

高等学校计算机教材

程序设计基础 (C语言) 第二版

郑平安 曾大亮 编
杨有安 崔珂梅
李之棠 审



清华大学出版社

程序设计基础（C 语言）

第二版

郑平安 曾大亮
杨有安 崔珂梅
李之棠 审
编

清华大学出版社

北 京

内 容 简 介

本书全面、系统地介绍了 C 语言程序设计的基本知识。包括 C 语言的基本概念、变量、运算符、表达式、顺序结构、分支结构、循环结构、数组、函数、指针、结构体、联合体和枚举类型、接口与函数库、文件。同时也介绍了一些与程序设计有关的知识,包括算法设计与分析、抽象数据类型、程序抽象等。

本书结构清晰,语言通俗易懂,内容由浅入深、循序渐进,实例丰富,习题具有代表性。全书贯彻传授知识、培养能力、提高素质的教学理念。

本书可以作为高等院校非计算机专业 C 语言程序设计的教材,也可以作为初次学习 C 语言程序设计的读者、准备计算机二级考试者和计算机工程技术人员的参考书。

版权所有,翻印必究。举报电话:010-62782989 13501256678 13801310933

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

本书防伪标签采用特殊防伪技术,用户可通过在图案表面涂抹清水,图案消失,水干后图案复现;或将表面膜揭下,放在白纸上用彩笔涂抹,图案在白纸上再现的方法识别真伪。

图书在版编目(CIP)数据

程序设计基础(C语言)/郑平安编 —2版. —北京:清华大学出版社,2006.1

ISBN 7-302-12398-5

I. 程… II. 郑… 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(2006)第003503号

出 版 者:清华大学出版社 地 址:北京清华大学学研大厦
http://www.tup.com.cn 邮 编:100084
社 总 机:010-62770175 客 户 服 务:010-62776969

责任编辑:马 丽

封面设计:范华明

版式设计:杨 洋

印 刷 者:清华大学印刷厂

装 订 者:北京市密云县京文制本装订厂

发 行 者:新华书店总店北京发行所

开 本:185×260 印张:26.5 字数:587千字

版 次:2006年1月第2版 2006年1月第1次印刷

书 号:ISBN 7-302-12398-5/TP·7942

印 数:1~8000

定 价:33.00元

前 言

C 语言是目前世界上最流行、使用最广泛的高级程序设计语言。它不但具有丰富的数据类型与运算符、灵活的控制结构、简洁而高效的表达式、清晰的程序结构和良好的可移植性等优点,而且还具有直接对计算机硬件编程的强大功能;它既具有高级语言的优点,又具有低级语言的许多特点。C 语言既适合于开发系统软件,又适合于开发应用软件,深受程序员的欢迎。

本书根据教育部高等学校计算机教学指导委员会 2003 年公布的《关于进一步加强高等学校计算机基础教学的几点意见》的精神,结合编者多年从事程序设计教学的经验,在第一版的基础上,增加了算法设计与分析、抽象数据类型、程序抽象等与程序设计密切相关的内容,以便进一步夯实学生程序设计的理论基础,培养程序设计的能力,养成良好的程序设计风格。

在编写过程中参考了大量同类教材并吸收了这些教材的优点,同时又保持了自己的特色。本书的主要特点是:文章的叙述通俗易懂,内容的编排由浅入深、循序渐进;通过精心设计的例题,着重介绍 C 程序设计的基本方法与基本技巧;通过精心选择的习题,训练程序设计的技能。全书体现“程序设计=算法+数据结构”的程序设计课程教学内涵,贯彻传授知识、培养能力、提高素质的教学理念。教材的内容不但有 C 语言的最基本部分,还有读者感兴趣的屏幕控制与绘图程序设计(依据 Turbo C 编译器),以便开阔学生的视野,培养程序设计的兴趣。

本书由华中科技大学网络与计算中心长期从事程序设计课程教学的教师编写。全书分为 12 章,其中第 1、8、9、10、12 章和附录由郑平安编写,第 2、3、4 章由曾大亮编写,第 5、6 章由杨有安编写,第 7、11 章由崔珂梅编写。

在本书的编写过程中得到网络与计算中心主任李之棠教授的关心、帮助和支持,并且审阅了书稿,在此编者表示衷心的感谢。

由于编者水平有限,加之时间仓促,书中难免有不足之处,敬请读者批评指正。

编 者

2005 年 12 月于华中科技大学

目 录

第 1 章 C 语言概述	1
1.1 C 语言的发展和特点	1
1.1.1 C 语言的发展过程	1
1.1.2 C 语言的主要特点	2
1.2 C 程序的结构	3
1.3 Turbo C 上机步骤	6
1.3.1 Turbo C 2.0 文件简介	6
1.3.2 Turbo C 启动	6
1.3.3 源程序的输入、编译、连接和运行	7
小结	8
习题	8
第 2 章 基本数据类型和运算符	9
2.1 关键字、标识符和保留标识符	9
2.2 基本数据类型	11
2.2.1 常量和变量的概念	11
2.2.2 整型常量和实型常量	14
2.2.3 整型变量	15
2.2.4 实型变量	18
2.2.5 字符常量和字符串常量	20
2.2.6 字符变量	23
2.2.7 用 char 定义小整数	25
2.2.8 符号常量	26
2.3 运算符和表达式	28
2.3.1 表达式的概念	28
2.3.2 算术运算符	30
2.3.3 标准系统库函数调用	31
2.3.4 数据类型的转换	33
2.3.5 赋值运算符	36
2.3.6 增量运算符	39
2.3.7 副作用和顺序点	41
2.3.8 关系运算符	42

2.3.9 逻辑运算符	43
2.3.10 条件运算符	47
2.3.11 逗号运算符和逗号表达式	48
2.3.12 运算符优先级和结合方向	48
小结	49
习题	50
第 3 章 顺序语句和选择语句	55
3.1 程序设计概述	55
3.2 scanf() 函数和字符输入、输出函数调用	57
3.2.1 数据输入的概念	57
3.2.2 scanf() 函数的调用	58
3.2.3 字符输入函数	60
3.2.4 字符输出函数	61
3.3 表达式语句	62
3.4 复合语句	62
3.5 if 条件语句	63
3.5.1 if 结构	63
3.5.2 if-else 结构	67
3.5.3 if-else-if 结构	69
3.5.4 条件语句的嵌套	73
3.6 switch 语句	77
3.7 程序设计举例	81
小结	85
习题	85
第 4 章 循环语句和转移语句	88
4.1 循环的概念	88
4.2 for 循环	89
4.3 while 循环	98
4.4 do-while 循环	102
4.5 break 语句	104
4.6 continue 语句	104
4.7 多重循环	106
4.8 goto 语句	114
小结	115
习题	115

第 5 章 数组	119
5.1 一维数组	119
5.1.1 一维数组的定义	119
5.1.2 一维数组元素的引用	120
5.1.3 一维数组元素的初始化	122
5.2 二维数组	124
5.2.1 二维数组的定义	124
5.2.2 二维数组的引用	125
5.2.3 二维数组元素的初始化	126
5.3 字符型数组	129
5.3.1 字符数组的定义	129
5.3.2 字符数组的引用	130
5.3.3 字符数组的初始化	131
5.3.4 字符串及其结束标志	132
5.3.5 字符数组的输入/输出	134
5.3.6 常用的字符串处理函数	136
5.4 使用数组的程序设计方法	140
5.4.1 排序	140
5.4.2 查找	143
小结	149
习题	150
第 6 章 函数和模块设计	155
6.1 结构化程序设计	155
6.1.1 结构化程序设计的基本概念	156
6.1.2 结构化程序设计的基本特征	156
6.2 函数的定义和调用	157
6.2.1 函数的定义	158
6.2.2 函数的调用	159
6.2.3 函数的返回值	162
6.2.4 函数参数及函数间的数据传递	165
6.3 函数的嵌套调用和递归调用	173
6.3.1 函数的嵌套调用	173
6.3.2 函数的递归调用	176
6.4 作用域和存储类型	178
6.5 内部函数和外部函数	187
6.5.1 内部函数	187

6.5.2 外部函数	188
6.6 模块化程序设计	189
6.6.1 模块化程序设计方法的指导思想	189
6.6.2 模块分解的原则	190
6.7 应用举例	191
小结	197
习题	197
第 7 章 指针	200
7.1 指针的基本概念	200
7.2 指针变量的定义和初始化	201
7.2.1 指针变量的定义	201
7.2.2 指针变量的初始化	202
7.3 指针运算符	202
7.3.1 取地址运算符 &	202
7.3.2 指针运算符 *	203
7.4 指针变量的运算	203
7.4.1 赋值运算	203
7.4.2 加减算术运算	205
7.4.3 两指针变量进行关系运算	207
7.4.4 指针变量还可以与 0 比较	207
7.5 指针变量作为函数参数	210
7.6 指针和数组的关系	213
7.6.1 指向数组的指针变量	213
7.6.2 通过指针引用数组元素	214
7.6.3 通过指针引用数组元素时应注意几个问题	216
7.6.4 数组名作函数参数	217
7.7 指向字符串的指针变量	219
7.7.1 字符串的表示形式	219
7.7.2 使用字符串指针变量与字符数组的区别	220
7.7.3 字符串指针作为函数参数	221
7.8 指向多维数组的指针变量	223
7.8.1 多维数组的地址	223
7.8.2 指向数组的指针变量——数组指针变量	226
7.9 指针数组和多级指针	228
7.9.1 指针数组的概念	228
7.9.2 指针数组的应用	229

7.9.3 多级指针（指向指针的指针）	232
7.10 指向函数的指针	234
7.10.1 函数指针的概念	234
7.10.2 函数指针的应用	236
7.11 指针型函数	240
7.12 命令行参数	242
7.12.1 命令行参数的概念	242
7.12.2 打印命令行参数	243
7.12.3 命令行参数使用形式	244
7.12.4 main()函数的返回	244
7.12.5 从命令行提取文件名	245
小结	247
习题	250
第8章 结构体与联合体	257
8.1 结构体	257
8.1.1 结构的定义和结构变量的引用	257
8.1.2 结构类型变量的说明	258
8.1.3 结构变量成员的引用	260
8.1.4 结构变量的赋值	260
8.1.5 结构变量的初始化	261
8.1.6 结构数组的定义	261
8.1.7 结构指针变量的说明和使用	263
8.1.8 结构数据的动态存储分配	267
8.1.9 链表及其基本操作	268
8.2 联合体	274
8.2.1 联合的定义	275
8.2.2 联合变量的说明	275
8.3 其他自定义数据类型	277
8.3.1 枚举类型	277
8.3.2 类型定义符 typedef	281
小结	282
习题	282
第9章 接口和函数库	285
9.1 预处理命令	285
9.1.1 概述	285
9.1.2 宏定义	285

9.1.3 文件包含	292
9.1.4 条件编译	293
9.2 C 的标准库	295
9.2.1 格式输出函数 (printf) 中的格式说明	295
9.2.2 格式输入函数 (scanf) 中的格式说明	297
9.3 图形库	297
9.3.1 图形模式初始化函数	298
9.3.2 基本图形函数	298
9.4 屏幕控制函数库	303
9.5 自定义接口与接口设计的原则	307
小结	312
习题	312
第 10 章 文件	313
10.1 文件的概念	313
10.2 文件类型指针	314
10.3 文件的基本操作	315
10.3.1 文件的打开函数 fopen	315
10.3.2 文件关闭函数 fclose	317
10.3.3 字符读写函数 fgetc 和 fputc	317
10.3.4 字符串读写函数 fgets 和 fputs	319
10.3.5 数据块读写函数 fread 和 fwrite	320
10.3.6 格式化读写函数 fscanf 和 fprintf	322
10.3.7 文件的随机读写	323
10.3.8 文件检测函数	326
10.4 程序设计举例	326
小结	329
习题	329
第 11 章 数据结构和数据抽象	331
11.1 数据抽象	331
11.1.1 抽象数据类型的基本概念	331
11.1.2 抽象数据类型对程序设计的意义	332
11.1.3 数据结构、数据类型和抽象数据类型	332
11.2 线性表	333
11.2.1 线性表的定义	333
11.2.2 线性表的基本操作	334
11.2.3 线性表的顺序存储	335

11.2.4	顺序表上基本运算的实现	336
11.3	堆栈	339
11.3.1	抽象堆栈的定义及基本操作	340
11.3.2	抽象栈的定义	340
11.3.3	顺序栈的基本运算的实现	340
11.4	队列	342
11.4.1	队列的定义	342
11.4.2	队列的存储结构及其相关算法	343
	小结	347
	习题	348
第12章	算法和程序抽象	349
12.1	算法的概念	349
12.2	算法的类型与结构	350
12.2.1	数值算法和非数值算法	350
12.2.2	算法的基本结构	350
12.3	算法的描述方法	351
12.3.1	流程图	351
12.3.2	N-S图	351
12.3.3	伪代码	352
12.4	算法设计与分析	353
12.4.1	算法的设计与实现	353
12.4.2	算法分析与算法复杂度	355
12.4.3	常用算法设计与分析	361
12.5	软件设计的基本原则	363
12.5.1	软件评判标准	363
12.5.2	内聚性	364
12.5.3	耦合度	365
12.6	数据封装和信息隐藏	366
12.6.1	数据封装	366
12.6.2	信息隐藏	368
12.7	算法抽象	369
12.7.1	函数指针类型定义	370
12.7.2	函数指针类型的使用	370
12.7.3	算法抽象举例	371
12.8	程序设计方法	374
12.8.1	可行性研究与项目开发计划	374

12.8.2 软件需求分析	375
12.8.3 软件概要设计	376
12.8.4 软件详细设计	377
12.8.5 程序编码	378
12.8.6 软件测试	378
小结	379
习题	379
附录 1 常用字符与 ASCII 代码对照表	381
附录 2 C 语言常用语法提要	382
附录 3 Turbo C 的集成开发环境的使用	387
附录 4 Turbo C 编译错误信息	393
附录 5 C 库函数	399
参考文献	407

第 1 章 C 语言概述

C 语言是国际上广泛流行的、很有发展前途的高级程序设计语言，它既可以用来编写系统软件，又可以用来编写应用软件。C 语言也是大学生学习程序设计选用最多的语言。本章主要介绍 C 语言的发展历史、C 语言的特点、C 程序的结构和 C 程序的上机步骤。通过本章的学习，读者应重点掌握 C 语言的特点、C 程序的结构和上机运行 C 程序的方法。

1.1 C 语言的发展和特点

自从计算机诞生以来，为了更好地进行软件的设计，各种高级程序设计语言也在不断地发展、进步和完善。C 语言就是其中最优秀的程序设计语言之一。

1.1.1 C 语言的发展过程

C 语言是目前世界上最流行、使用最广泛的高级程序设计语言。在设计操作系统等系统软件和需要对硬件进行操作时，使用 C 语言编程明显优于其他高级语言，许多大型应用软件和系统软件都是用 C 语言编写的。

C 语言的起源可以追溯到 ALGOL 60。1963 年英国的剑桥大学在 ALGOL 60 的基础上推出了 CPL 语言，但是 CPL 语言难以实现。1967 年英国剑桥大学的 Martin Richards 对 CPL 语言作了简化，推出了 BCPL 语言。1970 年美国贝尔实验室的 Ken Thompson 以 BCPL 语言为基础，又作了进一步的简化，设计出了很简单且接近硬件的 B 语言，并用 B 语言写了第一个 UNIX 操作系统，在 DEC PDP-7 型计算机上实现。1971 年在 DEC PDP-11 上实现了 B 语言。1972 年由美国的 Dennis M. Ritchie 在 B 语言的基础上设计出了 C 语言，并首次在 UNIX 操作系统的 DEC PDP-11 计算机上使用。

为了推广 UNIX 操作系统，1977 年 Dennis M. Ritchie 发表了不依赖于具体机器系统的 C 语言编译文本《可移植的 C 语言编译程序》。1978 年 Brian W. Kernighan 和 Dennis M. Ritchie 出版了名著《The C Programming Language》，从而使 C 语言成为目前世界上流行最广泛的高级程序设计语言。

随着微型计算机的日益普及，出现了许多 C 语言版本。由于没有统一的标准，使得这些 C 语言之间出现了一些不一致的地方。为了改变这种情况，1983 年美国国家标准研究所（ANSI）为 C 语言制定了第一个 ANSI 标准，称为 ANSI C。1987 年美国国家标准研究所又公布了新的 C 语言标准，称为 87 ANSI C。这个标准在 1989 年被国际标准化（ISO）组

织采用, 被称为 ANSI/ISO Standard C (即 C89)。Brian W.Kernighian 和 Dennis M.Ritchie 根据这个标准, 重写了他们的经典著作, 并发表了《The C Programming Language, Second Edition》。

1995 年又为 C 语言增加了一些新的函数, 使之具有 C++ 的一些特征, 使 C89 成为 C++ 的子集。1999 年推出的 C99 在基本保留 C 语言特征的基础上, 增加了一系列面向对象的新特征。C 语言也从面向过程的语言发展成为面向对象的语言。

C 语言是 C++ 的基础, 因此, 掌握了 C 语言, 再进一步学习 C++ 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。本课程是按照 ANSI/ISO Standard C (即 C89) 来讲授 C 语言的。

目前最流行的 C 语言有 Microsoft C 或称 MS C、Turbo C 和 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。本课程使用 Turbo C 作为上机环境。

1.1.2 C 语言的主要特点

C 语言发展迅速, 而且成为最受欢迎的语言之一, 主要因为它具有强大的功能。许多著名的系统软件, 如 dBASE III PLUS、dBASE IV 都是由 C 语言编写的。用 C 语言加上一些汇编语言子程序, 就更能显示 C 语言的优势, 像 PC-DOS、WordSTAR 等就是用这种方法编写的。归纳起来 C 语言具有下列特点:

1. C 语言简洁、紧凑、方便、灵活

C 语言共有 32 个关键字, 9 种控制语句, 程序书写自由, 主要用小写字母表示, 压缩了一些不必要的成分。

2. 运算符丰富

C 语言的运算符包括的范围很广泛, 共有 34 个运算符。C 语言把括号、赋值、强制类型转换等都作为运算符处理。从而使 C 语言的运算类型极其丰富、表达式类型多样化, 灵活使用各种运算符可以实现在其他高级语言中难以实现的运算。

3. 数据结构丰富

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、联合体类型等, 能用来实现各种复杂的数据类型的运算。并引入了指针概念, 使程序效率更高。另外 Turbo C 语言具有强大的图形功能, 支持多种显示器和驱动器。且计算功能、逻辑判断功能强大。

4. C 语言是结构化语言

结构化语言的显著特点是代码及数据的分隔化, 即程序的各个部分除了必要的信息交流外彼此独立。这种结构化方式可使程序层次清晰, 便于使用、维护以及调试。C 语言是

以函数形式提供给用户的，这些函数可方便地调用，并具有多种循环语句、条件语句来控制程序流程，从而使程序完全结构化。

5. C 语法限制不太严格，程序设计自由度大

一般的高级语言语法检查比较严，能够检查出几乎所有的语法错误。而 C 语言放宽了语法检查，允许程序编写者有较大的自由度。这是 C 语言的优点，也是 C 语言的缺点。限制严格就失去了灵活性，而强调灵活必然要放松限制。在程序设计中，程序员不要过分地依赖编译器的语法检查。因此，对于初学者，编写一个正确的 C 语言程序比编写一个其他高级语言程序更难些。

6. C 语言允许直接访问物理地址，可以直接对硬件进行操作

C 语言既具有高级语言的特点，又具有低级语言的许多功能，能够像汇编语言一样对位、字节和地址进行操作，而这三者是计算机最基本的工作单元，可以用来写系统软件。因此，有人把 C 语言称为“中级语言”。

7. C 语言程序生成的代码质量高

程序执行效率高，一般只比汇编程序生成的目标代码效率低 10%~20%。

8. C 语言适用范围大，可移植性好

C 语言有一个突出的优点就是适合于多种操作系统，如 DOS、UNIX，也适用于多种机型。

1.2 C 程序的结构

用 C 语言编写的程序称为 C 语言源程序，简称为 C 程序。为了说明 C 语言源程序结构的特点，先看以下几个程序。这几个程序由简到难，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 程序的基本部分和书写格式。

【例 1-1】输出一行信息的 C 程序。

```
void 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 程序都必须有主函数，且只能有一个主函数。

printf 是输出函数，它的功能是把要输出的内容送到显示器去输出。printf 函数是一个由系统定义的标准函数，可在程序中直接调用。

执行这个程序，会在显示器上输出：

Hello, world!

【例 1-2】计算三角函数的 C 程序。

```
#include<math.h>                /*#include 称为文件包含命令，扩展名为.h 的文件称为头文件*/
#include<stdio.h>
void main()
{
    double x,s;                  /*定义两个实数变量，以被后面程序使用*/
    printf("Input a angle:");    /*显示提示信息*/
    scanf("%lf",&x);             /*从键盘获得一个实数 x*/
    s=sin(x*3.14159/180.0);      /*求 x 的正弦，并把它赋给变量 s*/
    printf("sine of %lf is %lf\n",x,s); /*显示程序运算结果*/
    printf("Strike any key to continue!\n");
    getch();                     /*按任何键回到集成开发环境*/
}
```

程序的功能是从键盘输入一个数 x ，求 x 的正弦值，然后输出结果。在 `main()` 之前的两行称为预处理命令（详见后文）。预处理命令还有其他几种，这里的 `#include` 称为文件包含命令，其意义是把尖括号 `<>` 或引号 `" "` 内指定的文件包含到本程序来，成为本程序的一部分。被包含的文件通常是由系统提供的，其扩展名为 `.h`。因此也称为头文件或首部文件。C 语言的头文件中包括了各个标准库函数的函数原型。因此，凡是在程序中调用一个库函数时，都必须包含该函数原型所在的头文件。在本例中，使用了 3 个库函数：输入函数 `scanf`、正弦函数 `sin` 和输出函数 `printf`。`sin` 函数是数学函数，其头文件为 `math.h` 文件，因此在程序的主函数前用 `#include` 命令包含了 `math.h`。`scanf` 和 `printf` 是标准输入/输出函数，其头文件为 `stdio.h`，在主函数前也用 `#include` 命令包含了 `stdio.h` 文件。在程序的每行后用 `/*` 和 `*/` 括起来的内容为注释部分，程序不执行注释部分。

需要说明的是，C 语言规定对 `scanf` 和 `printf` 这两个函数可以省去对其头文件的包含命令。所以在本例中也可以删去第 2 行的包含命令 `#include<stdio.h>`。

在例题中的主函数体中又分为两部分，一部分为说明部分，另一部分为执行部分。说明是指变量的类型说明。例题 1-1 中未使用任何变量，因此无说明部分。C 语言规定，源程序中所有用到的变量都必须先说明后使用，否则将会出错。这一点是编译型高级程序设计语言的一个特点。说明部分是 C 程序结构中很重要的组成部分。本例中使用了两个变量 x 和 s ，用来表示输入的自变量和正弦函数值。由于正弦函数要求这两个量必须是双精度浮点型，故用类型说明符 `double` 来说明这两个变量。说明部分后的 4 行为执行部分或称为执行语句部分，用以完成程序的功能。执行部分的第 1 行是输出语句，调用 `printf` 函数在显示器上输出提示字符串，请操作人员输入自变量 x 的值。第 2 行调用 `scanf` 函数，接受键盘上输入的实数并存入变量 x 中。第 3 行是调用正弦函数并把函数值送到变量 s 中。第 4 行是用 `printf` 函数输出变量 s 的值，即 x 的正弦值。程序结束。

运行本程序时，首先在显示器屏幕上给出提示串 `Input a angle:` 这是由执行部分的第 1

行完成的。用户在提示串下输入某一数，如 60，按下回车键，接着在屏幕上给出计算结果。

【例 1-3】包含自定义函数的 C 程序。

```
#include<stdio.h>
void main()                                /*主函数*/
{
    int x,y,z;                             /*变量说明*/
    int max(int,int);                      /*函数原型声明*/
    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("maxmum=%d\n",z);              /*输出*/
    printf("Strike any key to continue!\n");
    getch();                              /*按任何键回到集成开发环境*/
}
int max(int a,int b)                       /*定义 max 函数*/
{
    if(a>b)return a;
    else return b;                        /*把结果返回主调函数*/
}
```

程序的功能是由用户输入两个整数，程序执行后输出其中较大的数。本程序由两个函数组成，即主函数和 max 函数，两函数之间是并列关系，可从主函数中调用其他函数。max 函数的功能是比较两个数，然后把较大的数返回给主函数。max 函数是一个用户自定义函数。因此在主函数中要给出函数原型。可见，在程序的说明部分中，不仅有变量说明，还可以有函数原型说明。关于函数的详细内容将在第 6 章介绍。

以上程序的执行过程是，首先在屏幕上显示提示串，请用户输入两个数，然后由 scanf 函数接收这两个数送入变量 x、y 中，接着调用 max 函数，并把 x、y 的值传送给 max 函数的参数 a 和 b。在 max 函数中比较 a、b 的大小，把大数返回给主函数的变量 z，最后在屏幕上输出 z 的值。

通过以上 3 个例子，可以概括出 C 语言源程序的结构特点：

- (1) 一个 C 语言源程序可以由一个或多个源文件组成。
- (2) 每个源文件可由一个或多个函数组成。
- (3) 一个源程序不论由多少个文件组成，都有一个且只能有一个名字为 main 的函数（主函数）。
- (4) 源程序中可以有预处理命令（#include 命令仅为其中的一种），预处理命令通常应放在源文件或源程序的最前面。
- (5) 每一个说明、每一个语句都必须以分号结尾。但预处理命令、函数头和最后一个花括号“}”之后不能加分号。
- (6) 标识符、关键字之间必须至少加一个空格以示分隔。若已有明显的分隔符，可不再加空格。