**1.**

#include <iostream>

using namespace std;

class Point

{

public:

Point(double X0, double Y0) : X(X0), Y(Y0) { cout << "Point constructor!" << endl; }

Point(const Point& srcPt) : X(srcPt.X), Y(srcPt.Y) { cout << "Point copy constructor!" << endl; }

void Set(double new\_X, double new\_Y) { X = new\_X; Y = new\_Y; }

void Print() const { cout << "(" << X << ", " << Y << ")"; }

~Point() { cout << "Point destructor!" << endl; }

private:

double X;

double Y;

};

class Line : public Point

{

public:

Line(double X, double Y, double \_S) : Point(X, Y), S(\_S) { cout << "Line constructor!" << endl; }

Line(const Line& srcLn) : Point(srcLn), S(srcLn.S) { cout << "Line copy constructor!" << endl; }

void Set(double new\_X, double new\_Y, double new\_S) { Point::Set(new\_X, new\_Y); S = new\_S; }

void Print() const { Point::Print(); cout << " Slope: " << S << endl; }

~Line() { cout << "Line destructor!" << endl; }

private:

double S;

};

int main()

{

Line demo1(1, 2, 3);

demo1.Print();

demo1.Set(0, 0, 9);

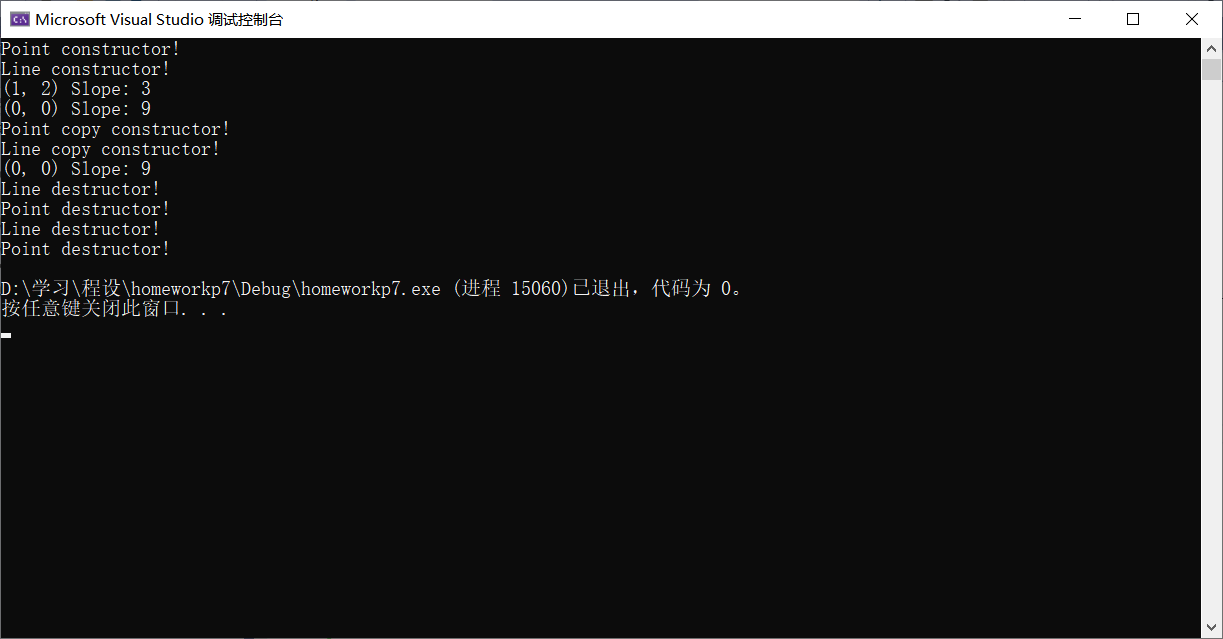
demo1.Print();

Line demo2(demo1);

demo2.Print();

return 0;

}



**2.**

#include <iostream>

using namespace std;

class Point

{

public:

Point(double X0, double Y0) : X(X0), Y(Y0) { cout << "Point constructor!" << endl; }

Point(const Point& srcPt) : X(srcPt.X), Y(srcPt.Y) { cout << "Point copy constructor!" << endl; }

void Set(double new\_X, double new\_Y) { X = new\_X; Y = new\_Y; }

void Print() const { cout << "(" << X << ", " << Y << ")"; }

~Point() { cout << "Point destructor!" << endl; }

protected:

double X;

double Y;

};

class Line : public Point

{

public:

Line(double X, double Y, double \_S) : Point(X, Y), S(\_S) { cout << "Line constructor!" << endl; }

Line(const Line& srcLn) : Point(srcLn), S(srcLn.S) { cout << "Line copy constructor!" << endl; }

void Set(double new\_X, double new\_Y, double new\_S) { Point::X = new\_X; Point::Y = new\_Y; S = new\_S; }

void Print() const { cout << "(" << Point::X << ", " << Point::Y << ") Slope: " << S << endl; }

~Line() { cout << "Line destructor!" << endl; }

private:

double S;

};

int main()

{

Line demo1(1, 2, 3);

demo1.Print();

demo1.Set(0, 0, 9);

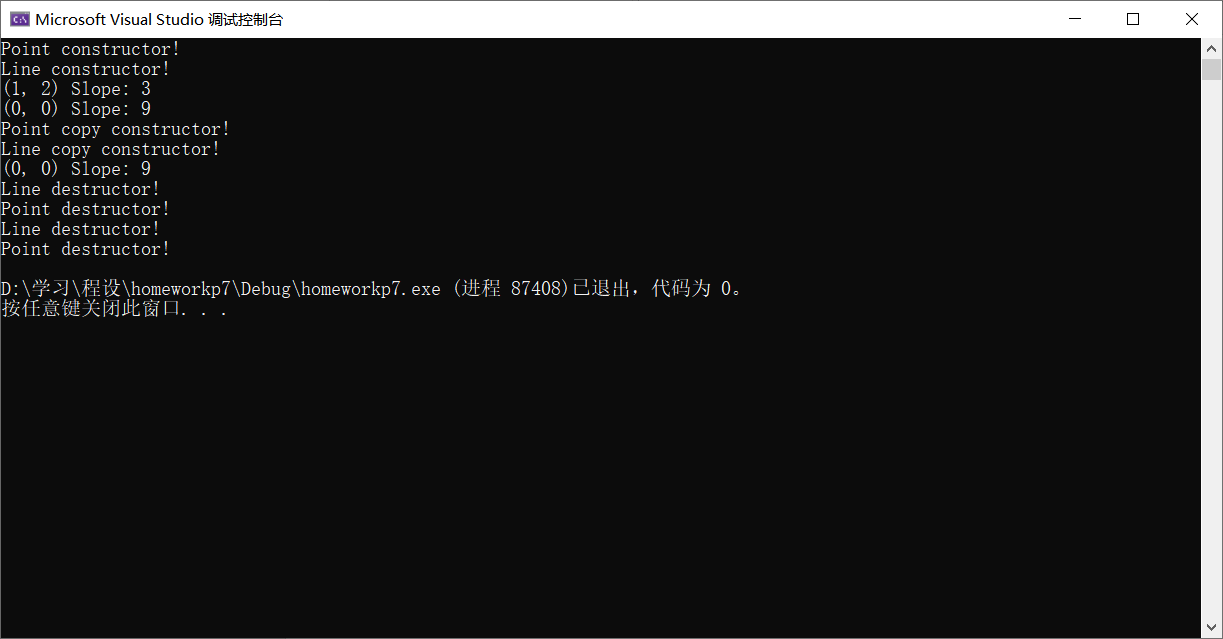
demo1.Print();

Line demo2(demo1);

demo2.Print();

return 0;

}



**\*3.**

#include <iostream>

using namespace std;

class Point

{

public:

Point(double X0, double Y0) : X(X0), Y(Y0) { cout << "Point constructor!" << endl; }

Point(const Point& srcPt) : X(srcPt.X), Y(srcPt.Y) { cout << "Point copy constructor!" << endl; }

void Set(double new\_X, double new\_Y) { X = new\_X; Y = new\_Y; }

void Print() const { cout << "(" << X << ", " << Y << ")"; }

double GetX() const { return X; }

double GetY() const { return Y; }

~Point() { cout << "Point destructor!" << endl; }

private:

double X;

double Y;

};

class Triangle : public Point

{

public:

Triangle(double X0, double Y0, double X1, double Y1, double X2, double Y2) : Point(X0, Y0), p1(X1, Y1), p2(X2, Y2) { cout << "Triangle constructor!" << endl; }

Triangle(const Triangle& srcTri) : Point(srcTri), p1(srcTri.p1), p2(srcTri.p2) { cout << "Triangle copy constructor!" << endl; }

void Set(double newX0, double newY0, double newX1, double newY1, double newX2, double newY2)

{

Point::Set(newX0, newY0); p1.Set(newX1, newY1); p2.Set(newX2, newY2);

}

void Print() const

{

cout << "Triangle: ";

Point::Print(); p1.Print(); p2.Print();

cout << endl;

}

double Area();

~Triangle() { cout << "Triangle destructor!" << endl; }

private:

Point p1;

Point p2;

};

double Triangle::Area()

{

double x1 = p1.GetX() - Point::GetX();

double y1 = p1.GetY() - Point::GetY();

double x2 = p2.GetX() - Point::GetX();

double y2 = p2.GetY() - Point::GetY();

double res = x1 \* y2 - x2 \* y1;

return 0.5 \* (res < 0 ? -res : res);

}

int main()

{

Triangle tri1(1, 1, 2, 3, 10, 1);

tri1.Print();

cout << tri1.Area() << endl;

tri1.Set(0, 0, 1, 1, -1, 2);

tri1.Print();

Triangle tri2(tri1);

tri2.Print();

cout << tri2.Area() << endl;

return 0;

}

