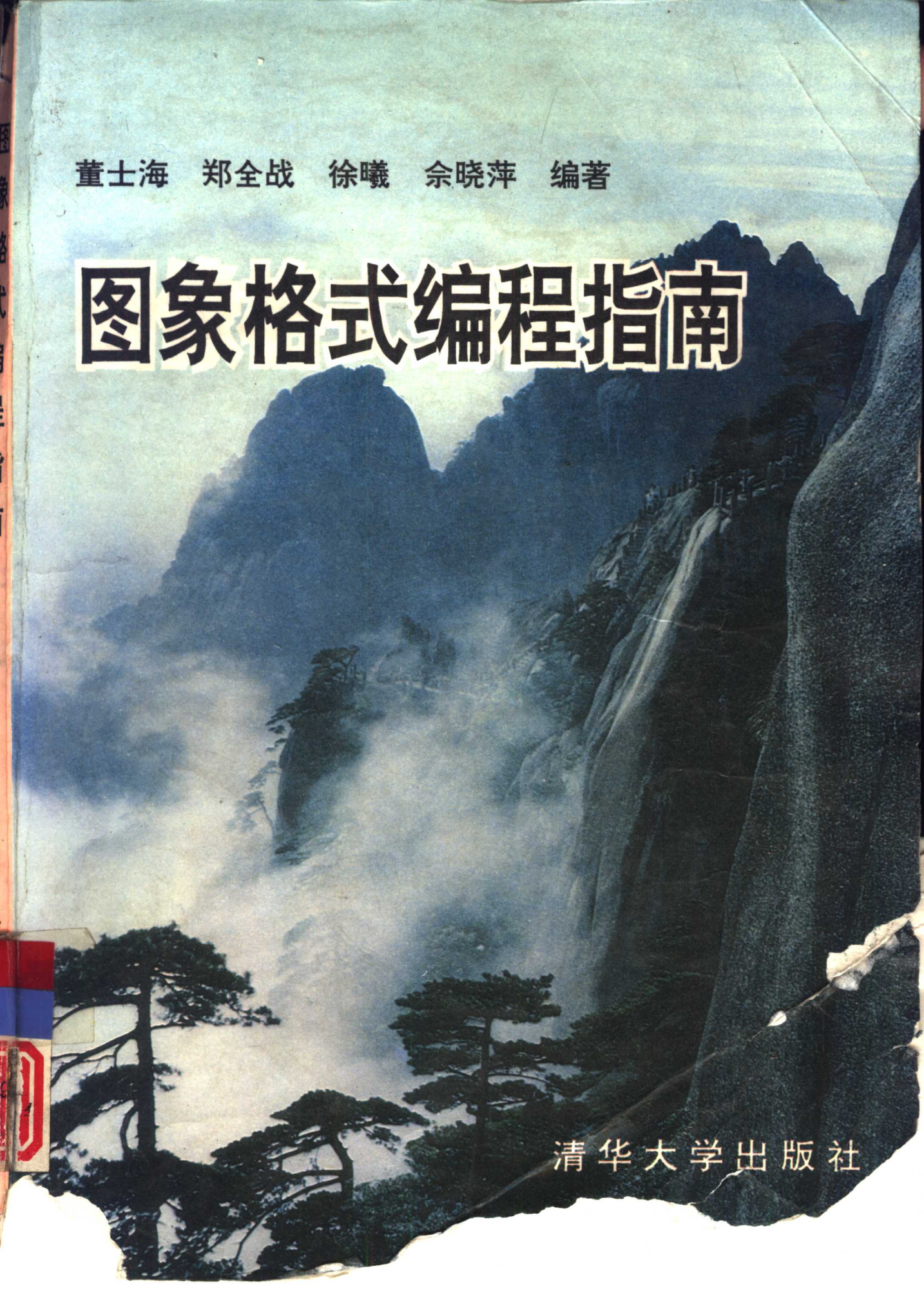
董士海　郑全战　徐曦　佘晓萍　编著

图象格式编程指南



清华大学出版社

图象格式编程指南

董士海　郑全战　徐　曦　佘晓萍　编著

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书讲述了 PC 机环境下最常见的图象格式，如 GIF、TIFF、BMP、PCX、IMG 等，以及处理图象时需要涉及的 CGA、EGA、VGA、超级 VGA 显示卡的显示驱动，PostScript 打印机，抖动技术和图象格式间的相互转换。

本书在介绍图象格式时提供了大量的程序实例，并全部上机调试通过，相信会对读者有所帮助。全部源程序和编译后的可执行程序都存放在一张高密盘上，由清华大学出版社软件部正式出版发行，该软盘定价 120 元。

本书可供 PC 机软件开发和应用人员参考，对计算机图象处理有关的人员也很适用。

图象格式编程指南

董士海　郑全战　徐　曦　余晓萍　编著

☆

清华大学出版社出版

北京　清华园

国防工业出版社印刷厂印刷

新华书店总店科技发行所发行

开本：787×1092 1/16　印张：18.5　字数：438 千字

1994 年 4 月第 1 版　1994 年 4 月第 1 次印刷

印数：0001—5000

ISBN 7-302-01436-1/TP・561

定价：15.80 元

前　言

计算机图象在计算机领域正占据越来越重要的地位。高分辨率显示器、扫描仪、高档激光打印机为高质量图象的显示、输入和输出提供了硬件环境，CorelDRAW、Micrografx Designer、Microsoft PowerPoint、Photostyler 等图形图象处理软件为图象的处理提供了很好的软件环境。正在兴起的多媒体技术，其中一个重要的内容就是对图象的支持。

图象资源正变得越来越丰富。这些资源能否为我所用？当我们涉及这个问题时，就应该去了解各种各样的图象格式。

本书讲述了 PC 机环境下最常见的几种图象格式，如 GIF、TIFF、BMP、PCX、IMG 等，以及处理图象时需要涉及的 CGA、EGA、VGA、超级 VGA 显示卡的显示驱动，PostScript 打印机的图象输出，抖动技术和图象格式间的相互转换。

本书具体章节安排如下：

第 1 章总体上介绍了图象格式的解压缩、可能的文件头以及处理图象时需要注意的问题。

第 2 至 7 章依次介绍 MacPaint、IMG、PCX、BMP、GIF、TIFF 图象格式的解压缩以及生成过程，每章都有具体而丰富的程序实例。

第 8 章与第 9 章分别介绍 CGA、EGA、VGA、超级 VGA 显示卡的单色与彩色显示驱动。

第 10 章讲述 PostScript 打印机输出图象时的程序驱动。

第 11 章讲述图象处理中的重要技术——抖动技术。

第 12 章介绍图象格式间的相互转换。

本书在介绍图象格式时提供了大量的程序实例，并全部上机调试通过，相信会对读者有所帮助。本书全部源程序可通过清华大学出版社软件部（邮编：100084）购取。

本书由郑全战、徐曦、佘晓萍共同编写，由董士海、郑全战最后定稿。在录入时得到了倪仲君的大力支持，在此致谢。

由于编写者水平有限，书中错误在所难免，欢迎广大读者给予批评、指正。

编　者

于北京大学计算机系

1993 年 6 月

目　录

第 1 章　有关图象的介绍

图象是一个古老的话题。在原始社会，我们的祖先就想到了应该为后代留下点什么，亦或是为了“流芳后世”，他们开始用“象形文字”记事，用一些能够描绘具体事物的符号来记录事情。随着人类社会的发展，信息量越来越大，人们再也无法用繁杂的“象形文字”来记录事情了，于是出现了现代文字。现代文字简则简矣，却失去了一目了然的优点。计算机的发展，使人类“返朴归真”的愿望成为现实。超文本(Hypertext)系统、多媒体(Multimedia)系统，图、文、声并茂，使人如临其境。图象在其中扮演着重要的角色。

“一幅图胜似千言万语”。当我们说祖国山河如何美丽的时候，我们并没有确切的感受，而当我们看了一些照片，或者一组风景画的时候，我们才会真正领略到它们的无限风光。

有必要提一下，图象并不是图形。计算机图形通常指计算机所绘制(draw)的东西，像圆、矩形、图表等。本书所要论述的图象则是指由扫描仪、摄像机等输入设备输入并存储到计算机中的数字信息，它能够在计算机屏幕上再次显现出来。本书主要讲述的位图(bitmap)，是由数字阵列信息组成的。

每一种图象软件几乎都用各自的方法处理图象，这也是图象处理中的最大问题，像 MacPaint、PC Paintbrush，就用不同的格式存储图象。如果你想利用现有的图象文件，或者在不同的软件中使用图象，就要注意图象格式的不同。可能需要进行必要的格式转换。

如果你想欣赏一幅图象文件，你还需要有关显示卡的性能指标，如模式、分辨率等。如果想把图象打印出来，同样也需要知道打印机的各种参数。

现在已有许多不同的图象格式，并且继续有新的格式出现。但是，有几种图象格式是众所周知的，像 TIFF、PCX、BMP 等，新出现的图象软件都会支持它们，否则这样的软件也不会具有强大的生命力。本书着重介绍以下几种比较流行的图象格式：MacPaint、PC Paintbursh、PCX、Microsoft Bitmap(BMP)、CompuServe GIF 以及 TIFF。本书还介绍了一些有关显示卡和打印机进行图象处理的内容，并介绍了一种图象处理技术——抖动(dithering)，最后介绍图象格式间的相互转换。

1.1　图象文件结构

图象在计算机屏幕上显示时，我们才能知道它的内容。为了简单起见，我们先讨论一幅单色(黑白)图象，假设它的长宽为 640×480，恰好为 VGA 显示卡的分辨率。这时如果用 EGA、CGA 显示卡显示，则只能显示它的一部分。

VGA 显示卡的分辨率水平方向为 640 个象素，垂直方向为 480 个象素，整屏共有 307200 个象素。显示在屏幕上的数据就存储在内存中，与存储其它程序代码和数据一样。位图(黑白图象)中的每一个象素只占一位，而一个字节包含八个象素，这样，上面的位图

存储在文件中只需要 38400 个字节。

如果知道 VGA 显示卡将位图存储在段值 A000H 处，我们就可以很容易地将位图的 38400 个字节保存到一个文件中。C 语言代码如下：

```
FILE *fp;
if(fp = fopen("SCRVGA.BIN","wb")) != NULL){
  fwrite(MK_FP(0xa000,0),1,38400,fp);
  fclose(fp);
} else
  printf("Creating file error !");
```

在 Turbo C 系列中，有 MK_FP 宏，而在 Microsoft C 语言中则没有，我们可如下定义：

```
# ifndef MK_FP
# define MK_FP(seg,ofs)\
    ((void far *)(((unsigned long)(seg<<16) | (ofs))
# endif
```

这个程序执行得很快，但很浪费存储空间。如果图中大部分是空白，存储这种原始数据，就会使文件中数据冗余量太大。

在本书中我们将陆续介绍彩色图象，它们用原始数据存储，将会占用几百 KB，甚至将近1MB。慢慢就会发现，用尽可能少的数据量来保存一幅位图是非常重要的。

图1-1的第一条扫描线全是空白，它占用80个字节，其实它可压缩为二个字节，甚至是一个字节。当我们想把图象从文件中重新读入内存，并显示在屏幕上时，如果不是要求一字节、一字节照实地读入内存，那么我们在写入文件时就可以采用一些技巧，将数据压缩。

下面介绍一种简单的图象压缩方案，它与 MacPaint 文件的压缩方法很类似。我们假设图象数据已被压缩到文件中，下面就看一下如何将这些数据解压缩，重新显示在屏幕上。

压缩图象的数据中共有三种类型：

·关键字节：它告诉解压缩程序，后面是什么样的数据包。

·索引字节：标识当前数据包内将包含多少数据。

·数据部分：压缩后的原始数据。

数据包是可以用相同方法处理的数据集合。一个数据包可以和一条扫描线一样长，也可以很短。关键取决于是否能高效率地压缩数据。

这里我们只介绍两种类型数据包。

第一种称为字节行程数据包，由三个字节组成。第一个字节是0xff，是本数据包的关键字节，它将告诉解压缩程序如何解码。第二个字节是0到80 间的一个数字，表明第三个字节的重复次数，即行程长度。很明显，第二个字节不可能为0，上限限制为80是为了不超过一个扫描行的长度。第三个字节才是真正的数据字节。

如果解压缩程序碰到如下的数据包：

```
0xff 0x09 0x55
```

它将9个字节的0x55显示到屏幕上。

第二种类型的数据包称为字节串数据包。一个字节串数据包的长度在3到82 之间。第

一字节是0x00，标识此数据包为字节串数据包，第二个字节表明随后字节串的长度，其余的字节都是不能压缩的原始数据。

下面就是一个字节串数据包：

0x00 0x09 0x65 0x12 0xa6 0x77 0x01 0x76 0x69 0xf1 0x98

解压缩程序将把0x65到0x98之间的9个字节显示到屏幕上。

这样，解压缩程序通过识别关键字节来处理两种不同的数据包，直至全部数据处理完毕。

解压缩程序的流程图如下：

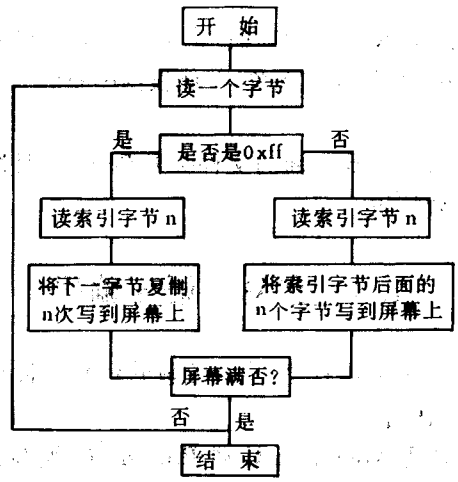


图 1-1 一个针对 VGA 显示卡的简单的图象解压缩过程

1.2 更可行的方案

上面所讲的方案还有一些缺点。关键字节和索引字节没必要占用两个字节，可以将它合并为一个字节，为了方便，称它为控制字节。让控制字节的最高位是否为1，来标志当前数据包是行程字节数据包还是字节串数据包。控制字节的低7位标识数据长度，因为 VGA 的每个扫描行是80个字节，所以用7位(可标识127 个字节)就足够了。

或许你对这种精打细算不以为然，慢慢就可以看到，这样一字节一字节地节约，一旦图象大一些，就能节约很可观的空间。本书一些图象文件格式的压缩方案就是很“经济”的。

上面的压缩方案存在的另外一个问题是：只适合于 VGA 显示卡。如果上述方案应用于别的显示卡(如 Hercules 显示卡)，图象解压缩之后，显示效果可能就变得很糟糕，这是因为 Hercules 显示卡的分辨率与 VGA 显示卡的分辨率并不相同。

还有一个问题是：解压缩程序无法识别一个文件是否是图象压缩文件。它可能是一个文本文件，也可能是一个别的什么文件。如果解压缩程序盲目地去解压缩一个未知文件，

就很可能陷入困境。

解决这些问题的最好办法是给图象文件加一个文件头。文件头是一个约定好的数据结构，它包含图象的长、宽大小以及解压缩程序可识别这类图象的特定标志，这个特定标志可以是一个唯一的字符串。

下面是一个可能的图象文件头数据结构：

```
typedef struct {
        char signature[5];
        int width,depth;
}HEADER;
HEADER myHeader={"PCIMG",640,480};
```

如果将这个文件头加在图象文件的前面，这个图象就能被正确地显示在其它显示卡上，因为相应的解压缩程序能识别这类图象，并能根据图象的大小和显示卡的分辨率来正确地显示图象。

大多数图象格式的文件头都比上面的文件头复杂一些，因为每一种图象格式都由于特定的原因而增加一些有效标识。

1.3 有关C语言编程

如果你想更好地利用本书的源程序，应该比较熟练地掌握C语言。如果你想改进本书的程序，那最好熟悉汇编语言。

本书的C语言程序是用Microsoft C或Borland C编写的，而汇编程序则是用MASM 6.0编写的。本书附录列出了Microsoft C与Borland C(Turbo C)之间的主要差别，供读者参考。

1.4 内存模式及其它注意事项

位图图象一般都很大。小一点的图象一般都是几十KB，几百KB；而大的图象往往达几MB，几十MB，PC机几乎无法处理。本书只限于比较小的图象。

有关PC机内存结构方面的书籍很多，这里不再多讲。64KB之内和超过64KB的处理方法是不同的，对于图象处理，这一点尤为重要。图象的大小一般都超过64KB，因此我们重点讨论超过64KB的内存处理方法。

在小内存模式中，内存寻址只需用一个近指针即可，因为内存大小仅限于64KB。在大模式中，内存寻址需用远指针，它包含一个段值和一个偏移值。处理超过64KB的图象，就需要使用远指针，以保证正确地处理图象数据。

在大模式中，处理大内存有很多困难之处。下面的例子会说明这一点。我们假设指针p指向一块大内存的起始处，将p进行下列运算：

```
p += 0xffff;
p++;
```

这时，你可能认为 p 将指向大内存块的0x10000L 处，但是，事实上并非如此。它可能又指向了内存的起始处，也可能是更靠前一些。

这是因为，在 Microsoft C 语言中，大模式中的指针运算只影响偏移值，段值并不改变。这意味着只靠指针的简单加减运算无法处理超过64KB 的连续内存区域。

这对于超过64KB 图象的处理是一个重要的问题。我们将用下面的函数 farptr()来解决这个问题。farptr 函数的功能是将一个远指针加上一个长整数，即将远指针偏移一定大小。

```
char far * farptr(char far *p,long l)
{
    unsigned int seg,off;

    seg = FP_SEG(p);
    off = FP_OFF(p);
    seg += off/16;
    off += (unsigned int)(l & 0x000fL);
    seg += (l/16L);
    p = MK_FP(seg,off);
    return(p);
}
```

本书中的程序将会多次使用 farptr 函数。

1.5 包含文件

在第8章、第9章中，我们开发了单色以及彩色显示驱动程序，它们都由汇编语言写成，使用的汇编语言编译器是 MASM 6.0。为了其它程序模块或其它程序的使用方便，我们提供了一个汇编语言的包含文件(ASM.INC)和一个 C 语言包含文件(ASM.H)，分别如图1-2、图1-3所示。

```
;--------------------------------------------------------------------------
;       Graphics mode for video cards
;--------------------------------------------------------------------------
CGA                     EQU     1
MCGA                    EQU     2
VGA                     EQU     2
EGA                     EQU     3
HERCMONO                EQU     7
PARADISE                EQU     59H
LARGEPIX                EQU     1 ;SET TRUE FOR SOURCE IMAGES > 64K

; * LoadPtr - Macro to load far address into segment:register pair, or
; * near address into register.
; *
; * Params:   sgmnt - Segment to be loaded with segment address
; *           reg - Register to be loaded with offset address
; *           pointer - Pointer to address
; *
```

```
; * Shows:    Instructions - lds     les
; *            Directives - MACRO IF          IFIDNI    ELSE
; *                         ELSE   IFENDIF   .ERR      ENDM
; *            Operators - < > ;;

LoadPtr MACRO sgmnt, reg, pointer          ;; Macro definition
    IF @DataSize                           ;; If far pointer, and
       IFIDNI <sgmnt>, <ds>                ;;   if 1st argument is DS,
          lds reg, pointer                 ;;   load DS:reg with far address
       ELSEIFIDNI <sgmnt>, <es>            ;;   or if 1st argument is ES,
          les reg, pointer                 ;;   load ES:reg with far address
       ELSE                                ;; Generate error if not DS or ES
          .ERR <First argument must be DS or ES>
       ENDIF
    ELSE                                   ;; If near pointer,
        IFIDNI <sgmnt>, <es>               ;;   and if segment is ES,
           push ds                         ;;   ensure ES points to
           pop es                          ;;   same segment as DS
        ENDIF
        mov reg, pointer                   ;; Then load reg with near address
    ENDIF
ENDM

PBYTE    TYPEDEF          PTR BYTE    ; Type for pointer to bytes
PWORD    TYPEDEF          PTR WORD    ; Type for pointer to words
PSWORD   TYPEDEF          PTR SWORD   ; Type for pointer to integers
PDWORD   TYPEDEF          PTR DWORD   ; Type for pointer to integers
NPBYTE   TYPEDEF NEAR     PTR BYTE    ; Type for near pointer to bytes
FPBYTE   TYPEDEF FAR      PTR BYTE    ; Type for far pointer to bytes
FPVOID   TYPEDEF FAR      PTR         ; Type for far pointer to void
PSEG     TYPEDEF WORD                 ; Type for segment value

;-------------------------------------------------------------------------------
;       PROTOTYPES FOR CHAPTER 8 & CHAPTER 9
;-------------------------------------------------------------------------------

;in MONOSCR.ASM (f8-1.asm)
Update          PROTO PASCAL line:FPBYTE,linelength:SWORD,linenumber:SWORD
DoTable         PROTO PASCAL cardno:SWORD

;in HERCPAGE.ASM (f8-6.asm)
HercPage                PROTO PASCAL pageno:SWORD

;in HERCGRAF.ASM (f8-8.asm)
IsHercGraf      PROTO PASCAL

;in CHNGMODE.ASM (f8-9.asm)
GraphicsMode    PROTO PASCAL graphmode:SWORD
TextMode                PROTO PASCAL

; common functions (f9-1.asm and f9-5.asm)
Gif2Ega         PROTO PASCAL egaline:FPBYTE,gifline:FPBYTE,linelength:SWORD
Text                    PROTO PASCAL

; in EGASCR.ASM (f9-1.asm)
ChangeToEgaPalette      PROTO PASCAL egapalette:FPBYTE,gifpalette:FPBYTE,num-
```

```
                    ber:SWORD
PaletteEGA      PROTO PASCAL palette:PBYTE,number:SWORD
EGAgraphics     PROTO PASCAL
ShowEGA         PROTO PASCAL line:FPBYTE,linelength:SWORD,linenumber:SWORD

; in VGA16SCR. ASM (f9-5. asm)
ChangeToVgaPalette    PROTO PASCAL egapalette: FPBYTE, gifpalette: FPBYTE, number:
                      SWORD
PaletteVGA16    PROTO PASCAL palette:PBYTE,number:SWORD
VGA16graphics   PROTO PASCAL
ShowVGA16             PROTO PASCAL line: FPBYTE, linelength: SWORD, linenumber:
                      SWORD

; in VGASCR. ASM (f9-3. asm)
SetVgaPalette   PROTO PASCAL palette:PBYTE,number:SWORD
VGA256graphics PROTO PASCAL
VGAText          PROTO PASCAL
ShowVGA256      PROTO PASCAL line:FPBYTE,linelength:SWORD,linenumber:SWORD
FarMove         PROTO PASCAL dest:FPBYTE,src:FPBYTE,l:WORD

;in PARSCR. ASM (f9-7. asm)
SetParPalette   PROTO PASCAL palette:PBYTE,number:SWORD
PARgraphics     PROTO PASCAL
PARtext         PROTO PASCAL
ShowPAR     PROTO PASCAL line:FPBYTE,linelength:SWORD,linenumber:SWORD
```

图 1-2　ASM. INC 文件

```
/*-------------------------------------------------------------------------------------------------*
 *        Graphics mode for video cards
 *-------------------------------------------------------------------------------------------------*/
#define CGA             1
#define MCGA            2
#define EGA             3
#define HERCMONO        7
#define PARADISE        0x59
#define VGA             MCGA

/* in MONOSCR. ASM */
void _pascal Update(char far *line,int linelength,int linenumber);
void _pascal DoTable(int cardno);

/* in HERCPAGE. ASM */
void _pascal HercPage(int pageno);

/* in HERCGRAF. ASM */
int _pascal IsHercGraf();  /* if graphics mode then return non_zero else return zero */

/* in CHNGMODE. ASM */
void _pascal GraphicsMode(int graphmode);
void _pascal TextMode();

/* in EGASCR. ASM & VGA16SCR. SAM */
void _pascal Gif2Ega(char far *egaline,char far *gifline,int linelength);
void _pascal Text(void);

/* in EGASCR. ASM */
```

```
void _pascal ChangeToEgaPalette(char far *egapalette,char far *gifpalette,int number);
void _pascal PaletteEGA(char *palette,int number);
void _pascal EGAgraphics(void);
void _pascal ShowEGA(char far *line,int linelength,int linenumber);

/* in VGA16SCR.ASM */
void _pascal ChangeToVgaPalette(char far *egapalette,char far *gifpalette,int number);
void _pascal PaletteVGA16(char *palette,int number);
void _pascal VGA16graphics(void);
void _pascal ShowVGA16(char far *line,int linelength,int linenumber);

/* in VGASCR.ASM */
void _pascal SetVgaPalette(char *palette,int number);
void _pascal VGA256graphics(void);
void _pascal VGAText(void);
void _pascal ShowVGA256(char far *line,int linelength,int linenumber);
void _pascal FarMove(char far *dest,char far *src,unsigned int l);

/* in PARSCR.ASM */
void _pascal SetParPalette(char *palette,int number);
void _pascal PARgraphics(void);
void _pascal PARtext(void);
void _pascal ShowPAR(char far *line,int linelength,int linenumber);
```

图 1-3　ASM.H 文件

许多程序，像 FASTPCX.C、SHOWBMP.C 等程序，都要使用 ASM.H 文件。

第 2 章　MacPaint 图象文件

MacPaint 图象文件不是 PC 机的产物，它来源于 Apple Macintosh。在 Macintosh 环境中，几乎所有的应用程序都采用这种图象格式；而在 PC 环境中存在着种类繁多的图象格式。

我国由于 Macintosh 机比较少，MacPaint 格式的图象也见得比较少。而在美国，使用 Macintosh 机比使用 PC 机更普遍，因此 MacPaint 格式图象也非常多，从电子布告板(bulletin－board)上能够获取许多有趣的图象。

与本书的其它图象格式相比，MacPaint 图象格式比较简单。它多数是黑白的，并且具有固定的大小。无论一幅什么样的 MacPaint 图象，它的水平方向总是576 个象素，垂直方向总是720个象素。

在 MacPaint 图象中，带有38个图案(pattern)，这与 PC 环境下的其它格式的图象也不相同。这些图案类似于 PCX，BMP 等格式图象中的调色板。在 PC 环境下的 MacPaint 图象，其中的图案并无多大用处。

在 Macintosh 中，所有的文件都存储为两部分。一部分是数据，另一部分是资源，资源部分包含代码。MacPaint 文件只包含数据部分，资源部分是空的。

当 MacPaint 文件被应用到其它系统时，如 PC 系统，它形成一个单一的文件，这个文件包含数据和资源两部分，另外还要加上一个 MacBinary 文件头。文件头的作用在于当 MacPaint 文件又回到 Macintosh 系统时，解释如何将文件分割为数据和资源两部分。另外，文件头还记录文件的类型及原文件名。Macintosh 文件名可长达31个字符。

MacBinary 文件头严格说来不是文件格式的一部分。在 Macintosh 文档资料中根本不存在这一数据结构。不过在 PC 环境下，我们还是把它作为 MacPaint 文件的一部分。本章在最后一部分对这一结构有详细描述。

MacPaint 文件中的图象数据位于 MacBinary 文件头和图案数据之后，并且是经压缩存储的。

2.1　文　件　头

在显示一幅图象之前，首先要确认一个文件是否是所需要的文件格式；另外还要知道图象的大小是否合乎要求。由于 MacPaint 图象都具有统一的大小，因此处理起来就比较方便。

MacBinary 文件头两个长整数，分别定义了 Macintosh 文件类型和文件创建者。每个文件中唯一的文件类型都是一个4字节代码。这个4字节代码既可看作4个字节，也可以看作一个长整数。MacPaint 文件类型是 PNTG，每个能产生文件的程序还有一个4字节标识，或称为创建者标志。如果文件来源于 MacPaint ，则创建者标识为 MPNT。因为还有

许多 Macintosh 应用程序也能产生 MacPaint 文件，所以不能认为创建者标识一定为 MPNT。

MacBinary 文件头中的文件类型域位于0x0041到0x0044之间，如果这4个字节是 PNTG，则可以断定这个文件是 MacPaint 图象文件。

在 Macintosh 中，MacPaint 图象文件的文件名保存在第二个字节起始处。文件头的第一个字节通常为0。文件名按 Pascal 风格字符串存储，这意味着字符串的第一个字节是一个表明字符串长度的数值，字符串不一定以 NULL 结尾。

图案数据起始于第132字节偏移处，也就是 MacBinary 文件头结束后的4 个字节处(MacBinary 文件头长为128个字节)每个图案是8个字节。

我们先编写一个小程序，对 MacPaint 文件做一些基本的判断工作并显示它的图案。为了简单起见，我们调用 Turbo C 语言图形库中的函数来显示图案，在后面我们将用更快的方法显示。

```
/*
 *      A program to look at MacPaint file patterns
 */

#include "dos.h"
#include "stdio.h"
#include "alloc.h"
#include "graphics.h"

/* how big each swatch will be */
#define blockSize          32

char header[640]; /* where the header lives */

main(argc,argv)
int argc;
char *argv[];
{
        FILE *fp;

        if(argc == 1)
        {
                printf("Usage: %s filename\n",strupr(argv[0]));
                return(1);
        }

        /* attempt to open the file */
        if( (fp=fopen(argv[1],"rb")) == NULL )
        {
                printf("Open %s error.\n",argv[1]);
                return(1);
```