

共创高质量软件开发之道

高质量 程序设计指南 — C++/C 语言

(第二版)

林锐 韩永泉
飞思科技产品研发中心

编著
审校

软件工程研究院



电子工业出版社

PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>

高质量 程序设计指南 ——C++/C语言 (第二版)

林 锐
韩永泉

飞思科技产品研发中心

编著

审校

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

如何进行高质量的程序设计是软件行业的薄弱环节,大部分企业只能依靠大量地测试和改错提高软件产品的质量,为此付出了高昂的代价。因此,如何让程序员熟练地掌握编程技术和编程规范,在开发过程中内建高质量,是IT企业面临的主要挑战之一。

本书作者以轻松幽默的笔调向读者论述了高质量软件开发方法与C++/C编程规范,令人耳目一新。它是作者多年从事软件开发工作的经验总结。书中大量的“提示、建议、规则”具有很好的实用价值。

本书第一版的电子文档曾经在Internet上广泛流传,被不少国内IT企业的软件开发人员采用。本书第二版比第一版的内容增加了一倍,并且论述更加细致和深入,可以认为是全新之作。

本书的附录C《大学十年》是作者在网上发表的一个短篇传记,文中所描述的充满激情的学习和生活的态度,感染了大批莘莘学子。

本书的主要读者对象是IT企业的程序员和项目经理,以及大专院校软件相关专业的本科生和研究生。本书亦可作为大学计算机系和软件学院的教学参考书。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有,侵权必究。

图书在版编目(CIP)数据

高质量程序设计指南——C++/C语言(第二版)/林锐,韩永泉编著. —2版. —北京:电子工业出版社, 2003.6

(软件工程研究院)

ISBN 7-5053-8628-X

I.高... II.①林...②韩... III.C语言—程序设计 IV.TP312

中国版本图书馆CIP数据核字(2003)第023156号

责任编辑:王树伟 杨 鹂

印 刷:北京市增富印刷有限责任公司

出版发行:电子工业出版社 <http://www.phei.com.cn>

北京海淀区万寿路173信箱 邮编:100036

经 销:各地新华书店

开 本:787×980 1/16 印张:30.25 字数:677.6千字

版 次:2003年6月第2版 2003年6月第1次印刷

印 数:8000册 定价:39.00元

凡购买电子工业出版社的图书,如有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系。联系电话:(010) 68279077

出版说明



软件工程是一类工程，而工程是将理论和知识应用于实践的科学。就软件工程而言，它借鉴了传统工程的原则和方法，以求高效地开发高质量软件，其中应用了计算机科学、数学和管理科学。计算机科学和数学用于构造模型与算法，工程科学用于制定规范、设计泛型、评估成本及确定权衡，管理科学用于计划、资源、质量和成本的管理。

根据调查，中国的现状几乎和美国 10 多年前的情况一样，软件开发过程没有明确规定，文档不完整，也不规范，软件项目的成功往往归功于软件开发组的一些杰出个人或小组的努力。这种依赖于个别人员的成功并不能为全组织的软件生产率和质量的提高奠定有效的基础。只有通过建立全组织的过程改善，采用严格的软件工程方法和管理，并且坚持不懈地付诸实践，才能取得全组织的软件过程能力的不断提高。而产生国内这种现状很重要的原因之一，就是国内缺乏软件工程方面优秀的图书。很多国外的软件工程方面的书进入中国以后，在应用上与中国实际国情还有一定的差距。鉴于上述现状，电子工业出版社飞思科技产品研发中心经过充分的市场调查，数次与国内外软件工程专家交流、座谈，精心推出了“软件工程研究院”系列图书。

本书由电子工业出版社飞思科技产品研发中心组织国内外软件工程专家精心编写，是一套陪伴中国软件人才成长的高质量图书。“软件工程研究院”具有如下特色：

- 内容清晰、体系完整，包括软件工程方法学理论、团队与项目管理、UML、软件架构、软件过程、软件测试、需求工程、软件质量、软件文档、面向对象技术等方面。
- 风格严谨，而表达方式不拘一格。
- 理论结合实际，为企业提供最具有实效的指导。
- 权威经典。引进国外专著，或由国内学术界和企业界最知名的专家学者编写。
- 丛书读者对象主要是软件学院的高年级学生、企业的中高级软件人才。

本丛书首次开门向读者献上的一份厚礼就是美籍华人阎宏的《Java 与模式》一书。该书作者阎宏博士，凭借深厚的自然科学和中国传统哲学功底，以及多年来对模式理论的研究，通过这本书向读者讲解软件设计模式的理论，它的简要历史，以及它与中国道家文化的渊源。这是一本实用的教材和手册，同时又是一本散发着思想火花的作品。

随着“软件工程研究院”品牌在国内的影响，越来越多的优秀作者汇聚到飞思旗下。后期我们还陆续推出了由国内软件工程专家林锐博士多年从事软件工程专业经验的结晶——《CMMI 3 级软件过程改进方法与规范》、《高质量 Java 程序设计》等高品质图书，共同奏响属于中国软件工程的优美乐章。同时，我们希望这乐曲能为中国软件工程的发展描绘出一幅欣欣向荣的蓝图。

我们的努力需要您的支持。为了使我们的脚步更坚实、使我们的队伍永葆活力和创造力，我们期待着您能为我们的前进贡献出您的意见和建议。同时，我们也在等待着您的加入。

我们的联系方式：

咨询电话：(010) 68134545 68131648

答疑邮件：support@fecit.com.cn

网 址：<http://www.fecit.com.cn> <http://www.fecit.net>

答疑网址：<http://www.fecit.com.cn> 的“问题解答”专区

通用网址：计算机图书、FECIT、飞思教育、飞思科技、飞思

电子工业出版社计算机研发部
飞思科技产品研发中心

前言

软件质量是被许多程序员挂在嘴上而不是放在心上的东西！

除了完全外行和真正的编程高手外，初读本书，你最先的感受将是惊慌：

“哇！我以前编造的 C++/C 程序怎么会有那么多的毛病？”

别难过，作者只不过比你早几年、多几次惊慌而已。

请花几个小时认真阅读这本经书，你将会受益匪浅，这是前面 N-1 个读者的建议。

编程老手与高手的误区

自从计算机问世以来，程序设计就成了令人羡慕的职业，程序员在受人宠爱之后容易发展成为毛病特多却常能自我臭美的群体。

如今在 Internet 上流传的“真正”的程序员据说是这样的：

(1) 真正的程序员没有进度表，只有讨好领导的马屁精才有进度表，真正的程序员会让领导提心吊胆。

(2) 真正的程序员不写使用说明书，用户应当自己去猜想程序的功能。

(3) 真正的程序员几乎不写代码的注释，如果注释很难写，它理所当然也很难读。

(4) 真正的程序员不画流程图，原始人和文盲才会干这事。

(5) 真正的程序员不看参考手册，新手和胆小鬼才会看。

(6) 真正的程序员不写文档也不需要文档，只有看不懂程序的笨蛋才用文档。

(7) 真正的程序员认为自己比用户更明白用户需要什么。

(8) 真正的程序员不接受团队开发的理念，除非他自己是头头。

(9) 真正的程序员的程序不会在第一次就正确运行，但是他们愿意守着机器进行若干个 30 小时的调试改错。

(10) 真正的程序员不会在上午 9:00 到下午 5:00 之间工作，如果你看到他在上午 9:00 工作，这表明他从昨晚一直干到现在。

.....

具备上述特征越多，越显得水平高，资格老。所以别奇怪，程序员的很多缺点竟然可以被当做优点来欣赏，就像在武侠小说中，那些独来独往、不受约束且带点邪气的高手最令人崇拜。我曾经也这样信奉，并且希望自己成为那样的“真正”的程序员，结果没有得到好下场。

我从读大学到博士毕业十年来一直勤奋好学，累计编写了数十万行 C++/C 代码。有这样的苦劳和疲劳，我应该称得上是编程老手了吧？

我开发的软件都与科研相关（集成电路 CAD 和 3D 图形学领域），动辄数万行程序，技术复杂，难度颇高。这些软件频频获奖，有一个软件获得首届中国大学生电脑大赛软件展示一等奖。在 1995 年开发的一套图形软件库到 2000 年还有人买。罗列出这些“业绩”，可以说明我算得上是编程高手了吧？

可惜这种个人感觉不等于事实。

读博期间我曾用一年时间开发了一个近 10 万行 C++ 代码的 3D 图形软件产品，我内心得意表面谦虚地向一位真正的软件高手请教。他虽然从未涉足过 3D 图形领域，却在几十分钟内指出该软件多处重大设计错误，让人感觉那套软件是用纸糊的华丽衣服，扯一下掉一块，戳一下破个洞。我目瞪口呆地意识到这套软件毫无实用价值，一年的心血白花了，并且害死了自己的软件公司。

人的顿悟通常发生在最心痛的时刻，在沮丧和心痛之后，我做了深刻反省，“面壁”半年，重新温习软件设计的基础知识。补修“内功”之后，又觉得腰板硬了起来。博士毕业前半年，我曾到微软中国研究院找工作，接受微软公司一位资深软件工程师的面试。他让我写函数 strcpy 的代码。

太容易了吧？

错！

这么一个小不点的函数，他从三个方面考查：

- (1) 编程风格。
- (2) 出错处理。
- (3) 算法复杂度分析（用于提高性能）。

在大学里从来没有人如此严格地考查过我的程序。我花了半个小时，修改了数次，他还不尽满意，让我回家好好琢磨。我精神抖擞地进“考场”，大汗淋漓地出“考场”。这“高手”当得也太窝囊了。我又好好地反省了一次。

我把反省后的心得体会写成文章放在网上，引起了不少软件开发人员的共鸣。我因此有幸和国内大型 IT 企业如华为、上海贝尔、中兴等公司的同志们广

泛交流。大家认为提高质量与生产率是软件工程要解决的核心问题。高质量程序设计是非常重要的环节，毕竟软件是靠编程来实现的。

我们心目中的老手们和高手们能否编写出高质量的程序来？

不见得都能！

就我的经历与阅历来看，国内大学的计算机教育压根就没有灌输高质量程序设计的观念，教师们和学生们也很少自觉关心软件的质量。勤奋好学的程序员长期在低质量的程序堆中滚爬，吃尽苦头之后才有一些心得体会，长进极慢，我就是一例。

现在国内 IT 企业拥有学士、硕士、博士文凭的软件开发人员比比皆是，但他们在接受大学教育时就“先天不足”，岂能一到企业就突然实现质的飞跃。试问有多少软件开发人员对正确性、健壮性、可靠性、性能、易用性、清晰性、可扩展性、安全性、兼容性、可移植性等质量属性了如指掌，并且能在实践中运用自如？“高质量”可不是干活小心点就能实现的！

我们有充分的理由疑虑：

(1) 编程老手可能会长期用隐含错误的方式编程，习惯成自然后，被人指出发现毛病时都不愿相信那是真的！

(2) 编程高手可以在某一领域写出极有水平的代码，但未必能从全局把握软件质量的方方面面。

事实证明如此。我到上海贝尔工作后，陆续面试或测试过近百名“新”、“老”程序员的编程技能，合格率低于 50%。很少有人能够写出完全符合质量要求的 if 语句，很多程序员对指针、内存管理一知半解，……

领导们不敢相信这是真的。我做过现场试验：有一次部门新进 14 名硕士生，在开欢迎会之前对他们进行“C++/C 编程技能”摸底考试。我问大家试题难不难？所有的人都回答不难。结果没有一个人及格，有半数人得零分。

竞争对手如华为、中兴、大唐等公司的朋友们也做过试验，也是类似结果。真的不是我“心狠手辣”或者要求过高（甚至变态），而是很多软件开发人员对自己的要求不够高。要知道这些大公司的员工素质在国内 IT 企业中是比较前列的，倘若他们的编程质量都如此差的话，我们怎么敢期望中小公司拿出高质量的软件呢？连程序都编不好，还谈什么振兴民族软件产业，岂不胡扯。

多年来，我在软件开发过程中的苦头吃得实在太多了，现在总算被折磨清醒了。我打算定义编程老手和编程高手，请您别见笑。

定义 1：能长期稳定地编写出高质量程序的程序员称为编程老手。

定义 2: 能长期稳定地编写出高难度、高质量程序的程序员称为编程高手。
根据上述定义, 马上得到第一推论: 我既不是高手也算不上是老手。

在写此书前, 我阅读了不少程序设计方面的英文著作, 越看越羞惭。因为发现自己在编程基本技能方面都未能全面掌握, 顶多算是二流水平, 还好意思谈什么老手和高手。希望和我一样在国内土生土长的程序员朋友们能够做到:

- (1) 知错就改。
- (2) 经常温故而知新。
- (3) 坚持学习, 天天向上。

本书出版经历

2001 上半年, 我负责给上海贝尔有限公司网络应用事业部的开发人员们培训软件工程和 CMM。那时公司尚缺乏内容详细的编程规范, 我就写了一本《高质量 C++/C 编程指南》当做公司内部培训教材。

当时那本书稿还不到 100 页, 由于我自己在编程方面没有专长, 不想着出版, 于是就放到网上了事。书稿的电子文档在网上转悠了一年, 很多热心的同行们修正了书稿中的不少错误, 并出乎意料地给予了一些称赞。

到 2002 年 6 月份, 我在上海贝尔有限公司从事软件工程研究与实践已经将近两年了, 与合作者们取得了一些成果。我们掂量后觉得写几本书不会误人子弟, 于是我就试探性地给几家名望较好的出版社发了 E-mail, 表达了出版意愿。

几个小时后, 电子工业出版社计算机研发部(即飞思)总经理郭晶当即决定出版, 并且纳入了他们策划的“软件工程研究院”系列。我也不再犹豫, 赶紧完善书稿, 2 个月后《高质量程序设计指南——C++/C 语言》就出版了。紧接着半年之内, “软件工程研究院”系列的另外 3 本著作问世, 出版社的效率之高、质量之好让签约作者们刮目相看。

《高质量程序设计指南——C++/C 语言》第一版上市后, 一度成为畅销书。网上评论甚多, 褒贬参半。我们分析了读者的批评和建议, 总结了本书第一版的主要不足:

由于第一版原本是企业的培训教材, 初衷是为了帮助程序员提高程序质量, 假设读者已经熟悉 C++/C 语法, 所以内容薄而精练、前后章节不连贯, 看起来更像专题讲座。出版社的宣传工作做得很好, 本书吸引了很多 C++初学者和高级程序员。由于书中不讲解入门知识, 导致很多初学者看不下去。有一些大学生听了我的讲座后, 为表敬意特地买书让我签名, 翻阅之后就塞进书架当做纪念品了。对于那些高级程序员而言, 本书的大部分内容他们早已经熟悉, 好不容易看到几

处精彩的章节，却翻了几页就没有了，真是不过瘾啊。

我的研究专长是软件工程和企业管理，而非程序设计。在 C++/C 编程方面自己仅仅是名老工匠而已，我的确没有时间没有能力写出让初学者和高级程序员都喜欢的 C++/C 书籍。这本书炒作得过火，让我骑虎难下了。从 2002 年 11 月起，我就开始考虑与真正的 C++/C 高手合作来写本书的第二版。

恰好上海大唐移动通信设备有限公司的韩永泉正在为公司写 C++/C 培训教材，他也是本书第一版的读者。韩永泉提出了很有价值的建议和批评，并把他写的培训教材发给我看，真是自投罗网啊！

韩永泉是在西安电子科技大学计算机系读的本科和硕士，内功扎实。我和他碰头交谈了 2 个小时，就把第二版的完善大事托付给他了。两个月后，他把第二版的书稿交给我审阅。第二版的内容比第一版多了一倍，其广度符合我的设想，其深度完全出乎我的意料。为了阐述清楚 C++/C 程序之中的许多“为什么”，本书给出了大量的“提示、建议、规则”，并从编译器实现的角度论述原理。这种深度非一般教科书能比，我用了一个月时间才审阅并且学习完毕，删除了几十页过分深奥的内容（免得让我自己看昏倒）。我相信第二版可以让大多数高级程序员看过瘾了。

网上有一些愤世嫉俗者认为计算机领域的每个分支都已经有了世界名著，不具有世界顶尖水平的中国人再写类似书籍都是欺世盗名的行为，这种极度自负和极度自卑的心态导致他们专爱骂国内作者。如果中国作者的书籍中的技术错误被他们逮住，经过放大、推理、演绎之后基本上就能断定作者是卑鄙之徒，于是砖头就拍过来了（简称“拍砖”）。拍砖者们遥相呼应，很快就能拍出江湖豪情，被拍的作者就成了倒霉蛋。有位好心的读者怕我经受不起，特意发给我一本拍砖大法——《拍砖十二流》以增强内功。

网上自由谩骂既是网络价值的体现又是民主的体现，这是物质文明和精神文明发展到一定境界的产物。《高质量程序设计指南——C++/C 语言（第二版）》即将出版，作者忐忑不安地等待第二轮“拍砖”。

本书导读

首先请做附录 A 的 C++/C 试题（请先不要看答案），考查自己的编程质量究竟如何。然后参照附录 B 的答案严格打分。

✧ 如果你只得了几十分，请不要声张免得让人知道，也不要太难过。编程质量差往往是由于不良习惯造成的，与人的智力、能力没有多大关系，还是有药可救的。成绩越差，可以进步的空间就越大，中国不就是在落

后中赶超发达资本主义国家吗？只要你能下决心改掉不良的编程习惯，第二次考试就能及格了。

- ◇ 如果你考及格了，表明你的技术基础不错，希望你能虚心学习、不断进步。
- ◇ 如果你考出 85 分以上的好成绩，你有义务和资格为你所在的团队做“C++/C 编程”培训。
- ◇ 如果你在没有任何提示的情况下考了满分，希望你能收作者做你的徒弟。

作者得知不少 IT 企业在招聘程序员时就拿本书试题来考试，由于本书第一版已经公布了试题和答案，这一招显然不灵了，请勿再用。

编程考试结束后，请阅读本书的正文。第 1 章至第 3 章是比较有趣的综述性文章，讲了不少做人做事的道理，基本上不涉及编程技术，读者可以快速浏览。从第 4 章起直至第 19 章，全部论述 C++/C 编程技术，这些章节的目的不是介绍 C++/C 语法，而是阐述如何编写高质量的 C++/C 程序。书中的【提示】、【建议】和【规则】均是重点，为了讲清楚“为什么”，几乎每章都有一半的篇幅是从编译器实现的角度阐述原理，这些内容非常适合于高级程序员，但是对于 C++/C 初学者而言可能太难了，请酌情取舍。

本书的所有示例程序均在 Visual C++ 6.0 环境下编写与测试。印刷出来的部分示例程序并没有严格遵循本书推荐的编程风格，显得自相矛盾。但这是为了节省书籍的版面，不得已而为之。

第 1 章论述“高质量软件开发之道”，本章的“道”指的是道理和道路。编程只是软件开发中的一个环节而已，如果程序编不好，产生的软件估计也不是个好东西。但是即使编程非常棒，却未必就能产生高质量的软件。本章在漫谈软件质量之际，阐述了“提高质量、提高生产率并且降低成本”的软件开发方法。

第 2 章论述“做好程序员”。我绝对没有过分推崇程序员这个职业，也不规劝你一辈子只当一名好程序员。如果你想在软件行业干下去，我建议你首先成为一名好程序员，然后成为技术专家、管理者或者老板，等等，这样的道路比较踏实。我自己这么说也是这么做的。

第 3 章介绍“编程语言发展简史”。近半个世纪来，一些了不起的企业及其英雄好汉们创造了不少优秀的编程语言，使得大批程序员拥有了好饭碗和好梦想。回忆历史是为了现在，趁着计算机工业还没有被人类淘汰，我们得赶紧做点有益于社会的事情。

第 4 章介绍“程序的基本概念”。不少程序员虽然能够使用 C++/C 语言来编

写程序，却无法准确地阐述什么是程序设计语言、语言实现、开发环境、程序的工作原理及其关系，以致走入了语言学习的误区。本章将把这些纠缠不清的概念和它们之间的关系阐述清楚，为学习后面的章节奠定基础。

第 5 章讲解“C++/C 程序设计的入门知识”，篇幅甚长。入门并不意味着简单，实际上本章内容出乎意料地深入。本书附录 A 的一些试题取自本章，的确曾经考败过一大片程序员。作者并非想给读者（尤其是初级程序员）当头一棒，而是想强调“良好的开端是成功的一半”。

第 6 章讲解“C++/C 常量”。常用的常量有字面常量、符号常量、契约性常量、布尔常量和枚举常量等，本章将逐一揭示每一种常量的本质及其用法。

第 7 章讲解“C++/C 函数设计基础”。函数是 C++/C 程序的基本功能单元，其重要性不言而喻。仅使函数的功能正确是不够的，因为函数设计的细微缺点很容易导致该函数被错用。本章重点论述函数的接口设计和内部实现的一些规则，再介绍一些比较高级的内容如存储类型、递归函数等，最后介绍如何使用断言以及 const 来提高程序的健壮性。

第 8 章讲解“C++/C 指针、数组和字符串”。在 C++/C 程序中几乎不可能不用到指针、数组和字符串，但是许多人不能清晰地说出它们之间的区别和联系，老是处于一种糊里糊涂的状态之中，因此滋生了许多错误。本章通过分析指针、数组、字符串、引用的内存映像，澄清了这些概念。

第 9 章讲解“C++/C 高级数据类型”。C++对 C 的结构、联合、枚举这些高级数据类型进行了必要的改造和增强，本章比较分析了异同点，总结了使用要点，对于那些从 C 语言向 C++语言过渡的程序员有较好的参考价值。

第 10 章讲解“C++/C 编译预处理”。预编译伪指令不是 C++/C 语句，但是它们可以出现在程序中的任何地方，只要展开后符合语法规则并且是有效的。本章讲述 C++/C 程序中经常用到的一些预编译伪指令，如文件包含、宏定义、条件编译及一些不常使用的预编译伪指令和符号常量，使用它们对提高代码的可移植性和清晰性很有好处。

第 11 章讲解“C++/C 的文件结构和程序版式”。虽然文件结构和程序版式并不影响功能，也无多少技术含量，但是能够反映出开发者的职业化程度。我们当然希望自己的程序看上去是专业级的而非业余级的。

第 12 章讲解“C++/C 应用程序命名规则”。据考察，没有一种命名规则可以适合所有程序员的口味，程序设计教科书一般都不指定命名规则。命名规则对软件产品而言并不是“成败攸关”的事，我们不要花太多精力试图发明世界上最好的命名规则，而应当制定一种令大多数项目成员满意的命名规则，并在项目中

贯彻实施。本章介绍了共性规则和简单的 Windows 应用程序命名规则。

第 13 章为“C++面向对象程序设计方法概述”。本章的目的不是阐述面向对象的理论，而是用浅显的示例来解说面向对象程序设计的一些重要概念，如封装、继承、组合、虚函数、抽象基类、动态决议、多态和对象模型等。**本章对后面各章具有指导意义。**

第 14 章讲解“类的构造、析构和赋值函数”。构造函数、析构函数与赋值函数是每个类最基本的函数，简称“Big-Three”。由于它们太普通，所以容易使人麻痹大意。其实这些貌似简单的函数就像没有顶盖的下水道那样危险。对于那些没有吃够苦头的 C++ 程序员，如果他说编写构造函数、析构函数与赋值函数很容易，可以不用动脑筋，表明他的认识还比较肤浅，水平有待提高。

第 15 章讲解“C++函数的高级特性”。对比 C 语言的函数，C++ 的函数增加了默认参数、重载、内联、const 和 virtual 等高级特性，它们展现了很多优点，但是这些优点的背后都隐藏着一些隐患。我们要辩证地看待 C++ 的新机制，应该恰如其分地使用它们，虽然这会使我们编程时多费一些心思，少了一些痛快，但这才是编程的艺术。

第 16 章讲解“C++的异常处理和 RTTI”。提高 C++ 应用软件健壮性的基本手段之一就是使用异常处理技术。C++ 的异常处理技术是一种面向对象的运行时错误处理机制，其思路完全不同于 C 语言旧式的返回值检查策略。本章主要探讨 C++ 异常处理机制的构成及其使用策略，介绍一些 RTTI 的知识。

第 17 章讲解“C++名字空间和模板”。C++ 的名字空间技术是为了解决各个程序员代码中存在的名字冲突问题。模板技术给 C++ 的编程风格带来了巨大的变化，类模板是 STL 泛型容器和迭代器的基础，而函数模板则是泛型算法的基础，它们和运算符重载共同支撑了泛型编程。本章讲述名字空间和模板的目的并不是希望能够把它们掌握和运用到多么深入的地步，而是为学习 STL 打下一些基础。

第 18 章讲解“内存管理”。C++/C 程序员们经常编写内存管理程序，往往提心吊胆，好比进了雷区。如果不想触雷，惟一的办法就是发现所有潜伏的地雷并且排除它们，躲是躲不了的。本章内容足以将程序员从内存雷区中解救出来。**根据本书第一版的读者反映，本章是全书最精彩最有价值的章节，连拍砖者都不忍心下手。第二版做了一些改进，应该更有价值。如果读者正被内存问题搞得焦头烂额，请跳过其他章节直接阅读本章。**

第 19 章讲解“学习和使用 C++ 的 STL”。许多 C++ 程序员对泛型编程和 STL 比较陌生，甚至没有意识到它的存在。本章篇幅长达四十多页，将把 C++ 程序员引领到这个“魔幻”般的世界中。

附录《大学十年》是作者的短篇传记，记录了一个程序员的成长过程，很有趣，并且能催人上进。读者在空闲之际可以翻阅此文，说不准意外收获比研读正文来得更多。

另外，建议读者阅读本书的参考文献，很多是经典名著。如果不读上十来本好的编程书籍，若想成为编程高手是很困难的。

如果你（或者团队）的编程质量已经过关了，不要就此满足，有待于提高的能力因素多着呢。一个企业的软件开发能力如果达到了 CMM/CMMI 3 级或更高水平，那么内功算是到家了。作者目前致力于研究和推广“IT 企业软件过程改进解决方案”（SPIS），相比之下“高质量编程”显得单纯多了。欢迎访问 <http://www.chinaspis.com> 获取更多的信息。

欢迎读者对本书提出批评和建议，请用电子邮件与作者联系：

林 锐：rui.lin@alcatel-sbell.com.cn linrui@vip.sina.com

韩永泉：hanyongquan@sdtm.online.sh.cn



2003 年 3 月于上海贝尔有限公司

目录

第1章 高质量软件开发之道	1
1.1 软件质量基本概念	2
1.1.1 如何理解软件的质量	2
1.1.2 提高软件质量的基本方法	4
1.1.3 “零缺陷”理念	5
1.2 细说软件质量属性	6
1.2.1 正确性	6
1.2.2 健壮性	6
1.2.3 可靠性	7
1.2.4 性能	8
1.2.5 易用性	9
1.2.6 清晰性	9
1.2.7 安全性	10
1.2.8 可扩展性	10
1.2.9 兼容性	10
1.2.10 可移植性	11
1.3 人们关注的不仅仅是质量	11
1.3.1 质量、生产率和成本之间的关系	11
1.3.2 软件过程改进基本概念	14
1.4 高质量软件开发的基本方法	16
1.4.1 建立软件过程规范	16
1.4.2 复用	19
1.4.3 分而治之	21
1.4.4 优化与折中	21
1.4.5 技术评审	22
1.4.6 测试	25
1.4.7 质量保证	27
1.4.8 改错	28

1.5	关于软件开发的一些常识和思考	30
1.5.1	有最好的编程语言吗	30
1.5.2	编程是一门艺术吗	30
1.5.3	编程时应该多使用技巧吗	31
1.5.4	换更快的计算机还是换更快的算法	31
1.5.5	错误是否应该分等级	31
1.5.6	一些错误的观念	32
1.6	小结	33
第2章	做好程序员	35
2.1	漫谈程序员	36
2.2	职业道德	37
2.2.1	上班时间不干与工作无关的事情	38
2.2.2	不损害集体利益	39
2.2.3	不干危害社会的事情	39
2.3	工作态度	40
2.3.1	认真负责	40
2.3.2	服务意识	41
2.4	高效率地工作	42
2.4.1	合理安排一天的时间	42
2.4.2	减少路上花费的时间	44
2.4.3	开会	45
2.4.4	处理电子邮件	45
2.4.5	随时记录	46
2.5	程序员佩服什么样的项目经理	46
2.5.1	丰富的产品开发经验和比较高的技术水平	47
2.5.2	懂得管事和管人	47
2.5.3	较好的人格魅力	48
2.6	将程序员培养成为经理	49
2.7	程序员升为项目经理后是否还要编程	50
2.8	学无止境	54
2.8.1	不断学习新技术	54
2.8.2	提高综合才能	55
2.8.3	向错误和失败学习	57

2.9 小结	59
第3章 编程语言发展简史	61
3.1 编程语言大事记	62
3.2 Ada 的故事	66
3.3 C/C++发展简史	67
3.4 Borland 与 Microsoft 之争	69
3.5 Java 阵营与 Microsoft 的较量	70
3.6 小结	73
第4章 程序的基本概念	75
4.1 程序设计语言	76
4.2 语言实现	77
4.3 开发环境	78
4.4 程序的工作原理	79
第5章 C++/C 程序设计入门	81
5.1 C++/C 程序的基本概念	82
5.1.1 不可缺少的函数 main()	82
5.1.2 main()的命令行参数和返回值	82
5.1.3 内部名称	85
5.1.4 声明、定义和语句	86
5.1.5 变量及其初始化	88
5.1.6 C Runtime Library	88
5.1.7 编译时和运行时的不同	89
5.1.8 编译单元和分离编译技术	90
5.2 基本数据类型和内存映像	91
5.3 类型转换	92
5.3.1 隐式转换	92
5.3.2 强制转换	94
5.4 标识符	96
5.5 转义序列	97
5.6 运算符	98
5.7 表达式	100
5.8 基本控制结构	102
5.9 选择(判断)结构	103