C 应用程序设计技术

李文兵　编著

清 华 大 学 出 版 社

(京)新登字 158 号

内 容 简 介

本书是作者开发 C 应用程序的经验总结。

全书分 14 章。前 4 章讲的是 C 程序结构方面的问题,内容包括程序控制结构、结构数据的设计方法与技巧,以及 C 程序的基本模块——函数的调用方法和设计中的问题,并对 C 的疑难问题做了详尽的解释。后 9 章介绍了 C 应用的 13 个专题,即文件处理程序设计、使用汇编语言的方法、PC 硬件资源管理方法、数据结构的实现方法及其应用程序设计、绘图函数及图形的程序设计、汉字处理技术、中断程序设计、C 的进程控制方法、时间与日期的应用程序设计、文本编辑程序设计、图文共面软件的程序设计、行式打印机的彩色图形打印程序设计、汉字下拉菜单的程序设计等。每个专题都对解决问题的思路、技术、技巧做了深入浅出的探讨,并给出了读者可照搬照用的功能函数及 222 个应用实例。附录介绍了 Turbo C 的程序调试技术。

本书为程序员、高级程序员而写。可作为计算机、自动化、软件、信息工程等专业大学高年级学生、研究生的教材和教师及有关人员的参考用书。

图书在版编目(CIP)数据

C 应用程序设计技术/李文兵编著 .—北京:清华大学出版社 .1994
ISBN 7-302-01468-X

Ⅰ.C… Ⅱ.李… Ⅲ.① C 语言-程序设计 ②程序设计-C 语言 Ⅳ.TP312C

中国版本图书馆 CIP 数据核字 (94) 第 01131 号

出版者:清华大学出版社 (北京清华大学校内,邮编 100084)
印刷者:北京丰台丰华印刷厂
发行者:新华书店总店北京科技发行所
开 本:787×1092 1/16 印张:28.25 字数:670 千字
版 次:1994 年 8 月 第 1 版 1996 年 5 月 第 3 次印刷
书 号:ISBN 7-302-01468-X/TP·579
印 数:20001—26000
定 价:23.50 元

前　言

很早就想写这么一本书。经过多年的钻研与开发实践,《C 应用程序设计技术》一书与广大读者见面了,相信大家会喜欢这本书。

为什么要编著这么一本书呢?

一方面是想通过这本书,向读者展示 C 语言能解决任何问题,而且能解决得很漂亮。就拿硬件管理功能来说吧,高级语言鞭长莫及,汇编语言又太专业;而 C 语言正象本书第 7 章所介绍的那样,可以使用多种方法管理计算机硬件资源。本书使用这些方法,在第 7 章介绍了 PC 机音响系统的控制与应用;还介绍了对 PC 机 CPU 中的寄存器及 PC 机内存的管理;在第 7 章和第 11 章介绍了 PC 机中断系统的调用;在第 12 章介绍了键盘的管理;在第 13 章介绍了微机显示系统的管理;还介绍了行式打印机的彩色图形打印程序设计。除此之外,第 5 章说明了 C 的文件管理功能相当强;第 6 章说明了 C 与其它语言的连接、混合使用方便可行;第 8 章说明了 C 可以构造出各种数据结构;第 9 章说明了 C 有很强的绘图功能;第 10 章说明了 C 处理汉字的能力无可比拟。

再就是想为推广 C、普及 C 进一步做点工作。虽然 C 语言是一流的程序员语言,在经济发达国家颇为流行,但在我国还有相当数量的软件工作者仍习惯于使用过去熟悉的语言开发项目,致使项目功能受到局限。作为计算机工作者,我深感有责任继续做好 C 的普及推广工作。本书正是想为用 C 开发项目的人员提供一个可照搬照用的蓝本。为此,本书起点高,力求做到实用性强,思路、技术与技巧尽量交待清楚,并针对学生及软件工作者普遍存在的问题,着力在如下三个方面:

· 如何设计出地道的 C 程序,这是许多程序员所追求的目标。本书 1～3 章的内容相信会对读者有所帮助。

· 许多学生及软件工作者对 C 的某些情况感到困惑,第 4 章对 C 的疑难问题做了详尽的解释。

· 本书力求做到以软为主,软硬结合,以求使读者知其然,又知其所以然。

感谢彭群力、王玉华、崔竹、蓝智斌、于忠民、谭峻、田鸿芬等同志及时调试程序。谭峻同志还参加编写了附录。

由于作者水平及条件所限,书中难免存有不妥甚至错误之处,希望得到专家与读者的指正。

李文兵

1993 年 10 月

目　录

1 章

C 程序控制结构

C 语言定义了 10 个控制语句,根据功能可把它们归纳为三类,如表 1.1 所示。

表 1.1　C 语言控制语句

用　　途	语　　句
用于条件结构或分支结构程序设计	if 语句 switch 语句
用于循环结构程序设计	while 语句 do—while 语句 for 语句
用于中断结构程序设计	break 语句 continue 语句 goto 语句 return 语句 空语句(;)

C 程序控制结构就是由这 10 个语句构成的。本章集中介绍 C 程序的控制结构。

1.1　条件控制结构与分支控制结构

1. 条件控制结构

(1) 结构与功能　条件控制结构是由 if 语句构成的,其结构形式如下所示:

if (条件表达式)　语句

这种控制结构的功能是,首先判断条件表达式的值,若不为 0,条件判断为真,则执行( )后的语句。

(2) 条件表达式的设计与注意事项

条件表达式一般是使用六种关系运算符(> 、> = 、< 、< = 、= = 、!=)和三种逻辑运算符(&&、 、!)设计的。设计中需要注意的是运算符的优先级别。例如,

a> b && b= = c 与 (a> b) && (b= = c)

c! = d　 d> b 与 (c! = d)　(d> b)

完全等效。原因就是> 和= = 的优先级别大于 &&;! = 和> 的优先级别大于　。因此,可不必使用圆括号。

但是,当 && 和 ! 用在同一个表达式中时,则由于 && 的优先级别大于 ! ,使得

a> b ! c= = d && d> a

等效于

a> b ! (c= = d && d> a)。

但由于 ! 和 && 是表达式的序列点,且 ! 是该表达式的第一个序列点,故运算顺序是先做 a> b,后做(c= = d && d> a)。

(3) PAD 图 ! if 结构的 PAD 图如图 1.1 所示。图中的 Q 为( )内的条件,S 为( )后要执行的语句。

2. 二分支结构

(1) 结构与功能 ! 二分支结构也是由 if 语句构成的,其结构形式为:

if(条件表达式) 语句 1

else ! 语句 2

功能是,如果条件表达式的解为非 0 值,即为真,则执行语句 1;否则执行语句 2。可见,该结构为分支结构。

图 1.1 ! if 结构的 PAD 图

图 1.2 ! if—else 结构的 PAD 图

(2) PAD 图 ! 这种结构的 PAD 图有两种画法,如图 1.2 所示。其中,Q 为( )内的条件,S1 为语句 1;S2 为语句 2。

3. 多分支控制结构

多分支结构分两种:一种是由 if 语句构成的;另一种是由 switch 语句构成的。

(1) if—else ! if 结构

① 结构与功能 ! 该结构是由 if 语句构成的,其结构形式为:

if(条件表达式 1) 语句 1

else if(条件表达式 2) 语句 2

else if(条件表达式 3) 语句 3

!

else if(条件表达式 n) 语句 n

else 语句 n+ 1

在这种结构中,程序只执行满足条件的语句;也就是说,若条件表达式 i 的解为非 0 值,就执行语句 i。如果哪个条件都不成立,就执行语句 n+ 1。可见,该结构可用来设计多

分支结构程序。

② PAD 图　该结构的 PAD 图如图 1.3 所示。

【练习 1.1】

```
main( )
{
 int x, y= 1, z
 if( y!= 0) x= 5;
 printf( "%d\n", x);
 if( y= = 0) x= 3;
 else x= 5;
 printf( "%d\n", x);
 x= 1;
 if( y< 0)if( y> 0) x= 3;
 else x= 5;
 printf( "%d\n", x);
 if( z= y< 0) x= 3;
 else if( y= = 0) x= 5;
 else x= 7;
 printf( "%d\t%d\n", x, z);
 if( z= (y= = 0)) x= 5; x= 3;
 printf( "%d\t%d\n", x, z);
 if( x= z= y); x= 3;
 printf( "%d\t%d\n", x, z);
}
```

图 1.3　if—else if 结构的 PAD 图

运行结果:

```
5
5
1
7      0
3      0
3      1
```

下面分析执行结果。

· 初始化 y= 1

```
if (y!= 0) x= 5;
```

其中(y!= 0)为:

(1!= 0)为真

故执行

```
x= 5;
```

所以, 第 1 次输出结果为 5。

·y 值仍保持 1

if(y= = 0)x= 3; else x= 5;

其中(y= = 0)为假,故执行 x= 5;所以,第 2 次输出结果也是 5。

·初始值: y= 1, x= 1

结构:

if (y< 0) if (y> 0) x= 3;

else x= 5;

相当于:

```
if (y< 0)
  { if (y> 0) x= 3;
    else x= 5;
  }
```

由于(y< 0)为假,故花括号中的语句不被执行,x 保持其原值,故第 3 次输出结果是 1。

·y 的值仍为 1

```
if (z= y< 0) x= 3;
else if (y= = 0) x= 5;
else x= 7;
```

其中,(z= y< 0)为:

(z= (1< 0))

故 z 为 0,条件判断为假,语句 x= 3;不执行。

条件(y= = 0)也为假,故语句 x= 5;也不执行。

因此,语句 x= 7;被执行,所以,第 4 次输出结果为 7 和 0。

·y 值仍为 1

if(z= (y= = 0))x= 5; x= 3;

其中,(z= (y= = 0))执行结果 z= 0,条件判断为假,故 x= 5;不执行,而执行 x= 3;故第 5 次输出结果为 3 和 0。

·y 值仍为 1

if(x= z= y); x= 3;

其中,(x= z= y)为:

(x= (z= 1))

故有 x,z 的值均为 1,条件判断为真,执行其后的空语句(;),x,z 值均不变;但后面还有一个 x= 3;语句,执行此语句后 x 的值变为 3,故最后一次输出结果为 3 和 1。

(2) switch 结构

① 结构 switch 结构是由 switch 语句构成的多分支程序结构,其结构形式为:

```
switch(条件表达式)
{case  常量表达式 1:
       语句 1
```

```
        break;
case    常量表达式 2:
        语句 2
        break;
        ┆
case    常量表达式 n:
        语句 n
        break;
default:
        语句 n+ 1
        break;
}
```

该结构的 PAD 图与图 1.3 相同。

② 功能　该控制结构的功能可归纳为两点:

·如果条件表达式的解等于常量表达式 i 的值, 则执行语句 i;

·如果条件表达式的解与任何常量表达式的值都不相同, 则执行 default: 后的语句。若 default 部分缺省, 则什么也不执行就跳出 switch 结构。

③ 注意事项　使用该结构应注意如下几点:

·switch 结构中的 break 语句不能缺省。每个 case 中的 break 语句, 使 switch 结构只执行该 case 中的语句。即一执行 break 语句, 便从 switch 结构中跳出, 如练习 1.2 所示。若无 break 语句, 则继续执行下面各 case 的执行语句, 如练习 1.3 所示。

【练习 1.2】

```
main()
  {
   int i= 2;
   switch(i)
   {
   case 1;
        printf( "case is in 1\n");
        break;
   case 2:
        printf( "case is in 2\n");
        break;
   case 3:
        printf( "case is in 3\n");
        break;
   default:
        printf( "it is default\n");
   }
}
```

运行结果:

```
case is in 2
```

【练习 1.3】

```
main()
{
 int i= 2;
 switch(i)
 {
 case 1:
      printf( "case is in 1\n");
      break;
 case 2:
      printf( "case is in 2\n");
 case 3:
      pritnf( "case is in 3\n");
 default:
      printf( "it is default\n");
 }
}
```

运行结果:

```
case is in 2
case is in 3
it is default
```

可见,当每个 case 中的 break 缺省后, switch 结构将达不到多分支的效果。

· 条件表达式的类型必须与常量表达式的类型一致。

· 各常量表达式的值必须互不相同。

· switch 结构允许若干个 case 公用一个执行语句。如练习 1.4 所示。

【练习 1.4】

```
# include < stdio.h>
main()
  { int i,j, nw, nd[10];
    char s[ ], c;
    j= 0;
    nw= 0;
    for (i= 0; i< 10; i+ + )
        nd[i]= 0;
    while((c= getchar())! = '\n')
    { s[j]= c;
      switch(s[j])
      { case '0':
        case '1':
```

```
            case '2':
            case '3':
            case '4':
            case '5':
            case '6':
            case '7':
            case '8':
            case '9':
            nd[s[j]-'0']++;
            break;
            default:
            nw++;
            break;
            }
            j++;
        }
        for (i=0;i<10;i++)printf("%d",i);
          printf("\n");
        for (i=0;i<10;i++)
          printf("%d",nd[i]);
        printf ("\nnw=%d\n",nw);
      }
```

运行结果:

```
247909347674e ewfqgq \= .,
0 1 2 3 4 5 6 7 8 9
1 0 1 1 3 0 1 3 0 2
nw= 12
```

· case 部分与 default 部分的顺序可以自由书写。如果 default 部分位于程序最后，则 default 部分的 break 语句便可以缺省，如练习 1.3 所示；否则，break 语句也是必不可少的，如练习 1.5 所示。

【练习 1.5】

```
main()
  { int i=5
    switch(i)
    { case 1:
        printf("case is in 1\n");
        break;
      default:
        printf("case is in default\n");
        break;
      case 2:
```

```
            printf( "case is in 2\n");
            break;
        case 3:
            printf( "case is in 3\n");
            break;
    }
}
```

运行结果:

case is in default

1.2 循环控制结构

1. while 结构

(1) 结构和功能　其结构形式如下:

while (条件表达式)语句

功能是,首先判断( )内最初的条件表达式,其解不为 0,则执行( )后的语句;一旦执行该语句,接着便再次判断条件表达式,进行与上述相同的处理:即若其解不为 0,就再次执行( )后的语句;若为 0,则跳出 while 循环。可见,使用 while 结构可用来设计循环结构程序。

(2) PAD 图　while 结构的 PAD 图如图 1.4 所示。

(3) 注意事项:

① while 结构所要重复执行的语句,可以是简单语句、复合语句,也可以是空语句。

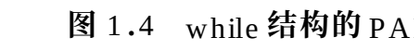

图 1.4　while 结构的 PAD 图

② 如果 while 结构的条件判别式为非 0 常数,则该 while 结构便是无限循环的 while 结构。

③ while 的条件表达式中可以引进库函数,如

```
while ((ch= getchar())! = '\n');
```

这里,在条件表达式中使用了函数 getchar( )。由于该 while 结构的执行语句是空语句(;),故只要从键盘上输入的字符不是'\n',输入操作就可以连续地进行下去。

2. do—while 结构

(1) 结构与功能　其结构形式如下所示:

```
do        语句
while     (条件表达式);
```

功能是,首先要执行的是 do 后面的语句,之后才判断条件表达式,其解不为 0,则再次执行 do 后的语句;为 0,则 do 后的语句执行终止。可见 do—while 结构和 while 语句的唯一区别就是,do—while 结构不管条件表达式的值如何,首先执行一次要循环执行的语句。就是说,即使第 1 次判断条件表达式时,其值为 0,也要执行一次要循环执行的语句。而 while 结构是首先判断条件表达式,如果第 1 次判断时,其值为 0,则什么也不执行,就

执行下面的语句去了。可见,do—while 结构同样可用来设计循环结构程序。

(2) PAD 图　do—while 结构的 PAD 图如图 1.5 所示。

图 1.5　do——while 结构的 PAD 图

3. for 结构

(1) 结构　其结构形式如下所示:

for (表达式 1; 表达式 2; 表达式 3)

　　语句

其中,表达式 1 一般用来初始化循环控制变量;表达式 2 为条件表达式,以此控制循环次数;表达式 3 用来修改循环控制变量。

(2) 执行过程　for 结构的执行过程是,首先求解表达式 1;其次判断表达式 2,不为 0,则执行( )后的语句;此后求表达式 3 的值,再返回来判断表达式 2,不为 0,则再次执行( )后的语句。这样一直重复执行到表达式 2 的值为 0 止,不再重复操作,而去执行下面的语句。因此,若第 1 次判断表达式 2 就是假,则 for 语句便只执行表达式 1 就停止了。for 语句也可用来设计循环结构程序。

由此可以看出,for 语句等价于如下程序:

```
表达式 1;
while(表达式 2)
{语句
 表达式 3;
}
```

(3) PAD 图　for 结构:

```
for (i= M; i< N; i+ = K)
        S
```

表示变量 i 的值从 M 开始,每执行一次语句 S,其值就增加 K,重复执行 S 到 N 为止。它的 PAD 图如图 1.6 所示。

图 1.6　for 结构的 PAD 图

图 1.6 可进一步简化,如图 1.7 所示。

(4) 无限循环的 for 结构

for 结构其三个表达式中的任何一个都可以缺省。当表达式 2 缺省时,C 编译系统则认为表达式 2 恒为真,故 for 结构就成为无限循环结构,其结构形式有如下三种:

· for(表达式 1; ; 表达式 3) 语句

图 1.7　简化的 for 结构 PAD 图

·for(表达式 1;;)语句

·for(;;)语句

(5) 逗号表达式的 for 结构

for 结构中的表达式 1 和表达式 3 都可以是逗号表达式,形成逗号表达式 for 结构。例如:

for (i= 0, j= 20; i< j; i+ + , j- -)语句

其中,表达式 1 为 i= 0, j= 20;表达式 3 为 i+ + , j- - 。根据逗号表达式的运算规则,这两个表达式都是从左到右运算,结果的类型和值是右操作数的类型和值。

这里举一个应用实例。该例是把字符串中的字符位置前后倒置,如练习 1.6 所示。

【练习 1.6】

```
# include < stdio.h>
main()
{
 int i, j= 0
 char c, s[100];
 while((s[j+ + ]= getchar())!= '\n');
 for (i= 0, j- = 2; i< j; i+ + , j- - )
 {
  c= s[j];
  s[j]= s[i];
  s[i]= c;
 }
 for(i= 0; s[i]!= '\n'; i+ + )
    printf("%c", s[i]);
}
```

运行结果:

```
abcdefghijklmnopqr
rqponmlkjihgfedcba
```

运行结果:

```
1234567890- = ;. ,
, . ;= - 0987654321
```

1.3 嵌套控制结构

嵌套控制结构是指控制结构嵌入到同一种控制结构中所构成的控制结构。可见,嵌套是同一种结构的重复使用。本节介绍 C 程序设计中常用的嵌套控制结构。

1. 嵌套 if—else 结构

(1) 结构与功能 if—else 结构中还可以嵌入 if 结构或 if—else 结构,构成嵌套的

if—else 结构。作为例子,这里给出一个三层嵌套的结构,其结构形式如下:

```
        if(条件表达式 1)
              if(条件表达式 2)
外     中     内 if(条件表达式 3) 语句 1
层     层     层 else   语句 2
              else   语句 3
        else   语句 4
```

其执行过程是,若条件表达式 1 的解为非 0 值,即为真,则执行中层 if—else 结构;否则执行语句 4。执行中层时,若条件表达式 2 的解为非 0 值,则执行内层 if—else 结构;否则执行语句 3。执行内层时,若条件表达式 3 的解为非 0 值,则执行语句 1;否则执行语句 2。

(2) 注意事项　使用嵌套 if—else 结构时,应注意如下几点:

① 注意 else 就近与 if 配对的原则。例如,

```
if (Q1)
    if (Q2) S2
    else    S3
```

相当于

```
if (Q1)
{ if (Q2) S2
  else    S3
}
```

② 注意不要浪费花括号。在 if—else 结构中嵌入 if—else 结构,即:

```
if (Q1)
    if (Q2)   S1
    else      S2
else  S3
```

相当于:

```
if (Q1)
    { if (Q2)   S1
      else      S2
    }
else  S3
```

因此,不必浪费这对花括号。

③ 注意 else;的作用。例如,结构:

```
if (Q1)
    if(Q2)   S2
    else;
else  S3
```