total:" << sum << endl;

}

1. 运行结果：



四、小结与收获

这次实验主要是基于对象的程序设计的一些相关知识。使我了解了虚函数的有关知识。