免费提供电子教案

高等院校规划教材
计算机科学与技术系列

C语言程序设计

主　编　曹　哲　刘　军
副主编　葛建梅　张凤君
参　编　刘太辉　薛曼玲　曹晶人　张润刚

机械工业出版社
CHINA MACHINE PRESS

本书以软件工程方法学为指导，以结构化、模块化程序设计方法为主线，由浅入深、循序渐进地介绍C语言的语法和程序设计的基本方法，详细介绍了算法设计过程。

本书共12章。其中第1章为C语言程序设计概述，第2章为C语言的基础知识，第3、4、5章分别为顺序、选择、循环3种基本结构的程序设计，第6章为指针的初步知识，第7章为数组，第8章为模块化程序设计，第9章为编译预处理命令，第10章为结构体与共用体，第11章为位运算，第12章为文件。

本书基础知识部分内容细致、准确，程序设计方法规范、实用，内容与习题配套，附录中给出了各章习题的参考答案。本书配套辅助教材《C语言实验与课程设计指导》也已在机械工业出版社出版。本书可作为高等院校和大、中专院校计算机专业以及理工科各专业“C语言程序设计”课程的主讲教材，也可作为自学用书或相关技术人员的参考书。

本书配套授课电子课件，需要的教师可登录www.cmpedu.com免费注册、审核通过后下载，或联系编辑索取（QQ：2399929378，电话：010-88379750）。

图书在版编目（CIP）数据

C语言程序设计／曹哲主编；刘太辉等编．—北京：机械工业出版社，2012.9
高等院校规划教材·计算机科学与技术系列
ISBN 978-7-111-40111-7

Ⅰ．①C…　Ⅱ．①曹…　②刘…　Ⅲ．①C语言－程序设计－高等学校－教材　Ⅳ．①TP312

中国版本图书馆CIP数据核字（2012）第246322号

机械工业出版社（北京市百万庄大街22号　邮政编码100037）
策划编辑：郝建伟
责任编辑：郝建伟　王　凯
责任印制：乔　宇

保定市中画美凯印刷有限公司印刷

2013年1月·第1版第1次印刷
184mm×260mm·24.25印张·601千字
0001—3000册
标准书号：ISBN 978-7-111-40111-7
定价：49.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务	网络服务
社服务中心：（010）88361066	教材网：http://www.cmpedu.com
销售一部：（010）68326294	机工官网：http://www.cmpbook.com
销售二部：（010）88379649	机工官博：http://weibo.com/cmp1952
读者购书热线：（010）88379203	**封面无防伪标均为盗版**

出 版 说 明

计算机技术的发展极大地促进了现代科学技术的发展，明显加快了社会发展的进程。因此，世界各国都非常重视计算机教育。

近年来，随着我国信息化建设的全面推进和高等教育的蓬勃发展，高等院校的计算机教育模式在不断改革，计算机学科的课程体系和教学内容趋于更加科学和合理，计算机教材建设也逐渐成熟。在“十五”期间，机械工业出版社组织出版了大量计算机教材，包括“21 世纪高等院校计算机教材系列”、“21 世纪重点大学规划教材”、“高等院校计算机科学与技术‘十五’规划教材”、“21 世纪高等院校应用型规划教材”等，均取得了可喜成果，其中多个品种的教材被评为国家级、省部级的精品教材。

为了进一步满足计算机教育的需求，机械工业出版社策划开发了“高等院校规划教材”。这套教材是在总结我社以往计算机教材出版经验的基础上策划的，同时借鉴了其他出版社同类教材的优点，对我社已有的计算机教材资源进行整合，旨在大幅提高教材质量。我们邀请多所高校的计算机专家、教师及教务部门针对此次计算机教材建设进行了充分的研讨，达成了许多共识，并由此形成了“高等院校规划教材”的体系架构与编写原则，以保证本套教材与各高等院校的办学层次、学科设置和人才培养模式等相匹配，满足其计算机教学的需要。

本套教材包括计算机科学与技术、软件工程、网络工程、信息管理与信息系统、计算机应用技术以及计算机基础教育等系列。其中，计算机科学与技术系列、软件工程系列、网络工程系列和信息管理与信息系统系列是针对高校相应专业方向的课程设置而组织编写的，体系完整，讲解透彻；计算机应用技术系列是针对计算机应用类课程而组织编写的，着重培养学生利用计算机技术解决实际问题的能力；计算机基础教育系列是为大学公共基础课层面的计算机基础教学而设计的，采用通俗易懂的方法讲解计算机的基础理论、常用技术及应用。

本套教材的内容源自致力于教学与科研一线的骨干教师与资深专家的实践经验和研究成果，融合先进的教学理念，涵盖计算机领域的核心理论和最新的应用技术，真正在教材体系、内容和方法上做到了创新。同时，本套教材根据实际需要配有电子教案、实验指导或多媒体光盘等教学资源，实现了教材的“立体化”建设。本套教材将随着计算机技术的进步和计算机应用领域的扩展而及时改版，并及时吸纳新兴课程和特色课程的教材。我们将努力把这套教材打造成为国家级或省部级精品教材，为高等院校的计算机教育提供更好的服务。

对于本套教材的组织出版工作，希望计算机教育界的专家和老师能提出宝贵的意见和建议。衷心感谢计算机教育工作者和广大读者的支持与帮助！

机械工业出版社

前　言

C 语言是目前使用最广泛的高级程序设计语言之一。“C 语言程序设计”课程是计算机及相关专业的一门基础主干课程。为了使读者更好地学习 C 语言，逐步掌握使用 C 语言进行结构化、模块化程序设计的方法，本书作者根据几十年程序设计教学的经验编写了本书。

本书具有以下特点。

1）以软件工程方法学为指导，以结构化、模块化程序设计方法为主线编写本书。

2）主要依据 Turbo C++ 3.0 集成开发环境，由浅入深、循序渐进地介绍了标准 C 语言的基础知识和程序设计的基本方法，重点介绍了结构化程序设计方法的步骤和算法的设计过程。

3）难点分散。以地址和指针这一难点为例，在第 2 章即引入了地址的概念，在第 6 章引入指针的概念，而在其后的 6 章中结合第 6 章内容由浅入深地介绍各种指针的概念和用法，从而达到深入理解、灵活运用的目的。

4）本书概念准确，内容丰富，书中每章后都配有和本章内容配套的习题。其中，填空题和单项选择题可加深对语法格式和概念的理解，这是程序设计的基础；读程序题可培养读者读懂别人编写的程序和理解其算法的能力，这是提高编程能力的重要手段之一；编程序题可培养读者的程序设计能力。此外还有程序改错、表达式求值等其他几种题型。

5）附录中给出了各章习题的参考答案。本书还配有辅助教材《C 语言实验与课程设计指导》。

本书可作为高等院校和大、中专院校“C 语言程序设计”课程的主讲教材，也可作为自学用书及相关技术人员的参考书。

本书由曹哲、刘军任主编，由葛建梅、张凤君任副主编，全书由曹哲统稿和定稿。编写分工如下：刘太辉编写第 1、2 章，葛建梅编写第 3、4 章，曹晶人编写第 5 章，薛曼玲编写第 6、7 章，刘军编写第 8 章，张凤君编写第 9、10、11 章，张润刚编写第 12 章，附录由编者们共同编写。

在编写本书的过程中，北华大学计算机科学技术学院“程序设计基础课程组”、实验室和 4 个系的全体教师为上机调试程序、稿件校对等工作给予了大力的支持和帮助；贺薪宇为本书制作了电子课件。在此对上述部门和相关人员的贡献表示衷心的感谢。

由于作者水平有限，书中难免有疏漏、不足之处，欢迎读者批评指正。

编　者

目　录

第1章　C语言程序设计概述

本章主要介绍了计算机程序和语言的概念、C 语言的发展简史、C 语言的特点，并通过几个简单的C程序例子介绍了C程序的结构，然后介绍了结构化程序设计方法，最后介绍了上机运行C程序的方法。在学习本章时，对于所介绍的几个C程序例子，不要急于读懂程序，应注意C程序的结构、初步了解结构化程序设计的方法和步骤。

1.1　计算机的程序和语言

语言是人们交流思想的工具。人们通过语言来描述事物，表达自己的思想。同样，人们要让计算机解决自己的问题，就必须使计算机懂得人的意图，接受人输入的数据和命令等信息。这些信息都是通过计算机语言来交换的。因此，计算机语言是人和计算机交换信息的工具。到目前为止，计算机语言主要有机器语言、汇编语言、高级语言、第四代语言——超高级语言、第五代语言——自然语言等几大类。

1．计算机的机器语言

计算机虽然有惊人的本领，但要使它工作，还要由人事先给它安排好工作步骤，计算机按照这些步骤的顺序，一步一步有条不紊地执行，才能完成人们所要求的任务。由于计算机硬件本身只能识别0和1这两个数码所表示的二进制数，所以在计算机中，0和1数码的组合不仅用来表示数据、符号，而且也用来规定计算机所进行的操作（如加、减、乘、除等）的命令。例如，在IBM PC中的一条加法命令：

```
0000000111011000
```

这条加法命令的功能是将寄存器AX和寄存器BX中的两数相加，并把加得的和存入寄存器AX中。

这种由0和1组成的数字代码是规定计算机进行某种操作的命令，我们称其为机器指令。一台计算机的机器指令的集合就是这台计算机的指令系统。指令系统就是计算机的机器语言。机器语言随着计算机的诞生而诞生，是计算机的第一代语言，或称低级语言。

人们根据要解决的问题用某种计算机语言为计算机安排的计算和操作步骤叫做程序。用机器语言编写程序就是用一系列机器指令为计算机安排工作步骤。编写程序的过程叫程序设计。从事程序设计的人员叫做程序员。用机器语言编写的程序常称为手编程序。

机器语言难学、难记，手编程序直观性差、难编、难检查、难修改。编好的程序如同天书，很难读懂。不仅如此，由于不同型号的计算机的指令系统截然不同，所以机器语言是不通用的。但机器语言的“优点”是编写的程序可以直接在机器上运行。

2．计算机的汇编语言

为了提高编程效率，减轻人的劳动，在20世纪50年代初就出现了汇编语言。所谓汇编语言，就是把机器指令代码用规定的便于记忆的符号来代替。因此汇编语言又称符号语言。

仍以上面的一条加法指令为例，写成汇编语言的形式为：

ADD AX , BX

这就是一条符号指令，也是将寄存器 AX 和寄存器 BX 中的两数相加，并把结果回送到 AX 中。汇编语言比机器语言易学易记。用汇编语言编程大大提高了效率，出错率也大为减少。然而，计算机只能直接执行机器指令，因此用汇编语言编写的程序计算机并不认识，也无法直接执行。为了使计算机能认识汇编语言，需要进行翻译。翻译工作由汇编程序（用机器指令写成的程序）来完成，它专门用来将汇编语言写成的程序翻译成机器语言程序。我们把用户用汇编语言写成的用来解决自己问题的程序称为汇编语言源程序，简称源程序，以便和担当翻译的汇编程序相区别。

用汇编语言上机的过程是：事先将汇编程序输入到计算机中担当翻译，再把汇编语言源程序输入到计算机中，然后启动汇编程序将源程序翻译成机器指令形式的程序（称为目标程序），最后执行目标程序便可得到结果，如图 1-1 所示。

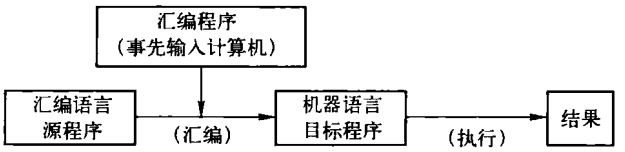

图 1-1　用汇编语言上机的过程

汇编语言的指令和机器指令是一一对应的，因此它与机器语言一样，都是为特定的机器服务的，称为面向机器的语言。汇编语言与人们习惯用的语言差别很大，自动化程度还很低，是计算机的第二代语言或称中级语言。

3．计算机的高级语言

为了克服机器语言和汇编语言的缺点，50 年代末以来，计算机科学家又相继发明了各种高级语言。高级语言很接近人们习惯用的语言和数学语言。高级语言允许用英文单词写解题程序，所用的运算符、运算式和数学公式差不多。C 语言就是高级语言的一种。比如，要想计算 $5\times\sin(\pi/5)$，只要写出一个表达式 “5 * sin(3.1415926 / 5)”，就可以计算出结果。很明显，它与数学公式几乎一致。所以用高级语言编程是比较方便的。有了高级语言，人们就可以直接用它编写程序。用某种高级语言编写的程序称为高级语言源程序，也简称源程序。和汇编语言一样，要使计算机理解高级语言源程序的意图，也必须将源程序“翻译”成机器指令形式的程序，然后再让计算机执行。不同的高级语言采用的翻译方式不同，但归纳起来有两种做法，即编译方式和解释方式。

1）编译方式：把事先编好的称为“编译程序”的机器指令程序放在计算机中担当“翻译”，然后把用高级语言编写的源程序输入计算机，启动编译程序将源程序翻译成机器指令目标程序，之后执行该目标程序得到结果，如图 1-2 所示。

2）解释方式：把事先编好的称为“解释程序”的机器指令程序放在计算机中做“翻译”。当高级语言源程序输入计算机后，解释程序逐句地翻译源程序，译出一句便马上执行一句，即一边解释一边执行。它不像编译方式那样先把源程序整个翻译成目标程序，然后再执行目标程序。解释方式的上机过程如图 1-3 所示。

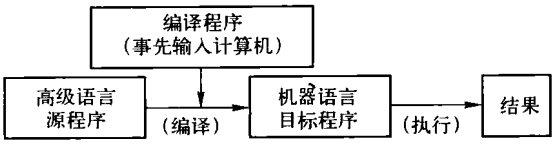

图 1-2　编译方式上机过程

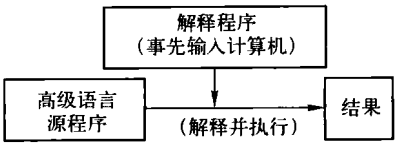

图 1-3　解释方式上机过程

这两种方式各有特点。其中编译方式可以节省计算机时间，而解释方式则可节省计算机存储空间。

FORTRAN、Pascal、C、C++、VB、VC++、C#、Ada 等高级语言采用编译方式，而 BASIC、Java 等计算机语言则基本采用解释执行方式。还有些语言，如 True BASIC 则两种方式均可采用。

高级语言对于使用计算机的人来说，只要有一点计算机的常识，经过短期学习就能掌握，就会编写程序在计算机上解决自己的问题。由于高级语言与计算机的内部逻辑结构无关，因此它可以适用于不同种类的计算机，通用性很强。用某种高级语言编写的源程序几乎可以不加修改或只做一些微小的修改就能在不同种类的计算机上使用，这极大地方便了用户。因此有人说，高级语言的出现是计算机发展史上“最惊人的成就”。

人们对计算机高级语言的研究已有 60 多年的历史了。在这一过程中，曾有过一千多种编程语言。到目前为止，重要的只有几十种，常用的仅十几种，且特点和适用范围也不同。比较通用的面向过程的高级语言有：ALGOL（算法语言）；FORTRAN（公式翻译），适于科学计算和数据处理；COBOL（通用商业语言），适于商业和经济管理；Pascal（结构化程序设计语言）；PL/1（大型通用语言）；C（系统程序设计语言），适于设计系统软件和大型应用软件；Ada 是直接体现软件工程方法学的一种语言，等。面向对象的高级语言有：C++，C#，VB（Visual Basic），VC++，PB（Power Builder），Delphi，Java 等。

高级语言是第三代语言（3GL）。

4．第四代计算机语言（4GL）

第四代语言是“面向问题”或“面向应用”的语言，即只需告诉计算机“做什么”，不需告诉计算机“怎么做”。目前，第四代语言正在发展之中，主要针对一些专门的应用。比如各种数据库的 SQL（Structured Query Language，结构化查询语言）、报表生成器、屏幕生成器等，都具有 4GL 的特征。

5．第五代计算机语言（5GL）

有人称第五代计算机语言为自然语言，也有人称其为知识库语言或人工智能语言。其目标是使其最接近于自然语言。目前，有人称 LISP 和 PROLOG 是第五代语言的萌芽。其实它们距离第五代语言还相当遥远，需要我们去探索。

1.2　C 语言的发展简史

C 语言是当代最流行的计算机语言之一，因为它是一个独立于机器、结构化的高级语言。它允许软件开发者（developers）开发程序而无需考虑其相应的硬件平台（hardware platforms）。

C 语言的根源（root）是 ALGOL 60 语言，它是在 20 世纪 60 年代初提出的。ALGOL 语言是第一个使用块结构的高级语言。计算机科学界提出的初步的结构化程序设计（structured

programming）的概念，由计算机科学家 Corrado Bohm、Guiseppe Jacopini 和 Edsger Dijkstra 在 60 年代进行了推广。

在 1963 年英国剑桥大学推出了 CPL（combined programming language，组合程序设计语言），它是在 ALGOL 60 的基础上开发的，但规模较大。

在 1967 年，英国剑桥大学的 Martin Richards 在 CPL 语言基础上，经过简化，开发出了 BCPL（Basic Combined Programming Language，基本组合程序设计语言），主要用于编写系统软件。

在 1970 年，美国贝尔（Bell）实验室的 Ken Thompson 对 BCPL 进行了进一步的提炼，创建了 B 语言（取 BCPL 的首字母）。在贝尔实验室，用 B 语言创建了 UNIX 操作系统的早期版本。BCPL 和 B 两者都是“无类型”（typeless）系统程序设计语言。

1972 年至 1973 年间，贝尔实验室的 D. M. Ritchie 在 B 语言的基础上推出了 C 语言（取 BCPL 的第二个字母）。C 语言保持了 BCPL 和 B 语言的精炼、靠近硬件等优点，并增加了数据类型的概念和其他强有力的特性。最初 C 语言只是为编写 UNIX 操作系统而开发的一种语言。比如，1973 年，Ken Thompson 和 D. M. Ritchie 合作将其原来用汇编语言开发的 UNIX 的 90%以上用 C 语言改写（即 UNIX 第 5 版）。随后，C 语言不断改进，但主要还是在贝尔实验室内部使用。这一时期的 C 语言现在被称为“传统 C”（traditional C）。

直到 1975 年 UNIX 第 6 版发布以后，随着许多商用 C 语言编译程序的发布和 UNIX 的日益流行，C 语言终于获得了计算机专业人士的广泛支持。

1978 年，由 Brian W. Kernighan 和 Dennis M. Ritchie 所著的《The C Programming Language》（C 程序设计语言）出版后，该语言变得更为流行。这本书中介绍的 C 语言成为后来广泛使用的 C 语言的基础，被称为“K&R C”。

C 的迅速发展导致该语言的不同版本不断推出，它们是类似的，但常常是不兼容的。这给系统开发者造成了一系列问题。

为了保证 C 语言的标准化，在 1983 年，美国国家标准化协会（American National Standards Institute，ANSI）指定专门的技术委员会来定义一个 C 标准，称为 83 ANSI C。1989 年，ANSI 公布了一个完整的 C 语言标准，即 ANSI X3. 159-1989，通常称为 ANSI C 或 C89。之后，在 1990 年，该标准被国际标准化组织（International Standards Organization，ISO）批准为国际的 C 标准，即 ISO C 标准。该标准在 1999 年被增订，成为新的国际标准，称为 C99。目前流行的 C 语言编译系统大多是以 ANSI C 为基础开发的。

本书的叙述主要以 ANSI C 为基础，书中的所有例题、习题的参考答案所涉及的 C 程序都是在 Turbo C++ 3.0 的集成开发环境下编译和运行的。

1.3 C 语言的特点

C 语言之所以能够得到广泛的使用，主要原因是和其他高级语言相比较，它具有突出的、不可取代的优点。C 语言的主要特点如下。

1. 支持结构化、模块化程序设计方法

C 语言具有良好的结构化的控制语句，比如具有顺序结构的语句（一般一条语句执行完都要接着执行下一条语句）、选择结构的语句（比如 if 语句、if-else 语句、switch 语句等）和

循环结构的语句（比如 while 循环语句、do-while 循环语句、for 循环语句等）。C 语言用函数来实现程序模块。也就是说，C 语言完全支持结构化、模块化的程序设计方法，这正是我们学习 C 语言程序设计要掌握的核心内容。

2．语言简洁、紧凑

C 语言总共只有 32 个关键字（keyword，详见附录 C）、9 种控制语句，比其他高级语言的语法更简练、源程序更短。也就是说，用 C 语言开发软件效率会更高。

3．数据类型丰富

C 语言提供了编程所需要的各种数据类型。主要有字符型、整型、浮点型、枚举类型、数组类型、结构体类型、共用体类型、指针类型、空类型等。丰富的数据类型可以表示各种复杂的数据结构。比如可以方便地实现堆栈、队列、树、图、链表等数据结构及其相关的算法。特别是指针类型，使用十分灵活有效，是 C 语言的精华。

4．运算符十分丰富

数据类型丰富，运算符也必然丰富，再加上 C 语言把括号、赋值、强制类型转换等都按照运算符来处理，使其表达式的种类多样化，运算类型也十分多样。C 语言共有 34 种运算符，分为 15 个优先级，有的从左往右运算（称左结合性），有的从右往左运算（称右结合性）。使用这些运算符可以实现其他高级语言难以实现的运算。

5．可直接对硬件进行操作

我们知道，机器语言和汇编语言都可以直接对硬件进行操作。C 语言能够实现汇编语言的大部分功能，也可以直接对硬件进行操作。比如 C 语言可以通过物理地址访问内存、能对二进制位（bit）进行操作、操作 CPU 内部的硬件、控制各种外围设备等。也就是说，C 语言既具有低级语言的绝大部分功能，又具有高级语言的强大功能。有人把 C 语言称为“高级语言中的低级语言”，也有人称其为“中级语言”。其实，C 语言是一种“带低级语言功能的高级语言”。因此，C 语言既是一种系统描述语言，可用于编写系统软件；又是一种通用的高级程序设计语言，适于编写应用软件。

6．生成目标代码质量高

C 语言编译系统采用了优化策略，其程序生成目标代码的效率一般仅比汇编语言低 10%～20%，程序执行效率高。

7．可移植性（portability）好

与汇编语言比较，用 C 语言编写的程序可移植性好。即一个 C 程序，基本不需要修改，或者只需稍加修改，就可以用于各种型号的计算机和各种操作系统中。

8．程序设计自由度大

一般的高级语言的语法检查比较严格，几乎能检查出所有的语法错误，而 C 语言则放宽了语法检查，允许程序员有较大的设计自由度。C 语言对数组下标越界不做检查，对变量和常量的数据类型的使用比较灵活，比如允许字符型、整型、逻辑型等数据通用等。语法限制不严格的优点增强了程序设计的灵活性、提高了容错能力、加强了处理能力；缺点是编写程序容易出错。当然，只要我们认真学习 C 语言的语法，正确利用 C 语言的灵活性，就可以编写出正确的、高质量的 C 程序。

总之，C 语言既是一种“面向机器”的语言，又是一种“面向过程”的通用高级语言。C 语言的上述特点使得它成为国际上最广泛流行的高级语言之一。它可以用来开发操作系统等

系统软件和各种应用软件。比如，目前，单片机系统和嵌入式系统中的软件都是用 C 语言开发的，因此，学好 C 语言，掌握结构化、模块化程序设计方法，就可以进一步学习各种实用的面向对象的高级语言和数据结构、数据库原理及应用、编译原理、操作系统等计算机专业的软件课程。

1.4 C 程序的结构

在本节中，我们首先介绍几个较简单的 C 语言程序，并从中总结出 C 程序的结构。这几个例子不要求读者深入理解，但要求了解 C 程序的组成和结构，例子中涉及的语法在以后各章节会慢慢介绍。

1．几个简单 C 程序的例子

【例 1-1】 运行如下的 C 程序。

```
#include <stdio.h>
int   main ( )
{
     printf ( "Good morning World!\n " ) ;
     return 0 ;
}
```

将在屏幕上输出如下一行信息：

```
Good␣morning␣World!
```

其中，“␣”表示一个“空格”，实际上只是空出一格。

该程序的第 2 行中的“main”是主函数的名字。main 函数是一个由 C 系统所使用的特殊的函数，它告诉系统该程序从这里开始执行。每一个 C 程序都必须有一个，且仅能有一个 main 函数。“main”前面的“int”表示该函数的类型是 int 类型，即整型，在执行完主函数后会得到一个整型的函数值。程序中的第 5 行“return 0 ;”是一个返回语句，其作用是当 main 函数执行结束前将整数 0 作为函数值返回到调用它的操作系统。返回 0 表示程序正常结束，如果程序执行异常或出错，则应返回一个非 0 的整数。紧接在“main”后面的一对空的圆括号指出该 main 函数没有参数。有关函数值和参数的概念将在第 8 章讨论函数时再详细介绍。

需要指出，该程序中的第 2 行也可以写成“void main ( )”，“void”表示 main 函数的类型是空类型，即 main 函数执行后，不把任何值返回给操作系统，此时也不需要“return 0 ;”这个返回语句。要实现这一点，应启动 Turbo C++ 3.0 的集成开发环境，并依次选择“Options”→“Compiler”→“Source...”，调出 Source Options（源选项）对话框，将其中的 Keywords（关键字）中的 4 个单选按钮中的 Turbo C++单选按钮选中。

第 3 行的左大括号“{”指示 main 函数的开始，而在最后一行的右大括号“}”指示该函数的结束。在该程序中，右大括号还指示整个程序的结束。在这两个括号之间的所有语句组成主函数的函数体。该函数体包含语句的一个集合，以便完成给定的任务。

本例中主函数的函数体中有一个输出语句，它是一个可执行语句（executable statement）。其中 printf 是标准函数库中的一个用于打印输出的预定义的标准 C 函数。预定义是指它是一个已经编写和编译过的函数，并且和我们的程序连接在一起。编译和连接的概念将在本章的后面讲解。printf 函数包含在一对圆括号之间的信息称为函数的参数。该参数是一个字符串（即用一对双引号括起的字符序列）。该函数将字符串中的所有字符原样输出，其中“\n”是

换行符，即在输出“Good␣morning␣World!”后，再将换行符换成回车符和换行符两个字符输出，这在概念上类似于打字机上的按 Enter 键。语句的最后以分号结束。

在使用标准函数库中的标准函数时，编译器要求提供所用函数的有关信息，比如所用函数的声明等。

标准输入/输出函数被定义并作为 C 标准输入/输出库的一部分保存。如果要使用其中的任何一个函数，就必须在程序的开头添加一个“#include <stdio.h>”编译预处理命令。其中“stdio.h”是含有所需函数的声明等信息的文件名，称为标准输入/输出头文件。这条命令的作用就是在编译之前将 stdio.h 整个文件中的内容插入到这条命令所在的位置（本例是程序的第 1 行）。于是后面就可以使用 printf 函数了。

【例 1-2】 求两个整数之和。

```
/*   一个加法程序
     由曹哲编写
     编写日期：2010-06-12   */
#include <stdio.h>
int   main( )   /* 主函数 */
{
   /* 以下 2 行是声明部分*/
   int   a , b ;    /* 定义 a、b 为整型变量 */
   int   sum ;      /* 定义 sum 为整型变量 */
   /* 以下 5 个可执行语句是可执行部分 */
   a = 15 ;
   b = 8 ;
   sum = a + b ;
   printf("Sum = %d\n" , sum ) ;
   return 0 ;
}
```

在该程序中，用“/*”开头并用“*/”结束的那些行称为注释（comment）。通常，注释可以插入到空白符（blank spaces，包括空格、Tab 符、换行符等）可以出现的任何地方，比如在一行的开头、中间或结尾，但不能出现在一个词的中间。在一个程序中使用注释可以提高程序的可读性（readability）和可理解性（understandability）。它们帮助程序员（programmers）和用户（users）理解一个程序的算法，并充当调试（debugging）和测试（testing）的助手。注释行不是可执行语句（not executable statement），因此在编译时编译器将忽略在/*和*/之间的任何东西。由于注释不影响所编译程序的执行速度和大小，所以在程序中应该多加使用。通常在程序的开头可以用注释行给出程序的名称、功能、作者、日期等信息，在某段程序之前可以用注释行说明该段程序的作用，而在某些关键语句的后面则可以用注释说明该语句的作用。可见，注释只是写给人看的，不是写给计算机的。

在本例的程序中，第 8 行和第 9 行是函数体的声明部分，分别定义变量 a、b 和 sum 为整型（int 型）变量。第 11、12、13 行是 3 个赋值语句。分别完成把 15 赋给变量 a，把 8 赋给变量 b，再把 a 的值 15 和 b 的值 8 相加，并把结果 23 赋给变量 sum，从而完成了求 2 个数之和的运算。第 14 行是一个输出语句，其中调用 printf 函数的圆括号中有 2 个实参，第 1 个实参"Sum = %d\n"是“格式控制字符串”，第 2 个实参 sum 是要输出的项，格式符“%d”表示以十进制整数格式在它出现的位置输出 sum 的值，而“格式控制字符串”中的普通字符则原样输出。运行上述程序，将在屏幕上输出如下结果：

```
Sum␣=␣23
```

【例 1-3】 调用自定义函数求两个数之和。

```
#include <stdio.h>
int   main( )   /* 主函数 */
{
    int add( int x , int y ) ; /* 对被调用函数 add 的声明 */
    int   a , b , sum ;    /* 定义变量 a、b 和 sum 为整型 */
    a = 15 ;
    b = 8 ;
    sum = add( a , b ) ; /* 调用 add 函数，将得到的函数值赋给 sum */
    printf("Sum = %d\n" , sum ) ;  /* 输出 sum 的值 */
    return 0 ;
}
/* 以下 6 行定义一个 add 函数，功能是求 2 个数之和 */
int add( int x , int y ); /* 定义函数名，函数值为整型，形式参数 x、y 为整型 */
{
    int   z ;        /* 函数的声明部分，定义 z 为整型变量 */
    z = x + y ;    /* 求出 x + y 的和，并且赋给变量 z */
    return ( z ) ; /* 带着 z 的值返回到主函数调用该函数的位置 */
}
```

本例的程序有 2 个函数。一个是主函数，即 main 函数，另一个是用户自定义函数，即 add 函数。其实，主函数的名字 main 是系统定义的，而其函数体里的语句还是用户自己编写的。在主函数中，程序的第 4 行“int add (int x , int y) ;”是对被调用函数的声明，由于 add 函数是在 main 函数的后面定义的，只有先声明才能被调用。程序的第 8 行是一个赋值语句，其右端的“add (a , b)”是对自定义函数 add 的调用，其中圆括号中的 a、b 是 2 个实在参数，简称实参。执行该调用时把实参值传递给对应的形参变量，即把实参 a 的值 15 传递给形参变量 x，把实参 b 的值 8 传递给形参变量 y，然后转去执行被调用函数 add 的函数体，在该函数体中，先声明一个整型变量 z，然后将 x 的值 15 和 y 的值 8 相加得 23，并赋给变量 z，接着执行返回语句“return (z) ;”带着 z 的值 23 返回到主函数中的“add (a , b)”处，使其得到函数值 23，然后再将 23 赋给该赋值语句左端的变量 sum。所以，运行本例的程序，也会得到如下的结果：

```
Sum␣=␣23
```

读者可以比较例 1-2 和例 1-3 的设计方法，以便理解自定义函数的作用。

2．C 程序的结构

通过以上的 3 个简单例子，我们可以初步了解 C 程序的结构。

（1）C 程序主要由函数组成

函数是 C 程序的基本单位。一个 C 程序可以只由一个 main 函数组成，也可以由一个 main 函数和若干个其他函数组成。被调用函数可以是 C 函数库中的函数，如标准输出函数 printf；也可以是用户自定义函数，如例 1-3 中的 add 函数。C 语言中的函数相当于其他高级语言中的子程序。每个函数都完成一个特定的功能。整个程序的功能就是通过依次调用各个函数来实现的。在一个程序中，编写的每个函数都可以看做一个程序模块。所以 C 语言支持模块化程序设计方法。为了软部件重用，在 C 函数库中提供了大量的库函数。比如，ANSI C 提供了 100 多个库函数、Turbo C 2.0 提供了 300 多个库函数、而 Turbo C++3.0 的 C 部分则提供了更多的库函数。

（2）一个函数由函数首部和函数体构成

1）函数首部，由它给出所定义函数的函数属性、函数类型、函数名、圆括号中各个形参的类型和名字等信息。例如，在例 1-3 中定义了 add 函数，其函数首部如下：

```
int      add(    int      x    ,    int      y    )
 ↑        ↑       ↑        ↑         ↑        ↑
函数类型  函数名  形参类型  形参名   形参类型  形参名
```

需要指出，函数名后面必须跟一对圆括号，它是定义函数的象征。括号内给出形参表列。如果有多个形参，则中间以“,”分隔。形参表列也可以为空，即可以定义一个无参函数。

2）函数体，即紧跟函数首部后面最外层一对花括号内的部分。函数体一般由声明部分（declaration part）和可执行部分（executable part）组成。

- 声明部分，包括对被调用函数的声明和本函数所用到的变量的声明和定义等。例如，在例 1-3 的 main ( )函数体的声明部分声明了 add 函数，定义了 3 个整型变量 a、b 和 sum。
- 可执行部分，可由若干可执行语句组成，以便实现该函数的功能。

C 语言规定，一般声明部分的语句必须放在可执行部分的所有可执行语句的前面。声明部分可以省去（如例 1-1 所示）。如果声明部分和可执行部分两者都省去，就定义了一个空函数。如 void modular1( ){ }就是一个空函数，这在编写多模块的大型应用程序的过程中是有用的。

（3）函数在程序中的位置

含有多个函数的 C 程序，各个函数的先后位置可以随意，但程序总是先从 main 函数开始执行。

（4）C 程序开始部分应包括的内容

在 C 程序的开头，所有函数的前面，应依次包括连接部分（link section，由若干个包含命令组成，如#include <stdio.h>等）、定义部分（definition section，定义所有用到的符号常量）和全局声明部分（global declaration section）。有些变量在多个函数中使用，这样的变量称为全局变量（global variables）。全局变量应在所有函数的外面加以声明。全局声明部分还可以声明所有用户自定义函数。有关内容将在以后各章中将详细讲解。

（5）每个 C 语句的最后必须以一个分号结束

分号是 C 语句的必要组成部分。例如声明语句“int add (int x , int y) ;”和赋值语句“sum = a + b ;”都以分号结束。

（6）C 程序书写格式较自由

C 语句可以分写在多行上，一行也可以写多条语句。通常建议一行只写一条语句，以便使程序更清晰。

（7）程序中应多加注释

在编写程序时，应使用 /*……*/ 对所编程序进行充分的注释，以便提高程序的可读性和可理解性。

（8）C 语言对输入/输出操作的处理

C 语言本身没有输入/输出等语句。输入/输出操作是由库函数来完成的。比如调用 scanf 函数可完成从键盘输入数据，调用 printf 函数可以把数据输出到屏幕上等。由于将涉及硬件的许多操作用函数来实现，使得 C 语言本身涉及硬件较少，从而提高了 C 语言本身和由它所开发的程序的可移植性（portability）。需要指出的是，不同版本的 C 语言的库函数除了提供标

准函数外，都进行了各自的扩充，提供了一些专门的非标准的函数，而用非标准函数编写程序将会降低程序的可移植性。

在例 1-2 中，用一个主函数求 2 个整数之和，如果现在要求 2 个实数之和，程序可改动如下：

```
#include <stdio.h>
int   main( )   /* 主函数 */
{
    float   a , b , sum ;    /* 定义 a、b、sum 为实型变量 */
    a = 12.3456 ;
    b = 33.123 ;
    sum = a + b ;
    printf("Sum =%6.2f\n" , sum ) ;
    return 0 ;
}
```

在上述程序中，第 4 行是函数体的声明部分，分别定义变量 a、b 和 sum 为实型（float 型，即浮点型）变量。第 5、6、7 行是 3 个赋值语句，分别完成把 12.3456 赋给变量 a，把 33.123 赋给变量 b，再把 a 的值 12.3456 和 b 的值 33.123 相加，并把结果 45.4686 赋给变量 sum，从而完成了求 2 个实数之和的运算。第 8 行是一个输出语句，其中调用 printf 函数的圆括号中也有两个实参，第 1 个实参"Sum =%6.2f\n"是“格式控制字符串”，第 2 个实参 sum 是要输出的项，格式符“%6.2f”表示以十进制小数格式在它出现的位置输出 sum 的值，其中数据占 6 格，保留两位小数，小数点后第 3 位四舍五入到第 2 位，结果为 45.47，左补 1 个空格，而“格式控制字符串”中的普通字符则按出现的顺序原样输出。运行上述程序，将在屏幕上输出如下结果：

```
Sum␣=␣45.47
```

1.5 结构化程序设计方法简介

我们学习计算机语言的目的就是用它来编写程序，以便解决科学计算、数据处理或事务管理等各种应用领域里的问题。需要指出，程序只是软件的一部分，软件由两部分组成：其一是机器可执行的程序和数据；其二是与软件开发、使用运行、维护和培训等有关的文档资料。如要开发一个大型软件，目前一般都采用“软件工程”的方法。这种方法把开发软件看成一项工程，即用工程、科学和数学的原则和方法开发、维护计算机软件的有关技术和管理方法。

使用“软件工程”方法开发软件的一般步骤是：可行性研究；需求分析；概要设计；详细设计；编码和单元测试；集成测试；验收测试；运行维护，直到最后退役。即软件开发与维护需经过 8 个阶段。由于软件工程学已经超出了本书的范围，这里仅介绍一种结构化程序设计方法。

结构化程序设计方法是由迪克斯特拉（E. W. Dijkstra）于 1965 年提出的一种程序设计技术，它采用自顶向下逐步细化的设计方法和单入口单出口的控制构件。采用这种方法进行程序设计的主要步骤如下。

1）问题分析。

2）概要设计。

3）结构化算法设计。

4）结构化编码。