用C语言编制文字处理软件

—附源程序详解

严洪华　强寿松　倪旭东　著

人民邮电出版社

用C语言编制文字处理软件

——附源程序详解

严洪华　强寿松　倪旭东　著

人民邮电出版社

登记证号（京）143号

内 容 提 要

本书向读者提供了一个用C语言编制的，主要语句均有中文注释的，完整的文字处理软件源程序清单。并通过对该软件的剖析，详细介绍了有关运用C语言处理计算机内存、屏幕、键盘、磁盘等方面的编程技巧，以及运用这些技巧编制一个完整的文字处理软件的方法。使读者在C语言的应用及文字处理软件的编制等方面有所补益。

本书共分五章，第一章介绍软件下拉式主菜单的编程方法。第二章介绍光标与屏幕的处理及编辑文件的输入、修改、插入、删除、存储等基本功能的编程方法。第三章介绍字块处理功能及字符串查找、替换、文章排版等辅助功能的编程方法。第四章介绍文件打印功能的编程方法。第五章介绍编辑软件其它功能的编程方法。

本书读者对象是初步掌握C语言编程以及从事计算机软件开发工作的初、中级技术人员。也可作为中、高等院校计算机软件专业学员的参考书籍。

用C语言编制文字处理软件
——附源程序详解

严洪华　强寿松　倪旭东　著

*

人民邮电出版社出版发行
北京东长安街27号
北京振华印刷厂印刷
新华书店总店科技发行所经销

*

开本：787×1092　1/16　　1994年4月　　第　一　版
印张：10.25　页数：82　　1994年4月　北京第1次印刷
字数：　255　千字　　印数：　1－5 000　册

ISBN7－115－05214－X/TP·098

定价：　10.00　元

前　言

我们写作这本书的目的是希望能在C语言的应用与文字处理软件的编制两个方面对读者有所裨益。

众所周知，C语言是世界上应用最广泛的几种计算机语言之一。其主要特点是能以高级程序设计语言的结构和编程环境，给人们提供了类似于汇编语言那样的系统资源操纵能力及程序的执行效率，从而使之成为开发计算机软件的极有效的工具。目前，随着计算机在我国的普及，系统软件与应用软件的开发工作日显重要，而C语言又具有高效、灵活、功能强大及入门容易的特点，所以深受广大计算机工作者的欢迎。目前，越来越多的中、高等院校在计算机课程中开展了C语言的教学。

然而，C语言的高效、灵活及功能强大的特点，也同时给人们带来了“入门容易得道难”的问题。一般情况下，一个有过其它高级语言编程经验的计算机工作者在七、八天内读完一本C语言的应用手册并上机编制一些简单程序并非难事，但要真正掌握C语言的精髓，充分发挥其巨大的潜力，做到应用自如却并不容易。即使是对C语言有一定了解的编程人员，在开发一个大、中型规模的实用系统软件过程时，也常常会被一些突如其来的、难以解释的错误搞得头昏脑胀。

当然，“入门”既不难，只要方法得当，且持之以恒，“得道”也是可以做到的。而研究与运行他人编制的现有程序则是深入学习C语言的一条捷径。因为一个精心编制的程序不仅是一个算法、数据结构、程序设计方法和语言运用的综合体，同时也是一种良好的编程技术与风格的体现。

基于以上想法，我们在这本书中向读者提供了一个由作者用C语言自行编制的并且大部分语句都有中文注释的完整的文字处理软件源程序清单。

我们给这个软件定名为MYWS，即取英语 MY WORDSTAR 之意。对于WS软件，计算机工作者并不陌生，它是我国计算机用户使用比较早而且应用面相当大的一个文字处理软件。中文WS编辑软件是在美国MicroPro公司推出的西文WS版本的基础上汉化而成的。源程序用汇编语言编制，设计十分精巧，结构相当严谨，以至成了文字处理软件编制中的一个样板。例如：Tubro C，Qbasic 等系统软件的编辑器以及目前在我国应用相当广泛的 WPS 软件的编辑功能，均采用了与 WS 兼容的方式。

与中文WS不同的是，MYWS 文字处理软件虽然沿袭西文 WS 的编辑功能，采用了与其兼容的操作方式，但它不是在西文 WS 基础上的汉化版本，而是用 C 语言重新设计与编制的一个汉字文字处理软件。在该书中，我们将通过对这个文字处理软件源程序清单进行的解剖式讲解，向读者详细介绍如何运用C语言进行文字的输入、修改、排版和打印的编程，怎样去处理计算机内存、屏幕、键盘与磁盘等方面的操作，以及如何将它们结合起来编制一个自己的文字处理软件的实现方法，其中一些模块的设计程序可作为整体应用于其它类似的软件编程中。并且，在本书的后面附有该软件的全部源程序清单，读者只要将其内容在 IBM PC 及其兼容机上正确的输入，并通过 Turbo C 2.0 以上版本的系统编译后，即可在 DOS 操作系

统下顺利地进行中西文文字编辑处理。

本书共分五章，第一章主要介绍下拉式系统主菜单的编程方法。第二章介绍光标与屏幕的处理以及编辑文件的输入、修改、插入、删除、存储等基本编辑功能的编程方法。第三章介绍字块处理功能与字符串查找、替换、文章排版以及帮助键应用等辅助功能的编程方法。第四章介绍文件打印功能的编程方法。第五章介绍编辑软件其它功能的编程方法。读者可以根据需要自行选择阅读。

本书主要是面向初步掌握C语言编程以及从事计算机软件开发工作的科技人员的。也可作为中、高等院校计算机软件专业的学习参考书籍。因为是立足于读者对C语言及其编程知识有一定了解的，所以，本书对C语言的基本知识与计算机编程的常用词汇以及Turbo C 的库函数等均未作介绍。读者需要了解这方面的知识，可以查阅有关计算机编程与C语言应用等方面的书籍。也就是说，本书主要告诉读者如何去做。当然，关键之处本书尽可能地予以说明。

限于水平，书中难免会有疏漏与失误之处，恳请读者指正。

作　者

目　录

第一章　下拉式系统主菜单的编程方法 …… 1

第一节　下拉式系统主菜单显示的编程 …… 1
一、定义菜单框架数组 …… 2
二、生成一个菜单框架 …… 2
三、菜单显示的编程 …… 4
第二节　下拉式系统主菜单操作的编程 …… 7
一、接收用户选择菜单响应的编程 …… 8
二、调用系统主菜单的主控模块的编程 …… 10

第二章　基本编辑功能的编程方法 …… 13

第一节　主控模块与文件存储方式的编程 …… 13
一、主控模块的编程 …… 15
二、编辑软件屏幕显示的编程 …… 18
三、数组数据操作的编程 …… 20
四、读写磁盘临时文件中数据的编程 …… 21
第二节　文件名输入与目录显示的编程 …… 24
第三节　光标使用与屏幕操作功能的编程 …… 28
一、光标移动功能的编程 …… 28
二、上下翻屏与光标直达指定行操作的编程 …… 36
第四节　文字输入与显示的编程 …… 40
第五节　文字插入与修改的编程 …… 45
第六节　文字删除的编程 …… 46
一、字符删除的编程 …… 46
二、删除一行文字的编程 …… 49
三、删除光标左边字符串的编程 …… 50
四、删除光标右边字符串的编程 …… 51
第七节　编辑中读入外部文件的编程 …… 52
第八节　存盘与不存盘退出处理 …… 56
一、存盘操作的编程 …… 56
二、不存盘退出处理的编程 …… 59

第三章　辅助编辑功能的编程方法 …… 61

第一节　文字编辑中的字块处理功能及其编程 …… 61
一、字块设定的编程 …… 61
二、字块复制的编程 …… 62
三、字块删除的编程 …… 65
四、字块移动的编程 …… 67
五、字块存盘的编程 …… 68
第二节　文字编辑中的字串查找与替换功能的编程 …… 69
一、字串查找的编程 …… 69
二、字串查找并替换的编程 …… 71
第三节　文字编辑中的排版功能及其编程 …… 72
一、单行排版操作的编程 …… 72
二、全文排版操作的编程 …… 74
第四节　编辑屏幕显示行数变更的编程 …… 74
第五节　编辑操作时帮助键的应用与编程 …… 75

第四章　编辑文件打印功能的编程方法 …… 77

第一节　进入和退出文件的打印状态 …… 77
第二节　窄页纸打印功能的编程 …… 78
第三节　宽页纸打印功能的编程 …… 81
第四节　打印字型与行、列间距的选择 …… 87

第五章　编辑软件其它功能的编程方法 …… 89

第一节　显示模式测试功能的编程 …… 89
第二节　操作信息提示功能的编程 …… 90
第三节　使用磁盘操作系统命令功能的编程 …… 90

附　录　MYWS 编辑软件的源程序清单 …… 92

第一章　下拉式系统主菜单的编程方法

文字处理软件是一个集文字的输入、修改、查找、替换、字块处理、排版、打印乃至一些常用的磁盘操作系统功能在内的自成系统的应用性软件。因此，设计这种软件一般均要为其编制一个系统主菜单。对用户而言，主菜单为其提供了一个友好的对话界面；对程序编制者而言，主菜单的内容往往就是整个软件设计结构的提要。在 MYWS 软件中，设计了如下一套目前比较流行的下拉式系统主菜单。其一级菜单式样如下：

1. 进行文件编辑操作
2. 进行文件打印操作
3. 执行操作系统命令
4. 退出文字处理系统

在主菜单下有两个二级菜单，一个为第一项功能“进行文件编辑操作”的二级菜单，式样如下：

1. 编辑纯文本文件
2. 编辑非文本文件

另一个为第二项功能“进行文件打印操作”的二级菜单，式样如下：

1. 窄页纸打印（16 开）
2. 宽页纸打印（8 开）

下面具体介绍该下拉式主菜单显示与操作的编程方法。

第一节　下拉式系统主菜单显示的编程

用 C 语言设计下拉式菜单，至少要进行以下几个方面的工作：即设置一个结构数组，建立各菜单的框架，编制菜单显示、存储屏幕内容、恢复屏幕内容等有关程序模块。

一、定义菜单框架数组

为了支持菜单框架的建立，需要设立一个存放与菜单有关信息的结构数组。这些信息包括：菜单在屏幕中的坐标位置、菜单内容、菜单项数、选择菜单热键、边框显示及菜单激活情况等等。MYWS 下拉式菜单的结构数组定义如下：

```
struct menu_jg {                          /* 定义菜单结构数组 */
 int beginy, endy, beginx, endx;          /* 存储菜单左上角与右下角坐标分量 */
 unsigned char *p;                        /* 存储屏幕资料的专用指针分量 */
 char * *menu;                            /* 存储菜单内容的指针分量 */
 char *keys;                              /* 存储热键字符串的指针分量 */
 int bder, tiao;                          /* 边框标志及菜单条数分量 */
 int act, att;                            /* 菜单激活标志及显示属性分量 */
} jg [3];                                 /* 结构数组名称 */
```

与其配套的是把要使用的各级菜单内容分别送入到相应存放其内容的指针数组之中，以供显示时调用。

```
char *c00 [] = {                          /* 存放一级菜单内容的指针数组 */
 " 1. 进行文件编辑操作",                  /* 指针数组中第一条菜单的内容 */
 " 2. 进行文件打印操作",
 " 3. 执行操作系统命令",
 " 4. 退出文字处理系统",
};

char *c01 [] = {                          /* 存放二级 1 号菜单内容的指针数组 */
 " 1. 编辑纯文本文件",
 " 2. 编辑非文本文件",
};

char *c02 [] = {                          /* 存放二级 2 号菜单内容的指针数组 */
 " 1. 窄页纸打印 (16 开)",
 " 2. 宽页纸打印 (8 开)",
};
```

二、生成一个菜单框架

有了上述菜单框架结构数组以及存放菜单内容的指针数组，即可通过下述子函数产生一个菜单框架。该函数主要由执行测菜单编号与菜单框架屏幕显示坐标是否越界、计算菜单屏幕显示宽度值、给菜单框架结构数组各分量赋值等方面的指令语句构成。

```
z_menu (num, menu, keys, tiao, y, x, bder, att)
                                        /* 菜单框架生成子函数 */
int num;                                /* 菜单号存储变量 */
char *menu [], *keys;                   /* 菜单内容与热键存储指针变量 */
int tiao, y, x;                         /* 菜单条数与屏幕左上角坐标变量 */
int bder, att;                          /* 菜单边框标志与显示属性变量 */
{
register int i, len=0;                  /* 循环变量与菜单单条长度值变量 */
int endy, endx;                         /* 菜单屏幕右下角坐标变量 */
unsigned char *p;                       /* 存储屏幕内容的中间指针变量 */
if (vgb==0) att=7;                      /* 单色显示器选择显示属性为黑白方式 */
if (num>2)                              /* 测试菜单号越界否 */
 {
 printf (" 菜单号选择超过规定 !");
 return 0;
 }
if ( (y>hs) || (y<0) || (x>79) || (x<0))   /* 测试左上角坐标越界否 */
 {
 printf (" 菜单框架坐标越界 !");
 return 0;
 }
for (i=0; i<tiao; i++)                  /* 测出各条内容的长度 */
if (strlen (menu [i]) >len) len=strlen (menu [i]);
                                        /* 取最长的一条菜单值为基准 */
endx=len+2+x;                           /* 设置菜单右下角坐标值 */
endy=tiao+1+y;
if ( (endy+1>hs) || (endx+1>79))        /* 测试右下角坐标越界否 */
 {
 printf (" 菜单框架坐标越界 !");
 return 0;
 }
p= (char *) malloc (4 * (endy-y+4) * (endx-x+4));
                                        /* 为存储屏幕内容的指针变量申请内存 */
if (! p)                                /* 如果内存不够，则退出运行 */
 {
 for (i=0; i<4; i++) free (jg [i]. p);
 clsy (0x0700, 0, 0x184f); exit (1);
 }
jg [num]. beginy=y; jg [num]. endy=endy;  /* 将菜单参数送入结构数组的各个分量 */
jg [num]. beginx=x; jg [num]. endx=endx;
jg [num]. p=p;
jg [num]. menu= (char * *) menu;
jg [num]. bder=bder;
jg [num]. keys=keys;
```

```
jg [num]. tiao=tiao;
jg [num]. act=0;
jg [num]. att=att;
return (1);                                    /* 返回原调用该子函数的语句处 */
}
```

其中调用的 clsy () 函数是一个自编的清屏子程序，特点是能根据送入参数的情况清除屏幕的不同部分。其实现方法为调用 BIOS 的 10H 中断的 6 号功能。

中断是 PC—DOS 中一种特殊类型的指令，其功能为导致现行程序的中止，在堆栈中保存系统现行状态并且跳转到由中断号决定的中断处理过程。一旦中断过程结束，则将返回中断处，继续执行被中断的程序。

为节省篇幅，在此约定：凡在已引用的程序清单中注释过的变量或指令语句，在其后引用的源程序清单中再次出现且作用相同时，不一定重新注释。同样，在前面已经介绍过的子函数再次出现时，也不再重新讲解。

```
clsy (b, c, d)                                 /* 清屏子函数 */
int b, c, d;                                   /* 显示属性、左上角与右下角坐标变量 */
{
union REGS r;                                  /* 中断调用的联合定义 */
r. x. ax=0x0600;                               /* 将功能号等送入 AX 寄存器 */
r. x. bx=b;                                    /* 将显示属性送入 BX 寄存器 */
r. x. cx=c;                                    /* 将左上角坐标送入 CX 寄存器 */
r. x. dx=d;                                    /* 将右下角坐标送入 DX 寄存器 */
int86 (0x10, &r, &r);                          /* 调用 10H 中断 6 号功能清屏 */
}
```

三、菜单显示的编程

一般来说，在菜单框架生成之后，即可以显示菜单了。但因下拉式菜单是一种允许多个菜单同时出现在屏幕上的多级菜单，所以在显示任一菜单前，都必须通过一个子函数保存该菜单显示前的屏幕状态。这种保存本号菜单显示前屏幕状态的子函数，通过调用 BIOS 的 10H 中断的 8 号功能对该菜单显示坐标内的屏幕进行扫描读屏实现。

```
s_v (num)                                      /* 保存 num 号菜单显示前屏幕的子函数 */
int num;                                       /* 菜单号存储变量 */
{
union REGS r;                                  /* 中断调用的联合定义 */
register int i, j;                             /* 循环变量 */
char *buf_ptr;                                 /* 存储屏幕内容的中间指针变量 */
buf_ptr=jg [num]. p;                           /* 给中间指针变量赋值 */
for (i=jg [num]. beginx; i<jg [num]. endx+2; i++)
```

```
                                        /* 以X坐标终值进行循环 */
for (j=jg [num]. beginy; j<jg [num]. endy+1; j++)
                                        /* 以Y坐标终值进行循环 */
 {
 locate (j, i);                         /* 确定光标位置 */
 r. h. ah=8;                            /* 8号功能 */
 r. h. bh=0;                            /* 0页显示 */
 int86 (0x10, &r, &r);                  /* 调用10H中断8号功能读屏幕内容 */
 *buf_ptr++=r. h. al;                   /* 将读到的内容赋予存屏指针变量 */
 *buf_ptr++=r. h. ah;                   /* 将读到的显示属性赋予存屏指针变量 */
 putchar (' ');                         /* 用空格清除屏幕信息 */
 }
}
```

其中调用的 locate () 子函数是一个光标定位子程序，其编程实现是通过调用 BIOS 的 10H 中断的 2 号功能完成的。

```
locate (y, x)                           /* 光标定位子函数 */
int y, x;                               /* 坐标变量 */
{
union REGS r;                           /* 中断调用的联合定义 */
r. h. ah=2;                             /* 2号功能 */
r. h. dl=x;                             /* 光标位置的列坐标 */
r. h. dh=y;                             /* 光标位置的行坐标 */
r. h. bh=0;                             /* 屏幕显示页号 */
int86 (0x10, &r, &r);                   /* 用10H中断2号功能定位 */
}
```

在保存当前屏幕之后，即可以调用显示菜单内容的子函数以显示菜单。显示要分两步进行。

第一步，调用画边框的子函数划出待显示菜单的边框线。

```
d_bder (num)                            /* 为菜单画边框子函数 */
int num;
{
register int i;                         /* 循环变量 */
for (i=jg [num]. beginy+1; i<jg [num]. endy; i++)
                                        /* 按y坐标循环显示" |" */
 {
 w_a (i, jg [num]. beginx," |", jg [num]. att);
 w_a (i, jg [num]. endx," |", jg [num]. att);
 }
for (i=jg [num]. beginx+2; i< (jg [num]. endx+1); i+=2)
```

```
                                        /* 按 x 坐标循环显示" —" */
 {
 w_a (jg [num]. beginy, i," —", jg [num]. att);
 w_a (jg [num]. endy, i," —", jg [num]. att);
 }
w_a (jg [num]. beginy, jg [num]. beginx,"┌", jg [num]. att);
w_a (jg [num]. beginy, jg [num]. endx,"┐", jg [num]. att);
w_a (jg [num]. endy, jg [num]. beginx,"└", jg [num]. att);
w_a (jg [num]. endy, jg [num]. endx,"┘", jg [num]. att);
}                                       /* 用边框线封菜单外框 */
```

第二步，调用显示菜单内容的子函数显示该菜单内容。

```
d_menu (int num)                        /* 执行显示菜单内容的子函数 */
{
register int i, y, x, mn;               /* 循环、坐标及菜单条长度变量 */
char * * m;                             /* 替代 * * menu 指针分量的中间变量 */
char mp [60];                           /* 单行清屏数组 */
m=jg [num]. menu;                       /* 给中间变量赋值 */
mn=strlen (m [0]);                      /* 给菜单条长度变量赋值 */
for (i=0; i<mn; i++) mp [i] =32; mp [mn] ='\0';
                                        /* 给单行清屏数组赋空格 */
y=jg [num]. beginy+1;                   /* 给坐标变量赋值 */
x=jg [num]. beginx+2;
for (i=0; i<jg [num]. tiao; i++, y++)   /* 以菜单条目作循环 */
 {
 w_a (y, x, mp, jg [num]. att);         /* 向屏幕写入单条空格以清屏 */
 w_a (y, x, m [i], jg [num]. att);      /* 向屏幕写入单条菜单内容 */
 }
}
```

这两个子函数均调用了在屏幕上显示字符串的子函数 w_a ()，其编程通过调用 BIOS 的 10H 中断的 9 号功能实现。

```
w_a (y, x, p, attrib)                   /* 在指定位置写字符串的子函数 */
int y, x, attrib;                       /* 写字符串的坐标与显示属性变量 */
char *p;                                /* 所写的字符串存储变量 */
{
union REGS r;                           /* 中断调用的联合定义 */
register int i;                         /* 循环计数变量 */
if (vgb==0&&attrib! =0x70) attrib=0x7;  /* 如是单显卡则将彩显属性改为单显 */
for (i=x; *p; i++) {                    /* 以字符串长度为终点循环 */
locate (y, i);                          /* 确定写字符的光标位置 */
r. h. ah=9;                             /* 9 号功能 */
```

```
r. h. bh=0;                          /* 屏幕显示页号 */
r. x. cx=1;                          /* 显示该字符的次数 */
r. h. al=*p++;                       /* 要显示的字符送入 AL 寄存器 */
r. h. bl=attrib;                     /* 要显示的字符属性送入 BL 寄存器 */
int86 (0x10, &r, &r);}               /* 用 10H 中断 9 号功能写屏 */
}
```

在菜单选择的操作中，用户有时需要退回到上一级屏幕菜单重新进行选择处理，这时将用到恢复当前菜单显示前的屏幕状态操作的子函数。恢复当前菜单显示前屏幕状态的编程，是一个以调用 BIOS 的中断进行写屏操作为主要内容的子函数。

```
r_v (num)                            /* 恢复 num 号菜单显示前屏幕的子函数 */
int num;
{
union REGS r;                        /* 中断调用的联合定义 */
register int i, j;                   /* 循环变量 */
char *buf_ptr;                       /* 存储屏幕内容的中间指针变量 */
buf_ptr=jg [num]. p;                 /* 给中间指针变量赋值 */
for (i=jg [num]. beginx; i<jg [num]. endx+2; i++)
                                     /* 以 X 坐标终值进行循环 */
for (j=jg [num]. beginy; j<jg [num]. endy+1; j++)
                                     /* 以 Y 坐标终值进行循环 */
 {
 locate (j, i);                      /* 先确定光标位置 */
 r. h. ah=9;                         /* 9 号功能 */
 r. h. bh=0;                         /* 屏幕显示页号 */
 r. x. cx=1;                         /* 显示该字符的次数 */
 r. h. al=*buf_ptr++;                /* 待显示的字符送入 AL 寄存器 */
 r. h. bl=*buf_ptr++;                /* 待显示的字符属性送入 BL 寄存器 */
 int86 (0x10, &r, &r);               /* 调用 10H 中断 9 号功能写屏 */
 }
}
```

以上我们介绍了用 C 语言编制的下拉式菜单的框架建立、显示操作、屏幕内容存储与恢复等有关模块的编程。菜单显示的目的是为了给用户提供一个友好的对话界面，下一节将要介绍如何来使用菜单。

第二节　下拉式系统主菜单操作的编程

对计算机软件设计人员而言，菜单显示的作用为提示用户该软件所具有的可供选择的功能。下面简要介绍菜单选择功能的操作与建立及其调用菜单显示主控模块的实现方法。

一、接收用户选择菜单响应的编程

用户在选择菜单时均希望简单方便，如既可用上下左右的光标键直观地选择，又可以通过菜单中定义的热键直接选择其中的某项功能等。下面这个接收用户响应的子函数，允许用户按其意愿，通过光标键或菜单定义的热键选择运行本软件的某项功能。

```
get_r (num)                                    /* 执行接收用户反应的子函数 */
int num;
{
int a_ch=0, key_ch;                            /* 光标及热键移动计数变量 */
int y, x;                                      /* 屏幕坐标变量 */
y=jg [num]. beginy+1;                          /* 屏幕坐标变量赋值 */
x=jg [num]. beginx+2;
locate (y, x);                                 /* 确定光标位置 */
w_a (y, x, jg [num]. menu [0], 0x70);          /* 反显第一项菜单内容 */
for (;;)                                       /* 接收用户响应操作的循环 */
 {
 while (! bioskey (1));                        /* 等待键盘输入 */
 cy. i=bioskey (0);                            /* 接收键盘输入值 */
 if (cy. ch [0])                               /* 键盘输入码低位不为零的处理 */
   {
  key_ch=is_in (jg [num]. keys, tolower (cy. ch [0]));
                                               /* 测试是否热键 */
  if (key_ch) return key_ch-1;                 /* 是热键返回序号 */
  switch (cy. ch [0])                          /* 非热键先查低八位 */
    {
   case '\r': if (num==9) return (13);
              else return a_ch;                /* 回车键返回光标计数号 */
   case 27:  return -1;                        /* ESC 键返回 -1 */
   }
  }
 else                                          /* 键盘输入码低位为零的处理 */
   {
  locate (y+a_ch, x);                          /* 光标定位位置 */
  w_a (y+a_ch, x, jg [num]. menu [a_ch], jg [num]. att);
                                               /* 将反显项菜单复原 */
  switch (cy. ch [1])                          /* 高位不为零的处理 */
    {
   case 72: a_ch--; break;                     /* 是↑键计数器减 1 */
   case 80: a_ch++; break;                     /* 是↓键计数器加 1 */
   }
  }
```

```
 if (a_ch==jg [num]. tiao) a_ch=0;          /* 光标计数器到底则复零 */
 if (a_ch<0) a_ch=jg [num]. tiao-1;         /* 光标计数器到顶则转底层值 */
 locate (y+a_ch, x);
 w_a (y+a_ch, x, jg [num]. menu [a_ch], 0x70);
 }                                          /* 反显重新选定的菜单项 */
}
```

在该子函数中，除调用前面介绍过的一些子函数之外，还调用了 bioskey（）接收键盘扫描码与 is_in（）测试输入值是否热键两个子函数。

```
bioskey (c)                                 /* 键盘输入测试子函数 */
int c;                                      /* 测试要求的形式参数 */
{
switch (c)                                  /* 根据形参值分别进行处理 */
 {
 case 0: return inkey ();                   /* 零值返回扫描码 */
 case 1: return kbhit ();                   /* 1 则返回键盘状态 */
 }
}

inkey ()                                    /* 接收键盘扫描码的子函数 */
{
union REGS r;                               /* 中断调用的联合定义 */
r. h. ah=0;                                 /* 0 号功能 */
return int86 (0x16, &r, &r);
/* 调用并返回 16H 中断 0 号功能所得代码，其中低八位为键值码，高八位为键位码 */
}

kbhit ()                                    /* 测试键盘状态的子函数 */
{
return ( (char) bdos (0xb, 0, 0));   /* 调用并返回 0xb 号中断 11 号功能所得代码 */
}

is_in (s, c)                                /* 测试热键字符的子函数 */
char *s, c;                                 /* 热键字串及键入字符形式参数 */
{
register int i;                             /* 循环变量 */
for (i=0; *s; i++)                          /* 以字符串长度为终点循环 */
if (*s++==c) return i+1;                    /* 是热键返回序号 */
return 0;                                   /* 非热键返回零 */
}
```

在接收用户响应的子函数中，还使用了如下一个名 cy 的联合。

```
union inkey {                                    /* 接收键盘输入值的联合 */
 unsigned char ch [2];
 int i;
} cy;
```

联合是C语言中的一种构造，可以将不同类型的数据项放在存储器的同一位置中。该联合中设有两个元素，一个是字符型数组ch [2]，一个是整数型变量i，由于它们共处于同一内存位置，所以当执行cy. i=bioskey (0) 指令接收键盘输入值时，联合中的cy. i分量将得到代表这个键盘码的全部数据。同时联合中的ch [0] 和ch [1] 则可分别得到代表这个键盘码的低位扫描数据（即键值码）和高位扫描数据（即键位码）。而其后的编程可以根据这些数据分别处理。

二、调用系统主菜单的主控模块的编程

以上介绍的子函数均为执行某一具体操作的功能模块，一般情况下，这些函数在系统主函数或其它一些需要用到这些功能的函数调用后运行。上述这些主菜单显示和操作的子函数，是在以下系统主函数中被调用：

```
main ()                                          /* 系统主函数 */
{
int i;                                           /* 循环计数变量 */
int dg1, dg2;                                    /* 存储用户返回信息的变量 */
char *z01=" 按 ↑ ↓ 键选择，按菜单号或回车键执行，按 ESC 键返回.";
                                                 /* 操作提示信息变量 */
clsy (0x0700, 0, 0x184f);                        /* 调用清屏子函数操作 */
mod ();                                          /* 调用测定显示模式子函数操作 */
        /* 以下则根据显示卡及显示方式确定下拉式菜单类型 */
if (hs<17)                                       /* 十六行以下的低分辨率屏幕时的处理 */
 {    /* 调用菜单框架生成子函数，按顺序分别生成不同的菜单框架供显示时调用 */
 z_menu (0, c1," 1234", 4, 1, 10, 1, 0x7);
 z_menu (1, c01," 12", 2, 3, 30, 1, 0x7);
 z_menu (2, c02," 12", 2, 3, 30, 1, 0x7);
 }
else if (vgb==0&&hs>16)                          /* 单显高分辨率屏幕的处理 */
 {
 z_menu (0, c1," 1234", 4, 3, 10, 1, 0x7);
 z_menu (1, c01," 12", 2, 6, 30, 1, 0x7);
 z_menu (2, c02," 12", 2, 6, 30, 1, 0x7);
 }
else if (hs>16)                                  /* 彩显高分辨率屏幕的处理 */
 {
 z_menu (0, c1," 1234", 4, 3, 10, 1, 0x17);
 z_menu (1, c01," 12", 2, 6, 30, 1, 0x47);
```