

● 主编 密君英



C语言 程序设计基础

◆ 苏州大学出版社

C 语言程序设计基础

主 编 密君英
副主编 刘海明 方 蓓

苏州大学出版社

图书在版编目(CIP)数据

C 语言程序设计基础/密君英主编. —苏州: 苏州大学出版社, 2005. 12

ISBN 7-81090-602-X

I. C… II. 密… III. C 语言-程序设计-高等学校; 技术学校-教材 IV. TP312

中国版本图书馆 CIP 数据核字(2006)第 003024 号

C 语言程序设计基础

密君英 主编

责任编辑 周建兰

苏州大学出版社出版发行

(地址: 苏州市干将东路 200 号 邮编: 215021)

丹阳兴华印刷厂印装

(地址: 丹阳市胡桥镇 邮编: 212313)

开本 787mm×1092mm 1/16 印张 14.5 字数 361 千

2005 年 12 月第 1 版 2005 年 12 月第 1 次印刷

ISBN 7-81090-602-X/TP·42(课) 定价: 33.00 元

苏州大学版图书若有印装错误, 本社负责调换
苏州大学出版社营销部 电话: 0512-67258835

前 言

C 语言因其功能强大、内容丰富、使用灵活、程序执行效率高、可移植性好、既具有高级语言的特点、又具有汇编语言的一些特点,长期以来一直备受广大程序设计爱好者的青睐。在面向对象的程序设计理念流行的今天,有些人认为结构化程序设计已没有必要再专门学习,但作为程序设计的入门课程,我们必须严格地培养学生的编程习惯和思维,如一开始就让学生接触 Java、C++ 等语言则有点困难。无疑 C 语言成了一门最好的入门语言,严谨而又灵活的语法规则,必定能养成学生良好的编程思路和编程习惯,为学习其他高级语言奠定良好的基础。

全书共分 12 章,主要包括 C 语言概述、数据类型、运算符和表达式、顺序结构、选择结构、循环结构、数组、函数、指针、编译预处理、结构体与共用体、位运算、文件等。本书在表达上尽量采用通俗易懂的语言,以确切、详尽的例题来说明问题,使读者形象、快捷地掌握知识点并能够熟练运用,并在每章的最后配套了大量的、形式多样的习题,能帮助学生更好地学习和巩固所学知识。

本教材编写时注意以实例为主,强调了理论与实际结合,特别适合高职高专的学生学习。当然,本书也可作为其他学生学习 C 语言的参考书和等级考试的参考教材。

本书由密君英主编,刘海明、方蓓为副主编,吴凯益为参编,其中第一、二、三章由密君英编写,第四、五章由吴凯益编写,第六、七、十一章由方蓓编写,第八、九、十、十二章由刘海明编写。徐晓明教授在百忙之中通阅了全书,并提出了许多宝贵的建议,在此深表感谢。在本书的编写过程中,还得到了苏州大学李凡长教授、苏州农业职业技术学院电子系相关老师的指导与帮助,在此一并表示感谢。

由于作者水平有限,加上时间仓促,书中的错误和不足在所难免,我们真诚希望得到广大读者的批评指正。

编 者

2005 年 10 月



第1章 C语言概述

1.1 C语言的发展及特点	(1)
1.1.1 C语言的发展简史	(1)
1.1.2 C语言的特点	(2)
1.2 结构化程序设计的基本概念	(3)
1.2.1 顺序结构	(3)
1.2.2 选择结构	(4)
1.2.3 循环结构	(4)
1.3 简单的C程序介绍	(5)
1.4 C程序的上机步骤	(7)
1.4.1 运行一个C语言源程序的一般过程	(8)
1.4.2 TC的启动、退出与命令菜单	(8)
1.4.3 编辑并保存一个C语言源程序	(10)
1.4.4 编译、连接单个源程序文件	(11)
1.4.5 运行与查看结果	(11)
1.4.6 编辑下一个新的源程序	(12)
习题	(12)

第2章 数据类型、运算符和表达式

2.1 C语言的数据类型	(13)
2.2 常量、变量与标识符	(14)
2.2.1 标识符	(14)
2.2.2 常量	(14)
2.2.3 变量	(15)
2.3 整型数据	(16)
2.3.1 整型常量的表示方法	(16)
2.3.2 整型变量	(16)
2.3.3 整型常量的类型	(18)
2.4 实型数据	(18)
2.4.1 实型常量的表示方法	(18)
2.4.2 实型变量	(19)
2.4.3 实型常量的类型	(20)

2.5	字符型数据	(20)
2.5.1	字符型常量	(20)
2.5.2	字符型变量	(21)
2.5.3	字符型数据在内存中的存储形式及其使用方法	(21)
2.5.4	字符串常量	(22)
2.6	算术运算符与算术表达式	(23)
2.6.1	C 运算符简介	(23)
2.6.2	算术运算符与算术表达式	(23)
2.7	赋值运算符与赋值表达式	(25)
2.7.1	赋值运算符	(25)
2.7.2	赋值表达式	(26)
2.8	C 语言特有的运算和运算符	(26)
2.8.1	自增(++)、自减(--)运算	(26)
2.8.2	逗号运算符(,)和逗号表达式	(27)
	习题	(28)

第3章 顺序结构

3.1	C 语句概述	(31)
3.2	数据输入/输出的概念及在 C 语言中的实现	(32)
3.3	单个字符数据的输入/输出	(33)
3.3.1	putchar 函数(单个字符的输出)	(33)
3.3.2	getchar 函数(单个字符的输入)	(33)
3.4	格式化输出——printf() 函数	(34)
3.4.1	printf() 函数的一般格式	(34)
3.4.2	格式指示符	(35)
3.4.3	使用说明	(38)
3.5	格式化输入——scanf() 函数	(38)
3.5.1	scanf() 函数的一般格式	(38)
3.5.2	格式指示符	(39)
3.5.3	使用 scanf() 函数时应注意的问题	(40)
3.6	顺序结构程序设计举例	(41)
	习题	(44)

第4章 选择结构

4.1	关系运算	(48)
4.1.1	关系运算符的分类及优先级	(48)
4.1.2	关系表达式	(48)
4.2	逻辑运算	(48)
4.2.1	逻辑运算符的分类及优先级	(48)
4.2.2	逻辑表达式	(49)
4.3	if 语句	(49)

4.3.1 if 语句的三种格式	(49)
4.3.2 if 的嵌套	(52)
4.3.3 条件运算符	(54)
4.4 switch 语句	(54)
4.5 选择结构程序设计举例	(56)
习题	(58)
第5章 循环控制	
5.1 循环的概述及 goto 循环	(63)
5.1.1 循环的概述	(63)
5.1.2 goto 语句	(63)
5.2 while 循环	(64)
5.3 do-while 循环	(64)
5.4 for 循环	(65)
5.5 continue 语句和 break 语句	(68)
5.5.1 continue 语句	(68)
5.5.2 break 语句	(68)
5.6 循环的嵌套	(68)
5.7 循环结构程序设计举例	(69)
习题	(73)
第6章 数组	
6.1 一维数组	(79)
6.1.1 一维数组的定义	(79)
6.1.2 一维数组元素的引用	(80)
6.1.3 一维数组的初始化	(80)
6.1.4 一维数组程序举例	(81)
6.2 二维数组	(83)
6.2.1 二维数组的定义	(83)
6.2.2 二维数组的引用	(84)
6.2.3 二维数组的初始化	(85)
6.2.4 二维数组程序举例	(86)
6.3 字符数组	(87)
6.3.1 字符数组的定义	(87)
6.3.2 字符数组的初始化	(87)
6.3.3 字符数组的引用	(88)
6.3.4 字符串和字符串结束标志	(89)
6.3.5 字符数组的输入/输出	(90)
6.3.6 字符串处理函数	(91)
6.3.7 字符数组应用举例	(93)
6.4 数组程序举例	(94)

习题	(99)
第7章 函数	
7.1 函数的基本知识	(103)
7.1.1 函数概述	(103)
7.1.2 函数的定义	(105)
7.2 函数的调用及其数据传送	(106)
7.2.1 形式参数和实际参数	(106)
7.2.2 函数的返回值	(107)
7.2.3 函数的调用	(108)
7.2.4 函数的嵌套调用	(112)
7.2.5 数组作为函数的参数	(114)
7.3 函数的递归	(119)
7.4 变量的作用范围	(121)
7.4.1 局部变量	(121)
7.4.2 全局变量	(122)
7.5 变量的存储类别	(124)
7.5.1 动态存储方式与静态存储方式	(124)
7.5.2 auto 变量	(125)
7.5.3 static 变量	(125)
7.5.4 register 变量	(127)
7.5.5 extern 变量	(128)
7.5.6 用 static 声明外部变量	(130)
7.6 内部函数和外部函数	(130)
7.6.1 内部函数	(130)
7.6.2 外部函数	(131)
7.7 运行一个多文件程序	(131)
习题	(132)
第8章 指针	
8.1 地址和指针的概念	(137)
8.1.1 内存地址	(137)
8.1.2 直接和间接访问	(138)
8.2 指针的相关操作	(138)
8.2.1 指针变量的定义	(138)
8.2.2 指针变量的引用	(139)
8.2.3 指针变量的运算	(140)
8.2.4 指针变量作为函数参数	(140)
8.3 指针与数组	(141)
8.3.1 指针与一维数组	(141)
8.3.2 指针与二维数组	(143)

8.3.3 指针数组与数组指针	(145)
8.3.4 数组名作为函数参数	(146)
8.4 指针与字符串	(147)
8.4.1 字符串指针的定义	(147)
8.4.2 字符指针变量与字符数组的区别	(148)
8.4.3 指针数组与字符串应用	(148)
8.4.4 字符串指针作为函数的参数	(149)
8.5 指针与函数	(150)
8.5.1 指向函数的指针	(150)
8.5.2 返回指针值的函数(指针函数)	(151)
8.5.3 指针数组作为 main 函数的参数	(152)
习题	(153)
第9章 编译预处理	
9.1 编译预处理的概念	(159)
9.2 宏定义	(159)
9.2.1 不带参数的宏定义	(159)
9.2.2 带参数的宏定义	(160)
9.3 文件包含	(162)
9.4 条件编译	(162)
习题	(164)
第10章 结构体与共用体	
10.1 概述	(168)
10.2 结构体	(168)
10.2.1 结构体类型的定义方法	(168)
10.2.2 结构体变量的定义	(169)
10.2.3 结构体变量的引用	(170)
10.2.4 结构体变量的初始化	(170)
10.2.5 结构体数组	(170)
10.2.6 结构体指针	(172)
10.3 链表	(175)
10.3.1 链表的概念	(175)
10.3.2 链表的基本操作	(176)
10.4 联合	(178)
10.4.1 联合的概念	(178)
10.4.2 联合类型变量的引用	(179)
10.5 枚举类型变量的定义及引用	(181)
习题	(182)
第11章 位运算	
11.1 位运算符和位运算	(188)

11.1.1	“按位取反”运算符(~)	(188)
11.1.2	“左移”运算符(<<)	(189)
11.1.3	“右移”运算符(>>)	(189)
11.1.4	“按位与”运算符(&)	(189)
11.1.5	“按位异或”运算符(^)	(190)
11.1.6	“按位或”运算符()	(190)
11.1.7	位运算赋值运算符	(191)
11.1.8	不同长度的数据进行位运算	(191)
11.2	位运算举例	(191)
11.3	位段	(192)
	习题	(194)
第12章 文件		
12.1	文件概述	(196)
12.2	文件指针	(197)
12.3	文件的打开与关闭	(198)
12.3.1	文件的打开(fopen 函数)	(198)
12.3.2	文件的关闭(fclose 函数)	(199)
12.4	文件的读写	(199)
12.4.1	读/写文件中的一个字符	(200)
12.4.2	读/写一个字符串	(202)
12.4.3	读/写一个数据块	(204)
12.4.4	文件的格式化读/写——fscanf() 和 fprintf() 函数	(205)
12.5	文件的定位	(206)
12.5.1	rewind 函数	(206)
12.5.2	随机读写与 fseek() 函数	(206)
12.5.3	ftell() 函数	(206)
	习题	(207)
附录		
参考文献		

第1章

C语言概述

本章要点:

- ◆ 了解C语言的发展及特点。
- ◆ 了解几种典型的C程序结构。
- ◆ 掌握结构化程序的特点。
- ◆ 了解C程序的上机步骤。

1.1 C语言的发展及特点

1.1.1 C语言的发展简史

C语言是国际上广泛流行的计算机高级语言,既可以用来编写系统软件,也可以用来编写应用软件。

在C语言诞生以前,系统软件主要是用汇编语言编写的。由于汇编语言程序依赖于计算机硬件,其可读性和可移植性都很差;为了提高程序的可读性和可移植性,最好改用高级语言,但一般的高级语言又难以实现对计算机硬件的直接操作(这正是汇编语言的优势),因此,人们盼望有一种兼有汇编语言和高级语言特性的新语言。于是,C语言就在这种情况下应运而生了。

C语言是在B语言的基础上发展起来的。1970年,美国贝尔实验室的Ken Thompson设计出了很简单的而且很接近硬件的B语言,并用B语言编写了第一个UNIX操作系统。但由于B语言过于简单,功能有限,1972~1973年间,贝尔实验室的D. M. Ritchie在B语言的基础上设计出了C语言。C语言既保持了B语言精炼、接近硬件的优点,又克服了它们过于简单、数据无类型的缺点。

最初的C语言只是为描述和实现UNIX操作系统提供一种工作语言而设计的,后来C语言又作了多次改进,但主要还是在贝尔实验室内部使用。直到1975年,UNIX第6版公布后,C语言突出的优点才引起人们的普遍注意。于是,1977年出现了不依赖于具体机器的C语言编译文本《可移植C语言编译程序》,使C语言移植到其他机器时所需做的工作大大简化了,这同时也推动了UNIX操作系统在各种机器上的实现。应该说,C语言和UNIX操作系统是紧密相连的,在发展过程中相辅相成。1978年,C语言先后移植到大、中、小微机上时,已独立于UNIX操作系统了。现在,C语言在全世界的应用非常普及,成为世界上应用

最广泛的几种计算机语言之一。

以 1978 年发表的 UNIX 操作系统第 7 版中的 C 语言编译程序为基础, Brian W. Kernighan 和 Dennis M. Ritchie (合称 K&R) 合著了影响深远的名著《The C Programming Language》。这本书中介绍的 C 语言成为后来广泛使用的 C 语言版本的基础, 它被称为标准 C。1983 年, 美国国家标准化协会 (ANSI) 根据 C 语言问世以来各种版本对 C 的发展和扩充, 制定了新的标准, 称为 ANSI C。ANSI C 比原来的标准 C 有了很大的发展, K&R 在 1988 年修改了他们的经典著作《The C Programming Language》, 按照 ANSI C 标准重新编写了该书。1987 年, ANSI 又公布了新标准——87 ANSI C。1990 年, 国际标准化组织 ISO 接受 87 ANSI C 为 ISO C 的标准, 目前流行的 C 语言编译系统都是以它为基础的。本书基本上以 ANSI C 为基础。目前, 广泛流行的各种版本的 C 语言编译系统基本部分虽然是相同的, 但也有一些不同。在微机上使用的有 Microsoft C、Turbo C、Quick C、Borland C 等, 它们的不同版本又略有差异, 因此, 读者在使用前应先了解所用的计算机系统配置的 C 语言编译系统的特点和规定, 可以查阅相应的手册。

1.1.2 C 语言的特点

C 语言同时具有汇编语言和高级语言的优势。概括起来,C 语言主要有以下特点:

● 语言简洁、紧凑,使用方便、灵活。C 语言共有 32 个关键字(见附录)、9 种控制语句,程序书写形式自由,主要用小写字母表示,压缩了一些不必要的成分。表 1.1 列举了 C 语言与 PASCAL 语言之间的主要区别,请读者注意比较(表 1.1)。

表 1.1

C 语言	PASCAL 语言	含 义
{ }	BEGIN...END	复合语句
if (e) S	IF(e) THEN S	条件语句
int i;	VAR i :INTEGER	定义 i 为整型变量
int a[10]	VAR a;ARRAY[1..10] OF INTEGER	定义 a 为整型一维数组
int f()	FUNCTION f ():INTEGER	定义 f 为返回整型值的函数
int *p	VAR p: ↑ INTEGER	定义 p 为指向整型变量的指针变量
i +=2	i: = i +2	赋值语句,使 i +2→i
i ++, ++i	i: = i +1	自增值语句,使 i +1→i

- 运算符极其丰富。C 语言的运算符很多,共有 34 种运算符(见附录)。C 语言把括号、赋值、强制类型转换等都作为运算符处理,运算类型极其丰富,表达式类型多样化,从而可以在 C 语言中实现在其他高级语言中难以实现的运算。

- 数据结构丰富,具有现代化语言的各种数据结构。C 语言的数据类型有整型、实型、字符型、数组类型、指针类型、结构体类型、共用体类型等,能用来实现各种复杂的数据结构。

- C 语言是良好的结构化设计语言,具有结构化的控制语句(如 if...else 语句、while 语句、do...while 语句、switch 语句),用函数作为程序的模块单位,便于实现程序的模块化。

- 语法限制不太严格,程序设计自由度大。如对数组下标越界不作检查,由程序编写者自己掌握程序的正确性。整型数据和字符型数据之间可以通用。C 语言由于允许程序编写者有较大的自由度,因而放宽了对语法的检查。因此,程序员要仔细检查程序,保证其正

确,不要过分依赖于C语言编译程序去查错。一般来说,使用C语言比使用其他高级语言对程序设计的熟练度有更高要求。

- 生成的目标代码质量高,程序执行效率高。C程序一般只比汇编程序生成的目标代码效率低10%~20%。

- 与汇编语言相比,程序可移植性好。C程序基本上可以不做修改就能用于各种型号的计算机和各种操作系统。

- C语言能直接操纵硬件。C语言能进行位(bit)操作,能实现汇编语言的大部分功能,能直接对硬件进行操作。C语言既具有高级语言的功能,又具有低级语言的许多功能,因此可以用来编写系统软件。有人把C语言称为“高级语言中的低级语言”或“中级语言”,一般仍认为其是高级语言,因为C程序也要经过编译、连接才能得到可执行的目标程序。

上面这些是C语言的一般特点,至于C语言内部的一些特点,我们将在后面的章节中结合具体内容进一步介绍。

总之,C语言对程序员要求较高,程序员使用C语言编写程序会感到限制少,灵活度大,功能强,可以编写出任何类型的程序。现在C语言不仅可以用来编写系统软件,也可以用来编写应用软件。C语言因其具有理想的结构化设计语言,现在普遍用做教学语言,受到大多数人的青睐。

1.2 结构化程序设计的基本概念

一般我们在进行程序设计时,关键的一步是设计算法(即为解决一个问题而采取的确切且有限的步骤)。有了一个好的算法,就可以用任何一种计算机高级语言将算法转换为程序。算法一般用伪代码和流程图来表示。常用的流程图由以下几种基本框组成,如图1.1所示。



图 1.1 基本流程框图

用这些框和流程线组成的流程图来表示算法,形象直观,简单方便。但是这种流程图对于流程线的走向没有任何限制,可以任意转向,在描述复杂的算法时所占篇幅较多,费时费力且不易阅读。随着结构化程序设计的出现,1973年,美国学者 L. Nassi 和 B. Shneiderman 提出了一种新的流程图形式。这种流程图完全去掉了流程线,算法的每一步都用一个矩形框来描述,把一个个矩形框按执行的次序连接起来就是一个完整的算法描述,这种流程图用两位学者名字的第一个英文字母命名,称为 N-S 流程图。

下面简单介绍一下结构化程序的三种基本结构,分别用传统流程图和 N-S 流程图表示。

1.2.1 顺序结构

顺序结构是最简单的一种结构,程序中的语句是按其在程序中的先后顺序逐条执行的,没有分支和转向。一般赋值语句、输入/输出语句可以构成顺序结构。图 1.2 是顺序结构流

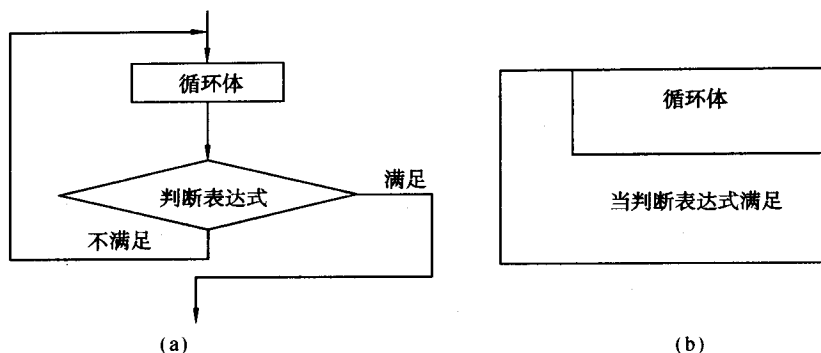


图 1.5 直到型循环

以上是结构化程序设计的三种基本结构,C语言是一种结构化程序设计语言,它直接提供了这三种基本结构的语句。在后续章节中我们将分别介绍这些内容。

1.3 简单的 C 程序介绍

下面先通过几个简单的 C 程序,介绍 C 程序的一些基本构成和格式,使读者对 C 语言程序有一个初步的了解。

例 1.1

```
main()
{
    printf("This is a C program. \n");
}
```

本程序的运行结果如下:

This is a C program.

其中,main 是主函数名,C语言规定每个 C 程序必须有一个主函数,且只能用 main 表示主函数名,其后的一对括号表示函数参数,中间可以是空的,但括号不能省略。main() 是主函数的起始行,一个 C 程序可以包含多个不同名的函数,但主函数只能有一个,一个 C 程序总是从主函数开始执行。

在函数的起始行下面是函数体,函数体用一对 {} 括起来,本例中函数体只有一个输出语句,有关 printf 的具体介绍详见第 3 章。注意,在 printf 语句最后有一个分号。C 语言规定程序中的每一条语句都必须用分号“;”结束,分号是 C 语句的一部分,不是语句之间的间隔符。

例 1.2

```
main()                                /* 求矩形面积 */
{
    float a,b,area;                   /* 定义变量 */
    a = 1.2;b = 3.6;                  /* 给矩形的两条边赋值 */
    area = a * b;                      /* 求出矩形的面积,并存入变量 area 中 */
    printf("a =%f,b = %f,area =%f\n",a,b,area); /* 输出矩形的两条边长和面积 */
}
```

本程序的运行结果如下:

a = 1.200000 , b = 3.600000 , area = 4.320000

本程序的作用是求出矩形面积,/*……*/表示注释部分,可用中文或英文作注释。注释只是为了便于别人能更好地理解程序,对编译和运行不起作用。注释可以加在程序中的任意位置。函数体中第1行是声明部分,定义变量 a、b、area 为实型变量;第2行是两个赋值语句,使 a 和 b 的值分别为 1.2 和 3.6,C 程序中多条语句可写在同一行,每条语句后必须加上分号;第3行是计算 a 和 b 的积并赋给变量 area;第4行是输出语句,其中%f 是输入/输出的格式字符串,有关这部分介绍详见第3章。

例 1.3

```
int max(int x,int y)
/* 定义 max 函数,函数值为整型,形式参数 x、y 为整型 */
{
    return(x > y? x:y);
/* 返回 x 和 y 中较大的数,并通过 max 函数带回调用处 */
}

main()
/* 主函数 */
{
    int num1,num2;          /* 声明部分,定义变量 num1 和 num2 为整型变量 */
    printf("Input the first integer number:");
    scanf("%d", &num1);      /* 输入变量 num1 的值 */
    printf("Input the second integer number:");
    scanf("%d", &num2);      /* 输入变量 num2 的值 */
    printf("max =%d\n", max(num1, num2));
/* 输出 num1 和 num2 中的较大数 */
}
```

本程序的运行结果如下:

Input the first integer number: 6

Input the second integer number: 9

max = 9

在这个程序中,有两个函数:main 函数和被调用的函数 max。max 函数的作用是将 x 和 y 中较大的数,通过 return 语句将值返回给主调函数 main,返回值是通过函数名 max 带回到函数的调用处的。主函数中 scanf() 是输入函数,其作用是输入变量 num1 和 num2 的值,有关 scanf() 的介绍详见第 3 章。本例中用到了函数调用、实际参数和形式参数等概念,读者在此可不必予以深究,在学到后续章节时,问题自然就迎刃而解。

通过上述几个例子,可以得出如下结论:

- C 程序是由函数构成的,函数是 C 语言程序的基本单位。一个 C 源程序至少包含一个 main 函数,也可以包含一个 main 函数和若干个其他函数。main 函数的作用相当于其他高级语言中的主程序,其他函数的作用相当于子程序。编写 C 程序就是编写一个个函数。

- 一个函数一般由两部分组成。

◇ 函数的首部,即函数的第一行。包括函数名、函数类型、函数参数名、参数类型。例如,例1.3中的max函数的首部为

函数类型	函数名	函数参数类型	函数参数名
↓	↓	↓	↓
int	max (	int x,	int y)

一个函数名后面必须有一对括号“()”,参数可以没有,如main()。

◇ 函数体,即函数首部下面{……}内的部分。如果一个函数有多对{},则最外层的一对{}为函数体的范围。函数体内一般有声明部分和执行部分。声明部分主要是对函数中的变量进行定义,函数体中的变量定义语句必须在所有可执行语句之前。执行部分由若干个语句组成,在某些特殊情况下,函数体内可以没有声明部分,也可以声明部分和执行部分都没有。例如:

```
dump()
{ }
```

这是一个空函数,不执行任何操作,但它是一个合法函数。

- 一个C程序总是从main函数开始执行,不论main函数在整个程序中的位置如何。可以把main函数放在程序最前面,也可以放在程序最后,或者放在某些程序之前、某些程序之后。当主函数执行完毕,也即程序执行完毕。习惯上,把main函数放在程序最前面。

- C程序书写格式比较自由,可以一行上写多个语句,也可以把一个语句写在多行上。C程序语句没有行号,对于从哪一列开始书写也无特殊规定。

- 每条语句必须以分号结束,函数的最后一条语句也不例外,分号是C语句的组成部分。

- C语言本身没有输入/输出语句,输入和输出的操作是通过库函数printf和scanf等函数来完成的。

- 程序中可以用/*……*/对C程序语句进行注释,以增加程序的可读性。“/*”和“*/”必须成对使用,且“/”和“*”以及“*”和“/”之间不能有空格,否则会出错。注释的位置可以独占一行,也可以跟在语句的后面,如果一行写不下,可另起一行继续写。

技巧:为避免可能会遗漏必须配对使用的符号,如注释符号、函数体的起止标识符{}和()等,在输入时可连续输入这些起止标识符,然后再在其中进行插入来完成内容的编辑。在起止标识符嵌套以及它们相距较远时,这样做很有必要。

1.4 C程序的上机步骤

前面我们已经介绍了一些用C语言编写的程序。要使计算机能按照人的意志进行工作,必须根据问题的要求,编写出相应的程序。所谓程序,就是一组计算机能识别和执行的指令。每一条指令使计算机执行特定的操作,程序可以用高级语言(如QBASIC、FORTRAN、PASCAL、C等)编写。用高级语言编写的程序称为源程序(Source Program)。从根本上说,计算机只能识别和执行由“0”和“1”构成的二进制指令,而不能识别由高级语言编写的指令。为了使计算机能识别和执行高级语言源程序,必须用一种称为“编译程序”的软件,将源程序编译成二进制形式的“目标程序”,然后将该目标程序和系统的函数库以及其他目标