前　言

C 语言是一种国内外广泛使用的面向过程的、抽象化的通用程序设计语言。C 语言能以简易的方式编译、处理低级存储器，仅产生少量的机器语言。同时，其不需要任何运行环境的支持便能运行，是一种高效率程序设计语言。它既可用来编写系统软件，也可用来编写应用软件。C 语言既具有低级语言可直接访问内存地址、能进行位操作、程序运行效率高的优点，又具有高级语言运算符和数据类型丰富、结构化控制语句功能强、可移植性好的优点。在 TIOBE 程序设计语言排行榜中，C 语言多年位居前列，成为程序设计语言的常青树。

本书以“夯实基础、注重实践、培养能力”为编写理念，严格遵循《普通高等学校本科专业类教学质量国家标准》中的计算机类专业相关的要求。党的二十大报告明确提出，“教育、科技、人才是全面建设社会主义现代化国家的基础性、战略性支撑”，为高等教育与信息技术领域的发展指明了方向。C 语言作为程序设计的“基石语言”，是培养计算思维、筑牢信息技术学科基础的关键载体，其教学不仅关乎代码编写能力的提升，更承载着落实立德树人根本任务、培育科技创新人才的重要使命。

通过本书的学习，读者可以理解和掌握结构化的编程思想，掌握 C 语言的程序结构、语法规则和编程方法，培养独立编写常规 C 语言应用程序的能力，同时为设计大型应用程序和系统程序奠定坚实的基础。此外，还可培养使用计算机解决实际问题的计算思维能力和创新意识。结合本书中所承载的育人功能和蕴含的思政元素，培养“工匠精神”。本书是“数据结构”、“面向对象程序设计”、“操作系统”和“软件工程”等课程的基础，并可为这些课程提供实践工具。

本书内容共 13 章：第 1 章概述 C 语言程序设计的基础内容，帮助读者建立对 C 语言的整体认知；第 2 章详细讲解数据、运算与表达式，夯实编程基础；第 3～5 章聚焦程序的基本结构，讨论算法的表示和结构化程序设计的基本方法，以及顺序、选择和循环三种程序基本结构和简单 C 程序设计方法；第 6、7 章深入剖析指针、数组与函数，这些是 C 语言的核心要素；第 8、9 章介绍结构体与共用体、指针与链表等复杂数据类型，拓展编程能力；第 10 章介绍编译预处理和位运算及混合编程，为嵌入式系统、底层驱动等领域的开发提供支撑；第 11 章讲解文件操作，实现数据的持久化存储；第 12 章通过诗词信息管理、智能停车场管理两个综合案例分析，展示 C 语言在智能系统开发中的应用；第 13 章介绍多种主流开发和调试环境，助力读者快速进行实践。

“C 语言”是一门实践性很强的课程，对读者的编程和调试能力的训练非常重要。在第 1 章和第 13 章中，专门介绍了 C 语言的上机步骤和 C 程序的调试技术，并介绍了多种主流开发环境，如 Visual C++ 6.0 环境、Dev-C++环境、Visual Studio Code 环境、Linux 下 C 程序开发环境、Arduino 环境和移动端环境等。全部代码都在 Visual C++ 6.0 环境下编写调试。

本书是作者根据多年从事 C 语言教学的经验编写的，在第二版教材基础上，根据广大读者使用中提出的要求和意见，进行了全面修改，增加了核心知识点视频讲解、典型题例、综合案例分析与实现、C 语言混合编程、主流开发环境等内容。将“计算思维”的概念引入本书，

以典型题例引入问题，以编程应用为驱动，重点讲解程序设计的思想和方法。同时，给出两个较大规模的智能系统开发综合案例，培养读者对复杂工程问题的分析能力和解决能力。

本书由王曙燕担任主编，王春梅担任副主编，王小银参与编写。具体编写分工如下：王曙燕编写第 1、3、4、5 章；王小银编写第 2、6、7、13 章；王春梅编写第 8、9、10、11、12 章；全书由王曙燕统稿。孙家泽教授对本书的编写提出了宝贵建议，作者在此表示衷心的感谢。

本书在编写过程中参考了一些文献，在此谨向文献的作者致以诚挚的谢意。

由于作者水平有限，书中难免存在疏漏之处，恳请读者批评指正。

作者联系方式：wsylxj@126.com。

作　者

2025 年 12 月

目　录

第 1 章　概　　述

C 语言不仅具有丰富的运算符和数据类型，便于实现各类复杂的数据结构，还可以直接访问内存的物理地址，进行位（bit）一级的操作，因此说 C 语言集高级语言和低级语言的功能于一体。C 语言还具有生成目标代码质量高、执行速度快、可移植性好等特点。因此，C 语言既可用于系统软件的开发，也可用于应用软件的开发。可以说所有上层语言都离不开底层硬件的支持，离不开 C 语言的支持。C 语言能够在内存有限的大量硬件设备中执行，如嵌入式硬件设备以及性能关键型场景中。

本章首先从程序设计语言发展的角度概要介绍 C 语言的产生、标准与版本和特点，以期帮助读者在比较直观的基础上建立起对程序、程序设计、程序设计语言的基本认识，然后介绍 C 语言的基本语法成分和 C 语言程序的组成，并通过简单实例介绍 C 语言程序的基本结构和书写格式。通过本章的学习，读者将对 C 语言和程序设计有一个大概的了解，并能掌握程序设计的一般过程。

1.1　程序设计语言简介

C 语言概述

1946 年，世界上第一台电子数字式计算机 ENIAC（electronic numerical integrator and computer）在美国宾夕法尼亚大学诞生，开辟了一个计算机科学技术的新纪元。在此后短短的几十年间，计算机的发展突飞猛进。伴随着硬件的发展，计算机软件、计算机网络技术也得到了迅速的发展，计算机的应用领域也由最初的数值计算扩展到人类生活的各个领域，现在计算机已成为最基本的信息处理工具。人们使用计算机解决问题时，必须用某种“语言”来和计算机进行交流。具体地说，就是利用某种计算机语言提供的命令来编写程序，并把程序存储在计算机的存储器中，然后利用这个程序来控制计算机的运行，以达到解决问题的目的。这种用于编写计算机可执行程序的语言称为程序设计语言。目前已发明的计算机程序设计语言有上千种，无论什么样的程序设计语言，其程序设计的基本方法都是相同的。

1.1.1　程序设计语言的概念

程序设计语言是人们描述计算过程（即程序）的规范书写语言。程序是对计算机处理对象和计算规则的描述。语言的基础是一组记号和语法规则。根据语法规则由记号构成记号串的全体就是语言。

人类是使用像英语、汉语这样的自然语言相互交流和表达思想的。人和计算机如何“交流”呢？人和计算机交流也要用人和计算机都容易接受和理解的语言，这就是程序设计语言。程序设计语言是根据计算机的特点而编制的，是有限规则的集合。程序设计语言不像自然语言那样包含复杂的语义和语境，而是用语法来表达程序员的思想，所以编写程序时必须严格遵守语法规则。

1.1.2　程序设计语言的发展

计算机是一种具有内部存储能力、由程序自动控制的电子设备。用户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。当用户需要结果时，就向计算机发出一条简单的指令，计算机就按指令顺序自动进行操作。用户把这种可以连续执行的一条条指令的集合称为程序。也就是说，程序是计算机指令的序列，编写程序的工作就是为计算机安排指令序列。

程序设计语言伴随着计算机技术的发展层出不穷，从机器语言到高级语言，从面向过程的语言到面向对象的语言。到目前为止，程序设计语言的发展大致经历了五代。

第一代语言也称为机器语言，它将计算机指令中的操作码和操作数均以二进制代码形式表示，是计算机能直接识别和执行的语言。它在形式上是由“0”和“1”构成的一串二进制代码，每种计算机都有自己的一套机器指令。机器语言的优点是无须翻译、占用内存少、执行速度快；缺点是随机而异，通用性差，并且因指令和数据都是二进制代码形式，难以阅读和记忆，编码工作量大，难以维护。

第二代语言也称为汇编语言，是用助记符号来表示机器指令的符号语言。例如，用 ADD 表示加法，用 SUB 表示减法，用变量表示各类地址。用汇编语言编写的程序称为汇编语言源程序。这种程序必须经过翻译（称为汇编），变成机器语言程序，才能被计算机识别和执行。汇编语言在一定程度上克服了机器语言难以阅读和记忆的缺点，但对于大多数用户来说，仍然不便于理解和使用。

第三代语言也称为高级语言，是具有国际标准，描述形式接近自然语言的程序设计语言。它采用了完全符号化的描述形式，用类似自然语言的形式描述对问题的处理过程，用数学表达式的形式描述对数据的计算过程。常用的计算机高级语言有 BASIC、FORTRAN、COBOl、Pascal 和 C 等。由于高级语言只要求用户关心计算机描述问题的求解过程，而不关心计算机的内部结构，所以又把高级语言称为面向过程的语言。使用面向过程的语言编程时，程序员的主要精力放在算法过程的设计和描述上。

第四代语言也称为非过程化语言，是一种功能更强的高级语言。其主要特点是非过程性、采用图形窗口和人机对话形式、基于数据库和“面向对象”技术，并且易编程、易理解、易使用、易维护。但是，程序的运行效率和语言的灵活性不如过程化语言。常用的非过程化语言有 Java、C++、Python、Visual Basic 和 Delphi 等。

如果说第三代语言要求用户告诉计算机“怎么做”，那么第四代语言只要求用户告诉计算机“做什么”。

第五代语言也称为智能化语言，主要使用在人工智能领域，帮助用户编写推理、演绎程序。

目前，国内外大多数计算机上运行的程序是用第三代或第四代程序设计语言编写的。因此，应当熟练地掌握用高级语言编写程序的方法和技巧。

由于面向过程的语言是程序设计的基础，可以说 C 语言是 C++、Java 和 C#语言的基础，还有很多专用语言也学习和借鉴了 C 语言，如进行 Web 开发的 PHP 语言、做仿真的 MATLAB 的内嵌语言等。现在流行的操作系统 Linux 的内核就是用 C 语言编写的。因此，本书将以面向过程的 C 语言为背景，介绍程序设计的基本概念和基本方法。虽然面向对象的语言对 C 语言形成挤压之势，但学好 C 语言对以后学习其他语言大有帮助。计算机的发展日新月异，唯

有掌握最基础的知识，才能做到举一反三，触类旁通，以不变应万变。

1.2　C 语言简介

1.2.1　C 语言的产生

C 语言是国际上流行的、使用最广泛的高级程序设计语言之一。它既可用来写系统软件，也可用来写应用软件。C 语言具有语言简洁紧凑、使用方便灵活及运算符丰富等特点，它具有现代化语言的各种数据结构和结构化的控制语句，并且语法限制不太严格，程序设计自由度大，能实现汇编语言的大部分功能。另外，C 语言生成目标代码质量高，不仅程序执行效率高，而且程序可移植性好。

C 语言的产生基于两个方面的需要：一是 UNIX 操作系统开发的需要，UNIX 操作系统是一个通用的、复杂的计算机管理系统；二是接近硬件的需要，即直接访问物理地址、直接对硬件进行操作的需要。C 语言集高级语言与汇编语言的优点于一身。C 语言面对实际应用的需要而产生，直至今日仍不改初衷。C 语言是从 BCPL（basic combined programming language）和 B 语言演化而来的。1960 年出现的 ALGOL 是一种面向问题的高级语言，远离硬件，但不适用于开发系统软件。1963 年，英国剑桥大学推出 CPL，CPL 比 ALGOL 接近硬件一些，但规模较大，难以实现。1969 年，剑桥大学的 M. Richards 对 CPL 进行简化，推出 BCPL。1970 年，贝尔实验室的 K. Thompson 为 DEC 公司 PDP-7 计算机上运行的一种早期 UNIX 操作系统设计了一种类 BCPL，称为 B 语言。B 语言规模小，接近硬件，1971 年在 PDP-11 计算机上实现，并编写了 UNIX 操作系统和绝大多数实用程序。由于 B 语言面向字存取、功能过于简单、数据无类型和描述问题的能力有限，而且编译程序产生的是解释执行的代码，执行速度慢，因此没有流行起来。1972～1973 年，贝尔实验室的 D. M. Ritchie 在保留 B 语言优点的基础上，设计出一种新的语言。这种新语言克服了 B 语言面向字存取、功能过于简单、数据无类型和描述问题的能力有限的缺点，扩充了很多适用于系统设计和应用开发的功能。因为这种新语言是在 B 语言的基础上开发出来的，不管是在英文字母序列中，还是在 BCPL 这个名字中，排在 B 后面的均为 C，所以将这种语言命名为 C 语言。1973 年，UNIX 操作系统被用 C 语言改写，称为 UNIX 第五版。最初的 C 语言只是一种 UNIX 操作系统的工作语言，依附于 UNIX 系统，主要在贝尔实验室内部使用。

UNIX 之后的第六版、第七版、System Ⅲ和 System Ⅴ都是在第五版的基础上发展起来的，C 语言也做了多次改进。1975 年，随着 UNIX 第六版的公布，C 语言越来越受到人们的关注。

UNIX 操作系统的广泛使用促进了 C 语言的迅速发展与普及。同时，C 语言的发展与普及也促进了 UNIX 操作系统的推广。1978 年，出现了独立于 UNIX 和 PDP 计算机的 C 语言，从而 C 语言被迅速移植到大、中、小与微型机上。当年，B. W. Kernighan 和 D. M. Ritchie 以 UNIX 第七版的 C 编译程序为基础，出版了影响深远的名著《C 程序设计语言》。

1989 年，ANSI 发布了第一个完整的 C 语言标准——ANSI X3.159-1989，简称 C89，也称为 ANSI C。ISO 官方给予的名称为 ISO/IEC 9899，所以 ISO/IEC 9899:1990 也通常简称 C90。1999 年，在做了一些必要的修正和完善后，ISO 发布了新的 C 语言标准，命名为 ISO/IEC 9899:1999，简称 C99。2011 年 12 月，ISO 又正式发布了新的标准，称为 ISO/IEC 9899:2011，简称 C11。截止到 2026 年 3 月，C 语言的最新标准版本是 C23，该标准于 2023 年 11 月被 ISO

正式批准。

1.2.2　C 语言的标准与版本

随着 C 语言的普及，各机构分别推出了自己的 C 语言版本。某些执行过程的微小差别不时引起 C 程序之间的不兼容，给程序的移植带来很大的困难。美国国家标准学会（ANSI）从 1983 年开始，经过多年的努力，制定了 C 语言的新标准——ANSI C。现在提及 C 语言的标准就是指该标准。ANSI C 比原标准 C 有很大的改进，解决了经典定义中的二义性，给出了 C 语言的新特点。任何 C 程序都必须遵循 ANSI C 标准，本书也以 ANSI C 为基础。尽管这样，各种版本的 C 语言编译系统还是略有差异，因此在使用具体的 C 语言编译系统时，还应参考相关的手册以了解具体的规定。

支持 C 语言的编译器版本众多，C 语言开发环境及编译器版本丰富且持续迭代。微软（Microsoft）推出的 Windows 平台开发工具已更新至 Microsoft Visual Studio 2026（全面兼容 C11/C17/C23 标准，替代 VC6.0、VS2010/2019 等版本）；Borland 的 BC3.1、BC5.0 已被淘汰，主流替代为开源跨平台的 Code::Blocks；轻量级 Windows IDE 仍保留 Dev-C++，适配入门教学；Linux 平台核心编译器仍为 GNU Compiler Collection（GCC）（最新版支持 C23）。当前高校常用工具为入门级 Dev-C++、Code::Blocks，跨平台通用型 VS Code，进阶开发用 Visual Studio 2022、CLion 等。

1.2.3　C 语言的特点

C 语言有如下几方面的优点。

（1）简洁紧凑，编写的程序短小精悍。C 编译程序的代码量较少，便于在微型机上应用。

（2）运算符丰富，数据类型丰富。C 的数据类型有整型、实型、字符型、数组类型、指针类型、结构体类型和共用体类型等，能实现各种复杂数据类型的运算，并引入了指针概念，使程序执行效率更高。另外，C 语言具有强大的图形功能。

（3）是一种结构化程序设计语言，具有结构化语言所要求的三种基本结构。这种结构化方式可使程序层次清晰，便于使用、维护和调试。C 语言是以函数形式提供给用户的，这些函数可方便地调用，并具有多种循环、条件语句，用于控制程序流向，从而使程序完全结构化。

（4）允许直接访问物理地址。C 语言能进行位运算，能实现汇编语言的大部分功能，能直接对硬件进行操作。这使 C 语言既具有高级语言的功能，又具有低级语言的许多特点，可以用来写系统程序。

（5）预处理机制。C 语言提供预处理机制，有利于大程序的编写和调试。

（6）可移植性好。C 语言编写的程序不需要做很多改动就可从一种机型上移到另一种机型上运行，并且 C 语言适用于多种操作系统，如 DOS、UNIX。

（7）语法限制不太严格，程序设计自由度大，对程序员要求不高。一般的高级语言语法检查比较严格，能够检查出几乎所有的语法错误，而 C 语言则宽松得多。

（8）程序生成代码质量高，程序执行效率高。一般地，C 程序只比汇编程序生成的目标代码效率低 10%～20%。

C 语言也存在某些缺点，如运算符较多、某些运算符优先顺序与习惯不完全一致、类型转换比较随便等。

1.3　C 语言的基本语法成分

C 语言的基本语法成分

1. 字符集

字符是可以区分的最小符号，是构成程序的基础。C 语言字符集是 ASCII 字符集的一个子集，包括英文字母、数字及特殊字符。

（1）英文字母：a～z 和 A～Z。

（2）数字：0～9。

（3）特殊字符：␣、!、#、%、^、&、*、—、–、+、=、～、<、>、/、\、|、.、,、:、;、?、′、″、(、)、[、]、{、}。

字符集中的字符可以构成 C 语言的语法成分，如标识符、运算符等。

2. 标识符

标识符在程序中用来标识各种程序成分，命名程序中的一些实体，如变量、常量、函数、类型和标号等对象的名字。

C 语言规定，合法的标识符必须由英文字母或下划线开头，是字母、数字和下划线的序列，不能跨行书写，自定义的标识符不能与关键字同名。

以下是合法的标识符：

y, d1, wang, count_1, radius, _1, PI

以下是不合法的标识符：

b.1, 2sum, a+b, !abc, 123, π, 3–c

在 C 语言中，大写字母和小写字母被认为是两种不同的字符，因此标识符 SUM 与标识符 sum 是不同的。习惯上，符号常量名用大写字母表示，变量名用小写字母表示。

在标准 C 中，标识符的长度（即一个标识符允许的字符个数）可以是任意的。一般的计算机系统规定前 8 个字符有效，如果多于 8 个字符，则多余的字符将不被识别。例如，name_of_student 和 name_of_teacher，就被认为是同一个标识符。

C 语言的标识符分为以下三类。

1）关键字

关键字又称为保留字，是 C 语言中用来表示特殊含义的标识符，由系统提供，是构成 C 语言的语法基础。

C 语言的关键字有 32 个，它们是：

auto, break, case, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, int, long, register, return, short, signed, sizeof, static, struct, switch, typedef, union, unsigned, void, volatile, while

关键字有特定的语法含义，不允许用户重新定义。关键字在程序中不允许随意书写，绝对不能拼错。关键字也不能用作变量名或函数名。

2）预定义标识符

C 语言预先定义了一些标识符，它们有特定的含义，通常用作固定的库函数名或编译预处理中的专门命令，如 C 语言提供的库函数名 scanf、printf、sin 等，编译预处理命令 define、

include 等。C 语言语法允许用户标识符与预定义标识符同名，但这将使这些标识符失去系统规定的含义。为了避免引起误解，建议用户为标识符取名时不要与系统预先定义的标准标识符（如标准函数）同名。

3）用户标识符

用户标识符是由用户自己定义的标识符，一般用来给变量、函数、数组或文件等命名，命名时应遵守标识符的命名原则，另外应尽量做到“见名知义”，以提高程序的可读性。一般选用相应英文单词或拼音缩写的形式，如求和 sum，而尽量不要使用简单代数符号，如 x、y、z 等。

如果用户标识符与关键字相同，程序在编译时将给出出错信息；如果与预定义标识符相同，系统编译正确，只是该预定义的标识符失去原定含义，代之以用户确认的含义，或者运行时发生错误。

3. 运算符

运算符实际上可以认为是系统定义的函数名字，这些函数作用于运算对象，得到一个运算结果。运算符通常由 1 个或多个字符构成。

根据运算对象的个数不同，运算符可分为单目运算符（如!、～、++、––和*）、双目运算符（如+、–、*和/）、三目运算符（如?和:）和其他，其中前三种又分别称为一元运算符、二元运算符和三元运算符。C 语言运算符非常丰富，详见第 2 章介绍。

1.4　C 语言程序的组成

1.4.1　简单的 C 语言程序介绍

下面通过介绍几个简单的 C 语言程序，来了解 C 语言程序的基本结构。

【例 1.1】　在计算机屏幕上输出一行文字：“Hello,world!”。

```
/* 该程序在屏幕上输出 Hello,world! */
#include <stdio.h>
int main()
{ printf("Hello,world!\n");
  return 0;
}
```

运行结果：

```
Hello,world!
```

该程序非常简单，其中 main()表示“主函数”，函数体由一对花括号“{}”括起来，表示函数的开始和结束。

本例中主函数内只有一条输出语句，即标准输出库函数 printf()（详见第 3 章）。“\n”是转义字符常量，表示回车换行，即在输出“Hello,world!”后回车换行，使屏幕上的光标在新行的行首。return 0;表示程序正常结束，返回整数 0（返回非 0 值通常表示程序出错）。

程序的第 1 行为注释行，注释行以“/*”开头，以“*/”结尾，其间的内容为程序员对程

序的注解，可以出现在程序的任何地方，用来说明程序段的功能和变量的作用，以及程序员认为应该向程序阅读者说明的任何内容。注释可以单独占一行，也可以跟在语句后面，如果注释内容超出一行，可以另起一行继续写。在将 C 程序编译成目标代码时，所有的注释都会被忽略掉。因此，即使使用了很多注释，也不会影响目标代码的效率。恰当地使用注释可以使程序清晰易懂，便于阅读和调试。在编写程序时，应养成撰写注释的良好习惯。但要注意，注释不能嵌套。

【例 1.2】　求三个整数之和。

```
/* 求三个整数之和 */
#include <stdio.h>
int main()
{ int x,y,z;
  int sum;
  printf("请输入三个整数 x,y,z:");
  scanf("%d,%d,%d",&x,&y,&z);
  sum=x+y+z;
  printf("sum=%d\n",sum);
  return 0;
}
```

运行结果：

```
请输入三个整数 x,y,z:3,5,8
sum=16
```

程序的第 1 行为注释行，说明本程序的作用。第 2 行是一条编译预处理命令，在编译程序之前，凡是以“#”开头的代码行，都要由预处理程序处理。该行是通知预处理程序把标准输入输出头文件 stdio.h 中的内容包含到该程序中。头文件 stdio.h 中包含了编译器在编译标准输出函数 printf()时要用到的信息和声明，还包含了帮助编译器确定对库函数调用的编写是否正确的信息。在 C 语言中，如果仅用到标准输入输入输出函数 scanf()和 printf()，可以省略该命令，C 语言默认包含，但建议每个用到标准输入输入输出函数的程序均写上该命令。关于编译预处理命令详见第 10 章。第 4 和 5 行是两条变量说明语句。在变量说明语句中，int 是一个关键字，说明后面的标识符 x、y、z 和 sum 是整型变量。

第 6 行是一条标准输出语句，在屏幕上显示“请输入三个整数 x,y,z:”的提示语。第 7 行是一条标准输入语句，scanf 是输入函数的名字，该函数的作用是输入三个整数值到变量 x、y 和 z 中。&x 中“&”的含义是“取地址”，也就是说将三个整数值分别输入到变量 x、y 和 z 的地址所标志的单元中，即输入给变量 x、y、z。“%d”是输入的“格式字符串”，用来指定输入输出时的数据类型和格式，“%d”表示“十进制整数”。

第 8 行是一条赋值语句，先计算表达式 x+y+z 的值，然后将计算结果赋给变量 sum。第 9 行是标准输出语句，其中的函数 printf()有两个参数，分别为 sum=%d\n 和 sum。第一个参数表示输出格式的控制信息，表示“sum=”照原样输出，类似“%d”的输出转换说明的位置用后面相应参数的值按指定的类型和格式依次代替输出。“\n”是换行符，在屏幕上没有显示，只是使光标移到下一行。

【例 1.3】　求三个数的平均值。

```
/* 该程序求三个数的平均值,这是一个自定义函数示例程序 */
#include <stdio.h>
/* 定义函数 average() */
float average(float x, float y, float z)
{ float aver;            /* 定义变量 aver 类型为单精度实型 */
  aver=(x+y+z)/3;        /* 求三个数的平均值 */
  return(aver);          /* 返回 aver 值,通过 average()带回调用处 */
}
int main()               /* 主函数 */
{ float a,b,c,ave;
  a=6.5;
  b=4.2;
  c=25.4;
  ave=average(a,b,c);            /* 调用自定义函数 average() */
  printf("average=%f",ave);      /* 输出 a,b,c 的平均值 */
  return 0;
}
```

运行结果：

```
average=12.033333
```

在这个程序中，共定义了两个函数：一个是自定义函数 average()；另一个是主函数 main()。程序的第 4～8 行定义了函数 average()，第 9～17 行定义了函数 main()。

程序的第 4 行是函数 average()的首部说明。因为在 average 后紧跟一对圆括号，说明 average 是一个函数名。在 average 的前面有一个类型关键字 float，说明该函数的返回值类型为单精度实型。在 average 后面的圆括号中是“形参表”，说明该函数有三个形式参数 x、y 和 z。

main()函数的第 6 行为调用 average()函数，在调用时将实际参数 a、b 和 c 的值分别一一对应地传给 average()函数中的形式参数 x、y 和 z。经过执行 average()函数得到一个返回值（即三个数的平均值），然后把这个值赋给变量 ave，最后输出 ave 的值。

这里用到了函数调用等概念，读者可能不大理解，等到学习后面的章节时问题自然迎刃而解。

1.4.2　C 语言程序的基本结构

由以上几个例子可以看到，C 语言程序的基本结构如下：

```
头文件                  /* C 系统中特有的文件或用户自定义的文件 */
全局变量说明            /* 用于定义在整个程序中有效的变量 */
int main()              /* 主函数说明 */
{ 局部变量说明          /* 主函数体 */
  执行语句组
}
子函数 1(参数)          /* 子函数说明 */
{ 局部变量说明          /* 子函数体 */
```

```
    执行语句组
}
子函数 2(参数)          /* 子函数说明 */
{ 局部变量说明          /* 子函数体 */
    执行语句组
}
…
子函数 n(参数)          /* 子函数说明 */
{ 局部变量说明          /* 子函数体 */
    执行语句组
}
```

其中，子函数 1～n 是用户自定义的函数。

由此可见，C 语言是一种函数式语言，它的一个函数实际上就是一个功能模块。一个 C 程序是由一个固定名称为 main 的主函数和若干个其他函数（可没有）组成的。

注意：一个 C 程序必须有一个也只能有一个主函数。主函数在程序中的位置可以任意，但程序执行时总是从主函数开始，并且在主函数内结束。主函数可以调用其他各种函数（包括用户自己编写的），但其他函数不能调用主函数。

函数是由函数说明和函数体两部分组成的。函数说明部分包含对函数名、函数类型和形式参数等的定义和说明；函数体部分包括局部变量说明和执行语句组，由一系列语句和注释组成。整个函数体由一对花括号括起来。

说明部分用于变量的类型定义，简单程序可以没有此部分，如例 1.1。

语句由一些基本字符和定义符按照 C 语言的语法规定组成，每条语句必须用分号“;”结束（注意，是“每条语句”而不是“每行语句”），编译预处理命令不是语句（行末不能用分号结束）。

C 语言本身没有输入输出语句，其输入输出功能须通过调用标准函数来实现。使用系统提供的标准函数或其他文件提供的函数时，必须使用“文件包含”（除了 printf 和 scanf 语句）。

1.4.3　C 语言程序的书写格式

C 语言程序书写格式自由，但一般应遵循以下原则，以使程序结构清晰，便于阅读。

（1）一行一般写一条语句。当然，一行可以写多条语句，一条语句也可以分写在多行上。

（2）整个程序采用缩进格式书写。表示同一层次的语句行对齐，缩进同样多的字符位置。例如，循环体中的语句要缩进对齐，选择体中的语句也要缩进对齐。

（3）花括号对齐的书写方式。花括号的书写方式较多，本书采用左边花括号处于第一条语句的开始位置，右边花括号独占一行，与左边花括号对齐的书写方式。

（4）在程序中恰当地使用空行，分隔程序中的语句块，增加程序的可读性。

下面给出两个例子，使读者对 C 语言程序的基本结构及书写格式有进一步的了解。

【例 1.4】　求两个数的较大值。

```
#include <stdio.h>     /* 编译预处理——文件包含(标准输入输入输出函数) */
int main()
{ int a,b;
  printf("请输入两个整数 a,b:");
```

```
    scanf("%d,%d",&a,&b);
    if(a>b)
      printf("%d",a);  /* 如果 a 大于 b,则输出 a 的值 */
    else
      printf("%d",b);  /* 否则,输出 b 的值 */
    return 0;
}
```

【例 1.5】 求圆面积程序。

```
/* 求圆面积程序 area.c */
#define  PI  3.14159          /* 编译预处理——宏替换 */
#include <stdio.h>            /* 编译预处理——文件包含(标准输入输入输出函数) */
#include <math.h>             /* 编译预处理——文件包含(数学函数) */
#include <stdlib.h>           /* 编译预处理——文件包含(常用函数) */
#include <conio.h>            /* 编译预处理——文件包含(文本窗口函数) */
int main()
{ float r,s;
  system("cls");              /* 清屏,在 conio.h 中定义 */
  printf("请输入半径 R=");     /* 人机对话提示语 */
  scanf("%f",&r);             /* 将键盘输入值存放在变量 r 对应的存储单元中 */
  if(r<0)                     /* 如果输入的半径值为负值 */
  { printf("输入出错,半径不能为负值!");          /* 显示出错提示 */
    exit(0);                  /* 停止程序执行,返回操作系统 */
  }
  s=PI*pow(r,2);
  printf("半径 R=%.3f 时,面积 S=%.3f\n",r,s);  /* 限制 R、S 小数位数*/
  return 0;
}
```

1.4.4　宏定义和条件编译

宏定义和条件编译是 C 语言预处理阶段的重要特性，用于在编译前对源代码进行处理，提升代码的灵活性和可维护性。

宏定义是用一个标识符（宏名）代表一段文本的预处理指令，分为无参数和带参数两种形式。

（1）无参数宏，一般形式如下：

```
#define  宏名  字符串
```

如例 1.5 中#define　PI　3.14159，预处理器会将代码中所有宏名替换为对应字符串，常用于定义常量或重复使用的代码片段。这个替换过程称为“宏替换”或“宏展开”，字符串也称为替换文本。宏定义的作用域是从定义处开始到源文件结束，但根据需要，可用 undef 命令终止其作用域，一般形式为：#undef　宏名。

（2）带参数宏，一般形式如下：

```
#define  宏名(形参表)  字符串
```