C程序开发技术及工具库

王锡江
潘金贵 编译

TOOLBOX

南京大学出版社
NANJING UNIVERSITY PRESS

C 程序开发技术及工具库

王锡江　潘金贵　编译

南京大学出版社

1992・南京

(苏)新登字第 011 号

内 容 简 介

本书以大量通用程序实例说明 C 语言的特点、结构和使用，以及程序设计的技术和算法，每个实例从简单到复杂，不断深化 C 语言的成份，十分便于学习和使用。

本书注重程序设计的方法和技术，以提高程序的质量。对于 C 语言的初学者来说，可通过大量的实例来学习 C 语言；而对于已经熟悉 C 语言或有 C 语言经验的程序人员来说，书中的大量实例可根据编程需要选择使用，是一本极好的工具书。编译者曾利用书中的程序实例，极其方便地实现了一个实用的终端仿真软件。

本书对于大专院校计算机类专业的教师、研究生和大学生等都是一本极好的教科书和教学参考书，对于在 IBM PC 机上工作且使用 C 语言的广大程序人员来说亦是一本极好的工具书。也可供从事计算机工作的科研人员、工程技术人员以及其他有关人员学习参考。

C 程序开发技术及工具库

王锡江　潘金贵　编译

*

南京大学出版社出版

（南京大学校内）

江苏省新华书店发行　江苏省国营练湖印刷厂印刷

*

开本：787×1092　1/16　印张：24　字数：601 千

1988 年 11 月第 1 版　1992 年 6 月第 3 次印刷

印数：8001—14000

ISBN　7-305-00228-3/TP・17

定价：8.80 元

前　　言

C语言是一种通用的程序设计语言，它有着经济实用的表达式，先进的控制流和数据结构以及丰富的运算符。语言简单、灵活，易于实现。用C语言书写的程序不但易读而且目标质量好。C语言不针对任何系统和具体的机器，语言本身考虑了移植性，因此有着良好的可移植性。C语言可用于书写系统程序，著名的UNIX系统就是用C语言写的。同样C语言也非常适合书写数值计算、文本处理和数据处理等应用程序，有着良好的应用前景。

本书不同于其他类型的C语言书籍，而是以大量通用程序实例说明C语言的特征、结构和使用，以及程序设计的技术和算法。每个实例从简单到复杂，不断深化C语言的成份，十分便于学习和使用。本书注重于程序设计的方法和技术，程序的风格和质量。对于C语言的初学者来说，可通过大量实例来学习C语言，这是学习计算机语言的另一种方法。而对于已经熟悉C语言或者有C编程经验的程序人员来说，书中的大量实例可根据自己的编程需要选择使用，是一本极好的工具书。本书的编译者曾利用书中的程序实例和工具，极其方便地实现了一个实用终端仿真软件。而利用工具库，将终端仿真软件修改和重组，又很容易地实现了一个实用的联机自动收/发报和自动译码软件。

全书共分九章。第1章和第2章快速且简洁地介绍C语言的基本概念、特征和术语，以及建立和编译C程序的过程，并且给出工具库的概念。这两章介绍的特征和概念对于理解本书的后面章节是非常重要的。

第3章介绍VIEW程序，进而引出文件输入、交互式键盘输入和显示技术。这一章强调了以模块为基础测试程序。

第4章介绍检查数据文件的工具，提供了一种方法来检查任何文件的格式。因为这章的程序功能接近于第3章的程序，所以这章主要强调怎样利用已有的程序来开发新的程序。

第5章建立基于排序算法的工具。首先介绍两种著名的算法，即插入排序和快速排序，然后实现基于算法的库函数，使其适用于任何种类数据。最后介绍合并技术，用以排序大型文件。

第6章建立BTREE模块，给出对数据文件按关键字存取的工具。这个模块相当长，给出了一些索引文件的基本概念和特征。最后介绍了一个应用实例。

第7章介绍一个低级工具库。虽然C语言能满足大部分需要，但有一些事情还是需要用汇编语言的函数工具来完成。这一章介绍了如何访问MS-DOS和BIOS以及IBM-PC的硬件，并且给出了相应的工具。

第8章介绍终端仿真程序，讨论实时程序的基本特征和实现技术，如查询、缓存和中断等。

第9章介绍如何优化程序和处理临界错误。本书的绝大多数程序均使用C语言一般特征实现，而使用高级特征可优化程序，提高执行速度，但也带来其他问题，本章比较了它们的得失。

与本书配套的还有一套《C语言工具库》软盘，使用者可利用工具库盘中的通用工具开发

自己的应用程序，可提高开发程序的效率和质量。如果读者需要这套《C 语言工具库》盘，可与本书的出版社或编译者联系而获得。

本书由王锡江和潘金贵合作编译，其中第四—六章由潘金贵同志完成，其余均由王锡江同志完成。

本书承蒙我系张福炎教授的支持和指导，对本书提出了许多宝贵的意见，特在此表示衷心的感谢。

王启祥副教授校阅了本书，值此一并表示感谢。

限于水平和时间仓促，书中难免会有许多错误。请读者指正并反馈给我们，以便再版时改正。

编译者　1987 年 11 月

于南京大学计算机科学系

目　　录

第一章 C语言简介

本章快速并且简单地介绍一下C语言的基本概念和术语，更高级的特征在后面的章节中陆续学习。短小的程序不能完成实用的功能，但是它们能够说明C语言的基本特征。本章就是通过一些典型的短小程序说明C语言的各种基本成份和特点，为后面的章节学习提供一个基础。

计算机的高级语言之间都有一些相似之处，如果读者熟悉BASIC、PL/1、PASCAL或FORTRAN等高级语言，那么就应该能够很容易地学习C语言，并且能够认识到引进C语言中的一些特征的目的。如果读者以前不熟悉C语言，那么本书末列出的参考书可帮助学习C语言。如果已经熟悉C语言，那么本章仅帮助复习一下概念和术语。

§1.1 HELLO程序：C程序的结构

第一个程序非常简单，它在屏幕上显示"hello，world"字符串。这里，使用它说明一个C语言程序的基本结构和如何将它变成一个可执行程序。图1.1给出了该程序的源文件清单。

```
1: * #include "stdio.h"
2: *
3: * main()
4: * {
5: *       printf("hello,");
6: *       printf("world");
7: * }
8: * ^C
```

图1.1 hello.c源程序

该程序的第1行告诉编译器在编译时，把文件名为stdio.h的文件嵌入到指定文件中去。stdio.h文件总是和编译器一起提供的，关于该文件中的内容暂时不讨论。第3行到第7行定义了一个名为main的函数，第3行建立函数名，第4行到第7行定义函数体，描述该函数做什么。

C语言的所有函数具有下列的一般格式：

```
函数名(…)
· · ·
{
函数体(做什么)
}
```

其中：…表示省略的成份，这里暂不说明。C语言的程序由函数定义组成，当调用函数时，每个函数定义描述了要做的事情。当执行HELLO程序时，也就是调用了main函数，当然该函

数也可以调用其他函数。

函数体由一个或多个语句组成，如果该函数不做任何事情，那么函数体就是空的，没有语句。HELLO 程序的第 5 行和第 6 行是这样一种形式的语句，这两个语句本身又是函数调用。第 5 行调用名为 printf 的函数，该函数的功能是将信息显示在屏幕上，该函数执行完成后，应返回到被调用点。第 6 行使用不同的信息重新调用 printf 函数，printf 函数做什么取决于提供给它的信息，这种信息称为参数。printf 函数是标准 C 函数库提供的，不需要用户自己写；该函数的使用应参照 C 编译器手册的文本，根据文本，能够使用 printf 函数而不需要知道它是怎样实现的。

图 1.1 中的语句具有下列形式：

函数名(…)；

这也是函数调用的一般形式，其中：函数名为调用函数名，…为调用函数的参数，可缺省。

后面的程序将给出其他形式的语句和该语句的更一般的形式。在第 5 行和第 6 行中使用的参数是字符行常数，字符行常数的格式是“可打印字符”，即由双引号括起来的字符串。

编译和执行 HELLO 程序

图 1.2a—d 说明了在 IBM-PC 机上使用 Lattice C 编译器准备 HELLO 程序的过程。细节取决于所使用的具体编译器，但是编辑、编译、链接和执行程序的过程和图 1.2 是相同的。

```
1: */*.....*/
2:                    Figure 1.2 — Creating and Executing Hello
3:
4:      Figure 1.2a — using an Editor
5,
6:      D )edlin hello.c
7:      New file
8:      *i
9:
10:
11:
12:
13:
14:
15:
16:
17:     *e
18:
19:     D )
20:
21:     Figure 1.2b — Compiling the Program
22:
23:     D )lc hello
24:
```

```
25:     D >Lc1 hello
26:     Lattice C Compiler(Phase 1) V2.00
27:     Copyright (C) 1982 by Lattice,Inc.
28:
29:     D >Lc2 hello
30:     Lattice C Compiler(Phase 2) V2.00
31:     Copyright (C) 1982 by Lattice,Inc.
32:     Module size P=0017 D=000E
33:
34:     Figure 1.2c — Linking the Program
35:
36:     D >link cs hello,hello,nul,lcs
37:
38:     IBM Personal Computer Linker
39:     Version 2.00 (C) Copyright IBM Corp 1981,1982,1983
40:
41:     D >
42:
43:     Figure 1.2d — Executing the Program
44:
45:     D >hello
46:     hello,world
47:     D >
48:/*-------------------------------------------------------------------*/
49:/*                        图 1.2  hello.fig                          */
```

如图 1.2 所示，首先使用 EDLIN 或 WordStar 输入 C 源程序文件(并且存储输入的文本作为文件)。图 1.2a 显示了使用 EDLIN 的编辑过程，这里建立的文件为源文件。编译器读入源文件并翻译成 IBM-PC 机能够执行的指令和数据，图 1.2b 显示了编译的过程。LC 是一个批处理文件 LC.bat，它执行两个程序 LC1 和 LC2。有些 C 编译器由 4 个分别程序组成，但是最终总是产生可执行的指令和数据。由编译器产生的文件称为目标文件，但是它还不完整，还不能执行，因为它要使用 printf 库函数。链接过程就是组合目标文件和任何必需的函数，产生一个完整的、可执行的程序。这个程序作为运行文件存储，取名为 hello.exe。图 1.2c 显示了链接过程，一个特殊的目标文件 cs.obj 总是和 Lattice C 编译器一起提供的，它建立 C 函数所需要的环境，并且在链接命令中总是第 1 个目标文件。在链接命令行上的最后一个名是函数库名 LCS.lib，它提供由 hello.obj 所需要的任何库函数。

现在可以执行这个程序了，输入程序的运行文件名就可执行，图 1.2d 显示了 HELLO 程序的执行结果。当执行 hello.exe 程序时，则调用 main 函数。C 希望每个程序中都有一个名为 main 的函数，这是一个主函数，当命名一个函数为 main 时，就是告诉 C 编译器，执行程序时从这儿开始。C 程序必须要有一个主函数，并且只能有一个主函数。

§1.2 SUMSQ 程序：变量、算术运算和循环

图 1.3a 给出了第二个程序，该程序显示数字 1 到 11 的平方和表。通过这个程序说明 C 语言中的注释、变量和算术运算等基本成份以及一些 C 语言的数据类型。图 1.3b 列出了这个程序的一个样本输出结果。

图 1.3a 中的第 1 行是一个注释，通常它被编译器忽略。在 C 语言中，注释是使用/*字符开始和*/字符结束的字符串。通常把具有文件名和说明该文件功能的一些词或句子作为这样的一个注释放在每个文件的开头，当列出源文件时，阅读注释就可知道每个文件的内容。

```
1:*     /*sumsq.c — print sum of squares table */
2:      #include "stdio.h"
3:
4:      main()
5:        {
6:          int i;
7:          int sum;
8:
9:          sum = 0;
10:         i = 0;
11:         while(i<11)
12:           { i = i + 1;
13:             sum =sum + i*i;
14:             printf(" %d     %d \n",i,sum);
15:           }
16:       }
17:/*-------------------------------------------------------------------*/
18:/*                          图 1.3a  sumsq.c                         */
```

```
1:*  D>sumsq
2:     1  1
3:     2  5
4:     3  14
5:     4  30
6:     5  55
7:     6  91
8:     7  140
9:     8  264
10:    9  285
11:    10  385
12:    11  506
13:
14:  D>
```

```
15:/*-----------------------------------*/
16:/*      图 1.3b  sumsq.fig */
```

和 HELLO 程序一样，第 2 行将 stdio.h 文件嵌入到该程序进行编译。

第 6 和第 7 行说明取名为 i 和 sum 的两个变量，这样的说明有两个目的：

（1）定义变量名并且赋于它们数据类型（这里变量 i 和 sum 的数据类型说明为整型，即 integer）；

（2）为存储变量的值分配存储空间。

C 语言与 BASIC 和 FORTRAN 语言不一样，它要求对数据变量进行明确的说明。

变量说明具有下列一般形式：

数据类型　变量名；

第 9、第 10 两行对变量 i 和 sum 赋初值 0，这是两个赋值语句，注意每个赋值语句必须使用分号（;）作为语句结束符。第 13 行是另一个赋值语句，在这个赋值语句中使用了算术运算符＋和＊；除了加（＋）和乘（＊）运算符，C 还提供减（－）和除（/）运算符。所有这些运算符都是双目运算符，它们使用两个操作数，一个在算符的前面，另一个在算符的后面，具有下列一般形式：

操作数　双目运算符　操作数

C 也提供单目运算符，它使用一个操作数。例如，很多其他语言都具有的负号（－），它求操作数的负数。下面是单目运算符的一般形式：

单目运算符　操作数

如何区分双目运算符减号和单目运算符负号呢?C 有一个规则区别它们的使用：如果运算符数两边都有一个表达式，则它是双目运算符减号；否则，它是单目运算符负号。

赋值语句的一般形式为：

变量＝表达式；

其中：表达式可以是单个变量或单个常数或者是包括多个运算符的表达式；在 C 中，定义赋值号（＝）左边的变量是存储值的地址。

第 11 行到 15 行是 while 循环语句，它表示只要变量 i 的值小于 11，则重复执行第 12 到 15 行。＜是比较操作符，它比较操作符两边的操作分量。其结果为逻辑值真或假，用于控制程序的执行。

C 提供了一组比较操作符：

a＜ b　如果 a 小于 b，结果为真；

a＜＝b　如果 a 小于或等于 b，结果为真；

a＞ b　如果 a 大于 b，结果为真；

a＞＝b　如果 a 大于或等于 b，结果为真；

a＝＝b　如果 a 等于 b，结果为真；

a! ＝b　如果 a 不等于 b，结果为真。

while 循环语句的一般格式如下：

while（测试的条件）

循环的语句

在 sumsq 程序中，因为希望第 12 行到 14 行的语句重复执行，所以使用一对花括号括起这

几个语句而建立一个复合语句。当测试的条件值为真时，整个复合语句被重复执行。

第 14 行调用在 HELLO 程序中使用的 printf 函数，这次传送三个参数：

第一个参数：字符串。其中%字符告诉 printf 函数后面要打印数据的格式，d 字符指出打印的整数值使用十进制形式，这样第 2 个参数将被转换成一串字符；printf 函数回溯扫描字符串，找到第二个%d 格式说明，再找第 3 个参数，也将它转换成一串字符；继续字符串的扫描，因为再没有%字符，则输出字符串的剩余部分，然后 printf 函数返回。

第二个参数：变量 i 的值。

第三个参数：变量 sum 的值。

HELLO 程序中的 printf 函数使用一个参数，这里使用三个，printf 函数如何理解这种区别呢？字符串中的两个%d 格式说明告诉 printf 函数寻找并且输出两个其他的参数。

第 14 行给出了语句的另一种形式：

函数名（…）；

事实上，C 语言中的语句可以是任何表达式，如下形式：

表达式；

因为 C 允许在一个复杂的表达式中使用赋值语句和函数调用，那么第 9 行和第 10 行的赋值语句以及第 14 行的函数调用就是 C 简单表达式的例子。为了增加程序的可读性，应尽量使用简单表达式。

第 14 行的字符串中包括一个新行符号\n，对于 C 编译器来说，某些输出字符具有特殊的含义并且不能直接使用在字符串中。新行字符就是这样的一个控制字符，C 识别该字符的一种特殊表达形式：即斜杠字符\指出下一个字符 n 是一个控制字符，当输出新行控制字符时，它使得后续的输出从新行开始。

§1.3 WEATHER 程序：控制台输入、for 语句、变量地址和符号常数

图 1.4a 给出的 weather 程序接收一星期的每天温度并且计算一星期的平均温度，它也计算温度低于冰点以下的天数。图 1.4b 给出该程序的样本运行结果。

```
1: *    /* weather.c — calculate average temp. and # cold days */
2:      #include "stdio.h"
3:
4:      #define FREEZE      32      /* freezing temperature   */
5:
6:      main()
7:        {
8:          int i ,temp , ncold ;
9:          float sum;
10:
11:         printf("enter temperatures for the week \n");
12:         ncold = 0;
```

```
13:          sum = 0;
14:          for(i=0 ;i <7;i=i+1)
15:            { scanf("%d",&temp);
16:              sum =sum+temp;
17:              if(temp< FREEZE)
18:                ncold = ncold+1;
19:            }
20:
21:          printf("average temperature was %3.1f \n",sum / 7.0);
22:          printf("%d days below freezing \n",ncold);
23:        }
24:/*-------------------------------------------------------------------------*/
25:/*                          图 1.4a  weather.c                             */
```

```
1:*     D >weather
2:      enter temperatures for the week
3:      15  20  32  40  65  72  55
4:      average temperature was 42.7
5:      2 days below freezing
6:
7:      D >
8:/*-------------------------------------------------------------------------*/
9:/*                          图 1.4b  weather.fig                           */
```

WEATHER 程序引进了几个新的 C 语言特征：scanf 函数接收键盘输入；for 语句提供程序循环；传送变量的地址作为参数和定义符号名作为数值常数。

程序的第 4 行定义单词符号 FREEZE 作为数值 32，这样后续程序部分使用数值 32 的地方都可以使用符号 FREEZE 代替，这是符号常数的概念。在程序中使用符号常数有很多优点，例如，如果转换该程序使用摄氏温度，则只要改变第 4 行的符号常数的定义，而程序的其他部分不需要改变。尽管这一点对这个小的程序例子不重要，但是对于实用程序，最好定义符号名代替数值常数。这样若要改变程序中所有使用该数值常数的地方时，只要改变一行，即改变该数值常数的符号常数的定义即可。注意第 4 行没有分号，#define 语句不是一个赋值语句，它仅表示从现在起，程序中所有使用数值 32 的地方都可用符号常数 FREEZE 代替。

第 9 行说明 sum 是一个浮点变量，C 使用明确的说明标识变量的数据类型。

第 14 行到 19 行是一个 for 语句，它的形式如下：

for(赋初值；测试条件；循环变量)

　　重复语句

它等价于下列的 while 语句：

赋初值；

while(测试条件)

　　{重复语句

　　循环变量

}

上例中，当 i＝0，1，2，…，7 时，for 语句重复执行第 15 行到 19 行。若 C 允许 for 语句中存在任何表达式时，则 for 语句可有如下的简单形式：

for(变量＝初值；变量＜终值；变量＝变量＋1)

语句

其中，可以使用其他的比较运算符和增量/减量操作符。

第 15 行从键盘上接收温度值，scanf 函数类似于 printf，但它是一个输入库函数，该函数的第一个参数告诉 scanf 函数怎样解释从键盘输入的字符，后面的参数告诉 scanf 函数从键盘上接收的数据放在哪儿。在 C 中，当传送参数到一个调用函数时，传送的是值，而不是存储变量的地址。这里使用 & 操作符给出温度变量的地址，这样就传送了地址到 scanf 函数，scanf 函数就能知道值存储在什么地方。

scanf 函数像 printf 函数一样，也是和 C 编译器一起提供的标准函数。然而 C 编译器并不知道 scanf 函数需要什么样的参数，程序员需要记住 printf 函数需要数据值作为参数，而 scanf 函数需要地址作为参数。

printf 和 scanf 都是控制台输入/输出函数，通常使用它们与 PC 的键盘和屏幕进行通信，它们完成和 BASIC 中的 PRINT 和 INPUT 相同的功能。

第 17 行和 18 行包含一个 if 语句，17 行测试输入的温度值，看它是否低于冰点，如果条件为真，执行第 18 行的语句。

第 16 行给出了一种混合数据类型的算术运算：整型和浮点型。C 有固定的规则处理混合数据类型的算术运算。

第 21 行给出了输出数据的另一种格式，%3.1f 格式说明指出应使用一个浮点数作为参数。输出的浮点数为 3 位整数、一个小数点和一位小数，在输出之前，printf 函数舍入数到一位小数。

§1.4 SORTNUM 程序：数组、函数返回值和指针

图 1.5a 给出的 SORTNUM 程序从键盘接收一批数据，并以递增顺序排序，然后输出排序结果。该程序最多排序 100 个数据，约定一个非数值的字符结束输入。图 1.5b 给出该程序的样本运行结果。该程序给出了几个新的特征：数组、函数返回值和指向变量的指针。

图 1.5a 的第 7 行说明一个整型数组，它为 100 个整型数分配存储空间，下标从 0 到 99。注意，下标的使用不要超出这个范围，C 不提供任何检查。第 12 和 19 行示范了数组元素的使用。

第 9 行到 14 行从键盘接收输入，当 scanf 函数被调用时，它返回一个值：成功接收的项数。该程序连续接收输入，一直到 scanf 调用失败，即遇到一个 q 或其他非数值字符时，停止接收。

第 16 行调用取名为 sortn 的函数，该函数的参数为数组和读进数组的数据个数。第 23 到 38 行定义了 sortn 函数，第 24 行和 25 行定义了 sortn 函数所需要的参数类型，这些说明类似于第 6 行和第 7 行的变量说明。数组的参数说明不需要指定数组的大小，这样任何大小的整数数组都能成功地使用。C 对待数组参数不同于普通变量，它传送数组的地址，而不是所有数组元素的值。

数据有选择地进行排序，首先找到最小的数并且放在最前面，然后从剩余的数据找到下一

个最小的数并且放在第二位。这个过程重复进行，一直到数组中仅剩下一个元素。第 29 行到第 37 行实现了这个算法，嵌套使用了两个 for 循环语句，里层的 for 语句仅仅是外层 for 语句循环体中的一个语句。

在选择最小数的每一步中，有时需要调用 swap 函数改变数据的顺序。对于 swap 函数，传送的参数是需要交换的数组元素的地址，和 WEATHER 程序一样，这里使用地址操作符 & 得到这些地址。

在 swap 函数中，第 41 行和第 42 行定义需要交换数据的地址，使用星号 * 说明 p1 和 p2 是指向整数的指针，而不是整数。当使用变量地址时，也使用一个星号在指针参数的前面。经常使用 p 作为指针变量名，以提醒用户它的数据类型。例如第 47 行：

```
*p1 = *p2;
```

表示取出 p2 指向地址的数据值，把它存储到 p1 指向的地址。

后面将介绍指针的其他使用，在 C 中变量地址的使用和变量值的使用一样容易。

```
1: *    /* sortnum.c -- sort input numbers */
2:      #include "stdio.h"
3:
4:      main()
5:        {
6:          int i,n,t;
7:          int a[100];
8:
9:          printf(" enter numbers-(type q to stop)");
10:         n=0;
11:         while(scanf("%d",&t)! =0)
12:           { a[n]=t;
13:             n=n+1;
14:           }
15:
16:         sortn(a,n);
17:
18:         for(i=0;i<n;i=i+1)
19:           { printf("%d",a[i]);}
20:       }
21:
22:
23:     sortn(x,nx)        /* put an array of ints into ascending order */
24:       int x[];         /* the array */
25:       int nx;          /* count of items to sort */
26:       {
27:         int i ,j ,pick;
28:
29:         for(i=0;i<(nx-1);i=i+1)
```