重载为类的成员函数 P110

#include <iostream.h>

class complex

{public:

complex(double r=0,double I=0);

complex operator+(const complex& c);

complex operator-(const complex& c);

complex operator-();

void print()const;

private:

double real,imag;

};

complex::complex(double r,double I)

{ real=r;imag=I;}

complex complex::operator +(const complex& c)

{ double r=real+c.real;

double I=imag+c.imag;

return complex(r,I);

}

complex complex::operator -(const complex& c)

{ double r=real-c.real;

double I=imag-c.imag;

return complex(r,I);

}

complex complex::operator -()

{ return complex(-real,-imag);

}

void complex::print()const

{ cout<<"("<<real<<","<<imag<<")"<<endl;

}

int main()

{ complex c1(2.5,3.7),c2(4.2,6.5);

complex c;

c=c1-c2;

c.print();

c=c1+c2;

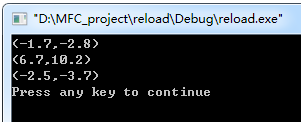
c.print();

c=-c1;

c.print();

return 0;

}



#include <iostream.h>

//using namespace std;

class complex

{public:

complex(double r=0,double I=0);

friend complex operator+(const complex& c1,const complex& c2);

friend complex operator-(const complex& c1,const complex& c2);

friend complex operator-(const complex& c);

void print()const;

private:

double real,imag;

};

complex::complex(double r,double I)

{ real=r;imag=I;}

complex operator+(const complex& c1,const complex& c2)

{ double r=c1.real+c2.real;

double I=c1.imag+c2.imag;

return complex(r,I);

}

complex operator-(const complex& c1,const complex& c2)

{ double r=c1.real-c2.real;

double I=c1.imag-c2.imag;

return complex(r,I);

}

complex operator-(const complex& c)

{ return complex(-c.real,-c.imag);

}

void complex::print()const

{ cout<<"("<<real<<","<<imag<<")"<<endl;

}

int main()

{ complex c1(2.5,3.7),c2(4.2,6.5);

complex c;

c=c1-c2;

c.print();

c=c1+c2;

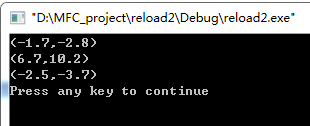
c.print();

c=-c1;

c.print();

return 0;

}



8.1公有继承 P124

#include<iostream.h>

class Location

{public:

void InitL(int xx,int yy);

void Move(int xOff,int yOff);

int GetX(){return X;}

int GetY(){return Y;}

private:

int X,Y;

};

void Location::InitL(int xx,int yy)

{ X=xx;Y=yy;}

void Location::Move(int xOff,int yOff)

{ X+=xOff;Y+=yOff;}

class Rectangle:public Location

{public:

void InitR(int x,int y,int w,int h);

int GetW(){return W;}

int GetH(){return H;}

private:

int W,H;

};

void Rectangle::InitR(int x,int y,int w,int h)

{ InitL(x,y);

W=w;

H=h;

}

int main()

{ Rectangle rect;

rect.InitR(2,3,20,10);

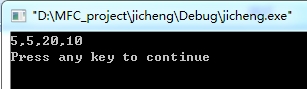
rect.Move(3,2);

cout<<rect.GetX()<<","<<rect.GetY()<<","

<<rect.GetW()<<","<<rect.GetH()<<endl;

return 0;

}



私有继承

#include<iostream.h>

class Location

{public:

void InitL(int xx,int yy);

void Move(int xOff,int yOff);

int GetX(){return X;}

int GetY(){return Y;}

private:

int X,Y;

};

void Location::InitL(int xx,int yy)

{ X=xx;Y=yy;}

void Location::Move(int xOff,int yOff)

{ X+=xOff;Y+=yOff;}

class Rectangle:private Location

{public:

void InitR(int x,int y,int w,int h);

void Move(int xOff,int yOff);

int GetX(){return Location::GetX();}

int GetY(){return Location::GetY();}

int GetW(){return W;}

int GetH(){return H;}

private:

int W,H;

};

void Rectangle::InitR(int x,int y,int w,int h)

{ InitL(x,y);

W=w;

H=h;

}

void Rectangle::Move(int xOff,int yOff)

{

Location::Move(xOff,yOff);

}

class V:public Rectangle

{public:

void Function();

};

void V::Function()

{ Move(3,2); }

int main()

{ Rectangle rect;

rect.InitR(2,3,20,10);

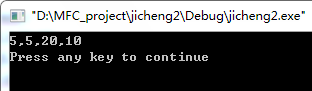
rect.Move(3,2);

cout<<rect.GetX()<<","<<rect.GetY()<<","

<<rect.GetW()<<","<<rect.GetH()<<endl;

return 0;

}



P144-145 动态绑定与虚函数

class Shape{

public:

Shape(double x,double y);

double Area() const;

private:

double X,Y;};

Shape::Shape(double x,double y):X(x),Y(y){}

double Shape::Area() const

{return 0.0;}

class Rectangle:public Shape{

public:

Rectangle(double x,double y,double w,double h);

double Area()const;

private:

double W,H;};

Rectangle::Rectangle(double x,double y,double w,double h):Shape(x,y),W(w),H(h)

{}

double Rectangle::Area() const

{

return W\*H;

}

#include<iostream>

using namespace std;

void fun(const Shape &s)

{ cout<<s.Area()<<endl;}

int main()

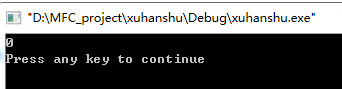
{

Rectangle rect(2.0,5.0,10.0,20.0);

fun(rect);

return 0;

}



P146

class Shape{

public:

Shape(double x,double y);

virtual double Area() const;

private:

double X,Y;};

Shape::Shape(double x,double y):X(x),Y(y){}

double Shape::Area() const

{return 0.0;}

class Rectangle:public Shape{

public:

Rectangle(double x,double y,double w,double h);

virtual double Area()const;

private:

double W,H;};

Rectangle::Rectangle(double x,double y,double w,double h):Shape(x,y),W(w),H(h)

{}

double Rectangle::Area() const

{

return W\*H;

}

#include<iostream>

using namespace std;

void fun(const Shape &s)

{ cout<<s.Area()<<endl;}

int main()

{

Rectangle rect(2.0,5.0,10.0,20.0);

fun(rect);

return 0;

}

