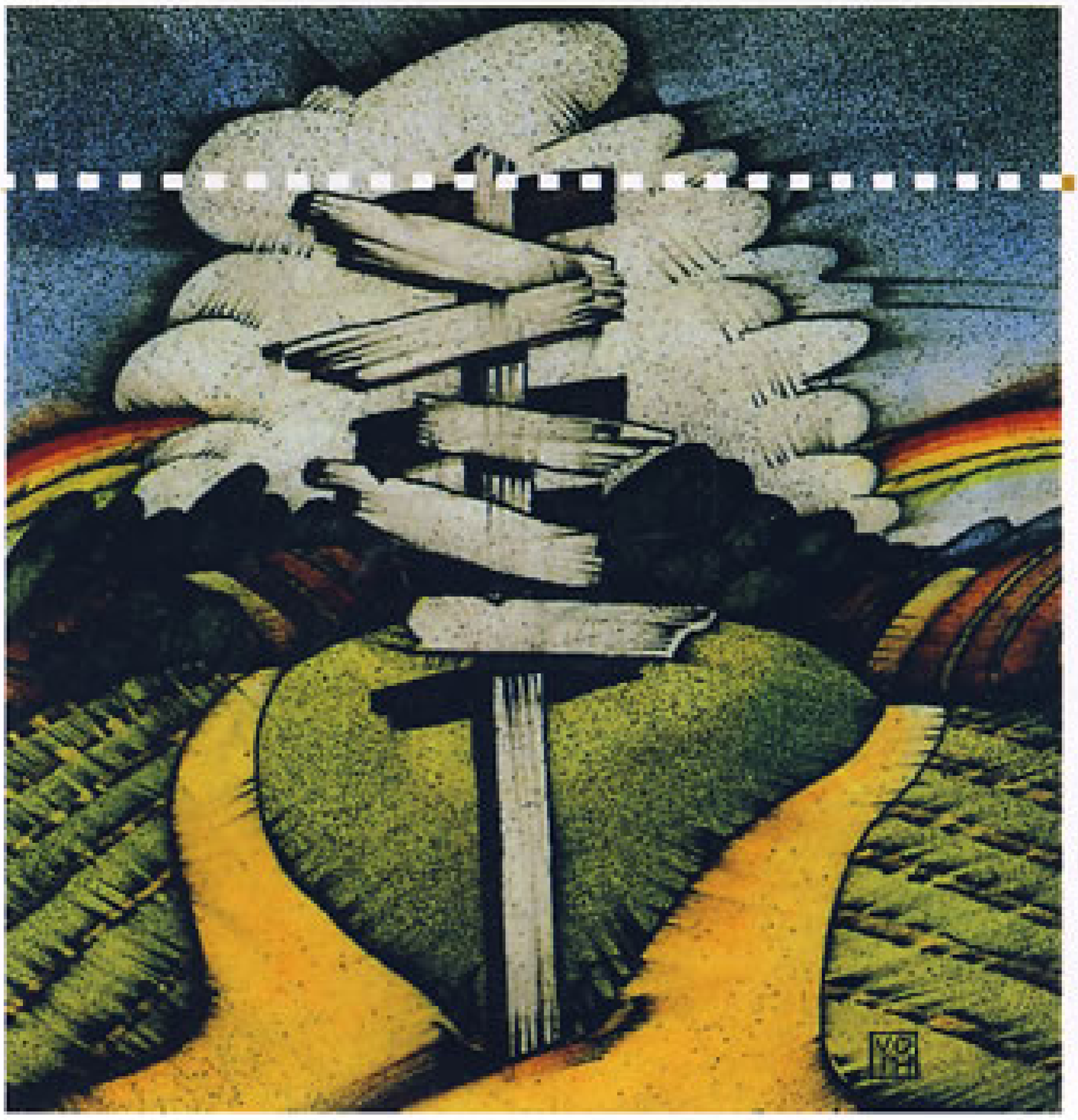
PEARSON
Addison Wesley

C和C++经典著作



C&C++ Classic

C和指针

POINTERS ON C

〔美〕Kenneth A. Reek 著
徐 波 译

人民邮电出版社
POSTS & TELECOM PRESS

内容提要

本书提供与C语言编程相关的全面资源和深入讨论。本书通过对指针的基础知识和高级特性的探讨，帮助程序员把指针的强大功能融入到自己的程序中去。

全书共 18 章，覆盖了数据、语句、操作符和表达式、指针、函数、数组、字符串、结构和联合等几乎所有重要的C编程话题。书中给出了很多编程技巧和提示，每章后面有针对性很强的练习，附录部分则给出了部分练习的解答。

本书适合C语言初学者和初级C程序员阅读，也可作为计算机专业学生学习C语言的参考。

前言

为什么需要这本书

市面上已经有了许多优秀的讲述 C 语言的书籍，为什么我们还需要这一本呢？我在大学里教授 C 语言编程已有 10 个年头，但至今尚未发现一本书是按照我所喜欢的方式来讲述指针的。许多书籍用一章的篇幅专门讲述指针，而且往往出现在全书的后半部分。但是，仅仅描述指针的语法、并用一些简单的例子展示其用法是远远不够的。我在授课时，很早便开始讲授指针，而且在以后的授课过程中也经常讨论指针。我描述它们在各种不同的上下文环境中的有效用法，展示使用指针的编程惯用法(programming idiom)。我还讨论了一些相关的课题如编程效率和程序可维护性之间的权衡。指针是本书的线索所在，融会贯通于全书之中。

指针为什么如此重要？我的信念是：正是指针使 C 威力无穷。有些任务用其他语言也可以实现，但 C 能够更有效地实现；有些任务无法用其他语言实现，如直接访问硬件，但 C 却可以。要想成为一名优秀的 C 程序员，对指针有一个深入而完整的理解是先决条件。

然而，指针虽然很强大，与之相伴的风险却也不小。跟指甲锉相比，链锯可以更快地切割木材，但链锯更容易使你受伤，而且伤害常常来得极快，后果也非常严重。指针就像链锯一样，如果使用得当，它们可以简化算法的实现，并使其更富效率；如果使用不当，它们就会引起错误，导致细微而令人困惑的症状，并且极难发现原因。对指针只是略知一二便放手使用是件非常危险的事。如果那样的话，它给你带来的总是痛苦而不是欢乐。本书提供了你所需要的深入而完整的关于指针的知识，足以使你避开指针可能带来的痛苦。

为什么要学习 C 语言

为什么 C 语言依然如此流行？历史上，由于种种原因，业界选择了 C，其中最主要的原因就在于它的效率。优秀 C 程序的效率几乎和汇编语言程序一样高，但 C 程序明显比汇编语言程序更易于开发。和许多其他语言相比，C 给予程序员更多的控制权，如控制数据的存储位置和初始化过程等。C 缺乏“安全网”特性，这虽有助于提高它的效率，但也增加了出错的可能性。例如，C 对数组下标引用和指针访问并不进行有效性检查，这可以节省时间，但你在使用这些特性时就必须特别小心。如果你在使用 C 语言时能够严格遵守相关规定，就可以避免这些潜在的问题。

C 提供了丰富的操作符集合，它们可以让程序员有效地执行一些底层的计算如移位和屏蔽等，而不必求助汇编语言。C 的这个特点使很多人把 C 称为“高层”的汇编语言。但是，当需要的时候，C 程序可以很方便地提供汇编语言的接口。这些特性使 C 成为实现操作系统和嵌入性控制器软件的良好选择。

C 流行的另一个原因是由于它的普遍存在。C 编译器在许多机器上实现。另外，ANSI 标准提高了 C 程序在不同机器之间的可移植性。

最后，C 是 C++的基础。C++提供了一种和 C 不同的程序设计和实现的观点。然而，如果你对 C 的知识和技巧，如指针和标准库等成竹在胸，将非常有助于你成为一名优秀的 C++程序员。

为什么应该阅读这本书

本书并不是一本关于编程的入门图书。它所面向的读者应该已经具备了一些编程经验，或者是一些想学习 C，但又不想被诸如为什么循环很重要以及何时需要使用 if 语句等肤浅问题耽误进程的人。

另一方面，我并不要求本书的读者以前学习过 C。我讲述了 C 语言所有方面的内容。这种内容的广泛覆盖性使本书不仅适用于学生，也适用于专业人员。也就是说，适用于首次学习 C 的读者和那些经验更丰富的希望进一步提高语言使用技巧的用户。

优秀的 C++书籍把精力集中于与面向对象模型有关的课题上（如类的设计）而不是专注于基本的 C 技巧，这样做是对的。但 C++是建立在 C 的基础之上的，C 的基本技巧依然非常重要，特别是那些能够实现可复用类的技巧。诚然，C++程序员在阅读本书时可以跳过一些他们所熟悉的内容，但他们会在本书中找到许多有用的 C 工具和技巧。

本书的组织形式

本书按照教程的形式组织，它所面向的读者是先前具有编程经验的人。它的编写风格类似于导师在你的身后注视着你的工作，不时给你一些提示和忠告。我的目标是把通常需要多年实践才能获得的知识和观点传授给读者。这种组织形式也影响到材料的顺序——我通常在一个地方引入一个话题，并进行完整的讲解。因此，本书也可以当做参考手册。

在这种组织形式中，存在两个显著的例外之处。首先是指针，它贯穿全书，将在许多不同的上下文环境中进行讨论。其次就是第 1 章，它对语言的基础知识提供了一个快速的介绍。这种介绍有助于你很快掌握编写简单程序的技巧。第 1 章所涉及的主题将在后续章节中深入讲解。

较之其他书籍，本书在许多领域着墨更多，主要是为了让每个主题更具深度，向读者传授通常只有实践才能获得的经验。另外，我使用了一些在现实编程中不太常见的例子，虽然有些不太容易理解，但这些例子显示了 C 在某些方面的趣味所在。

ANSI C

本书描述 ANSI C，是由 ANSI/ISO 9899-1990[ANSI 90]进行定义并由[KERN 89]进行描述的。我之所以选择这个版本的 C 是基于两个原因：首先，它是旧式 C（有时称做 Kernighan 和 Ritchie[KERN 78]，或称 K&R C）的后继者，并已在根本上取代了后者；其次，ANSI C 是 C++的基础。本书中的所有例子都是用 ANSI C 编写的。我常常把“ANSI C 标准文档”简称为“标准”。

排版说明

语法描述格式如下

```
if( expression )
    statement
else
    statement
```

我在语法描述中使用了 4 种字体，其中必需的代码（如此例中的关键字 `if`）将如上所示设置为 Courier New 字体。必要代码的抽象描述（如上例中的 *expression*）用 *Courier New* 表示。有些语句具有可选部分，如果我决定使用可选部分（如此例中的 **else** 关键字），它将严格按上面的例子以**粗体 Courier New** 表示。可选部分的抽象描述（如第 2 个 ***statement***）将以***粗斜体 Courier New*** 表示。每次引入新术语时，我将以**黑体**表示。

完整的程序将标上号码，以“程序 0.1”这样的格式显示。标题给出了程序的名称，包

含源代码的文件名则显示在右下角——这些文件都可以从 Addison Wesley Longman 的网站上找到。

文中有“**提示**”部分。这些提示中的许多内容都是对良好编程技巧的讨论——就是使程序更易编写、更易阅读并在以后更易理解。当一个程序初次写成时，稍微多做些努力就可能节约以后修改程序的大量时间。其他一些提示能帮助你把代码写得更加紧凑或更有效率。

另外还有一些提示涉及软件工程的话题。C 的诞生远早于现代软件工程原则的形成。因此，有些语言特性和通用技巧不为这些原则所提倡。这些话题通常涉及到某种特定结构的效率和代码的可读性与可维护性之间的利弊权衡。这方面的讨论将向你提供一些背景知识，帮助你判断效率上的收益是否抵得上其他质量上的损失。

当你看到“**警告**”时就要特别小心：我将要指出的是 C 程序员新手（有时甚至是老手）经常出现的错误之一，或者代码将不会如你所预想的那样运行。这个警告标志将使提示内容不易被忘记，而且以后回过头来寻找也更容易一些。

“K&R C”表示我正在讨论 ANSI C 和 K&R C 之间的重要区别。尽管绝大多数以 K&R C 写成的程序仅需极微小的修改即可在 ANSI C 环境运行，但有时你仍可能碰到一个 ANSI 之前的编译器，或者遇到一个更老式的程序。如此一来，两者的区别便至关重要。

每章问题和编程练习

本书每章的最后一节是问题和编程练习。问题难简不一，从简单的语法问题到更为复杂的问题诸如效率和可维护性之间的权衡等。编程练习按等级区分难度：★的练习最为简单，★★★★★的练习难度最大。这些练习有许多作为课堂测验已沿用多年。问题或编程练习前如果有一个☜符号，表示在附录中可以找到它的参考答案。

补充材料

Addison Wesley Longman 专门为本书维护了一个 World Wide Web 站点。该站点的 URL 是 http://www.awl.com/cseng/titles/0-673-99986-6/（或可直接访问作者主页 www.cs.rit.edu/~kar/）。这个站点包含本书所有程序的源代码，以章为单位分类。你还可以在上面看到本书的最新勘误表。你还可以联系附近的 Addison Wesley Longman 代表，获取 *Instructor's Guide*，它包含了书上未给出答案的问题和编程练习的所有答案。

如果你是一位教育工作者，也可以免费获取 UNIX 系统上自动递交和测试学生程序的软件[REEK 89, REEK96]，通过匿名 FTP：ftp.cs.rit.edu，目录是 pub/kar/try。

致谢

我无法列出所有对本书做出贡献的人们，但我将感谢他们中的所有人。我的妻子 Margaret 对我的写作鼓励有加，为我提供精神上的支持，而且她默默承受着由于我写作本书而带给她的生活上的孤独。

我要感谢 Warren Caithers 教授，他是我在 RIT 的同事，阅读并审校了本书的初稿。他真诚的批评帮助我从一大堆讲课稿和例子中生成了一份清晰、连贯的手稿。

我非常感谢我的 C 语言编程课程的学生们，他们帮助我发现录入错误，提出改进意见，并在教学过程中忍受着草稿形式的教材。他们对我的作品的反应向我提供了有益的反馈，帮助我进一步改进本书的质量。

我还要感谢 Steve Allan, Bill Appelbe, Richard C.Detmer, Roger Eggen, Joanne Goldenberg, Dan Hinton, Dan Hirschberg, Keith E.Jolly, Joseph F.Kent, Masoud Milani, Steve Summit 和 Kanupriya Tewary，他们在本书出版前对它作了评价。他们的建议和观点对我进一步改进本书的表达形式助益颇多。

最后，我要向我在 Addison-Wesley 的编辑 Deborah Lafferty 女士、产品编辑 Amy Willcutt 女士表示感谢。正是由于她们的帮助，才使这本书从一本手稿成为一本正式的书籍。她们不仅给了我很多有价值的建议，而且鼓励我改进我原先自我感觉良好的排版。现在我已经看到了结果，她们的意见是正确的。

现在是开始学习的时候了，我预祝大家在学习 C 语言的过程中找到快乐！

Kenneth A. Reek
kar@cs.rit.edu
Churchville，纽约

基础的重要性(程序员之路)

学习编程有几年了，感觉走了不少弯路，而不少的学弟学妹又在重蹈我当初的覆辙，不免有些痛心。最近在网上也看了许多前辈们的经验建议，再结合自己的学习经历在这里谈谈基础的重要性，希望帮助大家少走些弯路。

什么是基础呢？就是要把我们大学所学的离散数学,算法与数据结构，操作系统，计算机体系结构，编译原理等课程学好,对计算机的体系,CPU本身,操作系统内核,系统平台,面向对象编程,程序的性能等要有深层次的掌握。初学者可能体会不到这些基础的重要性，学习jsp,donet,mfc,vb的朋友甚至会对这些嗤之以鼻,但是一开始没学好基础就去学jsp或donet会产生很坏的影响,而且陷入其中不能自拔。

我上大二的时候还对编程没什么概念,就上了门C++也不知道能干什么，老师说MFC也不知道是什么东西，看别的同学在学asp.net就跟着学了,然后就了解到.net,j2ee,php是什么了，就觉得软件开发就是用这些了，而上的那些专业课又与我们学的sqlserver啊,css啊,ajax啊,毫无关系,就感慨啊，还不如回家自学去就为一个文凭吗？还不如去培训,浪费这么多钱.于是天天基本上没去上什么课,天天就在做网站,几个学期就做了三个网站。感觉做这些网站就是学到些技巧，没什么进步,这些技巧就好比别人的名字,告诉你你就知道了,网上也都可以搜到。那时候就觉得把.net学好就行了，搞j2ee的比较难，搞api编程就别想了，操作系统更是望尘莫及了。后来随着学习的深入和看了网上许多前辈们的建议才对这些基础的重要性有所体会。

虽然.net或java的开发并不直接用到汇编,操作系统这些,但是不掌握这些基础是有很大问题的，因为你只知其然不知其所有然，在mfc和.net里面控件一拖什么都做好了，很方便，但是出了问题可能就解决不了，有些在网上搜都搜不到。这就是基础没打好,不知道它的原理就不知道出错的原因。在学.net的时候常会讨论那些控件该不该用别人说尽量别用也不知道为什么？不让用是因为你在高层开发,你不知道它的原理出错了你可能解决不了，但其实是应该用的，不然人家开发它干嘛，但要在了解它的原理后去用就会很方便。

要编写出优秀的代码同样要扎实的基础，如果数据结构和算法学的不好，怎么对程序的性能进行优化,怎样从类库中选择合适的数据结构。如果不了解操作系统，怎样能了解这些开发工具的原理,它们都是基于操作系统的。不了解汇编，编译原理，怎么知道程序运行时要多长时间要多少内存，就不能编出高效的代码。

如果没有学好基础一开始就去学.net，java这些越往后就会觉得越吃力，它们涉及的技术太多了，而且不但在更新，对于三层啊，mvc,orm这些架构，你只会用也不明白为什么用，就感觉心里虚，感觉没学好。而你把面向对象，软件工程，设计模式这些基础学好了再去看这些就可以一不变应万变。

大家不要被新名词、新技术所迷惑.NET、XML等等技术固然诱人，可是如果自己的基础不扎实，就像是在云里雾里行走一样，只能看到眼前，不能看到更远的地方。这些新鲜的技术掩盖了许多底层的原理，要想真正的学习技术还是走下云端，扎扎实实的把基础知识学好，有了这些基础，要掌握那些新技术也就很容易了。

开始编程应该先学C/C++,系统api编程，因为它们更接近底层，学习他们更能搞清楚原理。学好了c/C++编程和基础，再去学习mfc,.net这些就会比较轻松，而且很踏实。假设学习VB编程需要4个月，学习基础课程和VC的程序设计需要1年。那么如果你先学VB，再来学习后者，时间不会减少，还是1年，而反过来，如果先学习后者，再来学VB，也许你只需要1个星期就能学得非常熟练。

编程就好比练功，如果学习.net,mfc,vb等具体的语言和工具是外功(招式)，对基础的学习就是内功,只注重招式而内功不扎实是不可能成为高手的。很多人会认为《射雕英雄传》中马玉道长什么都没有教郭靖，马道长教的表面看来是马步冲权实则都是内功心法，郭靖拜师洪七之后开始练习降龙十八掌凭借的就是这深厚的内功，吞食蝮蛇宝血又加上练习了周博通传授的九阴真经和外加功夫双手互博技之后，终于练就行走江湖的武功，由此可见马玉道长传授给了郭靖的是最基础的，也是最重要的观念,编程也好比盖高楼，根基没打好早晚有一天会挎掉的，而且盖得越高，损失也越惨重。这些底层知识和课本不是没有用也不是高深的不能学，而是我们必须掌握的基础。

这些是个人的愚见，说的不是很清楚，大家可以看看这些前辈们的经验，相信看完后大家一定会有所体会的。为了方便大家阅读，我把这些前辈们的建议的文章整理成了pdf,大家在下面下载吧!希望对大家有帮助。pdf地址:http://bbs.theithome.com/read-htm-tid-123.html

说了这么多无非是想告诫大家要打好扎实的基础，不要只顾追求时髦的技术，打好基础再去学那些技术或是参加些培训，对自身的发展会更好的。

基础这么重要怎样学好它呢？我觉得学好它们应该对照这些基础课程所涉及的方面,多看一些经典书籍,像算法导论,编程珠玑,代码大全(具体介绍在本论坛每本书的版块里)等,这些经典书籍不仅能帮助我们打好基础，而且对我们的程序人生也能产生莫大的影响，相信认真研究看完这些书籍后，我们的程序之路会十分顺畅。然而这些书籍并不好读,有些甚至相当难读，国内的大学用这些书当教材的也不多,这些书又偏向理论,自己读起来难免会有些枯燥无味。于是就想到建一个论坛，大家共同讨论学习这些书籍，就会学的更踏实更牢固更有趣,这样就能为以后的学习打下扎实的基础。

本论坛特色： bbs.theithome.com

1.为计算机初学者或基础不太扎实的朋友指明方向，要注重内功

2.为学习者推荐经典书籍，指明应看哪些书籍，怎样练内功

3.为学习者提供一个交流的地方，更容易学好，不会那么枯燥

4.对每本书分章分别讨论，更专，会学的更踏实更牢固

5.讨论的都是经典书籍，每一本都会让我们受益匪浅,对每本书分别讨论是很有意义的。

计算机经典书籍汇总,下载地址：http://bbs.theithome.com/read-htm-tid-308.html

1.计算机科学概论
计算机科学概论
2.计算机数学基础
高等数学
线性代数
概率论与数理统计
离散数学及其应用
离散数学教程(北大版)
什么是数学
具体数学：计算机科学基础
3.C语言
谭浩强C程序设计
C primer plus
The C programming language
C和指针
C专家教程
C陷阱与缺陷
c语言解惑
C标准库
你必须知道的495个C语言问题
4.算法与数据结构
数据结构(清华版)
数据结构与算法分析—C语言描述
编程珠玑
编程珠玑II
算法导论
计算机程序设计艺术卷1
计算机程序设计艺术卷2
计算机程序设计艺术卷3
5.电子技术基础
模拟电子技术(童诗白版)
数字逻辑与数字集成电路(清华版)
6.汇编语言
汇编语言(王爽版)
80X86汇编语言程序设计教程
Intel汇编语言程序设计
IBM PC汇编语言程序设计(国外版)
高级汇编语言程序设计
保护方式下的80386及其编程
黑客反汇编揭秘
Windows环境下32位汇编语言程序设计
7.计算机硬件原理
计算机组成-结构化方法
微机原理与接口技术(陈光军版)
计算机体系结构(张晨曦版)
计算机组成与设计硬件/软件接口
Intel微处理器结构、编程与接口
计算机体系结构(量化研究方法)
编程卓越之道卷1
编程卓越之道卷2
深入理解计算机系统
编码的奥秘
8.数据库系统原理
数据库系统概念
数据库系统导论
数据库系统实现
9.编译原理
编译原理(清华第2版)
编译原理及实践
编译原理:原则,技术和工具
现代编译原理-C语言描述
高级编译器设计与实现
10.操作系统原理
操作系统概念
现代操作系统
链接器和加载器
程序员的自我修养:链接、装载与库
自己动手写操作系统
操作系统设计与实现
11.计算机网络
计算机网络(Computer Networks)
TCP-IP详解卷1
TCP-IP详解卷2
TCP-IP详解卷3
用TCP/IP进行网际互联(第一卷)
用TCP/IP进行网际互联第二卷
用TCP/IP 进行网际互联第三卷
12.软件工程和面向对象程序设计
C++编程思想卷1
java编程思想
软件工程(Software.Engineering)
软件工程：实践者的研究方法
深入浅出面向对象分析与设计
head first设计模式
道法自然:面向对象实践指南
面向对象分析与设计
敏捷软件开发：原则、模式与实践
设计模式:可复用面向对象软件的基础
测试驱动开发
重构—改善既有代码的设计
代码大全
程序设计实践
程序员修炼之道：从小工到专家
卓有成效的程序员
代码之美
人月神话
计算机程序的构造和解释
观止-微软创建NT和未来的夺命狂奔
代码优化：有效使用内存[美]克里斯·卡巴斯基
编程高手箴言(梁肇新)
游戏之旅-我的编程感悟(云风)
13.windows编程基础
Windows操作系统原理
Inside Windows 2000
深入解析Windows操作系统
天书夜读：从汇编语言到Windows内核编程
windows程序设计
WINDOWS核心编程
14.linux/unix编程基础
鸟哥的Linux私房菜:基础学习篇
鸟哥的Linux私房菜:服务器架设篇
linux程序设计
UNIX环境高级编程
Unix网络编程卷1
UNIX网络编程卷2
UNIX编程艺术
UNIX Shell范例精解
15.Linux/unix内核源代码和驱动程序
Linux内核设计与实现
LINUX内核源代码情景分析
深入理解LINUX内核
Linux内核完全注释
Linux设备驱动程序
16.C++语言
C++编程思想2
Essential C++
C++ primer
C++程序设计语言
C++语言的设计和演化
Accelerated C++
Effective C++
More Effective C++
Exceptional C++
More Exceptional C++
C++设计新思维
深度探索C++对象模型
C++沉思录
C++ Templates: The Complete Guide
C++ FAQs
17.标准库STL使用
C++标准程序库
Effective STL
泛型编程与STL
18.STL源代码
STL源码剖析
19.java语言
java编程思想
Java编程规范