# C 语言教程

孙玉芳　孟其昌　编著

C 语言

C 语言

C 语言

北京希望电脑公司

# C 语 言 教 程

孙玉芳　孟其昌　编著

北京希望电脑公司
一九九二年一月重印

★北京市新闻出版局
准印证号：891169
★订购单位：北京8721信箱资料部
★电　　话：2562329
★电　　传：01-2561057
★地　　址：海淀影剧院北侧
☆乘　　车：320、332、302路至海淀黄庄下车
★办公地点：希望公司大楼101房间
★邮　　码：100080

# 前　言

十来年前UNIX和C还只是一个鲜为人知的实验室产品，现在正在成为众口交誉的优秀软件系统。由于它们的巨大成功，其主要作者，K.Thompson和D.M.Ritchie获得了1982年度Electronics最高成就奖，这是该奖设立以来第一次授予了在软件领域里作出杰出贡献的科学家。有关评论指出，他们的工作为程序设计的高生产率扫清了道路。继之，他俩又被授予1983年度计算机科学最高荣誉奖——图灵奖。评委会指出，UNIX系统作为一个样板已经引导一代软件设计者去思考程序设计的新途径。

毫无疑问，UNIX和C巨大的商品价值和显要的学术地位将吸引更多的人去研究和开发这个系统。UNIX系统是一个基本的软件环境，其中包含有众多的软件工具，而最基本的工具就是C程序设计语言。

C语言最早是用来描述UNIX操作系统及其上层软件的。但目前其应用已不只限于UNIX系统，而在很多不同种类的操作系统、很多不同类型的计算机硬件上实现了，从8位微型机到最高档的超高速机Cray—Ⅱ，遍及微、小、中、大型机。尽管它在精确和严密性上不及某些语言（如Pascal、Algol 68），但由于它具有简洁，实用，代码质量高，特别是可移植性好等特点，所以受到计算机用户特别是软件工程技术人员的赞赏，并得到越来越广泛的应用。

随着UNIX系统的推广，C语言在系统软件、软件工程、软件工具和应用方面的独特地位也越来越为人们所认识。并注重培养这方面的人材，

1983年12月，IEEE计算机学会公布了示范教学大纲，其中提到C语言的次数最多。虽然不能以此就断言C语言更重要，但确实反映了该教学大纲对C的重视。

由于人们对C的认识始于对UNIX系统的认识，所以即使是在C的诞生地美国，近几年也才展开对C的广泛而深入的研究，故前几年有关C语言的专著甚少。

我国教育界已决定以UNIX操作系统为背景编写有关方面的教材，来讲述操作系统原理。有不少大学和研究生院也以UNIX系统作为软件工具和环境、系统开发等课程的参考系统。此外，尚有不少研究单位和大学也都围绕UNIX系统进行研究和开发。这些都对C的研究和应用提出了新的要求。为此，许多高校和计算机培训班开设了C语言程序设计课程。

这些年我们在学习、研究UNIX系统的同时，在部分学校和有关讲习班里多次讲述了UNIX核心结构、UNIX操作系统原理及UNIX的使用等内容。深感熟悉C语言对研究和开发UNIX的重要性。为此我们曾在讲习班上或多或少地多次介绍过C语言，但是手上仍缺乏一本较系统完整的教材。为了提高教学质量，帮助学生和广大计算机用户更好地使用和开发UNIX系统，我们参照有关资料，结合自己讲授和使用C语言的体会，把历次讲授的手稿加以整理，写成了本书。

本书初稿是在1984年初以油印本的方式公诸于众，并以此作为基本教材开设了几期学习班，也曾作为大学本科生的试用教材进行讲授。在近一年的时间里，我们听取了学生和学员以及一些专家的批评建议，作了几次较大的改动。

在编写和修改中，我们力求做到以下几点

第一，说理与实例演示并重。以实例作为说理的基础，促进读者对一些概念的直观理解；而以说理来提炼归纳现象，促进读者把实例中的现象提到理性认识的高度。选取的实例力求能全面地代表各种类型：基本型的目的在于帮助读者理解概念；技巧型则侧重于程序设计方法的训练上，使学员能针对问题用C进行描述；实用型的目的则是帮助读者能综合运用所学的知识来理解和解决比较复杂的实际问题（所有实例均已上机验证过）。第二，论述体系的一致性和科学性。C语言应用十分广泛，但至今还没有严格的形式化定义。由于笔者一直在进行C形式的研究工作，所以本书即以语法图的形式向读者介绍C的各个主要部分和功能。这种叙述方法使本教材体系划一、风格协调。第三，在内容上的实出重点和兼顾系统完整性。作为一种语言，C的许多方面是与一般高级程序设计语言类同的，考虑到系统完整性，我们对此都作了介绍。但是，叙述的重点是放在C独有而又十分重要的一些特性上，尽可能做到详略得当。

本书共分七章，后面有若干个附录。

第一章简单叙述了C语言的发展历史、特点，并且结合简单例子对C语言的编辑、编译和运行全过程作了概括的介绍，目的是让读者熟悉C语言运行环境，以便在后面的学习中通过上机实习来加深对C语言的了解。

第二章介绍了C语言的基本概念和成分：标识符、变量、常量、数据及变量的简单类型。表达式是语言的基本组成部分，故本章中对各种表达式都作了介绍。在表达式基础上介绍了一种最简单的基本语句即赋值语句。最后介绍了C语言众多的运算符，给出其优先顺序。

第三章介绍C语言主要的语句和控制流：条件语句、开关语句、各种转移语句。

第四章介绍构成C程序的主要成分——函数，围绕函数介绍了函数结构、函数的返回类型、参数和递归调用。联系到函数，我们介绍了C语言特有的变量和函数存储类。最后介绍了C的程序结构和预处理程序。

除了第二章介绍的几种基本数据类型外，第五、六两章专门介绍几种复杂的数据类型。第五章介绍指针和数组。主要包括它们的定义、使用，它们之间及它们与别的语言成分（函数、其它类型的数据）之间的关系。第六章重点介绍结构、联合、枚举和类型定义。这涉及到结构的组成、其成分的访问以及联合的定义、联合与结构的不同之点等。另外还简介了枚举的意义及其使用方法，最后讲了类型定义的本质。

最后一章介绍的内容虽然不是C语言的成分，但是却是使用C语言完成工作所必需的。这主要包括输入输出的实现及使用提供的库函数、系统调用进行程序设计的方法。

学习语言的最好方法是实践，所以我们在书中给出了不少实例，各章后还有许多习题，可供读者上机实习。

附录A列出了C语言创建者Ritchie所给出的《C语言参考手册》；附录B是完整的C语言BNF语法图，供读者参考；附录C列出了在用C语言进行程序设计时所要用到的输入输出子程序、系统调用和库函数，虽然这些程序都是以UNIX系统为背景的，但同样适用于别的系统；附录D列出了C的适用领域；附录E介绍了C编译在小型机、微型机上的标准版本和各种变种；附录F介绍了C语言编译命令cc和连接装配程序ld，并比较了若干C语言版本使用方法上的细微差别；最后，附录G列出了本书编写时用到的一些参考文献。

最后需说明的是，在编写本书时，我们用语法图方式对C语言的语法成分进行描述，比较直观、形象，也比较严格。由于至今还没有公认的C的形式化描述，所以只能说是一种尝试。特别是笔者的经验和水平均有限，所以一定会存在一些问题。我们恳切地希望广大读者和有关专家提出批评。

本书编写和修改过程中得到许多同事的支持和鼓励，在此，对诸位同行的帮助、支持和鼓励表示衷心的感谢。

# 目　　录

# 第一章　C语言概述

## 1.1　C语言历史和特点

### 1.1.1 C语言的演变历史

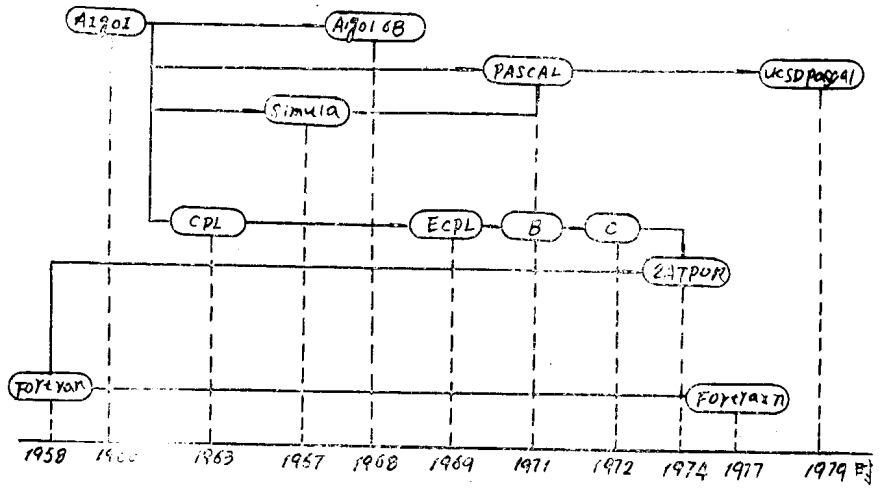
C是UNIX系统的主力语言，它与UNIX系统有着互相依存、休戚与共的紧密关系。UNIX系统是美国贝尔实验室的K·Thompson和D·M·Ritchie从1969年开始，用不到两个人年的时间研制成功的。该系统最早运行在DEC的PDP—7上，第一版本是在GE 635机上产生并通过纸带把可执行代码传送到PDP—7上的。在UNIX上实现了汇编语言后，UNIX系统又以汇编语言编写。由于汇编语言不可移植，并且描述问题的效率不如高级语言，特别是可读性差，所以K·Thompson决定开发一种高级语言来描述UNIX系统。1970年K·Thompson在PDP—11／20上实现了B语言并用B写了UNIX操作系统和绝大多数实用程序。B语言主要思想源于BCPL语言，BCPL是M·Richards基于CPL语言在1969年发表的一种单一数据型语言，至今仍显示出足够的生命力。不过，由于下列原因，B没有盛行起来而导致D·M·Ritchie于1971年开始开发C语言：第一，PDP—11是字节编址的，而B却面向字，这样就妨害了对单个字节进行存取的能力。第二，现代的高级语言都要求有强有力的类型结构，而BCPL和B语言无类型性（或者说只是一种机器字类型）在描述客观世界许多事物时会遇到相当多的困难。Alogl 68和Pascal语言是强类型型语言，它们（特别是后者）的数据类型丰富是其最大的优点之一。第三，B编译程序产生的是解释执行的代码，运行速度比较慢。当然还有一些次要的原因。总之，对C语言的开发已是势在必行。1972年C便投入了使用。

1973年K·Thompson和D·M·Ritchie把UNIX系统用C重写了一遍，系统的代码量比以前的版本大了三分之一，加进了多道程序设计功能，特别是整个系统（包括C语言编译程序本身）建立在C语言基础上，而C语言又具有良好的可移植性，所以第五版的UNIX系统（UNIX V5）就奠定了UNIX系统的基础。以后UNIX V6，V7，System Ⅲ和最新的System V都是在V5基础上发展、扩充的。今天UNIX系统已在全世界范围内取得了巨大的成功，它几乎成了16位微型机的标准操作系统，在PDP—11小型机，VAX—11超级小型机，甚至象IBM370，4300，UNIVAC，Honeywell等大型机上也有它的踪迹。从某种角度可以说，没有C语言就没有UNIX系统的今天。

虽然最初的C语言是附属于UNIX系统且在PDP—11上实现的，但是目前C语言却独立于UNIX系统，独立于PDP—11机而蓬勃发展。它适应的机种可从8位微型机（比如以Z80为CPU的Cromemco）直到Cray—Ⅱ巨型机。它附着的操作系统从8位微型机上的单用户CP／M到大型机的IBM VS／370。它与FORTRAN，Pascal等语言一样已经成了微、小、超小、大、超大和巨型机上共同使用的语言。

人们喜欢用 Pascal 与 C比较，其实它们主要都是 Algol 语言系统的发展，并各有其特点。图1—1列出了Alogol语言属系的一些主要演变情况。

C语言之所以起名为C，有人说因为C是BCPL经过B演化来的，所以延接B而称为C，而另有人说这是按字母 A、B、C、D、……的顺序，已有 B，接下去就应该是 C。C 语言的开发者D·M·Ritchie对这两种说法均不置可否。C之后的发展是叫P还是叫D，也任人猜测。不过从它的仅一个字母的名字倒常常令人联想到它的简洁性。




注：图中portran应为Fortran，RATPOR应为RATFOR，ECPL应为BCPL，1971应为1970。

图1—1　几种主要语言的演变

### 1.1.2 C语言特点

C 语言的特点是多方面的，人们从不同的角度总结出众多特点。根据笔者的体会并参照流行的说法大致归结为以下几点：

1. 语言表达能力强。C 是面向结构程序设计的语言，通用性好，不局限于某种机器。它可以直接处理字符、数字、地址，可以完成通常由硬件实现的算术、逻辑运算（如C 中提供了字位运算的能力，可对变址的地址做有关操作等）。它反映了当前计算机的性能，所以有效到足以取代汇编语言来编写各种系统软件和应用软件。最明显的例证是UNIX 系统。UNIX系统主要分三层：核心（操作系统）、与外层的接口shell命令解释程序以及外层大量的子系统，包括各种软件工具、实用程序和应用软件。这些众多的程序中，除了核心内部的1000行左右（整个核心的10%以下）因为效率、对机器硬件的表达等原因需要用汇编语言书写之外，其余的都是用C语言描述的。当然它同样可以表达数值处理，字处理功能，所以它又是一种通用语言。

2. 语言本身简洁，使用灵活，易于学习和应用。在表示方法上力求在明了易懂的前提下简单易行。如用一对｛、｝来代替复合语句括号begin、end，利用赋值运算符（如＋＝、－＝、＊＝、／＝、＆＝、｜＝等）对运算且赋值的形式进行精简等等。另外，C把一般语言的许多成分都通过显式函数调用来完成，使得编译程序小而精。比如C没有提供 I／O 设

施，也没有并行操作、同步或协同程序等复杂控制，而是提供了大量而有效的库函数来实现输入／输出、字符串处理及存储分配操作等功能，库函数可根据需要，方便地扩充。C本身进行时所需支持少，存储空间占得也少。

3. 具有构造数据类型的能力，并有很强的控制流结构。它可以在简单类型（如字符、整数、浮点数等）的基础上按层次产生各种构造类型（如数组、指针、结构和联合等），因而C的数据类型很丰富。同时，它的各种控制语句如 if、while、for、switch、break 等，功能很强，足以描述结构良好的程序。此外，它的存储类说明（如 extern、statie）在功能上有助于数据隐藏的模块化结构程序设计。

4. 语言生成的代码质量高。高级语言能否用来描述系统软件，特别是象操作系统、编译程序等，除要求语言表达能力强之外，很重要的一个因素是语言生成的代码质量如何。如果代码质量低，系统开销就大，无实用价值。许多试验表明，针对同一问题，用 C 编写程序，其生成代码的效率仅比用汇编语言写的代码低10～20%。由于用高级语言描述比汇编语言描述问题编程迅速、工作量小、可读性好，易于调试、修改和移植，所以C就成了人们描述系统软件和应用软件比较理想的工具。在代码质量上C确实可与汇编语言媲美。

5. 可移植性较好。可移植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。汇编语言因为依赖于机器硬件，所以根本不可移植，有一些高级语言如Fortran 等其编译也不可移植。目前许多不同机器都配有 Fortran，基本上都是根据国际标准重新实现的。而C语言目前在许多机器上出现，大部分却是由 C 语言编译移植得到的。统计资料表明，不同机器上的 C 编译程序 80% 的代码是公共的。C语言的编译程序便于移植，也就使在某个环境上用C 编写的许多程序可以很方便地移到另一个环境上。可移植性良好是C语言最为人称赞的特点。诚如荣获83年度图灵奖时评论中所说的，“对于UNIX 系统可移植性的关键贡献是Ritchie开发的C程序设计语言”。

C 语言的优点很多，但是也有一些不足和缺点，虽然有些已得到了弥补。这些不足和缺点包括：运算符优先级太多，不便于记忆，有些还与常规约定有所不同；类型检验太弱，转换比较随便，所以不太安全（当然目前有一个预处理程序 lint，它主要用来检验类型协调匹配，帮助解决移植中的一些问题）。尽管C 语言有这样那样的缺点，但仍不失为一个实用的通用程序设计语言，特别是一种强有力的系统程序设计语言。我们说它实用，是指它表达能力很强。C语言在理论研究方面可能不及Pascal、Algol 68 等语言，但从出产品、实用角度来说，要比它们强，这与Algol 60和Fortran 之间的关系有相似之处。由于它以上的几个突出优点而吸引人们对它倾注越来越多的关心，以至在世界上使用、研究C 语言的人数正迅猛扩大。

## 1.2 C语言的一般介绍

现通过若干小型例子给出C程序的一些概貌，使读者对C 程序有一个初步的印象，便于后面几章深入地开展对于C语言及用C进行程序设计的讨论。另一方面，也使读者对于 C 语言程序的编制、编译、运行的过程有一大致了解。本来程序的编制、编译、运行等应是使用系统（比如 UNIX ）而不是C语言本身讨论的课题，但是鉴于许多读者对C语言还感到生疏，所以在这里作一简略介绍可能有助于读者在阅读本教程的同时，上机进行实际练习。我

们坚信学习新语言的唯一途径是用此语言编写程序并且获得预定结果，故希望读者多多上机练习。

### 1.2.1 字汇表与语法图

任何一种高级语言，都需要有自己的基本字汇表，C 语言也不例外，其基本字汇表由下列几部分构成：

数字：0，1，2，…，9

英文字母A、B、C、…、Z，a、b、c、…、z；由于有些系统（比如 UNIX 系统）喜欢小写字母，在内部处理时往往以小写字母为主，所以只有大写字母的终端在使用时可能遇到困难。

下划字符：_，这个下线符起一个英文字母的作用，所以在构成标识符等语法成分时，以下线“_”打头，在C中是合法的，而在通常的语言中是不允许的。

特殊符号，主要包括下列一些运算符和关键字：

+、-、*、/、%（取余）、=（赋值）、<、>、<=、>=、!=（不等于）、==（全等比较）、<<（左移）、>>（右移）、&（逻辑位与）、|（逻辑位或）、&&（合取）、||（析取）、^（逻辑位异或）、~（按位求反）、（、）、〔、〕、→（指向）、·（成员）、!（反）、?：（三元运算符）；

int、char、float、double、struct、union、long、short、unsigned、auto、extern、register、static、typedef、goto、return、sizeof、break、continue、if、else、do、while、switch、case、default、for、enum、entry。

关键字 entry 目后还未由哪一个编译程序实现，但保留给将来使用。在某些具体实现中还保留了如下两个字：fortran和asm。

下面几个字严格地说虽不属于关键字，但建议读者把它们看成关键字而不要在程序中随意使用，以免造成混淆，这些字是：

define、undef、include、ifdef、ifndef、endif及line。这些字主要用在C 语言的预处理程序中。

在C语言中，关键字都有固定含义，不能作为一般标识符使用。

如同自然语言和编织、音乐等活动中的专用语言一样，C 语言也有它特定的语法规定。利用基本字汇表中的一些符号和关键字，按照给定的语法规则，就可以进一步构造其它符号、语句和程序。在程序中不允许出现有上述字汇表中没有的符号，比如，不允许将乘法的*写成×，也不允许出现违反语法规则而构成的符号。另外，在C语言中有些符号在不同的上下文中具有不同的含义，比如*，在变量的类型说明中，可能解释成“某某变量是一个指针”，即*是指针类型的标志；但在赋值语句赋值号的右边，它又表示“把某某变量的内容赋给左边的量”，即*又是“变量内容”的标志。这些地方需要特别小心，必要时我们会时常提醒读者注意的。

编写出的程序是否合法，完全取决于它们是否遵守给定的语法规则。通常，表示语法规则的形式有两种方法：即 BNF（Backus—Naur Form）和语法图。BNF 首先用于描述 Algol 60的语法规则，沿用至今已有20多年的历史。这种方法的最大优点是清晰、严谨，

在语言形式化描述、编译程序自动生成等研究领域中是一种最为有效的工具。语法图最早盛行于描述Pascal的语法规则，至今已有10年的历史。这种方法的最大长处是直观形象，在语言教学中是一个得力的工具。本书主要是为了讲述C语言，而不是研究C语言，所以采用了语法图形式，在附录B中我们给出C语言语法规则的完整的语法图和BNF表示，以便读者参考。要说明的一点是，无论是语法图还是BNF表示，都是笔者根据自己的学习体会并参照有关资料编制而成的，在某些方面可能有疏漏和不严格之处，望读者注意。

下面我们给出若干C语法成分的语法图。

例1.1在C语言中无符号整数unsigned int（或直接写成unsigned）的语法图为图1—2所示。

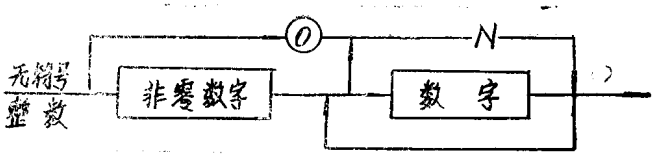



图1—2 C语言无符号整数语法图

方框中的数字是指字汇表中的一位数字，沿着箭头的方向，通过语法图的一条路线就定义了unsigned的语法规则。由此可知：C语言中的一个无符号整数至少包含一位数字，它定以某一非零数字开头，后面接着一串（包括空串）数字，单独一个0也是。如：

5，67，257，32765，0

这些都是C语言中合法的无符号整数。非零数字和数字的语法图如图1—3和图1—4所示。

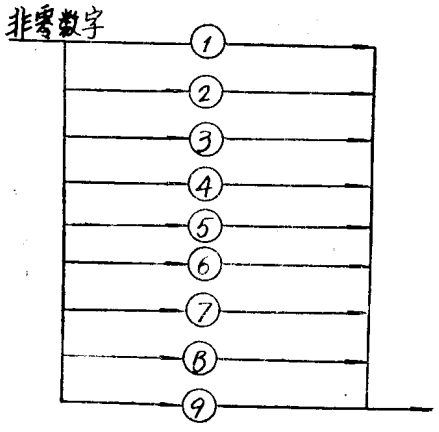



图1—3 非零数字语法图

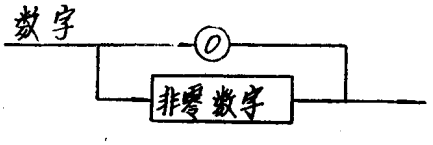



图1—4 数字语法图

例1.2 在C中，十进制整数的语法图如图1-5所示。

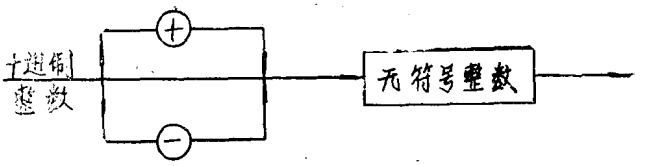



图1—5　十进制整数语法图

十进制整数前可冠以正负号（+或-），正负号缺省时可以认为该数是正数。

C中的十进制整数int是机器字长的自然表示，在16位机（PDP 11 系列，16 位微机）上其取值范围是-32768～+32767，在32位机（IBM370，Interdata 8/32、VAX-11 系列）上其取值范围是-2147483648～+2147483647。

在C中合法的十进制整数如：

4096，+10，-100，-32768，+32767，\0。

例1.3 在语言中，常常需要对一些变量、常量、程序等对象命名，一般均采用标识符来代表它们的名字。在C语言中，构造一个标识符的语法图如1—6所示：

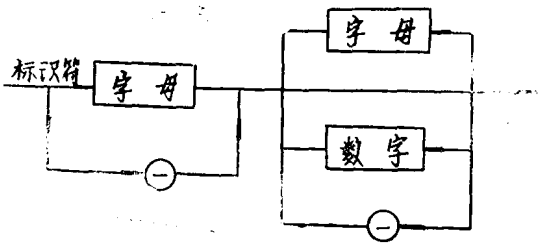



图1—6　标识符语法图

字母是指英文26个小写和26个大写字母。C与一般语言的标识符不同之处是，因为下线字符“_”也起一个字母的作用，所以可以作标识符的打头字符，后跟字母或数字串（包括空串和下线符）。合法的标识符是：

xyz，text，spls，__global，__FILE

等。在C 语言中，人们喜欢以小写字母作为标识符。以下线字符打头的标识符一般在内部使用，以示区别。所以用户一般不要以下线符作打头字符。另外，标识符最好起得有意义，便于联想和记忆。

从上述几个简单的语法图例，可初步得知如何用语法图来描述相应的语法规则。在语法图中，从箭头的入口到出口，按箭头的指向有一条以上的路线。按每条路线所定义的符号，语句或程序都是合法的。如果读者没有使用一般计算机语言的知识，那么在开始学习用C 语言编写程序时，应经常参照语法图或 BNF，以检查所编写的程序是否符合语法规则。如果读者有使用计算机语言的基本知识，那么重点应放在C 与别的语言的不同点上，通过类比就可以较快地掌握C语言的使用方法。

### 1.2.2 C语言程序结构

1. C程序和主函数

一个计算机的程序主要由两部分内容组成。一部分是关于程序所要实现的算法的描述，这种描述一般由一系列语句组成，而这些语句又可能按描述对象的要求以及语言本身的规则构成复合语句、分程序或函数、过程等。另外一部分是关于这些算法所要操作的对象（通常用数据来表示）的描述。在程序中出现的数据，或者是常量或者是变量。对数据的描述就是对程序中所用到的常量和变量进行说明，特别是对变量要进行类型说明。下面举几个简单例子，说明如何用C语言来编写程序及其程序的基本结构，使读者对它有一个直观的了解。

例1.4

```
/*small.c—The smallest C Program */
main ( ) /* There is no executable code */
 {
}
```

C程序一般是由一个或多个函数组成的，这些函数可驻留在一个或几个源文件中，这些文件都以.c结尾，譬如本例中，C语言程序只有一个函数，并且驻留在一个文件small.c中。组成一个程序的若干函数中必须有且只能有一个名为main的函数，可运行的C程序总是从main 开始执行。一个函数在其名字之后一定要有一对圆括号“（”和“）”，这是函数的标志。圆括号中放参数，参数可有可无，如本例中就没有参数。

程序中以“/*”开头到“*/”结束表示是一个注释，注释帮助读者阅读和理解程序，但在编译时，注释行是要忽略掉的，即它不产生代码行。注释在各种语言中都是希望有的，甚至是越多越好。注释行可以在程序的开头，这一般说明整个程序的功能和注意事项；也可以插在程序语句行中或在某行语句的尾部，主要用来说明一段程序的功能或某一行程序的作用。在本例中，程序开头及第一行后都有注释。要注意在C语言中注释不能嵌套。

在例1.4中，第3、4行实际上是一对花括号，在这里花括号可视为程序体括号，类似于Pascal等语言中的“begin”和“end”，这里不过是一种简略表示，它也可以用来括起任何一组语句，从而构成一个有意义的复合语句或分程序。花括号中可以有任何有意义的语句，包括空语句在内。要注意在一个函数中至少要有一对花括号，也就是程序体括号，而不管程序体是否为空（如本例那样）。本例是一个有意义、正确的C程序，虽然实际上它什么也没有干。例1 4中的程序也可以写成例1.5的形式。

例1.5

```
/*small2.c—The smallest C program, on one line*/
main ( ) { }
```

如例1.4和例1.5所示，C的一般函数或“main ( )”之后及最后的右花括号“}”之后，都没有通常的“;”或“.”之类的语法符号。

前已说过一个C程序可由若干函数（这些函数可在同一个或多个文件中）组成（其中，最少要有一个函数main）。如果一个完整的程序中根本没有函数main，那么编译时将出错。下面举例子说明，假设将例1.6所示程序放在test.c文件中。