C语言程序设计教程

史济建 何钦铭 陈小平 占宇飞 编著

浙江大学出版社

C 语言程序设计教程

史济建　何钦铭
陈小平　占宇飞　编著

浙江大学出版社

内容提要

本书是编著者根据多年从事C语言教学和C语言编程的经验，按照C语言自身的特点精心编写而成的。全书共分九章，分别介绍了绪论、C语言的基本数据类型和表达式、语句及控制流、函数和程序结构、数组和指针、结构和联合、类型定义与位段、文件操作，最后还针对C语言的特点，较详细地介绍了C语言程序设计方法和程序设计中应注意的问题。书中还有大量通过上机验证的实例，以进一步说明C语言的基本概念和基本的程序设计方法。

本书概念清晰、层次分明、深入浅出、易于掌握，是一本C语言初学者的好教材；同时，本书重点突出、内容丰富、兼顾实用，也可作为C语言程序设计者的工具书。

图书在版编目（CIP）数据

C语言程序设计教程 / 史济建等编著. —杭州：浙江大学出版社，1998.8(2001重印)
ISBN 7-308-02046-0

Ⅰ.C... Ⅱ.史... 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（2001）第060107号

责任编辑 孙海荣
出版发行 浙江大学出版社
（杭州浙大路38号 邮政编码310027）
（E-mail:zupress@mail.hz.zj.cn）
（网址：http://www.zjupress.com）
排　　版 浙江大学出版社电脑排版中心
印　　刷 浙江大学印刷厂
开　　本 787mm×1092mm 1/16
印　　张 14.5
字　　数 371千
版 印 次 1998年8月第1版 2004年1月第5次印刷
印　　数 13001—16000
书　　号 ISBN 7-308-02046-0/TP·174
定　　价 17.00元

前　言

C语言以其丰富灵活的控制和数据结构、简洁而高效的语句表达、清晰的程序结构、良好的移植性、较小的时空开销等特点，已被广泛地应用于系统软件和应用软件的开发中。尽管目前许许多多的C语言版本有各自的特点，但它们几乎都包括了ANSI C所规定的所有内容，从而使C语言程序设计可遵循统一的标准，这也更进一步增强了C语言的生命力。

由于C语言在程序设计中的广泛用途，不仅计算机专业的工作者喜欢使用，而且非计算机专业的工作者也越来越重视把C语言作为其自己应用领域中的主要程序设计语言。毫不夸张地说，是否掌握C语言可作为衡量程序员水平的一个尺度。但学习C语言与学习其他计算机语言相比较显得更困难些，特别是对于非计算机专业的初学者。因此，编写一本概念清晰、层次分明、深入浅出、易于掌握的C语言教程显得越来越迫切，本书的编写正是朝着这方向努力的。

多年来，我们一直在浙江大学从事C语言的教学工作，对学生学习C语言时的一些难点问题较为明了；同时，我们也有多年用C语言编程的实践经验，清楚C语言自身的特点、C语言编程技巧以及用C语言编程时应注意的问题。所以在编写本书过程中，尽量体现我们在教学和编程实践中的这些经验，使初学者少走些弯路。

学习计算机语言一个重要的途径是不断地上机编程实践。为此，我们在讲述C语言基本概念时，用大量的程序作为实例说明，帮助读者更好地理解这些基本概念。这些程序都是在DOS操作系统下用Turbo C 2.0运行通过的，所用到的C语言规范一般不会超出ANSI C的标准，所以大部分程序可不加修改地在其他C语言版本下编译与运行。我们特别建议读者在学习C语言时多进行上机编程练习，这样不仅可以加深对一些基本概念的理解，而且还可以提高C语言程序设计技巧，并逐渐克服C语言编程中易犯的一些错误。

同时，为帮助读者深入理解C语言的基本概念并进一步掌握其基本的程序设计方法，本书在每章最后配有多种形式的习题供读者练习。建议读者对其中的大部分习题上机编程、验证。

本书主要是为计算机基础教学和科研服务的，错误和不当之处，恳请读者和专家指正，以便有机会进行修改和提高。

作　者

1997年10月于浙大求是园

目　录

第一章　绪论 …… 1

1.1　程序与程序设计语言 …… 1
1.2　C语言的特点 …… 2
1.3　C语言程序概貌 …… 3
1.3.1　C语言程序基本结构 …… 3
1.3.2　C语言的基本输入输出函数 …… 7
1.4　C语言程序的编译与运行 …… 9
习题一 …… 10

第二章　基本数据类型和表达式 …… 12

2.1　标识符和保留字 …… 13
2.1.1　标识符的定义 …… 13
2.1.2　保留字 …… 13
2.1.3　特定字 …… 13
2.1.4　自定义标识符 …… 14
2.2　常量和变量 …… 14
2.2.1　常量与符号常量 …… 14
2.2.2　变量 …… 15
2.3　整数类型 …… 16
2.3.1　整数 …… 16
2.3.2　整型变量 …… 16
2.4　实数类型 …… 18
2.4.1　实型常量 …… 18
2.4.2　实型变量 …… 19
2.5　字符类型 …… 19
2.5.1　字符常量 …… 19
2.5.2　转义序列 …… 20
2.5.3　字符变量 …… 20
2.6　不同数据类型间的混合运算 …… 21
2.7　表达式和求值规则 …… 21
2.7.1　算术表达式 …… 21
2.7.2　赋值表达式 …… 25
2.7.3　关系表达式 …… 27
2.7.4　逻辑表达式 …… 28

2.7.5 条件表达式 …… 30
2.7.6 逗号表达式 …… 31
2.8 运算符及其结合性与优先级 …… 32
2.8.1 位运算符 …… 32
2.8.2 长度运算符 …… 34
2.8.3 特殊运算符 …… 34
2.8.4 运算符的优先级与结合性 …… 34
习题二 …… 36

第三章 语句及控制流 …… 39

3.1 基本语句 …… 39
3.1.1 表达式语句 …… 39
3.1.2 空语句 …… 40
3.1.3 复合语句 …… 40
3.2 选择控制语句 …… 41
3.2.1 if 语句 …… 41
3.2.2 switch 语句 …… 48
3.3 重复控制语句 …… 53
3.3.1 for 语句 …… 53
3.3.2 while 语句 …… 58
3.3.3 do－while 语句 …… 61
3.3.4 多重循环的嵌套 …… 63
3.4 简单控制语句 …… 64
3.4.1 break 语句 …… 64
3.4.2 continue 语句 …… 66
3.4.3 goto 语句 …… 66
3.4.4 return 语句 …… 68
习题三 …… 68

第四章 函数和程序结构 …… 72

4.1 函数的定义 …… 72
4.1.1 函数定义的格式 …… 73
4.1.2 形式参数和实在参数 …… 74
4.1.3 函数的返回值 …… 76
4.2 函数的一般调用格式 …… 78
4.3 函数的嵌套调用 …… 80
4.4 函数的递归调用 …… 82
4.5 C语言程序结构 …… 84
4.6 局部变量和全局变量 …… 89
4.6.1 局部变量 …… 89

4.6.2　全局变量…… 90
4.7　变量存储类型及生存期…… 92
4.7.1　变量存储类别…… 92
4.7.2　变量作用域规则…… 95
4.7.3　各种存储类别的生存期…… 96
4.7.4　变量初始化…… 96
4.8　函数的作用域…… 97
4.8.1　内部函数…… 97
4.8.2　外部函数…… 97
4.9　C语言预处理器…… 98
4.9.1　宏定义…… 98
4.9.2　文件包含处理 …… 100
4.9.3　条件编译 …… 101
习题四…… 103

第五章　数组和指针…… 107

5.1　数组 …… 107
5.1.1　一维数组的定义和引用 …… 107
5.1.2　多维数组的定义和引用 …… 112
5.1.3　字符数组 …… 114
5.2　指针 …… 115
5.2.1　指针的概念 …… 115
5.2.2　指针的定义 …… 116
5.2.3　指针与地址 …… 116
5.2.4　指针变量的初始化 …… 118
5.3　指针和数组 …… 119
5.3.1　指针、数组、地址间的关系 …… 119
5.3.2　元素为指针的数组 …… 120
5.3.3　指向数组的指针 …… 120
5.3.4　多级指针 …… 122
5.4　指针和字符串 …… 123
5.4.1　字符串的指针表示 …… 123
5.4.2　字符指针与字符数组 …… 124
5.5　指针和函数 …… 125
5.5.1　指针作为函数的返回值 …… 125
5.5.2　指针作为函数的参数 …… 126
5.5.3　指向函数的指针 …… 127
5.5.4　main 函数的命令行参数 …… 129
5.6　复合说明 …… 131
习题五…… 131

第六章 结构和联合 …… 136

6.1 结构 …… 136
6.1.1 结构的定义 …… 136
6.1.2 结构变量的定义与初始化 …… 137
6.1.3 结构成员的引用 …… 139
6.2 结构数组 …… 140
6.3 结构指针 …… 143
6.4 结构的嵌套定义与引用 …… 145
6.4.1 结构嵌套定义的基本方法 …… 145
6.4.2 单向链表的建立及操作 …… 147
6.5 联合 …… 153
6.5.1 联合的定义 …… 153
6.5.2 联合成员的引用 …… 154
6.6.3 结构与联合的异同 …… 155
习题六 …… 156

第七章 类型定义与位段 …… 160

7.1 枚举类型 …… 160
7.2 自定义类型 …… 164
7.3 位段 …… 167
7.4 不同数据类型间的相互转换 …… 170
7.4.1 类型的自动转换 …… 171
7.4.2 类型的强制转换 …… 171
习题七 …… 172

第八章 文件操作 …… 174

8.1 文件的基本概念 …… 174
8.2 标准文件的输入/输出 …… 175
8.2.1 字符的输入/输出 …… 175
8.2.2 格式化输入/输出 …… 177
8.3 缓冲文件系统 …… 182
8.3.1 文件类型指针 …… 182
8.3.2 文件的打开与关闭 …… 183
8.3.3 文件的读写操作 …… 185
8.3.4 文件的定位操作 …… 188
8.3.5 文件状态判断 …… 188
8.3.6 缓冲文件系统的实例 …… 189
8.4 非缓冲文件系统 …… 193
8.4.1 文件的建立、打开、关闭 …… 193

8.4.2　文件的读写操作 …… 194
8.4.3　文件的定位操作 …… 195
习题八 …… 196

第九章　C语言程序设计方法 …… 199

9.1　C语言程序设计风格 …… 199
9.1.1　标识符的命名 …… 199
9.1.2　注释 …… 200
9.1.3　程序书写格式 …… 200
9.2　结构化程序设计 …… 201
9.2.1　自顶向下的程序设计 …… 201
9.2.2　领域问题的模块化 …… 203
9.3　C语言编程中应注意的问题 …… 203
9.3.1　易混淆的运算符 …… 203
9.3.2　正确的数据类型操作 …… 204
9.3.3　正确的语句运用 …… 207

附　录 …… 209

附录1　ASCII码集 …… 209
附录2　C语言中的关键字 …… 211
附录3　C语言常用语法 …… 211
附录4　C语言常用库函数 …… 215

第一章 绪 论

本章要点

* 什么是程序？一般程序设计语言包含哪些内容？
* C语言有哪些特点？
* C语言程序的基本框架如何？
* 形成一个可运行的C语言程序需要经过哪些步骤？

1.1 程序与程序设计语言

计算机程序是人们为解决某种问题用计算机可以识别的代码编排的一系列加工步骤。计算机能严格按照这些步骤去做，包括计算机对数据的处理，都是人预先安排好的。程序的执行过程实际上是对数据的处理过程。程序设计语言是人用来编写程序的手段，是人与计算机交流的语言。人为了让计算机按自己的意愿处理数据，必须用程序设计语言来表达所要处理的数据，同时用程序设计语言来表达数据处理的流程，因此，程序设计语言必须具有表达数据和处理数据(称为控制)的能力。

世界上的数据有多种多样，为了使计算机程序设计语言能有效地、充分地表达各种各样的数据，一般将数据分为若干种数据类型。数据类型是对某些具有共同特点的数据集合的总称。对某种数据类型来说，一般涉及两方面的内容：即该数据类型代表的数据是什么(数据类型的定义域)、能在该数据类型上做些什么操作(运算)。在程序设计语言中，一般都事先定义好几种基本的数据类型，供程序员直接使用，如整型、实型(浮点型)、字符型等。同时，为了使程序员能更充分地表达各种复杂的数据，大多数程序设计语言还提供了构造新的具体数据类型的手段，如数组、结构、文件、指针等。因此，要掌握一门程序设计语言，首先得知道该语言提供了哪些基本数据类型，在这些基本数据类型上可以进行哪些操作，如何输入/输出这些基本数据类型的数据，同时还要熟练地掌握并使用数组、结构、文件、指针等。

程序设计语言除了能表达各种各样的数据外，还必须提供一些手段来表达数据处理的过程，即程序的控制过程。程序的最基本的控制是通过程序语言的控制语句来实现的。控制语句一般有两大类：选择控制和循环控制。计算机在执行程序时，一般是按照语句的顺序执行的，但在许多情况下需要根据不同的条件来选择所要执行的语句。有时，经常需要重复执行某些相同的操作，即进行循环控制。这些控制方式称为语句级控制。

当我们要处理的问题较复杂时，所编写的程序往往也较大。为了增强程序的可读性，便于程序维护，往往将程序分为若干个相对独立的“模块”，即子程序。因此，程序设计语言一般还提供一种手段来定义和使用这些程序模块。在C语言中，子程序的作用是由函数来完成的。

一个C语言程序可由一个主函数和若干个其他函数构成，由主函数调用其他函数，其他函数也可以相互调用。同一个函数可以被一个或多个函数(包括它自己)调用任意多次。函数调用时可传递零个或多个参数，函数被调用的结果将返回给调用函数。这种涉及函数定义和调用的控制称为单位级控制。

总的来说，要掌握一门程序设计语言，最基本的是要掌握如何用程序设计语言来表达数据，如何实现程序的控制。

1.2 C语言的特点

如果您想编写高质量的计算机程序，那您就应该采用C语言作为计算机编程语言，因为它功能强、语句表达简练、控制和数据结构丰富灵活、程序时空开销少，它既具有诸如Pascal、Fortran、COBOL等通用程序设计语言的特点，又具有汇编语言中字位、地址、寄存器等其他高级语言所不能及的低级操作能力。它既适合于编写系统软件，又可用来编写应用软件。它的这些特性是与其发展的过程分不开的。

早期的系统软件(包括操作系统)主要是用汇编语言编写的，因而程序与计算机硬件的关系十分密切，使程序编写困难、可读性差、难于移植，这样就要求有一种与硬件关系不紧密的高级语言用于编程，但以往高级语言缺少汇编语言的某些操作功能，使系统软件的编写十分困难。1972年工作于贝尔实验室的Dennis Ritchie在B语言的基础上设计并实现了C语言，随后，D. Ritchie又和Ken Thompson一起设想用C语言来构造一批软件工具作为软件工作者的开发平台，这些平台包括不大依赖于计算机硬件的操作系统和语言编译软件。由于UNIX操作系统也是由他们两人用汇编语言编写而成的，并作为一种典型的分时操作系统在当时有一定的代表性，因而他们在1973年对UNIX作了重写，其中90%以上的代码是用C语言改写的，增加了多道程序设计能力，同时大大提高了UNIX操作系统的可移植性和可读性。在以后的若干年中，C语言又作了多次修改，并渐渐地形成了不依赖于具体机器的C语言编译软件，成了当今应用最广泛的计算机语言之一。

目前，在不同型号计算机的不同操作系统下，都出现了不同版本的C语言编译程序，这些C语言编译程序有各自的特点。一般说来，1978年B. W. Kernighan和D. Ritchie(称K&R)合著的《The C Programming Language》是各种C语言版本的基础，称之为旧标准C语言。1983年，美国国家标准化协会(ANSI)制定了新的C语言标准，称ANSI C。现在出现的如Microsoft C、Turbo C、Quick C等都把ANSI C作为它们各自的一个子集，并在此基础上做了合乎它们各自特点的扩充。但无论是哪种版本的C语言，它们都具有如下一些共同的特点：

1. C语言是一种结构化语言

C语言的主要成分是函数，函数是C语言程序的基本结构模块，程序的许多操作可由不同功能的函数有机组装而成，从而易达到结构化程序设计中模块化的要求。另外，C语言还提供了一套完整的控制语句(如循环、分支等)和构造型数据类型机制(如结构、联合等)，使操作和概念描述也具有良好的结构性。

2. C语言语句简洁、紧凑，使用方便、灵活

C语言一共只有32个关键字，9种控制语句，程序书写的形式自由，压缩了一切不必要的成分。如用花括号“{”和“}”代替复合语句的开始与结束，用“++”和“--”运算符表示加1和减1，用三目运算符“?:”来表示一个简单的if-else语句，一行中可书写多个语句或一个语句可

书写在不同的行上，可采用宏定义和文件包含等预处理语句，等等。这些都使C语言显得非常简洁、紧凑。

3.C语言程序易于移植

这是因为C语言编译中与硬件有关的因素从语言主体中分离出来，通过库函数或其他实用程序实现它们。这特别体现在有关输入输出操作上，因为C语言不把输入输出作为语言的一部分，而是作为库函数由具体实用程序实现，这大大提高了程序的可移植性。

4.C语言有强大的处理能力

由于C语言引入了结构、指针、地址、位运算、寄存器存取功能，在许多方面具有汇编语言的特点，从而大大提高了语言的处理能力。

5. 生成的目标代码质量高，运行效率高

用C语言编写的程序，经编译后生成的可执行代码比用汇编语言直接编写的代码运行效率仅低15%～20%，这是其他高级语言无法比拟的。

当然，C语言也有一些不足之处，这主要表现在数据类型检查不严格，表达式易出现二义性，不具备数据越界自动检查功能，运算符的优先级与结合性对初学者难于掌握，等等。这些不足，要求学习C语言时引起充分的注意，否则会经常出现一些问题，而且对这些错误的查找也较为困难。

1.3 C语言程序概貌

1.3.1 C语言程序基本结构

C语言程序一般要存放在一个以".c"结尾的文本文件中(称为源程序)，但一个C语言程序可以由一个或多个源程序组成，而每个源程序的基本组成成分是函数，任何操作功能都应用函数完成。请看下面的例子。

【例1.1】 在屏幕上打印出Hello, World！的源程序如下：

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

运行结果为：

Hello, World！

在这个程序中，有两个函数：main和printf。任何一个C语言程序都是从函数main处开始执行的，这意味着main是一个特殊的函数，不管一个程序有多长或它要完成的功能是什么，都要在程序的某个地方有一个main函数，它常常还要引用其他的函数来完成某些特定的操作，这些被引用的函数可以是程序员自己编写的，也可以是以前写好的函数库中的函数(如例1.1中的printf函数)。

函数间数据通信的一种方法是通过参数，函数名后面的小括号把参数表括起来。例1.1中的main函数是无参数函数，所以括号中是空的，用"()"表示。大括号"{}"把构成函数的语

句括起来，这些语句称为函数体，在例 1.1 中的函数体只有一个语句：

printf("Hello, World！ \n");

该语句由两部分组成：函数调用和分号。printf("Hello, World！ \n")是一个函数调用，其作用是向屏幕打印出一串信息；而分号";"表示该语句的结束。要强调的是在 C 语言中，任何语句都要由分号结束。printf 是一个标准库函数，由系统提供。对这些函数，程序员可以直接引用。例 1.1 中对它引用时，用到了数据传递，参数是"Hello, World！ \n"，它是由双引号引起来的，称之为字符串。在字符串中，\n 是一个表示换行的字符，使屏幕的光标移到下一行的左边，它以转义序列方式给出。C 语言中对一些不可见字符都要用转义形式给出，如水平制表符用 \t 表示，回车符用 \r 表示等，这些在第二章中会详细讨论。

若在能够显示中文的环境中要打印中文信息，也可用中文字符串取代，如：

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

【例 1.2】 分析下面 C 语言程序的结构。

```
1   /* 把小写字母转换成大写字母的源程序 */
2   #include <stdio.h>
3   main()
4   {
5       int c;
6
7       while((c = getchar()) != EOF)      /* 字符还未读完 */
8         if( c >='a' && c <= 'z')
9             putchar(c+'A'-'a');      /* 把小写转换成大写 */
10        else
11            putchar(c);
12      printf("ok! \n");
13  }
```

注意：为了便于分析，在源程序中的每一行前加了行号，行号不是源程序的一部分。

第 1 行：由于该行中把内容放在"/*"与"*/"之间，所以它们被当作注释用。注释中可以出现任何字符，C 语言编译程序是不管的，它们不被编译成可执行的代码。注释可以出现在程序中的任何位置，如第 7、9 行中也出现了注释，目的是使程序更为人所理解。

第 2 行：由于是由"#"开头的，所以它只是一个预处理语句，include 预处理命令表示文件包含功能，<stdio.h>给出了一个被包含进来的文件名，文件 stdio.h 是系统事先定义好的标准输入输出文件，它里面说明了许多文件操作及定义了一些常量，如 getchar()、putchar()、EOF 等。

第 3 行:主函数定义头。

第 4 行:是主函数定义的开始,第 13 行是主函数定义的结束,两者之间的是主函数体。

第 5 行:是对变量 c 的定义,把它定义成整型 int。

第 6 行:是一个空行,它并不影响程序的任何功能,目的是使变量定义与语句分开,使程序书写更清晰、更易阅读。

第 7 行至第 11 行是一个循环。while 后紧跟循环条件:(c = getchar()) ! = EOF,它通过函数调用 getchar()从键盘上读入一个字符,并用赋值运算符"="把读到的字符赋给变量 c,然后判断是否为文件结束(EOF),若不是则执行循环体(第 8 行至第 11 行),否则执行第 12 行。

第 8 行至第 11 行是由 if-else 条件语句构成的循环体,条件是(c >= 'a' && c <= 'z'),判断 c 所含的字符是否为一个小写字符,若是,则执行函数调用 putchar(c + 'A' - 'a'),参数是一个表达式,其结果是把 c 所含的小写字符转换成大写,结果传递给 putchar()后就在屏幕上打印出了该字符。若 c 所含的字符是一个大写字母,就调用 putchar(c),直接在屏幕上打印出未经转换的字符。

第 12 行:不属于循环体,它打印"ok!"后便使程序执行结束。

【例 1.3】 统计输入中含数字'0'～'9'、空白字符(空格、制表符、回车符)和其他字符的个数。源程序如下:

```
/* 统计输入中含数字、空白字符(空格、制表符、回车符)和其他字符的个数 */
main()
{
    int c, i, nother;
    int ndigit[10];      /* 用于分别存放'0'～'9'数字的个数 */

    nwhite = nother = 0;
    for(i = 0; i < 10; ++i)
        ndigit[i] = 0;       /* 数组初始化 */
    while((c = getchar()) ! = EOF)
        if(c >= '0' && c <= '9')
          ++ndigit[c - '0'];   /* 统计数字个数 */
        else if(c == ' ' || c == '\n' || c == '\t')
          ++nwhite;            /* 统计空白字符个数 */
        else
          ++nother;            /* 统计其他字符个数 */
    printf("digits = ");
    for(i = 0; i < 10; ++i)
        printf(" %d", ndigit[i]); /*输出数字'0'～'9'个数*/
    printf(", white space = %d, other = %d \n", nwhite, nother);
}
```

程序中 nwhite、nother 分别统计空白字符和其他字符的个数；用一数组 ndigit 分别统计数字'0'～'9'出现的次数。数组 ndigit 的大小为 10，C 语言规定数组下标从 0 开始，因而数组 ndigit 含 10 个分量，它们分别是 ndigit[0]、ndigit[1]…ndigit[9]，其中 ndigit[0]用于统计'0'出现的次数，ndigit[1]用于统计'1'出现的次数…。程序首先将用于统计的各变量和各数组分量初始化为 0，while 循环每次读入一个字符，并存在变量 c 中，若 c 的内容为数字(c>='0' && c<='9')，则将相应统计分量加 1(++ ndigit[c - '0'])。例如若 c='2'，则'2' - '0' = 2，将 ndigit[2]的值加 1。若 c 为空白字符(c==' ' ‖c=='\n' ‖ c== '\t')，则将 nwhite 加 1，其他情况将 nother 加 1。最后，程序通过 for 语句将'0'～'9'的统计次数分别输出，然后输出 nwhite 和 nother 中的值。

【例 1.4】 输入两个整数，输出它们的最大者。源程序如下：

```
/* 求两个整数的最大者的源程序 */
main()                              /* 主函数 */
{
    int x, y, m;                   /* 变量定义 */
    int max();                      /* 说明将要调用的函数类型 */

    printf("Input x and y: "); /* 打印一个信息 */
    scanf("%d %d", &x, &y);    /* 输入变量 x 和 y 的值 */
    m = max(x, y);             /* 调用函数 max，并把返回值赋给 m */
    printf("Max = %d\n", m);       /* 打印 m 的值 */
}
/* 下面定义函数 max，函数值为整型，它有两个整型参数 a、b */
int max(int a, int b)       /* 函数名，函数类型，参数个数的定义 */
{
    if(a > b)
        return(a);    /* 当 a 大于 b 时，返回 a 的值，并由 max 带回至调用处 */
    else
        return(b);    /* 否则返回 b 的值，并由 max 带回至调用处 */
}
```

本例中有两个程序员自己编的函数：主函数 main 和被调用函数 max。而 max 的作用是从参数 a、b(称为形式参数)传递进来的两个整数中选择一个大的，作为 max 函数的值，并返回。

从上面的几个例子中可以看出 C 语言程序的一些基本概貌：

①C 语言程序是由函数构成的，而且必须要有一个以 main 命名的主函数。

②在程序中可以有以注释符"/*"开始，以注释符"*/"结束的注释内容，用来注释程序中的功能，帮助理解。但注释本身不会被编译成目标代码。

③任何一个 C 语言语句必须以分号";"结束。

④函数间的数据传递可以通过参数进行。

⑤在任何函数的定义中,可以有变量说明,而且任何变量在使用前必须有类型说明。

⑥函数定义中的函数体必须用大括号"{}"括起来。

⑦若在一个函数中要调用其他函数,事先要说明被调用函数的类型(如例 1.4 的 main 函数中说明:int max();)。

⑧一个源程序中可以定义多个函数(如例 1.4)。

⑨C 语言程序的书写格式是自由的。一行中可以有多个语句,一个语句也可分写在多行上。

⑩一个 C 语言程序中可以有预处理指令,以进一步增强 C 语言程序的结构和可移植性(如例 1.2)。

当然,只有在学完了 C 语言的全部知识后,才能全面了解 C 语言程序的结构与风格。

1.3.2 C 语言的基本输入输出函数

为了使以后的学习更加方便,这里要简要地介绍一下有关输入输出函数的基本用法(详细讨论见 8.2)。这些输入输出函数都是系统提供的库函数,它们定义在系统包含文件 stdio.h 中,所以最好在程序开始时用预处理命令 #include <stdio.h>。

①从键盘上输入一个字符用 getchar()函数,其一般用法如下:

```
char ch; /* 或 int ch; */
ch = getchar();
```

它从键盘读入一个字符,并赋给变量 ch,ch 的类型为字符型(char)或整型(int)。若遇到文件结束或读入错误,则返回 EOF。

②向屏幕输出一个字符用 putchar()函数,其一般用法如下:

```
putchar(ch);
```

即:向屏幕输出存放在变量 ch 中的字符,ch 的类型可以是字符型(char)或整型(int)。

③向屏幕格式化输出一串信息用 printf()函数,其一般调用格式为:

```
printf(格式控制, 输出参数1, ..., 输出参数n);
```

其中"格式控制"是一个字符串,它可以含有以"%"开头的控制字符,也可含有普通字符;而输出参数则是一些要输出的数据。如:

```
printf(" X=%d      Y=%d" ,x,    y);
       |_____格式控制_____|  |_输出参数_|
```

当 x、y 的值分别为 10 和 500 时,输出结果为:

```
X=10      Y=500
  |_____|_____|_____→ 格式控制中的普通字符
```

输出参数的类型、个数、位置一般要与"格式控制"中的控制字符的性质相对应。如上例中,输出参数 x 与第一个 %d 对应,输出参数 y 与第二个 %d 对应。

这里先介绍 printf()中可出现的几个常用控制字符:

d——将参数转换成十进制整数后输出。

c——将相应参数解释为单个字符后输出。

s——将参数解释为一个字符串后输出。

f——将参数转换成十进制浮点数(float 或 double)后输出。

在 printf()函数中,格式控制部分还可使用修饰符“-”来控制输出向左对齐或向右对齐。若在格式控制中%后紧跟符号-,表示输出向左对齐,否则向右对齐。在 printf()还可利用 C 语言中的转义字符:\n、\t、\0、\\、\f、\r、\101 等。

④从键盘上格式化输入数据用 scanf()函数,其格式与 printf()类似:

scanf(格式控制, 输入参数 1, ..., 输入参数 n);

其中“格式控制”是一个字符串,先介绍 scanf()几种常用的格式控制符:

d——以十进制方式输入整数。

c——读入一个字符。

s——读入一个字符序列。该序列以第一个非空白字符开始,读入字符后以第一个空白字符结束。

f——读入一个浮点数。

与 printf()一样,在 scanf()函数中,输入参数的个数、类型等应与格式控制中的格式符相对应,而且必须是指针,它指出转换后的每一个输入应存储的地址,对一般的变量要用“&”运算符取得地址。

【例 1.5】 求三角形的面积。请着重注意 printf()与 scanf()的用法。

源程序如下:

```
#include <stdio.h>              /* 输入输出的包含文件 */
#include <math.h>               /* 数学函数的包含文件 */
main()
{
    float a, b, c;
    double area(), areas;

    printf("请输入三角形的三个边长 A B C\n");
    scanf("%f%f%f", &a, &b, &c);
    if(a <= 0 || b <= 0 || c <= 0) {
        printf("三角形边长错:有一边的长小于等于零\n");
        exit(0);
    }
    if(a+b <= c || a+c <= b || b+c <=a) {
        printf("三角形边长错:有一边的长大于等于其他两边的和\n");
        exit(0);
    }
    areas = area(a, b, c);
    printf("三角形的面积为:%f\n", areas);
```