神话与谬误：争论C 前你应当知道什么 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/286/2021_2022__E7_A5_9E_E8_AF_9D_E4_B8_8E_E8_c67_286801.htm 哈雷将军的笑话想必大家都听过。一句话经口口相传，每个人都根据自己的主观意念加以润色，修补，歪曲…到最后就面目全非。这里最关键的一环就是主观意识，在历史学里面有这么一句话，大致意思是历史其实只存在于人的意念之中.就算完全客观的事件，通过不同的人的嘴说出来，造成的心理效应也往往不一样，每个人都会加上那么一两个形容词，驾驭语言能力高的更是能够舌绽莲花，而语言本就有自身的力量，其中的遣词造句对读者构成的心理影响力便应运而生。甚至于同一句话，用不同的语气说出来，都会造成不同的效果。同一句话，站在不同的立场上看，也会根本不是同一个意思。比如“ C 还算是门不错的语言 ”，站在C 拥护者的角度听是在怜悯加诋毁C，而站在C 反对者的角度听却是抬举了C 。 在一个长期被广泛争论的话题中，几乎无可避免的总是存在一些Fallacies和Myths。比如动态amp. i){…}}. std::for_each(v.begin(), v.end(), MyOp()). 这个方案实际很差。一是你还是得写v.begin()、v.end()，二是你得为此定义一整个新类。三是这个新类并不在你使用这个新类(for_each被调用)的点上，因为局部类不能做模板参数。 你要的是lambda function： for_each(v.begin(), v.end(), (intamp. i : v) {…} 可是C 98没有。 鉴于循环结构是编程中最常出现的结构之一。这个问题其实还是比较恼人的，如果你觉得不恼人可能只是因为你适应性习惯了，这未必是好事。比如每次都要

写std::vector::iterator就很让人恼火，如果我换个容器，就要修改一堆std::vector。那用typedef行不行啊?行。可仍然还是需要写一次typedef，我很懒，我什么多余的无用代码都不想写。要知道，每多出一行无用的(并非因表达思想所需要才出现)的代码，就增加一点维护负担，这也正是为什么语言的表达力如此重要的原因。 那怎么办?如果我告诉你，C 98里面其实你也可以写： foreach(int&amp. i , v){ …} 你怎么想? 废话。当然是求之不得了。有这么简洁的表达方式谁还不想用啊。 我需要告诉你的另一个事实是。为了在C 98里面几近完美地实现这个特性，有人把标准的角落挖了个底朝天。不，我不是在为钻语言细节找理由，我只是想告诉你，许多人所认为的钻语言细节的做法，其实一开始大多是由用户实际需求驱动的，这个foreach设施被C 程序员们试图实现了N遍N种做法，可见需求之强烈。可惜绝大多数实现都远远称不上好用，就连现在这个实现的作者也早在03年在CUJ上发了一个实现，也称不上好用。是后来又契而不舍才实现了最终这个真正好用的版本的。 我想说的是，上面这个美好的foreach，当然人人都想用。但问题是要在C 98下实现它只能靠挖标准，这是唯一的途径。要不然就得等语言进化，并忍受若干年，谁愿意?况且这个foreach设施还能作占位符，在C 09来临之前兢兢业业履行其职责，C 09加入内建foreach支持之后只消用正则表达式搜索全局替换，就OK了，没有任何的升级麻烦。 再举一个经典的例子：STL里面的traits。其实traits不应该是traits。traits最自然的实现方式应该是C 09的concept。但STL需要用到静态dispatch技术啊，那怎么办?要么用traits(增加语言复杂性)，要么不用(显然不行)。 再举个经典的例子：模板元编程

。模板元编程有啥用?日常开发者八辈子估计也用不到。但真的吗?没错，日常开发者并不会直接用到。但是，由模板元编程支持的各个boost子库呢?被选入C 0x的TR1的各个子库呢(间接用到)?那日常开发者用不用学模板元编程呢?不用学，根本不用学，这么复杂的技术学什么呢?也就是点技巧上的东西。那为什么偏有人学呢?待会再说。 还有大量的例子就不一一列了。其实STL的traits技术已经能够说明问题了。如果你仔细看一看，你会发现，那些所谓的利用C 黑暗角落的技术，几乎无一不是出现在库开发里面的，而之所以出现在库开发里面，是因为库开发中的需求驱动的为了开发出更好的库。难道你不想用更好的库? 哦，说到“ 更好的库 ”，肯定会有同学有意见了。 C 98都快十年了，标准库还是只有那一套STL。库进展缓慢，到现在GUI库也没有一个标准，都是四分五裂各自为营。网络库也是、文件系统库也是、日志库也是…不过这个问题已经是另一个问题了，容后再说。 问题是，“ 没有标准的库 ”并不意味着“ C 的库不好 ”，后者也并不意味着“ 那些晦涩的技巧并没有提升库的质量 ”，这个逻辑上的两环都不对。实际上，人们所谓的“ 晦涩而复杂的技巧 ”其实正是为了提升库的质量而被挖掘出来的。traits技术提升库的效率(静态转发)，type erase技术使得boost::function可以接受任何签名为void()的函数(灵活性)，包括仿函数，包括boost::bind后的函数。type list技术使得boost::tuple能够接受可变数目的模板参数。policy-based design使得可以对一个设施的功能进行正交分解… 就算把所有流行的C tricks都列出来，你也会发现，其实它们几乎每一个都对应了至少一个实际应用。而实际应用需求哪来的?库设计的需求。但归根到底，是使用库的人

终端程序员的需求。(效率、灵活性、抽象表达力，哪一样不是终端程序员的实际需求呢?) 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