实践一：VC 集成环境：建 Cpp1 动态分配内存：

```
/*动态分配输出*/
#include <iostream>
using namespace std;
void main()
{
	double *p;
	p=new double[3];
	for(int i=0;i<3;i++)
		cin>>*(p+i);
	for(i=0;i<3;i++)
		cout<<*(p+i)<<" ";
	delete p;
}
```

实践二：使用函数模板：

```
/*用函数模板实现 3 个数值中按最小值到最大值排序*/
#include <iostream>
using namespace std;
template <class T>
void max(T m1,T m2,T m3){
T temp;
temp=m1;
if (m1>m2){
temp=m1;m1=m2;m2=temp;
}
if (m1>m3){
temp=m1;m1=m3;m3=temp;
}
if (m2>m3){
temp=m2;m2=m3;m3=temp;
}
cout<<m1<<","<<m2<<","<<m3;
}
void main(){
max("abf","abz","aba");
cout<<endl;
max(18,9,24);
cout<<endl;
max(2.01,2.38,2.33);
cout<<endl;
}
```

实践三：使用多文件编程：

```
/*Point 使用（包含设计的）多文件编程示例*/
```

头文件：CPP10.H

```
#if ! defined(CPP10_H)
#define CPP10_H
#include<iostream>
#include<cmath>
using namespace std;
class Point
{
	double X,Y;
public:
	Point(double=0,double=0);
	Point(Point&);
	void Display()
	{cout<<X<<","<<Y<<endl;}
	double Distance(Point&);
	double Getx()
	{return X;}
	double Gety()
	{return Y;}
};

struct Cow
{
	int Color;
	int Width;
};

class Line
{
	Point a,b;
	Cow cw;
public:
	Line(Point&,Point&,Cow&);
	void Display();
	Line(Line&);
	double Distance();
	double Area();
};
#endif
```

实现类：CPP10.CPP

```
#include "cpp10.h"
Point::Point(double a,double b):X(a),Y(b)
{}
Point::Point(Point&a)
```

```
{X=a.X;Y=a.Y;}
double Point::Distance(Point&a)
{
    return sqrt((X-a.X)*(X-a.X)+(Y-a.Y)*(Y-a.Y));
}
Line::Line(Point&a1,Point&a2,Cow&a3):a(a1),b(a2),cw(a3)
{}
Line::Line(Line&s):a(s.a),b(s.b),cw(s.cw)
{}
void Line::Display()
{
    a.Display();
    b.Display();
    cout<<"Color="<<cw.Color<<","<<"Width="<<cw.Width<<endl;
}
double Line::Distance()
{
    double x=a.Getx()-b.Getx();
    double y=a.Gety()-b.Gety();
    return sqrt(x*x+y*y);
}
double Line::Area()
{return cw.Width * Distance();}
```

主函数 FIND10.CPP

```
#include"cpp10.h"
void main()
{
    Point a;
    Point b(7.8,9.8),c(34.5,67.8);
    a=c;
    a.Display();
    b.Display();
    cout<<"两点之距："<<a.Distance(b)<<endl;
    Cow cw = {3,5};
    Line s(a,b,cw);
    Line s1(s);
    cout<<"线段属性如下："<<endl;
    s1.Display();
    cout<<"线段长度："<<s1.Distance()<<endl;
    cout<<"线段面积："<<s1.Area()<<endl;
}
```

实践四：公有继承的赋值兼容性规则：

/*编写一个点类 Point，再由它派生线段类 Line，公有继承的赋值兼容规则*/

```
#include<iostream>
#include<cmath>

using namespace std;

class Point
{
private:
float x,y;
public:
Point(){}
Point(float a,float b):x(a),y(b){}
Point(Point& a);
float Distance(Point b);
};

Point::Point(Point& a)
{
x=a.x;
y=a.y;
}

float Point::Distance(Point b)
{
return sqrt((x-b.x)*(x-b.x)+(y-b.y)*(y-b.y));
}

class Line:public Point
{
private:
Point a,b;
public:
Line(float x1,float y1,float x2,float y2):a(x1,y1),b(x2,y2){}
float Display();
};

float Line::Display()//返回线段的距离
{
return a.Distance(b);
}
int main()
{
Point a;
Point b(7.8,9.8),c(34.5,67.8);
```

```
a=c;
cout<<"两点之间距离是："<<a.Distance(b)<<endl;
Line s(7.8,9.8,34.5,67.8);
cout<<s.Display()<<endl;
return 0;
}
```

实践五：使用向量：

```
/*向量出圈游戏*/
#include <iostream>
#include <vector>
using namespace std;
class SeqList
{
    char name[10];
public:
    void DispName()
    {
        cout<<name;
    }
    void SetName(char b[ ])
    {
        strcpy(name,b);
    }
    void Joseph(vector<SeqList>&);
};
void SeqList::Joseph(vector<SeqList>&c)
{
    int m,star,i,j,k;
    cout<<"请输入间隔数 m(m<=20)";
    cin>>m;
    while(m>20)
    {
        cout<<"间隔太大，请重新输入：";
        cin>>m;
    }
    cout<<"从第几个人的位置开始报数（不能大于"<<c.size()<<"):";
    cin>>star;
    while(star>c.size())
    {
        cout<<"开始位置大于人数，重新输入：";
        cin>>star;
    }
    cout<<"准备输入名字"<<endl;
    getchar();
```

```
        char s[10];
        for(i=0;i<c.size();i++)
        {
            cout<<"第"<<i+1<<"个人的名字：";
            gets(s);
            c[i].SetName(s);
        }
        i=star-2;
        vector<SeqList>::iterator P;
        P=c.begin();
        int length=c.size();
        for (k=1;k<=length;k++)
        {
            j=0;
            while(j<m){
                i++;
                if(i==c.size())
                    i=0;
                j++;
            }
            if (k==length) break;
            c[i].DispName();
            cout<<",";
            c.erase(P+i);
            --i;
        }
        c[i].DispName();
        cout<<endl;
}
void main()
{
        int length=0;
        cout<<"请输入人数：";
        cin>>length;
        vector<SeqList>c(length);
        SeqList game;
        game.Joseph(c);
}
```

实践六：运算符重载：

```
/*重载- -运算符*/
#include <iostream>
using namespace std;
class number
{
```

```
    int num;
public:
    number(int i)
    {num=i;}
    int operator--();
    int operator--(int);
    void print()
    {cout<<"num="<<num<<endl;}
};
int number::operator --()
{
    num--;
    return num;
}
int number::operator --(int)
{
    int i=num;
    num --;
    return i;
}
void main()
{
    number n(10);
    int i=--n;
    cout<<"i="<<i<<endl;
    n.print();
    i=n--;
    cout<<"i="<<i<<endl;
    n.print();
}
```

实践七：文件读写：

```
/*文件存取 student*/
头文件
#if !defined(STUDENT_H)
#define STUDENT_H
#include <iostream>
#include <string>
#include <iomanip>
#include <vector>
#include <fstream>
using namespace std;
class student
{
    string number;
```

```
    double score;
public:
    void SetNum(string s)
    {number=s;}
    void Setscore(double s)
    {score=s;}
    string GetNum()
    {return number;}
    double GetScore()
    {return score;}
    void set(vector<student>&);
    void display(vector<student>&);
    void read();
};
#endif
```

源文件：

```
#include "student.h"
//display  输出向量信息
void student::display(vector<student>&c)
{
    cout<<"学号"<<setw(20)<<"成绩"<<endl;
    for(int i=0;i<c.size();i++)
        cout<<c[i].GetNum()<<setw(12)<<c[i].GetScore()<<endl;
}
//set  为向量赋值并将向量内容存入文件
void student::set(vector<student>&c)
{
    student a;
    string s;
    double b;
    while(1)
    {cout<<"学号：";
    cin>>s;
    if(s=="0")
    {
        ofstream wst("stud.txt");
        if(! wst)
        {
            cout<<"没有正确建立文件！"<<endl;
            return;
        }
        for (int i=0;i<c.size();i++)
            wst<<c[i].number<<" "<<c[i].score<<" ";
        wst.close();
```

```
            cout<<"一共写入"<<c.size()<<"个学生的信息。\n";
            return;
        }
        a.SetNum(s);
        cout<<"成绩： ";
        cin>>b;
        a.Setscore(b);
        c.push_back(a);
        }
}
//read  显示文件内容
void student::read()
{
        string number;
        double score;
        ifstream rst("stud.txt");
        if(! rst)
        {cout<<"文件打不开\n"<<endl;
        return;
        }
        cout<<"学号"<<setw(20)<<"成绩"<<endl;
        while(1)
        {rst>>number>>score;
        if(rst.eof())
        {rst.close();
        return;
        }
        cout<<number<<setw(12)<<score<<endl;
        }
}

void main()
{
        vector<student>st;
        student stud;
        stud.set(st);
        cout<<"显示向量数组信息如下：\n";
        stud.display(st);
        cout<<"存入文件内的信息如下:"<<endl;
        stud.read();
}
```

实践八：虚函数的多态性：

/*Point 使用（继承设计的）多文件编程虚函数多态性示例*/

头文件：CPP101.H

```
#if ! defined(CPP101_H)
#define CPP101_H
#include <iostream>
#include <math.h>
using namespace std;
class Point
{
protected:
	double X,Y;
public:
	Point(double=0,double=0);
	Point(Point&);
	virtual void Display()
	{cout<<"X="<<X<<",Y="<<Y<<endl;}
	double Distance(Point&);
	virtual double Area()
	{return 0;}
	double Getx()
	{return X;}
	double Gety()
	{return Y;}
};
struct Cow
{
	int Color;
	int Width;
};
class Line:public Point
{
	double X2,Y2;
	Cow cw;
public:
	Line(double,double,double,double,Cow&);
	Line(Line&);
	void Display();
	double Distance();
	double Area();
	double Getx2()
	{return X2;}
	double Gety2()
	{return Y2;}
	double Getc()
	{return cw.Color;}
	double Getw()
```

```
    {return cw.Width;}
    friend void Disp(Line&t)
    {cout<<t;}
    friend ostream&operator<<(ostream&,Line);
};
#endif
```

实现类：CPP101.CPP

```
#include "cpp101.h"
Point::Point(double a,double b):X(a),Y(b)
{}
Point::Point(Point&a)
{X=a.X;Y=a.Y;}
double Point::Distance(Point&a)
{return sqrt((X-a.X)*(X-a.X)+(Y-a.Y)*(Y-a.Y));}
Line::Line(double a1,double a2,double a3,double a4,Cow&c):Point(a1,a2),X2(a3),Y2(a4),cw(c)
{}
Line::Line(Line&s):Point(s),X2(s.X2),Y2(s.Y2),cw(s.cw)
{}
double Line::Distance()
{
    double x=X2-X;
    double y=Y2-Y;
    return sqrt(x*x+y*y);
}
void Line::Display()
{
    cout<<"X="<<X<<",Y="<<Y<<",X2="<<X2<<",Y2="<<Y2<<",Color="<<cw.Color<<",W
idth="<<cw.Width<<endl;
}
double Line::Area()
{return cw.Width * Distance();}
ostream &operator<<(ostream&stream,Line obj)
{
    stream<<"使重载<<输入线段属性如下："<<endl;
    stream<<obj.Getx()<<","<<obj.Gety()<<","<<obj.Getx2()<<","<<obj.Gety2()<<","<<obj.Ge
tc()<<","<<obj.Getw()<<endl;
    return stream;
}
```

主函数：FIND101.CPP

```
#include "cpp101.h"
void main()
{
```

```
    Point a;
    Point b(7.8,9.8),c(34.5,67.8);
    a=c;
    a.Display();
    b.Display();
    cout<<"两点距离:"<<a.Distance(b)<<endl;
    Cow cw={3,5};
    Line s(7.8,9.8,34.5,67.8,cw);
    Disp(s);
    Line s1(s);
    cout<<"使用 Display 函数输出线段属性如下："<<endl;
    s1.Display();
    cout<<"线段长度:"<<s1.Distance()<<endl;
    cout<<"线段面积:"<<s1.Area()<<endl;
    cout<<"基类对象的属性"<<endl;
    a.Display();
    a=s;
    cout<<"派生类的对象赋给基类对象"<<endl;
    a.Display();
    cout<<"面积:"<<a.Area()<<endl;
    cout<<"派生类的对象赋给基类的指针"<<endl;
    Point *p=&s1;
    p->Display();
    cout<<"面积"<<p->Area()<<endl;
    cout<<"基类对象引用派生类的对象"<<endl;
    Point &d=s1;
    d.Display();
    cout<<"面积:"<<d.Area()<<endl;
}
```

其它

```
/*求两个数据中的最大值的函数模板程序*/
#include <iostream>
using namespace std;
template <class T>
T max(T m1,T m2)
{
    return(m1>m2)?m1:m2;
}
void main()
{
    cout<<max(2,5)<<"\t"<<max(2.0,5.)<<"\t"<<max('w','a')<<"\t"<<max("ABC","ABD")<<end
l;
}
```

```
/*将结构对象的两个域值相加，乘以 2 再加 50*/
#include <iostream>
using namespace std;
int result(int,int);
const int k=2;
struct point
{
    int x,y;
};
int main()
{
    int z(0),b(50);
    point a;
    cout<<"输入两个整数（以空格区分）：";
    cin>>a.x>>a.y;
    z=(a.x+a.y)*k;
    z=result(z,b);
    cout<<"计算结果如下："<<endl;
    cout<<"(("<<a.x<<"+"<<a.y<<")*"<<k<<")+"<<b<<"="<<z<<endl;
    return 0;
}
int result(int a,int b)
{
    return a+b;
}

/*求 ax2+bx+c=0 的根的程序*/
头文件
#if ! defined(EQUATION_H)
#define EQUATION_H
#include <iostream>
#include <cmath>
using namespace std;
class FindRoot
{
private:
    float a,b,c,d;
    double x1,x2;
public:
    FindRoot(float x,float y,float z);
    void Find();
    void Display();
};
#endif
```

实现类

```
#include "equation.h"
FindRoot::FindRoot(float x,float y,float z)
{
	a=x;b=y;c=z;
	d=b*b-4*a*c;
}
void FindRoot::Find()
{
	if(d>0)
	{
		x1=(-b+sqrt(d))/(2*a);
		x2=(-b-sqrt(d))/(2*a);
		return;
	}
	else if(d==0)
	{
		x1=x2=(-b)/(2*a);
		return;
	}
	else
	{
		x1=(-b)/(2*a);
		x2=sqrt(-d)/(2*a);
	}
}
void FindRoot::Display()
{
	if(d>0)
	{
		cout<<"X1="<<x1<<"\nX2="<<x2<<endl;
		return;
	}
	else if (d==0)
	{
		cout<<"X1=X2="<<x1<<endl;
		return;
	}
	else
	{
		cout<<"X1="<<x1<<"+"<<x2<<"i"<<endl;
		cout<<"X2="<<x1<<"-"<<x2<<"i"<<endl;
	}
}
```

主函数

```
#include "equation.h"
void Read(float&,float&,float&);
void main()
{
    float a,b,c;
    cout<<"这是一个求方程 ax2+bx+c=0 的根的程序"<<endl;
    for(;;)
    {
        Read(a,b,c);
        if(a==0) return;
        FindRoot obj(a,b,c);
        obj.Find();
        obj.Display();
    }
}
void Read(float&a,float&b,float&c)
{
    cout<<"输入方程系数 a:";
    cin>>a;
    if(a==0)
    {
        getchar();
        return;
    }
    cout<<"输入方程系数 b:";
    cin>>b;
    cout<<"输入方程系数 c:";
    cin>>c;
}
```

杨鼎 2008 年 11 月 7 日