Effective C++笔记

2015年10月7日

12:46

**条款1**

**视C++为一个语言联邦**

**1.** 4个部分 C   Object-Oriented C++   Templete C++   STL

**2.** 对于内置类型（C like）pass by value比pass by reference高效

对于OOC++以及Templete C++ 由于有构造 析构函数，pass by reference to const更好

STL因为是在C的指针上塑造 pass by value

（参数传递详见条款20）

**条款2**

**尽量以const enum inline替换#define**

**1.** enum和const可以代替define，而enum更像一些，因为不能取地址，而const可以。同时 可以在声明时定义。

**2.** 若以const修饰class内的常量 同时是static且为整数类型 如int char bool时可以在class的声明时候赋值 而不必再在实现文件的定义式内赋值，甚至定义式也可以不写。（不取变量地址的情况下）

**例：**

class A {

     static const int NUM = 5;

     int scores[NUM];

};

如果一定要定义式，则在cpp文件中 const int A：：NUM; 此时不要给予数值 因为已经有初值了

如果不是整数类型那么就只在class内声明 在cpp实现中给初值。注意不在构造函数内定义。

**3.**上例用enum可改写为

class A {

     enum{ NUM = 5 };

     int scores[NUM];

};

可以解决上例编译器不支持的情况 同时因为enum不能取地址，更像#define，而#define的作用域不会限定在class内，无封装性。

**4.** 形似函数的宏不要define，用inline函数进行替代，如图：

计算机生成了可选文字:
/ / 以 a 和 b 的 较 大 值 调 用 f 
#define CALL WITH ！ X ， b) f((a) > (b) ？ { 田 
(b)) 
这 般 长 相 的 宏 有 着 太 多 缺 点 ， 光 是 想 到 它 们 就 让 人 痛 苦 不 堪 · 
无 论 何 时 当 你 写 出 这 种 宏 ， 你 必 须 记 住 为 宏 中 的 所 有 实 参 加 上 小 括 号 ， 否 则 某 
些 人 在 表 达 式 中 调 用 这 个 宏 时 可 能 会 遭 遇 麻 烦 。 但 纵 使 你 为 所 有 实 参 加 上 小 括 号 ． 
看 看 下 面 不 可 思 议 的 事 情 ， 
int a = 5 ， b 
CALL WITH MAX(++a, b); 
CALL WITH MAX(++a, b + 10 
/ 被 累 加 二 次 
//a 被 累 加 一 次 
在 这 里 · 调 用 f 之 前 · 的 递 增 次 数 竟 然 取 决 于 “ 它 被 拿 来 和 谁 比 较 ” ， 
幸 运 的 是 你 不 需 要 对 这 种 无 聊 事 情 提 供 温 床 。 你 可 以 获 得 宏 带 来 的 效 率 以 及 一 
般 函 数 的 所 有 可 预 料 行 为 和 类 型 安 全 性 （ 呼 “ fe ， ） 
函 数 （ 见 条 款 30 ） ， 
template«typenartE T» 
inline void (const T& a. const TG 
f(a > b ？ ： b); 
只 要 你 写 出 templateinline 
/ / 由 于 我 们 不 知 道 
b) //T 是 什 么 ， 所 以 采 用 
//rnsbyreferare-to-c«l.st. 
/ / 见 条 款 

**条款3**

**尽可能使用const**

**1.**Const char\* p静态数据

Char \* const p 静态指针 注意＊位置 思考？？？

如果被指物常量const 写在＊左边

const Widget ＊p与Widegt const ＊p意思相同

**条款3**

**尽可能使用const**

**1.**Const char\* p静态数据

Char \* const p 静态指针 注意＊位置 思考？？？

如果被指物常量const 写在＊左边

const Widget ＊p与Widegt const ＊p意思相同

const＿iterator模拟const T＊

const vector＜int＞：：iterator iter模拟 T＊const

const返回值可避免对结果附值错误

查看＆的位置。 int \*&p = a是对指针a的引用   &p 可以看作是一种别名

函数返回值为引用目的是为了方便赋值如运算符重载函数

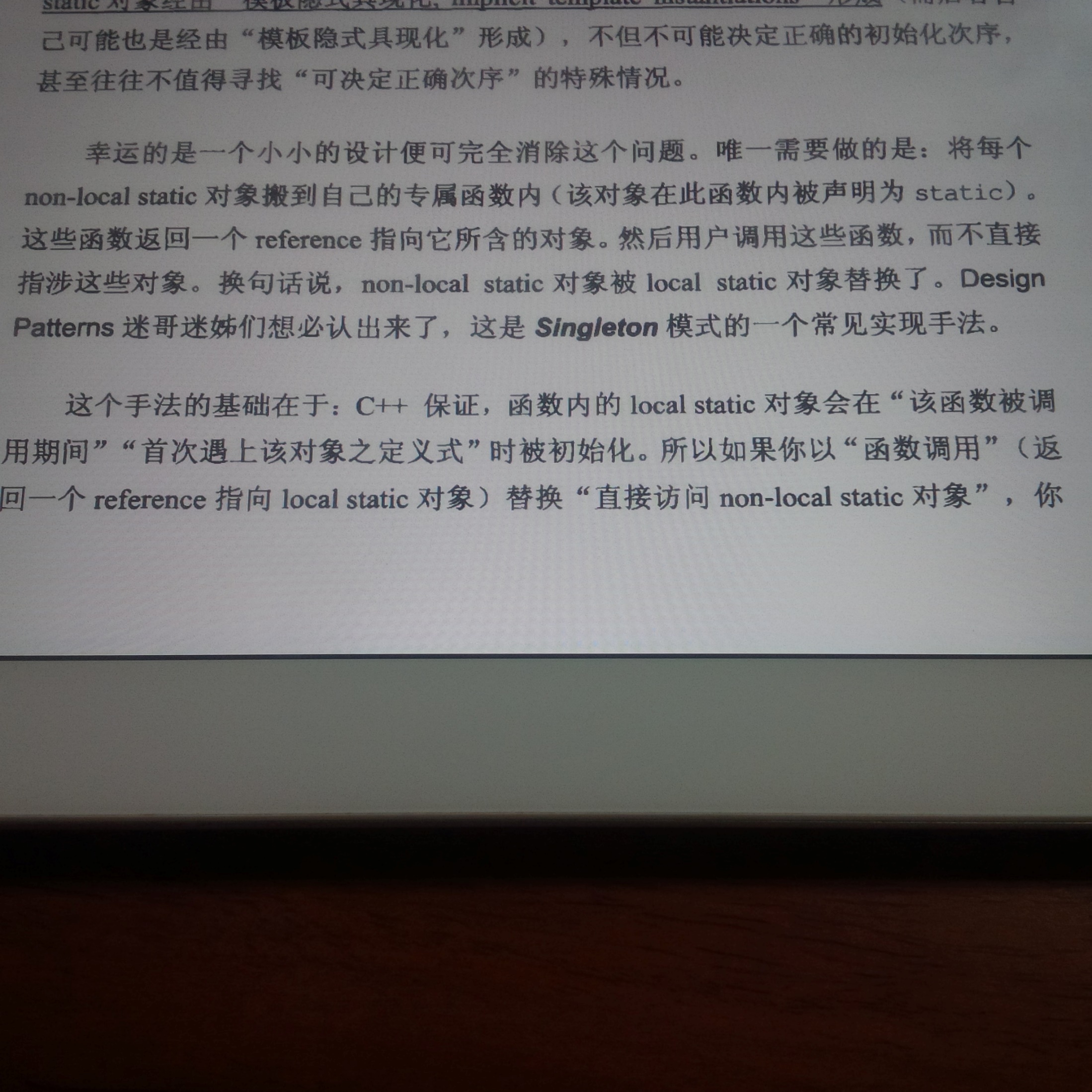
mutable用来在const函数内改变，如求length函数 length声明为mutable

const＿cast＜＞ static＿cast＜＞含义？？？

const成员函数与非const成员函数调用选择？？？

用local static代替non－local static解决初始化的问题

计算机生成了可选文字:
以 此 技 术 施 行 于 tfs 和 tempDir 身 上 ， 结 果 如 下 ， 
class Fi 工 System { 
FileSystem& tfs 〈 ） 
static FileSystem fs; 
return fS 冫 
class Directory { 
Directory ： ： Directory （ params ） 
/ / 同 前 
/ / 这 个 函 数 用 来 替 换 对 
//Fi1eSystemclass 中 可 能 
/ / 定 义 并 初 始 化 一 个 1 
/ / 返 回 一 个 fi n 指 向 上 受 
/ / 同 前 
/ / 同 前 ， 但 原 本 的 reference 
/ / 现 在 改 为 tfs 0 
std ： ： size 乜 disks ： tfs 0 •num-Disks （ ） 冫 
Directory& tempDir 0 
static Directory td; 
return 乜 d 冫 
/ / 这 个 函 数 用 来 替 换 tempDi 
/ / 它 在 Directory class 中 
/ / 定 义 并 初 始 化 1 № 又 
/ / 返 回 一 个 fi n 指 向 上 
这 么 修 改 之 后 ， 这 个 系 统 程 序 的 客 户 完 全 像 以 前 一 样 地 用 它 ， 
们 现 在 使 用 tfs 0 和 tempDir() 而 不 再 是 tfs 和 tempDiro 也 就 是 
返 回 的 “ 指 向 static 对 象 " 的 references, 而 不 再 使 用 static 对 象 自 
这 种 结 构 下 的 reference-returning 函 数 往 往 十 分 单 纯 ： 第 一 
个 local static 
回 它 · 1 寺 量 皿 宀 ' 、 一 “ 一 



static函数生命期 extern声明 global函数与static函数？？？？

只有claas中含有virtual函数时 才会明确虚析构函数 避免不必要内存开销和移植性的降低

explicit关键字？

检查析构函数调用顺序 vptr相关 用指针进行初始化与直接初始化区别？？？

不要在构造 析构函数中调用虚函数

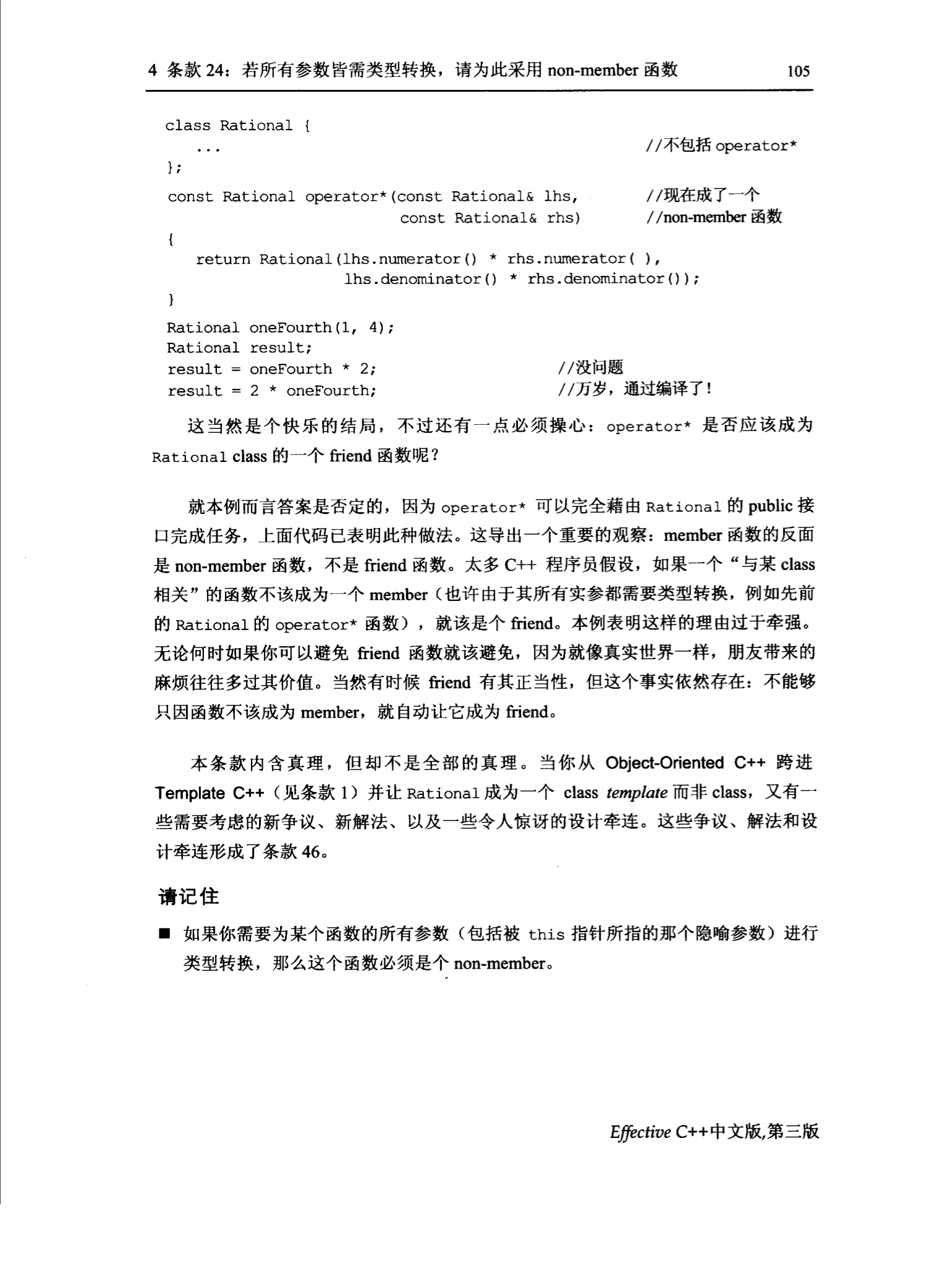
=重载由于左结合返回为\*this的引用？？？如果直接返回对象呢？？？

对于类对象等尽量使用pass by reference 不要pass by value因为可以减少构造函数调用 但是如果一定要构造新值pass by value以防止返回引用时引用被delete，指向空值。如使用场景 a= b\*c中 operator\*函数

派生类无法访问基类私有变量？？？？

隐式转换必须构造函数没有声明explicit。需要检查？？？？

计算机生成了可选文字:
4 设 计 与 声 明 
再 次 看 看 先 前 成 功 的 那 个 调 用 。 注 意 其 第 二 参 数 是 整 数 2 ， 但 
Rational ： ： 。 pera 乜 。 r * 需 要 的 实 参 却 是 个 Rational 对 象 。 这 里 发 生 了 什 么 事 ？ 为 什 
么 2 在 这 里 可 被 接 受 ， 在 另 一 个 调 用 中 却 不 被 接 受 ？ 
因 为 这 里 发 生 了 所 谓 隐 式 类 型 转 换 (implicit 呼 匹 conversion) 。 编 译 器 知 道 你 正 在 
传 递 一 个 int ， 而 函 数 需 要 的 是 Rational; 但 它 也 知 道 只 要 调 用 Rational 构 造 函 数 
并 赋 予 你 所 提 供 的 int ， 就 可 以 变 出 一 个 适 当 的 Rational 来 。 于 是 它 就 那 样 做 了 。 换 
句 话 说 此 一 调 用 动 作 在 编 译 器 眼 中 有 点 像 这 样 ： 
const Rational temp （ 2 
result oneHaIf * 乜 en 爳 冫 
/ / 根 据 2 建 立 一 个 暂 时 性 的 Rational 对 象 。 
/ / 等 同 于 oneHa1f.operator* (temp); 
当 然 ， 只 因 为 涉 及 non-expliclt 构 造 函 数 ， 编 译 器 才 会 这 样 做 。 如 果 Rational 
构 造 函 数 是 explicit, 以 下 语 句 没 有 一 个 可 通 过 编 译 ： 
result = oneHa1f * 2 
result ： 2 * oneHaIf; 
／ / 错 误 ！ （ 在 explicit 构 造 函 数 的 情 况 下 ） 
/ / 无 法 将 2 转 换 为 一 个 Rationale 
/ / 一 样 的 错 误 ， 一 样 的 问 题 。 
这 就 很 难 让 Rational class 支 持 混 合 式 算 术 运 算 了 ， 不 过 至 少 上 述 两 个 句 子 的 行 
为 从 此 一 致 0 。 
然 而 你 的 目 标 不 仅 在 一 致 性 ， 也 要 支 持 混 合 式 算 术 运 算 ， 也 就 是 希 望 有 个 设 计 能 
让 以 上 语 句 通 过 编 译 。 这 把 我 们 带 回 到 上 述 两 个 语 句 ， 为 什 么 即 使 Rational 构 造 函 数 
不 是 explicit' 仍 然 只 有 一 个 可 通 过 编 译 ， 另 一 个 不 可 以 ： 
result ： oneHa1f 女 2 冫 
2 * oneHaIf; 
result 
/ / 没 问 题 （ 在 non xplicit 构 造 函 数 的 情 况 下 ） 
/ / 错 误 ！ （ 甚 至 在 n 皿 № it 构 造 函 数 的 情 况 下 ） 
结 论 是 ， 只 有 当 参 数 被 列 于 参 数 列 (parameterlist) 内 ， 这 个 参 数 才 是 隐 式 类 型 转 
换 的 合 格 参 与 者 。 地 位 相 当 于 “ 被 调 用 之 成 员 函 数 所 隶 属 的 那 个 对 象 ” 一 一 即 thIS 对 
象 一 一 的 那 个 隐 喻 参 数 ， 绝 不 是 隐 式 转 换 的 合 格 参 与 者 。 这 就 是 为 什 么 上 述 第 一 次 调 
用 可 通 过 编 译 ， 第 二 次 调 用 则 否 ， 因 为 第 一 次 调 用 伴 随 一 个 放 在 参 数 列 内 的 参 数 ， 第 
二 次 调 用 则 否 。 
然 而 你 一 定 也 会 想 要 支 持 混 合 式 算 术 运 算 。 可 行 之 道 终 于 拨 云 见 日 ： 让 operator* 
成 为 一 个 non-member 函 数 ， 俾 允 许 编 译 器 在 每 一 个 实 参 身 上 执 行 隐 式 类 型 转 换 ： 
伲 c + + 中 文 版 ， 第 三 版 



不要typedef数组 防止new后delete缺少[]

如typedef string arry[4]；

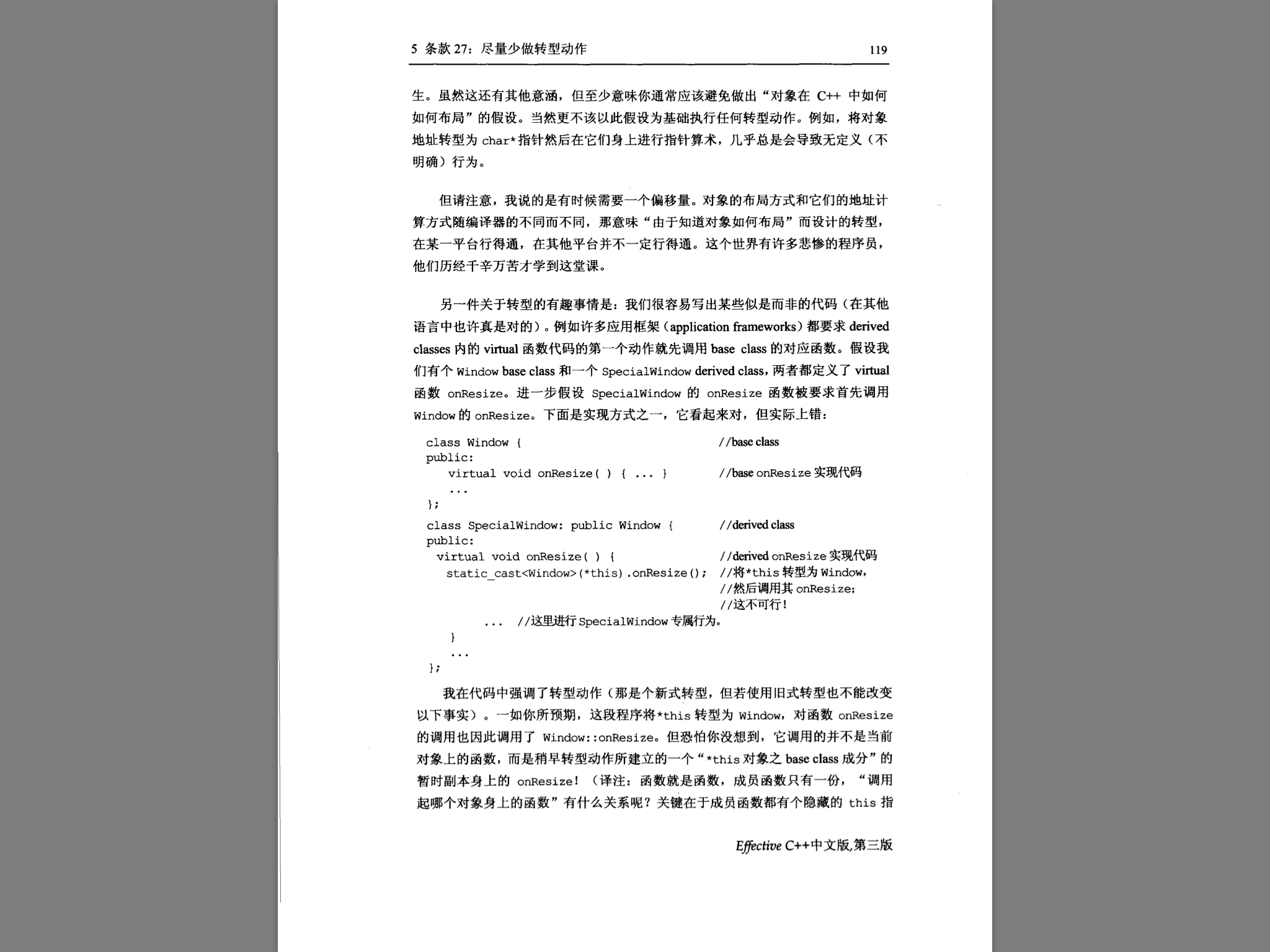
string \*pal = new arry;

Delete [] pal

其实可以直接用vector<string>代替

以独立语句将new对象置入智能指针，不要在函数声明实现 防止内存管理错误

dynamic\_cast不能将继承class的指针转换到base类的指针来调用基类的函数。一般用来将基类指针转换调用继承类的函数



based class指针指向derived class指针指针偏移？？？？

通过base指针指向虚函数 调用derived函数细节？？？

尽量class类中返回值不要返回local stack引用即不要指向对象内部。对于指向外部的情况 参考条例4 条例10 20 21等 防止未加const被用户改变或者指针悬挂。

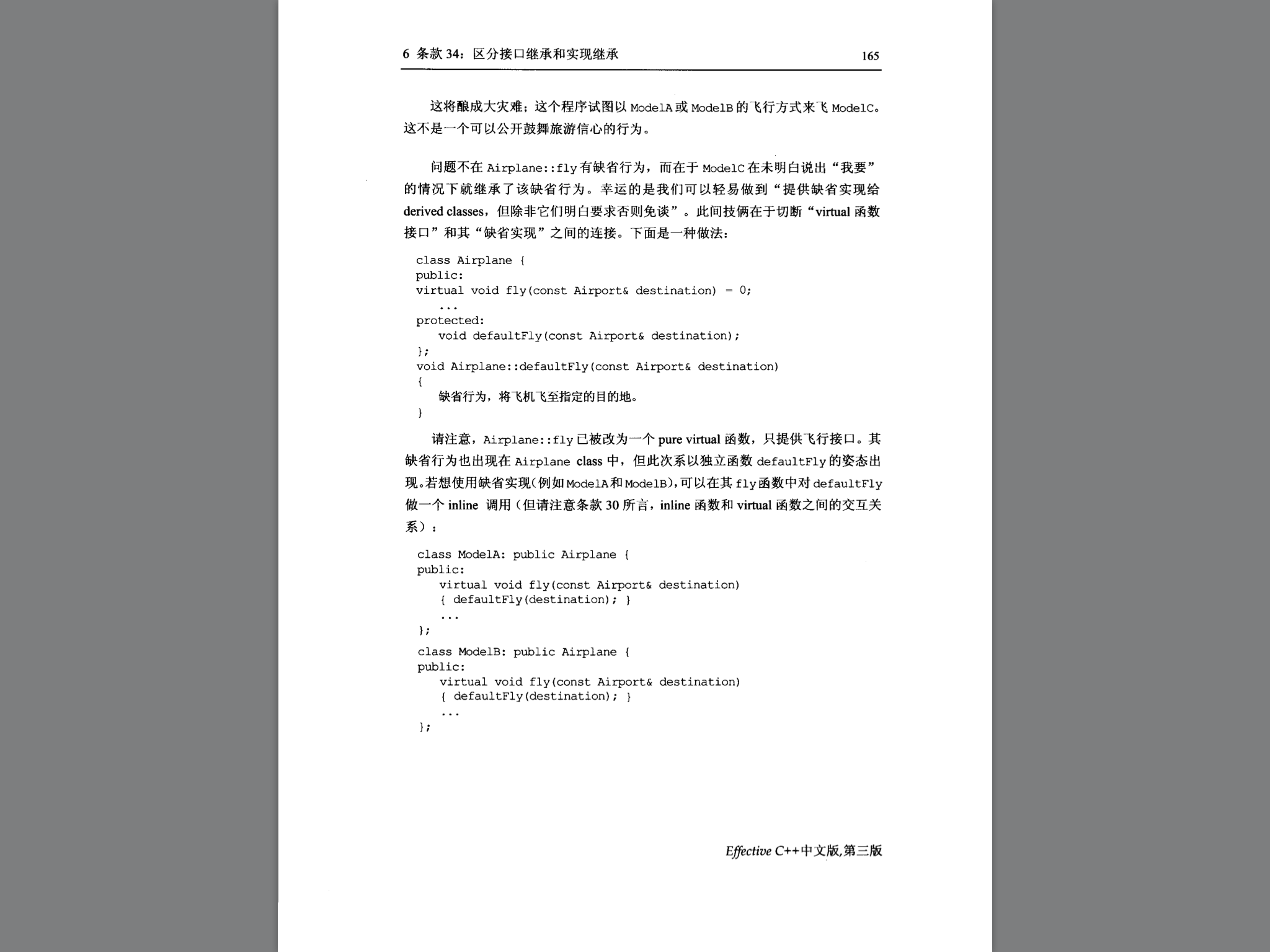


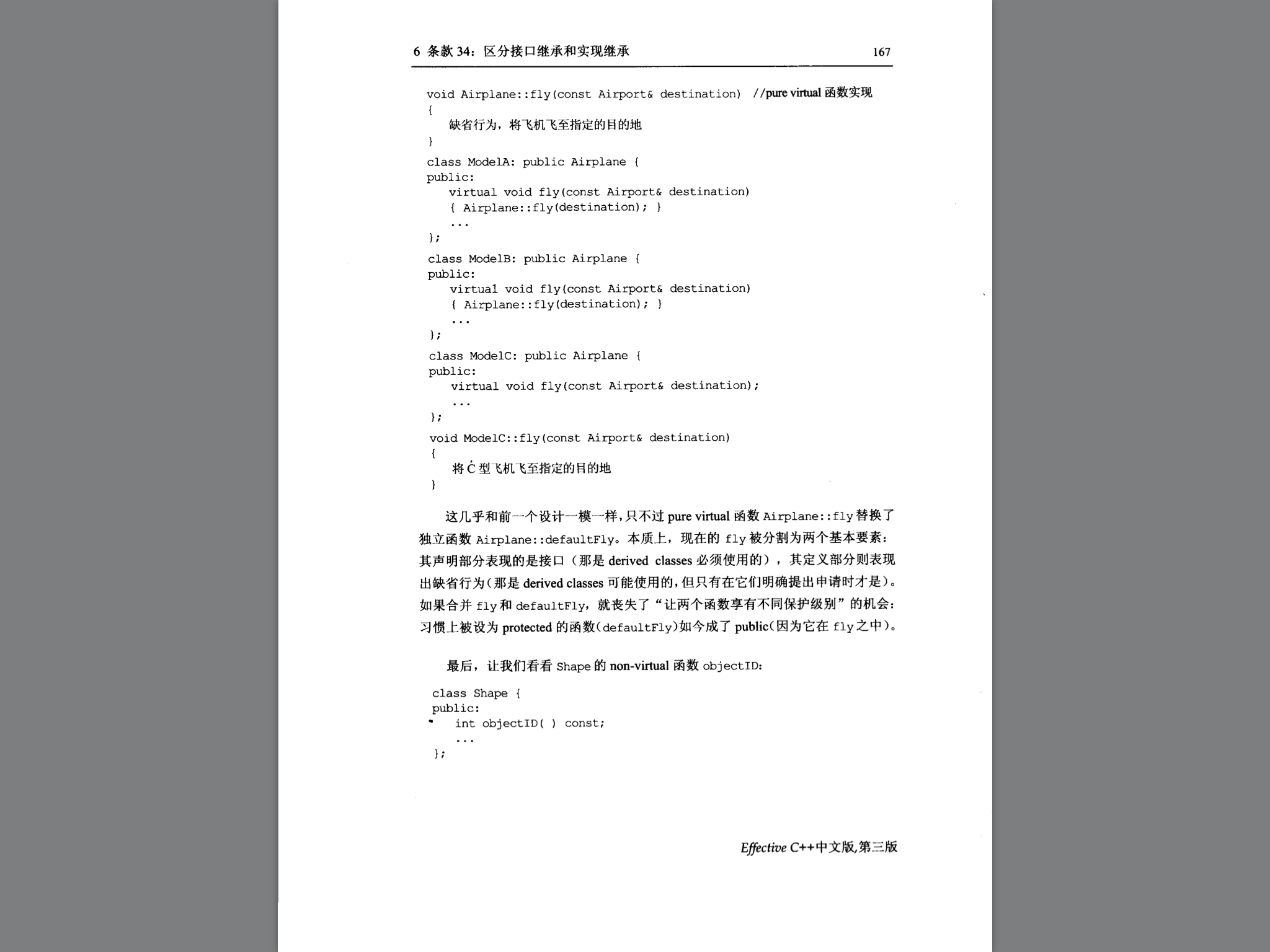
纯虚函数 的细节

inline函数调用 条款30 34

条款34

注意纯虚函数是可以有默认实现的 但是这种方法一般不常用 书中举的例子为defoult fly的实现 利用纯虚函数一定要重载实现 可以保证继承函数不会错误重载默认的fly方式。 但是这种方法可以在基函数中构造一个provited函数defoultfly 让fly函数调用即可。





基函数中 纯虚函数一定要重载

虚函数可以重载 有默认实现

非虚函数 不能被重载。这些是在创建类的接口时候应该考虑到的

条款 35

NVI手法 即将virtual函数定义为非public函数 用另一个public函数进行调用 优势是可以进行事前事后处理 如进程锁 验证约束条件等 【需要细看？？？

对于析构函数不能声明为NVI 因为在继承中需要调用

？？？检查protected 等作用范围

由函数指针实现的策略 即将原先的虚函数写在类的外部 通过函数指针进行调用实现

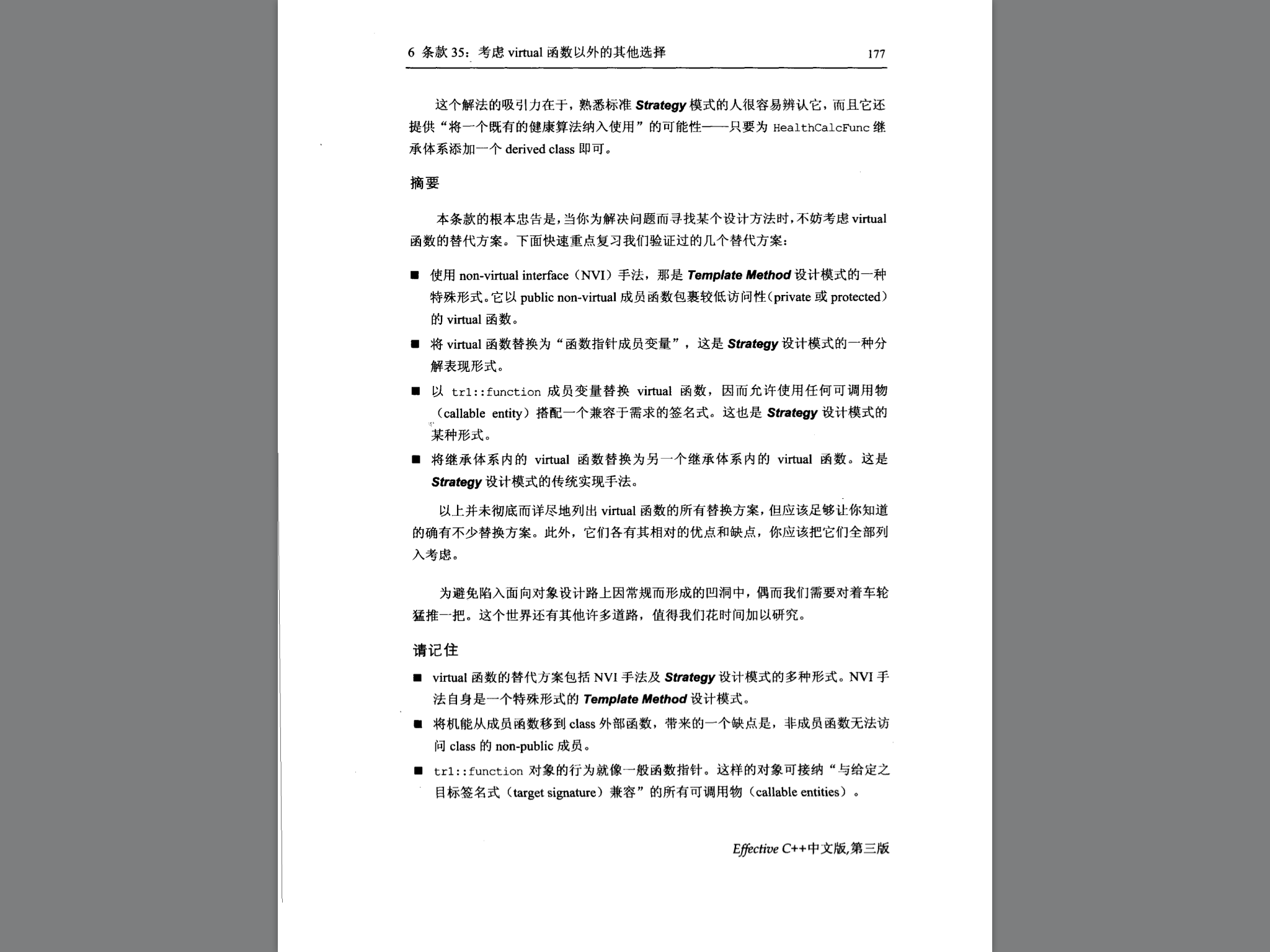
计算机生成了可选文字:
172 
class GameCharacter; 
6 
/ (forward declaration) 
int defaultHeaIthCaIc (const GameCharacter& gc) ; 
class GameCharacter 
public : 
typedef int ( *Heal thCa1cFunc) (const GarneCharacter&) ; 
explicit GameCharacter (HealthCa1cFunc hcf defaultHea1thCa1c) 
healthFunc (hcf) 
int healthVaIue ( ) const 
return healthFunc ( *this) ; 
private : 
Heal thCaIcFunc healthFunc; 
Strategy 
virtual 
class EvilBadGuy: public GameCharacter ( 
public : 
explicit EvilBadGuy (HealthCaIcFunc hcf — defaultHea1thCa1c) 
GameCharacter (hcf) 
int loseHea1thQuick1y (const GameCharacter&) ; 
int loseHeaIthS10wIy (const GameCharacter&) ; 
EvilBadGuy ebgl (loseHeaIthQuickIy) ; 
EvilBadGuy ebg2 (loseHea1thS10w1y) ; 
• GameCharacter 
setHea1thCa1cu1ator, 
" GameCharacter " 
defaultHeaIthCa1c EvilBadGuy non-public 

这种方案好处是可以对相同类的不同实体选择不同的函数实现 但是函数不能取到非public部分的变量解决方法是降低封装

所以需要考虑方案优点，不同实体有不同函数 是否能抵消降低封装带来的缺点

使用tr1:：function 来使得上面的方法返回值进一步增加弹性 因为用函数指针 返回值是固定的。

使用经典的strategy模式 将虚函数放到另一个类中 使用一个指针指向那个类的对象！注意是对象不是函数 所以需要先实例化



条款36

不要重定义非虚函数 因为会发生

若基类的指针指向继承类 它调用重定义的函数其实是基类的函数 而不是继承类中重写的 这样会产生二义性。应该用虚函数实现

查询二义性？？多继承等？？？