**点圆线：**

#include <iostream>

#include <string>

using namespace std;

class CPoint{

private:

    int X, Y;

public:

    CPoint(int x=0, int y=0){X=x; Y=y;}

    CPoint(CPoint &p){ X=p.X; Y=p.Y; }

    int GetX(){ return X; }

    int GetY() {return Y;}

};

class CShape{

private:

    string Color;

public:

    CShape(string c){Color = c;}

    virtual void Draw(){

     cout << "Draw a shape. The color is " << Color << endl;

    }

    void PrintColor(){ cout << Color << endl;}

};

class CLine:public CShape{

private:

    CPoint Start; //线段的起点

CPoint End; //线段的终点

public:

    CLine(CPoint s, CPoint e, string c):CShape(c),Start(s),End(e){}

    virtual void Draw(){

     cout << "Draw a Line from (" << Start.GetX() << "," << Start.GetY();

     cout<<") to ("<<End.GetX()<<","<<End.GetY() << "), with color ";

PrintColor();

    }

};

class CCircle:public CShape{

private:

CPoint Center;

int Radius;

public:

CCircle(CPoint ctr, int r, string c):CShape(c),Center(ctr){

     Radius = r;

}

virtual void Draw(){

     cout << "Draw a circle at center (" << Center.GetX() << "," ;

     cout << Center.GetY()<< ") with radius " << Radius << " and color ";

PrintColor();

}

};

int main(){

CShape \*ps[3];

CShape s("Red");

CPoint p1(10,10), p2(100,100),p3(50,50);

CLine l(p1,p2,"Green");

CCircle c(p3, 20, "Black");

ps[0] = &s; ps[1] = &l; ps[2] = &c;

for(int i=0; i<3; i++) ps[i]->Draw();

return 0;

}

**复数类：**

#include <iostream>

using namespace std;

class complex

{

public:

complex (double r = 0, double i = 0): re (r), im (i) { }

complex operator + (const complex &x){

return complex(re + x.re, im + x.im);

}

complex operator - (const complex &x){

return complex(re - x.re, im - x.im);

}

double real () const {

return re;

}

double imag () const {

return im;

}

private:

double re, im;

};

ostream&

operator << (ostream& os, const complex& x)

{

return os << '(' << x.real() << ',' << x.imag() << ')';

}

int main()

{

complex c1(2, 1);

complex c2(4, 0);

cout << c1 << endl;

cout << c2 << endl;

cout << c1+c2 << endl;

cout << c1-c2 << endl;

return 0;

}

**日期：**

class Date{

private:

double year,month,day;

public:

Date(double y=0,double m=0,double d=0):year(y),month(m),day(d){};

Date operator+(Date b);

Date operator-(Date b);

friend ostream &operator<<(ostream &os,Date &s);

friend istream &operator>>(istream &is,Date &s);

void Show();

};

Date Date::operator+(Date b){

if((day+b.day)>31){

month++;

int dd=day+b.day-31;

return Date(year,month,dd);

}

else

return Date(year,month,day+b.day);

}

Date Date::operator-(Date b){

if(day-b.day<=0){

month--;

int dd=day-b.day+30;

return Date(year,month,dd);

}

else

return Date(year,month,day-b.day);

}

void Date::Show(){

cout<<year<<"-"<<month<<"-"<<day<<endl;

}

ostream& operator<<(ostream &os,Date &s){

os<<s.year<<"\t";

os<<s.month<<"\t";

os<<s.day<<"\t"<<endl;

return os;

}

istream &operator>>(istream &is,Date &s){

cout<<"按顺序输入年 月 日"<<endl;

is>>s.year;

is>>s.month;

is>>s.day;

cout<<endl;

return is;

}

void main(){

Date t1(2013,10,11),t2(0,0,25),t3,t4;

t1.Show();

t3=t1+t2;

t4=t1-t2;

t3.Show();

t4.Show();

cin>>t1;

cout<<t1;

}

**点类：**

#include <iostream>

#include <cmath>

using namespace std;

class Position

{

private:

int x, y;

public:

Position(int xx = 0, int yy = 0)

: x(xx), y(yy) {}

void set\_x(int xx) {

x = xx;

}

void set\_y(int yy) {

y = yy;

}

int get\_x() {

return x;

}

int get\_y() {

return y;

}

double distance(Position p) {

double dis = 0;

dis = sqrt((double)((p.x - x) \* (p.x - x) + (p.y - y) \* (p.y - y)));

return dis;

}

};

int main() {

Position p1(1, 1);

Position p2(2, 2);

cout << "x is " << p1.get\_x() << " y is " << p1.get\_y() << endl;

p1.set\_x(3);

p1.set\_y(3);

cout << "x is " << p1.get\_x() << " y is " << p1.get\_y() << endl;

cout << "distance is:" << p1.distance(p2) << endl;

return 0;

}

**点类(运算符重载)：**

#include <iostream>

using namespace std;

class Point

{

private:

int x, y;

public:

Point(int xx = 0, int yy = 0)

: x(xx), y(yy) {}

Point & operator ++(int) {

this->x++;

this->y++;

return \*this;

}

Point & operator--(int) {

this->x--;

this->y--;

return \*this;

}

void print\_xy() {

cout << "now point is( " << x << ", " << y << ")" << endl;

}

};

int main(int argc, char const \*argv[]) {

Point p1(1, 1);

p1.print\_xy();

p1--;

p1.print\_xy();

p1++;

p1++;

p1.print\_xy();

system("pause");

return 0;

}