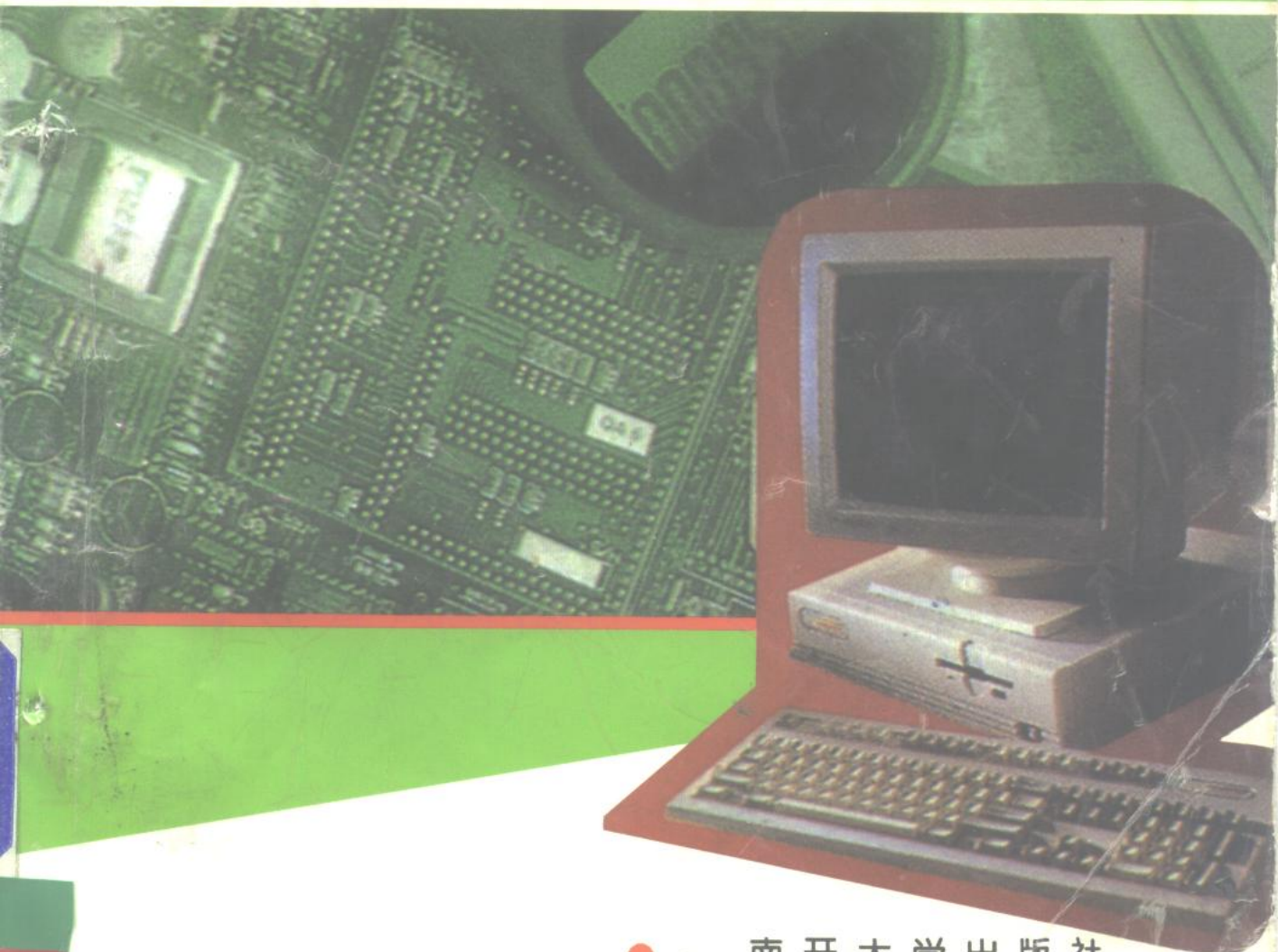


C 语言及其 高级编程技术

● 于春凡 编著



● 南开大学出版社

C 语言及其高级编程技术

于春凡 编著

南开大学出版社

[津]新登字 011 号

内 容 简 介

为了帮助读者全面掌握 C 语言,本书除了介绍 C 语言的基本数据类型、运算符及运算表达式、常量、变量、数组、字符串以及分支程序、循环程序、函数等基本程序设计技术外;还详细地介绍了指针、结构、联合、位域、位操作、文件 I/O、动态数据结构、图形软件的应用及开发技术以及 C 与 DOS、C 与 ROM-BIOS、C 与其它高级语言、C 与汇编语言之间的接口编程等高级编程技术和表达技巧。

本书内容丰富、全面、系统,并列出了大量的程序实例。它既可以作为大学本科生或研究生的教科书,又可以作为软件工程技术人员的参考书。

C 语言及其高级编程技术

于春凡 编著

南开大学出版社出版
(天津八里台南开大学校内)
邮编 300071 电话 3508542
新华书店天津发行所发行
天津市宝坻县印刷厂印刷

1995 年 6 月第 1 版 1995 年 6 月第 1 次印刷
开本:787×1092 1/16 印张:25.25 插页:2
字数:643 千 印数:1—5000

ISBN 7-310-00806-5
TP·34 定价:27.00 元

前 言

C语言是一种通用的程序设计语言。它以高级语言的结构和编程环境,提供了类似汇编语言那样的系统资源操作能力和程序的执行效率,使得它既适合于应用程序设计,又适合于系统程序设计。因此,C语言在操作系统、编译系统、人工智能、软件工具、图象处理、数值分析及数据库管理系统等许多方面都得到了广泛的应用。

C语言有许多突出的优点,它的丰富的数据类型、多种多样的运算符、结构化程序控制、利于模块化的函数,增强了语言的表达能力,提高了编程效率,有利于充分发挥计算机硬件的潜在功能。C语言比其它高级语言更具汇编语言特点,比汇编语言更具有可读性和可移植性,所有这些优点吸引了越来越多的程序员将C语言做为主要编程工具。同时,也吸引了越来越多的人学习C语言。

另一方面,由于C语言太灵活、功能太强,也使得初学者感到难以全面掌握,诸如指针、结构、联合、位域、位操作以及各种实际问题的表达等,都给初学者带来一定的难度。为了帮助读者全面掌握C语言,本书除了介绍C语言的基本数据类型、运算符及运算表达式、常量、变量、数组、字符串以及分支程序、循环程序、函数等基本程序设计技术外,还将详细地介绍指针、结构、联合、位域、位操作、文件I/O、动态数据结构、图形软件的应用及开发技术以及C与DOS、C与ROM-BIOS、C与其它高级语言、C与汇编语言之间的接口编程等高级编程技术和表达技巧。

本书内容丰富、全面、系统,并列举了大量的程序实例。它既可以作为大学本科或研究生的教科书,又可以作为软件工程技术人员的参考书。

在本书的编写过程中,得到了南开大学计算机与系统科学系的领导和老师们热情鼓励和指导。南开大学出版社王家骅编审不仅审阅了全部书稿,而且对本书的编排和选材提出了宝贵的意见,在此一并对他们表示衷心的感谢。

本书由于春凡编著,由曹海燕调试、验证书中的程序实例。

限于本人的水平,书中错误和不妥之处在所难免,诚望广大读者批评指正。

于春凡

目 录

1 绪论	(1)
1.1 C语言的发展历史及特点	(1)
1.1.1 C语言的发展历史	(1)
1.1.2 C语言的特点	(2)
1.2 IBM-PC 微型机所用的 C 语言	(3)
1.3 C 语言程序的结构及书写格式	(3)
1.3.1 C 语言程序的结构	(3)
1.3.2 C 语言程序的书写格式	(4)
1.4 C 语言程序的开发过程	(5)
1.4.1 编辑源程序	(5)
1.4.2 编译源文件	(6)
1.4.3 连接目标文件及库文件	(6)
1.4.4 运行程序	(6)
2 数据及运算	(7)
2.1 标识符命名	(7)
2.1.1 标识符的构成规则	(7)
2.1.2 注意事项	(7)
2.2 基本数据类型	(8)
2.2.1 C 语言的数据类型	(8)
2.2.2 基本类型数据的宽度及范围	(8)
2.2.3 基本类型修饰符	(9)
2.3 常量	(10)
2.3.1 数值常量	(10)
2.3.2 字符常量	(12)
2.3.3 转义字符常量	(12)
2.3.4 字符串常量	(13)
2.3.5 符号常量	(14)
2.4 变量说明及变量定义	(15)
2.4.1 变量定义语句	(15)
2.4.2 局部变量	(16)

2.4.3	全部变量	(17)
2.5	变量的存储类型及其寿命与可见性	(19)
2.5.1	变量的存储类型	(19)
2.5.2	变量的寿命与可见性	(20)
2.6	变量的初始化	(21)
2.7	数组	(23)
2.7.1	一维数组说明及初始化	(23)
2.7.2	字符型数组与字符串	(24)
2.7.3	多维数组	(25)
2.7.4	字符串数组	(26)
2.8	运算符	(27)
2.8.1	算术运算符	(27)
2.8.2	关系运算符及逻辑运算符	(28)
2.8.3	字位运算符	(29)
2.8.4	赋值运算符	(30)
2.8.5	其它运算符	(32)
2.8.6	运算符的优先级	(33)
2.9	表达式	(34)
2.9.1	表达式中的类型转换	(34)
2.9.2	赋值运算表达式中的类型转换	(34)
2.9.3	强制类型转换符	(35)
2.9.4	使用空格和括号增加可读性	(35)
2.10	常用 I/O 函数	(36)
2.10.1	字符 I/O 函数	(36)
2.10.2	字符串 I/O 函数	(37)
2.10.3	格式化 I/O 函数	(38)
3	语句	(41)
3.1	复合语句及分程序	(41)
3.2	条件语句	(42)
3.2.1	条件语句的一般形式	(42)
3.2.2	嵌套的条件语句	(43)
3.2.3	else if 结构的嵌套条件语句	(44)
3.2.4	用运算符?: 替代条件语句	(45)
3.3	循环语句	(45)
3.3.1	while 语句	(46)
3.3.2	for 语句	(48)
3.3.3	do-while 语句	(52)
3.3.4	循环语句小结	(56)
3.4	break 语句	(57)

3.5	continue 语句	(60)
3.6	switch 语句	(63)
3.7	goto 语句及标号	(68)
3.8	return 语句及 exit() 函数	(69)
3.8.1	return 语句	(69)
3.8.2	终止退出函数 exit()	(69)
3.9	空语句	(70)
4	指针	(73)
4.1	指针运算符 & 及 *	(73)
4.1.1	指针运算符 &	(73)
4.1.2	指针运算符 *	(74)
4.1.3	& 与 * 互为逆运算	(74)
4.2	指针的说明及初始化	(74)
4.2.1	指针的说明	(74)
4.2.2	指针的初始化	(75)
4.2.3	指针的特殊值	(76)
4.3	指针运算表达式	(76)
4.3.1	指针的算术运算表达式	(77)
4.3.2	指针的关系运算表达式	(77)
4.3.3	指针的赋值运算表达式	(77)
4.3.4	指针运算表达式的应用举例	(78)
4.4	指针与数组	(79)
4.4.1	两种方法访问数组	(79)
4.4.2	指针与数组表现形式的互换性	(80)
4.5	字符型指针与字符串	(83)
4.5.1	使用字符型指针处理字符串	(83)
4.5.2	用字符串常量初始化字符型指针	(83)
4.5.3	不要使用无指向的指针	(84)
4.6	指针数组	(86)
4.6.1	指针数组的说明	(86)
4.6.2	指针数组的初始化	(86)
4.6.3	指针数组与多维数组	(87)
4.6.4	指针数组与多个字符串	(88)
4.7	指针的指针	(90)
5	函数	(92)
5.1	C 语言函数的基本概念	(92)
5.2	函数定义	(93)
5.2.1	函数定义的一般格式	(93)

5.2.2	从函数中返回	(94)
5.3	函数说明及函数调用	(96)
5.3.1	函数说明	(96)
5.3.2	函数调用	(96)
5.4	函数参数的传送方式	(97)
5.4.1	参数的传值传送方式	(97)
5.4.2	参数的传址传送方式	(98)
5.5	函数返回值的传送	(100)
5.5.1	使用 return 语句传送返回值	(100)
5.5.2	使用传址参数传送返回值	(101)
5.6	数组参数的传送	(102)
5.6.1	向函数传送一维数组	(102)
5.6.2	向函数传送多维数组	(103)
5.7	字符串参数的传送	(105)
5.7.1	向函数传送一个字符串	(105)
5.7.2	向函数传送多个字符串	(106)
5.8	指针型函数	(108)
5.8.1	指针型函数定义	(108)
5.8.2	返回值为全局变量地址	(108)
5.8.3	返回值为 static 型的内部变量地址	(109)
5.8.4	返回值为调用函数内局部变量地址	(110)
5.9	函数指针	(111)
5.9.1	函数指针的定义	(111)
5.9.2	向函数传送函数参数	(113)
5.9.3	函数指针数组	(114)
5.10	嵌套调用及递归调用	(116)
5.10.1	嵌套调用	(116)
5.10.2	递归调用	(117)
5.11	命令行参数	(123)
5.11.1	命令行的一般格式	(123)
5.11.2	C 程序接收命令行参数	(123)
5.12	分割编译及文件间的通讯	(125)
5.12.1	多个源程序文件分割编译	(125)
5.12.2	源程序文件间的通讯	(125)
5.13	编译预处理命令	(127)
5.13.1	#define 命令	(127)
5.13.2	#include 命令	(128)
5.13.3	条件编译命令	(129)
5.13.4	其它编译预处理命令	(131)

6	结构	(133)
6.1	结构定义及结构变量	(133)
6.1.1	结构定义	(133)
6.1.2	结构变量说明	(134)
6.1.3	结构变量成员的访问	(135)
6.1.4	结构变量的初始化	(135)
6.2	结构数组	(137)
6.2.1	结构数组说明及初始化	(137)
6.2.2	结构数组的应用	(138)
6.3	结构指针	(142)
6.3.1	结构指针的说明及初始化	(142)
6.3.2	结构指针目标成员的访问	(142)
6.4	结构及结构成员在函数间的传递	(145)
6.4.1	向函数传递结构成员	(145)
6.4.2	向函数传递完整结构	(146)
6.5	结构型及结构指针型函数	(148)
6.5.1	结构型函数	(148)
6.5.2	结构指针型函数	(149)
6.6	结构成员数组及结构	(151)
6.6.1	结构成员数组	(151)
6.6.2	结构成员结构	(153)
7	I/O 函数与文件	(156)
7.1	C 语言文件的概念	(156)
7.1.1	磁盘文件	(156)
7.1.2	设备文件及标准设备文件	(156)
7.1.3	文件 I/O 系统	(157)
7.1.4	文件控制结构	(157)
7.1.5	文件型指针	(157)
7.2	fopen() 及 fclose() 函数	(158)
7.2.1	打开文件函数 fopen()	(158)
7.2.2	关闭文件函数 fclose()	(160)
7.3	putc() 及 getc() 函数	(160)
7.3.1	文件的字符输出函数 putc()	(160)
7.3.2	文件的字符输入函数 getc()	(161)
7.4	feof()、ferror()、rewind() 及 clearerr() 函数	(165)
7.4.1	测试文件结束函数 feof()	(165)
7.4.2	ferror()、clearerr() 及 rewind() 函数	(166)
7.5	fgets() 及 fputs() 函数	(168)

7.5.1	文件的字符串输入函数 fgetc()	(168)
7.5.2	文件的字符串输出函数 fputs()	(170)
7.6	fread() 及 fwrite() 函数	(171)
7.6.1	读数据块函数 fread()	(171)
7.6.2	写数据块函数 fwrite()	(172)
7.6.3	读写结构类型数据	(173)
7.7	fprintf() 及 fscanf() 函数	(176)
7.7.1	文件的格式化输出函数 fprintf()	(177)
7.7.2	文件的格式化输入函数 fscanf()	(177)
7.8	fseek() 函数与文件的随机访问	(177)
7.8.1	置文件位置指针函数 fseek()	(177)
7.8.2	文件的随机访问	(178)
7.9	设备文件的 I/O	(179)
7.9.1	设备文件的 I/O 处理	(179)
7.9.2	标准设备文件的 I/O 处理	(180)
7.9.3	控制台 I/O 函数	(180)
7.9.4	标准设备文件的重定向	(183)
7.9.5	标准设备文件的管道功能	(185)
7.10	非缓冲文件 I/O 系统	(185)
7.10.1	文件标识号	(185)
7.10.2	文件的建立、打开和关闭	(185)
7.10.3	文件的读写	(187)
8	动态存储分配函数与动态数据结构	(189)
8.1	动态存储管理系统及其函数	(189)
8.1.1	C 语言的动态存储管理系统	(189)
8.1.2	malloc() 及 free() 函数	(190)
8.2	链表数据结构	(192)
8.2.1	链表的概念	(192)
8.2.2	栈式链表的建立及遍历	(192)
8.2.3	队列链表的建立及遍历	(195)
8.3	链表的插入及删除	(198)
8.3.1	插入一个新元素	(198)
8.3.2	处理有序链表	(200)
8.3.3	建立有序链表	(201)
8.3.4	删除一个元素	(203)
8.4	双向链表	(205)
8.4.1	双向链表的建立	(205)
8.4.2	双向链表的插入及删除	(206)
8.5	二叉树	(209)

8.5.1	二叉树的概念	(210)
8.5.2	建立二叉排序树	(211)
8.5.3	遍历二叉树	(212)
8.5.4	二叉树的查找及删除	(215)
8.6	动态数据结构程序实例	(217)
8.6.1	文本编辑器	(217)
8.6.2	统计单词出现频度	(224)
9	屏幕、图形函数与统计简图	(229)
9.1	屏幕控制函数	(229)
9.1.1	选屏幕方式函数	(229)
9.1.2	屏幕选色函数	(230)
9.1.3	清屏幕函数	(232)
9.1.4	光标定位函数	(233)
9.2	基本图形函数	(234)
9.2.1	画点函数	(234)
9.2.2	画线函数	(235)
9.2.3	画方框及填充方框函数	(235)
9.2.4	画圆函数	(237)
9.3	统计简图程序设计	(237)
9.3.1	样条图	(238)
9.3.2	分布图	(240)
9.3.3	预测	(241)
10	字位运算、位域及联合	(246)
10.1	字位运算	(246)
10.1.1	字位与、字位或及字位异或	(246)
10.1.2	移位运算	(248)
10.1.3	反码运算	(249)
10.1.4	位操作赋值运算	(250)
10.2	位域	(254)
10.2.1	位域结构的定义及变量说明	(254)
10.2.2	位域结构成员的访问	(257)
10.3	联合	(257)
10.3.1	联合的定义及变量说明	(257)
10.3.2	联合成员的访问	(259)
10.3.3	联合在函数间的传递	(261)
10.4	类型定义	(262)

11	C 语言与 DOS 的接口技术	(264)
11.1	CPU 的寄存器及 DOS 功能调用	(264)
11.1.1	CPU 的寄存器	(264)
11.1.2	中断与 PC-DOS	(265)
11.1.3	DOS 功能调用	(266)
11.2	C 语言与 DOS 的接口函数	(270)
11.2.1	dos.h 首标文件	(271)
11.2.2	bdos() 函数	(272)
11.2.3	intdos() 函数	(272)
11.3	DOS 系统资源的利用	(272)
11.3.1	检查键盘状态	(273)
11.3.2	打印机的使用	(273)
11.3.3	直接从键盘读取字符	(274)
11.3.4	串行口的读写	(274)
11.3.5	目录列表	(274)
11.3.6	直接读时间	(275)
11.3.7	综合程序实例	(276)
12	C 语言与 ROM-BIOS 的接口技术	(278)
12.1	ROM-BIOS	(278)
12.2	C 语言与 ROM-BIOS 的接口函数	(281)
12.3	ROM-BIOS 系统资源的利用	(281)
12.3.1	清除屏幕	(281)
12.3.2	光标定位	(282)
12.3.3	获取键盘扫描代码	(284)
13	C 语言与图形软件开发技术	(286)
13.1	设置屏幕方式及调色板	(286)
13.1.1	设置屏幕方式	(286)
13.1.2	设置显示色调	(287)
13.1.3	设置背景色	(288)
13.2	画点	(288)
13.2.1	调用 ROM-BIOS 画点	(288)
13.2.2	直接画点	(289)
13.2.3	画点函数的使用实例	(291)
13.3	画线	(292)
13.3.1	采用比率画线	(292)
13.3.2	采用 Bresenham 算法画线	(294)
13.3.3	画线函数的使用实例	(295)

13.4	画矩形及矩形填充	(296)
13.4.1	画矩形	(296)
13.4.2	矩形填充	(296)
13.4.3	画矩形函数的使用实例	(296)
13.5	绘制任意图形	(297)
13.5.1	十字定位光标	(297)
13.5.2	绘图程序实例	(297)
14	C语言与其它高级语言的接口技术	(302)
14.1	语言间等效的程序调用及语言约定	(302)
14.1.1	语言间等效的程序调用	(302)
14.1.2	命名约定	(303)
14.1.3	调用约定	(305)
14.1.4	参数传递约定	(306)
14.2	混合语言程序的开发	(307)
14.2.1	混合语言程序的开发过程	(307)
14.2.2	选择适当的存储模式进行编译	(307)
14.2.3	与语言库的连接方法	(308)
14.3	C语言与其它语言的接口	(309)
14.3.1	建立接口的步骤	(310)
14.3.2	使用 fortran 或 pascal 关键字	(310)
14.3.3	C可采用其它语言的约定	(311)
14.4	C对 BASIC 的调用	(311)
14.4.1	C调用 BASIC 子程序	(312)
14.4.2	C调用 BASIC 函数	(313)
14.5	C对 FORTRAN 的调用	(314)
14.5.1	C调用 FORTRAN 子例程	(314)
14.5.2	C调用 FORTRAN 函数	(315)
14.6	C对 Pascal 的调用	(316)
14.6.1	C调用 Pascal 过程	(317)
14.6.2	C调用 Pascal 函数	(318)
14.7	Turbo C 与 Turbo Pascal 的接口技术	(319)
14.7.1	Turbo C 与 Turbo Pascal 的接口	(319)
14.7.2	Turbo C 调用 Turbo Pascal 过程	(323)
14.7.3	Turbo C 调用 Turbo Pascal 函数	(324)
14.7.4	Turbo C 与 Turbo Pascal 程序连接步骤	(325)
14.7.5	Turbo C 与 Turbo Pascal 相互调用综合例	(325)
15	C语言与汇编语言的接口技术	(329)
15.1	开发过程	(329)

15.2	C 语言的调用方法及命名约定	(330)
15.2.1	外部函数说明	(331)
15.2.2	调用外部汇编过程的方法	(331)
15.2.3	C 语言的命名约定	(331)
15.3	C 语言的参数传递及返回值约定	(332)
15.3.1	参数传递	(332)
15.3.2	传送返回地址	(332)
15.3.3	C 语言的返回值约定	(332)
15.4	外部汇编过程的编程方法	(333)
15.4.1	标准的汇编接口编程方法	(333)
15.4.2	嵌入 C 接口文件编程方法	(342)
15.5	外部变量的使用	(344)
15.5.1	汇编子程序使用 C 程序的变量	(344)
15.5.2	C 程序使用汇编子程序的变量	(346)
15.6	C 调用外部汇编过程程序实例	(347)
15.7	内嵌式汇编过程	(353)
15.7.1	内嵌式汇编过程的建立	(353)
15.7.2	内嵌式汇编过程对数据的访问	(356)

附录 A Turbo C 程序在集成环境中的开发 (359)

A.1	Turbo C 集成开发环境	(359)
A.1.1	基本导航操作	(359)
A.1.2	Turbo C 的“热键”	(360)
A.1.3	菜单中的命令、开关及命名约定	(360)
A.1.4	主菜单	(361)
A.1.5	快速参考行	(361)
A.1.6	编辑窗口	(361)
A.1.7	信息窗口	(364)
A.2	菜单命令	(364)
A.2.1	文件菜单	(364)
A.2.2	编辑命令	(365)
A.2.3	运行命令	(365)
A.2.4	编译菜单	(365)
A.2.5	工程菜单	(366)
A.2.6	选择项菜单	(366)
A.2.7	调试菜单	(371)
A.3	Turbo C 程序的编译及运行	(372)
A.3.1	在集成开发环境中编译及连接 Turbo C 程序	(372)
A.3.2	建立单个源文件的可执行程序	(372)
A.3.3	在集成开发环境中调试 Turbo C 程序	(374)

A.3.4 使用多个源文件	(375)
A.3.5 Make 的其它一些特性	(379)
A.3.6 MAKE 实用程序	(379)
A.4 Turbo C 程序的开发过程	(380)
A.4.1 建立第一个 Turbo C 程序	(380)
A.4.2 修改第一个 Turbo C 程序	(381)
A.4.3 建立第二个 Turbo C 程序	(382)
附录 B C 语言关键字	(383)
附录 C 各种版本 C 编译系统标准函数比较	(384)
主要参考书目	(389)

绪 论

在详细介绍 C 语言之前,为了使读者对 C 语言有一个概括的了解,本章将简单地介绍 C 语言的发展历史及特点、C 语言程序的结构与格式以及 C 程序的开发过程。

1.1 C 语言的发展历史及特点

60 年代,随着计算机科学的迅速发展,高级程序设计语言得到了广泛地应用,然而,还没有一种可以用于书写操作系统和编译程序等系统程序的高级语言,人们不得不用汇编语言(或机器语言)来书写,但汇编语言存在着不可移植、可读性差、研制软件效率不如高级语言等缺点,给编程带来很多不便。为此,人们对能用于系统程序设计的高级语言的开发就变得势在必行了,于是,70 年代初产生了一种能够用来研制各种系统程序的高级语言——C 语言。

1.1.1 C 语言的发展历史

C 语言的出现是与 UNIX 操作系统紧密联系在一起, C 语言本身也有一个发展历史,表 1.1 给出了 C 语言的发展历史。

表 1.1 C 语言的发展历史

语言名	设计者	年份
CPL	C. Strachey 等	1968
BCPL	M. Richards	1969
B	K. Thompson	1970
C	D. M. Ritchie	1972

C 语言起源于 1968 年发表的 CPL (Combined Programming Language) 语言。它的许多重要思想来源于 Martin Richards 在 1969 年研制的 BCPL (Basic Combined Programming Language) 语言,以及以 BCPL 语言为基础的而由 Ken Thompson 在 1970 年研制成的 B 语言。K. Thompson 用 B 语言写了第一个 UNIX 操作系统,用在 PDP-7 计算机(现已被淘汰)上。D. M. Ritchie 1972 年在 B 语言的基础上研制出 C 语言,并用 C 语言写了第一个在 PDP-11 计算机上实现的 UNIX 操作系统。UNIX 操作系统的巨大成功也伴随着 C 语言的巨大成功。

目前,从微型到大型计算机都配有 C 编译程序。不仅在装配 UNIX 操作系统的机器上,而

且在非 UNIX 操作系统的机器上也配有多种 C 的编译程序。由于 C 语言本身具有许多特点,现在它已经成为在微、小、大、巨型计算机上,从系统程序设计到工程应用程序都能使用的一种高级程序设计语言。

1.1.2 C 语言的特点

C 语言的特点可以从多方面来阐述,这里仅从使用者的角度加以讨论,其主要特点如下:

(1) 表达能力强且灵活。C 语言是处于汇编语言和高级语言之间的一种记述性程序设计语言。C 语言既有面向硬件和系统,像汇编语言那样可以直接访问硬件的功能,又有高级语言面向用户、容易记忆、便于阅读和书写的优点。

(2) 程序结构清晰且紧凑。因为 C 语言程序通常由若干个函数组成,所以它是一种模块化程序设计语言。因此,它十分利于把整体程序分割成若干相对独立的功能模块。并且,它为程序模块间的相互调用以及数据传递提供了便利。这种模块化结构的程序不但清晰而且紧凑。

(3) 书写简单、易学。例如,C 语言用 `{ }` 来代替 Pascal 语言中的 `begin` 和 `end` 作为复合语句标号,它的运算符也尽量缩写等。

(4) 目标程序的质量高。C 语言提供了一个较大的运算符集合。并且其中大多数运算符与一般机器指令相一致,可直接翻译成机器代码,因此,用它编写程序生成的代码质量高。实践证明,其它高级语言相对汇编语言的代码效率要低得多,而 C 语言的代码效率只比汇编语言低 10%—20%。但 C 语言在描述问题时编程迅速、可读性好、表达能力强等优点是汇编语言无法相比的。

(5) 可移植性好。C 语言的语句中,没有依存于硬件的输入/输出语句,程序的输入/输出功能是通过调用输入/输出函数实现的。而这些函数是由系统提供的独立于 C 语言的程序模块库,因此,C 语言虽然具有直接访问硬件的功能,但 C 语言程序本身并不依存于机器硬件系统,从而便于在硬件结构不同的机种间实现程序的移植。

(6) C 语言是一种结构化程序设计语言,它提供了一整套循环、条件判断和转移语句,实现了对程序逻辑流程的有效控制,有利于结构化程序设计。

(7) C 语言提供了丰富的数据类型。它不仅具有字符型和几种尺寸的整型数以及单、双精度的浮点数等基本数据类型,而且允许程序员自己设计更为复杂的数据类型,如数组、结构、联合等来适应特殊的程序需求。

(8) C 语言允许程序员定义各种类型的变量指针和函数指针。指针是与机器内存地址相关的说明项,因此指针是让程序员以相同于机器码的形式存取内存的数据。正确地使用指针可提高程序的效率。C 语言还支持指针运算,允许程序员直接访问和操纵内存地址。

(9) C 语言的预处理是一种正文处理,是编译之前对正文文件(源程序文件)的再安排。其中,用得最多的是定义程序的常量、代替函数调用的宏(可较快运行)和基于某种特定条件的编译指令。

C 语言是一种很灵活的语言,它允许程序员做出各种决定。为了保持这种特性,C 语言很少在类型转换等方面加以强制性的限制,这通常是很有益的。但是,C 语言类型检验太弱,转换比较随便,存在着不安全因素。程序员在使用 C 语言时必须注意到这一点。

由于 C 语言具有上述众多特点,近年来迅速地得到广泛普及和应用。C 语言被称为“高级汇编语言”。特别是在微处理机和微型计算机的软件开发,以及各种软件工具的开发中,使用 C 语言的趋势日益增强,最近呈现出 C 语言有可能取代汇编语言的发展倾向。