FoxBASE FoxBASE FoxBASE

FoxBASE⁺(2.10)
管理软件开发实用工具

● 陈大正 编著

● 作图标准程序 ● 密码程序设计 ● 系统出错处理
● 软件包装技术 ● 菜单系统设计 ● 主控程序设计

312FO
4

中山大学出版社

(粤)新登字 11 号

版权所有　翻印必究

图书在版编(CIP)目数据

管理软件开发实用工具/陈大正.
广州:中山大学出版社,1994.4
ISBN7-306-00876-5

Ⅰ F…
Ⅱ 陈…
Ⅲ ① 数据库 ② FoxBASE⁺③ 程序设计 ④ 管理软件
Ⅳ TP3

中山大学出版社出版发行
(广州市新港西路 135 号,邮码:510275)
服务电话:(020)4425565
责任编辑:吴相辉　　装帧设计:郑伟贞
中山大学印刷厂印刷
广东省新华书店经销

787×1092 毫米　16 开本　9.5 印张　22 万字
1994 年 9 月第 1 版　　1994 年 9 月第 1 次印刷
印数:0001—5000 册　定价:8.50 元

前　言

FoxBASE⁺(2.10)是当今国内外最流行、应用最广泛的数据库语言之一。用它来开发应用软件，深受广大微机用户欢迎。对开发者来说，直接用数据库语言来编写程序，工作量很大，特别是屏幕格式和报表格式的设计，大约占程序设计工作量的70%；在FoxBASE状态下直接作统计直方图、折线图，也是一件麻烦的事情。

一个应用软件设计开发出来以后，调试、修改、维护以及推广移植的工作量很大而且麻烦，往往出现这样的情况，宁可自己重新设计，也不愿意修改别人的程序。因此，如何把应用软件设计中共同遇到的问题，设计成标准过程和模板程序，提供不同的应用软件共同使用，软件开发者只须在此基础上作第二次开发，从而节省大量的程序设计时间，提高软件开发的效率。这是软件开发者共同的愿望。

在应用软件设计过程中，大概可分为软件包装、菜单系统(下拉)和主控程序设计、功能模块设计、报表生成和打印、统计直方图、折线图生成几个部分。另外，还有系统密码设置和管理；单用户到多用户程序转换以及工作进程显示，系统容错程序等等。

作者根据多年应用软件开发经验，收集、整理、设计和编制了一套解决上述问题的实用工具软件(标准过程和模板程序)，上述各功能都用汇编语言、C语言和FoxBASE⁺(2.10)语言编成标准过程和模板程序。用它来开发应用软件，比直接用数据库语言来编写程序可以节省程序设计工作量一半以上。另外，用它开发出来的应用软件，具有标准化和规范化等优点。方便系统修改、组装、调试、维护和推广移植。

本书第一章叙述软件包装技术，内容包括Turbo C直接读取FoxBASE数据库中的数据，获取汉字点阵数据的方法，汉字字模的结构和汉字的变形、放大及美术效果，C语言中的音乐程序设计，最后给出软件包装的一个标准程序。可供开发者直接使用；第二章叙述菜单系统(下拉)和主控程序设计，内容包括菜单系统(下拉)设计思想，二级、三级下拉菜单设计方法和标准程序，主控程序生成和实例。第三章叙述功能模块设计，内容包括数据的格式编辑、查询和打印程序的设计等。这部分是开发应用软件的主体，我们给出了完整的例子，可供开发者参考；第四章给出其它功能模块的设计。内容包括各种类型的统计直方图、折线图的全自动生成标准过程的设计及其使用方法；系统密码设置和管理；计算机工作进度显示的过程；单用户到多用户的程序处理的标准程序；系统容错程序等。

目前关于FoxBASE语言的书较多，在FoxBASE语言基础上的工具软件较少。本书按软件开发过程所遇的主要问题，应用模块化设计方法，力求把应用软件设计规范化、标准化，书中所述标准过程，收集于一个大过程中，经编译后可放于系统程序中，方便软件开发者使用。因此，本书既是应用软件开发的工具，又是程序设计的一本很好参考书。

本书在编写的过程中，中山大学计算机科学系89届学生吴惠群、李桂玉的毕业论文的部分内容分别收入本书第一章及第二章的第三节中，在此表示感谢。

由于作者水平所限，书中有不妥之处，请多指正。

作者

1994年5月于中山大学计算机科学系

本书所述工具软件及使用例子，全部程序大约 200 KB，装载于一张 360 KB 的软磁盘中，需要者可通过中山大学出版社理科编辑室与作者联系。

（电话：4446300 转 1998）

目　录

此为试读，需要完整PDF请访问：www.ertongbook.com

第一章　软件包装技术

一个应用软件，就像一件礼物一样，当它奉献给用户使用时，需要把它包装起来，这里指的不是磁盘外表的包装，而是指用户使用这个软件时，首先呈现在用户面前的一个画面，如何使这个画面更生动，更吸引人，这就是软件包装技术。软件包装技术涉及的内容广泛，例如如何使画面包含并现的图形、声音、汉字及其变形，而且具有动感等等。更重要的是，如何把这个动态画面用于汉字操作系统环境下的数据库管理系统（例如 FoxBASE⁺(2.10)）中，并且考虑机器内存限制（例如 1M），这个工作用 FoxBASE 命令是很难实现的。更进一步来说，为方便用户使用，即使用高级语言（例如 C 语言）来实现，也应该把它编成标准过程，然后在 FoxBASE 中调用。本章叙述我们用 FoxBASE⁺(2.10)开发应用软件时所用到的主要技术，内容包括：Turbo C 直接读取 FoxBASE 数据库中的数据；获取汉字点阵数据的方法；汉字字模的结构；汉字的变形；汉字的放大及美术效果；C 语言中的音乐程序设计；一个完整的软件包装程序的例子。

第一节　Turbo C 直接读取 FoxBASE 数据库中的数据

为了使程序更具有通用性，与用户有良好的界面，我们把所要显示的汉字信息及颜色等存放于数据库中，因而只要改变数据库的内容，就可以显示不同的汉字串。

为了用 Turbo C 读取 FoxBASE 数据库中的数据，首先必须清楚数据库的结构。一个完整的 FoxBASE 数据库由两部分组成：一部分为数据库结构描述，另一部分为数据库数据内容，而且结构描述在前，数据内容在后。数据库结构描述又可分成两个部分：数据库的整体结构描述与字段结构描述。

数据库的整体结构描述是从数据库的第一个字节开始，共 32 个字节，这 32 个字节的含义分别如下：

字　节	含　　义
0	该字节为 03H 时表示无 Memory 字段，为 80H 时表示 含 .DBT 文件（有 Memory 时产生的文件）
1—3	最后修改数据库的日期，这三个字节分别表示年、月、日。 日期是以二进制来表示的，不是 ASCII 码。
4—7	文件记录数，低位字节在前，高位字节在后，数据用二进制表示。
8—9	结构描述部分的长度，结构描述结束符为 0DH。 长度＝（第 9 字节）＊256＋（第 8 字节）
10—11	记录长度，记录长度＝（第 11 字节）＊256＋（第 10 字节）
12—31	保留，一般为 0。

从第 32 个字节开始，每 32 个字节描述一个字段：

字　节	含　　义
0—9	字段名，以 ASCII 码存放。
10	保留，一般为 0。
11	字段类型，根据不同的数据类型，分别是字母 C、D、L、N 等的 ASCII 码值。
12—15	首记录中该字段对应的内存地址，12—13 为偏移量，14—15 为段地址。
16	字段长度。即字段的字节，最多不超过 256 个字节。
17	数字型(N 型)字段的小数位位数。
18	保留，一般为 0。

字段描述部分结束符(0DH)以后，存储数据文件的各个记录内容。记录以定长格式顺序存储，每个记录的第一个字节是删除标记位，被删除的记录第一个字节为 2AH，即“*”，否则为空格(20H)，每个记录的字段之间没有分隔符，记录无终止符，各种类型的数据均以 ASCII 码存放，文件结束符为 1AH。

明确了数据库的结构，就可以用 Turbo C 关于文件的操作函数读取数据库中的数据。在 Turbo C 中应以二进制只读方式打开数据库文件，由于只需读数据库的记录内容，所以文件位置指针要跳过结构描述部分，而结构描述部分的长度就存放在这一部分的第 8—9 两个字节中，故只要读取这两个字节的内容就可知道数据库结构描述部分的长度，再将 Turbo C 的文件位置指示器从头开始跳过此长度的字节数，就可读取数据库记录的内容。在读取之前，先应在 Turbo C 中定义一个结构，因为数据库中记录的内容不论其类型均以 ASCII 码存储，所以结构的所有成员应定义为字符型数组，其中每个数组的大小与数据库中相应字段的长度相同。

上面介绍的是 Turbo C 读取 FoxBASE 数据库中的记录内容。同样，也可以用 Turbo C 对数据库进行写入、删除或修改记录的操作，这里不再一一介绍。

下面例子读取 TITLE. DBF 中的记录内容，存放在 bigti[]，smallti[]，bigcolor，smallcolor 中。

TITLE. DBF 数据库的结构为：

Field	Field Name	Type	Width	Dec
1	bigtitle	Character	20	
2	smalltitle	Character	40	
3	bigcolor	Numeric	2	
4	smallcolor	Numeric	2	

读取 TITLE. DBF 中的记录内容 Turbo C 程序为：

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

```
#include <string.h>
typedef struct          {/*说明一个与数据库相对应的结构类型*/
    char begin;
    char bigtitle[20]
    char smalltitle[10]
    char bigcolor;
    char smallcolor;
}orig;
main()
  {
    FILE *fp;
    orig *spo;              /*定义结构变量指针*/
    int i,bigcolor,smallcolor;
    char bigti[21],smallti[11],bcolor[3],scolor[3]
    unsigned long j;
    for (i=0;i<21;i++)
      bigti[i]=0;
    for (i=0;i<11;i++)
    smallti[i]=0;
    if( ( fp=fopen("title.dbf","rb"))==NULL) {        /*打开数据库文件*/
    {
       puts("DBF not exist !");
       exit(-1);
     }
fseek(fp,8,SEEK_SET);  /*定位文件位置指针于存储描述部分长度的字节*/
j=fgetc(fp)+256*fgetc(fp);        /*得到描述部分长度*/
while(ftell(fp)<j)          /*定位文件指针于记录内容处*/
     fgetc(fp);
fread(spo,sizeof(orig),1,fp);       /*读一条记录*/
strncpy(bigti,spo->bigtitle,20);
bigti[20]=0;
strncpy(smallti,spo->smalltitle,10);
smallti[10]=0;
strncpy(bcolor,spo->bigcolor,2);
bcolor[2]=0;
bigcolor=atoi(bcolor);          /*字符型转换为整型*/
strncpy(scolor,spo->smallcolor,2);
scolor[2]=0;
```

```
smallcolor=atoi(scolor);
fclose(fp);
```

数组的最后一个元素赋空，这是因为当数据库中的数据为定义的字段长度时，由于各字段间无空格，同时 Turbo C 对数组变量也不作边界检查，这就可能造成读取的一个字段数据中包括了下一个字段，采取上述办法后，就可以避免这种错误。

另外，由于数据库记录之间有一个空格，因此，在结构变量中也同样定义了一个没有用的结构成员 begin(这里未考虑这个位可存放数据库记录的删除标记"*")。

根据以上所述，在编写读取 FoxBASE 数据库中的 Turbo C 程序时，必须先进入 FoxBASE 环境中查看数据库的结构，才可以定义一个与之对应的结构变量。但如果直接读取数据库结构描述部分的相应字节，得到字段个数和每个字段长度，再用定义指针和动态分配内存的方法，就可以编写一个通用的按记录方式读取. DBF 的 TURBO C 程序。

第二节　获取汉字点阵数据的方法

众所周知，显示图形信息时，往往需要用汉字进行说明，并且需要汉字信息与图形信息共存取，而涉及到汉字的程序一般需要在汉字系统下运行，但加载汉字系统占用内存过多，当程序中用到汉字很少，例如只作显示版权说明或输出有关信息时，我们可以在西文方式下显示汉字。解决这一问题，可以先读取汉字的点阵数据，再用图形方式显示输出，这样就能实现美术放大汉字，在屏幕上任意位置显示所需要颜色的汉字。

归纳起来，取得汉字点阵数据的方法有两种。一种是直接读取汉字库，另一种是采用中断调用。两种方法各有所长，下面分别介绍之。

1. 中断调用获取汉字系统点阵数据

中断调用获取点阵数据是所有汉字系统均具有的功能。这种方法比较直接，但它需要加载汉字系统，而且当需要用 Turbo C 实现图形显示时，又将涉及到 Turbo C 与汇编的接口等问题。而且不同的汉字系统采用不同的中断调用。

SPDOS 关于汉字的中断调用是 INT 16H。

AH＝80/90　　读字库
　DX＝国标码或 ASCII 码
　ES:BX＝缓冲区
　CL＝0　　读16×16点阵字库
　　＝1　　读24×24点阵字库
　　＝6　　读宋体点阵
　　＝7　　读仿宋体点阵
　　＝8　　读楷体点阵
　　＝9　　读黑体点阵
　SI ＝点阵高度(8—256)
　DI＝点阵宽度(8—256)

如为西文字体，则 CL＝6—16

AH＝81H　卸去 SPDOS

下面介绍的程序，是在金山系统下，使用中断调用获得24×24汉字点阵数据，codehigh是国标码高位字节，codelow 是国标码低位字节，addr 是存放点阵数据的起始位置。只要把这段汇编程序嵌入 C 语言程序即可，但如前所述，这需要在中文操作系统下运行才行。

```
void readc(int codehigh,int codelow,int addr)
  {
    asm{
      pushbp
      movbp,sp
      push es
      push ds
      pop es
      mov dh,[bp+6]          /* 取得 codehigh */
      mov dl,[bp+8]          /* 取得 codelow */
      mov bx,[bp+10]         /* 取得 addr */
      mov cl,1           /* 读24×24点阵字库 */
      mov ah,90h
      int 16h
      pop es
      pop bp
    }
   return;
  }
```

另外，UCDOS 下是 INT 6AH，CCBIOS 2.13H cc 版为 INT 10H 中12H 功能，GW 版为 INT10H 中16H 功能（一般情况下，有汉字显示卡如 CEGA、CVGA 的机器为16H 功能，软汉字系统如 CCDOS 2.1、CCDOS 4.0 则为12H 功能）。

2. 直接读取汉字库方法

直接读取汉字库 CCLIB 的方法，需要利用内码来计算点阵数据在字库中的位置，比较繁琐，却能动态地处理汉字，而且不用装载汉字系统，少占内存，并且不用考虑汇编与 c 的接口问题，这是它的长处所在。

一个汉字，在机器内部以两个内码表示，（内码1—160）为区号，（内码2—160）为位号，即在该区中第几个汉字，而每个区存放94个汉字，所以，在某汉字之前有：（内码1—160—1）＊94＋（内码2—160—1）个汉字。24点阵每个汉字占72个字节，因此，某汉字字模之前的字节数为：

$$[(\text{内码}1-160-1)*94+(\text{内码}2-160-1)]*72$$

又因为汉字两个内码均大于127，故应加上256，而256—161＝95，所以，汉字字模在字库中的位置为：

此为试读，需要完整PDF请访问：www.ertongbook.com

offset=(long)((内码1+95)*94+(内码2+95))*72

如果是16点阵汉字，则每个汉字占32个字节，所以只需把72改为32即可。

找到汉字字模在字库中的位置以后，可利用fread()，fopen()，fseek()，fclose()，fgetc()等文件操作的函数读取汉字点阵。

下面的函数hanzstr()显示汉字串，clib24为24点阵字库，只须给它一汉字串，就可按给定的颜色在指定的位置显示汉字串。

```
void hanzstr(char *hz,int x,int y,int col)
  /*hz 代表汉字串 x,y 代表点的横纵坐标 col 为彩色编号*/
  {
  char wordmatrix[72];
  long offset1;
   char *s=hz;
    if((fp=fopen("c:clib24","rb"))==0){ /*打开字库只读 */
       printf("Can not open the filen");
       exit(1);
    }
    while(*s){
       offset1=(long)((*s+95)*94+(*(s+1)+95))*72;
       fseek(fp,offset1,SEEK_SET);      /*定位文件指针*/
       fread(wordmatrix,1,72,fp);     /*读点阵数据*/
       display(x,y,col);       /* 显示单个汉字*/
       x+=24;       /* 移动横坐标*/
       s+=2;      /* 指向下一个汉字*/
    }
     fclose(fp);
    }
```

函数display()在后面介绍了字模结构后再给出。

第三节　汉字字模的结构

取到汉字点阵数据以后，要显示汉字，必须知道汉字字模的结构，也就是要了解点阵信息是按什么顺序存放的。

16点阵字库中，每个汉字字模占32个字节，其点阵信息按行顺序存放，一般情况下，是按图1-1所示排列。但对于金山系统，却如图1-2所示。

图 1-1

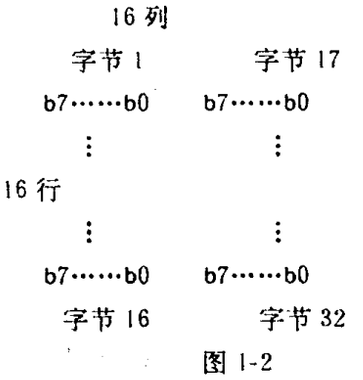

图 1-2

24 点阵字库中，每个汉字字模占 72 个字节，其点阵信息按列顺序存放。如图 1-3 所示排列。

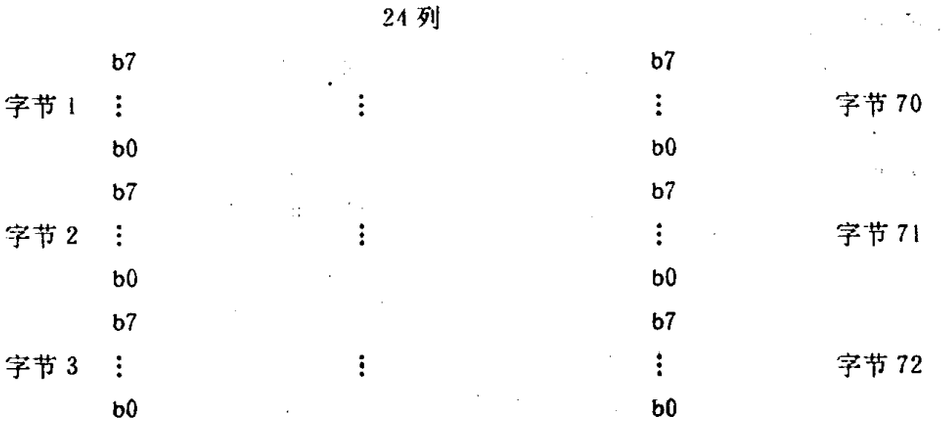

图 1-3

关于汉字点阵信息的处理，根据图 1-1 和图 1-2，程序中可用 3 重循环，外层分别控制 16 个行，中层控制每行中的两个字节，内层分别控制每个字节中的 8 个位，判断每位是 1 或 0 决定是否写点，从而可在屏幕的相应位置上显示出字型。

根据图 1-3，也用 3 重循环，外层分别控制 24 个列，中层控制每列中的 3 个字节，内层分别控制每个字节中的 8 个位。

下面就实现 24 点阵的显示给出 display()函数，wordmatrix[72] 存放了字模点阵的 72 个字节。

```
void display(int x1,int y1,int col)
  {
    int i,j,k,p,x,y;
    for(i=0;i<24;i++){
      x=x1+i;
      for(j=0;j<3;j++){
        y=y1+8*j;
        p=0x80;
        for(k=0;k<8;k++){
```

```
            if(wordmatrix[i*3+j]&p) /* 判断字模 7～0 位是否为 1 */
              putpixel(x,y+k,col);
              p=p>>1;
            }
          }
        }
      }
```

第四节　汉字的变形

一种点阵字库的汉字转换成矢量汉字后，字的基本大小是固定的。在实际输出时，还需要根据具体的要求进行各种变形变换，汉字的变形变换有纵横放缩和错切三种。

1. 纵横放缩变换

设 x，y 方向的放缩因子为(k_x,k_y)，则点(x,y)经放缩变换后变为(x_1,y_1)：

$$x_1=x*k_x \qquad y_1=y*k_y$$

2. 错切变换

设 x，y 方向的错切因子为(l_x,l_y)，则点(x,y)经错切变换后变为(x_1,y_1)：

$$x_1=x+y*l_x \qquad y_1=y+x*l_y$$

l_x 表示 x 方向的错切，取值为$-1\leqslant l_x\leqslant 1$，$l_x>0$ 时，向正方向错切；$l_x<0$ 时，向负方向错切；$l_x=0$ 时，x 方向不错切。l_y 同理。

为了计算方便，在系统设计时，l_x,l_y 一般取值 $-1,-0.5,-0.25,0,0.25,0.5,1$，因为对这些值在计算时可直接移位计算，另一方面，这些值基本满足了应用的要求。

错切因子的大小，决定了汉字倾斜扭动的角度。例如 A 点的坐标为(12,10)，经 X 方向错切后 A_1 坐标为$(12+10*L_x,10)$，汉字的倾斜角度为 α，$ctg\alpha=L_x$。

通常的汉字输出是两种变形的合成，先放缩后错切，合成变换为：

$$x_1=x*k_x+y*k_y*l_x$$
$$y_1=y*k_y+x*k_x*l_y$$

第五节　汉字的放大及美术效果

要显示美观的汉字，还必须对汉字进行放大和美术加工。

汉字的放大，有很多种方法，最简单的方法是以多个点代替一个点，字模中每一位在水平方向变成 n 个点，则实现水平方向放大 n－1 倍，字模中每一位在垂直方向变成 n 个点，则实现垂直方向放大 n－1 倍，放大的幅度可根据 n 的改变而改变。另外，也可以根据放大倍数和起始坐标、偏移量，用绘图语句 rectangle()或 circle() 画出，然后再进行填充，放大的幅度由半径的大小进行调节。

汉字的美术效果，可以有汉字的重叠，汉字的错开等。

对于字的重叠，我们知道，在字符中，可用的 ASCII 字符代码为 33～127，每个代码对

应一个字符，我们用一个字符写完一个字后；可再用另一个字符在同一位置写同一个字；多个字符在同一位置写同一个字，可以产生另一种效果。在同一位置用不同字符写同一个字时，可以写完后重新移笔到原来原点；也可以设置绝对间隔为 0。

对于字的错开，在空心字和实心字中，字的错开可以产生较好的立体效果和运动感。如在(0,0)的位置写一个字，然后再在(1;0),(2;0),(3,0)位置分别写同样大小的同一个字，则相当于一个字沿水平方向向右移动了三步，每一步都留下影子，产生动态和立体感。字的错开可沿水平方向或任意方向，错开的次数和步长可任意确定。

另外，可根据需要，对汉字实行水平显示或者垂直显示，是否旋转 90 度显示等变换。在了解了字模的结构以后，对于这些功能的实现是不难的，只须对 diaplay 函数稍加修改即可。

例如，旋转 90 度，可按下述程序实现。

```
void display1(int x1,int y1,int col)
  {
    int i,j,k,p,x,y;
    for(i=0;i<24;i++){
      y=y1+i;
      for(j=0;j<3;j++){
        x=x1-8*j;
        p=0x80;
        for(k=0;k<8;k++){
          if(wordmatrix[i*3+j]&p)    /* 判断字模 7—0 位是否为 1 */
            putpixel(x-k,y,col);
          p=p>>1;
        }
      }
    }
  }
```

第六节　C 语言中的音乐程序设计

音乐是时间的艺术。把各种音符按不同的时值演奏出来，就可以构成曲调。因此，音乐程序设计中的两个重要的因素是：如何用“曲调定义语言”来表示音符，即音高，如何控制音符的持续时间，即音长。

音调由音符构成，音调的高低由音符的频率决定，频率越高，音调也越高。音乐中使用的频率一般为 131～1976Hz，它包括了中央 C 及其前后 1 个 8 度的音程。这 1 个 8 度中各个音符的频率如表 1-1 所列(带 * 的为中央 C 调)。

表 1-1 音符及其频率

音符	频率(Hz)	音符	频率(Hz)
C	131	C	523
D	147	D	587
E	165	E	659
F	175	F	698
G	196	G	784
A	220	A	880
B	247	B	988
C*	262	C	1047
D	294	D	1175
E	330	E	1319
F	349	F	1397
G	392	G	1568
A	440	A	1760
B	494	B	1976

音长即一个音符的持续时间。在乐曲中，音长用全音符、半音符、4 分音符、……来表示，通常以 4 分音符为一拍。因此，全音符有 4 拍，8 分音符为半拍，等等。

在 C 语言中，音高可以用枚举类型来定义，音长可用枚举类型或 #define 来定义。表 1-1 中所列各音符用枚举类型定义如下：

```
enum NOTES
  {
    C10=131,D10=147,E10=165,F10=175,G10=196,A10=220,B10=247,
    C0=262,D0=296,E0=330,F0=349,G0=392,A0=440,B0=494,
    C1=523,D1=587,E1=659,F1=698,G1=784,A1=880,B1=988,
    C2=1047,D2=1175,E2=1319,F2=1397,G2=1568,A2=1760,B2=1976
  }
```

音长可以定义如下：

```
#define WHOLE 2400
#define HALF WHOLE/2
#define QUARTER WHOLE/4
#define EIGHTH WHOLE/8
#define SIXTEENTH WHOLE/16
```

定义中的 WHOLE、HALF、QUARTER、EIGHTH 和 SIXTEENTH 分别表示全音符、半音符、4 分音符、8 分音符、16 分音符。乐曲的速度可以通过改变 WHOLE 的值来调整，该值越大，速度越慢。根据不同的机器，经过试验来确定适当的 WHOLE 值。

C 语言中，提供了 sound()，delay()，nosound()等函数，利用这些函数，可以控制计算机的扬声器发声，实现发出指定频率，指定音长的操作。

第七节　一个综合标准程序

在前面所介绍的知识的基础上，下面我们给出一个软件包装的标准程序 PAPER. C，它是用 C 语言编写的，用户只要给出数据库 TITLE. DBF 中的记录内容，它提供系统的大标题、小标题以及颜色参数，程序就能按你所需要的标题及颜色生成一个图形、美术字、音乐并现的软件包装画面。

大标题经过放大移位美化后，斜放在屏幕的中间位置，而且两边对称，呈八字形；小标题经过放大移位美化后，平放在屏幕的上方位置，左右对称；屏幕的下半部安排一幅彩色图象。

程序运行后，首先画出屏幕下半部彩色图象，然后大小标题先后自动从屏幕右边动态进入自己的位置，整个画面完成后，屏幕循环改变底色并等待，用户按任意键后屏幕进入第二幅画面，这个画面和第一幅画面基本相同，但屏幕上方不停地生成彩色随机点，并配有音乐伴奏，直到用户按任意键来结束程序。

用户使用这个程序时，完全不必修改源程序，只需修改数据库 TITLE. DBF 的内容，就能为别的系统使用。

由于这个程序中的某些子程序在前面各节均有介绍，而且程序中有较为详细的注释，所以，这里不再重复，图 1-4 是程序流程图。其有关说明如下：

① 程序运行环境：IBMPC/XT、AT、386 及其兼容机、显示器为 EGA 或 VGA。

② 所需要文件：TITIE. DBF、PAPER. EXE、EGAVGA. BGI、CCLIB. DAT。其中 PAPER. EXE为 PAPER. C 编译后执行文件，CCLIB. DAT 为汉字库文件，TITIE. DBF 为要用户提供的数据库文件，EGAVGA. BGI 为图形初始化文件。

③ TITIE. DBF 数据库结构及内容。TITIE. DBF 数据库结构如下：

Field	Field Name	Type	Width	Dec
1	bigtitle	Character	20	
2	smalltitle	Character	40	
3	bigcolor	Numeric	2	
4	smallcolor	Numeric	2	

各数据项意义如下：

bigtitle—— 系统名称大标题（10个汉字），smalltitle—— 系统名称副标题（20个汉字），bigcolor—— 系统名称大标题颜色代码，smallcolor—— 系统名称副标题颜色代码。

例如，我们给出数据如下：

bigtitle＝"计算机信息管理系统"。

smalltitle＝"中山大学计算机科学系于一九九四年一月研制"。

bigcolor＝9。

smallcolor＝10。

此为试读，需要完整PDF请访问：www.ertongbook.com