PEARSON

C和C++实务精选
品味岁月积淀，读享技术菁华

你必须知道的495个C语言问题

[美] | Steve Summit | 著
孙云 朱群英 | 译

C Programming FAQs:
Frequently Asked Questions

- 全球C语言程序员集体智慧的结晶
- Amazon全五星好评图书
- 权威解答495个最常遇到的C语言问题

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

PEARSON

[美] | Steve Summit | 著
孙云 朱群英 | 译

你必须知道的495个C语言问题

C Programming FAQs:
Frequently Asked Questions

人民邮电出版社
北京

图书在版编目（CIP）数据

你必须知道的495个C语言问题 / （美）萨米特(Summit,S.）著 ; 孙云，朱群英译. — 北京 : 人民邮电出版社，2016.4
ISBN 978-7-115-37676-3

Ⅰ. ①你… Ⅱ. ①萨… ②孙… ③朱… Ⅲ. ①C语言—程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2015)第007122号

版权声明

◆ 著　　[美] Steve Summit
译　　孙　云　朱群英
责任编辑　傅道坤
责任印制　张佳莹　焦志炜

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京天宇星印刷厂印刷

◆ 开本：800×1000　1/16
印张：18.25
字数：393 千字　　2016 年 4 月第 1 版
印数：1 – 3 000 册　　2016 年 4 月北京第 1 次印刷
著作权合同登记号　图字：01 -2008 -5465 号

定价：45.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315

内 容 提 要

本书以问答的形式组织内容，讨论了学习或使用C语言的过程中经常遇到的一些问题。书中列出了C用户经常问的400多个经典问题，涵盖了初始化、数组、指针、字符串、内存分配、库函数、C预处理器等各个方面的主题，并分别给出了解答，而且结合代码示例阐明要点。

本书结构清晰，讲解透彻，是各高校相关专业C语言课程很好的教学参考书，也是各层次C程序员的优秀实践指南。

原 版 序

1979年的某段时间，我听到很多人在谈论C这个当时还挺新的语言和那本刚刚推出的书。我买了一本Brian Kernighan和Denis Ritchie写的*The C Programming Language*（也称K&R），但它在我的书架上空等了好一阵子，因为当时我并不急着需要它（况且我那时候还是一个余暇无多的大一新生）。后来证明这本书买得很幸运，因为当我最后拿起它以后，就再也没有放下了：从那以后，我就一直在用C语言编程。

1983年我结识了新闻组net.lang.c，这（以及它的后继者comp.lang.c）是一个绝佳的地方，你可以学习C语言的方方面面，发现别人关于C语言的各种疑问，认识到你可能根本还没有掌握关于C语言的一切。C语言尽管表面上很简单，但也还有一些并不显而易见的方面，有些问题不断有人问起。本书根据我从1990年5月开始在comp.lang.c上发布的常见问题（FAQ）列表收集了这样的一些问题，并提供了答案。

然而我得声明，这本书并不是对C语言的批评或诽谤。用户在使用时遇到困难，很容易迁怒于语言（或其他任何工具）或者要求正确设计的工具“应该”防止用户的误用。因此看到书中提及的各种误用以后，很容易将这样的书看作试图显示C语言的先天不足的长篇控诉。这实在是远悖我的本意。

如果我不认为C语言是一门伟大的语言，或者没有在这种语言的编程中获得那么多的乐趣，那我永远也学不到足够的关于C语言的知识来写出本书，而且也不会试图写出本书来让别人更爱用C语言。我很喜欢C语言，我教C的课并花时间参与网上讨论的原因之一，就是希望发现这门语言（或者说编程本身）在哪些方面比较难学，让人不易高效地编程。本书展示了我认识到的部分内容，这些问题毫无疑问就是人们遇到麻烦最多的，而答案则经过多年的反复修正，就是为了消除人们的麻烦。

如果这些答案中有任何错误，那么读者一定会遇到麻烦。尽管审稿人和我都尽力去除所有的错误，但从一部手稿中根除最后一个错误，就跟从程序中去掉最后一个bug一样困难。通过出版社转交或发往我的E-mail地址的任何修正和建议我都感激不尽。同时我也对任何错误的第一个发现者按惯例提供$1.00的报酬。如果你能够访问因特网，你可以在问题20.47提到的ftp和http网址中找到一份勘误表（和错误发现者的积分表）。

希望我已经澄清，这本书并不是对C语言的批评，也不是对我学习过的C语言的书或其作者的批评。从K&R中我不仅学到了C语言，还学会了编程。在试图用自己的贡献来丰富C语言的文献的时候，我唯一遗憾的就是本书没有做到K&R第二版发现的妙处，即“C不是一门复杂的语言，并不值得为它写本厚书”。我希望那些深深地欣赏C语言（及K&R）的简洁和精确的人，看到本

书反反复复讲述某些东西，或用3种稍稍不同的方式讲述一个问题的时候不要生气。

尽管封面上只印着我一个人的名字，但这本书的背后却有许许多多的人，简直不知道该从哪里开始感谢。从某种意义上讲，comp.lang.c的每个读者（现在约有320 000人）都做出了贡献：这本书背后的FAQ列表是为comp.lang.c写的，因而本书也保留了comp.lang.c良好的讨论氛围。

我希望这本书也保留了我开始阅读net.lang.c时所学习的正确C语言编程的思想。因此，我要首先感谢我所知的一贯清楚解释这种思想的人：Doug Gwyn、Guy Harris、Karl Heuer、Henry Spencer和Chris Torek。这些绅士们多年来不断耐心、慷慨而睿智地解答各种无穷尽的问题。是我出头写下这些常见问题的，但不要以为是我给出了这些答案。我曾经是个学生（我想正是Guy解答了我提出的问题，现为本书问题5.10），我对走在前面的大师们感激不尽。这本书与其说是我的，不如说是他们的。但对书中的不足和错误我愿一力承担。

在线FAQ在变成本书的过程中增长了3倍，它的增长一度太快也变得有些笨拙了。Mark Brader、Vinit Carpenter、Stephen Clamage、Jutta Degener、Doug Gwyn、Karl Keuer、Joseph Kent和George Leach阅读了部分或全部的手稿，帮我对这一过程施加了一些控制，感谢他们大量的仔细建议和修正。他们的努力都源自一个共同的愿望，期待在编程社区中提高对C语言的整体理解。感谢他们的贡献。

这些审稿人中有3个长期以来也是在线FAQ的贡献者。感谢Jutta Degener和Karl Heuer多年来的帮助，尤其感谢Mark Brader，从5年前我第一次在comp.lang.c上发布FAQ以来，他就一直给予我批评。我不知道他哪里来的毅力提出那么多的建议和修正，其中部分还遭到了我持久顽固的拒绝，即使（正如我最后意识到的）它们实际的确是改进。你可以感谢Mark为本书提供的很多解释的表述形式，而弄糟的部分就责怪我吧。

还要感谢：Susan Cyr设计的封面；Bob Dinse和Eskimo North提供的网络环境，这对这样的项目至关重要；Bob Holland提供的计算机，这本书的大部分内容都是用它写成的；Pete Keleher提供的Alpha文本编辑器；华盛顿大学数学研究和工程图书馆提供的图书查询便利；华盛顿大学海洋学系借给我的磁带驱动器，用来访问我尘封的新闻组旧帖。

感谢Tanmoy Bhattacharya提供的问题11.11中的例子，感谢Arjan Kenter提供的问题13.7中的代码，感谢Tomohiko Sakamoto提供的问题20.37中的代码，感谢Roger Miller提供的问题11.38中的一行文字。

感谢世界各地的人们通过提供建议、修正、建设性的批评或其他支持对FAQ的贡献：Jamshid Afshar, Lauri Alanko, Michael B. Allen, David Anderson, Jens Andreasen, Tanner Andrews, Sudheer Apte, Joseph Arceneaux, Randall Atkinson, Kaleb Axon, Daniel Barker, Rick Beem, Peter Bennett, Mathias Bergqvist, Wayne Berke, DanBernstein, Tanmoy Bhattacharya, John Bickers, Kevin Black, Gary Blaine, Yuan Bo,Mark J. Bobak, Anthony Borla, Dave Boutcher, Alan Bowler, breadbox@muppetlabs.com,Michael Bresnahan, Walter Briscoe, Vincent Broman, Robert T. Brown, Stan Brown, John R. Buchan, Joe Buehler, Kimberley Burchett, Gordon Burditt, Scott Burkett, Eberhard Burr, Burkhard Burow, Conor P. Cahill, D'Arcy J.M. Cain, Christopher Calabrese, Ian Cargill, Vinit Carpenter, Paul Carter, Mike Chambers, Billy Chambless, C. Ron Charlton, Franklin Chen,

Jonathan Chen, Raymond Chen, Richard Cheung, Avinash Chopde,Steve Clamage, Ken Corbin, Dann Corbit, Ian Cottam, Russ Cox, Jonathan Coxhead,Lee Crawford, Nick Cropper, Steve Dahmer, Jim Dalsimer, Andrew Daviel, James Davies,John E. Davis, Ken Delong, Norm Diamond, Jamie Dickson, Bob Dinse, dlynes@plenarysoftware, Colin Dooley, Jeff Dunlop, Ray Dunn, Stephen M. Dunn, Andrew Dunstan,Michael J. Eager, Scott Ehrlich, Arno Eigenwillig, Yoav Eilat, Dave Eisen, Joe English,Bjorn Engsig, David Evans, Andreas Fassl, Clive D.W. Feather, Dominic Feeley, Simao Ferraz, Pete Filandr, Bill Finke Jr., Chris Flatters, Rod Flores, Alexander Forst, Steve Fosdick, Jeff Francis, Ken Fuchs, Tom Gambill, Dave Gillespie, Samuel Goldstein, Willis Gooch, Tim Goodwin, Alasdair Grant, W. Wesley Groleau, Ron Guilmette, Craig Gullixson, Doug Gwyn, Michael Hafner, Zhonglin Han, Darrel Hankerson, Tony Hansen, Douglas Wilhelm Harder, Elliotte Rusty Harold, Joe Harrington, Guy Harris, John Hascall, Adrian Havill, Richard Heathfield, Des Herriott, Ger Hobbelt, Sam Hobbs, JoelRay Holveck, Jos Horsmeier, Syed Zaeem Hosain, Blair Houghton, Phil Howard, Peter Hryczanek, James C. Hu, Chin Huang, Jason Hughes, David Hurt, Einar Indridason,Vladimir Ivanovic, Jon Jagger, Ke Jin, Kirk Johnson, David Jones, Larry Jones, Morris M. Keesan, Arjan Kenter, Bhaktha Keshavachar, James Kew, Bill Kilgore, Darrell Kindred, Lawrence Kirby, Kin-ichi Kitano, Peter Klausler, John Kleinjans, Andrew Koenig, Thomas Koenig, Adam Kolawa, Jukka Korpela, Przemyslaw Kowalczyk, Ajoy Krishnan T, Anders Kristensen, Jon Krom, Markus Kuhn, Deepak Kulkarni, Yohan Kun, B. Kurtz, Kaz Kylheku, Oliver Laumann, John Lauro, Felix Lee, Mike Lee, Timothy J. Lee, Tony Lee, Marty Leisner, Eric Lemings, Dave Lewis, Don Libes, Brian Liedtke, Philip Lijnzaad, James D. Lin, Keith Lindsay, Yen-Wei Liu, Paul Long, Patrick J. LoPresti, Christopher Lott, Tim Love, Paul Lutus, Mike McCarty, Tim McDaniel, Michael MacFaden, Allen Mcintosh, J. Scott McKellar, Kevin McMahon, Stuart MacMartin, John R. MacMillan, Robert S. Maier, Andrew Main, Bob Makowski, Evan Manning, Barry Margolin, George Marsaglia, George Matas, Brad Mears, Wayne Mery, De Mickey, Rich Miller, Roger Miller,Bill Mitchell, Mark Moraes, Darren Morby, Bernhard Muenzer, David Murphy, Walter Murray, Ralf Muschall, Ken Nakata, Todd Nathan, Taed Nelson, Pedro Zorzenon Neto,Daniel Nielsen, Landon Curt Noll, Tim Norman, Paul Nulsen, David O'Brien, Richard A. O'Keefe, Adam Kolawa, Keith Edward O'hara, James Ojaste, Max Okumoto, Hans Olsson, Thomas Otahal, Lloyd Parkes, Bob Peck, Harry Pehkonen, Andrew Phillips, Christopher Phillips, Francois Pinard, Nick Pitfield, Wayne Pollock, Polver@aol.com, Dan Pop, Don Porges, Claudio Potenza, Lutz Prechelt, Lynn Pye, Ed Price, Kevin D. Quitt, Pat Rankin, Arjun Ray, Eric S. Raymond, Christoph Regli, Peter W. Richards, James Robinson, Greg Roelofs, Eric Roode, Manfred Rosenboom, J.M. Rosenstock, Rick Rowe, Michael Rubenstein, Erkki Ruohtula, John C. Rush, John Rushford, Kadda Sahnine, Tomohiko Sakamoto, Matthew Saltzman, Rich Salz, Chip Salzenberg, Matthew Sams, Paul Sand, David W. Sanderson, Frank Sandy, Christopher Sawtell, Jonas Schlein, Paul Schlyter, Doug Schmidt, Rene Schmit, Russell Schulz, Dean Schulze, Jens Schweikhardt, Chris Sears, Peter Seebach, Gisbert W. Selke, Patricia Shanahan, Girija Shanker, Clinton Sheppard, Aaron Sherman, Raymond Shwake, Nathan Sidwell, Thomas Siegel, Peter

da Silva, Andrew Simmons, Joshua Simons, Ross Smith, Thad Smith, Henri Socha, Leslie J. Somos, Eric Sosman, Henry Spencer, David Spuler, Frederic Stark, James Stern,Zalman Stern, Michael Sternberg, Geoff Stevens, Alan Stokes, Bob Stout, Dan Stubbs, Tristan Styles, Richard Sullivan, Steve Sullivan, Melanie Summit, Erik Talvola, Christopher Taylor, Dave Taylor, Clarke Thatcher, Wayne Throop, Chris Torek, Steve Traugott, Brian Trial, Nikos Triantafillis, Ilya Tsindlekht, Andrew Tucker, Goran Uddeborg, Rodrigo Vanegas, Jim Van Zandt, Momchil Velikov, Wietse Venema, Tom Verhoeff, Ed Vielmetti, Larry Virden, Chris Volpe, Mark Warren, Alan Watson, Kurt Watzka, Larry Weiss, Martin Weitzel, Howard West, Tom White, Freek Wiedijk, Stephan Wilms, Tim Wilson, Dik T. Winter, Lars Wirzenius, Dave Wolverton, Mitch Wright, Conway Yee, James Youngman, Ozan S. Yigit和Zhuo Zang。[①]我试图记录下我采纳建议的每一个人，但我担心可能还是遗漏了一些，对那些名字本该出现在这里而没有出现的人，我诚恳地致以道歉。

最后，我要感谢Addison-Wesley的编辑Debbie Lafferty，她有一天发邮件问我是否有兴趣写这本书。我有兴趣，你现在手里拿的就是它。我希望这本书能让你跟我一样觉得C语言编程令人快乐。

Steve Summit
scs@eskimo.com
1995年7月于华盛顿州西雅图市

① 致谢名单根据最新资料整理。——编者注

前　言

你可能在酒吧或聚会上有这样的经历，有人跟你打赌让你做一些看似简单，但最后却限于人体特质或物理规律而根本无法完成的事情。跟你打赌的人知道，他挑战的人越多，他持续获胜的可能性就越大，因为这些特质或规律虽然十分隐晦，却是相当稳定、可以预测的。

同样，如果你让很多人来完成一个复杂任务，如学习C语言，他们肯定会遇到同样的困难，提出同样的问题。在最初设计任务的时候这些困难和问题也许不能预见，而答案也恐怕是“后见之明”，但人们依然会不断遇到同样的困难，也会不断提出同样的问题。这些困难和问题并不表明任务就不能完成，只能说明它比较困难，从而变得很有趣。

毫不奇怪，这些问题在因特网尤其是互动讨论的新闻组上不断被问起。将这些常见问题收集起来的想法是顺理成章的，顺着这一想法形成了常见问题（FAQ）列表的传统。FAQ列表未必总能达成最初设想的减少常见问题发生率的目的，但如果问题是一贯的，那它们被经常问到并纳入FAQ列表的事实说明，它们也许正是你或本书的其他读者要问的问题。

关于本书

多数（关于C语言或其他任何主题的）书都是从作者的角度写成的。它们是用一种作者自己明白的方式来讨论作者认为你应该知道的主题。如果那种方式不适合你（在某种程度上，也不可能适合，因为作者预先已经知道那些内容，而你却全然不知），你很有可能被弄得满头雾水。

而这本书却不一样，它由400多个问题组织而成，所有问题都是人们在学习C语言编程的过程中提出的真实问题。本书不是针对作者认为重要的议题，而是针对真正的读者认为重要的议题，讨论他们提出的问题。如果你在学习或使用C语言，而你遇到的问题在别的书里都找不到答案，那么你很有可能会在这里找到答案。

本书不能保证解答你在C语言编程中遇到的所有问题，因为在编程实践中产生的很多问题都跟你的问题领域有关，而本书只涵盖了C语言本身。正如它不能涵盖每个人试图用C语言解决的每个问题的每个方面，本书也不能涵盖每个人用C语言编程时用的每个操作系统的方方面面或每个人希望用C语言实现的每个算法。具体的问题、具体的操作系统和通用的算法都有专门的书和其他材料进行讨论。不过，某些跟操作系统和算法相关的问题十分常见，因此第19章和第20章对其中一些问题提供了简单的、介绍性的答案，但不要期待它们很完备。

本书中的问题是人们在读完一本C语言入门书或上了一门C语言课程之后常常会提到的。因此本书不是一点点教你学C语言，也不会讨论任何C语言教材都会讨论的基础问题。而且，本书

的答案在极大程度上都应该是绝对正确，不会传播任何误解的。因此有些答案初看起来显得有些过于详细，它们要向你提供完整的图景，而不能过于简化而略去了重要的细节。（毕竟，很多这类细节正是本书的问答中提到的诸多错误观点的根源。）在这些详尽的答案中，必要的地方会有捷径和简化处理，而在术语表中你会找到术语的精确定义，帮你准确解释很多问题。当然，这些捷径和简化处理都是安全的，它们不会导致以后的误解，而如果你需要完整的版本，你总是可以找到更详尽的解释或者查到相关的参考文献。

正如我们会在第3章和第11章中看到的那样，C语言的标准定义并没有规定每个写成的C程序的行为。有些程序陷入了各种灰色地带：它们可能在某些系统上能运行，而且严格说来也不非法，但却不能确保在各处都能运行。本书介绍的是可移植的C编程，因此在答案中建议，只要可能就不用不可移植的方法。

本书所基于的在线FAQ列表是一种对话的方式，当人们看不懂的时候，就会直言不讳地提出来。这样的及时反馈非常有利于改善解答的形式。尽管出成书以后就不能这么动态了，但这样的对话方式依然适用：欢迎你的意见、批评和建议。如果你能访问因特网，可以发送意见到scs@eskimo.com，或者寄信让出版社转交。本书的堪误表可以在因特网上获得，并在网上进行维护，具体参见问题20.47中的信息。

问题格式

本书的内容包含一系列的问题及答案。很多答案中还列举了参考书目；有些还有脚注，如果你觉得它们太吹毛求疵，可以略过。

等宽字体用来表示C语法（函数和变量名、关键字等），也用来表示一些操作系统命令（如CC等）。偶尔出现的tty(4)这样的符号表示*UNIX Programmer's Manual*第4章的“tty”一节。

代码示例

这是本关于C语言的书，因此必要处给出了许多C程序段。这些例子主要用来清楚地展示。它们不一定总是用最高效的方式写成的，让它们更“快”往往会导致更不清楚。（关于代码效率的信息参见问题20.14。）它们通常都是用现代的ANSI风格的语法写成。如果你还在使用“经典的（classic）”编译器，参见问题11.31关于转换的提示。

作者和出版社欢迎你在自己的程序中使用和修改这些代码片段，当然如果你能提及作者，我们将十分感激。（某些片段来自其他来源，也采用同样的策略。如果你使用这些代码，请感谢对应的贡献者。）较大的例子的源码可以通过匿名ftp从aw.com的cseng/authors/summit/cfaq下载（参见问题18.12）。

为了强调某些要点，我不得不举一些不能那样做的反例子。在答案中，这样的代码片段都用`/*WRONG*/`这样明确的注释标示出来了，提示你不要模仿。（问题中的代码片段通常没有类似的标示，但从问题本身的提法上应该很容易看出代码片段有问题，因为这些问题通常是“为什么这样做不行”。）

组织

如前所述，本书的问题来自人们在实际的工作或学习中提出的真实问题，这些问题有时不能很好地归类。很多问题涉及多个主题：看似内存分配的问题实际原因却是声明错误。（有些问题在两章中都会出现，以便它们更容易找到。）无论如何，这不是一本必须从头读到尾的书：使用目录、索引和问题之间的交叉索引来找你感兴趣的主题。（如果你有空闲时间从头读到尾，可能会遇到你没有想到过的问题的答案。）

通常，在开始写代码之前要先声明数据结构，因此第1章从声明和初始化开始谈起。C语言的结构、联合和枚举类型足够复杂，值得为它们单独写一章。第2章讨论了它们的声明和使用方法。

程序中多数的工作由表达式语句完成，这是第3章的主题。

第4章到第7章讨论了许多新C程序员最头疼的内容：指针。第4章从总体上讨论指针，第5章专门讨论空指针这一特殊情况，第6章描述了指针和数组的关系，而第7章则探究了指针错乱背后的真正问题：底层的内存分配。

几乎所有的C程序都会操作字符和字符串，但这些类型却在语言的底层实现。程序员通常需要负责正确管理这些类型，第8章收集了管理这些类型时出现的问题。类似地，C语言也没有正式的布尔类型，第9章简单讨论了C语言的布尔表达式以及（在需要的时候）实现用户定义的布尔类型的正确方法。

C语言预处理器（编译器的一部分，负责处理`#include`和`#define`指令——实际上负责所有以#开始的行）非常独特，它几乎就是一门独立的语言，因此也单独有一章：第10章。

ANSI C标准委员会（X3J11），在澄清C语言的定义让它简单易懂的过程中，引入了一些新的功能并做出了一些重要的改变。与ANSI标准C相关的问题收集在第11章。如果你用过ANSI之前的C语言（也称“K&R”或“经典”C），你会觉得第11章介绍的差别很有用处。另一方面，如果你已经在顺利地使用ANSI C，那么二者的功能差别可能就没什么意思了。无论如何，第11章中涉及其他主题（如声明、预处理器和库函数等）的所有问题也会在其他章节出现或被交叉引用。

C语言的定义相对简洁，部分原因是许多功能并不是由语言本身而是由库函数提供的。其中最重要的就是“标准I/O”库，或称stdio函数，这在第12章讨论。其他的库函数在第13章讨论。

第14章和第15章讨论了两个更高级的主题：浮点数和可变参数列表。无论使用哪种系统或哪种语言，浮点数运算都颇有技巧。第14章简述了一些一般的浮点数问题和一些C语言特有的问题。函数可以接受可变参数的可能性虽然有人认为没必要而且危险，但有时却很方便而且也是`printf`函数的核心功能。处理可变参数列表的技巧在第15章讨论。

如果你对前面的内容已经比较熟悉，那么第16章的问题你可能希望最先看到：它们涉及偶尔出现的奇怪问题和程序中冒出的极难跟踪的神秘bug。

当有两种以上编写某个程序的“正确”方法时（通常都有），人们往往根据主观的标准进行选择，这些标准不止跟代码正确编译和运行有关系。第17章讨论了一些编程风格上的问题。

你不能孤立地创建C程序：你需要编译器，可能还需要一些附加的文档、源码或工具。第18章讨论了一些可获得的工具和资源，包括`lint`，一个快要被遗忘但曾经是检查程序正确性和可移植性的不可或缺的工具。

如前所述，C语言没有规定让一个真正程序运行所需的各个方面。像“怎样从键盘直接读入字符而不用等回车键”和“如何得到文件的大小”这样的问题十分常见，但C语言却没有给定答案，这些操作依赖底层的操作系统提供的工具。第19章提出了一些这样的问题，同时提供了一些在常用操作系统下的答案。

最后，第20章收集了一些不能放入其他章节的杂项问题：位操作、效率、算法、C语言和其他语言的关系，以及一些琐碎的问题。（第20章的介绍对内容有更详尽的划分。）

最后用两个跟本书而不是C语言关系更密切的预备问题来结束这个介绍。

问： 既然因特网上有免费版本可以用，我为什么还要花钱买这本书呢？

答： 这本书包含的内容大约是发表在comp.lang.c上的内容的三倍，而且尽管电子文档有各种优势，但阅读这么大的信息量用印刷的形式还是更容易一些。（从网上下载再打印这么多内容会花很多时间，而版面也没有这么漂亮。）

问： “FAQ”如何发音？

答： 我的发音是“eff ay kyoo”，而且我相信这就是FAQ在“发明”时的最初的发音。很多人现在读作“fack”，这很容易令人联想到单词“fact”。对复数形式，我会读作“eff ay kyooze”，但很多人读作“fax”。这些发音都没有什么严格的对错，“FAQ”是个新词，而流行的用法会在任何新词的演化过程中扮演重要的角色。

（另外，还有一个类似的疑问“FAQ”是仅仅表示某个问题，还是包括该问题和答案，还是全部问题和答案呢？）

现在，开始真正的问题之旅！

目　录