

后记 走向何方

我们的讨论就要结束了，读者或许还会带着疑虑问我：“你本人相信有可能写出无错程序吗？”当然不能绝对地、百分之百地保证这一点。但是可以相信，只要你坚持照做就能写出非常接近于无错的程序。这就象粉刷房间一样，在粉刷房间时可以不弄脏地毯，但必须在地上铺上布，围上挡板，并小心地粉刷，还要下决心坚持到底。同样，读者必须努力剔除代码中的错误，要想做到这点的唯一途径就是按照正确的方向坚持下去。

尽管你把写无错代码置于首位，但是只利用本书中给出的技术还不能完全达到这一目标。事实上，不存在一个能保证编码不出任何错误的准则表。因此，最重要的是，读者自己要坚持建立一个查错表列出你查出的错误，以避免重犯以前犯过的错误。这个表中的某些项也许会使你大吃一惊。

例如，我曾经将一个令人讨厌的微小错误引入了 Excel：在浏览一个文件时，不小心删了一行。当时我没有检测到这一错误，把这一改动了的文件连同其它文件一起合并到主程序。以后别人发现了这个错误并追踪到我这。我就想，怎样才能检测并防止这类错误呢，答案很显然：在把更改了的代码合并到主程序之前，利用源代码控制管理程序列出所做的改变。这一额外的步骤付诸实施并不占用多少时间，在其后的五年中，它帮助我发现了三个重大的错误以及一些不太恰当的小更改。这三个错误是否值五年的努力？对我来说是值得的，因为正如我前面所说，这样做费不了多大的事，我就可以知道不会把不希望的更改带入主程序。再说一遍，改正错误要放在首要地位。

读者或许会发现，评审代码就是为了解决问题，提供较好的文档就是为了帮助开发产品的内部工作人员。如果不采用单元测试，或许你早就这么做了。读者甚至会发现，特意加入第 3 章练习 6 中提到的 DEBUG 代码对帮助测试员是十分有用的。有时候这种解决问题的方法不太切合实际，这时最好的办法就是避开导致错误的算法和实现。

事实上，无法做到完全消除错误，但是通过不懈的努力，可以加大犯同样错误的时间间隔。为帮助大家做到这一点，在附录 A 中给出了程序员使用的检查清单，这个检查清单综合了本书的所有观点。

综上所述，成功地书写无错代码的关键可以总结为一个总的原则

决不允许同样错误出现两次