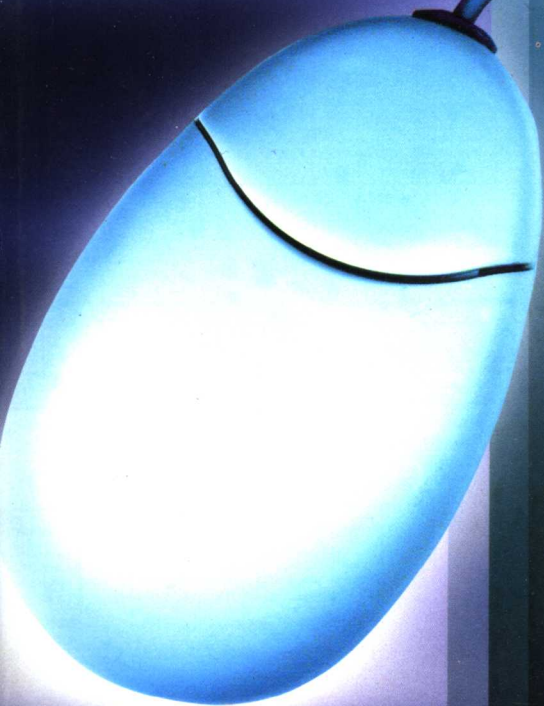


C 程序设计

范刚龙 王康平 主编



武汉理工大学出版社

C 程 序 设 计

主编 范刚龙 王康平

武汉理工大学出版社

【内 容 简 介】

C语言不仅是系统描述语言,而且是通用的程序设计语言。

本书共有12章,介绍了C语言的基本概念、语法规则和利用C语言进行程序设计的方法,并通过大量的程序实例进行了说明,每章后附有习题。

作者根据多年教学和写作的丰富经验,把C语言写得深入浅出、易于掌握。针对初学者的特点,对书的内容作了周密的安排,本书体系合理、概念清晰、例题丰富、逻辑性强、文字流畅、通俗易懂。

本书适合没有编程经验的读者,也可作为大专院校非计算机专业的教材,还可供学生自学。

图书在版编目(CIP)数据

C 程序设计/范刚龙,王康平主编. —武汉:武汉理工大学出版社,2006.9
ISBN 7-5629-2450-3

I. C… II. ①范… ②王… III. C语言-程序设计 IV. TP312

中国版本图书馆 CIP 数据核字(2006)第 109147 号

出版发行:武汉理工大学出版社(武汉市武昌珞狮路 122 号 邮编:430070)

印 刷 者:武汉理工大印刷厂

经 销 者:武汉理工大学出版社

开 本:787×1092 1/16

印 张:17

字 数:416 千字

版 次:2006 年 9 月第 1 版

印 次:2006 年 9 月第 1 次印刷

印 数:1~3000 册

定 价:26.00 元

(本书如有印装质量问题,请向承印厂调换)

前 言

C 语言功能丰富,使用灵活,可移植性好,既具有高级语言的优点,又具有低级语言的许多特点,既可以用来编写系统软件,又可用于编写应用软件,是国内外广泛使用的计算机语言。

本书是作者在总结多年来 C 语言教学经验的基础上编写的。作者近些年来曾先后写过适应不同层次需要的 C 语言教材和参考书,也得到一些读者的鼓励和批评以及希望和要求。根据教学中碰到的问题和读者的建议,以及不同层次学生对 C 语言的要求,我们编写了这本教材。该教材体系合理、逻辑清楚、例题丰富、通俗易懂;同时又采用了 C 语言的新标准,符合当前的需要。本书主要有如下几个特点:

针对性强。本书定位在初学高级语言的读者。通过这本书的学习使读者能够掌握一种计算机中较复杂的高级语言的用法。掌握了 C 语言再学习其他高级语言就容易多了。本书在写法上从头讲起,从浅入深,先讲概念再讲方法,从用法中加深对概念的理解。本书例题较多,每个例题讲述一个概念和一种方法。

可读性好。本书内容是作者多年讲述 C 语言教案的整理和补充。语言通俗易懂,重点突出,难点讲述清楚。应用实例贯穿于本书始终,所提出的 C 语言的概念和规则都将通过实例进行说明及验证,并结合实例进行详细的分析和说明。

可操作性高。学习高级语言的目的是为了编写程序。为了编好程序除了要学好高级语言外,还要学习算法。语言是编程的基础,在学习语言的过程中,一定要结合编程进行练习。本书给出的程序都是带有 main() 函数的完整程序,将该程序输入计算机后,都可获得其结果。读者在学习中可以对该程序的修改加深对其概念的理解。通过上机帮助读者搞清其语法和词法规则,加深对基本概念和规则的理解。本书附有上机实验指导书,为上机实践提供了大量的例题和习题。

程序设计是一门实践性很强的课程,只靠听课和看书就能掌握 C 语言程序设计是不可能的,应当十分重视自己动手编写程序和上机运行程序。

学习 C 语言时还有一点应当注意:应该把精力放在最基本、最常用的内容上。开始时不要钻牛角尖,在一些具体细节上死抠。有一些细节,随着对 C 语言的了解逐步深入和实践经验的逐步丰富,会自然掌握,而有一些细节则要通过长期的实践才能真正熟练掌握。

全书共 12 章。第 1、5 章由吴孝丽编写;第 2、4 章由王康平编写;第 3 章由郭力争编写;第 6、11 章由王妍玲编写;第 7、8 章由崔雪冰编写;第 9、10 章由范刚龙、张延红编写;第 12 章和附录由周焱编写;范刚龙、王康平任主编。

本书所有的例题和习题中的程序都在 Turbo C 2.0 版本的 C 语言编译系统下调试过。

由于作者水平有限,经验不多,再加上编写时间仓促,书中肯定有不少缺点或错误,希望得到专家和读者的指正,在适当的时间再作修订补充,以跟上计算机科学技术的发展需要。
E-mail:fangl@hncj.edu.cn。

范刚龙
2005 年 12 月

目 录

1 C语言概述	(1)
1.1 C语言出现的历史背景	(1)
1.1.1 C语言的出现	(1)
1.1.2 C语言的发展	(1)
1.2 C语言的特点	(2)
1.2.1 C语言的语言成分简洁、紧凑、书写形式自由	(2)
1.2.2 C语言拥有丰富的数据类型	(2)
1.2.3 C语言的运算符丰富、功能更强大	(2)
1.2.4 C语言是结构化的程序设计语言	(3)
1.2.5 C语言对语法限制不严格,程序设计灵活	(3)
1.2.6 C语言编写的程序具有良好的可移植性	(3)
1.2.7 C语言可以实现汇编语言的大部分功能	(4)
1.2.8 C语言编译后生成的目标代码少、质量高,程序的执行效率高	(4)
1.3 C语言基本语法成分	(4)
1.3.1 字符集	(4)
1.3.2 标识符(名字)	(4)
1.3.3 关键字(保留字)	(5)
1.3.4 运算符	(5)
1.3.5 分隔符	(5)
1.3.6 注释符	(5)
1.4 C语言程序结构	(5)
1.5 C语言程序的上机步骤	(8)
1.5.1 源程序、目标程序、可执行程序的概念	(8)
1.5.2 C语言程序的上机步骤	(8)
本章小结	(9)
2 数据类型、运算符与表达式	(10)
2.1 C语言的数据类型	(10)
2.2 常量和变量	(11)
2.2.1 常量与符号常量	(11)
2.2.2 变量	(11)
2.3 整型数据	(12)
2.3.1 整型常量	(12)
2.3.2 整型变量	(13)
2.4 实型数据	(14)

2.4.1	实型常量	(14)
2.4.2	实型变量	(15)
2.4.3	实型常量的类型	(16)
2.5	字符型数据	(16)
2.5.1	字符常量	(16)
2.5.2	字符变量	(18)
2.5.3	字符数据在内存中的存储形式及其使用方法	(20)
2.5.4	字符串常量	(20)
2.6	变量赋初值	(21)
2.7	各类数值型数据间的混合运算	(21)
2.7.1	自动转换	(22)
2.7.2	强制转换	(23)
2.8	运算符与表达式	(23)
2.8.1	C 运算符简介	(23)
2.8.2	算术运算符与算术表达式	(24)
2.8.3	赋值运算符与赋值表达式	(26)
2.8.4	自增、自减运算符	(27)
2.8.5	逗号运算符与逗号表达式	(28)
本章小结		(28)
3	顺序结构程序设计	(31)
3.1	C 语句概述	(31)
3.1.1	控制语句	(31)
3.1.2	表达式语句	(33)
3.1.3	复合语句	(33)
3.1.4	空语句	(33)
3.2	赋值语句	(33)
3.3	数据输出	(34)
3.3.1	格式化输出函数	(34)
3.3.2	格式字符串	(35)
3.4	数据输入	(37)
3.4.1	格式化输入函数	(37)
3.4.2	格式字符串	(37)
3.4.3	关于输入方法	(37)
3.5	程序举例	(38)
3.6	算法与程序设计	(40)
3.6.1	算法及其表示方法	(40)
3.6.2	结构化程序设计	(42)
本章小结		(44)

4 选择结构程序设计	(47)
4.1 关系运算和逻辑运算	(47)
4.1.1 关系运算符与关系表达式	(47)
4.1.2 逻辑运算符与逻辑表达式	(48)
4.2 if 语句	(49)
4.2.1 if 语句的三种形式	(50)
4.2.2 if 语句的嵌套	(53)
4.2.3 条件运算符	(54)
4.3 switch 语句	(55)
4.3.1 switch 语句的一般书写格式	(55)
4.3.2 switch 语句的执行过程	(55)
4.4 程序设计举例	(56)
本章小结	(59)
5 循环结构的程序设计	(61)
5.1 概述	(61)
5.2 goto 语句以及用 goto 语句构成循环	(61)
5.3 while 语句	(62)
5.4 do-while 语句	(64)
5.5 for 语句	(67)
5.6 循环的嵌套	(71)
5.7 几种循环的比较	(72)
5.8 break 和 continue 语句	(74)
5.8.1 break 语句	(74)
5.8.2 continue 语句	(75)
5.9 程序举例	(77)
本章小结	(81)
6 数组	(85)
6.1 一维数组的定义和引用	(85)
6.1.1 一维数组的定义	(85)
6.1.2 一维数组元素的引用	(86)
6.1.3 一维数组的初始化	(87)
6.1.4 一维数组程序举例	(87)
6.2 二维数组的定义和引用	(89)
6.2.1 二维数组的定义	(89)
6.2.2 二维数组的引用	(90)
6.2.3 二维数组的初始化	(91)
6.2.4 二维数组程序举例	(91)
6.3 字符数组	(92)
6.3.1 字符数组的定义、初始化及引用	(93)

6.3.2	字符串和字符串结束标志	(93)
6.3.3	字符数组的输入输出	(94)
6.3.4	字符串处理函数	(95)
6.3.5	字符数组应用举例	(99)
	本章小结	(100)
7	函数	(103)
7.1	概述	(103)
7.2	掌握函数定义的一般形式	(104)
7.3	函数参数和函数的值	(105)
7.3.1	形式参数与实际参数	(105)
7.3.2	函数的返回值	(107)
7.4	函数调用	(107)
7.4.1	函数调用的一般形式	(107)
7.4.2	函数调用的方式	(107)
7.4.3	函数实参类型	(109)
7.4.4	对被调用函数的声明和函数原型	(110)
7.5	函数的嵌套调用	(111)
7.6	函数的递归调用	(112)
7.7	数组作为函数参数	(113)
7.7.1	数组名作函数参数	(114)
7.7.2	用多维数组作函数参数	(115)
7.8	局部变量和全局变量	(115)
7.8.1	局部变量	(115)
7.8.2	全局变量	(117)
7.9	几种常用变量类型的区别和存储类别	(119)
7.9.1	动态存储方式与静态存储方式	(119)
7.9.2	auto 变量	(119)
7.9.3	用 static 声明局部变量	(120)
7.9.4	register 变量	(120)
7.9.5	用 extern 声明外部变量	(121)
7.9.6	用 static 声明外部变量	(122)
7.9.7	关于变量的声明和定义	(122)
7.9.8	存储类别小结	(123)
7.10	内部函数和外部函数	(125)
7.10.1	内部函数	(125)
7.10.2	外部函数	(125)
	本章小结	(125)
8	预处理命令	(128)
8.1	宏定义	(128)

8.1.1 不带参数的宏定义.....	(128)
8.1.2 带参数的宏定义.....	(130)
8.2 文件包含.....	(133)
8.3 条件编译.....	(133)
本章小结	(135)
9 指针	(138)
9.1 地址和指针的概念.....	(138)
9.1.1 内存和内存地址.....	(138)
9.1.2 变量的地址、指针、指针变量.....	(138)
9.2 变量的指针和指向变量的指针变量.....	(139)
9.2.1 指针变量的定义.....	(139)
9.2.2 指针变量赋值.....	(140)
9.2.3 指针变量引用.....	(140)
9.2.4 指针变量作为函数参数.....	(141)
9.3 数组的指针和指向数组的指针变量.....	(143)
9.3.1 指向数组元素的指针.....	(143)
9.3.2 通过指针引用数组元素.....	(144)
9.3.3 数组名作为函数参数.....	(147)
9.3.4 指向多维数组的指针和指针变量.....	(150)
9.4 字符串的指针和指向字符串的指针变量.....	(155)
9.4.1 字符串的表示形式.....	(155)
9.4.2 字符串指针作函数参数.....	(157)
9.4.3 字符数组与字符串指针区别.....	(158)
9.5 函数的指针和指向函数的指针变量.....	(159)
9.5.1 用函数指针变量调用函数.....	(159)
9.5.2 用指向函数的指针变量作函数参数.....	(161)
9.6 返回指针值的函数.....	(162)
9.7 指针数组与指向指针的指针.....	(164)
9.7.1 指针数组的概念.....	(164)
9.7.2 指向指针的指针.....	(166)
9.7.3 指针数组作 main 函数的参数	(167)
9.8 指针运算举例.....	(168)
9.9 有关指针的数据类型和指针运算的小结.....	(171)
9.9.1 有关指针的数据类型小结.....	(171)
9.9.2 指针运算小结.....	(171)
9.9.3 void 指针类型	(172)
本章小结	(172)
10 结构体与共用体.....	(174)
10.1 概述.....	(174)

10.2	定义结构体类型变量的方法	(175)
10.2.1	结构体的引入	(175)
10.2.2	定义结构体变量的方法	(175)
10.3	结构体类型变量的引用	(177)
10.4	结构体变量的初始化	(179)
10.5	结构体数组	(181)
10.5.1	定义结构体数组	(181)
10.5.2	结构体数组的初始化	(182)
10.5.3	结构体数组应用举例	(182)
10.6	指向结构体数据类型的指针	(184)
10.6.1	指向结构体变量的指针	(184)
10.6.2	指向结构体数组的指针	(185)
10.6.3	用结构体变量和指向结构体的指针作函数参数	(186)
10.7	用指针处理链表	(187)
10.7.1	链表概述	(187)
10.7.2	简单链表	(187)
10.7.3	处理动态链表所需的函数	(188)
10.7.4	建立动态链表	(189)
10.7.5	输出链表	(192)
10.7.6	对链表的删除操作	(193)
10.7.7	对链表的插入操作	(195)
10.7.8	对链表的综合操作	(198)
10.8	共用体	(199)
10.8.1	共用体的概念	(199)
10.8.2	共用体变量的引用方式	(200)
10.8.3	共用体类型数据的特点	(201)
10.9	枚举类型	(202)
10.9.1	枚举类型定义	(202)
10.9.2	枚举变量定义	(202)
10.9.3	有关枚举的说明	(202)
10.10	用 typedef 定义类型	(203)
	本章小结	(204)
11	文件	(206)
11.1	C 文件概述	(206)
11.2	文件类型指针	(207)
11.3	文件的打开与关闭	(207)
11.3.1	文件的打开(fopen 函数)	(208)
11.3.2	文件的关闭(fclose 函数)	(209)
11.4	文件的读写	(210)

11.4.1	fputc 函数和 fgetc 函数	(210)
11.4.2	fread 函数和 fwrite 函数	(213)
11.4.3	fprintf 函数和 fscanf 函数	(216)
11.4.4	其他读写函数	(216)
11.5	文件的定位	(217)
11.5.1	rewind 函数	(217)
11.5.2	fseek 函数和随机读写	(218)
11.5.3	ftell 函数	(220)
11.6	出错的检测	(220)
11.6.1	ferror 函数	(220)
11.6.2	clearerr 函数	(220)
	本章小结	(220)
12	常见错误和程序调试	(222)
12.1	常见错误分析	(222)
12.2	程序调试	(228)
附录 1		(229)
附录 2		(249)
附录 3		(255)
参考文献		(257)

1 C 语言概述

C 语言是国际上流行的、很有发展前途的计算机高级语言。C 语言适合作为“系统描述语言”。它既可以用来编写系统软件,也可以用来编写应用程序。

1.1 C 语言出现的历史背景

1.1.1 C 语言的出现

早期的操作系统等系统软件(包括 UNIX 操作系统)主要是采用汇编语言编写的。但是,汇编语言存在明显的缺点,它依赖于计算机硬件,程序的可读性、可移植性都比较差。为了提高可读性和可移植性,人们希望能找到一种既具有一般高级语言特性,又具有低级语言底层操作能力的语言来编写系统软件,于是 C 语言在 20 世纪 70 年代初应运而生了。

1.1.2 C 语言的发展

1978 年由美国电话电报公司(AT&T)的贝尔实验室设计出了 C 语言,同时由 B. W. Kernighan 和 D. M. Ritchie 合著了影响深远的《The C Programming Language》一书,通常简称为 K&R,也有人称之为 K&R 标准。但是,在 K&R 中并没有定义一个完整的标准 C 语言,许多开发机构推出了自己的 C 语言版本,这些版本之间的微小差别不时引起兼容性上的问题,后来由美国国家标准学会 ANSI(American National Standard Institute)在各种 C 语言版本的基础上制定了一个 C 语言标准,于 1983 年发表。通常称之为 ANSI C。1987 年 ANSI 又公布了新标准——87 ANSI C。目前广泛流行的各种 C 语言编译系统都是以它为基础的。

早期的 C 语言主要是用于编写 UNIX 系统,由于 C 语言的强大功能和各方面的优点逐渐为人们认识,到了 20 世纪 80 年代,C 语言开始进入其他操作系统,并很快在各类大、中、小和微型计算机上得到了广泛的使用,成为当代最优秀的程序设计语言之一。

在 C 语言的基础上,1980 年又由贝尔实验室的 Bjarne Stroustrup 推出了 C++。C++ 进一步扩充和完善了 C 语言,成为一种面向对象的程序设计语言。C++ 提出了一些更为深入的概念,它所支持的这些面向对象的概念容易将问题空间直接映射到程序空间,为程序员提供了一种与传统结构程序设计不同的思维方式和编程方法。因而也增加了整个语言的复杂性,掌握起来有一定难度。但是,C 语言是 C++ 语言的基础,C++ 语言和 C 语言在很多方面是兼容的。因此,掌握了 C 语言,再进一步学习 C++ 语言就能以一种熟悉的语法来学习面向对象的语言,从而达到事半功倍的目的。

目前最流行的 C 语言有以下几种:

- (1) Microsoft C 或称 MS C
- (2) Borland Turbo C 或称 Turbo C
- (3) AT&T C

这些 C 语言版本不仅实现了 ANSI C 标准,而且在此基础上各自作了一些扩充,使之更加方便、完美。这些不同版本 C 语言之间有一定的差别,但对初学者来说,不必过多理会他们的差别,重在理解 C 语言的特点和编程方法。本书叙述以 Turbo C 为准。

1.2 C 语言的特点

C 语言是从“组合编程语言”CPL 发展而来,C 语言既具有一般高级语言的特性(ALGOL60 带来的高级语言的特性),又具有低级语言的特性(BCPL 带来的接近硬件的低级语言特性)。C 语言具有下面特点(其中 1.2.1~1.2.6 属于高级语言特性,1.2.7~1.2.8 属于低级语言特性)。

1.2.1 C 语言的语言成分简洁、紧凑、书写形式自由

例:将 C 语言程序段与实现同样功能的 Pascal 语言程序段进行比较。

表 1.1 C 语言与 Pascal 语言比较

序号	C 语言	Pascal 语言	含义	说明
(1)	{...}	begin...end	复合语句(或语句块)	Pascal 显得啰唆
(2)	if(e)S;	if(e)then S;	条件语句	Pascal 至少多了一个 then 关键字
(3)	int i;	var i;integer	定义 i 为整型变量	Pascal 至少多了一个 var 关键字
(4)	int a[10];	var a: array[1...10] of integer	定义 a 为整型一维数组,10 个元素	Pascal 多了 var、array、of 等关键字
(5)	int f();	function f();integer	定义 f 为返回值整型的函数	Pascal 至少多了一个 function 关键字
(6)	int *p;	var p: ↑ integer	定义 p 为指向整型变量的指针变量	Pascal 至少多了一个 var 关键字
(7)	i+2=2;	i:=i+2	赋值语句	C 语言中如果将一个变量与另外一个操作数运算后赋值给原来的变量,使用复合的运算符可以不重复书写此变量。C 语言形式上更加简洁
(8)	i++, ++i	i:=i+1	i 自增 1	C 语言定义了常用的自增 1、自减 1 运算符,形式上显得相当简洁

1.2.2 C 语言拥有丰富的数据类型

C 语言具有整型、实型、字符型、数组类型、指针类型、结构体类型、共用体类型等数据类型,能方便地构造更加复杂的数据结构(如:使用指针构造链表、树、栈)。

1.2.3 C 语言的运算符丰富、功能更强大

例如:

(1) C语言具有复合的赋值运算符“+(-、*、/、%)=”,“>>(<<)=”,“&(^、|=)”。

$x+=5$ 等价于 $x=x+5$

(2) C语言有条件运算符“?:”可代替简单的 if/else 语句。

如果需要表示:“如果 x 小于或等于 0, y 为 0; 否则 y 为 1”可以采用:

$y=x \leq 0 ? 0 : 1;$

如果用一般的程序设计语言表示如下:

if($x \leq 0$) $y=0;$

else $y=1;$

(3) C语言中连赋值操作都定义为运算符,也就是说赋值操作本身可以作为表达式的一部分,参与运算。如:

```
if((p=malloc(sizeof(int)))==NULL){printf("Error!");exit(1);}
```

如果改写为一般形式:

```
p=malloc(sizeof(int));
```

```
if(p==NULL){printf("Error!");exit(1);}
```

又如下面算式是正确的:

```
x=y=z=6;
```

如果改写为一般形式:

```
z=6;y=6;x=6;
```

1.2.4 C语言是结构化的程序设计语言

(1) C语言具有结构化的控制语句(if/else, switch/case, for, while, do...while)

(2) 函数是 C语言程序的模块单位。

1.2.5 C语言对语法限制不严格,程序设计灵活

C语言不检查数组下标越界,不限制对各种数据转化(编译系统可能对不合适的转化进行警告,但不限制),不限制指针的使用,程序正确性由程序员保证。

实践中, C语言程序编译时会提示:“警告错误”,“严重错误”。“警告错误”表示你使用的语法可能有问题,但是你有时可以忽略,你的程序仍然可以完成编译工作,然后运行(但是一般情况下“警告错误”,往往意味着程序真的有问题,你应该认真地检查)。“严重错误”是不能忽略的,编译系统发现严重错误,就不会产生目标代码。

灵活和安全是一对矛盾,对语法限制的不严格可能也是 C语言的一个缺点,比如,“黑客”可能使用越界的数组攻击你的计算机系统。java 语言是优秀的网络应用程序开发语言,它必须保证安全性,它绝对不允许数组越界。此外 java 不使用指针,不能直接操作客户计算机上的文件,语法检查相当严格,程序正确性容易保证,但是 java 在编程时却缺乏灵活性。

1.2.6 C语言编写的程序具有良好的可移植性

编制的程序基本上不需要修改或只需要少量修改就可以移植到其他的计算机系统或其他的操作系统。

1.2.7 C 语言可以实现汇编语言的大部分功能

- (1) C 语言可以直接操作计算机硬件,如寄存器,各种外设 I/O 端口等。
- (2) C 语言的指针可以直接访问内存物理地址。
- (3) C 语言类似汇编语言的位操作可以方便地检查系统硬件的状态。
- (4) C 语言适合编写系统软件。

1.2.8 C 语言编译后生成的目标代码少、质量高,程序的执行效率高

有资料显示只比汇编代码效率低 10%~20%。

1.3 C 语言基本语法成分

本节介绍 C 语言的字符集、关键字、标识符、运算符、分隔符和注释符等基本语法成分。

1.3.1 字符集

字符是 C 语言的最基本的元素,C 语言字符集由字母、数字、空白符、标点和特殊字符组成(在字符串常量和注释中还可以使用汉字等其他图形符号)。由字符集中的字符可以构成 C 语言的语法成分(如标识符,关键字,运算符等)。

(1) 字母: A~Z, a~z

(2) 数字: 0~9

(3) 空白符: 空格,制表符(跳格),换行符(空行)的总称。空白符除了在字符、字符串中有意义外,编译系统忽略其他位置的空白符。空白符在程序中只是起到间隔作用。在程序的恰当位置使用空白符将使程序更加清晰,增强程序的可读性。

(4) 标点符号、特殊字符: 主要有 !、#、%、^、&、+、-、*、/、=、~、<、>、|、.、,、:、;、?、'、"、(、)、{、}、[、] 等。

1.3.2 标识符(名字)

在程序中使用的变量名、函数名、标号等统称为标识符,用来标识各种程序成分。除库函数的函数名由系统定义外,其余都由用户自定义。C 语言规定,标识符只能是由字母(A~Z, a~z)、数字(0~9)、下划线“_”组成的字符串,并且其第一个字符必须是字母或下划线。

以下标识符是合法的:

a, x, x3, BOOK1, sum5, num_1。

以下标识符是非法的:

3s 以数字开头;

s * T 出现非法字符 *;

-3x 以减号开头;

bowy-1 出现非法字符-(减号)。

在使用标识符时还必须注意以下几点:

- (1) 标准 C 不限制标识符的长度,但它受各种版本的 C 语言编译系统限制,同时也受到具

体机器的限制。例如在某版本 C 语言中规定标识符前八位有效,当两个标识符前八位相同时,则被认为是同一个标识符。Turbo C 中标识符最大长度为 32 个字符。

(2)在标识符中,大小写是有区别的。例如 BOOK 和 book 是两个不同的标识符。习惯上符号常量用大写字母表示,而变量名等用小写字母表示。

标识符虽然可由程序员随意定义,但不能与关键字同名,也不能与系统预先定义的标准标识符(如标准函数)同名。标识符是用于标识某个量的符号,因此,命名应尽量有相应的意义,以便阅读理解,做到“见名知义”。

1.3.3 关键字(保留字)

关键字是由 C 语言规定的具有特定意义的字符串,通常也称为保留字。如类型说明符 int, double 等;语句特征符 if, switch, while 等;预处理命令 include, define 等。关键字是构成 C 语言的语法基础,用户定义的标识符不应与关键字相同,也不能对关键字进行重新定义。

1.3.4 运算符

C 语言中含有相当丰富的运算符。运算符与变量、函数一起组成表达式,表示各种运算功能。运算符由一个或多个字符组成。根据参加运算对象的个数,运算符可分为单目运算符、双目运算符和三目运算符。

1.3.5 分隔符

C 语言中的分隔符有逗号和空格两种,逗号主要用在类型说明和函数参数表中,分隔各个变量。空格多用于语句各单词之间作间隔符。在关键字、标识符之间必须要有一个以上的空格符作间隔,否则将会出现语法错误。例如把“int a;”,写成“inta;”,C 语言编译器会把“inta”当成一个标识符处理,其结果必然出错。

1.3.6 注释符

注释符是以“/*”开头并以“*/”结尾的串。编译系统将/* ... */之间的所有内容看作为注释,编译时系统忽略注释。注释可出现在程序中的任何位置。注释用来向用户提示或解释程序的意义。

注释与软件的文档同等重要,要养成使用注释的良好习惯,这对软件的维护相当重要。记住:程序是给别人看的,清晰的注释有助于他人理解您的程序、算法的思路。

在软件开发过程中,还可以将注释用于在程序调试时暂时屏蔽一些语句。

例如,在调试程序时暂时不需要运行某段语句,而你又不希望立即从程序中删除它,可以使用注释符将这段程序框起来,暂时屏蔽这段程序,以后可以方便地恢复。

1.4 C 语言程序结构

为了说明 C 语言源程序结构的特点,先看以下几个程序。这几个程序由易到难,表现了 C 语言源程序在组成结构上的特点。虽然有关内容还未介绍,但可从这些例子中了解到组成一个 C 语言源程序的基本内容和书写格式。

【例 1.1】

```
main()
{
    printf("Hello world! \n");
}
```

main 是主函数的函数名,表示这是一个主函数,每一个 C 语言源程序都必须有,并且只能有一个主函数(main 函数),这是整个 C 语言程序运行的入口点。printf 是 C 语言中的输出函数(详见第三章),是一个由系统定义的标准函数,可在程序中直接调用,它的功能是把要输出的内容送到显示器去显示。双引号内字符串原样输出,“\n”是回车换行符。

【例 1.2】

```
/* 求两数的和 */
main()
{
    int a,b,sum;      /* 这是定义变量 */
    a=12;      b=34;
    sum=a+b;
    printf("sum is %d\n",sum);
}
```

本程序的作用是求两个整数 a 和 b 之和。/*...*/表示注释部分,为便于理解,这里用汉字作注释,当然也可以用英语或汉语拼音作注释。注释只是用来向用户提示或解释程序的意义,对程序编译不起任何作用。注释可出现在程序中的任何位置。程序第三行是变量说明部分,说明 a 和 b 是整型(int)变量。第四行是两个赋值语句,使 a 和 b 的值分别为 12 和 34。第五行使 sum 的值为 a+b。第六行中“%d”是输入输出的格式字符串,用来指定输入输出时的数据类型和格式(详见第三章),“%d”表示十进制整数类型。在执行输出时此位置上代以一个十进制整数值。printf 函数中括号内最右端 sum 是要输出的变量,现在它的值为 46(即 12+34=46),因此输出一行信息为:

sum is 46

【例 1.3】

```
/* 此函数的功能是输入两个整数,输出其中的大数 */
#include "stdio.h"
int max(int a,int b);      /* 函数说明 */
main()      /* 主函数 */
{
    int x,y,z;      /* 变量说明 */
    printf("input two numbers:\n");
    scanf("%d%d",&x,&y);      /* 输入 x,y 值 */
    z=max(x,y);      /* 调用 max 函数 */
    printf("maximum=%d",z); /* 输出 */
}
int max(int a,int b)      /* 定义 max 函数 */
```