C 箴言：声明为非成员函数时机 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/133/2021_2022_C___E7_AE_B4_E8_A8_80_EF_c97_133671.htm 我谈到让一个类支持隐式类型转换通常是一个不好的主意。当然，这条规则有一些例外，最普通的一种就是在创建数值类型时。例如，如果你设计一个用来表现有理数的类，允许从整数到有理数的隐式转换看上去并非不合理。这的确不比 C 的内建类型从 int 到 double 的转换更不合理（而且比 C 的内建类型从 double 到 int 的转换合理得多）。在这种情况下，你可以用这种方法开始你的 Rational 类： class Rational { public: Rational(int numerator = 0, // ctor is deliberately not explicit. int denominator = 1). // allows implicit int-to-Rational // conversions int numerator() const. // accessors for numerator and int denominator() const. // denominator - see Item 22 private: ... }. 你知道你应该支持类似加，乘等算术运算，但是你不确定你应该通过成员函数还是非成员函数，或者，非成员的友元函数来实现它们。你的直觉告诉你，当你拿不准的时候，你应该坚持面向对象。你知道这些，于是表示，有理数的乘法与 Rational 类相关，所以在 Rational 类内部为有理数实现 operator* 似乎更加正常。与直觉不符，将函数放置在它们所关联的类的内部的主意有时候与面向对象的原则正好相反，但是让我们将它放到一边，来研究一下将 operator* 作为 Rational 的一个成员函数的主意： class Rational { public: ... const Rational operator*(const Rational&amp. rhs) const. }. （如果你不能确定为什么这个函数声明为这个样子返回一个 const by-value 的结果，却持有一个

reference-to-const 作为它的参数。） 这个设计让你在有理数相乘时不费吹灰之力： Rational oneEighth(1, 8). Rational oneHalf(1, 2). Rational result = oneHalf * oneEighth. // fine result = result * oneEighth. // fine 