**6.2.3 When the Going Gets Tough**

**处境艰难之时**

据说肯尼迪总统的父亲约瑟夫·帕特里克·肯尼迪,曾这样教导他的孩子们：“当处境变得艰难,顽强意志者会更努力去迎接挑战!”。但是他从未调试过软件。本节包含了一些做什么会使调试变困难的具体的例子。

* 看下列通常犯错的例子。你有没有：
* 参数传递给一个函数的顺序错了
* 拼错一个名字,例如：当你应该输入一个大写字母，你却输入了小写字母。
* 重新初始化一个变量时失败
* 测试两个浮点值是否完全相等（==）而不是近似等于（记住，浮点运算和你在学校里所学的运算是不一样的）
* 在你想要两个对象相等时（例如：id（L1）==id（L2）），检查它们的数值是否相等（例如：通过写出两列表达式L1==L2来比较）
* 忘记一些内置函数所具有的副作用
* 忘记“（）”会将对函数类型对象的引用变成函数的调用
* 创建了一个没有定义的名称。
* 或者犯一些有你自己特点的错误。
* 停止一直问自己为什么程序没有像你所想象的那样运行，相反的你应该问自己为什么程序会像现在这样运行。那可能会是一个更加容易回答的问题，并且这可能是找出如何修复这个问题的最好的第一步。
* 时刻谨记错误可能并不会出现在你认为它应该出现的地方。如果它真的出现在了那儿，那么你应该早就找到了它。一个找错误实用的方法就是排除那些错误不可能在的地方。就像夏洛克·福尔摩斯所说，“消除所有其他的可能，剩下的那一个就是真相。”
* 尝试向其他人解释这个问题。我们每个人都有盲点，很多时候在向别人解释自己的问题的时候可能会帮助你发现自己所遗漏的东西。一个最好的方法是尝试着去解释为什么错误不能发生在某些地方。
* 不要相信你所读过的任何东西，尤其不要相信那些文档。代码或许并不会像建议中所说的那样运行。
* 停止调试并开始编写文档。这将帮助你从另一个角度找到解决问题的方法。
* 停止调试并离开，明天再继续尝试。这可能意味着在时间上，你会比一直坚持找错误的方法更晚的解决这个问题，但是你可能会花更少的时间来寻找并解决错误。换句话说，这可能是在用滞后的时间换取效率。（学生们，这是一个告诫我们早一些而不是晚一些去做编程错误集的非常好的理由！）

**6.2.4 And When You Have Found “The” Bug**

**在你找到“这”错误之时**

当你认为你找到了程序中的一个错误时，那种想要马上开始编程并调试修改的渴望是几乎不可抗拒的。然而，能使自己放慢一点往往更好一点。记住最终的目标并不是去修补好一个错误，而是要更快更有效率的向没有错误的程序前进。

问问自己这个错误是否能解释所有可观察到的症状，或者它是否只是错误的冰山一角。如果是后者，那么你最好在将这个错误和它引起的其他变化联系起来。举个例子来说明这个观点，你已经发现了错误的原因是因为意外地改变了一个列表，你可以局部地避开这个问题（也许是通过将列表备份），或者可以考虑用多元组来替代列表（因为多元组是保持不变的），这样也许能够消除在代码的中的其他部分出现的相同错误。

在做任何改变之前，尝试着去思考这个“修改”可能产生的其他问题。这样的修改会破坏其他地方吗？它会使程序变得过于复杂吗？修改之后还有机会整理其他地方么？

你要永远确保自己能够回到修改之前。没有什么比在意识到自己做的一长串的修改反而使你离目标越来越远并且没有办法能再次回到你最开始的地方更加让人沮丧。磁盘的空间通常都很充足，所以记得用它来储存你老版本的程序。

最后，如果出现了许多难以解释的错误，你也许应该思考一下逐个找到错误并修改是不是正确的方法，也许你最好思考一下是否有更好的方法来组织你的程序或一些简单的算法,这将更容易修改正确。