TURING 图灵程序设计丛书

C程序员必读经典

原版畅销11年

「毒舌程序员」
为你揭开指针的
真实面纱

C征服指针

[日] 前桥和弥 著
吴雅明 译

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

征服C指针

[日] 前桥和弥 著
吴雅明 译

人民邮电出版社
北京

图书在版编目（CIP）数据

征服C指针 /（日）前桥和弥著 ；吴雅明译. -- 北京 ：人民邮电出版社，2013.2
（图灵程序设计丛书）
ISBN 978-7-115-30121-5

Ⅰ. ①征… Ⅱ. ①前… ②吴… Ⅲ. ①C语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2012)第280432号

内 容 提 要

本书被称为日本最有营养的C参考书。作者是日本著名的“毒舌程序员”，其言辞犀利，观点鲜明，往往能让读者迅速领悟要领。

书中结合了作者多年的编程经验和感悟，从C语言指针的概念讲起，通过实验一步一步地为我们解释了指针和数组、内存、数据结构的关系，展现了指针的常见用法，揭示了各种使用技巧。另外，还通过独特的方式教会我们怎样解读C语言那些让人“纠结”的声明语法，如何绕过C指针的陷阱。

本书适合C语言中级学习者阅读，也可作为计算机专业学生学习C语言的参考。

图灵程序设计丛书

征服C指针

◆ 著　　[日] 前桥和弥
译　　吴雅明
责任编辑　傅志红
执行编辑　乐　馨

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街14号
邮编　100061　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京隆昌伟业印刷有限公司印刷

◆ 开本：800×1000　1/16
印张：16.5
字数：333千字　　2013年2月第1版
印数：1-4 000册　　2013年2月北京第1次印刷

著作权合同登记号　图字：01-2012-4256号

ISBN 978-7-115-30121-5

定价：49.00元

读者服务热线：(010)51095186转604　印装质量热线：(010)67129223

反盗版热线：(010)67171154

版权声明

译 者 序

在平时的工作中，我时常遇到两种人：一种是刚毕业的新人，问他们："以前学过C语言吗？"他们大多目光游离，极端不自信地回答说："学过，但是……"；另一种是做过几年C语言开发并自我感觉良好的人，他们大多可以使用指针熟练地操作字符数组，但面对菜鸟们提出的诸如"为什么数组的下标是从0而不是从1开始"这类"脑残"问题时，总是不耐烦地回答道："本来就是这样嘛。这是常识，你记住了就行！"（可本来为什么是这样的呢？）

本书的作者不是大学老师，更不是那些没有写过几行程序的学究，而是一位至今还工作在开发一线的程序员（在国内，工作了5年的你如果还在做"码农"，肯定会坐立不安了吧）。他带给大家的不是教科书中死板的说教，而是十多年经验沉淀下来的对无数个"脑残"问题的解答。在这本书初版面世的11年后，我在东京一个大型书店的C语言类别的书架上，依然还能看见这本书被放在一个非常醒目并且触手可及的位置上。

能从书架上挑出本书的人，我想大多都是对C语言指针带有"恐惧感"的程序员吧！其实所谓的"恐惧感"来源于"困惑"，而"困惑"又来自于"对知识点不够透彻的理解"。作者运用幽默风趣并且不失犀利的笔法，从"究竟什么是C语言指针"开始，通过实验一步一步地为我们解释了指针和数组、内存、数据结构的关系，以及指针的常用手法。另外，还通过独特的方式教会我们怎样解读C语言那些让人"纠结"的声明语法。

带着学习的态度，我对原著的每一个章节阅读三次以上后才开始动笔翻译。每次阅读我都会有新的收获，建议购买本书的读者不要读了一遍就将其束之高阁（甚至一遍读不下来就扔到一边）。隔一段时间再来读一遍，收获会更多。

在翻译的过程中，我身边的许多人给了我莫大的支持和鼓励。我的同事的宋岩、王红升在C语言方面都具有10年以上的编程经验，他们经常牺牲个人的休息时间帮我试读译稿，提出了诸多宝贵的意见和建议。开始翻译这本书时，我儿昀好刚出生三个月。新的生命改变了一家的生活状态，带给我们更多的是感动和欢乐。妻子葛亚文在我翻译本书的期间默默承受了产后在身体上和精神上的巨大压力，这不是一句感谢能够回报的。借此祝愿一家——四季有昀，岁月静好！

吴雅明

2012/11/26　于北京

前　言

这是一本关于C语言的数组和指针的书。

一定有很多人感到纳闷："都哪朝哪代了，还出版C语言的书。"

C语言确实是非常陈旧的语言，不过也不可能马上放弃对它的使用。至少在书店里，C语言方面的书籍还是汗牛充栋的，其中专门讲解指针的书也有很多。既然如此，还有必要旧瓶装新酒吗？这才是最应该质疑的吧。

但是，每当我看到那些充斥在书店里的C语言入门书籍，总会怀疑这些书的作者以前根本没有使用C开发过大规模的系统。当然，并不是所有书的作者都这样。

指针被认为是C语言中最大的难点，对它的讲解，很多书都搞得像教科书一样，叙述风格雷同，让人感觉有点装腔作势。就连那些指针的练习题，其中的说明也让人厌倦。

能够炮制出这样的书籍，我想一般都得归功于那些连自己对C语言语法都是一知半解的作者。特别是面对那些在封面上堂堂正正地印上"第 2 类信息处理考试"[①]字样的书，这种感觉更加强烈。

当我还是个菜鸟的时候，也曾对数组和指针的相关语法感到非常"纠结"。

正是抱着"要是那个时候上天能让我遇见这样一本书，那可真帮了大忙"的想法，我写了这本书。

本书的内容，是基于我很久以前（1998年7月）就开始在网上公开的内容：

"深入学习数组和指针"

http://kmaebashi.com/programmer/pointer.html

"当我傻呀？既然可以在网上阅读，我干嘛还买你的书？"我想对有此想法的人说："我敢打包票，**绝不会让你吃亏的**，请放心地拿着这本书去收款台结账吧！"因为此书在出版过程中追加

① 日本国内关于计算机信息处理方面的考试，主要面向计算机系统开发、维护、运用领域的初级技术人员。——译者注

了大量的文字和插图，实际上已经比网上公开的内容丰富了许多。

另外，在阅读本书的过程中，请留心以下几点。

- 本书的读者群虽然定位于“学习过 C 语言，但是在指针的运用上遇到困难”的读者，但还是能随处可见一些高难度的内容。那是因为我也不能免俗，偶尔也喜欢把自己掌握的知识拿出来显摆一下。

  对于初学者，你完全没有必要从头开始阅读。遇到还不太明白的地方，也不要过分纠结。阅读中可以跳跃章节。对于第 0 章和第 1 章，最好还是按顺序阅读。如果认为第 2 章有点难度，你可以先去啃第 3 章。如果第 3 章也不懂，不妨尝试先去阅读第 4 章。这种阅读方式是本书最大的卖点。

- 在本书中，我会经常指出一些“C 的问题点”和“C 的不足”。可能会有一些读者认为我比较讨厌 C 语言。恰恰相反，我认为 C 是一门伟大的开发语言。倒不是因为有“情人眼里出西施”、“能干的坏小子也可爱”这样的理由，毕竟在开发现场那些常年被使用的语言中，C 语言还是有相当实力的。就算是长得不太帅，但论才干，那也是“开发现场的老油条”了。

  所以，因阅读本书而开始抱怨“C 语言真是很差劲”的读者，你即使计划了什么“去揍 Dennis Ritchie[①]之旅”，我也不会去参加的。如果有“去揍 James Gosling[②]之旅”，那还是有点心动的。哈，还是算了吧，得过且过就行啦。

在本书的写作过程中，我得到了很多人的帮助。

繁忙之中阅读大量原稿并指出很多错误的泽田大浦先生、山口修先生、桃井康成先生，指出本书网上公开内容的错误的人们，还有那些受到发布在公司内部的内容的影响而沦为“实验小白鼠”的人们，以及通过 fj.com.lang.c 和各种邮件列表进行讨论并且提供各种信息的人们，正是因为你们，本书的内容才能更加可靠。当然，遗留的错误由我来承担所有责任。

发现我的网页，并给予出版机会的技术评论社的熊谷裕美子小姐，还有给予初次写书的我很多指导的编辑高桥阳先生，如果没有他们的大力协助，这本书是不可能诞生的。

在这里，我谨向他们致以深深的谢意。

2000 年 11 月 28 日 03：33 J.S.T.

前桥和弥

① C 语言之父。本书中对他做了介绍。

② Java 语言之父。

目　录

第0章

本书的目标与结构——引言

0.1 本书的目标

在C语言的学习中，指针的运用被认为是最大的难关。

关于指针的学习，我们经常听到下面这样的建议：

> “如果理解了计算机的内存和地址等概念，指针什么的就简单了。”
> “因为C是低级语言，所以先学习汇编语言比较好。”

果真如此吗？

正如那些C语言入门书籍中提到的那样，变量被保存在内存的“某个地方”。为了标记变量在内存中的具体场所，C语言在内存中给这些场所分配了编号（地址）。因此，大多数运行环境中，所谓的“指针变量”就是指保存变量地址的变量。

到此为止的说明，所有人都应该觉得很简单吧。

理解“指针就是地址”，可能是指针学习的必要条件，但不是充分条件。现在，我们只不过刚刚迈出了“万里长征的第一步”。

如果观察一下菜鸟们实际使用C指针的过程，就会发现他们往往会有如下困惑。

- 声明指针变量 `int *a;`……到这里还挺像样的，可是当将这个变量作为指针使用时，依然悲剧地写成了`*a`。

- 给出 int &a;这样的声明（这里不是指 C++编程）。
- 啥是“指向 int 的指针”？不是说指针就是地址吗？怎么还有“指向 int 的指针”，“指向 char 的指针”，难到它们还有什么不同吗？
- 当学习到“给指针加 1，指针会前进 2 个字节或者 4 个字节”时，你可能会有这种疑问：“不是说指针是地址吗？这种情况下，难道指针不应该是前进 1 个字节吗？”
- scanf()中，在使用%d 的情况下，变量之前需要加上&才能进行传递。为什么在使用%s 的时候，就可以不加&？
- 学习到将数组名赋给指针的时候，将指针和数组完全混为一谈，犯下“将没有分配内存区域的指针当做数组进行访问”或者“将指针赋给数组”这样的错误。

出现以上混乱的情形，并不是因为没有理解“指针就是地址”这样的概念。其实，真正导演这些悲剧的幕后黑手是：

- C 语言奇怪的语法
- 数组和指针之间微妙的兼容性

某些有一定经验的 C 程序员会觉得 C 的声明还是比较奇怪的。当然也有一些人可能并没有这种体会，但或多或少都有过下面的疑问。

- C 的声明中，[]比*的优先级高。因此，char *s[10]这样的声明意为“指向 char 的指针的数组”——搞反了吧？
- 搞不明白 double (*p)[3];和 void (*func)(int a);这样的声明到底应该怎样阅读。
- int *a 中，声明 a 为“指向 int 的指针”。可是表达式中的指针变量前*却代表其他意思。明明是同样的符号，意义为什么不同？
- int *a 和 int a[]在什么情况下可以互换？
- 空的[]可以在什么地方使用，它又代表什么意思呢？

本书的编写就是为了回答以上这样的问题。

很坦白地说，我也是在使用了 C 语言好几年之后，才对 C 的声明语法大彻大悟的。

世间的人们大多不愿意承认自己比别人愚笨，所以总是习惯性地认为“实际上只有极少的人才能够精通C语言指针”，以此安慰一下自己那颗脆弱的心。

例如，你知道下面的事实吗？

- 在引用数组中的元素时，其实 a[i]中的[]和数组毫无关系。
- C 里面不存在多维数组。

如果你在书店里拿起这本书，翻看几页后心想：“什么呀？简直是奇谈怪论！”然后照原样把书轻轻地放回书架。那么你恰恰需要阅读这本书。

有人说：“因为 C 语言是模仿汇编语言的，要想理解指针，就必须理解内存和地址等概念。”你可能会认为：

“指针”是 C 语言所特有的、底层而邪恶的功能。

其实并不是这样的。确实，“C 指针”有着底层而邪恶的一面，但是，它又是构造链表和树等“数据结构”不可缺少的概念。如果没有指针，我们是做不出像样的应用程序的。所以，凡是真正成熟的开发语言，必定会存在指针，如 Pascal、Delphi、Lisp 和 Smalltalk 等，就连 Visual Basic 也存在指针。早期的 Perl 因为没有指针而饱受批评，从版本 5 开始也引入了指针的概念。当然，Java 也是有指针的。很遗憾，世上好像对此还存有根深蒂固的误解。

在本书中，我们将体验如何将指针真正地用于构造数据结构。

“指针”是成熟的编程语言必须具有的概念。

尽管如此，为什么 C 的指针却让人感觉格外地纠结呢？理由就是，C 语言混乱的语法，以及指针和数组之间奇怪的兼容性。

本书旨在阐明 C 语言混乱的语法，不但讲解了“C 特有的指针用法”，还针对和其他语言共有的“普遍的指针用法”进行了论述。

下面，让我们来看一下本书的具体结构。

0.2 目标读者和内容结构

本书的目标读者为：

- 粗略地读过 C 语言的入门书籍，但还是对指针不太理解的人
- 平时能自如地使用 C 语言，但实际对指针理解还不够深入的人

本书由以下内容构成。

- 第 1 章：从基础开始——预备知识和复习
- 第 2 章：做个实验见分晓——C 是怎样使用内存的

- 第3章：揭秘C的语法——它到底是怎么回事
- 第4章：数组和指针的常用用法
- 第5章：数据结构——真正的指针的使用方法
- 第6章：其他——拾遗

第1章和第2章主要面向初学者。从第3章开始的内容，是为那些已经具备一定经验的程序员或者已经读完第1章的初学者准备的。

面向初学者，第1章和第2章从“指针就是地址”这个观点开始讲解。

通过printf()来“亲眼目睹”地址的实际值，应该说，这不失为理解指针的一个非常简单有效的方式。

对于那些“尝试学习了C语言，但是对指针还不太理解”的人来说，通过自己的机器实际地输出指针的值，可以相对简单地领会指针的概念。

首先，在第1章里，针对C语言的发展过程（也就是说，C是怎样“沦为”让人如此畏惧的编程语言的）、指针以及数组进行说明。

对于指针和数组的相互关系，市面上多数的C语言入门书籍只是含混其辞地做了敷衍解释（包括*K&R**，我认为该书是诸恶之源）。这还不算，他们还将本来已经用数组写好的程序，**特地用指针运算的方式重新编写**，还说什么“这样才像**C语言的风格**”。

*K&R*被称为C语言的宝典（中文版叫《C程序设计语言（第2版·新版）》），在后面我们会提及这本书的背景。此书的作者之一就是C语言之父Dennis Ritchie本人。

> 像C语言的风格？也许是可以这么说，但是以此为由炮制出来的难懂的写法，到底好在哪里？哦？执行效率高？为什么？这是真的吗？

产生这些疑问是正常的，并且，**这么想是正确的**。

了解C语言的发展过程，就能理解C为什么会有“指针运算”等这样**奇怪的**功能。

第1章中接下来的内容也许会让初学者纠结，因为我们将开始接触到数组和指针的那些**容易让人混淆的**语法。

第2章讲解了C语言实际上是怎样使用内存的。

在这里同样采用直观的方式将地址输出。请有C运行环境的读者一定亲手输入例程的代码，并且尝试运行。

对于普通的局部变量、函数的参数、static变量、全局变量及字符串常

量（使用" "包围的字符串）等，知晓它们在内存中实际的保存方式，就可以洞察到C语言的各种行为。

遗憾的是，几乎所有使用C语言开发的程序，运行时检查都不是非常严密。一旦出现诸如数组越界操作，就会马上引起“内存区域破坏”之类的错误。虽然很难将这些bug完全消灭，但明白了C如何使用内存之后，至少可以在某种程度上预防这些bug的出现。

第3章讲解了与数组和指针相关的C语言语法。

虽然我多次提到“究竟指针为什么这么难”，但是对于“指针就是地址”这个观点，在理解上倒是非常简单。出现这种现象，其实缘于C语言的数组和指针的语法比较混乱。

乍看上去，C语言的语法比较严谨，实际上也存在很多例外。

对于那些平时和我们朝夕相处的语法，究竟应套用哪条规则？还有，哪些语法需要特殊对待？关于这些，第3章里会做彻底的讲解。

那些自认为是老鸟的读者，可以单独拿出第3章来读一读，看看自己以前是如何上当的。

第4章是实践篇，举例说明数组和指针的常用用法。如果读者理解了这部分内容，对付大部分程序应该不在话下。

老实说，对于已经将C语言使用得像模像样的读者来说，第4章中举出的例子并没有什么新意。但是，其实有些人对这些语法只是一知半解，很多时候只不过是依照以前的代码“照猫画虎”罢了。

阅读完第3章后去读第4章，对于那些已经能够熟练使用的写法，你也会惊呼一声：“原来是这个意思啊！”

第5章中，解说指针真正的用法——数据结构的基本知识。

前四章中的例子，都是围绕C语言展开的。第5章里则会涉及其他语言的指针。

无论使用哪种语言编程，“数据结构”都是最重要的。使用C语言来构造数据结构的时候，结构体和指针功不可没。

“不仅仅对于C语言的指针，连结构体也不太明白”的读者，务必不要错过第5章。

第6章中，对到此为止还没有覆盖到的知识进行拾遗，并且为大家展示一些可能会遇到的陷阱以及惯用语法。

和类似的书籍相比，本书更加注重语法的细节。

提起语法，就有“日本的英语教育不偏重语法”，以至于给人“就算不明白也没有什么”的印象。确实是这样，我们早在不懂“サ行”怎样变形之前，就已经会说日语了。

但是，C语言可不是像日语那样复杂的自然语言，它是一门编程语言。

单纯地通过语法来解释自然语言是非常困难的。比如，日语“いれたてのおちゃ”，利用假名汉字变换程序只能变换成“淹れたてのお茶”（中文意思是‘沏好的茶’），尽管如此，同样通过假名变换程序，“いれたてのあついおちゃ”竟然也可以变换成“入れた手の厚いお茶”（中文意思是‘沏好的热腾腾的茶’）[①]。编程语言最终还是通过人类制订的语法构成的，它要做到让编译器这样的程序能够解释。

“反正大家都是这么写，我也这么写，程序就能跑起来。”

这种想法，让人感觉有点**悲哀**。

我希望不仅是初学者，那些已经积累了一定经验的程序员也能阅读本书。通过深入理解 C 的语法，可以让我们真正领会直到今天还像“口头禅”一样使用的那些程序惯用写法。

无论如何，让我们做到“知其然知其所以然”，这样有利心理健康，不是吗？

① 中文里也有类似的例子，如：他是先知。→他是先知道那件事的人。——译者注

第 1 章

从基础开始——预备知识和复习

1.1 C 是什么样的语言

1.1.1 比喻

在 Donald C. Gause 和 Gerald M. Weinberg 合著的《你的灯亮着吗？》[1]一书*中，有这样一节（根据需要，我做了必要的删减）。

> 某计算机制造商开发了一种新型打印机。
>
> 技术小组在如何保证打印精度的问题上非常苦恼，每次进行新的测试时，工程师都不得不花很长的时间测量打印机的输出结果来追求精确性。
>
> 丹（Dan Daring）是这个小组中最年轻但或许是最聪明的工程师。他发明了一种工具，即每隔 8 英寸就在铝条上嵌上小针。使用这个工具，可以很快地找到打印机输出位置的误差。
>
> 这个发明显著地提高了生产效率，丹的上司非常高兴，提议给丹颁发一个公司的特别奖赏。他从车间里拿了这个工具，带回办公室，这样他写报告的时候还可以仔细地研究一下。
>
> 这个上司显然还用不惯这个工具，当他把这个工具放在桌子上的时候，将针尖朝上了。更不幸的是，当丹的上司的上司友好地坐到桌角上，打算谈谈给丹颁发奖励时，部门内的所有人都听到了他痛苦的尖叫声——他的屁股上被扎了两个相距 8 英寸的孔。

C 语言就恰如这个工具。也就是说，它是一门

* 这本书的副标题为“发现问题的真正所在”，它通过一些趣闻轶事来告诉世人“不要急于寻找问题的答案，而是应该先去考虑当前的问题是什么”。