542

程序设计

夏宽理　王春森　　编著

复旦大学出版社

图书在版编目(CIP)数据

程序设计/夏宽理,王春森编著.—上海:复旦大学出版社,2000.8
ISBN 7-309-02575-X

Ⅰ.程… Ⅱ.①夏…②王… Ⅲ.C语言-程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2000)第 31523 号

出版发行	复旦大学出版社
	上海市国权路 579 号 200433
	86-21-65102941(发行部) 86-21-65642892(编辑部)
	fupnet@fudanpress.com http://www.fudanpress.com
经销	新华书店上海发行所
印刷	上海第二教育学院印刷厂
开本	787×1092 1/16
印张	23.25
字数	580 千
版次	2000 年 8 月第一版 2000 年 8 月第一次印刷
印数	1—4 000
定价	40.00 元

如有印装质量问题,请向复旦大学出版社发行部调换。

版权所有 侵权必究

前　言

要用计算机来求问题的解，就要为计算机编写能解决相应问题的程序。这首先要求会编写计算机程序，要掌握程序设计技术得从学习程序设计开始。

编写程序，就是将解决问题的算法用某种语言描述后告诉计算机，为此人们为编制计算机程序开发了许许多多种计算机程序设计语言。本书采用 C 语言作为计算机程序设计的描述语言，这是因为 C 语言是一种具有功能丰富、表达能力强、使用灵活方便、可移植性好等优点的程序设计语言。它曾被广泛用于开发应用软件和系统软件，特别是后来引入面向对象机制，从它发展而来的 C++ 语言更是现在广泛应用的程序设计语言。它的许多概念和描述方法被现行的更好的程序设计语言所采纳，学好了 C 语言就能方便地学习并使用 C++ 语言和 Java 语言。

开发 C 语言的最初目的是研制一种编写系统软件的程序设计语言，为此引入了许多为提高程序执行效率和编制大型程序系统为目的的概念和机制。虽然它的描述多样性和灵活性，会给初学程序设计的读者对某些方面的正确理解带来一些困难，但这些概念对于读者更进一步理解计算机程序设计的内容会有很大的帮助。本书力求概念叙述准确、内容介绍循序渐进，设法让读者正确了解和完整掌握 C 语言的概念和编程方法，通过实践，达到能熟练使用 C 语言编写程序的目的。

要能熟练地进行程序设计，除需要掌握一种程序设计语言外，还需要掌握算法、数据结构以及程序设计技巧和方法等多方面的知识。本书特别注重介绍如何正确编写程序，详细地介绍从算法开发到程序编写的全过程；在介绍许多实例程序时先给出求解的想法或算法的开发过程，最后才给出程序。本书介绍的有关数据结构的基础知识和常用算法的开发方法更是进一步迎合了这个要求，让读者学习本书后，不仅能正确了解 C 语言、掌握初步的程序设计方法和技巧，而且对最经常使用的数据结构和常用的算法开发方法也有一定的了解。

本书共分十章，各章内容安排如下。

第 1 章介绍程序设计基本概念、结构化程序设计、C 语言的基础知识。

第 2 章介绍 C 语言的基本数据类型、各运算符的意义和表达式的书写规则、指针基础知识，同时介绍最基本的数据输入输出方法。

第 3 章介绍 C 语言的控制结构、函数基础知识和正文文件的简单用法。这些内容是进入计算机世界，编写简单程序所必须的。

第 4 章介绍数组和字符串的有关概念、定义和它们的使用方法；还介绍指针变量与数组的关系、多级指针和指针变量应用实例等。掌握这些内容，读者已具有组织数据并进行某种较复杂处理所必需的知识和能力。

第 5 章介绍函数各类参数的设定和使用、函数指针的用法、递归函数的基础知识。本章还用大量的实例说明函数的编写方法。至此，读者已具备将具有独立功能的程序段编写成函数的能力，为编写更复杂的程序打下了一定的基础。

第6章介绍局部变量、全局变量、变量的作用域、存储类等概念；还介绍宏定义、文件包含和条件编译等编译预处理方面的基本知识。这些内容是编写大程序所必须具备的知识。

第7章介绍结构、结构数组、结构指针的用法，特别还重点介绍了链表程序设计的基本技术及其应用；同时还介绍联合、位域、枚举和类型定义方面的一些基本知识。这些知识是用于组织复杂数据结构的，特别是为具有用动态数据结构方法求解问题的能力打下一定的基础。

第8章介绍文件的基本概念、文件应用程序结构和常用文件操作库函数的用法，以及基于文件的应用程序实例。这是编写处理大量数据问题的程序所必需的。

第9章介绍栈、队列的基本知识和应用，树与二叉树的基本概念、它们的遍历算法；以及常用算法设计方法，如迭代法、穷举搜索法、递推法、递归技术、回溯法、贪婪法、分治法、动态规划法等。这些内容帮助读者了解计算机程序经常采用的数据结构和最主要的计算机算法的开发方法。

第10章介绍面向对象程序语言的基本概念和机制、类和抽象数据类型、运算符重载、继承和C++输入输出流等基本知识。为进一步学习和使用C++语言提供必要的帮助。特别是用C++语言的开发环境作为学习程序设计的实习环境的读者，在学习C程序设计时，同时了解C++语言是一种非常好的学习程序设计方法。

本书主要是为学生学习程序设计而编写的，与其他介绍C语言程序设计的教材相比，主要有两个特色：一是有大量的程序设计实例。在实例程序设计中，强调介绍程序的开发过程，即通过分析问题，先用逐步求精方法开发问题的求解算法，最后给出问题的程序解。作者认为，这正是一般小程序或程序模块设计的完整过程。二是进一步介绍了数据结构的基础知识和算法设计的常用方法，这是因为程序设计是与数据结构和算法紧密相连的，其目的是让读者对程序设计的有关内容有更全面的了解，并使读者学习了本书以后，不仅了解程序语言，并确实能用程序设计语言编写一些有一定难度的程序，为读者通过进一步实践，由简单到复杂，逐步熟练，为编写复杂程序打下坚实的基础。

本书也非常适宜于用作自学程序设计的教材，亦可作为“计算机软件专业技术资格和水平考试”及计算机培训班的教材和参考书。

本书力求叙述通顺，为了便于初学者自习，按学习要求对内容作了分类，对于一些非常专业的章节标上了加号(+)，对于难度较高的章节则标上星号(*)。另外，在章节顺序上也作了一些调整，如将有关文件的最基本内容提前在语句和简单程序设计章节中介绍，便于读者在上机实习时，尽早编写直接从文件中读入数据的程序，这既便于程序调试，也更接近于编写实际应用程序的习惯。

在本书编写过程中，得到多位老师的关心和支持，在此深表谢意。

编　者

2000年7月

目　录

此为试读，需要完整PDF请访问：www.ertongbook.com

第 1 章　程序设计基础

学习程序设计是一件非常辛苦的事情，要有非常强的耐心。学习程序设计除要求具有一定的数学知识外，要学习和熟练掌握一门与人们习惯使用的自然语言非常不一致的程序语言，要能用程序语言熟练描述问题求解的方法，要学习程序设计的许多常规方法，要求不断学习计算机的新知识和新内容，还要求反复地上机实践。对于初学者来说，最困难的可能还是很难适应描述计算机算法的思维习惯，人们几乎无法承受程序必须将算法描述得几乎绝对地精细和精确。但对计算机来说，这又是非常必要的。一个计算机系统能正确完成预定的任务，除要有性能高、质量可靠的计算机等硬件设备外，还要为计算机系统设计功能齐全、使用方便的程序。计算机运行的过程就是计算机执行程序的过程。要让计算机去完成一项新的任务，就必须为它编写一个能让计算机正确完成该项新任务的程序。开发一个程序，特别是一个软件系统，是一件非常复杂的工作，要经历许多阶段。对于一个功能相对简单的计算任务来说，编写一个相对简单的程序，则是程序设计和程序编码阶段的任务。培养开发软件系统的能力，这要从设计和编写简单的程序开始，所以学习程序设计是达到掌握开发软件系统技术这个大目标的第一步。

1.1　程序设计基本概念

在介绍程序设计技术之前先介绍一些与程序设计有关的基本概念。

程　　序

要使计算机能完成人们预定的工作，就必须把要完成工作的具体步骤编写成计算机能执行的一条条指令。计算机执行这个指令序列后，就能完成指定的功能，这样的指令序列就是程序。所以，程序就是供计算机执行后能完成特定功能的指令序列。

通常，一个计算机程序主要描述两部分内容：描述问题的每个对象及它们之间的关系；描述对这些对象作处理的处理规则。其中关于对象及它们之间的关系是数据结构的内容，而处理规则是求解的算法。针对问题所涉及的对象和要完成的处理，设计合理的数据结构常可有效地简化算法，数据结构和算法是程序最主要的两个方面。

计算机程序有以下共同的性质：

- 目的性——程序有明确的目的，程序运行时能完成赋予它的功能。
- 分步性——程序为完成其复杂的功能，由一系列计算机可执行的步骤组成。
- 有序性——程序的执行步骤是有序的，不可随意改变程序步骤的执行顺序。
- 有限性——程序是有限的指令序列，程序所包含的步骤是有限的。
- 操作性——有意义的程序总是对某些对象进行操作，使其改变状态，完成其功能。

程序设计

程序设计就是根据要计算机完成的任务，提出需求，设计数据结构和算法，编制程序和调试程序，使计算机程序能正确完成需求所设定的任务。简单地说，程序设计是设计和编制程序的过程。

程序首先应能正确完成任务，且是可靠的。同时，由于程序在使用过程中，因使用环境改变或需修改其原来的功能等原因，可能会经常修改。因此，除了为程序编写详尽正确的文档外，编写容易阅读的结构化程序也是对一个好程序的要求。总的来说，一个好的程序有可靠性、易读性、可维护性等良好特性。为达到这些目标，应采用正确的程序设计方法，以便从方法上更有助于设计出具有上述良好特性的程序。

机器语言、汇编语言和高级语言

程序设计语言是人与计算机进行信息通信的工具，是用来书写计算机程序的语言。只有用计算机指令编写的程序才能被计算机直接执行，而用其他任何指令编写的程序还需要通过中间的翻译过程。用于编写计算机程序的语言少说也有几百种，它们大致可分为三类，机器语言、汇编语言和高级语言。计算机的指令系统称为机器语言，所有的计算机都只能直接执行由其自身机器语言编写的程序。机器语言与计算机的硬件密切相关，机器语言中的计算机指令通常用一个二进制形式的代码，由若干位 1 和 0 组成。一条计算机指令指示计算机一次完成一个最基本的操作。

由于用机器语言编写程序实在太慢和太枯燥，用类英语单词的缩写代表计算机的二进位代码指令，这些类英语单词缩写的符号指令构成了汇编语言的基础。用汇编语言编写的程序比用机器语言编写的程序清晰了许多，编写程序也方便了许多。但用汇编语言编写的程序要在计算机上执行，先要将用汇编语言编写的源程序转换成机器语言程序，称完成这个转换功能的程序为“汇编程序”。

随着计算机应用范围的迅速扩大，各种应用程序也越来越大，为加速程序的开发速度，人们开发出用语句表示要计算机完成有一定意义任务的高级语言。而把用高级语言写的源程序转换成机器语言程序的翻译程序称为“编译器”。显然用高级语言比用机器语言或汇编语言编写程序更为简便。高级语言是一种既能方便地描述求解问题的处理对象和对象的处理规则，又能借助于编译器为计算机所接受、理解和执行的人工语言。高级语言习惯上称程序设计语言，它有统一的语法，独立于具体机器，便于人们编码、阅读和理解。本书将用 C 语言作为程序的描述语言，C 语言是功能强、应用范围广的高级语言之一。本书最后还将介绍目前广泛应用的C++ 语言。

面向过程的语言

目前最流行最经常使用的程序设计语言属面向过程型的语言，如广为流行的 Fortran 语言、Pascal 语言、C 语言、Cobol 语言和 Ada 语言等。面向过程的语言虽可独立于计算机编写程序，但用这类语言编写程序时，程序不仅要说明做什么，还要非常详细地告诉计算机如何做，程序需要详细描述解题的过程和细节。

面向问题的语言

为了进一步推广和便于应用计算机，目前已研制了多种面向问题的语言。用面向问题的语言解题时，不仅摆脱了计算机的内部逻辑，也不必关心问题的求解算法和求解的过程，只需指出问题是要计算机做什么，数据的输入和输出形式，就能得到所需结果。面向问题语言又称为非过程化语言或陈述性语言，如报表语言、SQL(structured query language)语言等。SQL语言是数据库查询和操纵语言，能直接使用数据库管理系统。由于使用面向问题语言来解题只要告诉计算机做什么，不必告诉计算机如何做，能方便用户的使用和提高程序的开发速度。但实现面向问题语言的系统从最一般的意义下实现问题如何做，通常实现的效率较低。另外，面向问题语言要求问题已有确定的求解方法，目前其应用范围还比较狭窄。

面向对象语言

为克服面向过程语言过分强调求解过程的细节，程序不易复用的缺点，推出了面向对象程序设计方法和面向对象语言。面向对象语言引入了对象、消息、类、继承、封装、抽象、多态性等机制和概念。用面向对象语言进行程序设计时，以问题域中的对象为基础，将具有类似性质的对象抽象成类，并利用继承机制，仅对差异进行程序设计，这大大提高了程序的复用能力和程序开发效率。面向对象程序语言已是程序语言的主要研究方向之一。

算　法

算法就是问题的求解方法，一个算法由一系列求解步骤组成，算法的描述由经明确说明的一组简单指令和规则组成，计算机按规则执行其中的指令能在有限的步骤内解决一个问题或者完成一个函数的计算。正确的算法要求组成算法的规则和步骤的意义应是唯一确定的，是没有二义性的；由这些规则指定的操作是有序的，必须按算法指定的操作顺序执行，并能在执行有限步骤后给出问题的结果。

对求解同一个问题可能会有多种算法可供选择，选择算法的标准是算法的正确性和可靠性，算法的简单性和易理解性。其他的标准还有算法所需要的存储空间少，执行时间短等。

描述算法的常用工具有流程图，又称框图。流程图是算法的图形描述，由于流程图往往比程序更直观清晰，容易阅读和理解，所以它不仅可以作为编写程序的依据，而且也是交流的重要工具。在逐步求精的结构化程序设计方法中，目前多数采用结构化的伪代码来描述算法。本书采用与C语言控制结构一致的伪代码描述算法。

通常算法的开发过程是由粗到细分多步完成的。先给出粗略的计算步骤，然后对其中的粗略步骤作深入考虑，添加上一些实现细节，变成较为详细一些的描述。可能其中还包含有实现细节不明确的部分，则还需对其中不明确的部分作进一步的细化。直至算法所包含的计算步骤全部都足够清楚，能用某种程序语言完全描述为止。这种算法开发方法称为逐步求精开发方法。

数 据 结 构

计算机程序的处理对象是描述客观事物的数据，由于客观事物的多样性，会有不同形式

此为试读，需要完整PDF请访问：www.ertongbook.com

的数据，如整数、实数、复数和字符，以及所有计算机能够接收和处理的各种各样符号集合。在计算机程序中，形式不同的数据采用数据类型来标识。变量的数据类型说明变量可能取的值的集合，以及可能施于变量的操作的集合。例如，程序对表示苹果个数的变量，它们只能大于或等于零的整数，对它们能施行加减操作，不能施乘除操作，但这些变量能与整数一起施行乘除操作等。所以数据类型不仅定义了一组形式相同的数据集，也定义了对这组数据可施行的一组操作集。

数据结构是指数据对象及其相互关系和构造方法，程序中的数据结构描述了程序中的数据间的组织形式和结构关系。

数据结构与算法有密切的关系，只有明确了问题的算法，才能较好地设计数据结构；要选择好的算法，又常常依赖于合理的数据结构。数据结构是构造算法的基础。

对计算机而言，程序是一组供计算机执行的指令序列。程序告诉计算机如何对指定的数据进行操作，最终得到希望的结果。从问题求解来看，求解问题的程序是对一个抽象的求解算法的详细描述，这种描述是建立在数据的特定的表示方式和结构的基础上的。不了解对数据进行操作的算法就无法决定如何构造数据。同样，确定算法结构和算法步骤也依赖于作为基础的数据结构。也就是说，程序的构成是和数据结构不可分割的。程序在描述算法的同时，也必须完整地描述作为算法的操作对象的数据结构。对于一些复杂的问题，常因数据的表示方式和结构的差异，问题的抽象求解算法也会完全不同。所以程序、数据结构和算法三者之间的关系，可以说，程序是对数据结构和算法的描述，即

$$\text{程序} = \text{数据结构} + \text{算法}$$

1.2 结构化程序设计

对简单的问题进行程序设计，主要工作是选择或设计能解决问题的算法和确定数据结构。一旦算法和数据结构确定后，就可选用合适的程序设计语言编制程序。程序设计的大致步骤依次是设计数据结构和算法、用结构化的伪代码描述算法、编写程序和测试程序。

随着计算机应用的日益广泛，计算机软件的规模和复杂性不断增加，对软件的测试和维护也越来越困难。人们为克服软件危机，提出了结构化程序设计的方法和软件工程的概念，要求软件开发必须遵循一套严格的工程准则，以得到可靠、结构合理、容易维护的软件产品。

结构化程序设计主要包括程序结构的自顶向下模块化设计方法、模块算法的逐步求精设计方法以及用结构化控制结构描述算法和编写程序。

自顶向下模块化设计方法

限制程序的复杂性是程序设计的核心问题。程序的自顶向下模块化设计能有效地解决这个问题。程序结构自顶向下模块化设计方法就是把一个大程序按功能划分成一些较小的部分，每个完成独立功能的小部分用一个程序模块来实现。分解模块的原则是简单性、独立性和完整性。按模块化设计方法开发程序，能使程序具有较高的可靠性和灵活性，同时便于程序的测试和维护。在用模块化方法划分程序模块时，应尽量让模块具有如下所述的多项良好性质：

- 让模块具有单一入口和单一出口。
- 模块不宜过大,应让模块具有单一的功能。
- 模块的执行不能对环境产生副作用。
- 让模块与环境的联系仅限于明确定义的输入参数和输出参数,模块的内部结构与调用它的程序无关。
- 尽量用模块的名字调用模块。

根据由粗到细,由抽象到具体的原则,程序结构的开发应自顶向下进行,首先粗略地划分出程序的宏观结构,在深入分析的基础上,逐步细化划分出一个个独立功能的模块和小模块。

逐步求精设计方法

程序设计的基本方法是抽象、枚举和归纳。抽象包括算法的抽象和数据抽象。算法抽象是指算法的寻求(或开发)采用逐步求精、逐层分解的方法。数据抽象也指在算法抽象的过程中逐步完善数据结构和引入新的数据及确定关于数据的操作。

模块算法的设计采用逐步求精设计方法,即先设计出一个抽象算法,这是一个在抽象数据上实施一系列抽象操作的算法,由粗略的控制结构和抽象的计算步骤组成。抽象操作只指明"做什么",对这些抽象操作的细化就是想方设法回答它"如何做"。采用逐步求精的方法,由粗到细,将抽象步骤(大任务)进一步分解成若干子任务。分而治之,对仍不具体的抽象子任务再进行分解。如此反复地一步步细化,算法越来越具体,抽象成分越来越少,直至可以编程为止。

结构化控制结构

在早期程序设计活动中,因没有统一的现成方法,因人习惯而异,编制的程序的控制结构混乱,很难阅读和修改。经计算机科学工作者多年努力,已归结出顺序、选择和循环三种结构化的控制结构,并在理论上证明了可计算问题的程序都可用这三种控制结构来描述。

1. 顺序结构

通常,一个复杂的计算过程不可能用一个操作步骤来表达,为此,需把复杂的计算工作分解成一系列逐条执行的操作序列。顺序结构为把一个复杂的计算用若干简单的顺序计算实现提供控制手段。顺序结构是一个操作序列,顺序结构执行时,从序列的第一个操作开始,顺序执行序列中的操作,直至序列的最后一个操作执行后完成。若顺序结构只有两个操作 A 和 B 组成,可写成以下形式:

```
{
  A;
  B;
}
```

例如,为交换变量 x 和 y 的值,可分解为顺序执行的三个操作步骤,写成顺序结构:

```
{
  temp = x;      /* 将 x 的值暂存于 temp */
  x    = y;      /* 将 x 置成 y 的值 */
```

```
    y    = temp;  /* 将 y 置成暂存于 temp 的值 */
}
```

2. 条件选择结构

条件选择结构为描述按不同情况自动选择操作步骤执行提供控制手段。条件选择结构由一个判断条件 P 和两个供选择的分支操作 A 和 B 组成,可写成以下形式:

```
if (p)
  A;
else
  B;
```

条件选择结构执行时,先计算条件 P 的值,如果 P 的值为真,即条件成立,则执行分支操作 A;否则,若条件 P 的值为假,即条件不成立,则执行分支操作 B。注意,无论 P 为何值,条件选择结构只能执行 A 或 B 之一。

条件选择结构中的分支操作又可以是任何控制结构,特别当分支操作又是条件选择结构时,就呈现嵌套的条件选择结构。

3. 循环结构

循环结构为描述循环操作提供控制手段。循环结构有多种表现形式,最常见的有 while 型循环结构和 do-while 型循环结构。

while 型循环结构由一个条件 P 和一个循环操作 A 组成,可写成以下形式:

```
while (P)
  A;
```

while 型循环结构的执行过程是:每次循环前,先求条件 P 的值,当条件 P 成立(真)时,就执行操作 A,并接着再次求条件 P 的值,以确定操作 A 是否再次被执行。当 P 的值一开始为假,或某次循环后 P 的值为假,则结束循环操作。

do-while 型循环结构也由一个条件 P 和一个循环操作 A 组成,可写成以下形式:

```
do
  A;
while (P);
```

do-while 型循环结构的执行过程是:每次循环前,先执行操作 A,接着再求条件 P 的值,当条件 P 成立(真)时,再执行操作 A。如此反复,直到条件 P 的值为假,结束循环操作。

以上所说的操作 A 和 B 可以是一个简单的操作步骤,也可以又是某种控制结构。理论上,可计算问题的程序可以不用如 goto 那样的控制转移操作,而全由一些简单的操作和用上述三种控制结构反复嵌套来描述。如是这样,这种程序的结构性和可读性就比较良好,建议读者要编写结构良好、可读性高的程序。

1.3 C语言基础知识

1.3.1 几个简单的C程序

本节从C程序例子出发介绍C语言的基础知识，让读者对C语言有一个初步的印象。其中提及的概念、名称在以后的章节中都将有详细的介绍。对于已有其他高级语言使用经验的读者，可能有些概念的提法与已熟悉的语言稍有不同，只要浏览阅读一遍就行。

【例1.1】一个只输出一行信息的C程序。

```
#include   <stdio.h>
void main()        /*主函数*/
{
  printf("This book is <Programming with C and c++  languages> . \n");
}
```

该程序只输出一行信息：

This book is <Programming with C and C++ languages> .

以这样一个非常简单的C程序为例，对C程序可以指出以下几个特点：

(1) 一个C程序有一个名为main的函数，称它为主函数。实际上C程序可以由多个函数组成。例1.1程序只有一个主函数。组成程序的多个函数可存放在一个或多个源程序文件中。习惯将含有函数定义的程序文件名称的后缀定为c。

(2) 主函数main()前的关键字void表示该主函数的执行不返回结果。

(3) 在函数名之后有一对圆括号“()”，圆括号内可包含函数的参数。

(4) 函数体用花括号“{}”括起来。花括号可以用来括住任何一组C代码，从而构成复合语句或分程序。例1.1的函数体只有一个printf()函数调用。printf()函数是C系统的函数库中的输出函数，其中用双引号“""”括住的字符串用来指明printf()函数的输出格式。例中的输出格式表示以字符串原样输出，而“\n”是换行的意思，即输出以下字符列：

This book is <Programming with C and C++ languages> .

之后换行。

(5) 简单C语句之后有一个分号“;”，它是某些语句的结束符。

(6) 程序中的“/*…*/”表示程序的注释部分。程序中的注释是给阅读程序的人看的，对程序编译和运行都没有作用。它用来说明该程序的文件名称、程序的功能、使用方法、最后更改日期等；注释出现在程序代码行中或某行语句之后，用来说明一段程序代码或一个语句的功能、意义等。

(7)上述程序中的第一行

```
#include   <stdio.h>
```

是编译预处理命令行，它告诉编译系统将包含有关输入和输出标准函数信息的stdio.h文件

内容作为当前源程序文件的一部分。

【例 1.2】读入两个整数,输出它们的和。

```
/*1*/   #include <stdio.h>
/*2*/   void main()
/*3*/   {/*变量定义部分*/
/*4*/     int x,y,sum;                    /*定义整型变量 x,y,sum*/
/*5*/       /*以下为语句序列*/
/*6*/     printf("Input x and y\n");      /*输出提示输入数据的字样*/
/*7*/     scanf("%d%d",&x,&y);            /*输入 x 和 y 的值*/
/*8*/     sum=x+y;                        /*完成 x+y 的计算,求 sum=x+y*/
/*9*/     printf("x+y=%d\n",sum);         /*输出结果*/
/*10*/  }
```

例 1.2 程序的功能是输入两个整数,求出它们的和并输出。为表示变化的数据对象,程序引入变量,程序中的第 4 行就是用来定义变量,它定义了三个整型变量,分别命名为 x、y 和 sum。第 6 行实现输出文字"Input x and y",用来提示程序已开始执行,请输入 x 和 y 的值。第 7 行的 scanf()是 C 函数库中的输入函数,其中"%d"是十进制整数输入格式,用来指定输入数据的数据类型和格式,这里表示键入的数据以十进制整数理解,将它转换成整型量的机内表示。&x 和 &y 中的"&"的含义是取地址,对于本例,意指顺序输入的两个十进制整数将它们分别存于用 x 和 y 命名的变量所对应的内存单元中,直观地理解就是输入值给变量 x 和 y。第 8 行是完成 x+y 的计算,并将计算结果赋给变量 sum。第 9 行的 printf()函数调用将输出"x+y="字样和变量 sum 的值,并换行,其中"%d"意指将 sum 的值按十进制整数形式输出。该程序运行情况如下:

```
Input x and y
    12  15     (假定输入 12 和 15 两个整数)
x+y=27
```

上面三行中的第 1、3 行是程序输出的,第 2 行是用户键入的,12 与 15 之间用空格或回车分隔。

【例 1.3】利用公式:C=(5/9)(F-32)输出 F 氏温度与 C 氏温度对照表,设已知 F 氏温度取 0、20、…、200。

```
#include   <stdio.h>
void main()
{  float f, c;          /*变量定义*/
   int lower, upper, step;
   lower=0;   upper=200;   step=20;   f=(float)lower;
   while (f<=upper) {   /*循环计算*/
     c=(float)(5.0/9.0*(f-32.0));
     printf("%3.0f   %6.1f\n",f,c);
     f=f+step;
   }
```

```
}
```

为描述循环计算,程序用循环控制结构来描述。本程序对指定范围内的 F 氏温度进行循环计算。程序首先为有关变量设定初值,然后用循环语句实现循环计算。在循环体内,对每个 F 氏温度计算出对应的 C 氏温度后输出,然后增加 F 氏温度值。循环直至 F 氏温度超出要求后结束。循环控制用循环语句实现,上述程序用的是 while 循环语句。C 程序的循环控制也可有 for 语句或 do - while 语句实现。

【例 1.4】输入两个实数,输出它们中的小的数。

```
#include   <stdio.h>
void main()
{   float x,y,c;              /*变量定义*/
    float min(float, float);/*函数说明*/
    printf("Input x and y.\n");
    scanf("%f%f",&x,&y);
    c=min(x,y);              /*调用函数 min()*/
    printf("MIN(%.2f,%.2f)=%.2f\n",x,y,c);
}
/*以下定义函数 min()*/
float min(float a, float b)
{   float temp;/*函数内使用的变量定义*/
    if(a<b)   /*如果 a<b*/
      temp=a;
    else
      temp=b;
    return temp;          /*返回 temp 到调用 min()函数处*/
}
```

为控制程序的复杂性,程序把一些功能相对独立的程序段编写成函数。例 1.4 的程序包含两个函数:主函数 main()和求两个实数中小的数的函数 min()。函数 min()的作用是将 a,b 中的较小的值赋给变量 temp,然后用返回语句(return)将 temp 的值返回给对它的调用处(主函数中的 c=min(x,y))。本例 main()函数要调用 min()函数,调用时将实参 x 和 y 的值分别传送给 min()函数中的形参 a 和 b。经执行 min()函数后,得到一个值,这个值被赋给变量 c。程序中的输入输出函数调用中的格式"%f"与前面例子中介绍的格式"%d"相似,格式"%f"是用于输入输出浮点型数据的。注意:主函数中的函数说明"float min(float, float);",它是用于说明名字 min 是一个返回浮点型(float)值的函数,它有两个 float 型形参。

C 语言中遵守对象"先说明或定义,后使用"的原则。若主函数未对函数 min()作说明,则主函数中的"c=min(x,y)"的 min()函数被省缺设定为整型函数,而后面的 min()函数定义却是浮点型,导致类型不一致的错误。另请读者注意:C 语言中变量和函数的说明与定义的区别,直观的理解是说明只给一个名赋予某种意义,如变量说明只宣布变量的类型等特性、函数说明只宣布它的结果类型等特性;而定义除表明全部特性外,更主要的是确定实体,如变量定义要求为变量分配存储,函数定义是指定形参和实现其功能的程序代码。

此为试读,需要完整PDF请访问: www.ertongbook.com