HOPE

美国国家标准

C 语言
编程指南

〔美〕Ioseph Sant
张玉亭 编译
李建国 薛斌校

中国科学院希望高级电脑技术公司

美 国 国 家 标 准

C 语言编程指南

〔美〕Joseph Sant'

张玉亭 编译

李建国 校

薛 斌

中国科学院希望高级电脑技术公司

一九九一年四月

前　言

我在从事有关C语言教学与工作实践中，产生了编写这本书的念头。有一次，我承担了一项编写支持计算机指令包的合同，其中指令包要用C语言编写，加之设计项目需尽快完成，使得我必须很快掌握C语言。然而，遗憾的是我翻阅了几本有关C语言的著作，发现要想从书中获取必要的信息是较为困难的，因为许多书常常用很多篇幅解释C语言中一些不常用的特点，而对重要的使用特点强调不够。尽管二叉树和递归循环之类的特点很有趣，但并不能帮助我完成该设计项目。

我所需要的是这样的书：着重解释重要特性，而对不常用的特点仅做简要提示，这样可迅速提高工作效率，我认为：C语言也象其它语言一样，经常使用的仅是其中的一小部分而已。本书正是基于这种观点编写而成，它旨在使读者快捷而有效地掌握C语言。所以用更多的篇幅谈及一些我认为重要的特性，而简要提及一些不重要的特点，如：二叉树，梵塔问题等。尽管程序设计理论是非常重要的，但本书的重点并不在于此。因此，本书将要讨论的程序设计理论侧重于程序举例。

书中的程序是相互关联的。首先，读者要学会编写简单的实用程序。然后，在此程序上，进一步设计相对复杂的程序，书中通过许多实例说明如何使用C语言的基本特性，在较为复杂的程序中，则体现了个别特性的应用。为说明这一点，大部分章节之后都附有“小结”，对在大程序中如何恰当地使用这些特性做了进一步解释。

学习C语言最好的方法是实践。所以，每章后都附有习题，绝大部分习题在10～60分钟内便可完成。本书可供大学计算机专业教学之用，对专业程序员定会有所裨益。

要判定一种语言中什么是最重要的，这带有很大的主观性。在此，我想谈谈我的看法（或许是偏见）。要想成为出色的C语言程序员还应该理解C语言的如下两个重要特性。

- 函数和存储类：用于设计和执行好结构程序。
- 指针：用于巧妙地支配存储器。

此外，C语言的另一个重要特性就是有效地使用结构。因为，结构可用来减小大程序的逻辑复杂度。

阅读本书的读者应对编程原理有所了解，至少用过其它程序语言编写过程序，并熟悉如下概念：循环、算术、简单的数据类型和数组。

本书以讲ANSI C语言为主，但当ANSI C语言与“旧版C语言”在标准之间有大的区别时，本书会对“旧版C语言”做出说明。

作者：

约瑟夫·桑特

Joseph Sant

注：ANSI为美国国家标准协会的英文缩写。

目　录

第一章　工作子集

对大多数计算机语言来说，仅仅掌握其中最基本的语句和语法规则便可编写程序。所以只要掌握这一章的内容，就能很快编写程序并对其进行测试，本章内容包括

- 程序结构：语句，块
- 定义数据：整型、浮点型和字符型
- 输出：printf( )
- 循环结构：'while'
- 算术操作符：+，－，*，／，++，－－
- 判断：'if'，'else'；
  关系操作符：==、<、>、>=、<=、!=
- 输入：scanf( )

1.1　程序结构

所有C语言程序都包括函数、块和语句，每个C语言程序必须有main( )函数，main( )函数放在程序开始执行处。请看如下工作程序。该程序含一个main函数，而main( )函数中包括语句和块

```
main( )
{    int    count;

     count=1;                          语句

     while(count  <=10)
     { printf("loop 10 times\n");
       count=count+1;                  块
     }
}
```

函数

1.1.1　程序元素

语句：语句就是一个有效的C语言指令，后加“；”。

```
count = 1;
```

块：块由语句组成。它表示程序段，if语句和循环语句的开始和结束。在下面的程序段中，“while”有两个语句，并用大括号（{ }）括起来、大括号一般是成对出现，块可以嵌套。

```
while(count  <=10)
{ printf("loop 10 times\n");
  count=count+1;
}
```

函数：函数就象其它语言中的过程和子程序一样，函数允许给程序段的块起名字，每个程序必须有 main( ) 函数。

1.2 数据类型／说明

首先我们来谈谈三种数据类型：整型，浮点型和字符型。在使用变量之前，必须对它加以说明。用来说明变量类型的几个关键字如下：

float——浮点数

int ——整型（带符号）

char ——单字符

定义变量先要列出其类型，然后是数据名。数据名开头必须是字母或下线“_”，后面可是字母，下线或数字。允许大、小写字母，但没有明文规定变量名只用小写字母，一般来说，仅前八个字母是有意义的。

```
float interest, principal_amt, no_of_loops;
```

定义变量时，可通过赋值进行初始化。

```
int limit=40;
```

使用整型和浮点型的程序：

注：最后一句打印语句中的“%f”是一个专用格式符号。它由“prin cipal”的值自动代替。

```
main( )
{ int  iterations;
  float principal, interest;

  iterations=1;
  principal=10000;
  interest= .08;
  while(iterations <=10)
    {principal=principal+(principal*interest);
     iterations=iterations+1;
    }
  printf("principal after 10 years is %f/n", principal);
}
```

1.3 输出：`printf( )' 函数。

`printf( )' 函数是标准库中的函数，用于向标准输出设备（一般是荧光屏）输出。它是函数，不是C语言的一部分，它可打印字符串和数字变量，但要打印数字变量，必须将特别字符放在字符串中，以表明你希望它们用数字来代替。

打印字符串内容：

```
printf("this string is in quotes\n");
```

“\n”是换行符，打印下一个字符将另起一行。“\t”为水平制表符。

打印整型或浮点型

要打印变量必须将专用转换符放在字符串中，并把要打印的变量放在字符串后。转换符号是：

%c　　打印字符

%d　　打印十进制整型

%f　　打印浮点数

%s　　打印字符串

```
int shares=1000;
float price=15.50;

printf("You have %d shares at %f a share\n", shares, price);
```

输出：

```
You have 1000  shares at 15.500000 a share
```

1.4 循环结构：'while' 语句

"while"是顶部检查循环，只要条件满足，便执行下去，下面的while循环执行20次。

```
int  num=1;
while(num<=20)
{printf("this  is printed 20 times\n");
 num=num+1;
}
```

C语言中有一个有趣特性：对于循环或if关系表达式不是绝对必要的。它使零值为假，非零值为真，对循环，零为停，非零为执行。下面程序段将无限循环下去。

```
while(1)
   printf("will this ever end\n");
```

如下程序使用while，看看10.000美元存到一百万美元要用多少年。

```
main()
{
   int    iterations, magicyear;
   float principal, interest;

   iterations =0;
   principal  =10000.0;
   interest   =.08;

   while(principal<1000000.00)
   {
      principal=principal+(interest * principal);
      iterations=iterations+1;
   }
```

```
    magicyer = iterations;
    printf("Your magic year is %d\n", magicyer); }
```

1.5 算术／赋值

C语言中，简单的算术运算符就象其它语言中的算术运算符一样，如：+，－，*，／。

除此之外，还有余数除，运算符为“%”。当某数被另一数除时，其值为余数。见如下两个例子。

```
c = 22 % 7;
d = 7 % 22;
```

c值为1，d值为7。

递增和递减运算符（++和--）

这两个专用运算符每次加1和减1。可放在任意数字变量的前面或后面（如：++count，或count++）。如果变量用在算术语句或比较语句中，运算符放前放后会影响其结果的。

如果“++”放在变量名前，在算术语句用变量值之前，变量加1。

“--”运算符与“++”运算符相同，见下面几例。

```
int a = 4; b = 9, c, d, e;

c = ++a;
d = b++;
e = ++b;
```

在此情况下，“c”的值为5，因为“a”的值在使用之前加1，“d”的值为9，因为“b”值在加1之前已用。“e”值为11，为什么？

```
main( )
{ int    iterations = 0;
  int    limit = 5;
  int    loops = 0;

  while( ++iterations <= limit)
              loops++;

  printf("the number of loops is %d\n", loops);
 }
```

在上述程序段中，“++”在变量名“iterations”之前，这就是说“iterations”在与“limif”值（即5）比较之前加1，在第一次循环中，“iterations”从0增值为1，然后与“limit”比较，“iterations”与“limit”比较的有效值依次为1，2，3，4，5。循环次数为5次。

递增运算符还用于循环，以计算循环次数。在这一特殊情况下，它等价于

“loops = loops + 1;”。要想“loops”从1开始递增，我们可以使用“++loops”，而不是“loops++”。

那么，如果我们把“++”放在“iterations”的后面（即iterations++）会怎么样呢？在每次开始循环时，“iterations”的初值（为0）与‘limit’比较。比较后，“iterations”从0增值为1，“iterations”与‘limit’比较的有效值依次为0，1，2，3，4，5。这次循环了6次。这是为我们比较后再增加1，所以多了一次循环。

用过这两个运算符的变量不能用在别的表达式中，在运算过程中，这两个运算符会造成变量值的改变，即副作用。标准C语言并没有规定整个表达式计算后，什么时间值发生变化或使用运算符后立即发生变化。因此，象‘C = a++ - ++a’这样的语句将会产生不可预测的结果。

1.6 ‘if’语句

if语句与其它语言的if语句相同。即：是否执行与if连接的语句或块取决于条件的真假。条件必须放在if的后面，并用圆括号括起，象while语句，关系条件并不是绝对需要的。因为零值为假，非零值为真，关系和连接运算符如下：

运算符	表示
关　系	
==	等于
<	小于
>	大于
<=	小于等于
>=	大于等于
!=	不等于
连　接	
&&	逻辑与
||	逻辑或
!	逻辑反

下面程序段说明对简单条件，复杂条件，单语句和程序块如何使用if结构。不要把复合条件搞得太复杂，这会变得很难理解。

单一条件，单条语句

```
if ( number_of_days == 31 )
  monthno++;
```

复合条件，程序块

```
if ( number_of_days == 31  &&  month_length == 31 )
{
```

```
    month__no++;
    number__of__days = 0;
  }
```

有时理解和处理条件的反比理解和处理条件本身更容易。非运算符“!”将单一或复合条件的真值变反（非零变为0，0变为1）。如果你不想用复合语句，必须用圆括号把条件括起，因为运算符优先级问题，以上处理是必须的，优先级在下一章讨论。

让我们举个销售汽车的例子，除拥有8个汽缸的车和3型车外，我们想降低所有车的价格。下面程序段使用了非运算符。

```
if(!(model==3 || cylinders==8))
  price = price*0.90;
```

注意：“非”是将整个复合条件变反。此外，要注意，我们不用一个“!”而用2个‘!=’运算符也同样产生同样的效果。

```
if(model != 3 && cylinders != 8)
  price = price*0.90;
```

对C语言程序设计的新手来说，常见的一个错误是在if后面的条件中用“=”而不是“==”。编译程序会以为你希望赋值，简单地测试一下赋值的结果是零（假）而不是非零（真）。

做为一个例子，我们接收一个代表dice__roll的整型，并测试一下是否为2。下面程序段中的scanf( )用来接收键盘键入的数据。

```
/*  This code docsn't work properly */
int dice__roll=0;
scanf("%d", &dice__roll);
if(dice__roll=2)
  printf("you rolled a 2, you lose\n");
```

上面程序段是完全合法的。不管输什么数，都将信息打印出来。其原因是我们使用了赋值运算符（‘=’）而不是比较运算符（‘==’）。这样编译程序会认为你要执行赋值而不是比较。由于括号内表达式结果总是非零值（2），所以为真。

1.7 if…else语句

如果if的条件为假，else允许执行某个动作，或多个动作。每个else通常与最近的且无匹配的if相匹配，使用if/else你必须留心，因为这会影响到将要执行的语句。

简单的if…else语句

```
if(principal >= 10000)
  interest__rate = .1075;
else
  interest__rate = .1025;
```

嵌套的if…else语句

下面程序段将某股票交易与一最小数量（2000.00）相比较。如果小于这个最小数

量，则赋最小费用（30.00）；否则依据股票价格而定。

```
int no_shares;
float(transaction, stock_price, charge, com_rate;

if(transaction<2000.00)
  charge=30.00;
else
{ if(stock_price>20)
    com_rate=.05;
  else if(stock_price>10)
    com_rate=.04;
  else
    com_rate=.03;
  charge=30.00+no._shares*com_rate;
}
```

1.8 输入：'scanf( )' 函数

函数scanf( )接收键盘键入的用户信息，并将信息变换成任意数据类型，象printf( )函数一样，我们必须使用一组转换符（如：%c，%d）。转换符必须在格式串中（双引号里）。存储输入值的变量地址必须在格式串后，通过将'&'放在变量名前来确定变量地址。

如果我们想将输入值接收到整型'idno'的话，我们会提供变量字符串"%d"和'idno'的地址，即：&'idno'。所需语句如下：

```
int idno;
scanf("%d", &idno);
```

几个变换符号如下：

%d	带符号十进制数	（int型变量）
%f	浮点数	（floaf型变量）
%c	字符	（char型变量）
%s	字符串	（字符数组）

scanf( )函数通常是用空白，水平制表，回行等来区分输入数据，如果我们用scanf( )来接收下面一行中的三个浮点数（即：scanf("%f%f%f", &a, &b, &c)；它将不管数据间的空白。请注意，尽管我们可以在一行中放入几个输入值，但只有敲回行或输入键才执行转换。

```
23.78   345.1   78.9   (enter)
```

当使用scanf( )来接收字符串时，要有所留意，一旦使用转换符'%c'，从键盘接收的下个字符便存在变量中，而不管空白否。见如下语句及用户输入数据后所得结果：

```
char a, b, c, d;
```

scanf("%c%c%c%c", &a, &b, &c, &d);

用户输入

1<enter >2 <enter>

结果

a 是 1

b 是 '\n' (换行)

c 是 2

d 是 '\n' (换行)

要接收字符串必须建立一个字符数组，你可以这样做，确定数组名，在方括号内写入所需字符数量，对字符串来说，常常可确定比你希望使用的最大字符串多一个字符。这就是说要给字符串留有空间。下面这个例子保留了一个足以装10个字符的数组。

```
char name[11];
```

当scanf( )用于字符串时，不必用运算符'&'的地址，因为数组名可理解为地址，下面是scanf( )接收字符串的例子。

```
scanf("%s", name);
```

对scanf( )，使用"%s"时要注意。首先它不管数组是否超界。如果scanf( )接收一个10字符数组，而用户输入了20个字符，那scanf( )只将头10个字符接收到'name'中，余下字符被删除。

第二，用户输入内容用空格区分。人们常常愿在字符串中留有空格。为了避免这一问题，我们可用gets( )函数，它接收一个输入行(输入行由回行来确定)，并将它存储在已确定的字符数组中。此外，因为这是字符串，必须比输入的最长字符串多存储一个字符。gets( )一次仅可取一个字符串。

```
char sentence[81];
gets(sentence);
```

1.9 小结——商业程序1

现在我们可以试着使用我们在本章中学过的几条指令写一个小程序：计算两种商品：金子和银子的价值，其计算公式是：商品价格(即：silver_rate 和 gold_rate)乘以盎司量(即：amount)。用户输入商品和数量。

在写该程序段时要注意如下几种情况：

1.9.1 选择和说明数据类型

'type'为整型，因为它代表商品类型，且有数值。所有其它数据项目为float。因为它们带有小数点。数据说明在main( )函数的顶部。在它们用于指令之前。

使用'printf( )'：

printf( )语句包括象换行符‘\n’和制表符这样的控制字符以及转换符来显示变量值。

使用‘scanf( )’：

对scanf( )我们必须使用两个不同的转换符（%d和%f），这是因为我们先接收整型，然后接收浮点型。

在scanf( )中，我们可以使用运算符‘&’提供变量地址，而变量名本身不提供。

1.9.2 使用嵌套的if语句

我们使用一个嵌套的if语句。这样，在我们打印出商品的全部价值之前，我们可以测试商品类型中是否有错，大多数if和else控制单个语句，有一个else控制一块（即用大括号括起的那块），该块包括另一个if结构。

程序说明

在一个程序中，我们可在‘/*’和‘*/’之间写上我们的说明。

```
/* COMMCD.C    This program asks the user for a commodity
                type (gold or silver) and amount (ounces)
                and prints out the total value.

*/
main( )
{
    int type;
    float amount;
    float total_value;
    float gold_rate = 450.00;
    float silver_rate = 7.00;

    printf("/n/t1  Gold");
    printf("/n/t2  Silver/n");
    printf("/nEnter Type of Commodity: ");
    scanf("%d", &type");
    printf("/nEnter number of ounces: ");
    scanf("%f", &amount);

    if( type! =1 && type! =2)
      printf("Invalid commodity type/n");
    else
```

```
}
  if ( type = =1 )
     total_value = gold_rate *amount;
  else if ( type = =2 )
     total_value = silver_rate *amount;
  printf("The total value for your metal is %f/n",
          total_value );
}
```

1.10 练　习 1

1．写一个程序，用户输入半径，计算圆面积。最好用循环来处理，当用户输入－99时，循环结束，圆面积公式为：

面积 = 3.14159 × 半径 × 半径

2．写一个程序段，将华氏温度变为摄氏温度，用户输入华氏温度，最好用循环来处理，当用户输入－99时，循环结束，变换公式为

摄氏 = (华氏 － 32) × 5.0／9.0

用5和9代替5.0和9.0试验一下。

3．某国家公园在生态环境没遭破坏的情况下，仅能提供1000只鹿，如果目前有300只鹿，且每年生长率为28%。生态环境不被破坏要用多少年？

最好考虑到狼有10只，其生长率为10%，每只狼每年吃掉 5 只鹿，森林不被破坏要用多少年？

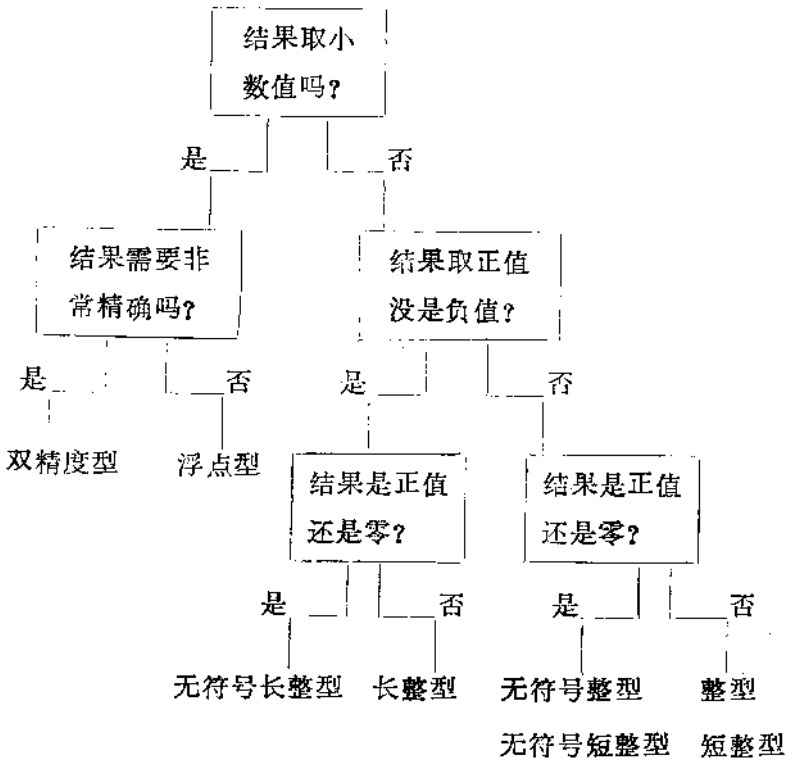
第二章　扩 展 子 集

目前，我们已经品尝了编写简单 C语言程序的滋味。现在我们可以看一看如何扩展我们的子集，以便更灵活地解决问题。C是小语言，但它提供了许多行之有效的工具——丰富的运算符，众多的数据类型和一套少而有效的 C语言指令。

2.1 数据类型和常数

前面谈到的数据类型是带符号整型，浮点数和字符。C提供了用户在程序中可用的许多其它数据类型。

我们已经有了存储数字和小数的类型，为什么还需要更多的数字数据类型呢？回答很简单，数据类型越多，设计程序会更灵活。下面的问题也许会帮助你决定如何使用数据类型。



选择数据类型还要注意另外几点：

一般来说，使用double（双精度型）而不用float（浮点型）或使用 long（长整型）而不用int（整型）需使用更多的存储器。你可经常使用sizeof运算符来确定数据类型的大小。见下面的例子。

```
printf("a double takes up %d bytes\n", size of(double));
```

数据类型的范围（即：类型可代表的最低和最高值之间的变化范围）视机器的不同而有所差异。有时，变量的数据类型范围即便在同一个机器上也许会有所不同，这取决于编译时，你的选择是什么。数据类型简表及其范围如下：

数据类型	一般小型机	一般大型机
短整型	-128…127	-32768…32767
整　型	-32768…32767	-2147483648…2147483647
无符号整型	0…65535	0…4294967295
长整型	-2147483648…2147483647	-2147483648…2147483647

带符号与无符号数据类型间的区别是范围的起始点不同。下面是大多数机器中短整型和无符号短整型的范围。在这两种情况下，最高值大于最低值65,535，但带符号短整型的起始点是-32,768，而无符号短整型的起始点是0。