5元书

知识不贵

只花 5 元就成为

程序语言高手

中科普 5元书系 成都时代出版社 计算机信息与技术丛书之一

图书在版编目（CIP）数据

程序语言高手/李镭著，--成都：成都时代出版社，2003
（计算机信息与技术丛书）
ISBN 7-80548-849-5

Ⅰ.程… Ⅱ.李… Ⅲ.程序语言--程序设计
Ⅳ.TP312

中国版本图书馆CIP数据核字(2003)第022843号

责任编辑： 莫晓涛
特邀编辑： 唐晓曦
封面设计： 聂　培
版式设计： 朱　艳
责任校对： 梅平航

计算机信息与技术丛书

程序语言高手

李镭 兰弘 陶然 胡蓉 **编著**

成都时代出版社出版发行
（成都市庆云南街19号　邮编：610017）
新华书店经销　重庆现代彩色书报印务有限公司印刷
850mm×1168mm　32开　印张：33　700千字
2003年4月第1版　2003年4月第1次印刷
印数：1-20000册
ISBN 7-80548-849-5/TP·1　全套定价(共五册)：25.00元

电话：（028）86619530（综合类）86613762（编辑类）86615250（发行部）

目录

第四章　C 和 C++

第五章 Pascal 和它的父亲

第六章　自由散漫的 Perl

第七章 PHP

前 言

“怎样才能成为一个程序员？学哪种语言最好？怎样快速入门？”很多初学程序的朋友往往爱提出这样的问题。

而另一些已经入门的程序员更爱追问的则是：“我到底是一个 Code Writer，还是一个 Programmer？”

在书店里，关于程序语言的书比比皆是，但是，没有哪一本书告诉你，会写代码的人怎样能成为真正的程序员；也没有哪一本书明确指出如何才能够进入计算机文化的殿堂，并最终成为高手。

本书是一个尝试。这里列举的每个人，几乎都是计算机史上的里程碑式的人物，是程序员文化的某种化身。我们要探讨的，不是具体某种语言、某种系统、甚至某种程序，而是方法和思路，是那些已经成为经典的人物的现身说法，看看他们如何想？如何做？如何去完善自己开发的程序？

大师为什么是大师？天才为什么成为天才？

才华不是最重要的，最重要的是，他们拥有足够的背景知识，所有的语言与软件对他们来说终究是工具。他们超越了具体的技术，用更超然的态度去面对技术，把编程和写代码当作是程序员的散文与诗歌。

这才应该是我们努力的方向。

你手上捧着的这本书，是一本关于“天才”和“技术”的书，同

时，它探讨了计算机时代的发展史以及大师们带给我们的方法论。

虽然，你并不一定因此成为天才和大师，但你至少知道如何成为一个真正的程序员。

第一章

程序文化和历史

本章提要：

自 1945 年 Eckert 和 Mauchily 发明了有史以来第一个计算机系统"ENIAC"(埃尼亚克)之后，计算机技术便吸引了世界上最聪明、最富于创造力的头脑。而这时，充满热情的程序员们，也就是写软件玩软件的人们，或多或少地、自觉或不自觉地逐渐开始形成一种技术性的亚文化。

20 世纪 80 年代，纯程序员(Real Programmer)开始正式登上历史舞台，标志着这种亚文化的成熟。

1.1 程序、程序员及其衍生体

1945 年，Eckert (埃克特) 和 Mauchily (莫契利) 发明了有史以来第一个计算机系统"ENIAC" (埃尼亚克)，计算机技术便吸引了世界上最聪明、最富于创造力的头脑。而这时，充满热情的程序员们，也就是写软件玩软件的人们，或多或少地、自觉或不自觉地逐渐形成一种技术性的亚文化。

20 世纪 80 年代，纯程序员（Real Programmer）开始正式登上历史舞台，标志着这种亚文化的成熟。

所谓“纯程序员”，往往具备工程或者物理方面的学科背景，穿着白袜子和化纤衬衣、戴着厚厚的眼镜，使用机器语言、汇编语言、Fortran 语言，以及一大堆早被人们遗忘的语言编程（在本书以后的章节，将会提到这些语言）。

自从第二次世界大战结束到 20 世纪 70 年代初期，是批处理及打卡计算机与所谓“巨无霸”大型计算机的黄金时代，纯程序员们的技术文化在计算领域独占鳌头。一些被后人仰视的经典程序员就在此时奠定了基础，还有各种各样的墨菲定律（比如，当有两条路让你抉择，若其中一条会导致失败，必定有人会选到它。就如一个程式在 demo 前测试几千几万次都正确无误，但 demo 哪一天偏偏就会出 Bug）也大行其道起来。

还有些纯程序员至今仍相当活跃。据说 Seymour Cray（Cray 系列超级计算机的创始人）有一次干过这么一桩活儿：他单枪匹马用八进制代码写了一个操作系统，又把它加载到自己鼓捣出来的硬件上。调试一次成功，运作良好，没有 Bug。

随着单独个体的 Real Programmer 的时代步入尾声，取而代之的是逐渐盛行的交互计算以及网络的时代，这一时代的到来，把程序员的影响力大大推前了一步，从程序员的技术文化中逐渐衍生出电话飞客、电脑黑客的破解客等等亚亚文化。

这新的一批程序员，是在垮掉及一代，嬉皮士、摇滚乐还有车库文化氛围中成长起来的一代人。如果看到有程序员留着大胡子

穿着喇叭裤听摇滚乐你可千万别奇怪。他们向往的，不单单是技术，更多的是对技术所能达到的自由的追求。这些家伙在 Unix 系统捣鼓出了 E-mail，也捣鼓出了蠕虫病毒；能弄一套 Linux 跟微软对着干，也能通过电话线上对 FBI 的资料库进行修改。

无论是技术，还是自由，都是双刃剑。在这一代程序员身上，纯程序员那种严谨消失了一点，取而代之的是更加狂放、激烈的热情和干劲，虽然有时候会干出些令人吃惊的坏事来——不过，你可别小瞧了他们，现代计算机应用技术的 90%以上，是从这些人的脑袋里蹦蹬出来的。

是他们写出一套又一套的程序语言，一个又一个的操作系统，一款又一款的应用软件。

他们是英雄吗？是的。

他们创造的不仅仅是技术，还是新的集体文化和思想意识。这也是程序员文化中的最紧要的一环。

1.2 程序语言简史

有了计算机，就得有程序语言。早期的计算机，每一台都有自己独立的操作系统，同时配有自己特殊的程序语言。

1822 年，查尔斯·巴巴奇 (Charles Babbage) 发明了他的差分机 (difference engine)，从那时起，就必须给计算机找到一种指示它们完成特定任务的方法。这种方法，现在我们把它叫做“程序语

言”。

所谓“计算机语言”是这样工作的：先由一系列步骤组成一个特别的程序，然后将这些程序输入计算机并让它照此执行。随着以后的发展，计算机语言又提出了更进一步的要求，比如逻辑分叉、面向对象等特征。

一开始，查尔斯·巴巴奇的差分机仅能执行计算齿轮的任务。也就是说，最早的计算机语言的形式是物理运动。

直到 1942 年美国政府制造出 ENIAC 之后，物理运动才被电子信号所取代，它遵循了巴巴奇引擎的许多原则。因此，ENIAC 要执行新的计算指令时，只能预先把“规划”预置到系统内才行。整个过程自然相当乏味。

1945 年，约翰·文·纽曼 (John Von Neumann) 在 Institute for Advanced Study 工作时，提出了直接影响计算机程序语言发展方向的两个重要概念。

第一个概念是“程序分享技术”。这一技术指出，实际的计算机硬件应该非常简单，不需要做相应的手工设置来针对每个程序。而复杂的指令应该被用来控制简单的硬件，让它能更快的运算。

第二个概念对程序语言的开发也极其重要，纽曼叫它“有条件的控制传输”。这个想法产生了子程序的概念，小块的代码可以在任意顺序中执行，无须设定为单一顺序步骤；其次，计算机代码应该可以基于逻辑语言表达式分叉 (如 IF…Then 语句) 和循环 (如 For 语句)。

“有条件的控制传输”催生了“库” (Libraries) 的概念，所谓

“库”，即能被重复使用的代码。

1949 年，“短码”（Short Code）语言出现。它是电子设备的第一个计算机语言，它的要求是程序员手工把代码全部转成 0 和 1。虽然难度很大，却迈出了通往今天复杂程序语言的第一步。

1951 年，格蕾丝·霍波（Grace Hooper）写出第一个程序语言编译器 A-0。编译器就是把计算机语言的语句转换为 0 和 1 的程序。这使得程序员们编程的速度大大加快，而且无需手工转换。

1957 年，第一个真正的计算机语言出现了，名叫 FORTRAN。这个名字是 FORmula TRANslating 的缩写，它是专为 IBM 公司设计的。现在看来，其组件非常简单，却向程序员提供了底层数据的存取功能。

这个语言仅仅包括“IF”“DO”“GOTO”语句，以大家现在的眼光来看是非常简陋和有局限性的，但在当时却是了不起的大进步。

今天还在使用的基本数据类型（包括逻辑变量 TRUE 和 FALSE），是 FORTRAN 里首先提出来的。

尽管 FORTRAN 非常擅长数字的处理，但对输入输出的支持并不太好，偏偏输入输出又和大多数的商业计算息息相关。1959 年，当商业计算飞速发展起来之后，COBOL 语言诞生了。COBOL 的最初设计理念就是成为商业使用的基础语言，它的数据类型只包括数字和字符串。为了能更好地跟踪和组织数据，数字和字符串便被组织到数组和记录中。COBOL 程序就像散文那么优雅，由四到五节组成一章，它又像英语一样有语法，学习起来非常容易。它

所有的设计都是为了使用户能更好更容易地接受和学习。

1958 年，为了研究人工智能 (AI)，麻省理工学院的约翰·麦卡锡 (John McCarthy) 设计了 LISt Processing (也叫 LISP) 语言。因为它针对的领域非常专业，所以它的句法相当少见。这种语言和别的语言最显著的差别是，它只支持“表”这种数据类型。

LISP 语言是把自己也当成一套表来描述，因此它具有自我完善的功能，也由此发展了起来。

LISP 的句法以所谓“Cambridge Polish”而著称，它与标准的布尔逻辑表达式完全不同。

具体区别如下：

x V y ——Cambridge Polish 句法，用来描述 LISP 程序。

OR(x,y) —— LISP 语言使用的逻辑表达式。

x OR y ——标准布尔逻辑表达式。

由于 LISP 的高度专业化，所以一直流传至今。

1958 年，一个科学委员会开始开发 Algol 语言。它的突出贡献是为类似 Pascal、C、C++、以及 Java 这类的语言奠定了基础。它也是第一个有语法形式的计算机语言，比如大家知道的 Backus-Near 以及 BNF。

Algol 提出了一些很新奇的概念 (如递归功能)，造成新版本的 Algol68 异常庞大，根本无法使用。于是体积更小结构更紧凑的语言 Pascal 出现了。

Pascal 是由尼克劳斯·维斯 (Niklaus Wirth) 在 1968 年提出

的。维斯开始只是想设计一个好的教学语言，没想过要让它为世人广泛接受。因此，他集中精力设计了调试器（Debugger）和编辑器，还把它设计成支持一种在当时教学机构内使用广泛的老古董微处理器机。

Pascal 整合了当时通用语言 Cobol、Fortran、Algol 的最佳特点，在整合的同时，整理清除了这些语言的不规则陈述。因此，大量的用户开始使用 Pascal，让它逐渐发展成一种非常成功的语言。

Pascal 改进了“Pointer”，增加了“CASE”语句，允许指令执行它后以树形方式分支。

例如：

```
CASE expression OF
        possible-expression-value-1:
                statements to execute...
        possible-expression-value-2:
                statements to execute...
END
```

Pascal 改进了动态变量，让它可以在程序执行时通过“NEW”和“DISPOSE”命令出现。但是，Pascal 没有实现动态数组和变量组的功能，这导致了它后来的衰落。

维斯后来又开发了一个 Pascal 的继承者——Modula-2，可惜 Modula-2 生不逢时，C 语言已经把市场都占领了。

1972 年，新泽西贝尔实验室。

丹尼斯·里奇（Dennis Ritchie）开发了 C 语言。C 语言的直系

父母应该是 B 和 BCPL 语言，但它和 Pascal 之间的继承关系也显而易见。Pascal 的所有特性（比如“CASE”语句），都能在 C 里执行。但是 C 修正了 Pascal 的错误，立刻就赢得了前 Pascal 用户们的心。

里奇是为当时同步开发的 Unix 系统而设计的 C 语言，因此，C 和 Unix 是一对儿哥俩。

C 有很多先进的特点，比如动态变量、多任务处理、中断处理、Forking 等，而且它在输入输出上也很强。在 Unix、Windows、MacOS、Linux 系统下，C 的应用都非常普遍。

到了 20 世纪 70 年代末 80 年代初，一种新的编程方法，也就是面向对象编程（Object Oriented Programming 或 OOP）被设计出来。

所谓对象，是程序员能包装和能操作的数据块。

布贾恩·斯特斯特普（Bjarne Stroustroup）很欣赏这个方法，把它扩展到了 C 语言上，这也就是“C With Classes”的由来。

1983 年，C++带着全部的新特性，正式闪亮登场。

C++的设计思路是，把 OOP 融合到 C 语言上，同时维持 C 的运算速度，能在不同的计算机上运行。C++常用于模拟运算，比如追踪升降机内上百人的操作或者监控军团里不同类型士兵行为，所以，它至今还是美国 AP Computer Science 必修的语言。

20 世纪 90 年代早期，交互电视机技术看起来很有前途，为了配合这种交互技术，Sun 公司决定开发一个便携式跨平台的语言。这语言最终发展成了 Java。

1994 年，交互式电视项目失败，Java 开发组把重点转到网络应用上。第二年，Netscape 浏览器内支持 Java 程序。这时 Java 变成了“未来的语言”，若干公司都宣布自己的应用程序将用 Java 开发（不过尚没有一个应用程序开发完毕）。

从某种角度来说，尽管 Java 的目标高远，而且是一种好语言的样本，但很可能它会变成一种“不是语言”的语言。它的代码优化有很多问题，用它编写的程序执行速度太慢。而且，Sun 公司和微软之间的商业斗争，严重阻碍了 Java 被人们接受的速度。

但 Java 的跨平台思路、真正地面向对象工作、垃圾代码收集等等特性，必然会为未来的程序语言好好上一课。

1964 年，约翰·凯门伊（John Kemeny）和托马斯·克茨（Thomas Kurtz）开发了 BASIC 语言，随后在此基础上改进的 VB，目前已经是软件教学中最常用的语言。

BASIC 语言是为没有计算机基础的人设计的语言，能力非常有限。微软把 BASIC 扩充成了自己的主打产品 VB。VB 的核心是 FORM（表单），能在空白窗口上拖动部件、滚动窗、图像。这些小东西的学名叫“窗口部件”（Widgets）。Widgets 中有属性（如颜色）和事件（点击，或者双击），能够开发任何用户所需的界面。

VB 现在常用来为微软的其他产品（如 Excel、Access）开发简单快速的接口界面，当然用它也能开发完善的应用程序。

Perl 是互联网上应用最广的脚本和控制语言，因为它有很强的文本匹配功能，由拉里·沃尔 1987 年在 Unix 系统下开发。

就这样，计算机语言走过了这么个历程，开始为最简单的功能