北京科海
培训中心系列教材

C语言习题与解答

C语言习题与解答

王鹰翔 王念军 等 编译

清华大学出版社

北京科海培训中心系列教材

C语言习题与解答

王鹰翔　王念军　等　编译

清华大学出版社

内 容 简 介

本书为C语言习题及其解答，在编排中考虑了题目的独立性和完整性，可与有关C语言的书籍相配合使用。

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版)在IBM-PC/XT和80286机器上编译调试通过。

第八章“与UNIX系统的接口”中的程序使用XCO的XENIX（2.2.3版）在80386机器上编译调试通过。

对一题多解情况，尽量给出两种或两种以上答案，习题中用到的函数，通常附在题后。此外，在书后附录中，列出了全书中用到的函数，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作教学参考书，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。

书中的全部习题和库函数已存入两张软盘，需要者请与北京科海培训中心或与清华大学出版社软件部联系。

C语言习题与解答
王鹰翔 王念军 等编译
☆
清华大学出版社出版
北京 清华园
河北省蓟县印刷厂排版
北京市昌平印刷厂 印装
新华书店总店科技发行所发行
☆
开本：787×1092 1/16 印张：10.75 字数：252千字
1990年4月第1版 1991年7月第2次印刷
印数：5001—15000
ISBN 7-302-00682-2/TP·233
定价：6.50元

前　言

近年来，C这种兼有高级语言和汇编语言优点的、功能很强而又十分简洁的程序设计语言，正日益得到广大计算机界的赏识。随着UNIX操作系统和微型机、小型机的推广应用，C语言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的原著模式，在每一章节的后面附有一些习题，以帮助读者掌握各章的内容。遗憾的是，对这些习题没有一套较完整的答案，读者做完习题后无法判断自己做得是否正确，是否完全掌握了这一章的内容。这一情况给读者，尤其是给自学者带来了不便。

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予解答。全部解答均由B.W.Kernighan亲自审阅过。

本书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C语言的书籍相配合，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，是学习C语言的必备参考手册。

中国工商银行总行王鹰翔、机电部经济信息中心王念军和商业部计算中心杨松涛参加了本书的编译工作。商业部计算中心孙钢同志校对了初稿，并参加了程序调试工作。科学院软件所白光野同志审校了全书。软件所汤晓丹同志在本书的编写和出版过程中，做了许多有益的工作，在此一并表示感谢。

特别要感谢科海培训中心对本书的出版给予的极大支持。

对书中所存不足之处，望读者不吝赐教。

编者

一九八九年十二月

目　录

第一章　C语言概貌

练习1-1

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。

```
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“；”。因为每个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。

练习 1-2

试看当printf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。

```
main()
{
```

```
    printf("hello, world\a");
    printf("hello, wordl 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。

练习1-3

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 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                            */
    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，在表中相对某列上面就产生一个表头。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                            */
    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;
    }
}
```

练习 1-4

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;
```

本程序在逻辑上与原著第一章中打印华氏——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变化的步长。变量celsius在初始时赋值为最低温度（即最小值），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超过它的上限（最高温度）之前，while循环一直进行下去。

附：Fahrenheit——Celsius的温度转换程序

```
main()    /* Fahrenheit-Celsius table for fahr = 0, 20, ···, 300 */
 {
    int lower, upper, step;
    floac fahr, celsius;
    lower = 0;          /* lower limit of the temperature table */
    upper = 300;        /* upper limit                          */
    scep = 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;
    }
}
```

练习 1-5

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)
        printf("%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 循环语句中第一部分：

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）变量是否超过了或是否符合它的下限最低温度。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

语句使华氏温度变量以它的步长20递减。

练习 1-6

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, tabs, and newlines */
 {
    int c, nb, nt, nl;
    nb = 0;                                  /* number of blanks    */
    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、制表符和换行符号进行计数。在初始条件，这三个变量中都赋值为0。

在while循环体中，把从输入字符中取出的每个空格、横向制表符和换行符号都记录下来。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, tabs, and newlines */
 {
    int c, nb, nt, nl;
    nb = 0;                                     /* number of blanks     */
    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);
}
```

练习 1-7

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。每当查找到一个空格符后，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符。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号。

程序中的整形变量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循环的循环体中第一个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;
        }
    }
```

练习 1-8

编写一个程序，用三个字符序列>、退格、-，打印出→符号来代替各个制表符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成为可以看见的了。

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);
    }
}
```

有三种可能发生的条件，在while循环体中检测这三种条件。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，对应于这个字符的三字符序列（→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。在遇到文件结束标记EOF之前，while循环反复进行。

在CRT显示屏幕上，无法重叠显示打印。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：

```
#include <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');
            printf("- \b<");
        else
            putchar(c);
    }
}
```

练习1-9

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

再试验一个包含两个字符的字，那么输出结果应当是：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，还可以再试验一些其它的字。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，

- 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。
- 在输入字之前，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) ;
}
```

练习1-10

编写一个程序，打印输入的字，每行只打印一个字。

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字。因为我们不知道要输出的字从哪里开始，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中是不是一个字分隔符。如果c中是一个字分隔符，这时第二个if语句

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

开始检测这个字分隔符是否标志着字的结尾。如果c是一个字的结尾分隔符，程序就打印一个换行符号；否则，不执行任何动作，程序继续往下执行。

如果变量c不是一个字分隔符，那它就是一个字的第一个字符或是组成某个字的其它字母。当c是一个字中第一个字符时，变量inword中的内容就被修改。但不论是哪种情况，都把这个字符打印出来。

练习1-11

使用对“单词”的改进定义，修改字计数程序。

例如定义可以是：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;
        else 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? */
                ;                                  /* within a word  */
        else
            inword = NO;
    }
    printf ("%d %d %d\n", nl, nw, nc) ;
}
```

使用关于“单词”的改进的定义，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'))
```

用于检测某个字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中的内容，同时字计数器加 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字和一个字符，而这个字符不是空格符，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）是由一串字母、数字和省略号所组成（仅仅是ASCII码）。

新加的最后一条语句是：

```
else
    inword = NO;
```

因为除了空格、换行符号和制表符tab之外，还增加了非字字符，所以这条语句也是必不可少的。如果遇到了那些附加字符中的某个字符，程序就修改变量inword中的内容，