P5

不改变成员变量时用const

double real() const {return re};

任何的成员函数都有一个隐藏的pointer，谁调用，this指针就指向谁。

如：c2 += c1，+=是由c2调用，重载时就指向c2

return by reference

传递者无需知道接收者是以reference接收（只要传递的是一个值）

操作符重载可以返回引用或void，但返回void不能进行连串使用

c3+=c2+=c1

重载+=时应该：

complex& complex::operator +=(...){...}

若返回为void，则不能连串使用

临时对象，临时产生的对象，没有名字，到下一行就消亡，用()表示

如：return int(5);

输出符重载，只能用非成员函数

P7

三大函数：拷贝构造、拷贝赋值、析构函数

//拷贝构造，形参为自己，新建一个对象并初始化为同类现有对象时使用

String(String& str);

//拷贝赋值，形参为自己，将已有的对象赋给另一个对象时使用

String& operator=(String& str);

～String();//析构函数

有指针的类就一定要拷贝构造和拷贝赋值函数。

浅拷贝和深拷贝

拷贝赋值：1.清空自己，2.创建空间，3.拷贝；检测自我赋值很重要

P11

Container

。。。

复合composition

Component

Component

Container

构造由内而外

Container::Container(...):Component(){...};先调用Component的构造函数，再调用Container的构造函数

析构由外而内

P12 重要re

委托delegation

经典方法：Pimpl pointer implement，在指向的地方实现，handle/body

继承时，父类的析构函数必须是virtual

Template Method重要

P13设计模式

composite

prototype？

Part2

P19 const

|  |  |  |
| --- | --- | --- |
|  | const object  (data member不能改变) | non-const object  (data member可能改变) |
| const member function  (保证不改变data member) | 可 | 可 |
| non-const member function  (不保证data member不变) | 不可 | 可 |

const的成员函数：若成员函数不改变成员变量的值，则应该尽量用const，如：

//定义

double real() const {return re; } // 只读，不改变成员变量re的值。

//调用

const complex c;//若定义的real()函数没有加const，就是表格中不可的一项

cout << c.real();

Remark：当成员函数const和non-const版本同时存在时，const object只会调用const版本，non-const object只会调用non-const版本。

COW：Copy On Write，例如：

operator[] (size\_type pos) const

{.../\*不用考虑COW\*/...}

operator[] (size\_type pos)

{.../\*必须考虑COW\*/...}

两者的标签不同，可以重载，非const函数必须考虑COW