瘦身前后兼谈C 语言进化(3) PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/469/2021_2022__E7_98_A6_E8_BA_AB_E5_89_8D_E5_c67_469827.htm 如果不想卷入C functional的噩梦的话，你也可以这么写：struct Op{int operator()(int a1, int a2) { return f(a1) f(a2). }}.transform(…, Op()). 稍微好一点，但这种做法也有很严重的问题。 为什么Java加入closure，其实还是一个语法问题。从严格意义上，Java的anonymous class已经可以实现出一样的功能了，正如C 的functor一样。然而，代码是给人看的，语言是给人用来写代码的，代码的主要代价在维护，维护则需要阅读、理解。写代码的人不希望多花笔墨来写那些自己本不关心的东西，读代码的人也希望“ 所读即所表 ”，不想看到代码里面有什么弯子，最好是自然语言自然抽象才好呢。 所以，尽管closure是一颗语法糖，但却是一颗很甜很甜的糖，因为有了closure你就可以写：transform(…, (a1, a2){ f(a1) f(a2) }).Simple things should be simple! 此外，closure最强大的好处还是在于对局部变量的方便的引用，设想我们想要创建的表达式是：int weight1 = 0.3, weight2 = 0.6.transform(…, f(_1)*weight1 f(_2)*weight2). 当然，上面的语句是非法的，不过使用closure便可以写成：int weight1 = 0.3, weight2 = 0.6.transform(…, (_1, _2){ f(_1)*weight1 f(_2)*weight2 } ). 用functor class来实现同样的功能则要麻烦许多，一旦麻烦，就会error-prone，一旦error-prone，就会消耗人力，而人力，就是金钱。 C 09也有希望加入lambda，不过这是另一个话题，下回再说。 The Real Dealvariadic templates C 的callback类，google一下，没有一

打也有半打。其中尤数boost.function实现得最为灵活周到。然而，就在其灵活周到的接口下面，却是让人不忍卒读的实现；03年的时候我写的第一篇boost源码剖析就是boost.function的，当时还觉得能看懂那样的代码牛得不行...话说回来，那篇文章主要剖析了两个方面，一个是它对不同参数的函数类型是如何处理的，第二个是一个type-erase设施。其中第一个方面就占去了大部分的篇幅。 简而言之，要实现一个泛型的callback类，就必须实现以下最常见的应用场景：function caller = f.int r = caller(1, 2). // call f 为此function类模板里面肯定要有一个operator()，然而，接下来，如何定义这个operator()就成了问题：templateclass function{operator()(???).}. 