北京科海培训中心系列教材

C语言习题精解

王鹰翔　王念军　编

宇航出版社

C 语言习题精解

王鹰翔　王念军　等编

宇航出版社

内 容 简 介

本书精选C语言习题及其解答100余道，编排中考虑了题目的独立性和完整性，可与有关C语言的书籍配合使用，也可以单独使用。

本书的编排共分八章。第一章至第七章的全部习题解答均使用Turbo C（1.5以上版本）已在80286、IBM-PC及其兼容机上调试通过；第八章“与UNIX系统的接口”中的程序使用SCO XENIX(2.2.3)，在80386上编译调试通过。

对习题中的一题多解情况，尽量给出两种以上的解答，习题中用到的函数通常附在题后。此外，在书后的附录中列出了书中用到的函数，这些函数被编入一个库中。

本书可供大、中专院校计算机专业的师生用做教学参考书，也可作为广大计算机工作者和工程技术人员学习C语言的参考书，还可以用于参加全国计算机软件水平考试和全国自学考试人员的学习辅导材料。

书中的全部习题和库函数已存入两张软盘，需要者可与北京科海培训中心或宇航出版社软件部联系。

C语言习题精解

王鹰翔　王念军　等编

责任编辑：赖巧玲

*

宇航出版社出版发行

北京和平里滨河路1号　邮编：100013

各地新华书店经销

河北蔚县印刷厂印刷

*

开本：787×1092　1/16　印张：11.875　字数：300千字

1992年5月第一版　1992年5月第一次印刷

印数：0001—7000

ISBN 7-80034-325-1/TP·030

定　价：6.85元

前　言

近年来，C 这种兼有高级语言和汇编语言优点、功能很强但又十分简洁的程序设计语言，正日益得到计算机界的赏识。随着 UNIX 操作系统和微型机、小型机的推广应用，学习计算机程序设计语言时，C 语言几乎成了必不可少的一种，C 语言的应用程序也遍地开了花。

当前，B. W. Kernighan 和 D. M. Ritchie 合著的《C 程序设计语言》一书，已经成为了解和掌握 C 语言的启蒙著作。国内许多专家学者直接翻译了 K&R 这本书，或以其为蓝本向广大读者介绍了 C 语言。这些书中都依照 K&R 原著的模式，在每一章后面附有一些习题，以帮助读者掌握各章内容。遗憾的是，对这些习题没有一套完整的答案，读者做完习题后无法判断自己做的是否正确，是否完全掌握了这一章的内容。这一情况给读者，尤其是自学者带来了不便。

为了解决这个问题，我们抱着普及、推广 C 语言的热情，对《C 程序设计语言》一书中的全部习题给出了解答，并在机器上调试通过。书中的习题曾经 B. W. Kernighan 审阅过。

本书曾在清华大学出版社以《C 语言习题与解答》出版过，这次对原书进行了修正和增补。在编排中考虑了题目的独立性和完整性，所以可以单独使用或与其它有关的 C 语言书籍配合使用。既可作为大、中专院校计算机专业的教学参考书，也可以作为广大计算机工作者和工程技术人员学习 C 语言的参考书，还可以用做参加全国计算机软件水平考试和自学考试人员的学习辅导材料，是学习 C 语言的必备参考手册。

本书首次出版时，王鹰翔、王念军、孙钢、杨松涛等同志参加了编写和程序调试工作，科学院软件所白光野同志审校过全书。这次由王鹰翔、王念军进行了修订。北京科海培训中心对本书的出版给予了极大的支持，在此一并表示感谢。

由于出版工作紧迫，加之水平有限，对书中存在不足之处，望读者斧正、赐教。

编　者

一九九二年三月

目　录

此为试读，需要完整PDF请访问：www.ertongbook.com

第一章　C语言概貌

习题 1

请在你的系统上运行下面的程序，试将程序中某些部分去掉，看看会出现什么错误信息。

解答：

```
main()
{
    printf("hello,World");
}
```

本例中丢失了换行符号(\n)，这样将引起输出时后面没有换行(即新的一行)。

```
main()
{
    printf("hello,World\n")
}
```

第二个例子中，语句 printf()后面丢失了分号“;”。每个C语言的语句都必须以分号结束，否则C语言的编译程序将认为本句出错，并打印出相应的出错信息。

```
main()
{
    printf("hello,World\n');
}
```

第三个例子中，\n 后面的双引号丢失，而接在\n 后面的单引号、右半圆括号和分号“;”被当做是整个字符串的一部分。C的编译程序认为本句出错，将提示双引号丢失或字符串太长等出错信息。

习题 2 •

试看当函数 printf 的变量串中包括\x 时，会发生什么情况。此处的 x 是前面没有列出的某个字符。

解答：

```
main()
{
    printf("hello,World\a");
    printf("hello,World\7");
    printf("hello,World\?");
}
```

在C语言参考手册附录A中提到：

如果跟在反斜线“\”后面的字符不是被指定的字符，则反斜线“\”无效。

由 Dennis Richie 编著的“C程序设计语言”的语言参考手册(1983年6月版)中指出：

"如果接在反斜线号'\'后面的字符不是那些指定的字符,则状态无法确定"。

因此,本例中的结果与用户所用的C的编译系统有关,其中一种可能的输出结果是:

hello,worldahello,world(BELL)hello,world?

这里(BELL)指由ASCII码7所引起的短促鸣笛声。

反斜线'\'后面跟着的1至3位八进制数都可以表示一个字符,所以\7和\007都可以引起短促的鸣笛声。

习题3

修改温度转换程序,使之能在该表上打印出表头。

解答:

```
main()   /* Fahrenheit-Celsius talbe for fahr=0,20,…,300 */
{
    int lower, upper, step;
    float fahr, celsius;
    lower=0;        /* lower limit of the temperature table */
    upper=300;      /* upper limit                          */
    step=20;        /* step size                            */
    printf("Fahr Celsius\n");
    fahr=lower;
    while(fahr <= upper)   {
        celsius=(5.0/9.0) * (fahr-32.0);
        printf("%4.0f %6.1f\n",fahr,celsius);
        fahr=fahr+step;
    }
}
```

本解答在原温度转换程序的循环体之前,增加了语句:

```
printf("Fahr Celsius\n");
```

在循环体前加入这条语句后,在表中相应的某列上面产生一个表头。程序的其余部分与原温度转换程序相同。

附:原温度转换程序

```
main()   /* Fahrenheit-Celsius table for fahr=0,20,…300 */
{
    int lower, upper, step;
    float fahr, celsius;
    lower=0;        /* lower limit of the temperature table */
    upper=300;      /* upper limit                          */
    step=20;        /* step size                            */
    fahr=lower;
    while (fahr <= upper)   {
        celsius=(5.0/9.0) * (fahr-32.0);
        printf("%4.0f %6.1f\n", fahr, celsius);
        fahr=fahr+step;
```

```
    }
}
```

习题 4

编写一个能打印出由摄氏温度转换到华氏温度的对照表。华氏温度(用 F 表示)与摄氏温度(用 C 表示)有如下关系:

C=(5/9)(F−32)

解答:

```
main()   /* Celsius−Fahrenheit table for celsius=0,20,…,300 */
{
    int lower, upper, step;
    float fahr, celsius;
    lower=0;        /* lower limit of the temperature table */
    upper=300;      /* upper limit                          */
    step=20;        /* step size                            */
    printf("Celsius Fahr\n");
    celsius=lower;
    while (celsius <= upper)      {
        fahr=(9.0*celsius) / 5.0+32.0;
        printf("%4.0f %6.1f\n", celsius, fahr);
        celsius=celsius+step;
    }
}
```

本程序产生一个包括摄氏(Celsius)温度(0—300 度)和对应的华氏(Fahrenheit)温度的对照表。其中华氏温度是通过下列语句,由已知的摄氏温度换算出来的:

```
fahr=(9.0*celsius)/5.0+32.0;
```

本程序在逻辑上与《语言》第一章中打印华氏—摄氏对照表的转换程序相同。其中整型变量 lower、upper 和 step 分别代表最低温度、最高温度和摄氏温度变量 celsius 的变化步长。变量 celsius 在初始时赋值为最低温度(即温度下限),在 while 循环中,计算出与 celsius 对应的华氏(Fahrenheit)温度值,然后打印出摄氏温度(Celsius)和对应的华氏温度(Fahrenheit)。变量 celsius 以步长 step 增加。在变量 celsius 超过它的最高温度(温度上限)时,while 循环结束。

附:Fahrenheit—Celsius 的温度转换程序

```
main()   /* Fahrenheit−Celsius table for fahr=0,20,…,300 */
{
    int lower, upper, step;
    float fahr, celsius;
    lower=0;        /* lower limit of the temperature table      */
    upper=300;      /* upper limit                               */
    step=20;        /* step size                                 */
    fahr=lower;
    while (fahr <= upper) {
        celsius=(5.0/9.0)*(fahr−32.0);
```

```
        printf("%4.0f %6.1f\n", fahr, celsius);
        fahr=fahr+step;
    }
}
```

习题 5

修改 Fahrenheit—Celsius 温度转换程序，以反序打印对照表，即从 300 度开始打印，打印到 0 度为止。

解答：

```
main()                          /* Fahrenheit—Celsius table for fahr=300,280,…,0 */
{
    int fahr;
    for(fahr=300; fahr>=0; fahr=fahr-20)
        prinft("%4d %6.1f\n",fahr,(5.0/9.0* (fahr-32));
}
```

程序中只修改了一条语句：

```
for(fahr=300; fahr>=0; fahr=fahr-20)
```

for 循环语句第一部分：

```
fahr=300;
```

把华氏温度(Fahrenheit)变量 fahr 的初始值置为它的上限最高温度。第二部分，即控制 for 循环的条件控制语句：

```
fahr >=0
```

检查华氏温度(Fahrenheit)变量 fahr 是否超过了或是否达到了它的下限最低温度。只要这个条件语句为真，for 循环就继续执行。

语句中第三部分，是步长重置语句：

```
fahr=fahr-20
```

此语句使华氏温度变量以它的步长 20 递减。

习题 6

编写一个程序，对空格、制表符和换行符号进行计数。

解答：

```
#include<stdio.h>
main()   /* count blanks, tabls, and newlines */
{
    int c, nb, nt, nl;
    nb=0;          /* number of blacks      */
    nt=0;          /* number of tabs        */
    nl=0;          /* number of newlines    */
    while ((c=getchar()) != EOF) {
        if(c==' ')
```

```
            ++nb;
        if(c=='\t')
            ++nt;
        if(c=='\n')
            ++nl;
    }
    printf("%d %d %d\n", nb, nt, nl);
}
```

本程序中的整型变量 nb、nt 和 nl 分别用于对空格、制表符和换行符号的个数进行计数。在初始条件时，这三个变量都赋值为 0。

在 while 循环体中，把从输入字符中取到的每个空格、横向制表符和换行符号都记录下来。在整个循环中，每一次，对循环中所有的 if 语句全都执行一遍。如果取回的字符不是空格、制表符或换行符号，计数器就不计数；如果取回的字符是这三种符号中的某一类，对应的计数器就加 1。最后，当出现了文件尾标记 EOF 时，程序把最终结果打印输出。

在此之前，没有讲到 if—else 语句，所以本程序中没有使用 if—else 语句。如果使用 if—else 语句，程序改写如下：

```
#include<stdio.h>
main()              /* count blanks, tabls, and newlines */
{
    int  c, nb, nt, nl;
    nb=0;           /* number of blacks          */
    nt=0;           /* number of tabs            */
    nl=0;           /* number of newlines        */
    while ((c=getchar()) != EOF) {
            if (c==' ')
                ++nb;
            else if (c=='\t')
                ++nt;
            else if (c=='\n')
                ++nl;
    }
    printf("%d %d %d\n", nb, nt, nl);
}
```

习题 7

编写程序，证明 getchar( ) != EOF 的结果是 0 或 1。

解答：

```
#include<stdio.h>
main()
{
  int c;
  while (c=getchar() != EOF)
      printf("%d\n", c);
```

```
        printf("%d—at EOF\n", c);
    }
```

在解答中，表达式 c=getchar( ) !=EOF 等价于表达式 c=(getchar( ) !=EOF)，程序用上述表达式从标准输入中读入字符，当 getchar 检测到文件结束标志 EOF 前，一直读入字符，此时表达式 getchar( ) !=EOF 为真，于是 c 被赋值 1。反之，当程序检测到文件结束时，则上述表达式为假，c 被赋值 0，循环终止。

习题 8

编写程序，打印 EOF 的值。

解答：

```
#include<stdio.h>
main()
{
    printf("EOF is %d\n", EOF);
}
```

符号常量 EOF 在<stdio.h>中定义，在函数 printf 中，位于双引号外的 EOF 将由跟在 #define EOF 后的值所替代。在这里，EOF 定义为－1，但在不同的系统中，EOF 的值定义也不同。采用 EOF 这样的符号常量，是为了更好地进行程序移植。

习题 9

编写一个程序，把输入复制到输出，并用单个空格来代替一个以上的空格所组成的字符串。

解答：

```
#include <stdio.h>
#define NONBLANK      'a'
main()              /* replace string of blanks with a single blank */
{
    int c, lastc;
    lastc=NONBLANK;
    while ((c=getchar()) != EOF)  {
        if (c != ' ')
            putchar(c);
        if (c==' ')
            if (lastc != ' ')
                putchar(c);
        lastc=c;
    }
}
```

本程序在输入的字符中查找空格符。每当找到一个空格符后，程序就测试紧接在这个空格之后的那个字符是否还是空格。如果后面的字符还是空格符，这空格就被省略掉；如果后面的

此为试读，需要完整PDF请访问：www.ertongbook.com

字符不是空格符，就打印输出这个空格符。这样，程序保证所有的单个的空格都被打印输出，而对由两个以上的空格组成的字符串，只打印输出第一个空格符。

程序中的整型变量 c 用于记录从输入取来的当前字符的 ASCII 码值。另一个变量 lastc 用于记录从输入取来的当前字符之前一个字符的 ASCII 码值。符号常量 NONBLANK 在初始化时，给变量 lastc 赋一个任意的非空格字符，在本程序中，用的是 'a'。

程序中 while 循环体中的第一个 if 语句用于处理所遇到的非空格字符，并将这些字符打印输出；第二个 if 语句处理空格；而第三个 if 语句，则测试一个单个的空格符或者是由两个以上的空格组成的字符串中头一个空格。最后，变量 lastc 被更新。这个处理过程反复进行下去。

本题目之前尚未讲到 if－else 语句，如果使用 if－else 语句，解答如下：

```
#include <stdio.h>
#define   NONBLANK      'a'
main()              /* replace string of blanks with a single blank */
{
    int c, lastc;
    lastc=NONBLANK;
    while ((c=getchar()) != EOF)    {
        if (c != ' ')
            putchar(c);
        else if (lastc != ' ')
            putchar(c);
        lastc=c;
    }
}
```

另外，本节之前，也没有讲到“逻辑或”(||)操作符，如果使用“逻辑或”，解答可改写如下：

```
#include <stdio.h>
#define NONBLANK      'a'
main()              /* replace string of blanks with a single blank */
{
    int c, lastc;
    lastc=NONBLANK;
    while ((c=getchar()) != EOF) {
        if (c != ' ' || lastc != ' ')
            putchar(c);
            lastc= c;
    }
}
```

习题 10

编写一个程序，把输入信息直接复制到输出，在复制过程中，用\t 符号代替制表符，用\b 符号代替退格符，用\\符号代替右斜线，使这些符号成为可见符号。

解答：

```
#include<stdio.h>
main()              /* replace tabs and backspaces with visible char */
```

```
{
    int c;
    while ((c=getchar()) ! = EOF)
    {
        if (c==='\t')
            printf("\\t")
        if (c==='\b')
            prinft("\\b");
        if (c==='\\')
            printf("\\\\");
        if (c ! ='\b')
            if (c ! ='\t')
              if (c ! ='\\')
                putchar(c)
    }
}
```

由于从输入中获得的字符可以是包括制表符、退格符、右斜线在内的任意符号，若是这三种，分别以\t、\b 和\\进行替换，对其他字符照原样输出。在 C 语言中，右斜线表示为'\\'，如要用 printf 函数将其打印出来，需要写成"\\\\"形式的字符串。如果使用 if—else 语句，解答可改写如下：

解答：

```
#include<stdio.h>
main()              /* replace tabs and backspaces with visible char */
{
    int c;
    while ((c=getchar())! = EOF) {
        if (c==='\t')
            printf("\\t")
        else if (c==='\b')
            prinft("\\b");
        else if (c==='\\')
            printf("\\\\");
        else
            putchar(c);
    }
}
```

习题 11

编写一个程序，用三个字符的序列，如>、退格和－，打印出→符号来代替制表符 tab。用相似的序列打印出←来代替退格字符 backspace。这样，就使制表符 tab 和退格字符 backspace 成为可以看见的。

解答：

```
#include <stdio.h>
```

```
main()              /* replace tabs and backspaces with a 3 char sequence */
{
    int c;
    while ((c=getchar())!= EOF) {
        if (c=='\t')
            printf("->\b>");
        if (c=='\b')
            printf("-\b<");
        if (c!='\b')
            if (c!='\t')
              putchar (c);
    }
}
```

用 while 循环体测试三种可能发生的条件。从输入中取出的字符可能是一个横向制表符 tab，或者是一个退格符 backspace，或者是其它任意一种字符。

每当检测出 tab 或 backspace 时，对应的三字符序列(→或是←)就被打印输出。如果取得的字符既不是制表符 tab，也不是退格符 backspace，那么就打印输出这个字符本身。在遇到文件结束标志 EOF 之前，while 循环反复进行。

在 CRT 显示屏幕上，无法重叠显示字符。这样，"-\b>"的显示结果为>，而">\b-"的显示结果为-(即后一字符覆盖了前一字符)；所以我们建议，在使用 CRT 终端的情况下，最好使用序列-\b>和-\b<来进行工作，这样在屏幕上至少还能保留一个箭头指示符。

在本章前面还没有讲到 if—else 语句。如果使用 if—else 语句编写程序，解答可改写如下：

```
#includs <stdio. h>
main ()             \* replace tabs and backspaces with a 3 char sequence */
{
    int c;
    while ((c=getchar()) != EOF) {
        if (c=='\t')
            printf("->\b>");
        else if (c=='\b');
            prinft("-\b<");
        else
            putchar(c);
    }
}
```

习题 12

怎样测试字计数程序？有些什么边界条件？

解答：

为了检测字计数程序，首先测试无输入状态，这时，输出结果应该是：000(意即：零个换行，零个字，零个字符)。

然后，测试包含一个字符的字，这时输出结果应该是：112(即：一个换行，一个字，两个字符——一个字符后面接着一个换行符号)。

再测试包含两个字符的字，那么输出结果应该是：113(即：一个换行，一个字，三个字符——两个字符后面接着一个换行符号)。此外，还可以再测试一些其它的字。如：试两个字，每个字包含一个字符(输出结果是：124)；试两个字，每个字包含一个字符，但每个字分别各在一行中(输出结果应是：224)。

题目中问到的某些边界条件如下所述，这些都是可能发现出错的测试条件：

- 无输入状态。
- 仅有换行符号，但无字输入。
- 无字输入，仅有空格符、tab 制表符和换行符号。
- 每行一个字，但没有空格符和制表符 tab。
- 输入的字从每一行最前面开始出现。
- 在输入的字前面，出现一些空格符，等等。

附：单词(字)计数程序

```
#include <stdio.h>
#define   YES       1
#define   NO        0
main()              /* count lines, words, chars in input */
{
    int c, nl, nw, nc, inword;
    inword=NO;
    nl=nw=nc=0;
    while ((c=getchar()) != EOF) {
        ++nc;
        if (c=='\n')
            ++nl;
        if (c==' ' || c=='\n' || c=='\t')
            inword=NO;
        else if (inword==NO) {
            if ((c>='a' && c<='z') || (c>='A' && c<='Z')) {
              inword = YES;
              ++nw;
            }
        }
            else if ((c>='a' && c<='z') ||
               (c>='A' && c<='Z') ||
               (c>='0' && c<='9') ||
                c=='\'');
            else
              inword=NO;
    }
    printf("%d %d %d\n",nl,nw,nc);
}
```

习题 13

编写一个程序，打印输入的“单词”，每行只打印一个“单词”。

解答：

```
#include <stdio.h>
#define   YES          1
#define   NO           0
main()              /* print words one per line */
{
    int c, inword;
    inword=NO;
    while ((c=getchar()) != EOF)    {
        if (c==' ' || c=='\n' || c=='\t') {
            if (inword==YES) {
                putchar('\n');              /* finish the word */
                inword=NO;
              }
        } else if (inword==NO)  {
            inword=YES;             /* beginning of word */
            putchar(c);
        } else                  /* within a word */
            putchar (c);
    }
}
```

程序中的变量 inword 是整型布尔变量，这个变量用于记录程序当前是否输入了一个单词。因为我们不知道要输出的单词从哪里开始，所以在程序的开头，变量 inword 被赋初始值为 NO。

程序中第一个 if 语句

```
if (c==' ' || c=='\n' || c=='\t')
```

用于检测变量 c 中是不是一个单词分隔符。如果 c 中是一个单词分隔符，这时第二个 if 语句

```
if (inword ==YES)
```

开始检测这个单词分隔符是否标志着单词的结尾。如果 c 中分隔符是一个单词的结尾，程序就打印一个换行符号并更新变量 inword；否则，不执行任何动作，程序继续往下执行。

如果变量 c 不是一个单词分隔符，那它就是一个单词的第一个字符或是组成这个单词的其它字母。当 c 是一个单词中第一个字符时，变量 inword 中的内容就被更新。但不论是哪种情况，都把这个字符打印出来。

习题 14

使用对“单词”的改进定义，修改单词计数程序。

例如定义可以是：单词是以字母开始的，由字母、数字和省略号组成的字符串，等等。

解答：

```
#include <stdio.h>
```

```
#define   YES          1
#define   NO           0
main()                 /* count lines, words, chars in input */
{                      /* ASCII only                         */
    int c, nl, nw, nc, inword;
    inword=NO;
    nl=nw=nc=0;
    while ((c=getchar()) != EOF)   {
        ++nc;
        if (c=='\n')
          ++nl;
        if (c==' ' || c=='\n' || c=='\t')
            inword = NO ;
        else if (inword==NO) {
        if ((c >='a'   &&   c<='z') || (c>='A'&& c<='Z')) {
                                   /* lower case ? or upper case ? */
            inword=YES;
            ++nw
        }
} else if ((c>='a'  &&   c<='z') ||      /* lower case? */
        (c>='A'   &&   c<='Z') ||        /* or upper case? */
        (c>='0'   &&   c<='9') ||        /* or digit? */
        c=='\'')                         /* or apostrophe? */
        ;                                /* or within a word? */
    else
            inword = NO;
    }
        printf("%d %d %d\n", nl, nw, nc);
}
```

使用针对“单词”的改进定义，需要在程序中再增加三条语句。第一条语句是：

```
if ((c>='a'  &&  c<='z' || (c>='A'  &&  c<='Z'))
```

用于检测某个单词的第一个字符是大写字符还是小写字符(仅仅是 ASCII 码)，如果第一个字符是大写字母，就是一个新词的开始，于是就修改变量 inword 的内容，同时单词计数器加 1。

第二条增加的是测试语句：

```
else if ((c>='a' && c<='z') ||
        (c>='A'  &&  c<='Z') ||
        (c>='0'  &&  c<='9') ||
        c=='\'')
              ;
```

这样，只有在程序当前输入了一个单词并找到一个字符，而且这个字符既不是空格符，也不是换行符号或制表符 tab 时，这个测试语句才发生作用。这条测试语句保证了字(单词)是由字母、数字和省略号所组成的序列(仅仅是 ASCII 码)。

新加的最后一条语句是：

```
else
```


此为试读，需要完整PDF请访问：www.ertongbook.com