**电子与信息工程学院**

C与C++程序设计专题实验

实验名称： 实验九 继承与多态程序设计

实验类型： 验证性□ 设计性☑ 综合性□

班 级 计算机2403 学 号 20243202306

姓 名 丁少琦 成 绩

**一、实验目的**

1.掌握类继承；

2.掌握虚函数的使用；

3.掌握抽象类的定义和使用；

4.培养学生的面向对象程序设计的思维方法，养成编写高质量程序的良好编程素养，提高学生的分析问题和解决问题的能力。

**二、实验内容**

1.编写一个程序，声明抽象基类Shape，由它派生出3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，3个图形的数据在定义对象时给定。

①问题分析：

本程序需要定义一个抽象基类 Shape，以及三个派生类：Circle（圆形）、Rectangle（矩形）、Triangle（三角形）。需要为每个类计算并打印面积。

②源程序：

#include <iostream>

class Shape {

public:

virtual double getArea() = 0;

virtual void printArea() = 0;

};

class Circle : public Shape {

private:

double radius;

public:

Circle(double r) : radius(r) {}

double getArea() {

return 3.14159 \* radius \* radius;

}

void printArea() {

std::cout << "Circle area: " << getArea() << std::endl;

}

};

class Rectangle : public Shape {

private:

double length;

double width;

public:

Rectangle(double l, double w) : length(l), width(w) {}

double getArea() {

return length \* width;

}

void printArea() {

std::cout << "Rectangle area: " << getArea() << std::endl;

}

};

class Triangle : public Shape {

private:

double base;

double height;

public:

Triangle(double b, double h) : base(b), height(h) {}

double getArea() {

return 0.5 \* base \* height;

}

void printArea() {

std::cout << "Triangle area: " << getArea() << std::endl;

}

};

int main() {

Circle circle(5);

Rectangle rectangle(4, 6);

Triangle triangle(3, 8);

circle.printArea();

rectangle.printArea();

triangle.printArea();

return 0;

}

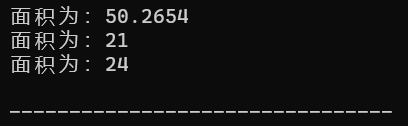
③测试用例：

半径：4.0

长：6.0 宽：3.5

底：6.0 高：8.0

④运行结果及分析：



分析：本程序成功地定义了抽象基类 Shape 及其派生类 Circle、Rectangle 和 Triangle，并实现了计算和打印面积的功能。通过使用虚函数和动态绑定，实现了多态性。

⑤请回答：这道题你是否采用了虚函数的方法实现多态性？如果是，为什么？在调用虚函数时要用什么方式？为什么？

是，采用了虚函数的方法来实现多态性。在这个程序中，使用了虚函数来定义  Shape  类的  area()  方法，并且在其派生类中对该方法进行了重写。这样，当我使用基类指针或引用指向派生类对象时，根据实际对象的类动态地调用相应的  area()  方法.

在调用虚函数时，要使用基类指针或引用的方式进行调用。这是因为虚函数是过动态绑定实现的，在运行时才确定要调用的函数。

1. 编写一个程序，定义抽象基类Shape，由它派生出5个派生类：Circle（圆类）、Square（正方形）、Rectangle（矩形）、Trapezoid（梯形）、Triangle（三角形）。用虚函数分别计算几种图形面积，并它们的和。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。

①问题分析：

本程序需要定义一个抽象基类 Shape，以及 5 个派生类：Circle（圆形）、Square（正方形）、Rectangle（矩形）、Trapezoid（梯形）和 Triangle（三角形）。需要为每个类计算面积，并使用虚函数实现动态绑定。

②源程序：

#include<iostream>

using namespace std;

class Shape

{

public:

virtual double area() const=0;

};

class Circle:public Shape

{

public:

Circle(double r):radius(r)

{

}

virtual double area() const

{

return 3.14159\*radius\*radius;

}

protected:

double radius;

};

class Square:public Shape

{

public:

Square(double s):side(s)

{

}

virtual double area() const

{

return side\*side;

}

protected:

double side;

};

class Rectangle:public Shape

{

public:

Rectangle(double w,double h):width(w),height(h)

{

}

virtual double area() const

{

return width\*height;

}

protected:

double width,height;

};

class Trapezoid:public Shape

{

public:

Trapezoid(double t,double b,double h):top(t),bottom(b),height(h)

{

}

virtual double area() const

{

return 0.5\*(top+bottom)\*height;

}

protected:

double top,bottom,height;

};

class Triangle:public Shape

{

public:

Triangle(double w,double h):width(w),height(h)

{

}

virtual double area() const

{

return 0.5\*width\*height;

}

protected:

double width,height;

};

int main()

{

Circle circle(12.6);

Square square(3.5);

Rectangle rectangle(4.5,8.4);

Trapezoid trapezoid(2.0,4.5,3.2);

Triangle triangle(4.5,8.4);

Shape \*pt[5]=

{

&circle,&square,&rectangle,&trapezoid,&triangle

};

double areas=0.0;

for(int i=0;i<5;i++)

{

areas=areas+pt[i]->area();

}

cout<<"total of all areas="<<areas<<endl;

return 0;

}

 ③测试用例：

半径：8.0

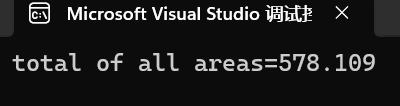
边长：10.0

长：10.0 宽：4.6

边：10.0 4.0 8.0

底：8.0 高；5.7

④运行结果及分析：



输出每个图形的面积，并计算它们的总面积。

⑤请回答：这道题用基类指针数组，可以给编程带来什么好处？如果不用基类指针数组，还可以用什么方法求出它们的面积和？

使用基类指针数组的好处是可以通过一个指针数组来管理不同类型的派生类对象，从而实现动态绑定和多态性。在这个程序中，通过基类指针数组存储了不同类型的图形对象，然后通过虚函数调用它们的面积计算方法。

如果不使用基类指针数组，还可以使用模板（Template）来实现