记录1：**重构**

养成不断地批判对待自己的代码的习惯。寻找任何重新进行组织、以改善其结构和正交性的机会。重构！！！

记录2：**原型**

原型制作是一种学习经验。其价值并不在于所产生的代码，而在于所学到的经验教训。

记录3：**曳光弹**

曳光弹，用户能够及早看到能工作的东西。

记录4：**估算**

估算，通过学习估算，并将此技能发展到你对事物的数量级有直觉的程度，你就能展现出一种魔法般的能力。

记录5：**工具**

花时间学习使用这些工具，有一天你将会惊奇地发现，你的手指在键盘上移动，操纵文本，却不用进行有意识的思考。工具将变成你的双手的延伸。

记录6：**调试**

调试就是解决问题，要据此发起进攻。发现了他人的bug之后，你可以花费时间和精力去指责让人厌恶的肇事者。在有些工作环境中，这是文化的一部分，并且可能是“疏通剂”。但是，**在技术竞技场上，你应该专注于修正问题，而不是发出指责**。

bug是你的过错还是别人的过错，并不是真的很有关系。它仍然是你的问题。

最容易欺骗的人是一个人自己。 不要恐慌。 要总是设法找出问题的根源，而不只是问题的特定表现。

不要假定，要证明。

调试不能改变被调试系统的行为。

记录7：**元程序设计**

再多的天才也无法胜过对细节的专注

记录8：**深思熟虑的编程 / 靠巧合编程**

不主动思考他们的代码的开发者是在靠巧合编程——代码也许能工作，但却没有特别的理由说明它们为何能工作。在“靠巧合编程”中，我们提倡要更积极地参与编码过程。

一开始就不知道它为什么能工作。有时很容易把“幸运的巧合”与有目的的计划混为一谈。

并非以明确的事实为基础的假定是所有项目的祸害。

不要做历史的奴隶。不要让已有的代码支配将来的代码。如果不再适用，所有的代码都可被替换。

重写、重做和重新架构代码合起来，称为重构（refactoring）

记录9：**等你准备好**

倾听反复出现的疑虑——等你准备好再开始

不要做形式方法的奴隶