挑战30天C 入门极限：对C 递增(增量)运算符重载的思考 PDF 转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/134/2021_2022__E6_8C_91_E6_88_9830_E5_A4_c97_134496.htm 在前面的章节中我们已经接触过递增运算符的重载，那时候我们并没有区分前递增与后递增的差别，在通常情况下我们是分别不出 a与a 的差别的，但的确他们直接是存在明显差别的。 先看如下代码： C 代码 #include iostream> using namespace std. int main() { int a=0. (a).//正确,(a)返回的是左值 (a) .//错误,(a)返回的不是左值 system("pause"). } 代码中(a) 编译出错误，返回“ ”需要左值的错误，这正是前递增与后递增的差别导致的，那么又是为什么呢？ 原因主要是由C 对递增(增量)运算符的定义引发的。 他们之间的差别主要为以下两点： １.运算过程中，先将对象进行递增修改，而后返回该对象（其实就是对象的引用）的叫前递增(增量)运算。在运算符重载函数中采用返回对象引用的方式编写。 ２.运算过程中，先返回原有对象的值，而后进行对象递增运算的叫后递增(增量)运算。在运算符重载函数中采用值返回的方式编写（这也正是前面(a) 出错误的原因，(a)返回的不是引用，不能当作左值继续参加扩号外部的 运算），重载函数的内部实现必须创建一个用于临时存储原有对象值的对象，函数返回的时候就是返回该临时对象。 那么在编写运算符重载函数的时候我们该如何区分前递增运算符重载函数与后递增运算符重载函数呢？ 方法就是：在后递增运算符重载函数的参数中多加如一个int标识，标记为后递增运算符重载函数。 具体见如下实例（例一为非成员方式，例二为成员方式）： C 代码 //例一 //程序作者:管宁 //站

点:www.cndev-lab.com //所有稿件均有版权,如要转载,请务必著名出处和作者 #include iostream> using namespace std. class Test { public: Test(int a=0) { Test::a = a. } friend Testamp.). friend Test operator (Testamp. operator (Test amp.temp,int)//后递增,int在这里只起到区分作用,事实上并没有实际作用 { Test rtemp(temp).//这里会调用拷贝构造函数进行对象的复制工作 temp.a . return rtemp. } int main() { Test a(100). (a). cout cout//这里正是体现后递增操作先返回原有对象值地方 cout (a) . cout//由于后递增是值返回状态，所以(a) 只对a做了一次递增操作，操作后为104而非105。 system("pause"). } C 代码 //例二 //程序作者:管宁 //站点:www.cndev-lab.com //所有稿件均有版权,如要转载,请务必著名出处和作者 #include iostream> using namespace std. class Test { public: Test(int a=0) { Test::a = a. } Testamp. Test::operator ()//前递增 { this->a . return *this. } Test Test::operator (int)//后递增 { Test rtemp(*this).//这里会调用拷贝构造函数进行对象的复制工作 this->a . return rtemp. } int main() { Test a(100). (a). cout cout//这里正是体现后递增操作先返回原有对象值地方 cout (a) . cout//由于后递增是值返回状态，所以(a) 只对a做了一次递增操作，操作后为104而非105。 system("pause"). } 通过对前后递增运算的分析，我们可以进一步可以了解到，对于相同情况的单目运算符重载我们都必须做好这些区别工作，保证重载后的运算符符合要求。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com