高等学校计算机基础教材系列

C语言程序设计教程

（第二版）

谭浩强　张基温　唐永炎　编著

高等教育出版社

高等学校计算机基础教材系列

C语言
程序设计教程

（第二版）

谭浩强　张基温　唐永炎　编著

高等教育出版社

(京)112 号

内 容 提 要

C 语言是一种结构化的计算机程序设计语言,它既具有高级语言的特点,又具有低级语言的功能。

本书是学习 C 语言程序设计的基础教程,采取循序渐进的内容安排,通俗易懂的讲解方法,并辅以大量便于说明问题的例题,使得凡学过一门高级语言的读者都能通过学习本书掌握 C 语言的基本内容,并应用它编写程序。

本书主要内容包括:C 语言的基本概念、各种数据类型的使用技巧、C 语言模块化程序设计的方法、文件的基本操作和使用规则;考虑到 C 语言的一个重要用途是直接操作硬件,还特地安排了综合应用举例一章,详细地介绍了这方面的几个应用实例;每章后附有习题。

本书可作为大专院校计算机专业和非计算机专业的教材,也可供计算机培训班或其他自学者使用。

图书在版编目(CIP)数据

C 语言程序设计教程/谭浩强等编著. -2 版-北京:高等教育出版社,1998

高等学校计算机基础教材系列

ISBN 7-04-006407-3

Ⅰ.C… Ⅱ.谭… Ⅲ.C 语言-程序设计-高等学校-教材 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(98)第 11842 号

*

高等教育出版社出版

北京沙滩后街 55 号

邮政编码:100009 传真:64014048 电话:64054588

新华书店总店北京发行所发行

[illegible]印刷厂印刷

开本 787×1092 1/16 印张 23.25 字数 580 000

1992 年 4 月第 1 版

1998 年 7 月第 2 版 1998 年 7 月第 1 次印刷

印数 0 001—50 093

定价 18.60 元

前　言

近年来，C语言无论在国内还是在国外都得到迅速地推广应用。C语言以它的功能丰富、表达能力强、使用灵活、应用面广、目标程序效率高、可移植性好、能对硬件直接进行操作等优点愈来愈赢得人们的青睐。

每一种语言都有自己的特点和适用的领域。例如，BASIC语言以其容易学习、使用方便赢得千万初学者，但它不适宜编制大型程序。有许多高级语言适合写应用程序，但不适合写系统软件。C语言与其它高级语言相比有其显著的优点，它既具有高级语言的优点，又具有低级语言的许多特点。因此，它既适合编写应用程序，又适合编写系统程序，应用领域是很宽广的（当然C语言也不是万能的，不能以它取代其它一切高级语言。例如C语言的数值运算功能不如FORTRAN，它也不是进行事务管理的理想语言）。C语言是一种理想的结构化语言，用它能编写出紧凑、高效、风格优美的程序。因此，一个真正的计算机程序人员应该学会使用C语言。

C语言功能强、使用灵活，但是一般认为它比较难学，不熟练的程序员常常出错而不知其所以然。一般认为，C语言适合于有一定经验的程序员使用。学习和使用C语言需要有一定的软件和硬件的基础知识。

许多非专业的程序人员和大学生很希望学习C语言，他们迫切要求有一本学习C语言的入门教材。考虑到这一情况，我们决定尝试为初学者编写一本适用的教材。只需要有一门以上计算机语言程序设计知识（例如BASIC）的基础就可以学会并掌握本书的基本内容。在本书的体系结构上，我们按照初学者的认识规律作了细致的安排，以使读者能循序渐进地逐步深入。我们有意地将难点分散，在学习每一章时都不会感到太困难。本书从一开始就提出和介绍函数，使读者能较早地掌握函数这一概念并利用它构造程序，因为C语言是函数式语言，应当能十分熟练地使用函数。

考虑到一般读者已有程序设计的初步知识，对算法也有一定的了解，在本书中不再详细讲解算法，而着重介绍如何根据已知的算法去编写C程序。这样可以更好的消化C语言本身。

本书介绍的C语言是其最基本的部分，C语言在使用上有许多细节，无法在这本基础教材中全部涉及，这有赖于读者将来在实践中逐步掌握。本书的前九章，不需要系统的硬件知识就能学会。考虑到C语言的一个重要用途是直接操作硬件，可以进行系统调用，为了帮助读者了解这一方面的应用，并为以后的提高打下初步基础，我们在第十章（综合应用举例）中介绍了几个这方面的例子。要看懂这几个例子需要有一定的硬件知识，如果没有条件，第十章的内容可以暂时不学，待以后需要用到时再参考。

本书的叙述以87 ANSI C（美国国家标准局87年公布C语言）为基础，采用现代风格定义和声明函数，使程序具有更好的易读性，并使系统对函数和参数的类型进行检查。本书用N-S结构化流程图描述算法，所有程序都按照结构化程序设计方法编写，程序写成锯齿形格式，这些都会有助于读者养成良好的程序设计习惯，编写出质量较高的程序。

由于各种C编译系统的实现有所不同,它们与标准C有一定的差异,有时一个程序用不同的C编译系统编译后,运行所得的结果不完全相同。本书的程序是以IBM PC机上使用的Turbo C 2.0版本为背景的。这些程序大多数也适用于任何一种计算机系统的C版本。少数程序在其它计算机系统上运行可能会有一些小的差别,如果出现这种情况,请参阅该系统的手册,了解所用系统的具体规定。

参加本书编写工作的有谭浩强教授、张基温教授和唐永炎副教授。在共同讨论提纲的基础上分别收集材料和编写,最后由谭浩强教授修改定稿。袁玫和谭亦峰同志参加了部分收集资料和调试程序的工作。我们还编写出版了《C语言习题集和上机指导》,作为本书的配套参考书(由高教出版社出版)。本书在1991年出版后,被许多高校选用为教材,对本书给予肯定和好评。根据教学实践和读者的意见,我们在97年对之进行一次修订,对部分章节内容进行了调整和补充。

由于计算机科学技术发展迅速,加以我们水平不高,本书定有不少缺点或错误,我们希望在本书出版使用后根据广大读者的意见,在适当的时候修改再版,以满足需要。

编 者

1997.12

目　录

第一章　C语言程序设计初步 …… (1)
§1.1　程序设计语言 …… (1)
1.1.1　程序设计语言的发展 …… (1)
1.1.2　程序设计语言的支持环境 …… (3)
1.1.3　源程序的编辑、编译、连接与执行 …… (4)
§1.2　用库函数组装C程序 …… (4)
§1.3　自己设计C函数 …… (8)
习题 …… (15)
第二章　数据描述与基本操作 …… (16)
§2.1　数据类型 …… (16)
2.1.1　数值的定点表示形式与浮点表示形式 …… (16)
2.1.2　字符类型数据的表示和存储形式 …… (17)
2.1.3　数据的存储空间长度及取值范围 …… (18)
2.1.4　带符号的数据类型与无符号的数据类型 …… (19)
§2.2　常量和变量 …… (22)
2.2.1　直接常量和符号常量 …… (22)
2.2.2　直接常量的书写格式 …… (23)
2.2.3　变量和对变量的赋值 …… (26)
2.2.4　变量的声明 …… (27)
2.2.5　标识符 …… (27)
§2.3　运算符与表达式 …… (28)
2.3.1　算术运算 …… (30)
2.3.2　关系运算、逻辑运算与条件运算 …… (33)
§2.4　不同类型数据间的转换 …… (38)
2.4.1　几个概念 …… (38)
2.4.2　不同类型数据的隐式转换 …… (40)
2.4.3　不同类型数据的显式转换 …… (42)
§2.5　数据的输入和输出 …… (43)
2.5.1　printf函数 …… (43)
2.5.2　scanf函数 …… (46)
2.5.3　getchar函数与putchar函数 …… (51)
习题 …… (52)
第三章　C程序的流程设计 …… (56)
§3.1　算法 …… (56)
3.1.1　算法的性质与组成要素 …… (56)
3.1.2　算法的描述 …… (57)
§3.2　用C语句描述算法 …… (64)
3.2.1　表达式语句 …… (64)
3.2.2　形成流程控制结构的语句 …… (65)
3.2.3　限定转向语句 …… (66)
3.2.4　goto语句 …… (67)
3.2.5　C基本语句一览 …… (67)
3.2.6　复合语句 …… (68)
3.2.7　停止函数exit …… (69)
§3.3　选择型程序设计 …… (69)
3.3.1　if…else结构的应用 …… (69)
3.3.2　else if结构的应用 …… (74)
3.3.3　switch结构的应用 …… (75)
§3.4　循环型程序设计 …… (79)
3.4.1　穷举与迭代算法 …… (79)
3.4.2　while结构的应用 …… (83)
3.4.3　do…while结构的应用 …… (89)
3.4.4　for结构的应用 …… (90)
习题 …… (96)
第四章　模块化程序设计 …… (100)
§4.1　函数 …… (100)
4.1.1　C程序结构 …… (100)
4.1.2　函数定义与函数声明 …… (104)
4.1.3　函数的传值调用 …… (109)
4.1.4　函数的嵌套调用 …… (110)
4.1.5　函数的递归调用 …… (111)
§4.2　变量的存储属性 …… (115)
4.2.1　动态变量 …… (116)
4.2.2　静态变量 …… (119)

4.2.3 外部变量 …… (121)
§4.3 编译预处理 …… (127)
4.3.1 宏替换 …… (127)
4.3.2 文件包含 …… (134)
习题 …… (136)
第五章 数组 …… (142)
§5.1 一维数组 …… (142)
5.1.1 一维数组的定义 …… (142)
5.1.2 一维数组的初始化 …… (142)
5.1.3 数组元素的引用 …… (143)
5.1.4 数组作为函数参数 …… (144)
5.1.5 一维数组应用举例 …… (146)
§5.2 二维数组和多维数组 …… (155)
5.2.1 二维数组和多维数组的概念及其定义 …… (155)
5.2.2 二维数组和多维数组的引用 … (157)
5.2.3 二维数组和多维数组的初始化 …… (158)
5.2.4 二维数组程序举例 …… (160)
§5.3 字符数组和字符串 …… (163)
5.3.1 字符串和字符串的存储方法 … (163)
5.3.2 字符数组的初始化 …… (165)
5.3.3 字符串的输入 …… (165)
5.3.4 字符串的输出 …… (166)
5.3.5 字符串运算函数 …… (168)
5.3.6 二维的字符数组 …… (171)
5.3.7 字符数组应用举例 …… (172)
习题 …… (175)
第六章 指针 …… (178)
§6.1 指针概述 …… (178)
6.1.1 地址与指针 …… (178)
6.1.2 指针的类型与指针的定义 …… (179)
6.1.3 指针变量的引用 …… (180)
6.1.4 指向指针的指针 …… (182)
§6.2 指针与数组 …… (184)
6.2.1 一维数组的指针表示方法 …… (184)
6.2.2 二维数组的指针表示方法 …… (188)
6.2.3 指针与字符串 …… (192)
6.2.4 指针数组 …… (195)
§6.3 指针与函数 …… (201)
6.3.1 指针作为函数参数 …… (201)
6.3.2 数组指针作为函数参数 …… (204)
6.3.3 指向函数的指针 …… (211)
6.3.4 返回指针值的函数 …… (217)
6.3.5 main 函数中的参数 …… (218)
§6.4 指针数据小结 …… (221)
6.4.1 常用的指针类型变量归纳 …… (221)
6.4.2 指针的运算 …… (221)
6.4.3 指向 void 类型的指针 …… (223)
习题 …… (224)
第七章 结构体、共用体和枚举类型数据 …… (228)
§7.1 结构体类型概述 …… (228)
7.1.1 结构体的概念及其定义 …… (228)
§7.2 结构体类型变量的定义和引用 …… (230)
7.2.1 定义结构体类型变量的方法 … (230)
7.2.2 结构体变量的初始化 …… (233)
7.2.3 结构体变量的引用 …… (233)
7.2.4 结构体的输入和输出 …… (235)
§7.3 结构体数组 …… (237)
7.3.1 结构体数组的定义方法 …… (237)
7.3.2 结构体数组的初始化 …… (238)
7.3.3 结构体数组的引用 …… (239)
§7.4 结构体变量作为函数参数以及返回结构体类型值的函数 …… (244)
7.4.1 结构体变量作为函数参数 …… (244)
7.4.2 返回结构体类型值的函数 …… (246)
§7.5 结构体变量与指针 …… (250)
7.5.1 指向结构体变量的指针 …… (250)
7.5.2 指向结构体数组的指针 …… (252)
7.5.3 用指向结构体变量的指针作函数参数 …… (253)
§7.6 动态存储分配——链表 …… (255)
7.6.1 动态存储分配和链表的概念 … (255)
7.6.2 用包含指针项的结构体变量构成结点 …… (257)
7.6.3 用于动态存储分配的函数 …… (258)
7.6.4 链表应用举例 …… (260)
§7.7 共用体类型数据 …… (265)
7.7.1 共用体的概念 …… (265)
7.7.2 共用体变量的引用 …… (266)
7.7.3 共用体变量的应用 …… (268)

§7.8　枚举类型数据 …… (271)
§7.9　用 TYPEDEF 定义类型 …… (272)
习题 …… (274)
第八章　位运算 …… (275)
§8.1　位运算概念 …… (275)
§8.2　位运算符使用方法 …… (276)
8.2.1　按位与运算符 …… (276)
8.2.2　按位或运算符 …… (279)
8.2.3　按位异或运算符 …… (280)
8.2.4　按位取反运算符 …… (281)
8.2.5　左移运算符 …… (282)
8.2.6　右移运算符 …… (282)
§8.3　位运算应用举例 …… (284)
§8.4　位段 …… (289)
8.4.1　位段的概念和定义方法 …… (289)
8.4.2　位段的引用方法 …… (291)
习题 …… (292)
第九章　文件 …… (293)
§9.1　文件概述 …… (293)
9.1.1　文件的概念 …… (293)
9.1.2　缓冲文件系统(标准 I/O)和非缓冲文件系统(系统 I/O) …… (293)
9.1.3　文件(FILE)类型指针 …… (295)
§9.2　文件的打开与关闭 …… (296)
9.2.1　文件的打开(fopen 函数) …… (296)
9.2.2　文件的关闭(fclose 函数) …… (298)
§9.3　文件的顺序读写 …… (298)
9.3.1　输入和输出一个字符 …… (298)
9.3.2　输入和输出一个字符串 …… (303)
9.3.3　格式化的输入和输出 …… (306)
9.3.4　按"记录"的方式输入和输出 … (309)
§9.4　文件的定位与随机读写 …… (312)
9.4.1　文件的定位 …… (313)
9.4.2　随机读写 …… (313)
§9.5　文件操作的出错检测 …… (317)
§9.6　非缓冲文件系统(系统级 I/O) …… (318)
9.6.1　非缓冲文件系统的主要特点 … (318)
9.6.2　打开文件 …… (319)
9.6.3　读文件和写文件 …… (321)
9.6.4　关闭文件 …… (321)
9.6.5　缓冲区的设置 …… (321)
§9.7　文件重定向 …… (323)
习题 …… (324)
第十章　综合应用举例 …… (325)
§10.1　简单的数据库管理 …… (325)
§10.2　ROM－BIOS 的利用 …… (330)
§10.3　利用中断调用 DOS 例行程序 …… (335)
§10.4　端口访问实例——发声程序 …… (337)
§10.5　数据采集 …… (339)
§10.6　简单的弹出式菜单 …… (340)
§10.7　用 C 语言编制画图程序 …… (343)
附录Ⅰ　ASCⅡ字符编码一览表 …… (350)
附录Ⅱ　关键字及其用途 …… (351)
附录Ⅲ　运算符的优先级别和结合方向 …… (351)
附录Ⅳ　C 库函数 …… (352)

第一章　C语言程序设计初步

§1.1　程序设计语言

程序设计语言是人与计算机进行信息交流的工具。程序设计要在一定的程序设计语言环境下进行。

1.1.1　程序设计语言的发展

从计算机诞生到今天，程序设计语言也在伴着计算机技术的进步不断升级换代。

(一) 机器语言

一种CPU的指令系统，也称该CPU的机器语言，它是该CPU可以识别的一组由0和1序列构成的指令码。下面是某CPU指令系统中的两条指令：

10000000　加

10010000　减

用机器语言编程序，就是从所使用的CPU的指令系统中挑选合适的指令，组成一个指令系列。这种程序虽然可以被机器直接理解和执行，却由于它们不直观，难记、难认、难理解、不易查错，只能被少数专业人员掌握，同时程序的生产效率很低，质量难以保证。这种繁重的手工方式与高速、自动工作的计算机极不相称。

(二) 汇编语言

为减轻人们在编程中的劳动强度，20世纪50年代中期人们开始用一些“助记符号”来代替0,1码编程。如前面的两条机器指令可以写为

A+B⇒A　或　ADD A,B

A−B⇒A　或　SUB A,B

这种用助记符号描述的指令系统，称为符号语言或汇编语言。

用汇编语言编程，程序的生产效率及质量都有所提高。但是汇编语言指令是机器不能直接识别、理解和执行的。用它编写的程序经检查无误后，要先翻译成机器语言程序才能被机器理解、执行。这个翻译转换过程称为“代真”。代真后得到的机器语言程序称为目标程序(object program)，代真以前的程序，称为源程序(source program)。由于汇编语言指令与机器语言指令基本上具有一一对应的关系，所以汇编语言源程序的代真可以由汇编系统以查表的方式进行。

汇编语言与机器语言，都是依CPU的不同而异，它们都称为面向机器的语言。用面向机器的语言编程，可以编出效率极高的程序。但是程序员用它们编程时，不仅要考虑解题思路，还要熟悉机器的内部结构，并且要“手工”地进行存储器分配。这种编程的劳动强度仍然很大，给计算机的普及推广造成很大的障碍。

(三) 面向过程的语言

汇编语言和机器语言是面向机器的,随机器而异。1954 年出现的 FORTRAN 语言以及随后相继出现的其它高级语言,使人们开始摆脱进行程序设计必须先熟悉机器的桎梏,把精力集中于解题思路和方法上,使程序设计语言开始与解题方法相结合。其中一种方法是把解题过程看作是数据被加工的过程。基于这种方法的程序设计语言称为面向过程的程序设计语言。C 语言就是一种面向过程的程序设计语言。下面是一个计算圆柱体体积的 C 语言程序片段:

```
main(  )                    /*告诉编译器 C 程序由此开始执行*/
{                           /*这一程序段开始*/
int r,h;                    /*半径 r 与高 h 为整数*/
float v;                    /*体积 v 为实数*/
v=3.14159* r* r* h;         /*计算体积 v*/
printf("%f",v);             /*输出体积 v 的值*/
}                           /*程序结束*/
```

我们看到这个程序是很好理解的,其中计算体积的语句与我们所习惯的数学式子没有什么根本的区别("/*"与"*/"之间的内容称为注释,目的是让程序更容易被理解)。显然,使用高级语言编程可以较大的降低编程过程的劳动强度,提高编程的效率。高级语言的诞生是计算机技术发展史上的一个里程碑。它使人们能摆脱具体机器指令系统的束缚,用接近人们习惯的语言来构思解题过程,从而大大提高了编程效率,使人们能够编制出规模越来越大的程序,以满足日益广泛而深入的应用需求。

实际上,程序是对现实世界的运动状态的模拟。面向过程的程序设计认为,每个程序都要完成一些规定的功能。每个功能的实现是通过对数据进行一系列的加工的过程而实现的。因而程序设计包括组织数据——设计数据结构,以及设计对数据结构进行加工的过程——设计算法两个部分。

从 1954 年第一种高级语言 FORTRAN 问世后不久,不同风格、不同用途、不同规模、不同版本的面向过程的高级语言便风涌而起。据统计,全世界已有成千种的高级语言,其中使用较多的有近百种。图 1.1 为几种使用最广泛的面向过程的高级语言的发展变迁情况。在这些语言中,C 语言以其高效、灵活、功能丰富、表达力强、移植性好而受青睐,被称为近十年来在计算机程序设计实践中做出了重大贡献的一种语言。它的前身是 Martin Richards 于 20 世纪 60 年代开发的 BCPL 语言,这是一种计算机软件人员在开发系统软件时,作为记述语言使用的程序语言。它既具有高级语言的功能,又具有机器语言的一些特征。

1970 年 UNIX 的研制者 Ken Thompson 应用汇编语言和称为 B 的语言,发表了在 PDP-7 上实现的 UNIX 的初版。B 语言是他对 BCPL 语言的发展和改进。此后美国 Bell 研究所的 Dennis Ritchie 和 Brian Kernighan 对 B 语言做了进一步的充实和完善,于 1972 年推出了一种新型程序设计语言——C 语言。

C 语言反映了当前计算机的能力。随着适合于各种不同操作系统(UNIX,MS-DOS,CP/M-80,86 等)和不同机种(字长为 8bit~32bit)的 C 语言编译系统相继出现。1983 年美国国家标准局(ANSI)语言标准化委员会对 C 语言问世以来的若干发展、补充和修改进行订正后公布了一个 C 语言标准草案(83 ANSI C),1987 年又推出了 87 ANSI C。本书将以 87 ANSI C 为基础,同时兼顾各种版本的通用性和一致性予以介绍。

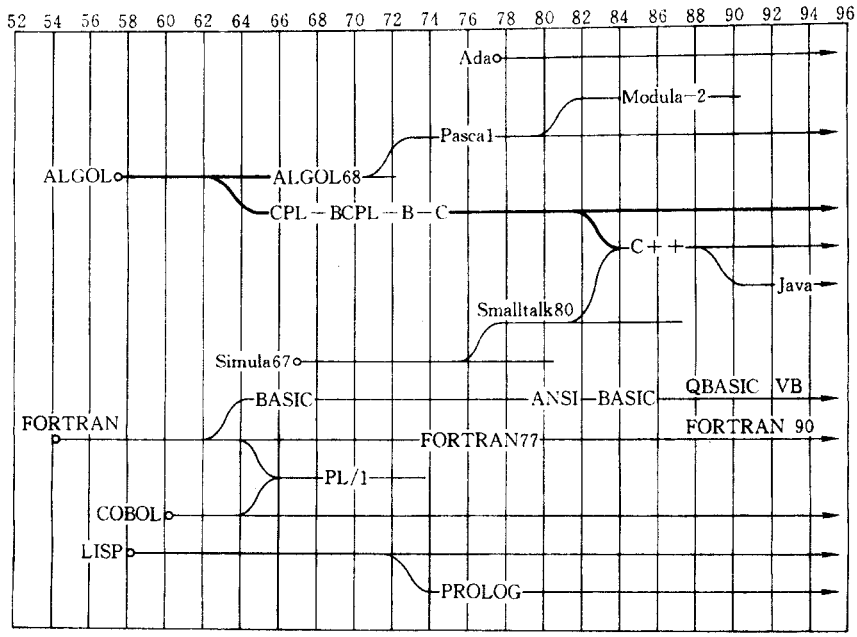

图 1.1

(四) 面向对象的程序设计语言

面向对象的程序设计是一种结构模拟方法,它把现实世界看成是由许多对象(object)所组成,对象之间通过互相发送和接收消息进行联系;消息激发对象本身的运动,形成对象状态的变化。从程序结构的角度,每个对象都是一个数据和方法的封装体——抽象数据类型。

从分类学的观点看,客观世界中的对象都是可以分类的。也就是说,所有的对象都属于特定的"类"(class),或者说每一个对象都是类的一个实例。因而,面向对象的程序设计的一个关键是定义"类",并由"类"生成"对象"。

面向对象的程序比面向过程的程序更清晰、易懂,更适宜编写更大规模的程序,正在成为当代程序设计的主流。面向对象的程序设计语言有 Simmla－67、Smalltalk80、Java 等。还要特别指出的是由 C 语言派生出的 C＋＋语言。它是一种多范型程序设计语言,不仅可以用它编写面向对象的程序,还可以用它编写面向过程的程序。

1.1.2 程序设计语言的支持环境

现在几乎每一种计算机都配备了操作系统。操作系统是各种软件中最重要的一种,或者说它是各种软件的核心与基础。所有其它程序的运行都要在操作系统的控制下进行。操作系统的作用是有效地组织和利用计算机的软、硬件资源,使各种程序能在操作系统的管理下协调工作。操作系统的类型有许多种,例如有分时操作系统、批处理操作系统、多道作业操作系统等。图 1.2 表示高级语言源程序必须经过编译系统处理(编译),然后在操作系统控制下才能为计算机执行。

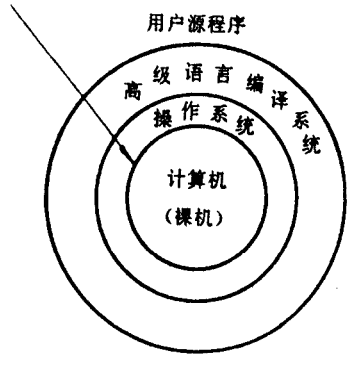

图 1.2

操作系统的功能主要包括:CPU 管理、存储管理、文件管

理、设备管理、作业管理等。不同的操作系统的功能和工作方式是不同的。目前最广泛流行的操作系统有 UNIX,MS－DOS,OS/2,Windows 等。

使用高级语言编写源程序时应当注意程序的运行环境,要充分运用所使用的计算机系统提供的软件环境支持(例如充分利用操作系统提供的功能),并了解源程序在所运行的操作系统管理下是如何工作的。例如在不同的操作系统下编译和运行 C 语言程序的方式和命令是不同的。

1.1.3　源程序的编辑、编译、连接与执行

C 语言采用编译方式将源程序转换为二进制的目标代码。编写好一个 C 程序到完成运行一般经过以下几个步骤。

(一) 编辑

所谓编辑,包括以下内容:① 将源程序逐个字符输入到计算机内存;② 修改源程序;③ 将修改好的源程序保存在磁盘文件中。编辑的对象是源程序,它是以 ASCII 代码的形式输入和存储的,不能被计算机执行。目前使用较多的编辑软件有:UNIX 下的编辑程序 ed,vi 等,MS－DOS 下的 EDLIN,Wordstar,Windows 下的 Write、Word 等字处理软件。关于编辑软件的使用方法请参阅《C 语言习题集与上机指导》一书或其它有关手册。

(二) 编译

编译就是将已编辑好的源程序(已存储在磁盘文件中)翻译成二进制的目标代码。在编译时,还要对源程序进行语法检查,如发现有错,则在屏幕上显示出错信息,此时应重新进入编辑状态,对源程序进行修改后再重新编译,直到通过编译为止。编译后得到的二进制代码在 UNIX 下是后缀为".o"的文件,在 MS－DOS 下是后缀为".obj"文件。应当指出,经编译后得到的二进制代码还不能直接执行,因为每一个模块往往是单独编译的,必须把经过编译的各个模块的目标代码与系统提供的标准模块(如 C 语言中的标准函数库)连接后才能运行。

(三) 连接

将各模块的二进制目标代码与系统标准模块经连接处理后,得到具有绝对地址的可执行文件,它是计算机能直接执行的文件。在 UNIX 下它以".out"为后缀(例如,f.out),在 MS－DOS 以下".exe"为后缀(例如,f.exe)。

(四) 执行

执行一个经过编译和连接的可执行的目标文件。只有在操作系统的支持和管理下才能执行它。图 1.3 用以表示编辑、编译、连接、运行的过程。

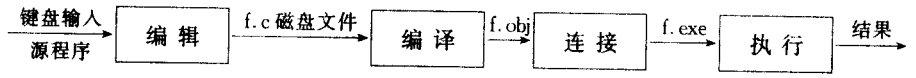

图 1.3

其中文件取名为 f,后缀按 MS－DOS 的规则表示。

近来"集成化"的工具环境已将编辑、编译、连接、调试工具集于一身(例如 Turbo C),用户可以方便地在窗口状态下连续进行编辑、编译、连接、调试、运行的全过程。

§1.2　用库函数组装 C 程序

C 语言本身语句很少,许多功能都是通过函数来完成的。例如,输入输出的功能并不是 C

语言本身提供的,而是C的库函数所提供的。这是C的一个重要特色。这样做的目的是使C语言的核心部分规模较小,而外围(函数)可以做得很丰富,并且可以根据需要增加新的函数,使C语言有较大的灵活性和多方面的功能。在编写C程序时应尽量利用C的库函数所提供的函数功能,来实现自己的目的。

下面是利用库函数编程序的一些简单例子:

例1.1 显示"Programming is fun."的C语言程序

```
main( )
{
printf ("Programming is fun. /n");
}
```

运行这个程序时,在屏幕上显示一行英文:

```
Programming is fun.
```

说明:

(1).C程序是由函数组成的,每一个函数完成相对独立的功能。本程序是由一个称为main的函数构成的。main是函数名,函数名后面一对圆括号内是写函数参数的,本程序的main函数没有参数故不写,但圆括号不能省略。一个完整的程序必须有一个main函数,它称为主函数,程序总是从main函数开始执行的。

main()后面有一对花括号,花括号内的部分称为函数体。本程序中的函数体只由一个语句组成。一般情况下函数体由"说明部分"和"执行部分"组成。说明部分的作用是定义数据类型;执行部分给出操作指令,执行部分由若干语句组成。本例中只有执行部分而无说明部分。

(2) C规定每个语句以分号(;)结束,分号是语句不可缺少的组成部分。本程序中main函数的函数体内只有一个语句,也必须有一个分号。

(3) printf是C的库函数中的一个函数,它的作用是在显示屏上输出指定的内容,此例输出"Programming is fun."字符串。字符串末尾的"\n"是C语言中规定的一个特殊符号,作为控制代码,其作用是"回车换行"。以后还会讲述在反斜杠"\"后面跟一个指定的字符,就会组合成一个具有专门含义的"转义字符"。本程序中"\n"的作用是:在输出"Programming is fun."以后执行一个回车换行操作,如果以后还有输出的话,将从下一行的左端开始输出。

请看以下程序。

```
main( )
{
  printf ("Programming \n");
  printf("is fun. \n");
}
```

执行这个程序将得到如下输出:

```
Programming
is fun.
```

没有"\n"时,后面的输出将不换行。请看下面的程序:

```
main( )
{
  printf("Prog");
  printf("ramming i");
  printf("s fun. \n");
}
```

它的输出为:

```
Programming is fun.
```

C语言提供了丰富的库函数,每个函数实现一定的功能。例如可以用sin函数求正弦值,用abs函数求绝对值,用exp函数求一个数的指数等。C编译系统将这些函数集中存放在一些库文件中,按函数名调用,十分方便。现代程序设计方法认为,程序设计是一种艰巨的脑力劳动,人在进行程序设计的过程中,犯错误的机会很多,如果能找到已经验证的程序模块使用,是提高程序设计效率和程序可靠性的有效措施。按照这一思想,用C语言进行程序设计时,应优先采用C语言函数库中的函数,不要一拿到问题就立即动手从头到尾自己写程序。

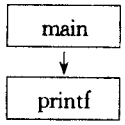

图 1.4

(4) 函数printf由主函数main调用,形成图1.4所示的层次结构。

例1.2　计算一个数的正弦值的C语言程序。

```
#include "math.h"
main( )
{
  float x;                    /*定义x为实型变量*/
  x=sin(0.19199);             /*调用sin函数*/
  printf("%f\n",x);           /*调用printf函数,输出x的值*/
}
```

执行程序的结果如下:

```
0.190813
```

说明:

(1) "float x;"是声明:x是一个实型变量。

(2) "x=sin(0.19199)"可执行一次函数调用,求出0.19199弧度的正弦值,并赋给实型变量x。sin是C系统的库函数,在调用数学库函数时,要用到一些系统提供的说明信息,这些信息(例如"宏定义"等,这将在以后介绍)包含在math.h文件中。".h"是"头文件"的后缀,头文件总是被用在函数的前头。编译命令"#include math.h"的作用是将math.h文件的内容代替这行#include命令,也就是使该文件的内容被整个地调到main的前面,C规定在调用库函数时往往需要用#include命令将有关的"头文件"调到所用函数的前面。"math.h"是调用数学

函数时所需要包含的文件。

(3) 本例中使用了利用输出函数 printf 输出数值数据的形式:在其后的括号中有两部分,一是双引号内的部分,一是双引号外的部分。双引号内的部分是"格式字符串",用它指定输出时的打印格式。此例双引号中有一个"%f",它是输出一个实数的格式字符,执行时将双引号外的变量的值取代"%f",并以实型格式输出。%f 格式提供小数后 6 位数字,小数点前的数字位数不指定,根据实际值的位数输出。其作用如下示:

```
printf("%f\n",x)
        ↑
        用 x 的值取代 %f
```

因此在输出 sin(x)的值 0.190813 后,执行"\n",回车换行。

用一个 C 函数可以调用另一个 C 函数。在例 1.1 中已经使用过用 main 调用 printf,在本例中也可以用 printf 来调用函数 sin,直接在 printf 函数中输出 sin(0.19199)的值。程序为:

```
#include "math.h"
main( )
{
  printf ("%f\n", sin(0.19199));
}
```

运行结果仍为:

```
0.190813
```

这样就形成图 1.5 所示层次结构。但要注意,main 是任何函数都不能调用的。

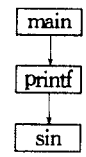

图 1.5

C语言规定了使用 printf 函数输出不同类型数据所用的格式字符,输出十进制整型数据时用%d,字符数据用%c,字符串用%s……,这些将在第二章中详细介绍。

用 printf 函数输出数值数据时,双引号中除了用于指定输出格式的格式字符外,还可以包含其它的一般字符,这些字符在输出时原样输出,例如:

```
printf ("sin 0.19199= %f\n", sin(0.19199));
```

输出结果为:

```
sin 0.19199=0.190813
```

可见除"%f"被 sin(0.19199)的值取代外,其它字符原样输出。在例 1.1 中所使用的 printf 就是一种没有格式字符的形式。没有格式字符,当然也就没有双引号外的数值部分了。

(4) /* 与 */之间为注释信息,对程序运行结果不发生影响,也不被编译,只是为了帮助人们(包括自己)更好地理解程序中有关部分内容而写的。

例 1.2 中介绍了计算一个数的正弦值的方法。C 语言还提供了其它多种数学函数(见附录Ⅳ),读者不难模仿这几个例子设计出求其它函数值的程序。

§1.3　自己设计C函数

用库函数组装程序是程序设计的一条捷径和重要方法。但是，任何函数库都不可能是包罗万象的，当我们在函数库中找不到所需功能的函数时，便要动手设计自己所需的函数。设计的基本方法是从主函数开始，按调用关系，自上而下地进行。

例 1.3　求三个数的平均值。

按自顶向下的方法可写出主函数如下：

```
main(  )
{
   float a,b,c, ave;          /*声明:名字 a,b,c,ave 为实型变量*/
   a=3.5;b=4.6;c=7.9;         /*对 a,b,c 赋值*/
   ave=average(a,b,c);        /*调用 average 函数,得到的值赋给 ave*/
   printf("average=%f",ave);          /*输出 ave 的值*/
}
```

运行情况如下：

3.5,4.6,7.9 ↵

average=5.333333

说明：

(1) 语句：

```
float a,b,c,ave;
```

称为“声明语句”，它声明了后面用到的名字 a,b,c,ave 都是实型变量，也就是指定 a,b,c,ave 为实型变量，即可以用于存储带小数的数。float 称为关键字，意为实型(或称浮点型)。除 float 外，C 语言还有一些用于声明类型的关键字，如 int(整型)、char(字符型)等。如果有声明语句

```
int x,y,z;
```

则声明了名字 x,y,z 是用于存储整型(不带小数)的变量。而声明语句

```
char r,s,t;
```

则声明了名字 r,s,t 为可以存储一个字符的变量。

所谓变量，就是在程序运行过程中值可以变化的量，实际上是对存储空间的指定。

(2) 语句：

```
ave=average(a,b,c);
```

定义了一个复合操作：先调用一个函数 average，其参数为 a,b,c，再把函数 average 的计算结果(称为 average 的返回值)送给变量 ave(称赋值给变量 ave)。

那末，如何知道名字 average 不是一个变量，而是一个函数呢？这是因为它后面有一对圆括号。C 语言约定，名字后面带有一对圆括号者，均为函数名，如我们已经使用过的 main，printf 都是函数名。

究竟函数 average 要进行怎样的运算呢？这要根据 average 的定义来确定。这个定义可以由系统预先定义好放在函数库中，也可以由程序员自己定义。这里我们需要一个求三个数的

平均值的函数,但又从库函数中找不到求三个数的平均值的函数,因此就需要自己设计一个average函数来实现上述功能。可以这样定义average函数:

```
float average(x,y,z)          /*函数头*/
float x,y,z;                  /*声明参数x,y,z为实型*/
{ float aver;                 /*声明变量aver为实型*/
  aver=(x+y+z)/3;             /*求x,y,z的平均值,赋给aver*/
  return (aver);              /*将aver的值作为函数的返回值*/
}
```

上面定义了一个average函数,在函数执行完毕后,它带回的函数值是实型的。第一行称为函数头,用于说明函数的名字(这里为"average")、返回值的类型(这里为"float")、和对什么样的数据(称为参数,这里为x,y,z)进行操作。第二行指定参数的类型为实型。第三行定义变量aver的类型。注意形式参数的类型说明在花括弧之外,而aver的定义在花括弧之内。aver为三个数的平均值。return语句将aver的值返回主调函数。

应当注意,函数定义中所使用的参数(这里的x,y,z)称为形式参数(简称形参),只用于描述被加工的数据是一种什么样的角色,以及如何对它们进行加工,并不说明也不可能说明哪些具体的数据被加工。具体的数据是由调用语句在调用时传递的。

根据上述分析,将main函数和自己定义的average函数组成一个完整的求三个数的平均值的程序:

```
float average(x,y,z)
float x,y,z;
{
  float aver;
  aver=(x+y+z)/3;
  return (aver);
}

main( )
{
  float a,b,c,ave;
  a=3.5; b=4.6; c=7.9;
  ave=average(a,b,c);
  printf("average=%f",ave);
}
```

运行结果与前相同。

例1.4 求任意三个数的平均值。

上例是求三个固定值的平均值,这三个值是在main函数中用赋值语句确定的。本例不用赋值语句确定三个变量的值,而由键盘输入所需的三个实数,以增加程序的灵活性。程序可改