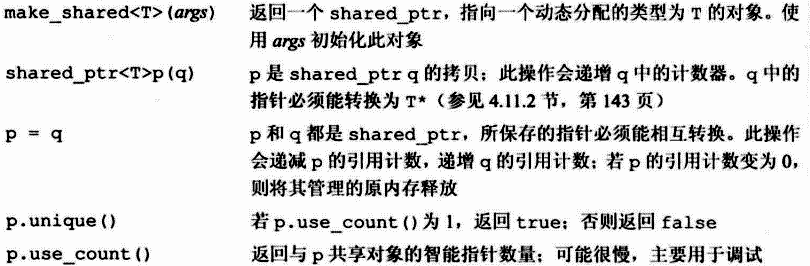
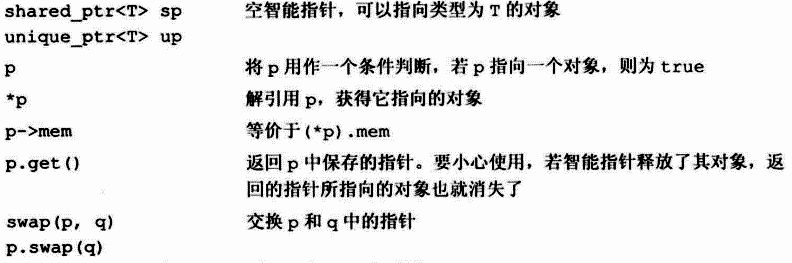
1. 静态内存保存局部static对象、类static数据成员、全局变量；占内存保存定义在函数内非static对象；分配在静态内存或占内存中的对象由编译器自动创建和销毁；(P436)；
2. 自由空间(free store)或堆(heap)：程序可用的内存池，保存动态分配的对象；(P436)；
3. 动态分配的(dynamically allocated)：在自由空间中分配的对象。在自由空间中分配的对象直到被显式释放或程序结束才会销毁；(P436)；

动态内存与智能指针

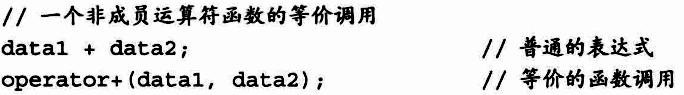


1. shared\_ptr：提供所有权共享的智能指针；对共享对象来说，当最后一个指向它的shared\_ptr被销毁时会被释放；(P436)；
2. unique\_ptr：提供独享所有权的智能指针；当unique\_ptr被销毁时，它指向的对象被释放；unique\_ptr不能直接拷贝或赋值；(P436)；
3. make\_shared函数：；(P400)；
4. 使用动态内存的原因：①程序不知道自己需要使用多少对象；②程序不知道所需对象的准确类型；③程序需要在多个对象间共享数据；(P403)；

动态数组

重载运算符基本概念

1. 重载的运算符(overloaded operator)：①重定义了某种内置运算符的含义的函数；重载的运算符函数含有关键字 operator,之后是要定义的符号；③重载运算符的优先级、结合律、运算对象数量都与其内置版本一致；③除重载的函数调用运算符operator()外不能含有默认实参；(P523)；
2. 重载的运算符的参数：①当一个重载的运算符是成员函数时，this绑定到左侧运算对象；②成员运算符函数（显式）参数数量比运算对象数量少一个；③重载的运算符必须含有至少一个类类型的运算对象（或是类的成员，或至少包含一个类类型的参数），因此作用在内置类型运算对象上时，无法改变该运算符的含义；(P490)；
3. 非成员运算符函数的等价调用和对成员运算符函数的等价调用：(P491)；



1. 不应重载的运算符：①通常不应该重载逗号（,）、取地址（&）、逻辑与（&&）和逻辑或（||）运算符；②其中，逻辑与运算符、逻辑或运算符、逗号运算符的运算对象求值顺序无法保留下来；③&&和||运算符的重载版本不能保留短路求值属性，两个运算对象总是会被求值；④C++语言已经规定逗号运算符和取地址运算符运用于类类型对象时的特殊含义；(P492)；
2. 运算符重载基本原则：①赋值=、下标[]、调用()和成员访问箭头->运算符必须是成员；②复合赋值运算符一般来说应该是成员，但并非必须；③改变对象状态的运算符或者与给定类型密切相关的运算符，如递增、递减和解引用运算符，通常应该是成员；④具有对称性的运算符可能转换任意一端的运算对象，例如算术、相等性、关系和位运算符等，因此它们通常应该是普通的非成员函数；（当我们把运算符定义成成员函数时，它的左侧运算对象必须是运算符所属类的一个对象）；(P493)；

输入和输出运算符

1. 重载输出运算符：①通常情况下，输出运算符的第一个形参是一个非常量对象的引用（非常量是因为向流写入内容会改变其状态，引用是因为我们无法直接复制一个ostream对象）；②第二个形参是一个常量的引用，该常量是我们想要打印的类类型（引用的原因是我们希望避免复制实参，常量是因为打印对象不会改变对象的内容）；③为了与其他输出运算符保持一致，operator<<一般要返回它的ostream形参；输入输出运算符必须是非成员函数，如果是成员函数则其左侧运算对象将是类的一个对象，因此IO运算符一般被声明成友元；④输出运算符应该主要负责打印对象的内容而非控制格式，因此不应该打印换行符；(P494)；
2. 重载输入运算符：①输入运算符必须处理输入可能失败的情况，如if(istream)；②当读取操作发生错误时，输入运算符应该负责从错误中恢复(P495)；

算术和关系运算符

1. 算数和关系运算符基本原则：①算术和关系运算符定义成非成员函数，以允许对左侧或右侧的运算对象进行转换；②形参都是常量的引用，因为一般不需要改变运算对象的状态；③不能返回应用，算术运算符通常会得到一个新值位于一个局部变量之内；④使用复合赋值来定义算术运算符；(P497)；
2. 相等运算符：①如果一个类含有判断两个对象是否相等的操作，则它显然应该把函数定义成operator==而非一个普通的命名函数（无须记忆一个全新的函数名字，容易使用标准库容器和算法）；②如果类定义了operator==则该运算符应该能判断一组给定的对象中是否含有重复数据；③相等运算符应该具有传递性；④如果类定义了operator=，则这个类也应该定义operator!=；⑤相等运算符和不相等运算符中的一个应该把工作委托给另外一个；(P498)；
3. 关系运算符：如果两个对象是!=的，那么一个对象应该<另外一个；(P498)；

赋值运算符

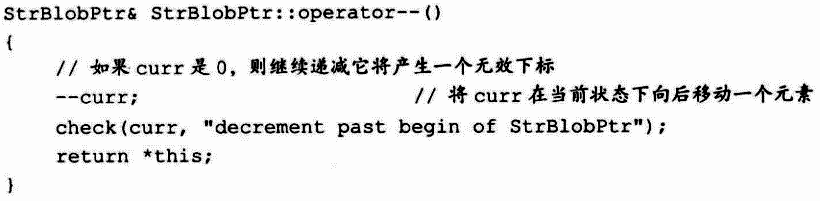
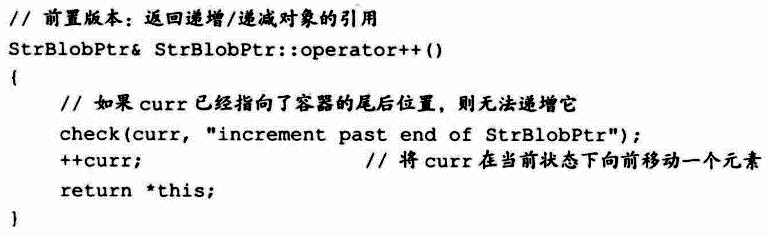
1. 拷贝赋值和移动赋值运算符：除上述两种外，标准库 vector类还定义了第三种赋值运算符，该运算符接受花括号内的元素列表作为参数；重载的赋值运算符必须先释放当前内存空间，再创建一片新空间；赋值运算符必须定义为成员函数；(P499)；
2. 复合赋值运算符：复合赋值运算符可以不是类的成员，但建议将复合赋值运算符和赋值运算符都定义成类成员，且复合赋值运算符要返回其左侧运算对象的引用；(P500)；

下标运算符

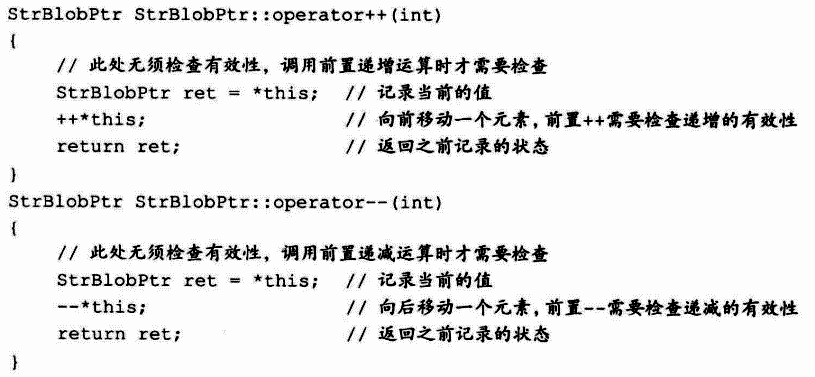
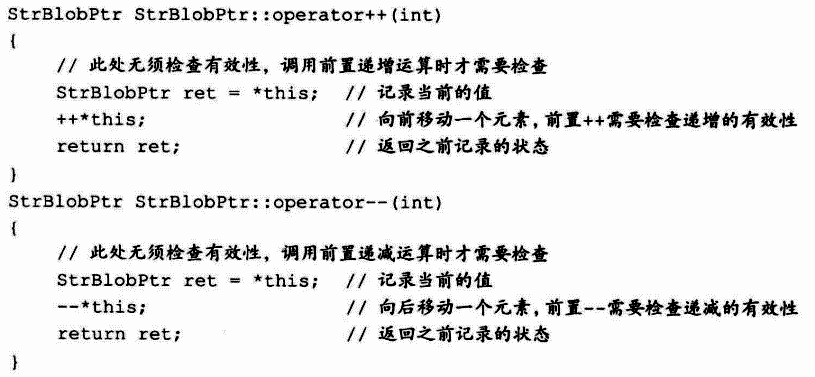
1. ①下标运算符必须是成员函数；②下标运算符以所访问元素的引用作为返回值（下标可以出现在赋值运算符的任意一端）；③同时定义下标运算符的常量版本和非常量版本（一个返回普通引用，另一个是类的常量成员并且返回常量引用）；(P501)；

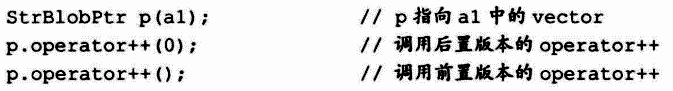
递增递减运算符

1. 定义增和递减运算符的类应该同时定义前置版本和后置版本；这些运算符通常应该被定义成类的成员；前置运算符应该返回递增或递减后对象的引用；递增递减运算符应该检查索引值是否有效(P502)；



1. 区分前置和后置运算符：后置版本接受一个额外的（不被使用）int类型的形参，编译器为这个形参提供一个值为0的实参；后置运算符应该返回对象的原值（递増或递减之前的值），返回的形式是一个值而非引用；我们的后置运算符调用各自的前置版本来完成实际的工作；(P503)；



1. 显示的调用后置运算符：(P504)；

成员访问运算符

函数调用运算符

重载、类型转换与运算符

面向对象的编程概述

1. 面向对象编程(object-oriented programming)：利用数据抽象、继承以及动态绑定等技术编写程序的方法；(P576)；
2. 继承( inheritance)：由一个已有的类（基类）定义一个新类（派生类）的编程技术；派生类将继承基类的成员；(P576)；
3. 基类( base class)：可供其他类继承的类；基类的成员也将成为派生类的成员；(P575)；
4. 派生类( derived class)：从其他类派生而来的类；派生类可以覆盖其基类的虚函数，也可以定义自己的新成员；派生类的作用域嵌套在基类作用域当中：派生类的成员能直接访问基类的成员；(P575)；
5. 类派生列表(class derivation list)：形式是在类名后一个冒号，后面紧跟以逗号分隔的基类列表，每个基类前面可以有类型说明符；如果没有提供访问说明符，则当派生类通过关键字 struct定义时继承是公有的；而当派生类通过关键字class定义时继承是私有的；(P575)；
6. 虚函数(virtual function)：用于定义类型特定行为的成员函数；通过引用或指针对虚函数的调用直到运行时才被解析，依据是引用或指针所绑定对象的类型；(P576)；
7. 动态绑定(dynamic binding)：直到运行时才确定到底执行函数的哪个版本；在C++语言中，动态绑定的意思是在运行时根据引用或指针所绑定对象的实际类型来选择执行虚函数的某一个版本；(P575)；