用行程编码对 WINDOWS 位图压缩的方法

杨　勇　（华中理工大学 430074）

目前 MS-WINDOWS 操作系统以其优良的图形介面赢得了广大计算机用户的喜爱，对位图(DIB)的使用也越来越多。但这些位图一般数据量都比较大，占用较多的硬盘空间。WINDOWS 推荐用 RLE(Run Length Encode 行程编码)算法加以压缩。由于 WINDOWS 中的定义不实用，一般用户很少压缩自己的位图。本文给出用有限行程长度编码压缩位图的方法。该方法较为简单，程序实现较为容易。特别适合于画笔(paintbrush)画出的图形。其压缩量依赖于被压缩图象，一般压缩比在 2:1 到 5:1 之间(压缩比＝被压缩图象的存储量/压缩后图象的存储量)。这种简单的不失真压缩算法，能有效地减少图象存储空间占用量，可作为图象处理系统的子模块。

WINDOWS 的图象格式是位图(Bitmap)图象格式，以文件形式存储的位图被称为与设备无关的位图(DIB)。其结构由截然不同的两部分组成：一个是描述位图的大小和颜色的 BITMAPINFO 结构，另一个是位图阵列。本文的压缩只处理位图阵列。BITMAPINFO 结构的介绍可参考其它资料，这里只介绍所用到的几个变量：

biWidth 位图的宽度

biHeight 位图的高度

biBitCount 每个象素所占的位数(1,4,8,24)

bf0ffBits 文件中存储位图阵列的开始位置相对于文件头的偏移量

与设备无关的位图的原点在位图的左下角。位图象素阵列从位图最下面开始，每行从最左边的象素开始，每个象素阵列中的 1 位(4、8、24)，对应颜色表中的某一个表项，从而获得该象素的颜色。对于 256 色的位图，每个象素占 8 位(一个字节)，通过该字节的值得到颜色表 256 个表项中的一个来获得象素的颜色。

以 256 色的图象为例说明 RLL(Run Length Limited)编码。RLL 编码用一个重复计数段(repcount)和一个数据段(data)，它们各占一个字节，来表示象素的行程。其中数据段存放该象素的值。重复计数段的最高 2 位置 1，这是重复计数段的标志，低 6 位是重复计数值，即行程长度。取值的范围从 0xc0 到 0xff，所以其行程长度的最大值不超过 63，称为有限行程长度编码。若象素值大于 0xc0 且不等于下一个象素值，则压缩后要增加一个字节(重复计数)。这是 RLL 编码的一个缺陷。

例如：象素值(十六进制)　　压缩值(十六进制)

02 05 05 05　　02 C3 05

B1 A6 A6 A6　　C1 B1 C3 A6

以下是以 256 色图象为例，给出压缩(compress)和解压(decompress)两个函数。在 COMPAQ 486 上用 TURBO C2.0，BORLAND C++ 3.1 调试通过。读者可自行开发 16 色压缩程序，也可直接用本程序压缩。

```
#include <stdio.h>
#include <process.h>
typedef unsigned char BYTE;
typedef unsigned short WORD;
typedef unsigned long DWORD;
typedef unsigned int UINT;
typedef signed long LONG;
typedef struct tagBITMAPFILEHEADER
{
    UINT bfType;
    DWORD bfSize;
    UINT bfReserved1;
    UINT bfReserved2;
    DWORD bf0ffBIts;
}BITMAPFILEHEADER;
typedef struct tagBITMAPINFOHEADER
{
    DWORD biSize;
    LONG biWidth;
    LONG biHeight;
    WORD biPlanes;
    WORD biBitCount;
    DWORD biCompression;
    DWORD biSizeImage;
    LONG biXPelsPerMeter;
    LONG biYPelsPerMeter;
    DWORD biClrUsed;
```

```
    DWORD biClrImportant;
}BITMAPINFOHEADER;
BITMAPFILEHEADER bitfile;
BITMAPINFOHEADER bitinfo;
FILE * fpi, * fpo;
long offset, sum, count, width, height;
int i, repcount;
char data, dataty;
void compress(char filei[12], char fileo[12]
/* filei 是被压缩的位图文件名, fileo 是压缩后的位图文件名
*/
{
    if ((fpi = fopen(filei, "rb")) == NULL)
    {printf("can't open DIB file");exit(1);}
    if ((fpo = fopen(fileo, "wb")) == NULL)
    {printf("can't create a file");exit(1);}
    fread (&bitfile, sizeof(BITMAPFILEHEADER), 1, fpi);
    fread (&bitinfo, sizeof(BITMAPINFOHEADER), 1, fpi);
    offset = bitinfo.biWidth;
    height = bitinfo.biHerght;
    rewind(fpi);
    for(i = 0;i < offset;i++) fprintf(fpo, "%c", getc(fpi));
    /* 把 BITMAPINFO 结构存入压缩文件中 */
    if (bitinfo.biBitCount == 4) sum = (width * height)/2
    /* 计算 16 色位移图阵列大小 */
    if (bitinfo.biBitCount == 8) sum = width * height;
    /* 计算 256 色位移图阵列大小 */
    count = 01;
    data = getc(fpi);
    do
     {
     datatp = getc(fpi);
     count++;
     repcount = 1;
     while((data == datatp) && (repcount < 63) && (count <
sum))
       {repcount++;
         datatp = getc(fpi);
         count++;
       }
       if ((repcount > 1) || ((unsigned int )data > 0xbf))
        {
        repcount += 0xc0;
        fprintf(fpo, "%c", repcount);
       }
       fprintf(fpo, "%c", data);
         data = datatp;
    }while(count < sum);
    fclose(fpi);
    fclose(fpo);
    return;
}
void decompress (char filei[12], char fileo[12])
/* filei 是压缩文件的文件名 fileo 是解压后的文件名 */
{
    if ((fpi = fopen(filei, "rb")) == NULL)
    {printf("can't open a compressed file");exit(1);}
    if ((fpo = fopen (fileo. "wb")) == NULL)
    {printf ("can't create a file");exit(1);}
    fread (&bitfile, sizeof(BITMAPFILEHEADER), 1, fpi);
    fread (&bitinfo, sizeof (BITMAPINFOHEADER), 1, fpi);
    offset = bitinfo.biWidth;
    height = bitinfo.biHeight;
    rewind(fpi);
    for (i = 0;i < offset;i++) fprintf(fpo, "%c", getc(fpi));
    if (bitinfo.biBitCount == 4) sum = (width * height)/2;
    if (bitinfo.biBitCount == 8) sum = width * height;
    count = 01;
    do
    {
    data = getc(fpi);
    if(((data&0xc0) >> 6) == 3)
    {
    repcount = data&0x3f;
    data = getc(fpi);
    for (i = 0;i < repcount;i++)
    {
    fprintf(fpo, "%c", data);
    count++;
    }
    }
    else
    {
       fprintf(fpo, "%c", data);
       count++;
    }
    }while(count < sum);
    fclose(fpi);
    fclose(fpo);
    return;
    }
```