
Broadview
www.broadview.com.cn
Code Craft
——The Practice of Writing Excellent Code
编程匠艺
——编 写 卓 越 的 代 码
[美] Pete Goodliffe 著
韩江 陈玉 译
THE PRACTICE OF WRITING
EXCELLENT CODE
PETE
GOODLIFFE
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn


Code Craft
編程匠艺
THE PRACTICE OF WRITING
EXCELLENT CODE

Code Craft

——The Practice of Writing Excellent Code

编程匠艺

——编写卓越的代码

[美] Pete Goodliffe 著
韩江 陈玉 译

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

如果你可以编写出合格的代码，但是想更进一步、创作出组织良好而且易于理解的代码，并希望成为一名真正的编程专家或提高现有的职业技能，那么《编程匠艺——编写卓越的代码》都会为你给出答案。本书的内容遍及编程的各个要素，如代码风格、变量命名、错误处理和安全性等。此外，本书还对一些更广泛的编程问题进行了探讨，如有效的团队合作、开发过程和文档编写，等等。本书各章的末尾均提供一些思考问题，这些问题回顾了各章中的一些关键概念，可以促使你像专家一样思考，从而使本书成为那些渴望作为团队的一分子，职业并高效地编程的新手们的一本绝佳的参考书。

图书在版编目（CIP）数据

编程匠艺：编写卓越的代码 /（美）古德利弗（Goodliffe, P.）著；韩江，陈玉译. —北京：电子工业出版社，2008.9
书名原文：Code Craft：The Practice of Writing Excellent Code
ISBN 978-7-121-06980-2

I. 编… II. ①古… ②韩… ③陈… III. 程序设计 IV. TP311

中国版本图书馆 CIP 数据核字（2008）第 093400 号

责任编辑：顾慧芳
印　　刷：北京市天竺颖华印刷厂
装　　订：三河市金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×980　1/16　印张：38.75　字数：840 千字
印　　次：2008 年 9 月第 1 次印刷
定　　价：79.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。
质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。
服务热线：（010）88258888。

本书在正式发行之前所获得的赞誉

“为了掌握某种技艺，不仅仅需要诀窍和工具，还需要态度和技能。对于那些真正用心的程序员来说，这就是他们可以从《编程匠艺——编写卓越的代码》一书中学到的东西。在许多代码猴子形象（译注：本书采用的卡通形象）的帮助下，这本书鼓励读者对他们所从事工作进行反省和思考。”

——Kevlin Henney，独立咨询师

“容易理解，非常吸引人，甚至常使人忍俊不禁。这本书是从多年的实际工作中提炼出的智慧，饱含着软件开发世界的辛酸和甜蜜……真希望我刚开始当程序员的时候能够拥有这本书。”

——Steve Love，资深开发人员

“《编程匠艺——编写卓越的代码》是一座知识的金矿，每位职业软件开发人员都应该掌握这些知识。”

——Tim Penhey，C VU 编辑

“良好的判断来源于经验。而经验——是的，却来源于你所做出的糟糕的判断！这本书是一个机会，可以让你从别人艰难得来的经验中学习，从而少走许多弯路。”

——Lois Goldthwaite，C++和 POSIX BSI 标准委员会的召集人

“这就是那种你应该给新员工们看的书。它讲的都是事实，很容易理解，并且涵盖的主题范围极广，这些主题都是新入行的程序员们应该搞明白的。”

——Jon Jagger，软件培训师、设计师、咨询师、导师、程序员

“这是一本独特的、极为实用的指南，可以指导你成为现代软件工厂中的一名职业程序员。”

——Andrew Burrows，软件开发人员

“Pete 拥有一种罕见的能力。他不仅能指出最优秀的职业软件开发人员们所使用的技术（他们自己往往都没有意识到），还能用一种清晰和简明的方式来描述它们。”

——Greg Law，Undo Ltd 的 CEO

“我真希望在我职业生涯的初期接受指导的时候就能读到这本书。至少，现在我可以用它来指导跟我学习的程序员。”

——Dr. Andrew Bennett，高级工程师、工程学学士、博士、MIET、MIEEE

“那些有幸聆听过 Pete Goodliffe 的演讲的人，都能马上识别出他那轻松幽默、清晰明了地阐述问题的方式。在教学的环境下，这本书可以直接用来作为教材，不论是新手还是有经验的从业人员，都能从中学到东西，并得到提高。”

——Robert D. Schofield，硕士、MIET FOUNDER、Scientific Software Services Ltd.

“Pete 要阐明的不仅仅是怎样编写代码，而且还有怎样更好地编写代码，让程序员们能够在工作中使用正确的工具和技巧。《编程匠艺——编写卓越的代码》这本书广泛地探讨了编程的各个方面，书中所提出的原则是所有重视工作的开发人员都应该熟知的。”

——Chris Reed，软件开发人员

“Pete Goodliffe 对促进软件开发职业化的贡献，在业界是有目共睹的。凭借他那让人信赖的知识，以及风趣和详实的写作风格，不管是对于新手还是对于有经验的开发人员来说，Pete 都可以称得上是一名出色的导师。”

——Rob Voisey，Akai Digital Ltd.的技术总监

“我最喜欢书中的代码猴子形象。”

——Alice Goodliffe，四岁半

推 荐 序
——打造开发者的核心竞争力

信息产业几乎是开放程度最高的产业，其发展之快，让身处其中的我们既感到兴奋和激动，也时常困惑和惶恐。信息技术的变化太快、太激烈，一切都在迅速革新当中，作为技术人员，我们今天熟悉并且赖以安身立命的东西，很快就会变成故纸堆里的古董。人生总是需要不断积累才能越走越高的，如果找不到一项可以积累提升、而且被市场所承认的核心竞争力，事业的大厦是建不起来的。于是一个严峻的问题就摆在每一个技术人员的面前——什么才是我们的核心竞争力？

二十年前，人们认为软件开发者的核心竞争力应该体现在他的聪明才智和扎实的基本功上，要精通算法、硬件、编译、计算机体系结构，要有天马行空般的想像力和创造力，能够单枪匹马单打独斗。至于什么团队协作、编码规范、单元测试、最佳实践，都是对天才的不必要的羁绊。

大约到了 20 世纪 90 年代中期，风水变了。随着软件的规模越来越大，以及当代操作系统、面向对象技术、软件开发工具、程序库和框架的普及，“平台”和“工具”变得越来越重要了。大多数时候，开发者不需要自己从头实现所有功能，他所依赖的平台已经准备好了现成的组件供使用，有的时候甚至应用程序中最繁琐、最复杂的部分已经由框架或程序库实现好了。因此，人们普遍认为，一个开发者的核心竞争力很大程度上表现为对平台与工具的理解和掌握，并依托这种理解创造性地构筑产品，或者更确切地说，一种创造性地组装产品的能力。你还在为刚刚琢磨出来的一个高效算法而得意吗？旁边那位借助程序库里提供的组件，已经快速地完成了整个功能模块；你发明了一个新的视频格式以及对应的 codec？很了不起！不过另一个初出茅庐、连快速傅里叶变换算法都写不出来的团队已经基于 sourceforge 上的开源组件做出产品来，并且播得满互联网都是了。当然，黑客们仍然有着不可取代的价值，总是能找到自己的位置；但是对于那些强调团队作战、快速行动的企业来说，一个熟练掌握 MFC、Delphi、J2EE、ASP.NET 或者 LAMP 的开发者可能更实用，能更有效地为企业带来价值。因此，这样的程序员便一时成为企业的宠儿，众人眼中的高手。

然而不到十年下来，问题又出现了。流行的平台和工具如走马灯般你方唱罢我登场：昨天还在为领悟了 MFC、Delphi 而沾沾自喜，今天就发现应用主流已经是 Web 了；刚刚啃完艰深的 EJB2，抬眼一看却发现它已经被 Spring 的拥趸们批倒批臭了；上个月还是冲在敏捷 Java 领域的改革派，这个月就被一群嘴上无毛的 RoR 粉丝给划

到改革的对立面去了；ASP.NET 还没学踏实呢，微软又准备好 ASP.NET MVC、ASP.NET AJAX、Silverlight 等等一大堆新玩意让你啃了。这样下去，什么时候是个头？把自己的核心竞争力建立在这些转瞬即逝的昙花上，难道不是把有限的生命投入到无限的瞎折腾之中吗？难道只有钻到一间舒舒服服的大公司里，到了三十多岁就寻求所谓的“转型”，顺着一条十分确凿的“职场路线”攀或是混，最后在公司没有倒闭或者自己没有被“战略裁员”的幸运之下头顶玻璃天花板光荣退休，才是中国程序员的归宿？什么才是程序员可以长期积累，不断提高，不但足以安身立命，而且能够实现梦想、成就事业的核心竞争力呢？ 回答好这个问题，对于今天的开发者来说，可能比掌握和精通某项具体技术意义重大得多。

在我看来，当代程序员的核心竞争力至少应该体现在这么几点上：有扎实的基本功，活跃的想像力与创造力，快速的学习能力，具备行业和领域知识，以及专业的软件工艺能力。而在这其中，专业软件技能是最基本、也是最重要的一项。

什么是专业软件技能呢？就是正确地开发软件的能力，更具体地说，是通过一系列有组织的、有原则、流程化、可检验、可重复的实践行为，协作式开发高质量程序的能力。对于一个程序员来说，这是你的看家老本，对于一个软件团队来说，这是你们的立足之基。算法不会，可以查资料慢慢掌握；不理解行业，可以边做边学，逐渐深入；缺乏创新，可以站在巨人肩膀上耐心摸索；甚至基本功不足，也可以自我弥补，可是如果没有做软件的专业态度和实践技能，没有制作合格软件的工艺水平，连一段高质量的程序都写不出来，试问你还剩下什么？

经过近三十年的时间，人们最终认识到，在规模化团队协作的情况下，决定软件产品质量的不再是个人的聪明才智，也不是靠什么神仙技术，而是团队的工艺实践。是否在一开始就形成了开发计划？是否对这个计划进行了必要的确认、维护和跟踪？必要的规范文档是否撰写了？是否形成了合理的架构？是否恰当地选择了开发工具和编程语言？是否建构了适于团队渐进协作的良好的工具和工作平台？是否一开始就形成了有力的缺陷核查、控制和跟踪策略并始终严格地执行？是否制定了连续一致的编码标准，并且通过诸如代码走查等加以保证？是否有完整的测试制度？是否具有明确的性能优化和软件安全性保障过程？是否在整个生命周期贯彻了严格的版本管理、配置管理、发布管理和软件维护退役管理措施？这些实实在在的问题，是需要耐心与细心地用具体实践细节来回答的。当一个团队对于这些问题都给出了明确而一致的回答并且用行动来执行的时候，他们就是一个专业的、具有核心竞争力的团队。而当一个个体开发者能够对这些问题具备正确的观念，并且通过施加自己的影响力促进团队向正确的方向前进的时候，他就是一个具有核心竞争力的开发者。一个具有核心竞争力的团队和开发者，是可以不断进步的，是具备把握

机遇的能力的；一旦时机合适，他们就完全有可能实现更大的目标。

十多年以前国内外软件界对工艺的问题并不重视。大部分人要么执迷于技术本身，指望某一天一个面向某某的技术能够一劳永逸的解决软件开发中的所有问题，要么就是把问题大而化之为“软件工程”，企图以指令性的方式，在宏观的层面上用管理取代工艺。在这两个方向上，程序员要么被视为可以充分放纵的孤胆英雄，要么被视为伟大编程技术最终出现之前不得不存在的过渡品，或者管理指令的机械的执行体，“人”的维度消失了。这种对于人和工艺细节的忽视也体现在技术著作方面。软件工程、面向对象、编程技巧和产品手册之类的著作汗牛充栋，而认真谈到软件工艺的书屈指可数。

直到 20 世纪 90 年代中期，随着一些软件产品的规模越来越大，微软率先认识到工艺问题的重要性，于是出版了诸如《代码大全》、《编写清晰的代码》等一系列探讨这一问题的著作。直到 20 世纪 90 年代末期，当整个工业界从面向对象和软件工程的幻影泡沫中走出来之后，才开始认真全面地审视软件工艺的问题，而且通过敏捷运动、把软件工艺的重要性和基本实践提到了一个令人瞩目的位置上。事实上，敏捷运动可以认为是软件工艺的复兴运动。此外，随着《代码大全 2》、《软件工艺》、《代码阅读》、《程序员修炼之道》等经典作品的出版，在技术图书领域也陆续出现了一批专门探讨软件工艺的著作。这本《编程匠艺》也是这个领域中的一本佳作。

本书是一部全面讨论软件构造工艺实践的著作，从软件开发的计划到架构设计，从编码风格规范到软件缺陷的检测与管理，从程序员工具箱的配备到团队协作精神的塑造，这本书都给予了翔实、风趣而具有启发性的讨论。这些讨论，既有原则性、理论性一面，也有技术性的具体建议，对于团队领导者、高级开发者和每一个希望快速进步的程序员具有明确的指导意义。如果读者认同软件工艺的重要性，那么可以说这本书是帮助读者建构自己核心竞争力的一本难得的作品。特别值得一提的是，这本书中文版的翻译流畅自然，在很多地方都体现出译者的认真态度和翻译功力。对于一本翻译自英文的技术著作来说，这无疑是一个大大的加分。

当然，一本书的覆盖面和功效毕竟是有限的，核心竞争力的确立和建构归根到底是一个艰苦实践的过程，不同性格的人也一定有着不同的目标和方式。但是我相信，对于有心人来说，只要我们不断地探索和实践，都会获得自己的核心竞争力，做一个有准备的人，争取和等待机会的垂青，最终实现自己的人生目标。

读此书有感而发，借题发挥，是为评论。

《程序员》杂志技术主编

孟 岩

2008 年 7 月于北京

译 者 序

作为从事软件开发的程序员，你肯定遇到过这样的情况：自认为完美的代码，在项目快要结束的时候，却总是会发现还有好多内容需要修改。更有甚者，由于人员的变动，那些他们遗留下来的“老代码”，作为时间留给程序员与项目组的最大遗产，却可能会成为项目组的灾难。

除了受制于人类自身的缺陷之外，还有由于组织而带来的问题，如客户需求不断变更、必须在有限的时间和预算之内完成项目，来自内部所谓“项目管理”的种种压力，等等。天哪，这些问题我们绝大部分人都赶上了。

列宁曾在监狱中写下了《怎么办？》，指导了俄国的十月革命。而在软件业，从一代宗师 Frederick P. Brooks 的《人月神话》开始，就在找“怎么办”这个“银弹”了。然而，“狼来了”在多次被喊出来后，已经很少有人相信了。我们必须承认，这些都是根本层面的问题，目前还不能得到解决。但是，本书的作者 Pete Goodliffe 认为，至少我们可以采取一些方式，减少一些开发上的痛苦。因为，除了开发，人生还有许多更为美好的事物在等着我们。我们这次也可以高喊“银弹来了”。没有最好，只有更好，谁知道这次不是真的呢？

著名国画大师齐白石在年轻的时候，曾经做过木匠。据说有一次他和师傅去给地主干活，在路上迎面走来另外一对木匠师徒。齐先生的师傅说，赶紧给别人让路。师徒俩站在路边，老师恭敬地目送那两人渐渐走远。齐白石不解，问师傅：同是木匠，你我师徒为什么要给他们让路。老师傅回头说：为什么？别人是做细活的，我们是做粗活的。

Pete Goodliffe 在业界的年头快要超过好多人的年龄了，此君曾经涉猎多个领域、不同的编程语言以及多种架构，并且曾经在采用不相同流程的公司里从事开发。在本书中，他把多年压箱底的一些观念想法和技巧告诉了大家，这些都是时间与智慧的结合，相信无论是开发人员、项目经理甚至测试人员，都可以从中发现阿里巴巴开启金库的钥匙。

那么本书有什么特色呢？对于想了解内容的普通读者来说，本书至少有以下特点：

1. **贴近实际** 《编程匠艺——编写卓越的代码》是本书的书名，但也是作者的用心所在。人生有三个境界，最后一个就是“看山是山，看水是水”。这是废话吗？当然不是，作者对此给出了最好的解答。作为程序员，我们最喜欢争论不同工具、平台、方法之间的优劣。而作者却通过多年经验，力图告诉我们应该如何提高质量，

并成为一名优秀的程序员。这些方法就像点石成金的手指，它们是方法论，而不是针对具体的工具或者平台的说教。我们现在所缺的，恰恰是这些能使自己更进一阶的手段，而不是那些特殊的技术细节。

2．**内容丰富翔实** 很少有一本书能涵盖如此多的领域，并且还如此扎实。作为一名程序员，我们可能永远无法达到完美。而需要处于一种持续不断地提高的状态，总会有更多的东西需要学习。那么下一步应该做什么呢？这里就有答案。

3．**可作为“秘要心法”** 本书不仅适合入门者，也适合需要提高的开发人员，以及那些想管理好所谓代码猴子的项目经理们。与《项目经理案头手册》一样，这本书也将成为每人的案头手册或者枕边书，可以作为应急或者提升的手段。如果以后碰到了问题，可以随时参阅相关的章节。

4．**心态决定一切** 这句话对吗？有了良好心态，不一定行，如果没有，肯定不行。我们常常羡慕于老外以四五十岁的年纪仍然能继续从事编程，为什么我们不行呢？可能不同的读者都会找到属于自己的答案！Pete Goodliffe 具有宽阔的视野，扎实的基础，广泛的爱好，带有一种程序员应该具有的高雅和恬淡。这正是我们这个浮躁的时代中积极探索的一代程序员所不具备的。

最后禁不住要抱怨一下，作者 **Pete Goodliffe** 以他丰富的阅历和爱好，给译者带来了不小的麻烦，比如出于它对于音乐的爱好，所有章节的标题都来自英国的歌曲名称。为了理解上的直观，我们在翻译的过程中采取的是“信达雅”中的“雅”，以保证国内读者能很快切入主题。本书每章开始和行文的过程中，作者都引用了历史上或者现在社会中一些名人的名言，这给翻译增加了不少的难度，但是由于贴切精辟，这些名言也可称之为点睛之笔。尤为值得高兴的是，此君对我中华文化竟然也有一定的造诣，孔夫子和老子的哲理名言竟然多次出现，而且能够贴切地表达出这些圣人的思想对软件开发有哪些启示，这非常不简单，难为了作者，也着实难为了译者。从外国作者的笔下，让我们着实体会到了自己国家的文化源远流长。这从一个侧面也体现出东海西海，千圣一心。

此书给了我们一个快速成功进阶的好范例。我觉得它更像一个程序员的入门或者修行心法。从此入门，我们可以少走很多弯路。同时，我们也要争取像佛经中“般若波罗密”所讲的那样：大智慧到彼岸，最后连佛法也像渡河的筏子一样，成佛后立即丢弃。我更希望的是，看过此书的读者们，最后能够拍案而起，大声说：我可以了。

译 者

2007-12-2 于北京

前　言

有许多事情是智者不想知道的。

——拉尔夫·瓦尔多·爱默生（Ralph Waldo Emerson）

这本书是从战壕中走出来的。好吧，它实际上来自软件工厂的深处，但有些时候这并没有太大的区别。本书写给那些“很在意”自己事业的程序员。如果你不在意，那么请现在就合上这本书，然后原封不动地把它放回到书架上吧。

对我而言书中写了什么？

编程就是你的激情。这很令人遗憾，但却千真万确。作为痴迷于技术的人，你几乎连做梦都在编程。现在，你身处现实世界的中心，深入这个行业，做着你从来没有想像过的事情：在玩电脑的同时还领着薪水。事实上，你还愿意向别人付钱来得到这个特权哩。

但这里却是一个奇怪的地方，根本不是你所期望的那样。面对不现实的最后期限和糟糕的管理（如果他们称之为管理的话），以及不断改变的需求和一团糟的遗留代码，你感到很惊讶，不禁怀疑这是真实的世界吗？整个世界都在阻止你编写出你一直以来所梦想的代码。这就是现实，欢迎来到软件工厂中的生活。你站在一场创造艺术精品和科学天才的艰苦战争的最前沿。祝你好运！

这就是《编程匠艺——编写卓越的代码》要达到的目的。这本书的内容也许从未有人向你传授过：如何“在现实世界中”正确地编程。《编程匠艺——编写卓越的代码》填补了教科书中所缺少的内容。的确，这本书讨论了优秀代码的技术细节和难点。但除此之外它还包括更多的内容：如何以正确的方式编写正确的代码。

这是什么意思呢？在现实世界中，编写优秀的代码有多重的含义：

- 编写在技术上优雅的代码
- 编写可维护的代码，让其他人也可以看得懂
- 理解并改写其他人所编写的杂乱的代码
- 与其他程序员良好地并肩工作

要成为一个编码高手，你需要所有这些能力（甚至更多）。你必须理解代码那神秘的生命：当你输入代码之后，会发生什么？你必须拥有审美的能力：区分优美的代码和丑陋的代码。你还必须拥有在实践中运用理论的头脑：认真思索并解决以下问题，什么时候使用简化的操作是合理的，什么时候开始着手代码设计，以及什么时候停止和继续（实用的“提前退出”原则）等。本书将帮助你实现这些目标。你将学习到如何在软件工厂中求得生存，如何观察战场并了解敌人，如何制定战术以避开敌人的陷阱，以及如何在种种困难中开发真正出色的程序。

软件开发是一个有趣的职业。这个职业在迅猛地发展，充满了瞬间即逝的流行元素与变幻莫测的风尚、致富的计划以及新理念的传播者。它并不成熟。我不是宣称我有什么神奇的答案，但我的确有一些来自经验的、有用的建议与大家分享。本书中没有象牙塔中的理论，而仅仅是现实世界的经验和一些好的习惯。

一旦消化了这本书的内容，你将不仅仅会成为一名更好的程序员，而且你将成为软件工厂中一位更优秀的常住居民，一名真正的代码战士，你将学到编程匠艺的技艺。如果这听起来并不令人激动，那么也许你应该考虑在军队里谋职了。

做得更好

那么，是什么将“优秀”的程序员和“糟糕”的程序员区分开的呢？更重要的是，是什么将“杰出”的程序员与仅仅是“够格”的程序员区分开的呢？其秘密并不仅仅在于他们的技术能力——我曾经见过很多对语言标准了然于心、能够写出非常紧凑和让人欣赏的 C++程序的程序员，他们才华横溢，但是他们写出来的代码却非常糟糕。我也曾经见过更多谦逊的程序员，他们坚持编写非常朴素的代码，但是他们所编写的程序却非常优雅和缜密。

真正的区别是什么？好的程序设计来自于你的态度。而好的态度在于你了解职业化的方法，以及对编写最好的软件的不懈追求，而不管软件工厂的压力有多大。态度就像一面透镜，我们通过它来看世界。这面透镜为我们的工作和行为增添了色彩。优秀的代码需要由艺术大师精心编写，而不是由懒散的程序员随意地粗制滥造。通向优秀代码的道路是由良好的意愿铺就成的。要成为杰出的程序员，我们必须学

会从良好的意愿起步，培养积极的看法，并使这种健康的态度发扬光大。

在本书中，我们将看到如何做到这一点。书中包含了大量的主题，小至实用的代码编写问题，大到组织架构性问题。在所有这些主题中，我都着重强调了什么是正确的态度和方法。

态度——前进的角度

面对软件开发的世界，随着调查和分类的深入，我越来越相信使杰出的程序员脱颖而出的是态度。字典中对“态度”（attitude）的定义是这样的：

态度

1. 心态，看法：心理或感情的状态；性情。
2. 飞机姿势，姿态：飞机轴相对于某一参照直线或平面（如地平线）所处的位置。

第一个定义并没有什么令人意外之处，那么第二个呢？实际上，第二个定义比第一个更有揭示意义。

我们想像有三条贯穿飞机的轴线：一条从一侧的机翼到另一侧的机翼，一条从机头到机尾，还有一条与前两条轴线垂直相交于其交点处。飞行员靠这三条轴线来定位飞机，它们定义了飞机前行路线的角度。这称为飞机的飞行姿态（attitude）。如果飞机的飞行姿态不正确，那么只要有很少的外力施加到飞机上，飞机就会极大地偏离它的目的地。飞行员必须时刻注视飞机的飞行姿态，特别是在起飞和着落等关键时刻。

虽然这听起来像一个乏味的励志片，但它确实是我们的软件开发工作的一个贴切比喻。飞机的“态度”决定了它前进的角度，而我们的态度则决定了我们通往成功编码之路的方向。程序员的技术能力有多强并不重要，如果他或她的能力没有受到健康态度的支持，那么工作仍将是一件痛苦的差事。

错误的态度可以造成或中断一个软件项目，所以保持正确的前进角度来进行编程是至关重要的。你的态度要么阻碍，要么促进你的个人成长。要想成为更优秀的程序员，我们需要保证我们有正确的态度。

谁应该阅读这本书

显而易见，那些希望提高他们代码质量的人应该读这本书。我们都渴望成为更好的程序员；如果你没有这种渴望，那么这本书就不适合你。你也许是一名职业的程序员，很可能已入行多年。你也许是一名高年级的学生，熟知编程的概念，但是不太肯定应该如何将这些概念付诸于实践。如果你正在接受指导，或正在指导某个新手，读这本书也会大有裨益。

你必须已经具有编程经验。本书并不是要教你如何编程，而是告诉你如何更好地

编程。当我尽力避免语言的歧义和教条时，我需要举出一些代码示例。这些示例大部分是使用 C、C++或 Java 语言编写的，因为这些语言是当今流行的语言体系。读这些例子并不需要你是某种语言的专家，所以即使你不是一流的 C++程序员，也不要惶恐。

我假设你作为本书的读者，正在或者将要在软件工厂的压力下编写代码。这通常意味着你受雇于一个商业性的开发组织，但你有可能在开发一个混乱的开放源码开发项目，或成为一个受雇为第三方提供软件的枪手（项目承包方）。

本书包含哪些内容

本书的重点在于程序员的态度，但它绝不是一本心理学教科书。我们将深入探讨许多主题，包括：

- 源代码的样式
- 防御性的编码技巧
- 如何有效地调试程序
- 良好的团队合作技巧
- 管理你的源代码

快速地浏览一下目录，你就可以了解本书所包含的内容。选择这些主题的理由是什么呢？我曾有好多年从事培训新程序员的工作，在此期间这些主题反反复复地被提出。我也曾在软件工厂中任职多年，看到过很多一遍又一遍地出现的问题，而这些问题都是与以上主题相关的。

如果你能够征服这些编程的羁绊，那你将从一个学习编程的新手成长为一位真正的编码高手。

本书的组织形式

我已经尽力将这本书写得易于阅读。传统的格言这样说，你应该从头开始并努力工作到最后。忘掉这句话吧。**你完全可以拿起这本书，打开你感兴趣的章节，并从那里读起**。每个章节都是独立的，其中有很多有用的交叉引用，可以使你了解各章之间是如何结合在一起的。当然，如果你喜欢遵循传统，从头读起也不失是一个好的选择。

各个章节都以相似的结构编写，你不会发现任何异样。每一章都可以分为下面几个部分：

本章主题

在每章的一开头，我会列出该章的要点。你可以从这里了解这章的内容梗概。浏览一下这些主题，你就会知道我们将涉及哪些方面的内容。

章节主体

这里都是会让你感到本书物有所值的引人入胜的内容。

在章节主体中会时而出现一些“关键理念”。这些关键理念强调了重要的技巧、问题和态度，所以对这些内容要格外关注。其格式是这样的：

> **关键概念** 重要内容。注意！

总结

在每章的结尾，这一小节将总结该章所讨论的主题。它为整章内容提供了一个概括性的视图。如果你真的时间有限，也可以只阅读每章的关键理念和这些总结部分。千万别告诉别人这是我说的。

然后，我将对比优秀程序员和糟糕程序员的行为方式，来总结你应该力求采取的重要态度。如果你有勇气，也可以根据这些例子来评价一下自己，但愿事实不会让你太伤心！

另请参见

这个列表会将你带入相关的章节，并解释它们是如何与当前的主题相关的。

思考

最后，我将列出一些需要思考的问题。这些问题并不游离于全书之外——它们是各章不可缺少的一部分。它们并不是要你回顾刚才所阅读的主题，相反，这些问题的目的是使你进行思考，并联想到该章之外的内容。这些问题分为两组：

- **深入思考** 这些问题将深入地研究各章的主题，并提出一些重要的论点。
- **了解自己** 这些问题将探查你及你的软件开发团队的工作习惯和编码成熟度。

不要略过这些问题！即使你很懒惰，以至于不想坐下来认真思考每个问题的答案（相信我，思考这些问题的答案会使你受益无穷的），至少也应该读一读这些问题并顺便做些思考。

本书的最后一部分包含了对这些问题的答案和讨论。这并不是一套直接的答案集——这些问题中很少可以直接用“是”或“否”来回答。在你思考并得到答案之后，不妨与我的答案做个比较。我的许多“答案”都包含了一些各章中未曾涉及的额外信息。

章节——详细说明

本书的每一章都将讨论一个单独的主题，涉及现代软件开发的一个具体问题领域。这些问题是人们编写糟糕的代码或糟糕地编写代码的常见原因。各章所描述的正确方法和态度，将使站在第一线的你的生命力更旺盛。

本书共分为六篇；每一篇的目录页列出了各篇所包含的章节，并简要描述了每个章节所包含的内容。这些篇将由内而外地展开讨论。我们将从应当编写“何种”代码开始入手，最终着眼于我们应当“如何”编写这种代码。

我们的讨论从代码表面开始，主要是从宏观的角度分析如何编写源代码。我特意将这部分放在全书的最前面，因为剪裁代码是程序员们所真正关心的。

第Ⅰ篇：代码表面

本篇描述了开发源代码的具体细节。我们将研究防御性编程的技巧，以及如何编排代码的格式和版式。然后，我们将继续探讨如何为代码命名和做记录。编写注释和处理错误的技巧也包含在本篇中。

第Ⅱ篇：代码的神秘生命

接下来，我们将讨论编写代码的过程，即如何创建代码和处理代码。我们将了解构造代码的工具、测试方法、调试技术、编译可执行程序的正确过程以及优化等内容。最后，我们将思考如何写出安全的程序。

第Ⅲ篇：代码的形成过程

然后，我们将从更广泛的角度讨论构造源代码的问题。我们将讨论代码设计方案的开发、软件的体系结构以及源代码是如何随时间而成长（或老化）的等问题。

下一步，我们将进入“宏观”层面，抬起头看看身边发生的事——软件工厂中的生活。如果不是一个开发小组的成员，我们是不可能写出大规模软件的，接下来的三篇包含了如何在团队中获益的技巧和方法。

第Ⅳ篇：“一群”程序员？

程序员不可能生活在真空中。（这需要有特殊的呼吸装备。）在本篇中我们将进入更广阔的世界，看一看好的开发习惯，以及如何将这些习惯融合到职业程序员的日常工作中。本篇还包含了良好的个人和团队编程技巧，以及版本维护系统的使用方法。

第Ⅴ篇：开发过程的组成部分

在本篇中，我们将描述软件开发过程的一些规则：书写规范、执行代码审查以及时间预算的魔法。

第Ⅵ篇：从高处鸟瞰

最后一篇将从高处向下审视开发过程，研究软件开发方法论，以及不同的编程规则。