C 箴言：视类设计为类型设计 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_133673.htm 在 C 中，就像其它面向对象编程语言，可以通过定义一个新的类来定义一个新的类型。作为一个 C 开发者，你的大量时间就这样花费在增大你的类型系统。这意味着你不仅仅是一个类的设计者，而且是一个类型的设计者。重载函数和运算符，控制内存分配和回收，定义对象的初始化和终结过程这些全在你的掌控之中。因此你应该在类设计中倾注大量心血，接近语言设计者在语言内建类型的设计中所倾注的大量心血。 设计良好的类是有挑战性的，因为设计良好的类型是有挑战性的。良好的类型拥有简单自然的语法，符合直觉的语义，以及一个或更多高效的实现。在 C 中，一个缺乏计划的类设计，使其不可能达到上述任何一个目标。甚至一个类的成员函数的执行特性可能受到它们是被如何声明的影响。 那么，如何才能设计高效的类呢？首先，你必须理解你所面对的问题。实际上每一个类都需要你面对下面这些问题，其答案通常就导向你的设计的限制因素： 你的新类型的对象应该如何创建和销毁？如何做这些将影响到你的类的构造函数和析构函数，以及内存分配和回收的函数（operator new，operator new[]，operator 0delete，和 operator 0delete[]）的设计，除非你不写它们。 对象的初始化和对象的赋值应该有什么不同？这个问题的答案决定了你的构造函数和你的赋值运算符的行为和它们之间的不同。这对于不混淆初始化和赋值是很重要的，因为它们相当于不同的函数调用。 以值传递（passed by value）对于你的新类型的

对象意味着什么？记住，拷贝构造函数定义了一个新类型的传值（pass-by-value）如何实现。 你的新类型的合法值的限定条件是什么？通常，对于一个类的数据成员来说，仅有某些值的组合是合法的。那些组合决定了你的类必须维持的不变量。这些不变量决定了你必须在成员函数内部进行错误检查，特别是你的构造函数，赋值运算符，以及 "setter" 函数。它可能也会影响你的函数抛出的异常，以及你的函数的异常规范（exception specification）（你用到它的可能性很小）。 你的新类型是否适合放进一个继承图表中？如果你从已经存在的类继承，你将被那些类的设计所约束，特别是它们的函数是 virtual 还是 non-virtual。如果你希望允许其他类继承你的类，将影响到你是否将函数声明为 virtual，特别是你的析构函数。 你的新类型允许哪种类型转换？你的类型身处其它类型的海洋中，所以是否要在你的类型和其它类型之间有一些转换？如果你希望允许 T1 类型的对象隐式转型为 T2 类型的对象，你就要么在 T1 类中写一个类型转换函数（例如 ，operator T2），要么在 T2 类中写一个非显式的构造函数，而且它们都要能够以单一参数调用。如果你希望仅仅允许显示转换，你就要写执行这个转换的函数，而且你还需要避免使它们的类型转换运算符或非显式构造函数能够以一个参数调用。 对于新类型哪些运算符和函数有意义？这个问题的答案决定你应该为你的类声明哪些函数。其中一些是成员函数，另一些不是。 哪些标准函数不应该被接受？你需要将那些都声明为 private。 你的新类型中哪些成员可以被访问？这个问题的可以帮助你决定哪些成员是 public，哪些是 protected，以及哪些是 private。它也可以帮助你决定哪些类和／或函数

应该是友元，以及一个类嵌套在另一个类内部是否有意义。什么是你的新类型的 "undeclared interface"？它对于性能考虑，异常安全（exception safety），以及资源使用（例如，锁和动态内存）提供哪种保证？你在这些领域提供的保证将强制影响你的类的实现。 你的新类型有多大程度的通用性？也许你并非真的要定义一个新的类型。也许你要定义一个整个的类型家族。如果是这样，你不需要定义一个新的类，而是需要定义一个新的类模板。 一个新的类型真的是你所需要的吗？是否你可以仅仅定义一个新的继承类，以便让你可以为一个已存在的类增加一些功能，也许通过简单地定义一个或更多非成员函数或模板能更好地达成你的目标。 回答这些问题是困难的，所以定义高效的类是有挑战性的。既然，在 C 中用户自定义类生成的类型至少可以和内建类型一样好，那就做好它，它会使一切努力都变的有价值。 