

附录：规则汇编

我发现的每个真理都变成了一条规则，它们为我以后发现其他的东西服务。

——René Descartes，《方法论》

本书有的章节里列出了一些规则或准则，作为有关讨论的总结，我们把它收集到一起，以方便读者查阅。请注意，这些规则各有各的出处，那里解释了它们的意义和适用性。

风格

全局变量用具有描述意义的名字，局部变量用短名字。

保持一致性。

函数采用动作性的名字。

要准确。

以缩行形式显示程序结构。

使用表达式的自然形式。

利用括号排除歧义。

分解复杂的表达式。

要清晰。

当心副作用。

使用一致的缩行和加括号风格。

为了一致性，使用习惯用法。

用else-if 处理多路选择。

避免使用函数宏。

给宏的体和参数都加上括号。

给神秘的数起个名字。

把数定义为常量，不要定义为宏。

使用字符形式的常量，不要用整数。

利用语言去计算对象的大小。

不要大谈明显的东西。

给函数和全局数据加注释。

不要注释不好的代码，应该重写。

不要与代码矛盾。

澄清情况，不要添乱。

界面

隐蔽实现的细节。

选择一小组正交的基本操作。
不要在用户背后搞小动作。
在各处都用同样方式做同样的事。
释放资源与分配资源应该在同一层次进行。
在低层检查错误，在高层处理。
只把异常用在异常的情况。

排错

寻找熟悉的模式。
检查最近的改动。
不要两次犯同样错误。
现在排除，而不是以后。
取得堆栈轨迹。
键入前仔细读一读。
把你的代码解释给别人。
把错误弄成可以重现的。
分而治之。
研究错误的计数特性。
显示输出，使搜索局部化。
写自检测代码。
写记录文件。
画一个图。
使用工具。
保留记录。

测试

测试代码的边界情况。
测试前条件和后条件。
使用断言。
做防御性程序设计。
检查错误的返回值。
以递增方式做测试。
首先测试最简单的部分。
弄清所期望的输出。
检验那些应当保持的特征。
比较相互独立的实现。
度量测试的覆盖面。
自动回归测试。
建立自包容测试。

性能

- 自动计时测量。
- 使用轮廓程序。
- 集中注意热点。
- 画一个图。
- 使用更好的算法或数据结构。
- 让编译程序做优化。
- 调整代码。
- 不要优化无关紧要的东西。
- 收集公共表达式。
- 用低代价操作代替高代价操作。
- 铺开或者删除代码。
- 缓存频繁使用的值。
- 写专用的存储分配程序。
- 对输入输出做缓冲。
- 特殊情况特殊处理。
- 预先算出某些值。
- 使用近似值。
- 在某个低级语言里重写代码。
- 使用尽可能小的数据类型以节约存储。
- 不存储容易重算的东西。

可移植性

- 盯紧标准。
- 在主流中做程序设计。
- 警惕语言的麻烦特性。
- 用多个编译系统试验。
- 使用标准库。
- 只使用到处都能用的特征。
- 避免条件编译。
- 把系统依赖性局限到独立文件里。
- 把系统依赖性隐藏在界面后面。
- 用正文做数据交换。
- 数据交换时用固定的字节序。
- 如果改变规范就应该改变名字。
- 维护现存程序与数据的相容性。
- 不要假定是ASCII。
- 不要假定是英语。