1)成员函数的实现

void Clock::setTime(int newH, int newM, int newS){

hour = newH;

minute = newM;

second = newS; }

2)组合构造函数的实现

Line::Line(Point xp1, Point xp2):p1(xp1),p2(xp2){

cout<<"Calling constructor of Line"<<endl;

double x = static\_cast<double>(p1.getX()-p2.getX());

double y = static\_cast<double>(p1.getY()-p2.getY());

len = sqrt(x\*x+y\*y);}

3）静态数据成员初始化和调用静态成员函数需要用类名限定

4）类成员中的静态变量和常量都应当在类定义外加以定义，除非静态常量是整数类型（或枚举类型）

5）常成员函数命名方式

类型说明符 函数名（参数表）const

void print() const

6）类型兼容规则（基类及其共有派生类的关系）

假设D是B的共有派生类

B b, \*pb; D d;

【1】子类给父类赋值

b = d;

【2】子类初始化父类引用

B &rb=d;

【3】子类地址转换为指向父类的指针

pb=&d;

7）重载运算符

7.1) 运算符重载为成员函数

7.1.1) 普通加法

...

public:

complex operator+ (const complex &c2) const;

...

complex complex::operator+(const complex &c2) const{

return complex(real+c2.real, imag+c2.imag); }

7.1.2) 重载前置++

complex& operator++() {

value++;

return \*this;

}

7.1.3) 后置++

complex operator++(int){

complex old = \*this;

++(\*this);

return old;

}

7.2) 运算符重载为非成员函数

7.2.1)普通加法

//在类里声明运算符重载函数是friend类型

...

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

return complex(c1.real+c2.real, c1.imag+c2.imag);

}

7.2.2) cout

ostream& operator<<(ostream& out, complex &c1){

out<<"..."<<endl;

return out;

}

8)printf标准化格式输出

%[flags][width][.prec][length]type

%[标志][最小宽度][.精度][类型长度]类型。

bool 型：

printf() 输出 bool 类型无专用类型标识符，实际输出时按照整型 0 或 1 输出布尔值

整数型：

int a=9;

printf("%d用五位输出为：%5d",a,a) // 9

printf("&d用五位输出且前面用0填充：%05d",a,a) //00009

浮点数型：

double a = 3.1415;

printf("%lf 保留1为有效数字为：%.1lf",a,a) //3.1

字符串类型 %s

9）cout

std::oct 八进制输出

std::hex 十六进制输出

setbase(n) n=8,10,16 以n进制输出

cout<<fixed<<setprecision(n)<<...; //强制控制输出位数（n）

cout<<setiosflags(ios::left)<<setw(8); //设置左对齐

