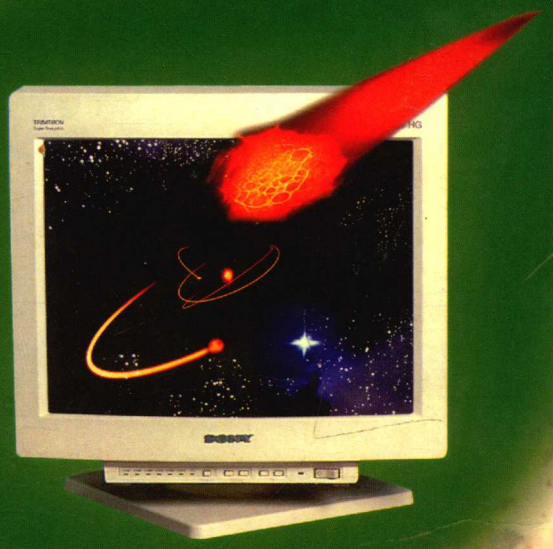

计算机等级(水平)考试系列教材

C 语言程序设计教程

石竹　吴国凤　孙家启　编著

安 徽 大 学 出 版 社

计算机等级(水平)考试系列教材

C语言程序设计教程

石 竹 吴国凤 孙家启 编著

安徽大学出版社

图书在版编目(CIP)数据

C语言程序设计教程/石竹等编著．－合肥：安徽大学出版社，1999.3
计算机等级(水平)考试系列教材
ISBN 7－81052－227－2

Ⅰ.C… Ⅱ.石… Ⅲ.C语言－程序设计－水平考试－教材 Ⅳ.TP312

中国版本图书馆CIP数据核字(1999)第01372号

C语言程序设计教程 石 竹 吴国凤 孙家启 编著

出版发行	安徽大学出版社	印 刷	中国科学技术大学印刷厂
	(合肥市肥西路3号 邮编230039)	开 本	787×1092 1/16
联系电话	总编室 0551－5107719	印 张	13
	发行部 0551－5107784	字 数	316千
责任编辑	李 虹	印 数	0001～5000册
封面设计	孟献辉	版 次	1999年3月第1版
经 销	新华书店	印 次	1999年3月第1次印刷

ISBN 7－81052－227－2/TP·20 定价 17.00元

目 次

第一章　C语言概貌

在本章，我们首先看看C语言的程序是怎样的，以便对C语言的概貌有所了解，搭起一个全面深入学习C语言的框架。

1.1　C语言的来由

1971年，贝尔实验室的D·M·Ritchie在B语言的基础上，用了一年的时间写了第一个C语言编译程序，1972年投入使用。之后，成为unix或Xenix操作系统的主力语言。是当今最为广泛的程序设计语言之一。

1.2　起　步

先从一个最简单的程序看起，该程序的功能是在屏幕上显示：Hello!

程序如下：

```
main()
{
    printf("Hello! \n");
}
```

任何一个C语言源程序，都必须经过编译，连接后方可执行，不同种类的C语言，其编译，连接方式也不同。由于当前大部分的教学和考试都是以Turbo C 2.0为主，故本书以此作为主要考虑对象。要想运行该程序，按下Ctrl+F9，系统对程序进行编译、连接，若程序没有什么错误，则在屏幕上输出Hello!，然后回到程序编辑状态，想看程序运行结果，可按Alt+F5键。

现在，对程序本身作一解释。一个C语言程序，无论其大小，皆由一个或多个“函数”组成，而“函数”完成要做的操作。上例中，main( )就是一函数。但main是一个特殊名，任何程序有且仅有一个main，且程序总是从main的开头执行。main的具体结构为：

```
main( )          /*函数名*/
{                /*函数开始*/
  ⋮              /*函数体*/
}                /*函数结束*/
```

一个函数可调用另一个函数。函数间也可用参数来交换数据。函数名后的一对括号内括着一个参数表。例子中的main函数没有使用参数，故为一对空括号(不能省)。花括号{ }用以包围构成函数的语句(即函数体)。例子中，函数体是由一条语句组成的。

```
printf("Hello! \n");
```

它是一个函数调用，它调用一个名叫printf的函数，带有参数“Hello! \n”。printf是一

个屏幕上输出的库函数。它打印输出作为其参数的字符串(由括号所括的字符序列)。

串中的序列\n是C语言中用作换行符的记号,当遇到它时,屏幕上的光标前进到下一行的最左边。因此,如漏掉了\n,会发现本语句输出结束时并未换行,若多次调用后,得到一个长的输出行。

例如:

```
printf("Hello ,");
printf("This is a test.");
printf("\n");
```

其效果等于 printf("Hello, This is a test.\n");

它们都是在屏幕上输出:Hello, This is a test.

实际上,C语言中,一个反斜线"\"后带一个字符,表示转义序列。转义序列是用来表示ASCII字符集内的控制代码如"回车"、"退格"、"制表"等不易获得或不可见的特殊字符。C语言中定义的转义序列有:

\n	换行	\\	反斜线
\t	制表	\'	单引号
\r	回车	\b	退格
\f	换页	\0	空(null)

要注意的是,虽然转义序列由反斜线"\"和另一个字符组成,但它却等效于一个单独字符,与其他单个字符性质相同。

转义序列也可和其他字符混合使用,例如:

```
printf("A\n BC\n AD\t\ EFG\n");
```

输出为:

```
A
BC
AD      EFG
```

最后说一下,C语言对大小写是敏感的,即字母大小写是不等价的,在程序中一般情况下用小写。

1.3 变量、语句和算术运算

下面的程序是输入一个华氏温度,要求输出其对应的摄氏温度。转换公式为:

C(摄氏)=(5/9)×(F-32)。

```
/* Print Celsius-Fahrenheit */
main( )
{
    float fc, ff;              /*变量说明*/
    scanf("%f",&ff);           /* 输入华氏温度 */
    fc=(5.0/9.0)*(ff-32.0);
    printf("f=%5.1f\t c=%7.2f\n",ff,fc);
```

}

程序的头一行，是所加的注释，用符号/*和*/括起来，可在这两个符号之间加入任意文字作为注释。注释可加在空白字符或换行符出现的任何地方。编译程序对源程序进行编译时，对注释部分弃之不顾，即不产生代码行。

C语言中，所有变量(程序执行过程中其值可以变化的量)在使用之前必须加以说明，通常把说明放在函数的开始部分。说明由类型及该类型变量的一个变量表组成。

例如： float fc, ff;

类型 float 表示变量 fc，ff 代表的浮点数，即带有小数部分的数。Turbo C 中，浮点数是由 4 个字节(32 位)组成，量值范围在 10^{-38} 至 10^{+38} 之间。

除 float 外，C 还提供了其他几种基本数据类型：

char	字符	/*一个字节(Turbo C 中)*/
short	短整数	/*一个字节(Turbo C 中)*/
int	整数	/*二个字节(Turbo C 中)*/
long	长整数	/*四个字节(Turbo C 中)*/
double	双精度浮点数	/*八个字节(Turbo C 中)*/

此外，还有这些基本类型的数组、结构、联合、枚举、指向它们的指 针以及返回这些基本类型值的函数。上述所有这些，将在以后章节讨论。

函数体中包含了一系列语句，这些语句描述了该函数要做的事。每个语句必须以分号结束。

程序实际运行时将从下面语句开始：

scanf("%f",&ff);

这是一个函数调用语句，scanf 是一个库函数，功能是从键盘上读取数据赋给变量，例中，它以浮点数形式从键盘读入数据赋给变量 ff。

接下来的是一个赋值语句，它先对赋值号右边的算术表达式进行计算求值后，赋给赋值号左边的变量 fc，表达式中用了 5.0/9.0 而不用 5/9 是因为 C 语言中，整数相除其结果也为整数，即要舍去所有的小数部分。5/9 得 0，0 乘任何数为 0，当然不行。

最后一个语句 printf，它是如何工作的呢？ printf 实质上是一个通用格式变换函数，它的第一参数是要打印字符串，其中%指明其他参数在什么地方被替换，以什么格式打印。如例中%5.1f 是指浮点数 ff 要打印为 5 个字符的宽度(包含小数点)，小数点后边有一位数字。该浮点数替在“f=”之后，“c=”之前打印。同样，fc 被打印为 7 个字符的宽度，带有 2 位小数的数。格式说明的有些部分是可以省略的：例如①%6f 指明浮点数按 6 个字符输出，但小数位数不限；②%.2f 则要求小数点后边有二位，但宽度不限；③%f 仅表明该数按浮点数输出。除%f 外，printf 还能识别%d 是个十进制整数；%o 是八进制整数；%x 是十六进制；%c 是字符；%s 是字符串，而%%是%本身。

printf 第一个参数内的%结构必须同其第二个、第三个等相应的参数配对。否则，就会出错。

1.4 分 支

在上节温度转换程序中，假设当转换后的摄氏温度高于35度时，屏幕还需显示“热”的字样，则程序改为如下：

```
main( )
  {  float fc, ff;
     scanf("%f",&ff);
     fc=(5.0/9.0)*(ff-32.0);
     print("f=%5.1f\t c=%7.2f\n",ff,fc);
     if(fc>35.0) printf("c=%7.2f>35,热",fc);
  }
```

该程序除了增加最后一条语句外，其余与前同。最后一条语句是条件(分支)语句。其形式(语法)一般为：

```
if (表达式)
        语句1
[else
        语句2]
```

在计算机文献中，方括号中的内容表示可选。即在条件语句中，else部分可有可无。这条语句的意思是，如果表达式(也称为条件)值为“真”(非零值)，就执行语句1，如果表达式值为“假”(零值)，同时又有else部分，则改为执行语句2；若无else部分，则什么也不执行。

程序中，如果fc的值大于35，则打印“热”的字样，否则什么也不做。

1.5 循 环

若要求程序输出如下的温度转换对照表：

```
0        -17.8
20       -6.7
40       4.4
60       15.6
 ⋮        ⋮
260      126.7
280      137.8
300      148.9
```

则程序为：

```
main( )
{  int nf;
   nf=0;
   while (nf<=300)
     {   printf("%4d%6.1f\n",nf,(5.0/9.0)*(nf-32));
```

```
        nf=nf+20;
    }
}
```

程序中的变量 nf 意义同前例中的 ff,用来代表华氏温度,所不同的是,本例中将它说明成了整型,这是因为要输出的对照表左列为整数之故,对应地在 printf 语句中,nf 的输出格式也变成了"%4d"。

因为该表的每一行都用同样的方法计算,所以采用循环,每行重复计算一次,使用 while 语句的目的就在于此。

```
while (nf<=300)
{
  ⋮          /*循环体*/
}
```

括号内的条件被测试,若为真(nf 小于等于 300),循环体(用花括号包围着的所有语句)就执行一次。然后再测试条件,若仍为真,就再执行循环体,一直到条件为假(nf 大于 300)后,循环才结束,并继续执行本循环之后的语句。因为本程序无后继语句,故程序终止。

while 后面循环体,可有一条或多条语句,当超过一条语句时,须用花括号将循环体包围起来,若只有单个语句时,没有必要使用花括号。例如:

```
while (i<j)
  i=i+1;
```

上例中,while 所控制的语句书写时一般缩进一个制表符宽度,这样一眼就能看出什么语句在循环体中了。缩进写法强调了程序的逻辑结构,使程序易读。虽然它不是必须的要求,但这却是个良好的书写习惯,推荐大家使用。例中程序的 printf 语句按 4 个宽度的整数形式输出 nf,再根据 nf 的值计算出摄氏温度值后输出。下面的语句:

```
nf=nf+20;
```

用来控制循环的间隔(步长)与次数。

一般情况下,一个程序可用不同的方法编写,现在来书写另一个温度转换程序。

```
main( )
  {  int nf;
     for (nf=0;nf<=300;nf=nf+20)
       printf("%4d%6.1f\n",nf,(5.0/9.0)*(nf-32.0));
  }
```

以上程序同前一例产生同样的答案,但程序本身却大不一样。主要是采用了另一种循环——for 语句。

for 同 while 相比,比较清楚,它包含有用分号隔开的三个部分。第一部分 nf=0 是初始化部分,在进入循环之前做一次。第二部分是控制循环的测试条件 nf<=300,该条件被测后,若为真,就执行循环体(这里仅为一条 printf 语句)。然后循环变量加步长 nf=nf+20 再对条件重新测试,若为真,继续循环下去,直到条件为假后循环终止。同 while 一样,循环体可以是一个单独的语句,或用花括号括着的一组语句。for 的初始化部分及再初始化部分(循环变量加步长)可以是任一单个的表达式。

选用 while 循环还是 for 循环可以随意而定，这是个人习惯问题，只要能使程序清楚就行。

1.6 符号常数

在以上的程序中，用了若干个常数，如 300 及 20 等。这类常数叫直接常数，可直接用到程序中。但直接常数是一些纯粹的数字，是人们不易看出其意义的“神仙数”。往往使程序的可读性、维护性变差。特别是在程序中修改这些常数时，需要一个一个地去修改，既费时又容易出错。为避免这种情况，C 语言提供了一种预处理语句 #define，它能把常数定义为称做“符号常数”的一串特定字符。此后，编译程序读到这些符号常数时，就会用具体的常数值，去代替那些符号。实际上，用 #define 命名处理的，并不限于数字，还可以是其他类型的数据。这将在以后讨论。下面看看程序中如何使用符号常数。

```
#DEFINE    LOWER  0
#DEFINE    HIGHER  300
#DEFINE    STEP  20
MAIN( )
  {
    INT NF;
    FOR (NF=LOWER;NF<=HIGHER; NF=NF+STEP)
    PRINTF("%4D%6.1F\N",NF,(5.0/9.0)*(NF-32.0));
  }
```

程序中，LOWER，HIGHER 和 STEP 是符号常量。符号名通常用大写字母来书写，从而容易与用小写字母写的变量名相区别。符号常数定义一行之后不加分号，因为被定义名之后的整个一行是一起被代替的，若加上分号，这些分号就会被一起替换到程序中去，这将可能使程序出错。

1.7 一组实用的程序

现在来讨论对字符数据进行简单加工且互相关联的一整套程序。

1. 字符输入和输出

在 C 语言的标准程序库中提供了一次输入一个字符和输出一个字符的函数。每调用 getchar( )一次，它就取来一个字符，并将这个字符作为函数值返回。当执行

c=getchar( )

之后，变量 c 就包含了下一个输入字符。字符一般是从键盘送入的。函数 putchar(c)是 getchar( )的对偶。

putchar(c)在某一输出介质上，通常是在显示屏上显示 c 的内容。

同 printf 一样，getchar 和 putchar 也都不是 C 语言本身的一部分，而是放在标准程序库中，所不同的是当要用 getchar 和 putchar 函数时，需在程序的开头写上如下语句：

#include<stdio.h>

#include 同 #define 一样是一条予处理语句，其功能是 在 C 程序编译前将 include<

stdio.h>的全部内容插入到原程序开始处。在许多C语言程序中,当使用标准输入输出除scanf和printf之外的函数时,须在源程序的开始部分加上此语句。

2.文件复制

有了getchar和putchar,无须对I/O了解更多,也可写出数量惊人的有实用价值的程序。最简单的例子是从输入复制到输出的程序。程序的算法为:

1:取一字符
2:当(字符不是文件结束符)
3:(复制)
{输出刚读入的字符
取下一个字符}

此时可得C语言程序:

```
#include<stdio.h>
main()
{
int c;
c=getchar( );
while (c!=EOF)
 { putchar(c);
   c=getchar( );
 }
}
```

程序的第一行是一条预处理语句,称为文件包含命令,其意义是把<>内指定的文件包含到本程序来,成为本程序的一部分。被包含文件通常由系统提供,其扩展名为.h,因此也称为头文件。

stdio.h是标准输入、输出函数原型所在的头文件。因此凡是在程序中有输入、输出时,原则上都应包含stdio.h文件。

关系运算符!=的意思是"不等于"。

符号常数EOF在stdio.h中已定义,当getchar遇上输入结尾时,EOF为真(非零);否则EOF为假(零)。

程序中,把变量c说明为int而不是char,是想使它能妥善保存由getchar返回的值。因为该值除了代表所有可能字符外,还要表示EOF。

有经验的C程序员在编写以上程序时,实际上会写得更简练一些。在C语言中,例如,

c=getchar( ); 这样的赋值语句也可用在表达式中,其值就是赋到左边的值。如果赋给c的字符是放在while语句的测试部分,则上述程序可写成:

```
include <stdio.h>
main( )
{
  int c;
  while((c=getchar ( ))!=EOF)
```

```
    putchar(c);
}
```

这个程序输入一个字符给 c,然后测试该字符是否为文件结束标志。若不是就执行 while 循环体,打印该字符。然后 while 再重复,当输入结束符(^Z)时,while 终止,整个程序也就终止了。

将以上程序和前面的一个程序比较可看出,它将输入集中处理,只用了一个 getchar,并使程序缩短了,将赋值语句嵌套在测试部分正是 C 语言简明性的例子之一。

要注意在条件部分中包围着赋值语句的括号是十分必要的。因为!=的优先级比=的优先级高。若无括号,它就会先做关系运算!=,再做赋值运算。例如:

```
c=getchar( )!=EOF
```

就相当于

```
c=(getchar( )!=EOF)
```

结果 c 的值由 EOF 决定,要么为 1,要么为 0,视 getchar 的调用是否遇到文件结尾而定,显然,这不是所希望的结果。

3. 字符计数

将复制程序稍作改动,就可得到下面这个对字符进行计数的程序。

```
#include <stdio.h>
main ( )
{
    long nc;
    nc=0;
    while (getchar( )!=EOF)
      ++nc;
    printf ("%ld\n", nc);
}
```

语句++nc 是 C 语言中新的运算符++,意思是"增加 1",也可写成 nc=nc+1,但++nc 更简明,且通常运行效率也高。对应的运算符--,意思是"减少 1"。算符++和--既可用作前缀算符(++nc),也可作为后缀算符(nc++)。这两种形式在表达式中可能有不同的值,但++nc 和 nc++都是使 nc 加 1。在本程序中,仅用前缀形式。

在上述字符计数程序中,用来计数的变量 nc 是用的 long 型,而不是用的 int 型,这是因为用 long 型可以计较多的数的缘故。当需要处理更大的数字时,可以使用 double 型(双倍字长的浮点数)。在本程序中,printf 语句中的转换说明%ld 告诉 printf,其对应的参量是一个 long 型整数。

下面这个程序采用了 for 来代替 while,将进一步说明用 for 语句来建立循环的方法。

```
main( )
{
    double nc;
    for (nc=0;getchar( )!=EOF;++nc)
        ;
```

```
    printf("%.0f\n",nc);
}
```

printf 用%f 表示 float 和 double;%.0f 制止输出小数部分。

由于所有的工作都在测试和再初始化部分完成了,所以 for 的循环体为空。但 C 语言规则要求给 for 语句一个循环体,故用一个孤单的分号";"满足要求。从技术上讲这一分号叫做空语句。

4.行计数

下面的程序是对输入行进行计数。这里假定每个输入行都一定以换行字符"\n"结束。

```
main( )
{
    int c ,nl;
    nl=0;
    while ((c=getchar( ))!=EOF)
      if (c=='\n')
        ++nl;
    printf("%d\n",nl);
}
```

while 的循环体是由 if 语句构成,双等号"=="是 C 语言"等于"的记号。用双等号是为了将其与赋值号"="相区别。

将单个字符写在单引号之间,就可获得该字符在 ASCII 字符集中对应的数字值,这就叫做字符常数。例如,'A'是字符常数,ASCII 字符集中它的值为 65,也即 65 是字符 A 的内部表示。

字符串中的转义序列也是合法的字符常数。因此,在算术表达式或测试中,'\n'代表换行字符的值。应十分注意'\n'是单个字符,在表达式中等效于一个整数,而"\n"是只含一个字符的字符串。

5.单词计数

下面程序是对行、单词及字符计数。在 C 语言中,单词定义为:不包括空格、制表符及换行的字符序列。

```
#include <stdio.h>
#define YES 1
#define NO 0
main( )
{
  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)
    {
      inword)=YES;
      ++nw;
    }
  }
  printf("%d,%d,%d\n",nl,nw,nc);
}
```

每当程序遇到单词的第一个字符时，nw 就进行计数(累加)。变量 inword 记录着程序当前是否在一个单词上，初始化时，它被赋值 NO，以指出“不在单词上”。用符号常数 YES 和 NO 而不用值 1 和 0，以使程序易读。同时也便于对程序的修改和维护。

语句 nl=nw=nc=0;　使所有三个变量都置为 0。这是因为 C 语言中，允许一次为多个变量赋值。

算符 || 代表或(OR)运算，故下面这条语句：

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

表达式含义：“如果 c 是一个空格，或者 c 是一个换行，或者 c 是一个制表字符…”。C 语言中，还有与(AND)对应的操作符 &&。

本例出现了 C 的 ef-else 语句，当 if 的条件部分为假时，指定作另一种动作。

1.8 数　组

下面编写一个程序来对每一个数字、空格及其他所有字符的出现次数进行计数。

输入可分为 12 类：10 个数字、空格、其它字符。用一个数组来保存每一个数字的出现次数，较通常采用的 10 个独立变量方便得多。程序如下：

```
#include<stdio.h>
main()
{    int nc, ni, nwhite, nother;
     int ndigit[10];
     nwhite=nother=0;
     for (ni=0;ni<10;++ni)
          ndigit[ni]=0;
     while((nc=getchar())!=EOF)
          if(nc>='0'&& nc<='9')      /* nc是否为数字字符*/
               ++ndigit[nc-'0'];
          else if(nc==''||nc=='\t')  /*是否为空格*/
               ++nwhite;
```

```
            else    ++nother;
        for (ni=0;ni<10;++ni)
            printf("digit %d=%d\n",ni,ndigit[ni]);
            printf("space=%d\n other=%d\n",nwhite, nother);
}
```

数组说明语句:int ndigit[10];说明 ndigit 是有 10 个整数元素的数组。C 语言中,数组下标是从 0 开始,故元素有 ndigit[0],ndigit[1],…,ndigit[9]。下标可以是任意的整数表达式,指出该元素在数组中的位置。

因为数字在 ASCII 码表中是连续编码的,即'0'~'9'是正的并以增序排列,同时它们之间无其他非数字字符。故程序可依赖数字的字符表示特性。例如:

```
if(nc>='0'&& nc<='9')
```

决定在 nc 中的字符是否为数字,若是,该数字的值为 nc-'0'。C 语言中,在算术运算的情况下,字符变量和常量实际上都和整数相同。如对应在存于 nc 中的字符 0 至 9,nc-'0'就是一个整数表达式,其值在 0 至 9 之间。对于数组 ndigit 来说,这个表达式是有效的下标。

1.9 函 数

函数为程序设计提供了方便的手段,它通常把一些运算封装进一个黑箱之中,以后用时,只需知道某函数是干什么的,而无需知道它是怎样完成任务的。

标准库函数是事先将一些常用的计算或操作编好程序后放入编译系统中,这样当你要用这些计算或操作,无需再自己编程,而直接调用即可,如 scanf,printf,getchar 和 putchar 等,但我们更感兴趣的是自己编写函数。函数在引用之前必须定义,编写函数的过程就是定义函数的过程。

函数的定义包括二部分:函数头和函数体。函数头定义函数的名字和参数,函数体则定义该函数要完成的工作。现在来编写一个 absolute(x)的函数来说明函数定义方法,它是求整数 x 的绝对值。

```
#include <stdio.h>
main()
{ int na, nb;
  scanf("%d%d",&na, &nb);
  printf("%d,%d,%d\n",absolute(na),absolute(nb),absolute(5));
}
  int absolute(x)
  int x;
  {  int  na;
     if(x>=0) na=x;
     else na=-x;
     return(na);
  }
```