普通高等教育“十二五”规划教材

C语言程序设计

主　编　祁昌平
副主编　李晓霞　白春霞

科学出版社

普通高等

C语言程序设计

主　编　祁昌平

副主编　李晓霞　白春霞

参　编　申雪琴　贺登超

主　审　赵　柱　吴建军

科学出版社

北　京

内 容 简 介

本书根据 C 语言的发展和计算机教学的需要，精心设计案例，融计算思维于一体。在以知识点为主线的基础上，兼顾“数据表示”和“程序设计”线索，优化了 C 语言程序设计的知识安排。主要内容包括程序设计基础、基础数据类型和表达式、顺序结构程序设计、选择结构程序设计、循环结构程序设计、数组、函数、指针、结构体与共同体、文件、高级编程等。

本书内容先进，结构清晰，体系合理，案例丰富，代码专业，是初学者学习 C 语言程序设计的理想教材，可以作为高等院校非计算机专业的程序设计教材，也是适合广大编程爱好者自学的好教材，还可以作为全国计算机等级考试的辅导用书。本书配有授课课件，需要者请与作者联系，联系方式在前言结尾。

图书在版编目(CIP)数据

C 语言程序设计 / 祁昌平主编. —北京：科学出版社，2017.2
普通高等教育“十二五”规划教材
ISBN 978-7-03-051635-0

Ⅰ. ①C… Ⅱ. ①祁… Ⅲ. ①C 语言－程序设计－高等学校－教材
Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字(2017)第 009159 号

责任编辑：于海云 / 责任校对：郭瑞芝
责任印制：霍 兵 / 封面设计：迷底书装

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

文林印务有限公司 印刷

科学出版社发行 各地新华书店经销

*

2017 年 2 月第 一 版 开本：787×1092 1/16
2017 年 2 月第一次印刷 印张：16
字数：450 000

定价：42.00 元

(如有印装质量问题，我社负责调换)

前　言

随着计算机技术的快速发展和计算机网络的快速普及，当今社会已进入信息时代，计算机在各行各业中的应用越来越广泛。对高等院校的学生来说，不论是什么专业，学习和掌握一种计算机程序设计语言都是十分必要的。

本书以知识点为主线，以编程应用为驱动，理论联系实际，通过丰富的典型案例，详细介绍了C语言程序设计的思想和方法。全书力求概念清晰、准确，案例丰富、有趣，重点、难点突出。本书从最基本的计算机程序设计基础知识讲起，由浅入深，循序渐进，使读者学习本书后，可较快地掌握C语言。

在内容的指导思想上，本书以C语言为工具，介绍计算思维方法和程序设计的基本方法，把计算思维方法和程序设计中最基本、最有价值的思想和方法渗透到C语言的介绍中。目的是使读者在学习了C语言后，无论使用什么语言编程，都具有灵活应用这些思想和方法的能力。

全书共11章，内容包括程序设计基础、基本数据类型和表达式、选择结构、循环结构、数组、函数、结构体与共用体、指针、文件、高级编程。第1～10章以导读开头，指导学生整体把握本章主要内容；重点章节精心组织了一节程序设计举例，以一个个富有趣味的经典实例介绍一种新的编程思维，便于学生在轻松愉快的气氛中学习；每章安排了易错问题举例，汇总了本章容易出现的错误，使学生对本章知识的掌握更加准确；通过章后的习题可以进一步巩固所学知识；特设高级编程，详细介绍了几个综合性设计案例，可以根据专业不同，适当要求学生完成其中一项即可，既锻炼了学生的设计能力，又培养了学生分析、解决问题的综合素质。书中变量都取自程序中，形态与程序中保持一致。

本书编写过程中，王玲老师、董玉蓉老师和公维军老师负责校稿，赵柱教授和吴建军副教授负责审稿，在此，一并表示感谢。

本书由祁昌平担任主编，负责统稿、定稿等工作。参与编写的有李晓霞(编写第4～6章)，白春霞(编写第1～3章)，祁昌平(编写第7～10章)，申雪琴(编写第11章)，贺登超(编写了附录)。

本书可作为普通高等院校非计算机专业学习计算机程序设计语言的教材和广大编程爱好者的自学读物，还可以作为全国计算机等级考试的辅导用书。

因编者水平有限，书中难免有错误和不妥之处，恳请广大读者提出宝贵意见。我们会在重印时及时予以更正。编者的E-mail：hxuzhy@163.com。

编　者

2016年12月

目　录

第 1 章　程序设计基础

内容导读：

C 语言程序设计在程序设计语言中有很重要的地位和作用，是学习程序设计的一门基础课程。本章以基本操作、基本语法规则和基本编程方法技巧为主，强调理论联系实际，帮助读者掌握程序设计的思想和方法，解决工程实践中的实际问题。

- 程序和程序设计语言
- 算法
- 结构化程序的设计方法
- C 语言及其特点
- C 语言的上机步骤

1.1　程序和程序设计语言

1.1.1　计算机与程序

当今，计算机已广泛应用于社会生活的各个领域，成为大众化的现代工具。但是，不熟悉计算机的人仍然把它想象得十分神秘。其实，计算机不过是一种具有内部存储能力、由程序自动控制的电子设备。人们将需要计算机做的工作写成一定形式的指令，并把它们存储在计算机的内部存储器中，当人们给出命令之后，它就按指令操作顺序自动执行。人们把这种可以连续执行的一条条指令的集合称为“程序”。可以说，程序就是人与机器进行“对话”的语言，也就是我们常说的“程序设计语言”。

程序设计语言是用户用来编写程序的语言，是人机之间交换信息的工具。程序设计语言一般分为三类：机器语言、汇编语言和高级语言。

1. 机器语言

机器语言是用机器指令编写的程序，可以由计算机直接执行。程序中每一条机器指令都是以二进制编码的形式出现的，每一台机器的指令集就是机器语言，因此机器语言与计算机一一对应，有多少种计算机就有多少种机器语言，针对一种机器编写的机器语言程序不能在另一种计算机上运行。由于机器语言是针对机器硬件编写程序的，所以它的执行效率高，能充分发挥计算机性能。但是，编写程序的难度大，常常需要由计算机专业人员编写，一般人员不能进行编程。

2. 汇编语言

用助记符替代机器指令称为汇编指令，用汇编指令编写的程序称为汇编语言源程序。汇编语言的命令与机器语言指令一一对应，因此，汇编语言也与具体针对的计算机有关，一种机器上的汇编程序同样不能在另一种机器上运行。

汇编语言由于采用了人们比较容易记忆的助记符，相对于机器语言就直观得多，并且容易理解和记忆，但计算机不能直接识别，必须由针对某一种机器编写的“汇编程序”对汇编源程序进行解释，将其翻译成机器语言才能运行。这种翻译过程就称为“汇编”。

3. 高级语言

机器语言和汇编语言一般称为低级语言，是面向机器的语言，开发这类语言比较困难，一般用户很难胜任这一工作。随着计算机的不断发展，计算机用户队伍在不断扩大，要使普通用户也能参与软件开发工作，从20世纪50年代起就发展了面向问题的程序设计语言，这就是高级语言。

高级语言与计算机的硬件无关，其表达方式接近于人对问题的描述，容易被人接受和掌握。用高级语言编写程序比用低级语言容易得多，编制出的程序通用性强，可移植性高，容易修改。

高级语言从20世纪50年代发展到现在已有上百种之多，得到广泛应用的有十几种，几乎每一种高级语言都有它自身最适用的领域。表1-1列出了常用的高级语言的应用领域。

表1-1 常用的高级语言的应用领域

语言名称	语言特点	应用领域
BASIC	基础语言	教学和小型系统的开发
FORTRAN	基础语言	科学工程计算
PASCAL	结构化语言	教学和应用系统的开发
COBOL	基础语言	商业和管理应用系统开发
FOXBASE	专用语言	数据库管理系统
C	结构化语言	中小型系统软件开发
C++	结构化语言	面向对象程序开发
LISP	专用语言	人工智能
PROLOG	专用语言	人工智能
Java	结构化语言	应用程序开发

高级语言发展非常快，新的版本正在不断推出，例如数据库开发语言，适用于大、中型系统的网络数据库软件SQL-Server、Sybase、Oracle、Informix和DB2等。但必须指出，任何一种高级语言编写的程序(源程序)都要经过编译或解释程序，翻译成机器语言后计算机才能运行。

1.1.2 程序设计中的主要问题

通过程序设计方法的学习和实际练习，就能动手进行一般应用软件的设计。应用软件设计中有3个主要问题：功能设计，算法设计，结构设计。这三者之间既有联系又不能相互代替。

1. 功能设计

依据“解决什么样的问题，完成什么样的功能”，提出“面向计算机的、含义明确而无歧义”的说明书。

功能需求一般是由用户提出的。这些需求未必合理和利于在计算机上实现。设计者分析用户需求后，应使之“面向计算机且含义确切”，也就是说便于在计算机上实现。

例如，用计算机做教务管理，要求具备“学生学籍管理，学生成绩管理，课程安排”等功能，即要分析它的全局和细节，将各部分功能具体化，使之便于在计算机上实现。一个“面向计算机的、含义明确而无歧义性”的说明书，将是算法设计和程序设计的依据。

2. 算法设计

算法设计在程序设计中占有特别重要的位置。如果对被求解问题的算法模糊，则不可能求解出它的正确程序。算法设计是提出实现软件功能的合适算法，算法应是正确有效的，并且用自然语言或伪代码描述。

比如，计算 1+2+3+4+5+6+…+100 之和的算法，其算法为：

S1：设置一个累加和变量 sum 和一个计数变量 n，并设它们的初值都为 0；

S2：判断 n<=100，若成立转 S3，否则转 S5；

S3：sum+n=>sum，n+1=>n；

S4：转 S2；

S5：输出 sum。

3. 结构设计

结构设计就是选择适当的数据结构和程序结构实现算法。这和建筑一栋楼房时要进行周密的结构设计相似。

算法设计和结构设计之间有联系，但不能相互代替。

1.2 算　　法

1.2.1 算法的概念及特性

算法就是解决问题的方法与步骤。

1. 目标问题分析

对目标问题的分析是程序设计的基础。只有对问题进行充分的分析、理解后，才能寻找出正确的算法，有把握编制出高水平的程序，从而求得正确的结果。问题的分析很复杂，程序员面对的是各种各样的问题，当然不同的问题就需要不同的解法。为针对问题进行分析，一般来说应从下面几个方面着手。

1) 分析问题的性质

人面对的问题是各种各样的，而对于不同性质的问题，使用的方法、工具一般是不同的。首先程序员应分析所面对的问题属于数值型数据的计算还是非数值型数据处理的问题。对于数值型数据的计算问题要考虑计算结果的精度，从而定义输入数据和中间结果的数据类型，以求获得一个合理的精度要求。对非数值型数据的处理，则需要考虑输出结果与输入的关系，合理定义输入数据的数据类型，求得数据类型的统一。

2) 确定输入/输出

程序设计中分析问题常常通过从输出的要求回溯到输入，或从输入数据分析一步一步到输出。在这个分析过程中输入/输出的数据应从下面几个方面考虑。

(1) 数据的类型，即数据设为整型、实型(单精度或双精度)、字符型等。

(2) 输入/输出时数据定义格式。

(3) 由哪些设备完成数据输入/输出。

3) 数学模型

通过分析问题的性质，确定输入/输出数据类型后，一般就要考虑数学模型的设计，寻找一个适合于该问题的算法。

2. 算法的设计

算法一般说来是在有限步骤内解决一个目标问题，因此它是一个有明确意义描述的步骤的集合，即指对解题过程的准确而完整的描述。例如，要求一个圆的周长和面积，就应知道求它们的公式：周长=$2\pi R$，面积=πR^2。写出对这个问题的算法如下：

(1) 从键盘输入半径 R，π=3.1415926；

(2) 计算周长 Z=2*π*R；

(3) 计算面积 S=π*R*R；

(4) 显示 Z 和 S。

从上面例子中看到，算法不同于一般公式，算法应具有以下一些基本特征。

1) 可行性

描述对某一问题的算法，每一步必须都能实现。例如，求算术平方根中出现在实数范围内对负数求平方根这样的算法描述，在除法运算中出现分母为零的情况等，在算法设计中就要避免，不允许出现。

2) 唯一性

算法是针对某一问题提出的解决本问题的操作步骤。每一步必须确定，不允许出现模棱两可的解释，也不允许多义性，而是针对此问题的唯一的执行步骤。

3) 有穷性

算法不允许无限制地进行计算，必须在一定的时间内完成。对于像数学中的无穷级数，在算法描述中只能按实际要求取有限项。

4) 有零个或多个输入

一个算法应该有明确的数据源，要根据提供的数据进行实际运算，数据获得的方式可以直接赋值(0 个输入)，也可以从外部输入(至少 1 个输入)。

5) 有一个或多个输出

将运算的结果输出，判断结果是否满足设计要求，这就要求必定通过一定的方式将数据输出。

综上所述，算法是一个针对某一问题的一组严谨的运算规则，它的每一规则都有明确的定义，并且都将在有限次的操作后终止。又因为算法是针对某一问题建立的，所以算法按其操作的数据对象又分为：数值型求解算法和非数值型数据处理算法。

3. 算法的类型

算法是为解决特定问题而设计的，而要解决的问题门类繁多，因此，算法也会呈现出多样性与复杂性，但这并不妨碍对算法做一个大致的分类。对于计算机处理而言，算法根据其应用方向可以大概分为数值算法和非数值算法两种。

数值算法可以定义为“数学问题构造性解法的一个完备而确切的描述，并规定方法中仅允许加、减、乘、除等基本算术运算”。数值算法常用于科学计算领域中。

非数值算法则广泛应用在信息(数据)处理的场合。这类问题常常要对大量的数据进行加

工处理(搜集、转换、分类、组织、检索、存储、维护等)，有时还要绘制数据分布曲线或打印出报表，还可以根据加工后得到的信息寻找规律，进行预测。这些处理工作一般不涉及复杂的数学问题，但数据量大，数据的类型和结构也较复杂。

4. 算法的基本结构

一个算法是由“结构”和“原操作”两个部分构成的。最基本的结构有 3 种，它们是顺序结构、分支结构和循环结构。下面用图来分别描述这 3 个基本结构。

1)顺序结构

这个结构是由若干个依次执行的处理块组成的。图 1-1 是包含两个处理块的序列，其中，A、B 分别代表不同的处理块。

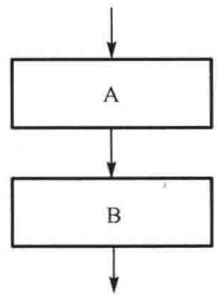


图 1-1　顺序结构示意图

顺序结构是任何一个算法都离不开的基本主体结构。

2)分支结构

最基本的分支结构是二分支结构。它是根据某一逻辑条件是否成立而决定选择哪一个分支上的处理块去执行，所以分支结构又称选择结构。如图 1-2 所示是分支结构示意图。

分支结构总是以条件或情况的判断为起始点的，它是人脑思维判断活动的抽象。

3)循环结构

循环结构是指在算法设计中，从某处开始有规律地反复执行某一处理块，该处理块称为循环体。循环体执行多少次是由一个控制循环的条件决定的。当控制条件成立时，重复执行处理块。典型循环结构示意图如图 1-3 所示。

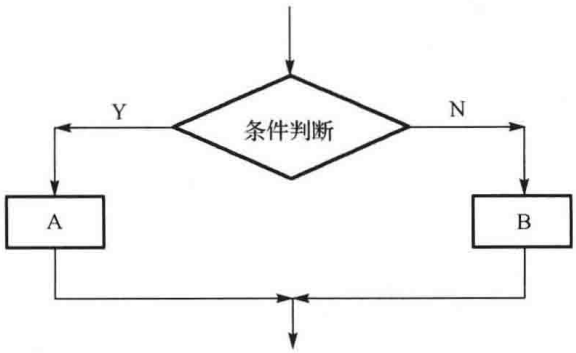
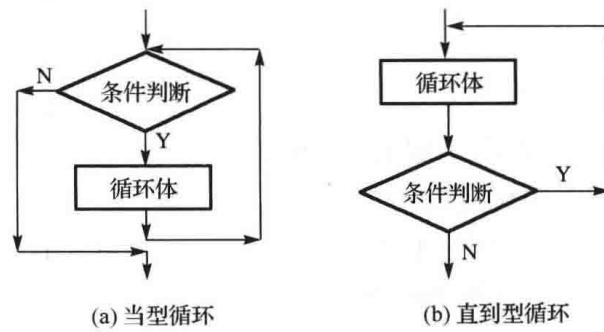


图 1-2　分支结构示意图



(a) 当型循环　　(b) 直到型循环

图 1-3　循环结构示意图

循环结构反映了人们在处理某一事件时，对不同数据执行同一操作的工作方式。

这三个结构中的每一个块都具有一个入口和一个出口，而结构中的每一个处理块，如图 1-1 中的 A、B，也都具有一个入口和一个出口。由这三种基本结构可以繁衍出无限多的结构来，可以表示任意一个复杂的算法。

1.2.2 算法的描述工具

程序设计的过程中，可以使用不同的算法描述工具确定解决问题的详细步骤。常用的描述工具有以下几种。

1)自然语言

自然语言指人类在日常生活、学习、工作中通用的语言，这种语言不需要设计者专门学习和训练。但是使用自然语言做系统描述时，要求用语简练，尽量减少语言修饰。例如，用自然语言描述打印两数之和及求平均值的算法。

(1)从键盘读入两个数 A1 和 A2；

(2)计算两数之和 S=A1+A2；

(3)求两数的平均值 V=S/2；

(4)打印和 S、平均值 V。

自然语言描述算法容易理解，一般适合于较小的程序设计，但对于较大的程序设计工程，使用自然语言进行算法描述会令人感到不直观，容易产生多义性，也增加了理解的难度。使用算法描述工具是改善程序设计环境、提高设计效率和质量的一条重要途径。常用算法描述工具有流程图、NS 图、PAD 图和 PDL 语言等。其中流程图和 NS 图使用最广。

2)流程图

流程图是用图形来描述问题的处理过程的工具。流程图普遍用于复杂计算和工程设计，它能直观、灵活地表现条件、活动和转移。如图 1-4 所示为国家标准规定的一些常用图符。流程图把控制和执行顺序表达得十分清晰，看起来也直观易懂，这就使得程序员习惯用流程图表示算法。其实流程图也存在严重的缺陷，由于流程线方向任意，如使用不当，会设计成非结构化方式，影响程序设计质量。

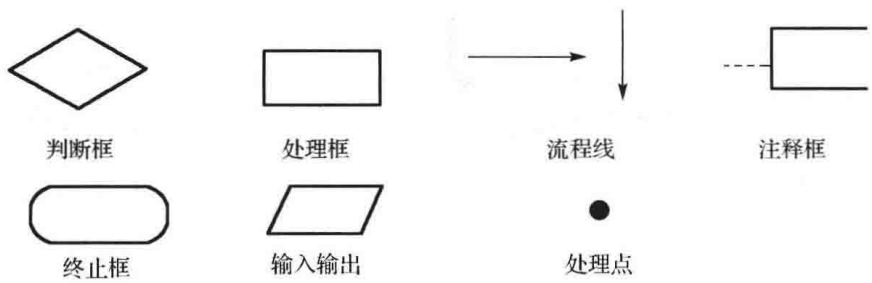


图 1-4　常用流程图框

图 1-1、图 1-2、图 1-3 也是算法的流程图表示方法。

3)NS 图

NS 图是 1973 年美国 I.Nassi 和 B.Shneiderman 提出的一种新的符合结构化程序设计的流程图，该算法的描述写在一个框内，去掉了容易引起麻烦的流程线。NS 图表示 3 种结构的流程图，如图 1-5 所示，图 1-5(a)所示表示顺序结构，图 1-5(b)表示选择结构，图 1-5(c)表示循环结构。

从 NS 流程图中看到，NS 图强迫设计者按结构化的要求构造算法。它有助于培养良好的按结构化原则进行程序设计的习惯。NS 图的缺点是由外框开始逐步向内画，可能图的整个布局由于考虑不周，使内部矩形功能域太小，无法向内层扩展，在设计时用户应尽量给出起始框余量。

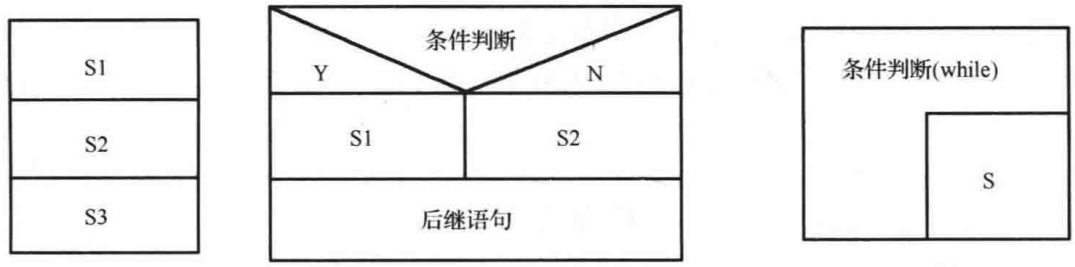


图 1-5　NS 流程图

1.3　结构化程序的设计方法

顺序结构、选择(分支)结构及循环结构称为程序设计的基本结构，由它们组成的程序称为结构化程序。一个结构化程序具有易读性好、可靠性高、便于维护和易于移植等优点。

任何一个结构化程序只能由这 3 种基本结构组成。

1.3.1　顺序结构

顺序结构是最简单、最基本的程序结构。在这种结构中，程序的各块是按其书写顺序依次执行的，如图 1-1 所示。执行完 A 块操作后，再执行 B 块操作。这里所说的每一块可以由一条或若干条不产生控制转移的语句组成。

1.3.2　选择结构

选择结构又称分支结构，在这种结构中通过对给定条件的判断，来选择一个分支执行，如图 1-2 所示。当条件为“真”时，执行 A 块操作；当条件为“假”时，执行 B 块操作。无论何种图形，A、B 两块操作不能同时执行。

1.3.3　循环结构

循环是指在给定的条件下，重复执行某段程序，直到条件不满足为止。循环结构分为以下两种形式。

1) 当型(WHILE 型)循环

这种循环结构的执行过程是先判断条件，若条件为“真”，重复执行某段程序，直到条件为“假”时结束，如图 1-3(a)所示。由于其先判断条件，所以当一开始条件就为“假”时，则 A 块操作将一次也不被执行。

2) 直到型(UNTIL 型)循环

这种循环结构的执行过程是先执行某段程序，然后再判断条件，当条件为“真”时，重复执行这段程序，直到条件为“假”时结束，如图 1-3(b)所示。由于其先执行循环体操作，然后再判断条件，所以无论一开始条件是否满足，循环体都至少被执行一次。

从上述 3 种基本结构可以看出，它们都具有以下特点：

(1) 每种基本结构只能有一个入口和一个出口；

(2) 没有死语句，即程序中所有的语句都有被执行的机会；

(3) 不包含死循环(无终止的循环)。

使用这 3 种基本结构可以组合成各种复杂的程序结构，并且具有自上而下设计的功能。单一的顺序结构只适用于很简单的问题，大多数实际问题的处理都包含了选择结构或循环结构。

1.4 C 语言及其特点

目前，在社会上使用的程序设计语言有上百种，它们都被称为计算机的“高级语言”，如 BASIC、PASCAL、C 语言等。C 语言是 20 世纪 70 年代初美国贝尔(Bell)实验室 Dennis M. Ritchie 设计的一种程序设计语言，正式发表于 1978 年。它是在一种被称为 B 语言的基础上进行重新设计的一种语言，由于是 B 语言的后继，故称为 C 语言。随着 UNIX 操作系统的广泛使用，C 语言也得到迅速的普及。1978 年，Brian W. Kernighan 和 Dennis M. Ritchie(合称 K&R)合著了一本影响深远的书 *The C Programming Language*，这本书所介绍的 C 语言成为后来广泛使用的 C 语言版本的基础，称为标准 C。这些语言都是用接近人们习惯的自然语言和数学语言作为语言的表达形式，人们学习和操作起来感到十分方便。1983 年，美国国家标准化协会(ANSI)制定新的标准，称为 ANSI C，它比原来的标准 C 有了很大的发展。目前，广泛流行的各种版本 C 语言编译系统基本相同，但也有些区别。在微机上使用的 C 语言有不同的版本，常用的编译软件有 Microsoft Visual C++、Borland C++、Borland C++ Builder、Watcom C++、GNU DJGPP C++、Lccwin32 C、Microsoft C、Turbo C、High C 等，它们的不同版本也略有差异，读者可以通过参阅有关手册，了解所用计算机系统 C 编译的特点和规定。

1.4.1 C 语言的特点

1)简洁紧凑、灵活方便

C 语言一共有 30 多个关键字，9 种控制语句，程序书写自由，主要用小写字母表示。它把高级语言的基本结构和语句与低级语言的实用性结合起来。C 语言可以像汇编语言一样对位、字节和地址进行操作，而这三者是计算机中最基本的工作单元。

2)运算符丰富

C 的运算符包含的范围很广泛，共有 34 个运算符。C 语言把括号、赋值、强制类型转换等都作为运算符处理，从而使 C 的运算类型极其丰富，表达式类型多样化，灵活使用各种运算符可以实现在其他高级语言中难以实现的运算。例如，要将变量 i 的值增加 1，可以用以下 4 种表达式完成：①i++；②++i；③i+=1；④i=i+1。

3)数据类型丰富

C 语言的数据类型有：整型、实型、字符型、数组类型、指针类型、结构体类型、共用体类型等，能用来实现各种复杂的数据类型的运算，并引入了指针概念，使程序效率更高。另外 C 语言具有强大的图形功能，支持多种显示器和驱动器，且其计算功能、逻辑判断功能强大。

4)C 语言是结构式语言

结构式语言的显著特点是代码及数据的分隔化，即程序的各个部分除了必要的信息交流外彼此独立。这种结构化方式可使程序层次清晰，便于使用、维护以及调试。C 语言是以函数形式提供给用户的，这些函数可以方便地调用，并具有利用多种循环、条件语句控制程序流向的功能，从而使程序完全结构化。

5)C 语言语法限制不太严格，程序设计自由度大

一般的高级语言语法检查比较严，能够检查出几乎所有的语法错误。而 C 语言允许程序编写者有较大的自由度。如，数组的长度为 N，在数组的引用中，下标的取值为 0～N-1，若编程人员疏忽，将下标取为 N，尽管下标越界，系统是不会检查下标越界的情况，但会导致结果的无法预料性。

6)C 语言允许直接访问物理地址

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，能够像汇编语言一样对位、字节和地址进行操作，而这三者是计算机最基本的工作单元，所以 C 可以用来写系统软件。

7)C 语言程序生成代码质量高

C 程序执行效率高，一般只比汇编程序生成的目标代码效率低 10%～20%。

8)C 语言适用范围大

C 语言有一个突出的优点就是适合于多种操作系统，如 DOS、UNIX，可移植性好，也适用于多种机型。

9)C 语言继承和发扬了高级语言的长处

C 允许递归调用，由于采用递归，使有些算法的实现简明、清晰。

C 语言的优点很多，但也有不足之处。例如，运算符优先级太多，不便记忆，有些与常规约定有所不同；数据类型转换比较灵活，类型检验能力弱，不够安全；编程自由度大，给不熟练的程序员带来一定困难。

综上所述，C 语言把高级语言的基本结构与低级语言的高效实用性很好地结合起来，不失为一个出色而有效的通用程序设计语言。

1.4.2 C 源程序的结构

【例 1-1】 输出信息“This is the smallest C program”。

```
#include<stdio.h>
void main()
{    printf("This is the smallest C program.\n");   // 输出函数
}
```

这是一个最“小”的 C 源程序，仅由一个主函数构成，且函数体为空，程序运行时无任何实际的执行动作。“//”后面是注释，注释表明程序员的编程意图和思想，便于程序员和用户理解程序的作用和功能。我们提倡大家在程序中多用注解。第 2 行是 C 语言的主函数首部，main 是主函数名，这是一个特殊的函数，每个 C 语言程序都必须有一个且只能有一个主函数，它是 C 程序运行的起点。main 后的“()”是函数的参数部分，其中可以为空，但“()”不能省略。

【例 1-2】 空语句。

```
#include<stdio.h>
void main()
{  ;  }
```

这是一个最简单的 C 源程序，包含一条语句(此语句是空语句，由一个“;”构成)，程序运行时也无任何实际的执行动作，但注意和例 1-1 的区别。

【例 1-3】 打印字符串。

```
/* 打印字符串 */
```

```
main()
{   printf("Hello!\n"); }
```

这是一个完整的 C 语言程序，包含一条语句，即输出函数 printf 调用语句。printf 是一个标准的输出函数，括号内双引号括起来的是所带的参数，即要打印的内容，对应的输出设备是终端屏幕。上述程序运行后在屏幕上打印出字符串，其运行结果如图 1-6 所示。

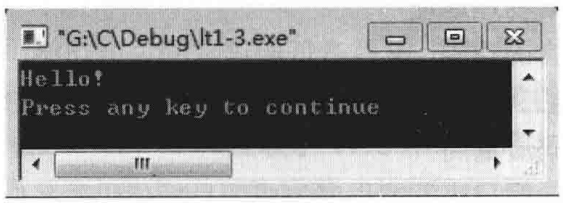


图 1-6 【例 1-3】运行结果

以上例子中，使用了 printf 函数，它称为库函数，实现标准输出功能。在 C 语言中，函数分为两类，一类是系统本身提供的库函数(标准函数)，编程时只要在需要用的地方写上函数名，再加上正确的参数格式就可以调用此函数。一般情况下要在主函数 main 之前加上相应的包含函数库名，比如要调用数学库函数，应该在 main 之前加上包含命令，即“#include <math.H>”。C 语言提供了十分丰富的库函数，经常用到的有输入输出函数(stdio)、数学函数(math)和字符串处理函数(string)等。另一类称为自定义函数，程序员可以根据需要自己设计一段程序来完成一个特定的功能，它相当于 PASCAL 中的过程或函数，等价于一个子程序。

C 语言函数使用简单方便，执行效率高。在 C 程序设计中要养成良好的设计风格，即尽量用多个小函数或小程序通过函数调用的方式来构成一个大程序，而每个小函数或小程序仅完成一个独立的功能。

【例 1-4】 现有两组数据，求每组数中的较大者，并求出两个较大者之中的较大者。

```
#include "stdio.h"
main()
{   int a1,b1,a2,b2,max1,max2,max;
    int Max(int  x,int  y);  //对函数 Max 的声明
    printf("input a1,b1,a2,b2, please:");
    scanf("%d,%d,%d,%d",&a1,&b1,&a2,&b2);
    max1=Max(a1,b1);          //调用函数 Max，将得到的值赋给 max1
    max2=Max(a2,b2);          //调用函数 Max，将得到的值赋给 max2
    max=Max(max1,max2);       //调用函数 Max，将得到的值赋给 max
    printf("max1=%d,max2=%d,max=%d\n",max1,max2,max);
}
int Max(int  x,int  y)        //定义函数 Max
{   int z;
    if(x>y) z=x;
    else  z=y;
    return(z);                //返回 z 值
}
```

这是一个求较大值的 Max(x, y)函数及调用它的主程序所构成的 C 程序，主程序中调用 Max(x, y)共 3 次，但每一次参数不同，即 Max(a1, b1)、Max(a2, b2)和 Max(max1, max2)。其运行结果如图 1-7 所示。