# 作业四

1. 翻译

### 6.2.3 当前进的路变得艰难的时候

据说肯尼迪总统的父亲在教育他的孩子的时候说“当前进的路变得艰难的时候，就是困难要走的时候”。但是他从来没有调试过程序。这段话中包含了在调试变得艰难的时候的几个实用线索。

寻找普遍的错误，例如：你有没有

1. 传参数给函数时弄错了顺序，
2. 拼错了名词，例如：本来应该大写的小写了，
3. 检查两个浮点是不是完全相等（==）而不是约等于（注意浮点的计算和学校里学的计算是不一样的），
4. 在你想要两对象相等时（例如：id（L1）==id（L2））检查数值是否相等（例如：通过写出表达式L1==L2来比较两列），
5. 忘记一些内置函数有副作用，
6. 忘记“（）”会将函数类型对象的引用转换成函数调用，
7. 创建了一个没有定义的对象，
8. 或者任何你经常犯的错误。

不要再问自己为什么程序没有像你想要的一样工作，而是问自己为什么它会像现在这样运行。这个问题回答起来应该更简单，而且可能也是找出解决方法的好的第一步。

要记住错误可能不是出现在你以为的地方。如果是在那儿，你早就发现了。一个实用的决定从哪儿开始检查的方法就是问自己错误不可能在哪儿。就像福尔摩斯说的：“消除所有其他的可能，剩下的就是真相。”

试着向其他人阐述出现的问题。我们都有盲点。很多时候就在试图去向别人解释问题时你就会发现自己忘记了什么。一个很好的去解释的问题是为什么错误不会发生在某个地方。

不要相信你读的任何东西，特别是文献资料。程序可能不会像意见中所说的那样工作。

不要再调试程序而是开始把它记录下来。这样做可以帮助你从另一个角度看问题。

今天先到这儿明天再试。这可能意味着解决错误的时间会比你一直坚持下去要晚，但是可能你就不用花这么长的时间来寻找错误。也就是说，可能是可以用等待来换取效率的。（同学们，这是一个很棒的早一些而不是更晚去做编程错误集的理由！）

### 6.2.4 在你找到错误的时候

当你找的一个程序中的错误时，那种想要马上开始编程调试修改的冲动是几乎不可抵抗的。然而最好是放慢一点。记住最终目标不是修改一个错误，而是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不是解释了所有观察到的症状，或者它是不是只是冰山一角。如果是后者，考虑修改这个错误要和其他变化相一致会更好。举一个例子来支持这一点，如果你发现错误的原因是不小心改变了一个列表，你可以局部的避开这个问题（可能通过把列表备份），或者可以用元组替代列表（因为元组是不变的），这样做说不定可以消除出现在程序其他地方的同样的错误。

在做更改之前，试着去想到修改带来的衍生问题。修改会不会破坏其他地方？它会不会使程序过于复杂？它是不是有机会整理好程序其他部分？

永远记得要确定你可以回到更改之前。没有什么比发现一长串的更改让你离开始时的目标更远并且无法回到开始的地方更让人沮丧的了。磁盘的空间一般都很充足，可以用它来储存老版本的程序。

最后，如果有很多无法解释的错误，你可能会考虑一个一个找到错误并且修改是不是正确的做法。思考是否有更好的组织程序的办法或者一些能更轻松正确使用的简单算法，这样可能会使你的状况变好。

1. 体会

这两章主要提示了在发现程序出错并且不知道如何解决时的一些技巧。里面的内容并没有技术性很强，更多的是我们平时编程的时候也知道但是遇到问题就忽略的方法。其中有几个建议对我帮助很大。

1. 去想错误不可能出在哪而不是反复的检查自以为出错的地方。

在编程的时候，尤其是编写里面包含比较复杂算法的程序的时候，出现错误我总想着一定出在复杂算法的地方，但很多时候浪费了很多时间才发现有可能只是最开始忘记定义或者少加括号之类的小错误。

1. 做修改之前考虑有没有其他影响。

我往往是发现一个错误就马不停蹄的修改，但很多时候修改完这个地方正确了，后面之前没错的地方又报错了。所以在修改之前好好看看修改点是不是对其他语句影响很重要。

1. 在错误很多的时候选择换一种算法

在学习了一种新的算法后，很想把它马上运用到程序中，但是有的时候，看似更高级的算法运用起来由于不熟练的关系可能会更不容易完全写对，所以，在错误过多修改没有头绪的时候选择换一种更为熟悉的算法可能更能节省时间、提高效率。