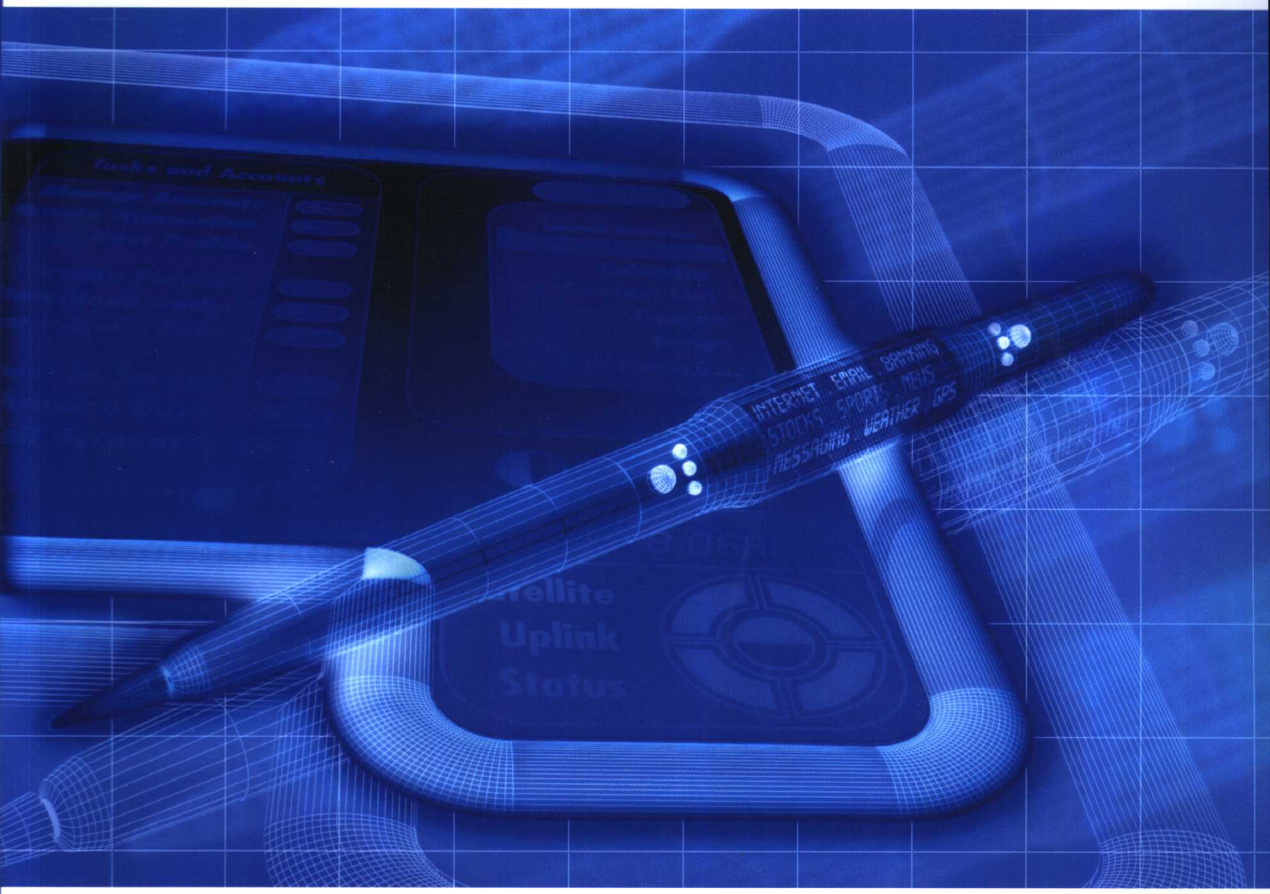
21世纪高等院校教材

C语言程序设计

赵永哲 李雄飞 戴秀英 编著



科学出版社
www.sciencep.com

21世纪高等院校教材

C语言程序设计

赵永哲　李雄飞　戴秀英　编著

科学出版社
北京

内 容 简 介

本书系统地介绍了标准C语言的基本构成、语法规则及C程序的编辑、编译和执行过程，并针对初学者的特点，由浅入深，从一般程序设计语言的共性到C语言自身的特性，从C语言的语法规则到其内部实现，对C语言进行了系统介绍。

对计算机专业或已学过其他高级程序设计语言的学生，本书多从语言实现的角度来阐明C语言所特有的一些属性，使学生不但能知其然，还能知其所以然，以便举一反三，熟练地应用C语言。此外，本书还对最新的面向对象的程序设计技术做了相应的介绍，并尽可能从C语言的角度来模拟和实现有关的技术。这样，既能消除读者对相应技术和名词的神秘感，同时也便于其更深刻地理解C语言。本书实例丰富，每章之后附有习题。

本书可作为大专院校理工科各专业的教材，也适于自学者参考。

图书在版编目（CIP）数据

C语言程序设计/赵永哲，李雄飞，戴秀英编著. —北京：科学出版社，2003

(21世纪高等院校教材)

ISBN 7-03-012100-7

Ⅰ.C… Ⅱ.①赵… ②李… ③戴… Ⅲ.C语言-程序设计-高等学校-教材 Ⅳ.TP312

中国版本图书馆CIP数据核字（2003）第075037号

责任编辑：马长芳 匡 敏 / 文案编辑：董 斌 / 责任校对：宋玲玲

责任印制：安春生 / 封面设计：陈 敬

科学出版社 出版

北京东黄城根北街16号

邮政编码:100717

http://www.sciencep.com

西源印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2003年9月第 一 版 开本：B5（720×1000）

2003年9月第一次印刷 印张：19 1/4

印数：1—2 500 字数：372 000

定价：25.00元

（如有印装质量问题，我社负责调换〈新欣〉）

前　言

C 语言是目前运用最为广泛的程序设计语言。和其他语言相比，C 语言素以代码效率高、功能丰富、程序精练和可移植性好而著称。由于它既具有高级语言面向问题的特点,又具有低级语言面向机器的特点，因此既可以用其取代汇编语言来编写系统软件，也可以用来编写应用软件。

C 语言所取得的成就要归功于其创建者 D.Ritchie，他创建 C 语言的初衷便是要为程序员提供最大程度的设计自由度。这使得 C 程序员能充分展示自己的创作才华，并从编程中得到最大的满足和成就感。这对使用过像 BASIC、FORTRAN 这样的强语法限制语言而后改用 C 的程序员来说，感受尤为深刻，而且往往在用过 C 语言之后便再也无法割舍。

鉴于 C 语言在计算机应用和软件开发中的重要地位，目前许多高等院校已将 C 语言从专业基础课改为公共基础课。但令人诧异的是，目前仍有相当数量从事计算机专业的人员未能掌握 C 语言，甚至从未接触过 C 语言。作者认为，很多人之所以学不会 C 语言，主要是心理作祟。他们往往在道听途说中认为 C 语言难学，便产生了畏难的心理。这也难怪，因为这些言论往往出自于已经掌握 C 语言的人之口，所以对 C 语言的初学者具有极大的杀伤力。

就作者学习和教授各种程序设计语言的体会而言，C 语言实际上是最容易掌握的一种语言，这是因为 C 语言对语法的限制最少。之所以有人会说 C 语言难学，是因为其将“掌握”与“精通”混为一谈。就 C 语言本身而言，掌握它是一件很容易的事，只需记住其 20 多个关键字的用法和一些必要的运算符及语法规则，便可以借助 C 库函数手册来编写 C 语言程序。这个过程对本书的自学者来说两个月便足够了。但要说到精通某一种语言，则是一件永无止境的事情，且这已经超出了语言本身的范畴，因为这需要具备许多语言之外的知识。但作者相信，读者如果能将本书认真地读完，就能感悟出下一步应如何去做。

作者写此书的目的有二：一是要让初学者掌握 C 语言，能使用 C 语言来编写一些复杂的算法和简单的实用程序；二是通过对 C 语言的学习，使读者对现代程序设计语言的共性和实现手段有一个感性的认识，以达到举一反三，从而更快掌握其他程序设计语言，并理解有关新技术的实现机理。出于此目的，作者没有单纯从语法规则的角度对 C 语言做面面俱到的介绍，而是侧重于从 C 语言的设计和实现者的角度来对 C 语言的一些具体特性进行剖析，并结合大量的简单实例，力求使初学者能尽快地入门。

关于 C 语言的书籍已经很多，想快速学会并掌握 C 语言的人更是愈来愈多，

教师的使命感促使作者提起笔来。虽然不知本书是否能达到作者的初衷，但至少感觉自己正在做一件有意义的工作 —— 尝试教会大家一种重要的语言！

作　者

2003 年 7 月于吉林大学

目　录

第1章 概 论

1.1 C语言简介

C语言是一种程序设计语言（programming language），也是目前国际上最流行的程序设计语言。C语言是1972年前后在美国的贝尔实验室（Bell Laboratories）开发出来的，其设计和发表都是由D. Ritchie一个人完成的，当时他正和Ken Thompson一起从事UNIX操作系统的研发工作。

最早的UNIX操作系统是D.Ritchie和K.Thompson两人从1969年开始，用不到两个人年的时间研发出来的。当时整个UNIX系统都是用汇编语言编写的。由于汇编语言不可移植，并且易读性差，K.Thompson便决定开发一种更高级的语言来描述UNIX系统。1970年，K.Thompson在PDP-11/20机器上开发出了B语言，并用B语言重写了UNIX系统和绝大多数的实用程序。B语言的主要思想源于BCPL语言（basic combined programming language）。BCPL语言是剑桥大学的M.Richards基于CPL语言（combined programming language）在1967年发表的一种语言。但BCPL和B语言在设计上过于偏重硬件实现，从而使该语言在功能和使用上受到了极大的限制，故它们并没有像设计者期望的那样流行开来。这导致D.Ritchie在1971年着手进行C语言的开发工作。

D.Ritchie在C语言的设计过程中，既保持了BCPL和B语言“面向机器”的特征，同时又克服了它们的缺点（过于简单，数据无类型等）。同时，D.Ritchie极力避免使C语言成为一种“大而全”的语言，和BCPL及B语言一样，C语言也把重点放在“小而精”上，不要求C语言能面面俱到。事实证明，这样做丝毫没有妨碍C语言处理复杂问题的能力。恰恰相反，用一些简单的程序单元共同构成复杂的程序正是C语言的一个超常之处。

至于D.Ritchie为什么用“C”来命名C语言，有两种说法，一是由于C语言是在B语言之后开发的，故按英文字母的次序命名为C；二是由于C语言和B语言均起源于BCPL语言，故依次取BCPL的每个字母来对后续语言命名。

由个人所设计和实现的语言往往在某些方面反映出其创作者的专长。D.Ritchie的特长表现在系统软件、计算机语言、操作系统、程序生成和文字处理等方面，故用C语言书写这些方面的软件是非常高效的。

1973年，K.Thompson和D.Ritchie两人合作将UNIX操作系统用C语言重编（即UNIX V5）。系统的代码量比以前的版本大了1/3，加进了多道程序设计功能，

特别是整个 UNIX 操作系统（包括C编译程序本身）都建立在C语言的基础之上，所以 UNIX V5 便奠定了 UNIX 操作系统的基础。此后，UNIX 迅速在 Bell Labs 普及开来并毫不费力地为全美大多数高校所认可。C语言也取代了以往大家熟悉的 UNIX 上的语言而成为贝尔实验室的“官方语言”。此后，C语言的名声大振，用户日增，愈来愈多的程序员抛弃了其旧爱（如 FORTRAN 和 PL/I）甚至新宠（如 PASCAL 和 APL）而转投到C语言的怀抱。这种情况令 D.Ritchie 感到吃惊。这可能和C语言的特性有关，因为C语言从属于一个优秀的“语言家族”，该语言家族的传统着重于可靠性（reliability）、规范性（regularity）、简单性（simplicity）和易用性（easy of use）。这个语言家族的成员通常被称为“结构化语言（structured languages）”，这是因为它们非常适用于“结构化程序设计”，其中最规范的程序设计语言当属 PASCAL。C语言的家谱如下：

ALGOL 60（designed by an international committee, 1960）

↓

CPL（Cambridge and the University of London, 1963）

↓

BCPL（Martin Richards, Cambridge, 1967）

↓

B（Ken Thompson, Bell Labs, 1970）

↓

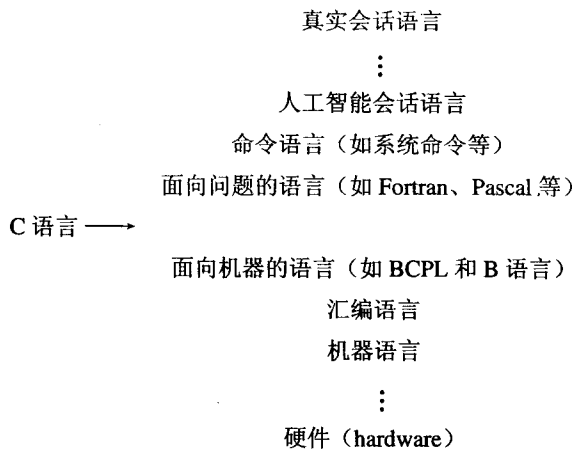
C（Dennis Richie, Bell Labs, 1972）

从某种意义上说，没有C语言就没有UNIX今天的巨大成功。当然，没有UNIX系统，C语言也不会有现在这样的知名度。C语言和 UNIX 可以说是一对孪生兄弟，在发展的过程中相辅相成。虽然 D.Ritchie 设计C语言的初衷是为了用其重写 UNIX 系统，但自 1978 年以后，C语言已先后移植到大、中、小、微型机上，并已不再局限于 UNIX 系统。现在C语言已风靡全球，成为世界上应用最广泛的几种计算机语言之一。

1.2 C语言在计算机语言体系中的地位

C语言特别适用于编制系统软件。这主要有两个原因：首先，C是一种相对“低级”的程序设计语言，这就使得程序员可以对许多具体的机器细节进行描述、控制和处理，从而能最大限度地发挥计算机的效率；另外，C语言又是一种相对“高级”的程序设计语言，它屏蔽了许多具体的机器特性，使得具体的计算机对程序员来说是“透明”的，从而能极大地提高编程效率。我们可以通过下面的层

次结构来说明C语言在程序设计语言体系中的地位：



从下往上来看，表现为从“具体”到“抽象”；从“巨细”到“概括”；从“面向机器”到“面向人”，这也意味着计算机语言的“过去”到“未来”。可能有一天人们可以通过和机器的自然会话来“编程”，但这对现在的技术水平来说还有一段漫长的路要走。

我们通常称面向问题的语言为“高级语言”，而称面向机器的语言为“低级语言”。C语言恰好介于这两类语言之间，故它足够地“接近”硬件，便于程序员对具体的硬件进行控制。同时它又适当地“远离”硬件，使得程序员在编程时可以无视具体的硬件细节。这一点是高级语言和低级语言所不及的，也是许多软件开发人员对C语言情有独钟的原因之一。

1.3 C语言的特点

C语言的主要特点如下：

1. 语言简洁、紧凑、使用灵活

C语言一共只有32个保留字（关键字），除了在表示方法上尽可能简洁（比如，以{、}代替通常的begin、end做复合语句、运算符缩写等）以外，语言的许多成分都是通过显式的函数调用来完成。特别是许多和机器硬件相关的操作（比如I/O操作等）都一律交给函数来完成，这样就使得C编译和硬件的相关性很小，故C语言编译程序的体积很小。另外，C语言是一种自由格式的语言，没有像FORTRAN语言那样的书写格式限制，故用C语言书写程序可以随心所欲，自由方便。

2. 运算符丰富

C语言的运算符种类很多，共有34种运算符（见书后附录）。它可以进行字符、数字、地址、位等运算，并可完成通常由硬件实现的普通算术、逻辑运算。灵活使用各种运算符可以完成许多在其他高级语言中难以实现的运算或操作。

3. 具有数据类型构造能力

C语言可以在基本数据类型（如字符型、整型、浮点型等）的基础上按层次结构构成各种更复杂的数据类型。

4. 具有很强的流控制结构

C语言的各种控制语句（如if、while、do-while、for、switch等）功能很强，便于书写结构优良的程序。

5. 语言生成的代码质量高

高级语言是否适用于编写系统软件，除了语言表达能力之外，还有一个很大的因素是该语言的代码质量。如果代码质量低，则系统开销就会增大。一般说来，语言越低级其代码质量就越高，但编程的工作量也越大。故用汇编语言编程的代码质量最高。许多实验表明，针对同一个问题，用C语言描述，其代码效率只比汇编语言低10%～20%。故现在许多系统软件都用C语言来描述，从而大大提高了编程效率。

6. 可移植性较好

可移植性是指程序可以从一个环境不加或稍加改动就可搬到另一个完全不同的环境上运行。对汇编语言而言，由于它只面向特定的机器，故不可移植。而一些高级语言（比如FORTRAN）其编译程序（compiler）也不可移植，而只能根据国际标准重新实现。但C语言在许多机器上的实现是通过将C编译程序移植得到的。据统计，不同机器上的C编译程序80%的代码是共同的。

7. 语法限制不够严格，程序设计自由度大

用C语言所编写的程序的正确性和合法性在很大程度上要由程序员而不是由C编译程序来保证。如C编译程序对数组下标不做越界检查，类型检验功能较弱且转换比较随便等。故对C语言不熟悉的人员，编写一个正确的C语言程序可能会比编写其他高级语言程序难一些。所以用C语言编写程序，要对程序进行认真的检查，而不要过分依赖C编译程序的查错功能。正是因为C语言放宽了语法的限制，所以换来了程序设计的较大自由度和灵活性。使用得好，你便会体会到

这非但不是C语言的缺点，而恰恰是其一大优点。

8. 允许直接访问物理地址

用C语言能进行位（bit）操作，能实现汇编语言的大部分功能，可以直接对硬件进行操作。因此，C语言既具有高级语言的功能，又具有低级语言的许多功能，可用来写系统软件。C语言的这种双重性，使它既是成功的系统描述语言，又是通用的程序设计语言。

9. 支持程序注释

可以用 / * … * / 对C程序中的任何部分作注释，增加C程序的可读性。

10. 模块化的程序结构

C语言的程序是由函数构成的，一个函数既为一个“程序模块”。一个C语言源程序至少包含一个 main 函数，也可以包含一个 main 函数和若干其他函数。C程序总是从 main 函数开始执行的，而不论 main 函数在整个程序中的位置如何。因此，函数是C程序的基本单位。同时，系统也提供了丰富的库函数，用户可以在程序中直接引用相应的库函数，也可根据需要编制和设计自己的函数。

以上我们粗略地介绍了C语言的一些突出特点，至于具体细节将在以后的各章节内容中介绍。对初学者来说，C语言的这些特点现在可能体会不出来，但待学完整个课程之后，相信必会对其有一个比较深刻的认识。为了让读者对程序设计语言有一个比较全面的认识，下面将暂时抛开C语言而转对一般程序设计语言的共性和基本实现方法进行相应的介绍。

1.4 程序设计语言

从广义和逻辑上说，任何一种描述算法和数据结构的“语言”都可称为是程序设计语言，这些语言有的已在某种型号的计算机上实现了，而另一些到目前为止还没有在任何型号的计算机上实现。从狭义上，可以把程序设计语言理解为目前已经在某种型号的计算机上实现了的“语言”，比如 FORTRAN、ALGOL、COBOL、BASIC、C 和 PASCAL 等等。

目前已经设计和实现了的程序设计语言有几百种之多，常用的也有几十种，而且，几乎所有型号的计算机，不论其大小，都有它独特的语言。为了更深刻地了解和学习程序设计语言，我们有必要从整体上对它们的特性进行一下分析。

1.4.1 程序设计语言的主要构成

程序设计语言主要由以下几个部分构成：

（1）数据——程序设计语言必须提供用于加工的各种类型的初等数据项和数据结构。

（2）运算——程序设计语言必须为其各种类型的数据提供一组有效的基本运算。在程序设计语言中，运算与数据是不可分割的两个成分。数据作为被动的成分，是存储的信息；而运算是主动的成分，它生成、废弃和变换数据。数据和运算通过控制结构以一定的方法依次结合起来，以便在适当的时侯，对适当的数据进行适当的运算。

（3）控制结构——程序设计语言必须为一系列被执行的基本运算提供控制执行顺序的机构。任何一个程序，不管使用那种语言，都可以看成是一系列特定的运算，并且这些运算是使用一定的数据，按照一定的顺序执行的。各种程序设计语言的主要区别就在于语言所提供的数据类型、运算类型以及控制运算顺序的机构有所不同。所以学习任何一种程序设计语言都要紧紧围绕这三个方面来学。

（4）存储管理——对于程序设计语言的设计和实现者来说，存储管理是他们共同关心的首要问题之一。因为程序流程的控制和数据表示都离不开存储分配，而且语言的某些特征也影响到存储管理技术，如允许递归的语言就必须采用动态存储管理技术。虽然某种程序设计语言采用何种存储管理技术以及所涉及的细节和硬、软件表示都是实现者的事情，但对于程序员来说，了解所用语言的存储管理技术也是很重要的。这使得程序员可以通过直接的方式（如 FORTRAN 的 EQUIVALENCE 和 C 中的 union 和 auto 变量等）或间接的方式（如表达式的优化组合、参数值采用地址传递等）来控制和影响存储管理，从而更高效率地使用存储器和提高程序的运行效率。

（5）运算环境——程序设计语言应提供内部和外部交换信息的机构。在程序内部，全局变量、局部变量、子程序及参数传递等构成了信息的内部运算环境。此外，程序设计语言中还应提供和外部的输入/输出设备进行信息交换的运算环境。这往往表现在一些和此有关的 I/O 指令或 I/O 函数。这一点对程序设计语言来说也是很重要的，因为任何一个实用的程序都必须和外界交换某些信息和产生一些输出结果。

1.4.2 程序设计语言的实现

对于计算机来说，它只能识别和执行机器语言程序。然而，程序的编写通常总是采用比机器语言更高级的程序设计语言。虽然从理论上可以设计一个硬件计算机直接地执行以任何具体的程序设计语言所编写的程序，但是建造这样的计算机是很不经济的。基于速度、灵活性和成本的实际考虑，人们仍然倾向于利用低级机器语言的实际计算机。因此，我们便面临着语言的实现问题，就是如何能够从高级语言程序得到能在实际计算机上执行的程序，而不论实际计算机的机器语言为何。对这个问题的解决办法主要有两种：