

03 | 优秀程序员的六个关键特质

2019-01-09 范学雷 来自北京

《代码精进之路》



上一讲我们从“五道关卡”的角度讨论了如何写出优秀代码这个话题。对我们每个程序员个体来说，其实第一道“程序员”关卡最重要。没有优秀的程序员，就没有优秀的代码。那么，什么样的程序员才是优秀的程序员呢？

如果去问一个程序员，得到的答案有可能是，“写的一手好代码，做过几个大项目，设计、编程样样不在话下”。如果去问程序员的同学，得到的答案有可能是，“刚毕业就去了大厂，中秋节发的月饼羡慕死个人”。如果去问程序员的同事，得到的答案有可能是，“人挺好，干活挺快的”。如果去问 HR，得到的答案有可能是“省钱、出活”。


你看，这些答案都没什么毛病，各有各的道理。你的答案是怎样的呢？我自己工作中见过特别多优秀的程序员，从他们身上发现了不少共性的特质，我总结为以下六项，下面——给你介绍。

掌握一门编程语言

第一点很重要，优秀的程序员需要写的一手好代码，再简单来说，最起码需要能够熟练操控一门编程语言。

我们都会使用筷子，吃饭的时候，我们不需要有意识地控制着筷子的力度、开合和角度，也能准确地使用它。这个使用筷子的效率，是我们小时候长期练习的结果。每个人拿筷子的方法可能会有些差异，但是都不影响我们现在精准地、高效地使用筷子。

编写程序也是这样。熟练了之后，很多语法、语句在我们编写程序的时候，会下意识地就流露出来。如果我们设计一个线程类，下面的代码里，我觉得至少 `class` 和 `extends` 这两个关键字的使用是不需要大脑有意识地参与的。

 复制代码

```
1 public class MyThread extends Thread {  
2     ...  
3 }
```

如果把编程语言看成一套功法，比如降龙十八掌，这套功法练得越多，练得越纯熟，用起来越得心应手。武侠小说里，一套功法只有练了全套，才有最大的威力。对于编程语言，我们了解得越多，熟知的招式就越多，可选择范围就越大，我们就有更多的活动空间和解决问题的办法。

编程语言，基本上是相通的。掌握了第一门编程语言后，第二门语言学起来就快很多，第三门语言学起来更快。现在我们几乎都是多语言使用者，但一定要先精通一门语言，达到像用筷子那样的熟练程度。

解决现实的问题

掌握了一门编程语言，然后呢？代码是要解决具体的问题的，我们需要通过编程语言把解决问题的办法和思路表达出来。

要解决具体的问题，仅仅做到熟练使用编程语言是远远不够的，我们还需要更多工具。如果做前端，需要理解 HTML 和浏览器；如果做后端，需要掌握数据库和操作系统；如果做云计

算，需要掌握 Kubernetes 等等。就像学了分筋错骨手，还要学降龙十八掌；学了七十二路空明拳，还要学左右互搏。俗话说，艺多不压身，工具箱永远都不嫌满。

有了工具还不够，优秀的程序员还要深入理解问题，懂得问题的最核心价值。只有理解了问题，看到了解决问题的价值，我们才能够真正解决好问题，并且从中获得满满的成就感。**我们一定要记得，程序员的存在不是为了写代码，而是为了解决现实问题，实现现实价值。**

真实的作品，都带着我们对于现实问题的理解。而打磨一个这样的作品，需要缜密的逻辑、突破创新和贯彻执行。通过使用合适的工具，把简单的、一行一行的代码，耐心地粘合、打磨成优秀的作品。

如果说花样的工具是外家功夫，思维能力和行为能力可以算是内功。

优秀的程序员，是一个内外双修的程序员。如果一个程序员可以熟练使用工具，有清晰的解决问题思路，能明晰地传达产品价值，那么他编写代码就不存在什么巨大的困难了。

发现关键的问题

有了工具，遇到问题能解决掉，我们就可以做事情了。优秀的程序员还有一项好本领，就是发现关键的问题。**能够发现关键的问题，我觉得是一个好程序员和优秀程序员的分水岭。**

优秀的程序员，能够发现一门编程语言的缺陷，一个顺手工具的局限。所以，他知道该怎么选择最合适的工具，该怎么避免不必要的麻烦。

优秀的程序员，能够发现解决方案背后的妥协和风险。所以，他可以预设风险防范措施，设置软件的适用边界。

优秀的程序员，能够敏锐地观察到产品的关键问题，或者客户未被满足的需求。所以，他可以推动产品持续地进步和演化。

能够发现关键的问题，意味着我们可以从一个被动的做事的程序员，升级为一个主动找事情的程序员。

能够发现关键的问题，往往需要我们对一个领域有很深入的研究和深厚的积累，并且对新鲜事物保持充分的好奇心和求知欲。

掌握一门编程语言，解决现实的问题，能发现关键的问题，做到这三点，你就已经是一名优秀的程序员了。如果说优秀程序员有一个评价标准的话，这三条一定是硬性指标，接下来再介绍三条软性指标。

沉静的前行者

首先，优秀的程序员，一定是懂得妥协，懂得选择，一步一步把事情沉静地朝前推动的人。

如果真的较起真来，每一行代码，就像孔乙己的茴香豆，都有不止四样的写法。可是，最终的程序，只能选择唯一的一种。优秀的程序员都有在不断平衡、不断妥协中推动事物前行的能力和修为。

如果一个人说要一个完美的代码、完美的算法，完美的程序、完美的产品，我立刻就会非常紧张。完美是不存在的，所以我们才追求完美。对完美的过分追求，可能是一个代价高昂，收获甚小的行为。很多时候，我们不需要完美的东西。如果我只是想看看泰山山顶的日出，你就不要问我是爬上去的还是乘索道上去了。

对完美的理解，也是千差万别的。如果你的完美和我的完美发生碰撞，一定有一方需要妥协，我们才可以共同迈出下一步。

而且，完美也可能意味着不承认缺陷，不承认未知。这样，我们可能在心理上就不会对代码的未知风险做出充分的预判，留出足够的安全缓冲空间。

我们写的每一行代码，都可能存在问题。有时候，我发现别人的代码的问题；有时候，别人发现我的代码的问题。我们最后都会明白，要坦诚地面对别人的问题，也要坦然地面对自己的问题。在解决问题和帮助别人解决问题中，我们把一个产品变得越来越好，问题越来越少。

可以依赖的伙伴

其次，优秀的程序员是他人可以依赖的伙伴。

如果我们把软件开发看成一个循环的流水线，参与其中的每个人，都要接受来自上一级的输入内容，在当前环节和同事合作，创造面向下一级的输出内容。优秀的程序员，知道团队合作的重要性，是一个优秀的团队成员。他在团队中能够快速学习、成长，变得越来越优秀，也能够帮助其他团队成员变得越来越优秀。



优秀的程序员是一个领导型的人。他能够倾听，持续地获取他人的优秀想法，以及不同的意见。他能够表达，准确地传递自己的想法，恰当地陈述自己的意见。他是一个给予者，给别人尊重，给别人启发，给别人指导，给别人施展才华的空间。他是一个索取者，需要获得尊重，需要获得支持，需要持续学习，需要一个自主决策的空间。他能够应对压力，承担责任，积极主动，大部分时候保持克制和冷静，偶尔也会表达愤怒。他具有一定的影响力，以及良好的人

际关系，能够和各种类型的人相处，能够引发反对意见，但是又不损害人际关系。他知道什么时候可以妥协，什么时候应该坚持。

上面的这些，通常称为“软技能”。如果说，编程语言、花样工具、逻辑思维、解决问题这些“硬技能”可以决定我们的起点的话，影响力、人际关系这些“软技能”通常影响着我们可以到达的高度。因为，无论我们是加入他人的团队，或者组建自己的团队，我们只有在团队中才能变得越来越出色，做的事情越来越重要。所以，我们需要成为优秀的团队成员，接受影响，也影响他人。

时间管理者

最后我想和你分享的一点是，优秀的程序员是高效的时间管理者。

时间总是我们最大的障碍，优秀的程序员更是如此。没完没了的会议，没完没了的讨论，没完没了的学习，没完没了的需求，没完没了的 bug，时间拦住了我们的雄心壮志和大好宏图。

时间面前，人人平等，没有人一天的时间比别人多一秒。优秀的程序员会更好地管理时间，或者提高效率，或者用好时间。

你有没有听说过这样的故事？一家工厂的发动机坏了，请了很多人都没有修好。无奈，请了一位工程师，他听了听声音，在发动机上画了一道线，说：“打开，把线圈拆了”。果然，发动机就修好了。不管这个小故事是真的也好，假的也好，类似的事情在软件公司时时刻刻都在发生。有经验的程序员三分钟就能发现的问题，外行可能需要折腾好几天。持续地提高我们的硬技能和软技能，可以让我们做事情更快更好。

坚持把时间用在对的地方，用在价值更大的地方。事情总是做不完的。一般的工程师，都有一种打破砂锅问到底的精气神，这是好事。可是，这顺便带来了一点点的副作用，很多人有一点点小小的强迫症，很多事情，喜欢自己动手整个清楚明白。可是，事情又特别多，很多事情根本就顾不上。怎么办呢？

要做只有你才能做的事情。是的，有很多事情，只有你可以做，只有你做得最快最好。其他的同事也是一样的，有很多事情，只有他们能做，只有他们做得最快最好。选择最合适的人做最合适的事，这不仅是领导的工作分配，也可以是我们自己的协商选择。

事情做不完，就需要面临选择。**要坚持做需要做的事情。**不需要的、不紧急的、价值不大的，我们可以暂时搁置起来。一个人，能做的事情是有限的，能把最重要的事情最好，就已经很了不起了。

学会选择，是我们进阶道路上的一个必修课。

总结

最后，总结一下，优秀的程序员是什么样的？优秀的程序员可以熟练地使用必要的工具，发现和解决复杂的现实问题；优秀的程序员可以在一个团队里，高效沉静地把项目和团队一步一步地朝前推进。

现在，把我们今天讲的优秀程序员六大特质当作一套自测题，和自己对照一下，自己在哪些方面做得比较好？在哪些方面还需要继续精进呢？

欢迎你留言聊聊自己的经验，如果有什么问题或困惑，也可以提出来我们一起讨论。

如果今天的内容对你有帮助，也请你分享给身边的朋友，和他一起精进。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

精选留言 (26)



Sisyphus235

2019-01-09

程序语言是人和机器的沟通语言，和汉语英语等自然语言的工具属性是一致的。不同的是机器更容易被理解，确定性更强，我们也更可能精确理解二进制背后的逻辑，而与人沟通的不确定性更强，每个人都不同。前者是硬技能，后者是软技能。

沟通是工作最重要的部分，与机器沟通先存储大量不变的知识再结合实际任务处理，与人沟通要理解对方的特点和当下的心境再共同完成任务。

作者回复：这个角度很赞👍



👍 37



pyhhou

2019-01-09

感觉自己的硬实力还是不太够，然后时间分配上也不知如何是好，小公司没有一个很好的技术框架，也没有经验丰富的前辈加以指导，对公司整体的业务也不太清楚只清楚自己手头的任务，而且自己的任务的需求一直都在变化，太多的技术都觉得可能需要用上，但是学起来却不知从何下手，有些时候知道需求，但是从来没做过类似的东西，不知道每一步具体该怎么实现会遇到哪些问题，写代码之前的学习时间都比任何一个环节都要多，学完了还得思考怎么去用，毕业工作不到一年，自己一直处在这种迷茫的阶段，望老师指点

作者回复：我们都是这么过来的。先从手头的小问题开始，一点一点的把学到的用上。比如说，我们以后讲标识符的命名，学完了，你就看看手头的代码，命名是不是合理的，能不能改进。在后面，我们还会讲这么编写安全的代码，比如检查外部数据的有效性，也是可以立即用上的。每一篇，我们都有一个检查清单，你也可以对照对照自己的代码，看看自己什么地方做的好，傲娇一下；什么地方，还可以更好，改进一点。

不用着急，每次进步一点点，很快就变成优秀的人了。最怕的是，停下来，没好奇心了，没怀疑精神了。



👍 23



beiler

2019-02-22

您好，请问会讲一下包的名字划分解决方案和模块归类的方法吗？

作者回复：这个问题没有讲到。你可以参考JDK的包的命名和模块归类。包名可以使用倒置的域名加模块名，比如：java.net, java.lang, com.sun.net, sun.security.ssl。模块的归类大原则是按照功能划分，工具类的，放在util下，比如java.util。需要注意的是，公开接口和内部实现，尽量使用不同的包，这样便于管理。比如，javax.net.ssl是公开接口，sun.security.ssl是内部实现。



👍 6



堵车

2019-01-12

这篇文章讲到优秀程序员的对工作的选择。我记得前两年，我的工作就是不断地写业务，改代码，满足领导的奇思妙想。称之为小步快跑，多版本试验，快速迭代。两年时间里，没有沉淀任何东西。所以这样的环境不适合成为优秀的程序员吧？

作者回复: 环境很重要, 我觉得更重要的, 还是自己要有改进环境的意识。小公司的存在很多很多的问题, 可是这些问题, 也正是我们的机会。把问题解决掉, 就是自己脱颖而出的机会。

有时候, 我们常说一句话, 就是, 每一次危机都不应该浪费。什么意思呢, 就是危机虽然让人头疼, 但是危机来的时候, 就是我们打怪升级的时候, 就是我们学习应对危机的时候。解决小公司的问题也是这样的。

如果刚毕业, 我们就从解决一些小问题开始, 比如怎么把代码写的更漂亮, 然后琢磨琢磨让其他的人代码也漂亮, 然后再琢磨琢磨让整个团队的代码更漂亮, 然后再琢磨琢磨让整个公司的代码都漂亮。然后, 再解决更大的问题。



👍 5



秦凯

2019-01-10

“解决现实的问题”的观点特别认同, 个人也认为程序员应该具备的是用计算机解决现实世界问题的思维, 将现实世界的问题转化为计算机可解决的具体方案。而将方案落地的过程应该是, 使用计算机、编程语言、数据结构与算法、协议等底层抽象工具封装成一个具体的用户可使用的工具的过程, 这个具体的工具本质上是对用户问题解决方案的一个封装。

作者回复: 赞!

要盯着用户有效需求。我们后面还有一小节讨论需求问题。



👍 4



轻歌赋

2019-01-11

才参加实习, 公司比较小, 没有质量管理, 是的, 没有。完全是代码能跑就行。来这已经3个月了, 虽然项目代码都能写, 而且时不时还能使用一些新学到的jdk的新工具包写代码, 但是仍然感觉对自己写的代码的稳定性不敢保证, 虽然很少错, 可是还是很担心。想问问老师, 如何在身边没有其他评审的情况下, 提供一些自己检查代码逻辑bug的方法呢? 而且对业务分析不熟悉, 经常会出现建表少了某个字段的情况, 请问老师有没有什么相对系统化的设计方面的知识可以学习呢?

作者回复: 每一篇结束, 我们都会总结一个检查清单, 你可以使用这三个清单检查代码。有了三个检查清单作为基础, 你可以扩展这个清单, 遇到好的经验就加上去, 过上一段时间, 你就能积累大量的

好习惯。

另外，认真检查编译器的警告和IDE的警告，这些警告，一般都是验证了的编码经验，你消除了警告，就学到了新东西。

编码规范，也有IDE插件，比如阿里巴巴Java编码指南的插件。找找看有没有你喜欢的插件。

专栏的小伙伴们，能不能帮着推荐些设计的书籍？



👍 4



小峰

2019-01-12

突然发现我师傅就是一个这样的人。

作者回复: 👍厉害了，你的师傅！



👍 2



Dream

2019-01-10

优秀的程序员，还需要一个优秀的上司，我习惯每次在编码前把思路理顺，发现潜在的问题和风险，其实正在编码可能占用不到工期三分之一的时间，但是目前领导一看前期根本没在编码，就觉得比较闲，拼命的压缩工期和增加任务，有点疲倦了

作者回复: 是的，团队的氛围很重要。😓过程而不是目标导向的上司太操心。



👍 2



王智

2019-01-09

自我感觉没有一个满足,哭.

第一点,掌握一门编程语言,对于Java,基础功学过了,当时记住了,过一段时间就忘了,难受.

第二点,我赞同,我感觉程序员就是现实和网络的桥梁,就是把现实的问题映射到网络上进行解决.

第三点,身为一名刚刚工作的程序员,没有这个胆量和能力.

第四点,不能坦然的面对自己的问题,这点一定要改

第五点,只要在我会的知识层面,还算是可以依赖

第六点,哭泣,定的计划都很难完成,一旦有了任务,计划全乱了套了.
还得继续磨练,加油加油

作者回复: 我们都在修炼的路上😊, 先有意识和目标。加油💪



👍 2



Neal

2019-03-10

要做自己才能做的事情真是醍醐灌顶, 项目中总是前后端兼顾, 导致自己后台进度一直很慢, 很累, 现在我终于放下前端, 专心于后端的迁移工作上, 渐渐发现, 前端没了我也没什么要紧的。

作者回复: 我自己花了很多年, 才懂的。



👍 2



小文

2019-02-14

文中说解决现实的问题, 如果一直沉浸于解决现实的问题, 总是解决手里这一摊也不行, 哪天干着干着突然发现自己岁数大了, 可是本事没长进, 还是会那些个技能解决着自己的一摊, 万一领导给你个别人任务, 或者跳槽了接触了其他的一些任务不会点技能是真不行啊, 程序员还是不能停止学习啊, 唉.....

作者回复: 解决现实的问题, 不能沉浸于只解决领导要求的问题。程序员是一个要终身学习的职业。



👍 1



刘滨涛(陆逾)

2019-01-17

优秀的程序员能在团队里正能量影响他人, 协同他人一起创造价值。不仅仅解决业务需求, 也能去创造或者开发一些对公众有益的东西。能力越大, 责任也就越大。

作者回复: 责任越大, 意味着大家越需要我们。 这就是竞争力的一部分, 对吧。



👍 2

小田



2019-01-12

优秀程序员的特质小结

硬指标

- 优秀的编程技能
- 解决问题能力
- 认知和发现问题的能力

软指标

- 为高效推进目标进程，作出合适的权衡和妥协
- 团队协作能力
- 时间管理能力



Aliliin

2019-01-10

深刻体会到沟通的重要性。

需求沟通不明确，提问都表达不清楚，，总是会把时间浪费在无用功上。

作者回复: 哈哈，看来是受过需求折磨的。有个说法我比较喜欢，信息传递会损耗，多几道环节，就变样了。软件开发，要设计好沟通过程，减少损耗啊。



Kai

2019-01-10

有一本书 程序员的软技能，写得很好。其中就说到写blog一是个非常好地锻炼沟通，表达，提升知名度和整理思维的过程

作者回复: 是的，有时间多写写blog。



ifelse

2022-07-14

如果说，编程语言、花样工具、逻辑思维、解决问题这些“硬技能”可以决定我们的起点的话，影响力、人际关系这些“软技能”通常影响着我们可以到达的高度。--记下来





进化圈
2021-11-12

这几个点总结的很到位，是优秀程序员特质的目标~
加油💪少年



SummerWindy-_-

2021-05-14

持续的提高硬技能和软技能。



Mr.yu

2021-05-06

优秀的程序员必须是能不断进化的。



底层小学生

2021-02-13

- 1.c语言只是基本掌握，但打算精通python
- 2.对于算法掌握很不全面，还不能良好解决现实问题
- 3.对于关键问题的发现，还需要掌握更多基础知识。
- 4.不要总想着这里缺那里也缺，本身起步阶段就需要不断地原始积累，所以关掉朋友圈，步步为营吧
- 5.先快速代码规范，再学数据结构郝斌王争，算法训练营，刷题

作者回复：赞，不学学Java？

