1.2.

Code:

#include <iostream>

using namespace std ;

class Point

{

private:

double x, y ;

public:

Point(int x = 0, int y = 0): x(x), y(y)

{

cout << "Point constructed" << '(' << x << ',' << y << ')' << endl ;

}

Point(Point &p): x(p.x), y(p.y)

{

cout << "Point copy constructed" << '(' << x << ',' << y << ')' << endl ;

}

virtual ~Point()

{

cout << "Point destructed" << '(' << x << ',' << y << ')' << endl ;

}

virtual void Display() const;

virtual double GetArea() const ;

} ;

void Point::Display() const

{

cout << '(' << x << ',' << y << ')' << endl ;

}

double Point::GetArea() const

{

return 0.0 ;

}

class Circle: public Point

{

private:

double r ;

public:

Circle(Point &p, double r): Point(p), r(r)

{

cout << "Circle constructed," ;

Display() ;

}

Circle(Circle &c): Point(c), r(c.r)

{

cout << "Circle copy constructed," ;

Display() ;

}

virtual ~Circle()

{

cout << "Circle destructed," ;

Display() ;

}

virtual void Display() const ;

virtual double GetArea() const ;

} ;

void Circle::Display() const

{

cout << "r=" << r << ", at" ;

Point::Display() ;

}

double Circle::GetArea() const

{

return (3.14159 \* r \* r) ;

}

class Rectangle: public Point

{

private:

double h, w ;

public:

Rectangle(Point &p, double h, double w): Point(p), h(h), w(w)

{

cout << "Rectangle constructed," ;

Display() ;

}

Rectangle(Rectangle &r): Point(r), h(r.h), w(r.w)

{

cout << "Rectangle constructed," ;

Display() ;

}

virtual ~Rectangle()

{

cout << "Rectangle destructed," ;

Display() ;

}

virtual void Display() const ;

virtual double GetArea() const ;

} ;

void Rectangle::Display() const

{

cout << "Height=" << h << ",Width=" << w << ", at" ;

Point::Display() ;

}

double Rectangle::GetArea() const

{

return h \* w ;

}

class Square: public Rectangle

{

public:

Square(Point &p, double a): Rectangle(p, a, a)

{

cout << "Square constructed," ;

Display() ;

}

Square(Square &s): Rectangle(s)

{

cout << "Square copy constructed," ;

Display() ;

}

virtual ~Square()

{

cout << "Square destructed," ;

Display() ;

}

virtual void Display() const ;

virtual double GetArea() const ;

} ;

void Square::Display() const

{

Rectangle::Display() ;

}

double Square::GetArea() const

{

return Rectangle::GetArea() ;

}

void f(Point &a)

{

cout << "The area is " << a.GetArea() << endl ;

}

int main()

{

Point p1(1,1), p2(2,2) ;

Rectangle r1(p1, 2, 3), r2(p2, 4, 1) ;

Square s1(p1, 5) ;

f(p1) ;

f(r1) ;

f(s1) ;

Point \*p[] = {new Point(p2), new Point(4,3), new Rectangle(r2), new Square(p1, 1)} ;

for (int i = 0; i < 4; i++)

{

f(\*p[i]) ;

}

for (int i = 3; i >= 0; i--)

{

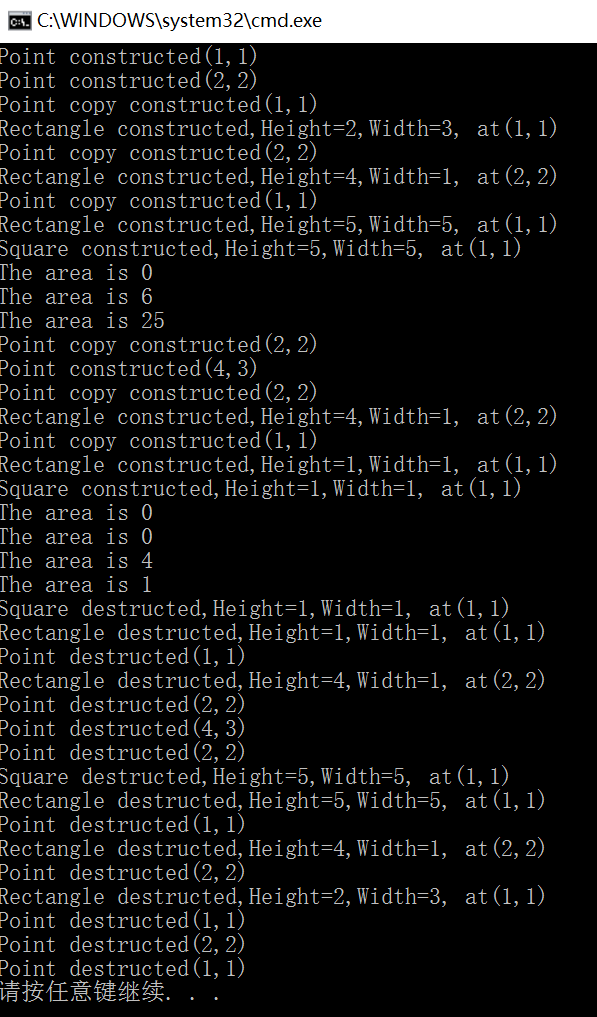
delete p[i] ;

}

return 0 ;

}

Result:



3.

Code:

#include <iostream>

using namespace std ;

class Point

{

private:

double x, y ;

public:

Point(double x = 0, double y = 0): x(x), y(y)

{

cout << "Point constructed" << '(' << x << ',' << y << ')' << endl ;

}

Point(Point &p): x(p.x), y(p.y)

{

cout << "Point copy constructed" << '(' << x << ',' << y << ')' << endl ;

}

virtual ~Point()

{

cout << "Point destructed" << '(' << x << ',' << y << ')' << endl ;

}

virtual void Display() const;

virtual double GetArea() const ;

} ;

void Point::Display() const

{

cout << '(' << x << ',' << y << ')' << endl ;

}

double Point::GetArea() const

{

return 0.0 ;

}

class Circle: virtual public Point

{

private:

double r ;

public:

Circle(Point &p, double r): Point(p), r(r)

{

cout << "Circle constructed," ;

Display() ;

}

Circle(double x, double y, double r): Point(x, y), r(r)

{

cout << "Circle constructed," ;

Display() ;

}

Circle(Circle &c): Point(c), r(c.r)

{

cout << "Circle copy constructed," ;

Display() ;

}

virtual ~Circle()

{

cout << "Circle destructed," ;

Display() ;

}

virtual void Display() const ;

virtual double GetArea() const ;

} ;

void Circle::Display() const

{

cout << "r=" << r << ", at" ;

Point::Display() ;

}

double Circle::GetArea() const

{

return (3.14159 \* r \* r) ;

}

class Square: virtual public Point

{

private:

double L ;

public:

Square(Point &p, double l): Point(p), L(l)

{

cout << "Square constructed," ;

Display() ;

}

Square(double x, double y, double l): Point(x, y), L(l)

{

cout << "Square constructed," ;

Display() ;

}

Square(Square &r): Point(r), L(r.L)

{

cout << "Square constructed," ;

Display() ;

}

virtual ~Square()

{

cout << "Square destructed," ;

Display() ;

}

virtual void Display() const ;

virtual double GetArea() const ;

} ;

void Square::Display() const

{

cout << "Length=" << L << ", at" ;

Point::Display() ;

}

double Square::GetArea() const

{

return L \* L ;

}

class Margin: public Circle, public Square

{

public:

Margin(double x = 0, double y = 0, double r = 0, double l = 0): Point(x, y), Circle(1, 2, r), Square(1, 2, l)

{

cout << "Margin constructed," ;

Display() ;

}

Margin(Margin &m): Point(m), Circle(m), Square(m)

{

cout << "Margin copy constructed," ;

Display() ;

}

virtual ~Margin()

{

cout << "Margin destructed," ;

Display() ;

}

void Display() const;

double GetArea() const;

} ;

void Margin::Display() const

{

Circle::Display() ;

Square::Display() ;

}

double Margin:: GetArea() const

{

return (Square::GetArea() - Circle::GetArea()) ;

}

int main()

{

Margin m1(1, 1, 3, 4), m2(2, 1, 5, 4) ;

m1.Display() ;

m2.Display() ;

cout << "The area of m1 is " << m1.GetArea() << endl ;

cout << "The area of m2 is " << m2.GetArea() << endl ;

return 0 ;

}

Result:

Point constructed(1,1)

Circle constructed,r=3, at(1,1)

Square constructed,Length=4, at(1,1)

Margin constructed,r=3, at(1,1)

Length=4, at(1,1)

Point constructed(2,1)

Circle constructed,r=5, at(2,1)

Square constructed,Length=4, at(2,1)

Margin constructed,r=5, at(2,1)

Length=4, at(2,1)

r=3, at(1,1)

Length=4, at(1,1)

r=5, at(2,1)

Length=4, at(2,1)

The area of m1 is -12.2743

The area of m2 is -62.5397

Margin destructed,r=5, at(2,1)

Length=4, at(2,1)

Square destructed,Length=4, at(2,1)

Circle destructed,r=5, at(2,1)

Point destructed(2,1)

Margin destructed,r=3, at(1,1)

Length=4, at(1,1)

Square destructed,Length=4, at(1,1)

Circle destructed,r=3, at(1,1)

Point destructed(1,1)

请按任意键继续. . .