计算机知识普及系列丛书

IBM PC系列机C语言工具包

C语言程序设计实例集

战晓苏　赵立军　编写
吴　言　　审校

学苑出版社
1994.

(京)新登字 151 号

内 容 提 要

本书中的程序绝大多数是实用性或工具性程序。学习和掌握这些程序，用户可以更方便地在 PC 机上编程。所选的程序能够充分说明 C 语言的算法和编程技巧，是 C 语言的百宝箱。如果读者有用 C 语言简单编程的经验，就可理解书中程序。一般来讲，使用 C 语言编程是编写高水平程序的第一步。本书即是掌握 C 语言高级编程技巧的有力工具。

书中每个程序都给出了明确的基本结构，并且给出了程序功能和程序流程的完整描述，在每个程序之后，还将讨论如何增强该程序性能。如果只用 10 行到 50 行 C 语言语句就可达到改进程序设计的目的，那书中就一定会给出详细步骤。

本书的内容是实用性的，如果读者第一次读一个程序不能完全弄懂它，是非常正常的，可以先粗略读一下程序的内容，然后再将注意力投向读者所关注的细节。

欲购本书的用户，请直接与北京 8721 信箱联系，电话 2562329，邮码 100080。

计算机知识普及系列丛书

C 语言程序设计实例集

编　　写：战晓苏　赵立军
审　　校：吴　言
责任编辑：甄国宪
出版发行：学苑出版社　　**邮政编码**：100032
社　　址：北京市海淀区万寿路西街 11 号
印　　刷：施园印刷厂
开　　本：787×1092　1/16
印　　张：19.25　　**字　数**：456 千字
印　　数：1～4000 册
版　　次：1994 年 3 月北京第 1 版第 1 次
ISBN7－5077－0821－7/TP・19
本册定价：15.00 元

目　录

第一章　C语言概述

本章论述了C语言程序的结构，所给出的较短程序并不一定能完成实用的功能，但这些程序说明了C语言的基本特征。本书不是C语言入门教课书，只在本章给出了以后各章要用的基础，同时，用到的术语及概念也在本章加以介绍。C语言的许多先进特性将在后续章节陆续讨论。

计算机高级语言之间存在相似性，一种语言的某一特性往往可在另外一种语言中找到。如果你对C语言还不甚了解，可以先学习一下附录I)中列出的C语言入门教材。如果你对C语言较熟悉，那么本章将助你复习一下。可能你对C语言的术语、概念也很熟悉，本章会介绍书中是如何使用这些概念和术语的。

1.1 HELLO程序：C语言程序结构

■ 第一个程序很简单，它将在屏幕上显示：

hello, world

这个程序给出了C语言程序的基本结构，以及如何将它转换成可执行的程序。

图1.1是程序清单。第1行告诉编译器用名为stdio.h的文件内容作为编译时的辅助输入。stdio.h文件一般由第译器提供，我们在以后的程序几乎都要引用该文件。

图1.1　hello.c

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

第3行到第7行定义了名为main的函数。第3行是函数名，第4行到第7行描述了该函数的功能。所有的C语言函数都有这样的格式：

函数名（…）

…

{

执行的功能

}

因为是省略表示，所以就没有给出具体的C语言语句。一般来说，C程序是由函数定义组成的，每一个定义描述了调用该函数所完成的功能。当执行HELLO程序时，我们就调用main函数。同理，main也可调用其它函数。

函数的执行部分由语句组成(如果函数什么也不做，则无语句)，HELLO程序中第5行第

6行就是语句。第5行调用printf函数，printf的作用是在屏幕上显示信息。printf执行完以后，程序返回到它被调用的那一点。调用printf时，我们应给出要显示的信息。第6行再次调用printf，而显示信息发生了变化。printf显示什么取决于编程者提供的信息，这个信息叫作参数或变量。printf是C语言的标准库函数，不需我们专门编写。C编译手册中应有printf的使用说明，有此说明，我们就能使用printf而不必关心其如何实现。

图1.1的语句形式是：

函数名（…）；

第5行和第6行的参数是字符串常量，其格式为：

"可打印的字符串"

1.1.1 HELLO程序的编译和执行

图1.2说明了在IBM PC机上执行HELLO程序的过程，本例中使用了Lattice C编译器。编译器不同时执行细节略有不同，但编辑、编译、链接和执行这几步都是类似的。

图1.2 hello.fig

```
1
2                    Figure 1.2 - Creating and Executing Hello
3
4       图      1.2a - Using an Editor
5
6   D>edlin hello.c
7   New file
8   *i
9          1:*#include "stdio.h"
10         2:*
11         3:*main ( )
12         4:* {
13         5:*     printf ("hello, ") ;
14         6:*     printf (" world") ;
15         7:* }
16         8:*^C
17  *e
18
19  D>
20
21  图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 1) V2.00
31  Copyright (C) 1982 by Lattice, Inc.
```

```
32    Module size p=0017 D=0005
33
34    图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    图1.2d - Executing the Program
44
45    D>hello
46    hello, world
47    D>
```

首先，使用文本编辑器（或字处理器）输入C程序。输入完之后，将其存入文件中，图1.2a给出了使用EDLIN编辑器的编辑过程。我们可以使用任何一种编辑器，只要它能生成ASCII字符文件，而不包含特殊格式信息。我们称编辑后的文件为源文件，因为它是编译的原始文件。

编译器读入源文件，将它翻译成IBM PC可执行的指令和数据。图1.2b给出了编译过程。输入lc后，机器运行lc.bat批处理文件，lc.out执行lc1和lc2这样两个程序。一些C编译器由四个独立的程序组成，但最终结果都是产生可执行的指令和数据。习惯上，C源文件具有文件扩展名'C，有些编译器要求这种命名方式。编译之后生成的文件叫目标文件。

编译是将源文件翻译成可执行的形式，在程序能实际运行之前，还需一步——链接。链接过程就是将目标文件与必要的库函数联在一起，产生一个完整的可立即执行程序，这个程序以可运行文件的形式存储，名为hello.exe。

图1.2c显示了链接过程。Lattice C编译器提供了专门的目标文件cs.obj，它建立了C函数所需要的环境，链接命令中的第一个目标文件就是cs.obj，最后一个名字指的是lcs.lib库，hello.obj所需要的库函数都要在该库里搜索。

现在我们的准备工作已作好，输入程序的运行文件名就可以执行了，图1.2d显示了HELLO程序的执行。注意：我们不必输入完整的文件名——.exe扩展名可以省略。执行hello.exe，就调用main函数。在每一个程序中C都找到main函数，main告诉C编译器："从这里开始执行。"

1.2 求平方和程序：变量、算术式和循环

图1.3a的程序将显示从1到11的平方和，该程序含有注释语句、变量、算术式等一些语句类型，图1.3b是该程序的输出结果。

图1.3a c:sumsq.c

```
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   }
```

图1.3b c:sumsq.fig

```
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  204
10    9  285
11    10  385
12    11  506
13
14    D>
```

图1.3a中第1行是注释，编译时跳过注释。C语言中，注释以/*开始，以*/结束。程序开始的注释部分含有文件名及程序的功能说明，注释可以帮助我们阅读源文件。和HELLO程序一样，第2行说明stdio.h文件将在编译时用到，引入这个文件是很有益的，除非你知道源文件根本不需要stdio.h文件中的信息。

第6行和第7行是i和sum的变量说明，说明的目的有二：（1）定义变量名及其数据类型（这息是整型），（2）为存储变量分配空间。与BASIC和FORTRAN不同的是，C语言要求对变量进行严格的说明。变量说明的格式为：

数据类型 变量名;

第9行和第10行将0赋给i和sum，注意赋值语句的结束是一个分号。第13行是另外一个赋值语句，该语句用了算术操作符+和*，此外，C还提供了减（-）和除（/）操作符。它们都是二元操作符，要求两个操作数，操作符前后各一个：

操作数 二元操作符 操作数

C也提供了一元操作符，只要求一个操作数。一元减（-）和其它语言是一样的，对操作数求反。下面的例子说明一元减的一般格式和值用。

一元操作符 操作数

-i （一元减）

减和一元减的操作符相同，C为了区分其不同而作出了规定：如果操作符的两边都是一个表达式，则为二元减操作符，否则为一元减操作符。

赋值语句的格式为：

存储的地方＝要存储的值；

要存储的值是表达式。表达式可以是单个变量或常量，或者象13行那样包含几个操作符。在本例中，我们将表达式的值放到一个变量中，但要存储的地方也可以是包含几个操作符的表达式。C语言要求表达式的左边定义一个能存储值的地址。

第11行和第15行是while语句，意思是：只要变量i的值小于11就反复执行第12行到第13行。＜操作符比较左右操作数的值，其结果（真或假）用来控制程序的执行情况。C中常用的比较操作是：

a＜b　若a小于b则为真

a＜＝b　若a小于等于b则为真

a＝＝b　若a等于b则为真

a＞b　若a大于b则为真

a＞＝b　若a大于等于b则为真

a!＝b　若a不等于b则为真

循环while语句的格式是：

while（测试条件）

要重复的语句

因为我们希望12到14行重复，故用大括号将它们括起来，产生一个复合语句，当条件为真时重复整个复合语句。

第14行调用printf函数，这时我们传递了三个参数：

1.一个字符串。%告诉printf要打印数据的格式，d字符指的是要打印一个十进制整数，即printf中第2个参数。数字被转换成一系列字符，printf返回扫描字符串。printf发现了第2个%d说明，寻找第3个参数，这个参数被转换成一系列字符，继续扫描字符串。由于没有%字符了，就打印出其余串，printf返回。

2.变量i的值。

3.变量sum的值。

printf怎样知道要显示的格式呢？在字符串中有两个%d说明，它告诉printf要找另外两个参数并输出之。

14行给出了语句的另外一种格式：

函数名（…）；

事实上，C语句可以是以下任何表达式：

表达式；

第9、10行的赋值语句和第14行的函数调用是C表达式的例子。尽管C语言在更复杂的表达式中允许使用赋值语句和系统调用语句，但出于易懂性，我们保留简单形式。

14行的字符串包含了新的东西：\n。在C编译器中一些字符有特殊的意义，因此不能直接使用输出。这个换行符就是这样的控制字符，控制字符的前面有一个特殊标记（下划线\）。下划线字符\表示：其后面字符n是控制字符，当换行字符出现后，其后的输出在

下一行开始。

1.3 天气程序：键盘输入语句、变量地址和带符号常量

图1.4a的程序要求输入一周内每天的温度，然后计算本周的平均温度，统计温度低于零度的天数。图1.4b是运行结果。

图1.4a weather.c

```
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
26
27
```

图1.4b weather.fig

```
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>
```

WEATHER程序使用了C的几个新特性：用来接收键盘输入的scanf函数、程序循环for语句、作为参数传递的变量地址、定义数字常量的符号名。

第4行将FREEZE定义成32，程序中其余部分可用FREEZE代替32。如果程序中要使用摄氏温度，就只需改变第4行，其余部分不动。虽然在我们这个小程序中它的用处不大，但在实用程序中，为数字常量定义一个符号名是很有益的，这样改变程序中某个常数只需改一行它的定义即可。#define不是赋值语句，所以其后没有分号。

第9行将sum定义为一个浮点变量，C要使用变量说明来定义变量的数据类型。

第14行到19行是一个for语句，格式为：

for（只在开始做一次；测试条件；每次循环均做（改变条件））

{重复语句}

用while语句，则可表示成：

只在开始做一次；

while（测试条件）

{重复语句

每次循环均做（改变条件）；

}

对i=0，1，2，…，6，for语句重复执行15行到19行。for语句中允许存在任何表达式，这本书中的程序具有简单的形式：

for（变量=初值；变量<终值；变量=变量+1）

语句

这样的for语句很象BASIC和Pascal中的for语句。

第15行从键盘接收一个温度值。scanf函数很象printf，但它用于输入。第1个参数说明要从键盘输入一个整数，第2个参数说明输入数据存储的地方。当我们调用函数时，传递的是参数的值而不是地址。使用&操作符的目的是给出变量的地址，我们将该地址传给scanf，这样scanf可以在这个地址存储参数的值。

scanf也是C编译器提供的库函数，必须记住：printf要求数据值作为参数，而scanf将地址作为参数。

printf和scanf都是控制台输入/输出（I/O）函数，即与键盘和屏幕打交道，这与BASIC中的PRINT和INPUT执行的功能相同。

第17行和18行含有一个if语句，第17行判断输入温度是否低于结冰点，仅当条件为真时才执行18行。

第16行含有混合数据类型的算术表达式：整型和浮点型，C语言允许这样编写。当然你可以进行数据类型转换。

第14行中，整数7作为for循环执行条件。21行的浮点数7.0用以计算每周平均温度，这里我们亦可用整数7而让C编译器将它转换成浮点数。

第21行是输出的另一种格式。%3.1f的意思是：显示一个十进制浮点数（由printf中的参数提供），显示三位整数、小数点、小数点后显示一位（经过四舍五入）。

1.4 排序程序：数组、函数返回值和指针

图1.5a的程序从键盘接收数字，按递增次序排序，然后显示。程序可对100个数字排序，输入时以非数字字符作为结束标志，图1.5b显示了程序的运行情况。该程序又给出了几个新的特性：数组、函数返回值和指向变量的指针。

图1.5a c:sortnum.c

```
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)
30        {                        /* find the smallest remaining number */
31         pick=i;
32         for (j=i+1; j<nx; j=j+1)
33             { if (x[j]<x[pick])
34                    pick=j; /* element x[pick] is smallest so far */
35             }
36         swap (& x[pick], &x[i]) ; /* exchange - smallest first */
37        }
38    }
39
40  int swap (p1, p2)                /* swap two numbers */
41   int *p1;                        /* points to first number */
42   int *p2;                        /* points to second number */
43    {
44      int temp;
45
46      temp=*p1;
47      *p1=*p2;
48      *p2=temp;
49   }
50
```

图1.6b sortnum.fig

```
1   D>sortnum
```

```
2    enter numbers - (type q to stop) 4 6 5 3 2 1 q
3     1 2 3 4 5 6
4    D>
```

图1.5a中第7行是一个整数数组的说明，要给100个整数分配空间，下标从0到99，我们不能超越此范围访问数组。第12行到19行告诉读者如何处理数组中的元素。

第9行到14行从键盘接收输入，调用scanf函数，返回一个值——接收成功的数字。如果scanf接收的不是数字，程序就不再接收键盘输入（因为刚才的输入可能是q或其它非数字字符）。

第16行调用函数sortn，参数为数组和从键盘中输入数字的个数。第23到38行定义了sortn函数。第24行、25行定义了sortn形式参数类型，类似于第6行、第7行的变量说明，不过这里的数组不必知道其大小，任意大小的整数组均可以。C语言在处理数组形式参数时与普遍变量不同，数组是传递地址，而不是其元素的值。

排序的方法是：首先找到最小的数字放到数组的第一个元素，然后从余下的数据中选出最小的放到第二个位置，重复这一过程直到剩下一个数据为止。第29行到第37行实现了这一算法，用了嵌套的两个for循环。

每次最小数找到后，将它和当时未排序元素第一个数交换（因为最小数是指未排序的），这是由swap函数实现的，传递的参数是待交换元素的地址，这些地址是通过地址操作符&得到的。

在swap函数中，要交换数的地址参数在第41行和第42行定义，*号说明P_1和P_2是指向整数的指针。需要用到变量的地址时，我们在指针参数前也使用*号。第47行可翻译为：

*p1 = *p2;

取由p2所指向的地址中的值，将它存到由p1所指向的地址。

以后我们将学习指针的其它用法，C语言允许方便地使用变量的地址。

1.5 SENTENCE程序：文件的输入/输出、字符和I/O重新定向

图1.6a给出的程序统计文件中的句子数，这里只是简单统计文件中句号、问号和感叹号的出现次数。该程序帮助我们了解文件的I/O及字符常量，图1.6b给出了一个文件的例子和处理该文件的结果。

图1.6a c:sentence.c

```
1    /* sentence.c - count sentences */
2    #include "stdio.h"
3
4    int ns;
5
6    main ()
7      {
8       int c;
9
10      ns=0;
11                          /* get each character and check it */
```

```
12      c=getchar () ;
13      while (c !=EOF)
14         {check_end (c) ;
15          c=getchar () ;
16        }
17
18      printf ("%d sentences", ns) ;
19    }
20
21
22  int check_end (ch)      /* check for end-of-sentence char. */
23    int ch;               /* char to check */
24    {
25      if (    (ch=='.')
26           || (ch=='?')
27           || (ch==';') )
28           ns=ns+1;
29    }
30
31
32
```

图1.eb sentence.fig

```
1   D>type test
2   I write short sentences. I like them. Do you?
3   I hope you agree;
4
5   D>sentence <test
6    4 sentences.
7   D>
```

程序从磁盘文件中得到信息必须依靠I/O重新定向。整个程序很简单：从文件中取得一个字符，判断它是否为句子结束字符，是则将计数加1，依次重复这两步直到文件结束。

我们用getchar库函数从文件中一个一个地读取字符，每次调用，getchar就返回文件中的下一个字符。getchar不需要任何参数，所以括号内无参数，但括号必须要有。

当文件中没有字符时，getchar返回一个特殊值EOF (end-of-file)，EOF在stdio.h文件中一般都有定义。getchar返回字符的值在0到255之间，EOF与这些值不同(常为-1)，这一点必须保证。对不同的编译器，getchar的字符返回值范围不同，EOF应能够区别于它们的值。

因为getchar返回整数值，所以必须将变量C定义为整型，而不是字符型。如果C定义为字符型，EOF与其它的字符就区分不开了。C允许将字符看成数字，而且提供了字符到整数的自动转换。

22-29行定义了check_end函数，检查字符是否为句子的结束字符，是则将统计计数加1。if语句中有几个比较，“||”操作符的意思是“或”。如果字符是句号、问号、或感叹号，就执行第28行。C还提供了“与”操作符：&&。对25-28行的比较，我们使用了（）

将其括起来。在一个表达式中，几个操作符的执行顺序是有规定的，但可以用小括号来控制执行顺序。

句子计数ns的说明位于程序中所有函数的外面，所以main及check_end都可使用，不必作为参数传递。

第25-27行的比较操作使用了字符常量——逗号、问号和感叹号。字符常量的顺序为：单引号、字符、单引号。字符常量是单个字符，而字符串常量是多个字符，1.6节讨论了C如何表示每一种常量。

1.5.1 sentence的运行：I/O重新定向

与printf和scanf一样，getchar也是控制台I/O函数。DOS 2.0给出了一个方法，如何用文件来代替键盘输入，当执行SENTENCE程序时，在程序名的后面接上“<”字符和用作输入的文件名，图1.6说明了这一特性。

C语言提供了另外一个库函数putchar来输出单个字符，一般输出到屏幕上，但可以重新定向到一个文件上。下面的程序段输出字母表中的字母，如果执行该程序时接上>afile（接在程序名的后面），则输出转到了afile文件中。

```
...
main ( )
  {
  int c;
  for (c='a'; c<='z'; c=c+1)
    {putchar (c) ; }
  }
```

1.6 REVERSE程序：字符数组、串和分别编译

图1.7a的程序判断一个字是否为回文字（左读右读都一样），是则输出 * * palindrome * *，否则打印这个字的原文字。REVERSE接收一行输入作为被判断的文字。

程序中有字符数组的说明，使用了几个处理字符串的库函数。程序由两个源文件组成，所以需要分别编译。

图1.7a的第6行和第7行是两个字符数组的说明，可以存放文字及其回文字。说明部分和SORTNUM程序相似，但这里的数据类型是字符型而不是整型。

第9行给出提示，第10行调用函数getstr来接收从键盘输入的文字。getstr没有在reverse.c文件中定义，而且它也不是C标准库函数，它是在另外一个源文件中定义的，其原因为：

1.其它程序也可能要使用getstr。在许多程序中经常要用到这样一个函数：从键盘接收一行输入放到字符串中，如果这个函数放到一个单独的源文件中，则可以象标准库函数那样使用。

2.编辑、编译较小的源文件要容易些。

3.小模块的检查容易而且高效，当程序由小模块组成时修改是非常容易的。

图1.7a c:reverse.c

```
1  /* reverse.c - checks.a phrase to see if it is a palindrome */
2  #include "stdio.h"
3
4  main ()
5   {
6      char phrase[81];
7      char rev_phrase[81];
8
9      printf ("type a phrase : \n") ; /* prompt for phrase from keybd */
10     getstr (phrase, 80) ;           /* and get it */
11
12     do_reverse (phrase, rev_phrase) ; /* construct reversed phrase */
13     if (strcmp (phrase, rev_phrase) ==0) /* compare original */
14           printf ("** palindrome **") ;
15     else  printf ("reverse=%s", rev_phrase) ;
16 }
17
18
19 int do_reverse (s1, s2)     /* reverse a string */
20  char s1[];                 /* the string to be.reversed */
21  char s2[];                 /* place its reverse here */
22  {
23   int i, j;
24                        /* copy characters starting at end of s1 */
25   i=0;
26   j=strlen (s1) -1; /* find index of last character in s1 */
27   while (j>=0)
28     { s2[i]=s1[j];
29       i=i+1;
30       j=j-1;
31     }
32   s2[i]=0;          /* mark end of string */
33 }
34
```

getstr接收的字，按照C提供的格式在字符数组中存储：字符连续存放，以\0作结尾。在以前的程序中我们曾经使用过字符串常量，C编译器以同一格式进行存储。例如，HELLO串的存储情况如下：

位置	0	1	2	3	4	5	6
字符值	h	e	l	l	o	,	\0

标记\0表示值0（或null），这同\n表示换行控制字符是一样的。

第19行到33行的do_reverse函数对字求反，将结果存于指定数组中。函数中使用了库函数strlen求串长度，strlen返回串的字符个数，但不包括结尾标记\0。由于C语言中数组下标从0开始，因此，对于n个字符的串，其最后一个字符的下标是n－1。