**编程获取bmp图像的信息**

**胡雅源**

（软件工程1班20172700）

**摘要**

BMP的图像分为1位、2位、4位、8位、16位和24位、32位，其中16、24、32位的图像不包含调色板结构信息，除此之外，其余的图像都包含调色板结构信息。BMP图像除了调色板信息包含在内，还有每个BMP文件都有的结构信息，如BMP文件头、BMP信息头、还有BMP像素数据信息。所以BMP文件的结构一次为BMP文件头，BMP信息头，[调色板]，像素数据信息。这个实验是写程序度BMP文件信息，分析BMP结构信息。首先分析BMP文件结构，然后实现代码，定义位图文件头、位图信息头、调色板的结构。读取位图信息，实现读取图片，解读bmp图像的信息。然后显示出来。结果可以通过打开位图文件的数据信息查看，验证准确性，尤其是像素数据信息。调色板和位图数据分析没有实现。所以只能分析位图文件头和位图信息头。

1. **BMP结构**

**(一)、BMP文件头**

BMP文件头结构体定义：

typedef struct tagBITMAPFILEHEADER

{

unsigned short int bfType; //位图文件的类型，必须为BM

unsigned long bfSize; //文件大小，以字节为单位

unsigned short int bfReserverd1; //位图文件保留字，必须为0

unsigned short int bfReserverd2; //位图文件保留字，必须为0

unsigned long bfbfOffBits; //位图文件头到数据的偏移量，以字节为单位

}BITMAPFILEHEADER;

BMP文件头数据表1如下：



表1

**（二）、BMP信息头**

typedef struct tagBITMAPINFOHEADER

{

long biSize; //4bytes,该结构大小，字节为单位

long biWidth; //4bytes,图形宽度以象素为单位

long biHeight; //4bytes,图形高度以象素为单位

short int biPlanes; //2bytes,目标设备的级别，必须为1

short int biBitcount; //2bytes,颜色深度，每个象素所需要的位数

short int biCompression; //4bytes,位图的压缩类型, 0:不压缩，1：RLE8，2：RLE4

long biSizeImage; //4bytes,四字节对齐的数据图像大小，以字节为单位

long biXPelsPermeter; //4bytes,位图水平分辨率，每米像素数

long biYPelsPermeter; //4bytes,位图垂直分辨率，每米像素数

long biClrUsed; //4bytes,位图实际使用的调色板索引数，0：使用所有的调色板索引

long biClrImportant; //4bytes,重要的调色板索引数，0：所有的调色板索引都重要

}BITMAPINFOHEADER;

信息头数据表2如下：



表2

**（三）、BMP调色板**

Typedef struct \_tagRGBQUAD

{

BYTE rgbBlue; //蓝色强度

BYTE rgbGreen; //绿色强度

BYTE rgbRed; //红色强度

BYTE rgbReserved; //保留，0

}RGBQUAD;

颜色表中RGBQUAD结构数据的个数由biBitCount来确定，biBitCount=1代表有两个表象，为2有四个表项，为8有256个表项，当biBitCount=1或4或8时，调色板字节长度分别为2\*4=8个字节、16\*4=64个字节、256\*4=1024个字节。当biBitCount=16,24,或32时，该位图没有颜色表。

1. **BMP图像数据区**

位图的图像数据区记录了位图的每一个像素值，扫描顺序是从左到右，从下到上，当biBitCount为1，表示8个像素占一个字节，当biBitCount=4时，表示2个像素占一个字节，当biBitCount为8时，表示1个像素占一个字节，当biBitCount为24时，表示1个像素占3个字节，当biBitCount为16时，表示1个像素占两个字节。

且一个扫描行所占的字节数是4的倍数，可以用0补充。

一个扫描行所占字节数的计算方法：

DataSizePerLine=(biWidth\*biBitCount+31)/8;

确保字节数是4的倍数:DataSizePerLine=DataSizePerLine/4\*4;

位图数据的大小：DataSizePerLine\*biHeight;(在不压缩的情况下)。

图像信息的大小：信息大小=（图像宽度\*图像高度\*记录像素的位数）/8；

1. **图像信息读取**

**（一）、读取文件头信息：**

位图信息头的定义结构：

typedef struct tagBITMAPFILEHEADER

{

//unsigned short int bfType; //位图文件的类型，必须为BM

DWORD bfSize; //文件大小，以字节为单位

WORD bfReserved1; //位图文件保留字，必须为0

WORD bfReserved2; //位图文件保留字，必须为0

DWORD bfbfOffBits; //位图文件头到数据的偏移量，以字节为单位

}BITMAPFILEHEADER;

位图文件头的读取方式，打印方式。其中pBmpHead为BMP文件中指向文件头的指针。

void showBmpHead(BITMAPFILEHEADER\* pBmpHead)

{

printf("位图文件头：\n");

printf("文件大小：%d\n",pBmpHead->bfSize);

printf("保留字：%d\n",pBmpHead->bfReserved1);

printf("保留字：%d\n",pBmpHead->bfReserved2);

printf("实际位图数据的偏移字节数：%d\n",pBmpHead->bfbfOffBits);

}

**（二）、读取信息头信息**

**1. 信息头结构**：

typedef struct tagBITMAPINFOHEADER

{

DWORD biSize; //该结构大小，字节为单位

LONG biWidth; //图形宽度以象素为单位

LONG biHeight; //图形高度以象素为单位

WORD biPlanes; //目标设备的级别，必须为1

WORD biBitcount; //颜色深度，每个象素所需要的位数

DWORD biCompression; //位图的压缩类型

DWORD biSizeImage; //位图的大小，以字节为单位

LONG biXPelsPermeter; //位图水平分辨率，每米像素数

LONG biYPelsPermeter; //位图垂直分辨率，每米像素数

DWORD biClrUsed; //位图实际使用的颜色表中的颜色数

DWORD biClrImportant; //位图显示过程中重要的颜色数

}BITMAPINFOHEADER;

然后以函数showBmpInforHead(tagBITMAPINFOHEADER \* pBmpInforHead)显示信息头信息：

void showBmpInforHead(tagBITMAPINFOHEADER\* pBmpInforHead)

{

printf("位图信息头：\n");

printf("结构体的长度：%d\n",pBmpInforHead->biSize);

printf("位图宽：%d\n",pBmpInforHead->biWidth);

printf("位图高：%d\n",pBmpInforHead->biHeight);

printf("biPlanes 平面数：%d\n",pBmpInforHead->biPlanes);

printf("biBitCount 采用颜色位数：%d\n",pBmpInforHead->biBitcount);

printf("压缩方式：%d\n",pBmpInforHead->biCompression);

printf("biSizeImage 实际位图数据占用的字节数：%d\n",pBmpInforHead->biSizeImage);

printf("X方向分辨率：%d\n",pBmpInforHead->biXPelsPermeter);

printf("Y方向分辨率：%d\n",pBmpInforHead->biYPelsPermeter);

printf("使用的颜色数：%d\n",pBmpInforHead->biClrUsed);

printf("重要颜色数：%d\n",pBmpInforHead->biClrImportant);

}

**（三）、关于调色板的数据读取，与分情况讨论问题：**

typedef struct tagRGBQUAD {

BYTE rgbBlue;

BYTE rgbGreen;

BYTE rgbRed;

BYTE rgbReserved;

}RGBQUAD;

这是调色板的数据结构。调色板可以由两个四字节的数值组成，也可以是由16个，256，1024个组成，具体多少个四个字节的数值，由信息头的颜色位数决定。颜色位数一般是1，2，4，8，16，24，32。一个颜色包括四个字节，最后一个字节是保留位，置0，然后从四个字节的右边往左第二个字节开始，分别是红色、绿色、蓝色的颜色强度值，决定这个颜色。

void showRgbQuan(tagRGBQUAD\* pRGB)

{

printf("(%-3d,%-3d,%-3d)",pRGB->rgbRed,pRGB->rgbGreen,pRGB->rgbBlue);

}

这是一个四字节的颜色数值信息。调色板由多个这样的结构组成的。

因为计算机画图程序只能保存单色，16色，256色和24位的bmp图片，所以用这四张图片来做实验。分别是t.bmp(单色)，t00.bmp(16色),t01.bmp(256色),t02.bmp(24位)。测试的结果都是正确的，如下图1，图2，图3.1,图3.2，图4 。

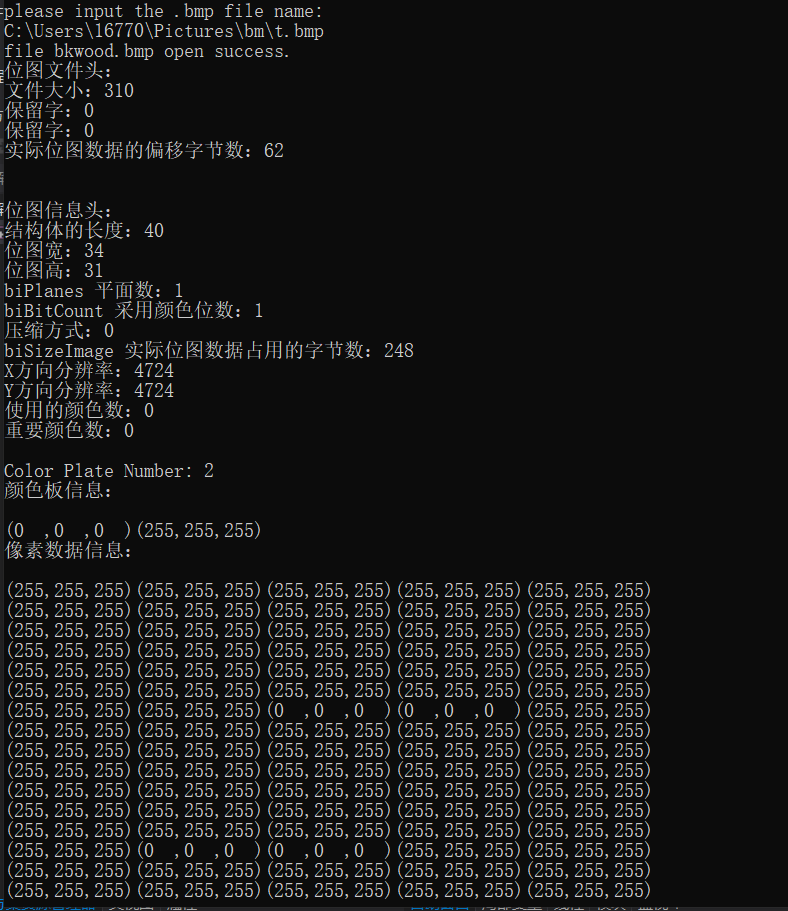


图1

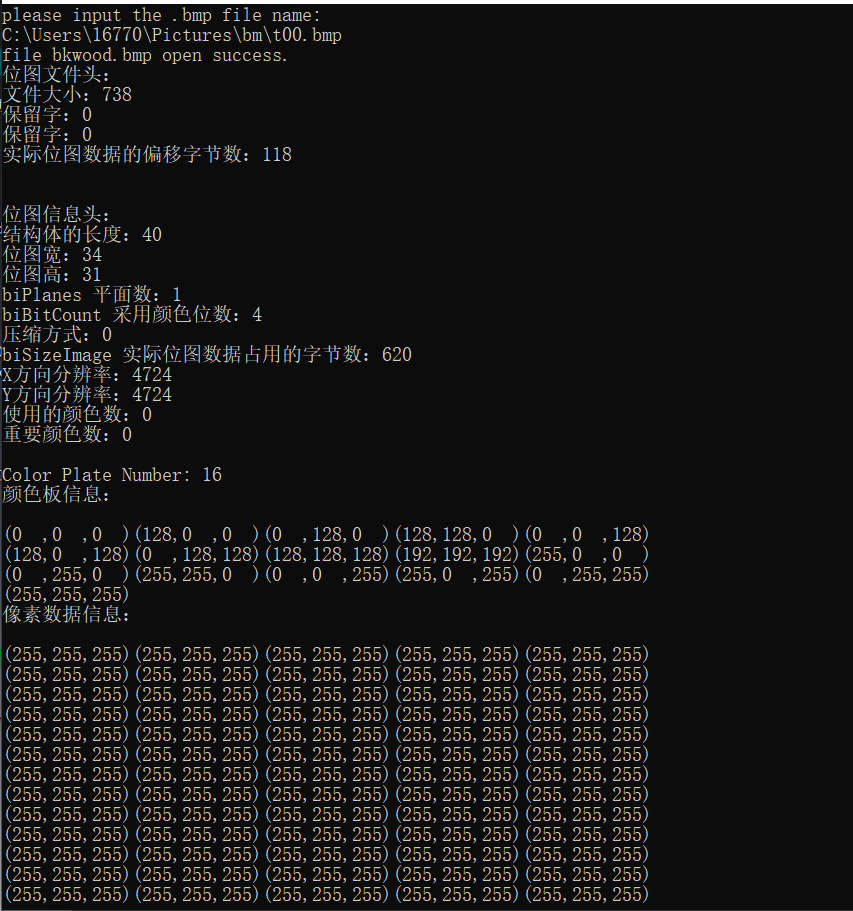


图2

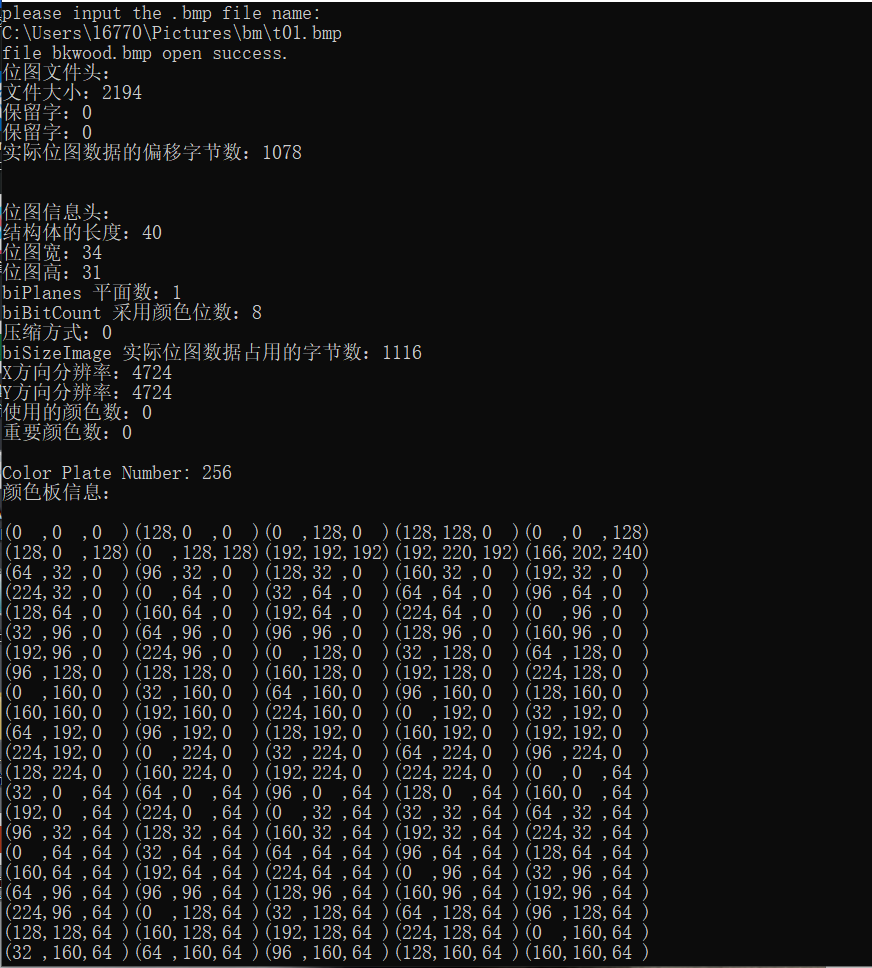


图3.1

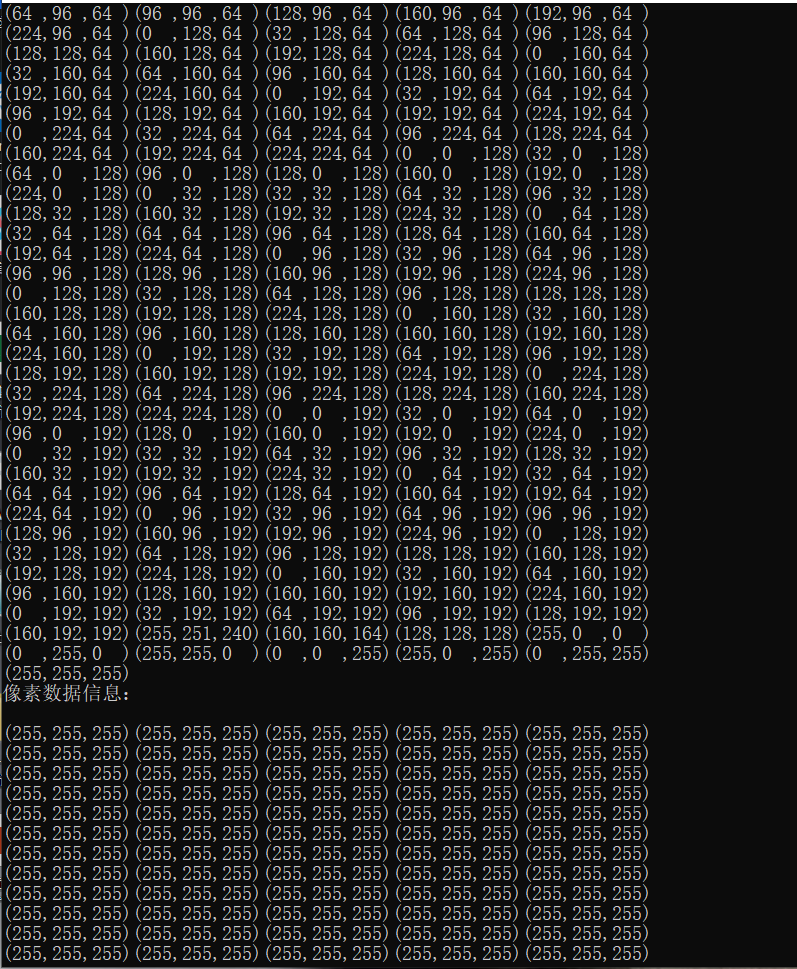


图3.2

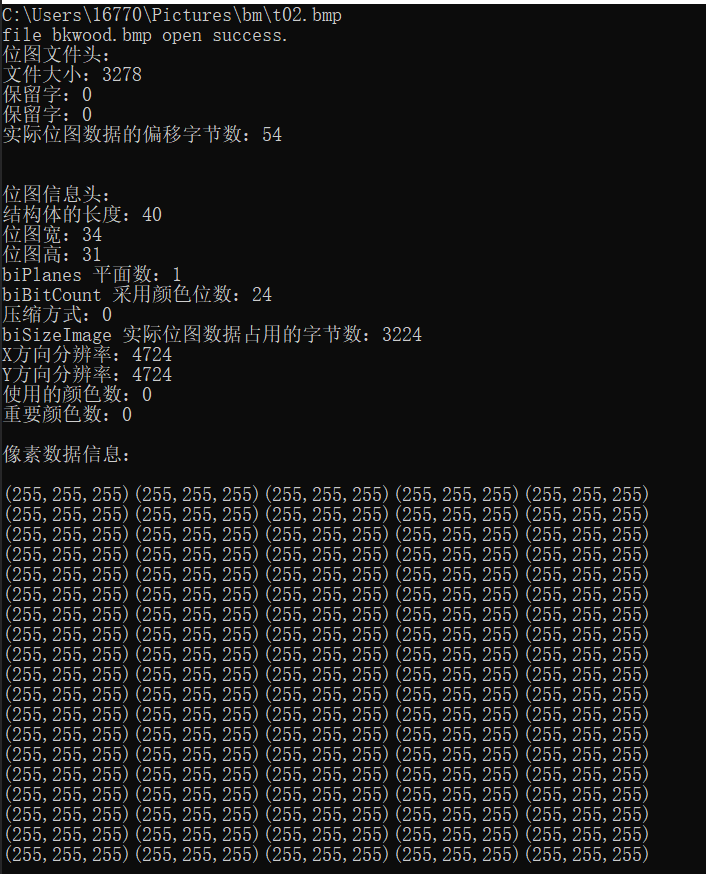


图4

原BMP文件如图5所示：



图5

通过biBitcount的数值进行讨论，分情况处理数据。

**（四）、像素数据的读取与打印**

像素数据的结构沿用调色板的结构。打印函数也是showRgbQuan（）；

1. **实验代码**
   1. **完整代码如下：**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <string.h>

#include <malloc.h>

#define WIDTHBYTES(bits) (((bits)+31)/32\*4)

typedef unsigned char BYTE;

typedef unsigned short WORD;

typedef unsigned long DWORD;

typedef long LONG;

typedef struct tagBITMAPFILEHEADER

{

//unsigned short int bfType; //位图文件的类型，必须为BM

DWORD bfSize; //文件大小，以字节为单位

WORD bfReserved1; //位图文件保留字，必须为0

WORD bfReserved2; //位图文件保留字，必须为0

DWORD bfbfOffBits; //位图文件头到数据的偏移量，以字节为单位

}BITMAPFILEHEADER;

typedef struct tagBITMAPINFOHEADER

{

DWORD biSize; //该结构大小，字节为单位

LONG biWidth; //图形宽度以象素为单位

LONG biHeight; //图形高度以象素为单位

WORD biPlanes; //目标设备的级别，必须为1

WORD biBitcount; //颜色深度，每个象素所需要的位数

DWORD biCompression; //位图的压缩类型

DWORD biSizeImage; //位图的大小，以字节为单位

LONG biXPelsPermeter; //位图水平分辨率，每米像素数

LONG biYPelsPermeter; //位图垂直分辨率，每米像素数

DWORD biClrUsed; //位图实际使用的颜色表中的颜色数

DWORD biClrImportant; //位图显示过程中重要的颜色数

}BITMAPINFOHEADER;

typedef struct tagRGBQUAD {

BYTE rgbBlue;

BYTE rgbGreen;

BYTE rgbRed;

BYTE rgbReserved;

}RGBQUAD;

void showBmpHead(BITMAPFILEHEADER\* pBmpHead)

{

printf("位图文件头：\n");

printf("文件大小：%d\n",pBmpHead->bfSize);

printf("保留字：%d\n",pBmpHead->bfReserved1);

printf("保留字：%d\n",pBmpHead->bfReserved2);

printf("实际位图数据的偏移字节数：%d\n",pBmpHead->bfbfOffBits);

}

void showBmpInforHead(tagBITMAPINFOHEADER\* pBmpInforHead)

{

printf("位图信息头：\n");

printf("结构体的长度：%d\n",pBmpInforHead->biSize);

printf("位图宽：%d\n",pBmpInforHead->biWidth);

printf("位图高：%d\n",pBmpInforHead->biHeight);

printf("biPlanes 平面数：%d\n",pBmpInforHead->biPlanes);

printf("biBitCount 采用颜色位数：%d\n",pBmpInforHead->biBitcount);

printf("压缩方式：%d\n",pBmpInforHead->biCompression);

printf("biSizeImage 实际位图数据占用的字节数：%d\n",pBmpInforHead->biSizeImage);

printf("X方向分辨率：%d\n",pBmpInforHead->biXPelsPermeter);

printf("Y方向分辨率：%d\n",pBmpInforHead->biYPelsPermeter);

printf("使用的颜色数：%d\n",pBmpInforHead->biClrUsed);

printf("重要颜色数：%d\n",pBmpInforHead->biClrImportant);

}

void showRgbQuan(tagRGBQUAD\* pRGB)

{

printf("(%-3d,%-3d,%-3d)",pRGB->rgbRed,pRGB->rgbGreen,pRGB->rgbBlue);

}

void main()

{

BITMAPFILEHEADER bitHead;

BITMAPINFOHEADER bitInfoHead;

FILE\* pfile;

char strFile[50];

printf("please input the .bmp file name:\n");

scanf("%s",strFile);

pfile = fopen(strFile,"rb");//打开文件

if(pfile!=NULL)

{

printf("file bkwood.bmp open success.\n");

//读取位图文件头信息

WORD fileType;

fread(&fileType,1,sizeof(WORD),pfile);

if(fileType!=0x4d42)

{

printf("file is not .bmp file!");

return;

}

fread(&bitHead,1,sizeof(tagBITMAPFILEHEADER),pfile);

showBmpHead(&bitHead);

printf("\n\n");

//读取位图信息头信息

fread(&bitInfoHead,1,sizeof(BITMAPINFOHEADER),pfile);

showBmpInforHead(&bitInfoHead);

printf("\n");

}

else

{

printf("file open fail!\n");

return ;

}

tagRGBQUAD \*pRgb;

if(bitInfoHead.biBitcount<24)

{

//读取调色盘信息

long nPlantNum = long(pow(2,double(bitInfoHead.biBitcount)));

pRgb=(tagRGBQUAD \*)malloc(nPlantNum\*sizeof(tagRGBQUAD));

memset(pRgb,0,nPlantNum\*sizeof(tagRGBQUAD));

int num=fread(pRgb,4,nPlantNum,pfile);

printf("Color Plate Number: %d\n",nPlantNum);

printf("颜色板信息：\n");

for(int i =0;i<nPlantNum;i++)

{

if(i%5 == 0)

{

printf("\n");

}

showRgbQuan(&pRgb[i]);

}

printf("\n");

}

int width = bitInfoHead.biWidth;

int height = bitInfoHead.biHeight;

//分配内存空间把源图存入内存

int l\_width = WIDTHBYTES(width\* bitInfoHead.biBitcount);

BYTE \*pColorData=(BYTE \*)malloc(height \* l\_width);

long nData = height\*l\_width;

//把位图数据信息读到数组里

fread(pColorData,1,nData,pfile);

//将位图数据转化为RGB数据

tagRGBQUAD\* dataOfBmp;

//用于保存各像素对应的RGB数据

dataOfBmp = (tagRGBQUAD \*)malloc(width \*height\*sizeof(tagRGBQUAD));

memset(dataOfBmp,0,width\*height\*sizeof(tagRGBQUAD));

if(bitInfoHead.biBitcount<24)//有调色板，即位图为非真彩色

{

int k;

int index = 0;

if(bitInfoHead.biBitcount == 1)

{

for(int i =0;i<height;i++)

for(int j=0;j<width;j++)

{

BYTE mixIndex= 0;

k = i\*l\_width + j/8;

mixIndex=pColorData[k];

switch(j%8)

{

case 0:

mixIndex = mixIndex<<7;

mixIndex = mixIndex>>7;

break;

case 1:

mixIndex = mixIndex<<6;

mixIndex = mixIndex>>7;

break;

case 2:

mixIndex = mixIndex<<5;

mixIndex = mixIndex>>7;

break;

case 3:

mixIndex = mixIndex<<4;

mixIndex = mixIndex>>7;

break;

case 4:

mixIndex = mixIndex<<3;

mixIndex = mixIndex>>7;

break;

case 5:

mixIndex = mixIndex<<2;

mixIndex = mixIndex>>7;

break;

case 6:

mixIndex = mixIndex<<1;

mixIndex = mixIndex>>7;

break;

case 7:

mixIndex = mixIndex>>7;

break;

}

//将像素数据保存到数组中对应的位置

dataOfBmp[index].rgbRed = pRgb[mixIndex].rgbRed;

dataOfBmp[index].rgbGreen = pRgb[mixIndex].rgbGreen;

dataOfBmp[index].rgbBlue = pRgb[mixIndex].rgbBlue;

dataOfBmp[index].rgbReserved = pRgb[mixIndex].rgbReserved;

index++;

}

}

if(bitInfoHead.biBitcount==2)

{

for(int i=0;i<height;i++)

for(int j=0;j<width;j++)

{

BYTE mixIndex= 0;

k= i\*l\_width + j/4;

mixIndex= pColorData[k];

switch(j%4)

{

case 0:

mixIndex = mixIndex<<6;

mixIndex = mixIndex>>6;

break;

case 1:

mixIndex = mixIndex<<4;

mixIndex = mixIndex>>6;

break;

case 2:

mixIndex = mixIndex<<2;

mixIndex = mixIndex>>6;

break;

case 3:

mixIndex = mixIndex>>6;

break;

}

//将像素数据保存到数组中对应的位置

dataOfBmp[index].rgbRed = pRgb[mixIndex].rgbRed;

dataOfBmp[index].rgbGreen = pRgb[mixIndex].rgbGreen;

dataOfBmp[index].rgbBlue = pRgb[mixIndex].rgbBlue;

dataOfBmp[index].rgbReserved = pRgb[mixIndex].rgbReserved;

index++;

}

}

if(bitInfoHead.biBitcount==4)

{

for (int i=0;i<height;i++)

for (int j=0;j<width;j++)

{

BYTE mixIndex = 0;

k= i\*l\_width +j/2;

mixIndex = pColorData[k];

if(j%2==0)

{

mixIndex = mixIndex<<4;

mixIndex= mixIndex>>4;

}

else

{

mixIndex = mixIndex>>4;

}

dataOfBmp[index].rgbRed = pRgb[mixIndex].rgbRed;

dataOfBmp[index].rgbGreen = pRgb[mixIndex].rgbGreen;

dataOfBmp[index].rgbBlue = pRgb[mixIndex].rgbBlue;

dataOfBmp[index].rgbReserved = pRgb[mixIndex].rgbReserved;

index++;

}

}

if(bitInfoHead.biBitcount==8)

{

for (int i=0;i<height;i++)

for (int j=0;j<width;j++)

{

BYTE mixIndex = 0;

k= i\*l\_width +j;

mixIndex = pColorData[k];

dataOfBmp[index].rgbRed = pRgb[mixIndex].rgbRed;

dataOfBmp[index].rgbGreen = pRgb[mixIndex].rgbGreen;

dataOfBmp[index].rgbBlue = pRgb[mixIndex].rgbBlue;

dataOfBmp[index].rgbReserved = pRgb[mixIndex].rgbReserved;

index++;

}

}

if(bitInfoHead.biBitcount==16)

{

for (int i=0;i<height;i++)

for (int j=0;j<width;j++)

{

WORD mixIndex = 0;

k= i\*l\_width +j\*2;

WORD shortTemp;

shortTemp = pColorData[k+1];

shortTemp = shortTemp<<8;

mixIndex = pColorData[k] + shortTemp;

dataOfBmp[index].rgbRed = pRgb[mixIndex].rgbRed;

dataOfBmp[index].rgbGreen = pRgb[mixIndex].rgbGreen;

dataOfBmp[index].rgbBlue = pRgb[mixIndex].rgbBlue;

dataOfBmp[index].rgbReserved = pRgb[mixIndex].rgbReserved;

index++;

}

}

}

else //位图为24位真彩

{

int k;

int index = 0;

for(int i=0;i<height;i++)

for(int j=0;j<width;j++)

{

k=i\*l\_width + j\*3;

dataOfBmp[index].rgbRed = pColorData[k+2];

dataOfBmp[index].rgbGreen = pColorData[k+1];

dataOfBmp[index].rgbBlue = pColorData[k];

index++;

}

}

printf("像素数据信息：\n");

for (int i=0;i<width\*height;i++)

{

if(i%5==0)

{

printf("\n");

}

showRgbQuan(&dataOfBmp[i]);

}

fclose(pfile);

if (bitInfoHead.biBitcount<24)

{

free(pRgb);

}

free(dataOfBmp);

free(pColorData);

printf("\n");

getchar();

system("PAUSE");

return ;

}

参考文献：

[1] taodavid.BMP图像数据详解[EB/OL].(2008-06-18). https://blog.csdn.net/taodavid/article/details/80724344