检查表及总结 - 《代码大全》

作者：88250
原文链接：https://ld246.com/article/1382667814761
来源网站：链滴

为了更好的评估代码写的哪里有问题，我把《代码大全》里核心的部分 checklist 整理出来了，家可以大概过一遍，不一定每写完一个程序都要一条一条的去检查，但心里应该有这么一张检查表，写代码和 review 代码时自然而然的想起来。
设计

设计是否经过多次迭代，并最终决定了最好的一个？
是否同时使用自上而下和自下而上的方法来解决设计问题？
类与类之间的交互关系是否已经设计为最小化？
设计被划分为层次吗？
你对把这一程序分解成为子程序，包和类的方式感到满意吗？
程序是不是易于维护？
设计是否精简？设计出来的每一个部分都绝对必要吗？
整体而言，你的设计是否有助于最小化偶然性和本质性的复杂度吗？

类的设计

你是否把程序中的类都看做是抽象数据类型了？是否从这个角度评估它们的接口了？
类是否有一个中心目的？
类的命名是否恰当？其名字是否表达了其中新目的？
类的接口是否展现了一致的抽象？
类的接口是否能让人清楚明白的知道如何用它？
类的接口是否抽象，使你能不必顾虑他是如何实现其服务的？你能把类看做黑盒子吗？
类提供的服务是否足够完整，让其它类无需动用其内部数据？
是否已从类中去除无关信息？
是否考虑过把类进一步分解？
在修改类时是否维持了其接口的完整性？
是否把成员的可访问性降到最小？
是否避免暴露类的数据成员？
类是否避免对其使用者，包括其派生类会如何使用它做了假设？
类是否不依赖于其它类？它是松散耦合吗？
继承是否只用来建立一个 is a 关系？派生类是否遵循了 LSP 原则。
继承层次是否很浅？
类中是否只有大约七个或者更少的成员？
是否把类直接或者间接调用其他类的子程序的数量减到最少？
类是否在绝对必要时才与其他类写作？
是否在构造函数中初始化了所有的数据成员？

子程序

创建子程序的理由充分吗？
一个子程序中所有适合单独提出的部分是不是已经被提出到单独的子程序中了？
过程的名字是否用了强烈、清晰的动词加宾语的词组，函数的名字是否描述了其返回值？
子程序的名字是否描述它所作的全部事情？
子程序是否具有强烈的功能上的内聚性？
子程序之间是否有较松散的耦合？子程序与其它子程序之间的连接是否是最小的，明确的，可见，灵活的？
子程序的长度是否是由其功能和逻辑自然确定，而非遵循任何人为的编码标准？
子程序的参数表是否表现出一种具有整体性且一致的抽象？
子程序参数的排列顺序是否合理？是否与类似的子程序的参数排列相符？
接口假定是否在文档中说明？
子程序的参数是否没有超过 7 个？
是否用到了每一个输入参数，是否用到了每一个输出参数？
子程序是否避免了把输入参数用作工作变量？
如果子程序是一个函数，那么它是否在所有可能的情况下都能返回一个合法的值？

防御式编程

子程序是否保护自己免遭有害输入数据的破坏？
你用断言来说明编程假定吗？其中包括了前条件和后条件了吗？
断言是否只说明从来不应该发生的情况？
你是否在架构或者高层设计中规定了一组特定的错误处理技术？
你是否在架构或者高层设计中规定了是让错误处理更倾向于健壮性还是正确性？
代码中用到辅助调试的代码了吗？
在防御式编程时引入的代码量是否适宜，既不过多也不过少？
你在项目中定义了一套标准化的异常处理方案吗？
如果可能的话，是否在局部处理了错误而不是把它当成一个异常跑出去？
所有异常是否都与抛出他们的子程序在同一抽象层次上？
每个异常是否包含了关于异常发生的所有背景信息？
代码中是否没有空的 catch 语句？
检查有害输入数据的代码是否也检查了故意的缓冲区溢出，SQL 注入，HTML 注入，整数溢出及他恶意输入数据？
是否检查了所有的错误返回码？
是否捕获了所有的异常？
出错消息中是否避免出现有助于攻击者攻入系统所需的信息？

伪代码

是否检查过已满足所有的先决条件？
定义好这个类要解决的问题了吗？
高层次的设计是否清晰？能给这个类和其中的每个子程序起一个好的名字吗？
考虑过该如何测试这个类及其中的每个子程序吗？
关于效率的问题，你主要从稳定的接口和可读的实现这两个角度考虑吗？还是主要从满足资源和度的预期目标的角度考虑过呢？
从标准库函数和其它代码库中寻找过可用的子程序或者组件吗？
从参考书中查过有用的算法了吗？
是否用详尽的伪代码设计好每一个子程序？
你在脑海里检查过伪代码吗？这些伪代码容易理解吗？
关注过那些可能让你重返设计的警告信息了吗？
是否把伪代码正确的翻译成代码了？
你反复使用伪代码编程过程了吗？
在做出假定的时候有没有对它们加以说明？
已经删除了那些冗余的注释了吗？
你是否采取了几次迭代中最好的那个结果？还是在第一次迭代之后就停止了？
你完全理解你的代码了吗？这些代码是否容易理解？

变量

变量声明位置靠近变量第一次使用的位置吗？
尽可能在变量声明的同时初始化变量吗？
计数器和累加器经过适当初始化了吗？如果需要再一次使用，之前重新初始化了吗？
适当的重新初始化“需要重复执行的代码里的变量”了吗？
代码在通过编译器编译的时候是不是没有警告信息？你启用了所有可用的警告信息了吗？
如果语言允许隐式声明，你为由此可能引发的问题做好补偿措施了吗？
如果可能，所有变量都被定义为具有最小的作用域吗？
各变量的引用点都尽可能集中在一起吗？对同一个变量的两次相邻引用，或者变量的整个生命期这样做了吗？
控制结构符合数据类型吗？
所有声明的变量都用了吗？

变量都在合适的时间绑定了吗？也就是说你有意识的在晚期绑定所带来的灵活性和增加复杂度之做出平衡了吗？
每个变量都有且仅有一项用途吗？
每个变量的含义都很明确且没有隐含含义吗？

变量命名

名字完整并且准确的表达了变量所代表的含义吗？
名字足够长，可以让你无需苦苦思索吗？
如果有计算限定符，它被放在名字后面吗？
名字中用 Count 或者 index 来掉提 Num 了吗？
循环小标的名字有意义吗？
所有临时的变量都重新命名为更有意义的名字了吗？
当布尔变量为真时，变量能准确表达其含义吗？
枚举中的名字含有能够表示其类别的前缀或者后缀吗？
具名常量是根据它所代表的抽象实体儿不是它所代表的数字来命名的吗？
命名规则能够区分局部数据，类的数据和全局数据吗？
规则能够区分类型名，具名常量，枚举类型和变量名吗？
规则能够在编译器不强制检测只读参数的语言里表示出子程序的输入参数吗？
规则能尽可能地与语言的标准规则兼容吗？
名字为可读性而加以格式化了吗？
是否避免只为了省一个字符而缩写的情况？
所有单词的缩写方式都一致吗？
名字能够读出来吗？
避免使用容易被看错和读错的名字吗？
在缩写对照表里对端名字做出说明了吗？

基本数据类型

代码中避免使用神秘数值了吗？
代码考虑了除零错误了吗？
类型转换很明显吗？
如果一条语句中存在两个不同类型的变量，那么这条语句会像你期望的那样求值吗？
代码避免了混合类型比较吗？
使用整数除法表达式能按预期的那样工作吗？
整数表达式避免整数溢出问题了吗？
代码避免了对数量级相差具体大浮点数做加减运算了吗？
代码系统地阻止了舍入错误的发生吗？
代码避免对浮点数值做等量比较了吗？
代码避免使用神秘字符和字符串了吗？
使用字符串时避免了 off-bye-one 错误了吗？
程序用额外的布尔变量来说明条件判断了吗？
程序用额外的布尔变量来简化条件判断了吗？
程序用枚举类型而非具名常量来提高可读性和可修改行了吗？
当变量不能用 true 和 false 表示的时候，程序用枚举类型来取代布尔变量了吗？
针对枚举类型的才测试检测了非法数值了吗？
把枚举类型的第一项条目保留为“非法的”了吗？
具名常量使用一致吗？没有在某些位置使用具名常量又在其他位置使用文字量？
所有的数组下标都没有超出数组边界吗？
数组引用没有出现 off-by-one 的错误吗？
所有的多维数组的下标顺序都正确吗？
在嵌套循环里，把正确的变量用于数组下标来避免下标错乱吗？

不常见的数据类型

你使用结构体而不是使用单纯的变量来组织和操作相关的数据吗？
你考虑过创建一个类来代替使用结构体吗？
所有的变量是否都是局部或者是类范围的？除非绝对有必要才是全局的？
你对所有的全局变量都加以文档说明吗？
避免使用伪全局数据，即被四处传递且含有杂乱数据的的巨大对象吗？
用访问器子程序来取代全局数据了吗？
把访问其子程序和数据组织到类里了吗？
访问器子程序提供了一个在底层数据类型实现之上的抽象层吗？
所有相关的访问器子程序都位于同一抽象层吗？
把指针操作隔离在子程序里了吗？
指针引用合法吗？或者指针有可能成为悬空指针吗？
代码在使用指针之前检查它的有效性了吗？
在使用指针所指向的变量之前检查其有效性了吗？
指针用完后被设置为空了吗？
就可读性而言，代码用了所有需要使用的指针变量了吗？
链表中的指针是按正确的顺序加以释放的吗？
程序分配了一片保留的内存后备区域，以便在耗尽内存的时候能够优雅地退出吗？
是不是在没有其他方法可用的情况下最终才使用指针的？

组织直线型代码

代码使得语句之间的依赖关系变得明显吗？
子程序的名字使得依赖关系变得明显吗？
子程序的参数使得依赖关系变得明显吗？
如果依赖关系不明确，你是否用注释进行了说明？
你用“内务管理变量”来检查代码中关键位置的顺序依赖关系了吗？
代码容易按照自上而下的顺序阅读吗？
相关的语句组织在一起吗？
把相对独立的语句组放进各自的子程序里吗？

使用条件语句

代码的正常路径清晰吗？
if-then 测试对等量分支的处理方式正确吗？
使用了 else 字句并加以说明了吗？
else 字句用的对吗？
用对了 if 和 else 字句，即没有把它们用反吗？
需要执行的正常情况维护 if 而不是 else 字句里吗？
if-then-else-if 把复杂的判断封装到布尔函数里了吗？
if-then-else-if 先判断最常见的情况了吗？
if-then-else-if 判断包含所有的情况吗？
if-then-else-if 是最佳的实现吗？比 Case 语句还要好吗？
case 子句排序的有意义吗？
case 子句的每种情况操作简单吗？必要的时候调用了其它子程序了吗？
case 语句检测的是一个真实的变量，而不是为了滥用 case 语句而而刻意制造变量吗？
默认字句用的合法吗？
用默认字句来检测和报告意料之外的情况了吗？
在 c,c++ 和 java 里，每一个 case 的末尾有一个 break 吗？

循环

在合适的情况下用 while 循环取代 for 循环了吗？
循环是由内到外创建的吗？

是从循环的头部进入循环的吗？
初始化代码是直接位于循环前面吗？
循环是无限循环或者事件循环吗？阿德结构是否清晰？
避免使用像 for i = 1 to 9999 这样的代码吗？
如果这是一个 c++,c 或 java 中的 for 循环，那么把循环头留给循环控制代码了吗？
循环使用了{}及其等价物来括上循环体，以防止因修改不当而出错吗？
循环体内有内容吗？他是非空的吗？
把内务处理集中地放在循环开始或者循环结束处了吗？
循环像定义良好的子程序那样只执行一件操作吗？
循环短的足以一目了然吗？
循环的嵌套层次不多于 3 层吗？
把长循环的内容提取成单独的子程序吗？
如果循环很长，那么它非常清晰吗？
如果这是一个 for 循环，那么其中的代码有没有随意修改循环下标值？
是否把重要的循环下标值保存在另外的变量里，而不是在循环体外使用该循环下标？
循环下标是序数类型或者枚举类型，而不是浮点类型吗?
循环下标的名字有意义吗？
循环避免了下标串话问题吗？
循环是在所有可能的条件下都能终止吗？
如果建立了某种安全计数器标准，循环使用了安全计数器了吗？
循环的退出条件清晰吗？
如果使用了 break 或者 continue，那么它们用对了吗？
不常见的控制结构
每一个子程序都仅在有必要的时候才使用 return 吗？
使用 return 有助于增强可读性吗？
递归子程序中包含了停止递归的代码吗？
子程序用安全计数器来确保子程序能停下来吗？
递归只位于一个子程序里面吗？
子程序递归深度处于程序栈容量可以满足的限度内吗？
递归是实现子程序的最佳方法吗？它要好于简单的迭代吗？
是否在万不得已的时候才使用 goto？如果用了 goto，是否仅仅处于增强可读性和可维护性呢？<
li>
如果处于效率因素而使用的 goto，那么对这种效率上的提升做出衡量并且加以说明了吗？
一个子程序里最多只用了一个 goto 标号吗？
所有的 goto 都向前跳转，而不是向后跳转吗？
所有的 goto 标号都用到了吗？

表驱动法

你考虑过把表驱动法作为复杂逻辑的替代方案吗？
你考虑过把表驱动法作为复杂继承结构的替代方案吗？
你考虑过把表数据存储在外部并在运行期间读入，以便在不修改代码的情况下就可以改变这些数
吗？
如果无法用一种简单的数组索引去访问表，那么你把机酸访问键值的功能提取成单独的子程序，
不是在代码中重复地计算键值吗？

一般控制问题

表达式中用的是 true 和 false，而不是 1 和 0 吗？
布尔值和 true 以及 false 做比较是隐式进行的吗？
对数值做比较是显式进行的吗？
有没有通过增加新的布尔变量，使用布尔函数和决策表来简化表达式？
布尔表达式是用肯定形式表达的吗？
括号配对吗？

在需要括号来明确的地方都使用括号了吗
判断是按照数轴顺序编写了吗？
如果适当的话，java 中的判断用的是 a.equals(b)方式，而没有用 a==b 方式吗？
空语句表述得明显吗？
用重新判断部分条件，转换成 if-then-else 或者 case 语句、把嵌套代码提取成单独的子程序、成一种更面向对象的设计或者其他的改进方法来简化嵌套语句了吗？
如果一个子程序的决策点超过 10 个，那么能提出不重新设计的理由吗？

<hr>
转自：http://www.cnblogs.com/onlytiancai/archive/2010/05/30/1747556.html