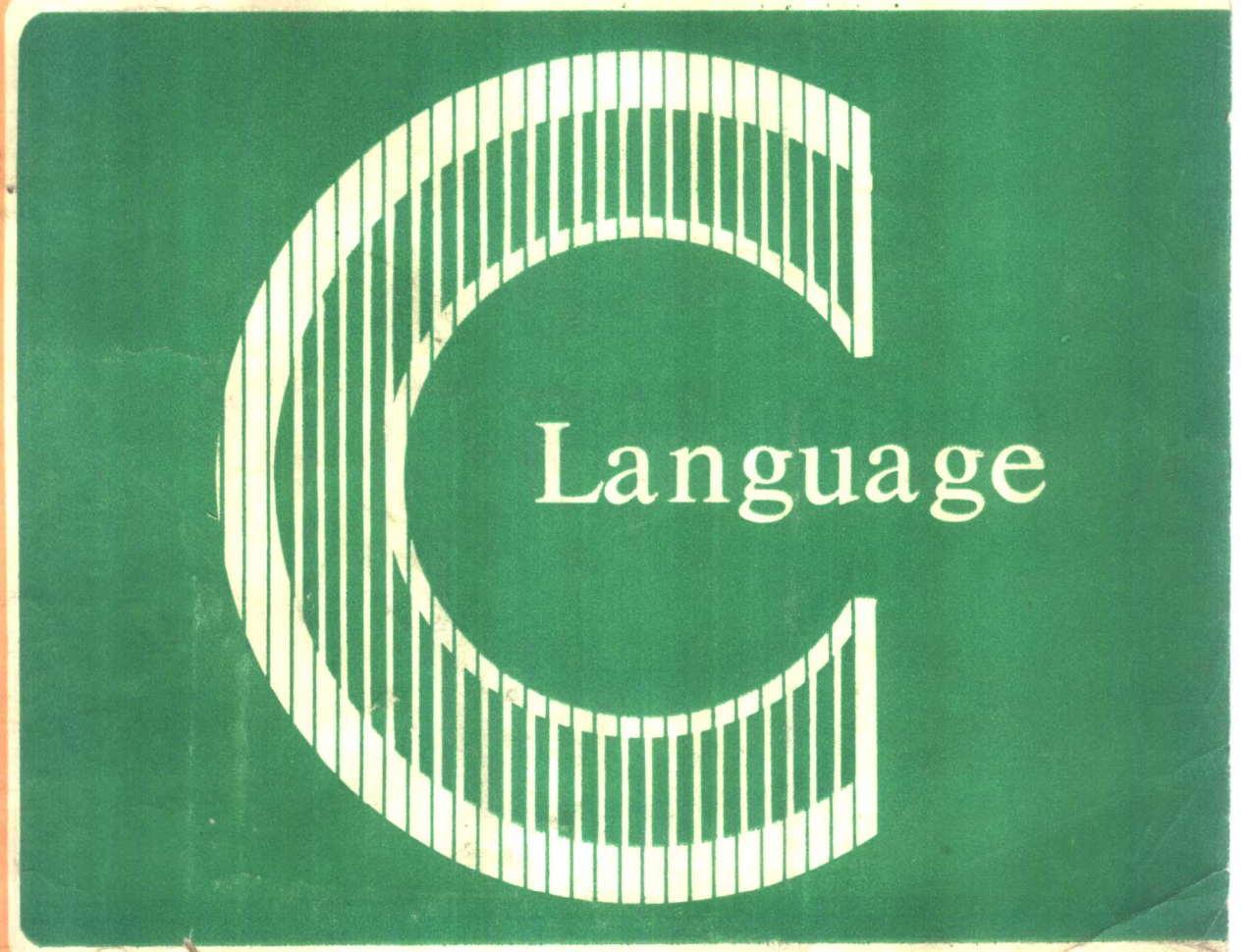
C语言解答

刘乃琦 译　　俸远祯 校



成都电讯工程学院出版社

C 语言解答

刘乃琦译

隽远祯校

成都电讯工程学院出版社

·1987·

C语言解答

刘乃琦 译

傅远祯 校

*

成都电讯工程学院出版社出版

成都电讯工程学院出版社印刷厂印刷

四川省新华书店经销

*

开本 787×1092 1/16 印张 7.875 字数 195千字

版次 1987年10月第一版 印次 1987年10月第一次印刷

印数 1—8,800册

中国标准书号：ISBN 7—81016—033—8/TP·4

统一书号：15452·37 定价 1.90元

译者的话

本书主要针对由 B•W• kernighan 和 D•M• Ritchie (简称 K&R)所写著名的《C 语言》(The C Programming Language)一书，提出的全部问题和习题的详细解答及程序说明。这对学习和使用 C 语言的人来说无疑是一个极大的帮助。

随着 C 语言以及 UNIX 操作系统的日益普及和发展，学习和使用 C 语言的人越来越多。目前，几乎每一种计算机，尤其是使用最广的微型计算机系统上都配备了 C 语言的编译系统。加之 C 语言的结构严谨灵活，功能强而简单，并兼有高级语言和汇编语言两者的许多优点和特点，如可移植性好，所占空间少，执行速度快等等很受计算机工作者的喜爱。它特别适宜用来编写各种软件、系统程序以及操作系统等。

本书的顺序按照 K & R 所著《C 语言》一书的顺序进度安排，对每一道问题及习题作了详细的解释说明。阅读本书是假定读者已经有了一定的 C 语言的基础，或已经读过了 K & R 原著或译本(成都电讯工程学院出版社出版的李智渊等编译的“C 语言”一书)，并做了相应的练习和习题。这样，读者可以仔细阅读本书，分析所给的程序，并上机实践，这将大大有助于对 C 语言的理解和掌握，并获得良好的 C 语言编程技巧。尤其对那些通过自学来掌握 C 语言的读者更有极大的帮助。

本书译自 C•L• Tondo 和 S•E• Gimpel 所著的“The C Answer Book”，编译时在顺序和结构上稍有变动，对书中的程序上机进行了验证，改正了某些印刷错误。当然，读者可以直接在自己的系统上验证和执行，也可以参照这些程序，利用自己的知识，写出更好的解答。

本书的形成得到成都电讯工程学院计算机系俸远桢副教授，刘心松副教授的热情关怀和支持，俸远桢副教授仔细审阅了全书及程序，提出了许多宝贵的意见和建议，在此表示衷心的感谢。

由于时间和译者水平所限，书中存在的不足之处恳请批评指正。读者在使用本书时所遇到的任何问题，可直接与译者联系。

译　者

一九八七年二月

原书序言

本书针对B•W•Kernighan和D•M•Ritchie所著“The C Programming Language”一书的全部问题和练习作了解答。读者若配合K & R的原书并仔细阅读本书，将大大有助于对C语言的理解和掌握，并获得良好的C语言编程技巧。

最好先学习K & R的《C语言》一书，逐章做练习，然后再研究本书提供的详细解答。本书所用的语言结构与K & R书中的C语言结构一样，并且遵循了同样的进度。读者自己学习和掌握了更多的C语言结构后，就可以自己写出更好的答案。

例如：对于

```
if（表达式）
    语句 1
else 语句 2
```

形式的语言结构，在原书没有讲到之前，我们在习题中没有采用。读者如已掌握可以自行采用，到时，我们也给出相应解答。

又如，原书谈到符号常数，当解答中用到这些标准符号常数时，（如文件结束符EOF），我们采用了：

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

形式。这里，文件stdio.h包含了标准输入输出库程序所用的宏调用与符号常数，这个文件的细节读者可参考原书。

本书解答备有详细解释说明，假定读者已读过K & R的原书并作了练习，本书不重复原书讲过的部分，只对解答的要点给予讲述。

要学习一种程序设计语言，只阅读语言结构是不行的，需要实际的编程——自己动手写出代码，以及学习研究别人所写的程序。本书中的程序，语言结构性良好，代码模块化，并扩展了库程序的使用，读者可以了解整个逻辑过程。

在此，谨向C语言开发的先驱：B•W•Kernighan和D•M•Ritchie表示衷心感谢，并以他们的精神和风格作为我们今后工作的楷模。特别感谢的是B•W•Kernighan仔细审阅了本书，给予了极大帮助。

另外，对负责本书编辑的K•V•Karlstrom，精心审阅手稿并指出遗漏的史坦福大学的R•W•Haddad，打印整理文稿的W•Brainerd，以及给我们以无私帮助的P•M•Adams，P•A•Gates和S•L•Mackey等其他朋友们的友谊，支持和帮助表示深切感谢。

C•L•Tondo

S•E•Gimpel

目　录

第一章　C语言概述

习题1—1

在用户自己的系统上运行所给的程序，在实际操作时注意程序中有意省略掉的部分，看看能得到什么样的错误信息。

第一个例子

```
main ( )
{
    Printf("hello, world");
}
```

在这个例子中，有意省去了换行符(\n)，结果在输出数据以后，没有换行操作。

第二个例子

```
main ( )
{
     Printf("hello, world \n")
}
```

这里，在Printf ( )函数后面有意漏掉了分号“;”，而C语言的语句是必须以；号作为终止符的，所以这个程序在提交编译时，编译程序会以某种方式给出错误信息。

第三个例子

```
main ( )
{
     Printf('hello, world \n');
}
```

这里，换行符\n后的双引号错写成了单引号，这样，单引号与右括号和分号一起被认为是某个字符串中的一部分。当然，编译程序会发现这个错误，并给出出错信息，比如“括号丢失或者字符串过长”等等。

习题1—2

做做实验，看看Printf的参数串在含有\x时会出现什么情况，这里x代表书中未列出的某一字符。

实验程序如下；

```
main ( )
{
   Printf("hello , world\a");
   Printf("hello, world\7");
   Printf("hello, world\? ");
```

```
}
```

C语言的参考手册指出：如果在右斜杠\后所跟的字符不是C语言所定义的字符，则这个右斜杠被忽略掉。

另外，在由Dennis Ritchie所著的《C语言参考手册》(1983年7月出版)，对此种规定有所改动，即：如果右斜杠后所跟的字符不是所定义的字符，则结果情况是不定的。

也就是说，这个实验的结果是与用户所采用的C语言的编译程序有关，对上述程序，一种可能的结果是：

hello, worldahello, world〈BELL〉hello, world

其中〈BELL〉代表ASCII码值7，产生一个简短的音响。

习题1—3

修改温度转换程序，使该程序在表格上面打印出一个表头。

程序如下：

```
main( )    / *Fahrenheit-Celsius table for fahr = 0,20,…, 300 */
{
    int lower, upper, step;
    float fahr, celsius;
    lower=0;          /*lower limit of 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;
    }
}
```

程序中，在循环前加入了一个printf函数，即：printf("Fahr Celsius \n")；这就在对应表项上面加了一个表头。程序的其余部分与K&R书中的程序相同。

习题1—4

编写一个打印出由摄氏温度转换为华氏温度的对应表格的程序。

程序如下：

```
main( )       /* Celsius-Fahrenheit table for celsius=0,20,…,300 */
{
    int lower, upper, step;
    float fahr, celsius;
    lower=0;          /* lower limit of 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;
     }
}
```

程序运行产生的表中，含有以摄氏温度值(0～300度)表示的温度和其对应的华氏温度值，华氏温度值是按已知的摄氏温度值经下列方程计算的：

fahr = (9.0 * Celsius)/5.0 + 32.0

这个解答过程与原书中华氏到摄氏温度制表程序的逻辑过程类似。整形变量 lower，upper 和 step 分别表示温度的下限、上限和摄氏温度变量 Celsius 的量值步长。变量 Celsius 的初始值为温度下限值，进入 while 循环后，进行对应的华氏温度值的计算，打印出两者的对应值，然后变量 Celsius 又按照步长增值。

当变量 Celsius 超过温度上限值时，while 循环结束。

习题1—5

修改上述的温度转换程序，从 300 度开始到 0 度为止反序打印温度表格。

程序修改如下：

```
main( )    /* Fahrenheit-Celsius table for hahr=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));
}
```

可见，程序只修改了一个地方，即：

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

这里，for 语句的第一部分：fahr = 300; 是将初值赋给华氏温度变量 fahr，这个初值即温度上限 300。第二部分是一个循环控制条件，fahr>= 0，用来测试变量 fahr 是否超过或者达到温度的下限值，如果测试条件满足，for 循环将一直进行。第三部分，即重新赋值部分，由 fahr = fahr - 20 完成，每次从华氏温度变量中减去步长值。

习题1—6

编写一个程序，该程序可以对空格、制表和换行字符进行计数。

程序如下：

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

这里，整型变量 nb，nt，nl 分别用来对输入的空格、制表和换行字符的个数进行计数，开始时，这三个自动变量的值都等于 0 。

程序进入 while 循环体后，从输入取得的数据中每当遇到空格、制表和换行符时就记录下来，在整个循环中，每一次循环都要执行所有的 if 语句。如果所取得的数据字符是除空格、制表和换行符之外的其他任何字符，则不作任何事情。如果是三者之一，那么对应的计数器加 1，这个过程一直进行到碰到文件结束符(EOF)时为止，然后，再打印出最后结果。

如果程序采用 if-else 控制流程，则可以加速 if 的判断，但由于课程的进度，此种语句结构在进行此练习前尚未引出，所以练习题中也没有采用。如果要采用这种结构，程序可以如下：

```
#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);
}
```

习题1—7

编写一个程序，它把输入复制到输出，用单个空格来代替一个以上的空格串。

程序如下：

```
#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);
        if (c== ' ')
            if (lastc != ' ')
                putchar(c);
            lastc=c;
    }
}
```

这个程序搜索空格的存在，如果找到一个空格，则继续测试该空格后是否还跟有另一个空格，如果是，这个空格就被忽略掉，否则将打印出来。这样，保证所打印的是每一个单个的空格和空格串的第一个空格。

整型变量c记录了从输入取得的当前字符的ASCII码值，变量lastc记录了所取得的前一个字符的ASCII码值。符号常数NONBLANK把变量lastc初始化为一个任意的非空格的字符，比如“a”字符。

While循环体中的第一个if语句处理出现的非空格字符，并把它们打印出来，第二个if语句处理空格串，第三个if语句则对单个空格或者空格串中的第一个空格符进行测试和处理，最后，对变量lastc予以更新，然后过程又开始重复。

在这个程序中也没有用if-else语句，因为按照C语言课程的进度，此时还没有引入这种结构。如果采用if-else控制流程，程序可以修改成为：

```
#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，如果引入“逻辑或”(||)运算符，还可以将上述程序改成下列形式。不过，按照C语言课程的进度，“逻辑或”的概念在K & R原书中到第19页以后才引入。程序如下：

```
#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1—8

编写一个程序，用三个字符的序列，如>，退格和—，打印出→来代替制表字符tab，同样，采用这样的序列打出←来代替退格字符backspace，这样，制表符和退格符就都成为可见的了。

程序如下：

```
#include <stdio.h>
main( )    /* replae 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);
        }
}
```

程序中引入了三个条件，并在while循环中对输入的字符进行检查测试，看是否有制表符，退格符或其他字符。如果发现是前两者，则打印出对应的三字符序列来代替它，若不是其中任一种，则直接将输入字符返回到输出，这个过程一直进行到文件终止。

在CRT终端上，字符是不能重迭显示的，所以，-\b>只能显示成>，>\b-只能显示成-。因此，最好采用-\b>和-\b<的形式，这样，至少可以保留箭头的方向。

如果采用if-else控制流程，程序也可以改写成如下形式，虽然这个结构后来才引入。

```
#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);
    }
}
```

习题1—9

如何进行一个单词计数程序的测试？有什么限制条件需要考虑？

要测试单词计数程序，首先测试在没有输入时的情形，这时，运行输出应当是0，0，0，(即表示无新行，无单词，无字符)。

其次，测试只有一个字符时的单词，这时运行输出应为1，1，2，(即1行，1个单词，2个字符——字符后跟有一个换行符)。

接着，再测试有两个字符的单词，运行输出应该是：1，1，3，(即1行，1个单词，3个字符——两个字符跟有一个换行符)。

此外，可测试两个单字符单词的情况(这时输出应为1，2，4)；以及两个单字符单词，但每个单词各放1行的情况，(此时输出应为2，2，4)。

由此可见，有以下一些限制情况。

* 无输入。
* 无单词——只换行。
* 无单词——只有空格，制表和换行符。
* 每行有一个单词——无空格和制表符。
* 单词位于行首。
* 单词在若干空格后出现。

习题1—10

编写一个程序，打印出输入的诸单词，每行打印一个。

程序如下：

```
#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);
    }
}
```

此程序中，变量 inword 是一个整型布尔变量，它用来标记程序当前是否正处于一个单词上。程序开始时，inword 的初始值是 NO，因为此时还不知道单词从何处开始。

第一个 if 语句：if (c = = ' '||c = = '\n'||c = = '\t')决定是否c是一个单词分隔符，如果是，则转入第二个 if 语句。

第二个 if 语句：if (inword = = YES)，用来检查这个单词分隔符是否表示单词的终止，如果是，则另换新的一行，并将变量 inword 更新，反之，进行空操作。

如果 c 不是单词分隔符，那它要么是单词的第一个字符，要么是单词中的另一个字符。如果是新单词的开始，则要更新变量 inword，同时将两种情况下的对应字符打出。

习题1—11

重新编写单词计数程序，对“单词”采用较好的定义，比如：单词是以字母开头的，由字母，数字和省略符号组成的字符串。

修改后的程序如下：

```
#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' ) ||          /* lower case        */
        (c >= 'A' && c <= 'Z'))                /* 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);
    }
```

针对单词的较好的定义，程序需要加入三条语句：

```
语句1    if ( (c >= 'a' && c <= 'z' ) ||          /* lower case */
    (c >= 'A' && c <= 'Z'))                       /* upper case */
```

这个语句决定了单词的第一个字符是大写字母还是小写字母(ASCII码)，如果是，它就是新单词的开头，则将变量inword更新，把单词计数器也增值。

```
语句2    else if ((c >= 'a' && c <= 'z' ||          /* lower case ?*/
        (c >= 'A' && <= 'Z' ) ||                   /* or upper case ? */
        (c >= '0' && c <= '9' ) ||                 /* or digit ?      */
```

```
        c == '\'')                          /* or apostrophe ?  */
        ;                                   /* within a word    */
```

这条语句要执行，程序当前要正处于单词上，且该字符不是空格、制表和换行符时才进入此结构。通过这项检查，可以保证单词是字母、数字和省略字符(ASCII码)的序列。

语句3 else inword = NO; 是必须的，因为除了空格，制表、换行符以外的非单词的字符需要处理，如果是这些字符，则更新inword，以退出单词。

习题1—12

编写一个程序，打印出输入单词的长度出现频度的直方图，在水平方向上打印直方图是较容易的，而在垂直方向上的打印则稍复杂一些。

下列程序为打印水平直方图程序

```
#include <stdio.h>
#define MAXHIST  15                         /* max length of histogram */
#define MAXWORD  11                         /* max length of a word */
#define YES      1
#define NO       0
main ( )                                    /* horizontal histogram   */
{
    int c, i, inword, nc;
    int len;                                /* length of each bar     */
    int maxvalue;                           /* maxmium value for wl[ ] */
    int wl[MAXWORD];                        /* word length counters   */
    inword = NO;
    nc = 0;                                 / number of chars in a word */
    for (i=0;  i < MAXWORD; i++)
        wl[ i ] = 0;
    while ( ( c = getchar( ) ) != EOF)
    {
     if (c == ' ' || c == '\n' ||c == '\t')
      {
           inword = NO;
           if (nc > 0 && nc < MAXWORD)
             ++wl [ nc ];
           nc = 0;
    }
    else if (inword == NO)
    {
       inword = YES;
       nc = 1;                                  /* beginning of a new word */
    }
    else
          ++nc;                             /* within a word                */
```

```
}
    maxvalue = 0;
    for (i = 1; i<MAXWORD; i++)
        if (wl [i] >maxvalue)
            maxvalue = wl [i];                    /* max value found */
    for (i = 1; i<MAXWORD; i++)
{
   printf ("%5d - %5d: ", i, wl [i]);
   if (wl [i] >0)
   {
      if ((len = (wl [i] * MAXHIST / maxvalue)) <= 0)
         len = 1;
   }    else len = 0;
   while (len>0)
   {
         putchar('*');
         --len;
   }
      putchar('\n');
    }
}
```

程序中，一个空格，换行或者制表字符标记一个单词的结束。如果碰到一个单词，即(nc>0),并且单词的长度比规定的最大的单词长度小(nc<MAXWORD)，那么，对应的单词长度计数器加1(++wl[nc])。如果碰到的不是单词而是空格，换行和制表符，或者该单词的长度超过范围(nc>=MAXWORD)，单词长度计数器则不作任何修改。

一旦所有单词都已读入，那么所得到的最大值(maxvalue)取决于数组wl。

程序中，变量len按照MAXHIST和maxvalue的限制来标记数组wl[i]的值，当wl[i]大于0时，至少打印一个星号“*”。

最后打印出水平直方图。

下列程序为打印垂直直方图的程序：

```
#include <stdio.h>
#define MAXHIST  15                    /* max length of histogram  */
#define MAXWORD  11                    /* max length of a word     */
#define YES      1
#define NO       0
main()
{                                     /* horizontal histogram */
   int c, i, j, inword, nc;
   int len;                            /* length of each bar        */
   int maxvalue;                       /* maximum value for wl [] */
   int wl [MAXWORD];                   /* word length counters      */
   inword = NO;
   nc = 0;                            /* number of chars in a word */
```