A person in a small boat is on a calm lake, with mountains in the background. The scene is peaceful and scenic.

## 第二章

# BMP图像文件结构

# 基本概念

- 像素
  - 分辨率
- 像素值
  - bits
- 图像种类
  - 灰度
    - 二值（二  
级）
    - 多值（多  
级）
  - 彩色
    - 真彩
    - 伪彩

# 基本概念

- 像素
  - 分辨率
- 像素值
  - bits
- 图像种类
  - 灰度
    - 二值（二级）
    - 多值（多级）
  - 彩色
    - 真彩
    - 伪彩



Pixel: Y

8bits = 256

1bits

2bits

4bits

# 基本概念

- 像素
  - 分辨率
- 像素值
  - bits
- 图像种类
  - 灰度
    - 二值 (二  
级)
    - 多值 (多  
级)
  - 彩色
    - 真彩
    - 伪彩



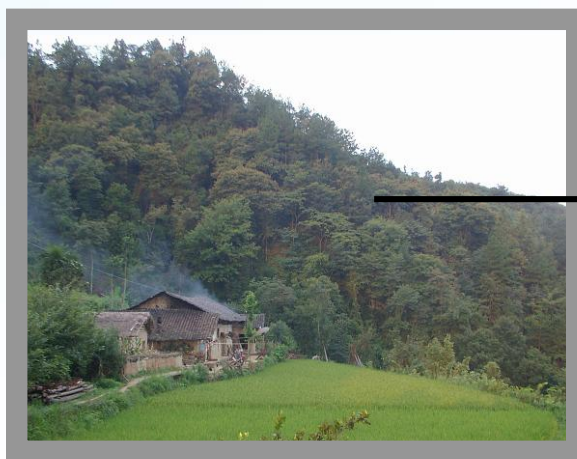
Pixel: Y

8bits = 256

1bits

2bits

4bits



Pixel: R, G, B

8, 8, 8 bits

24bits = 3 bytes

$256 \times 256 \times 256 = 16777216$  色

# 基本概念

- 像素
  - 分辨率
- 像素值
  - bits
- 图像种类
  - 灰度
    - 二值（二级）
    - 多值（多级）
  - 彩色
    - 真彩
    - 伪彩



Pixel: Y

8bits = 256

1bits

2bits

4bits



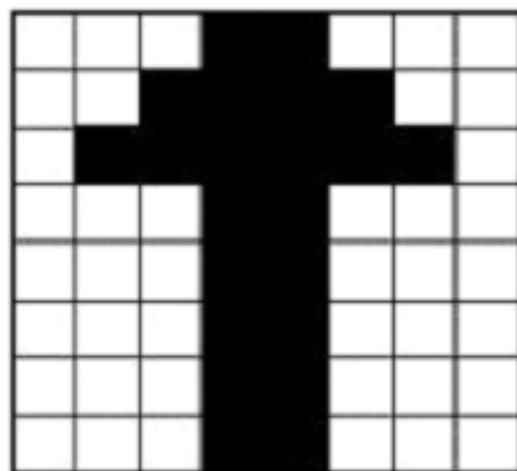
Pixel: R, G, B

8, 8, 8 bits

24bits = 3 bytes

$256 \times 256 \times 256 = 16777216$  色

彩色显示器能显示  
灰度图像吗？



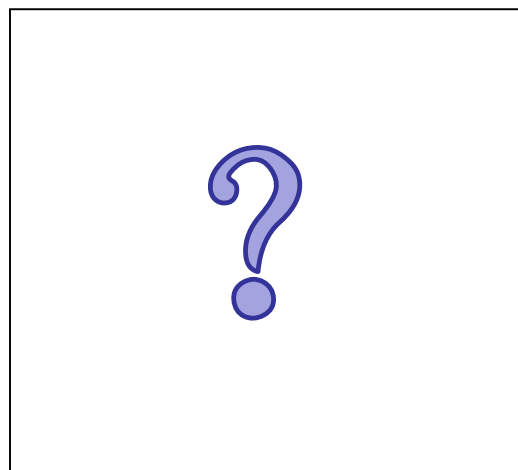
原图像

```

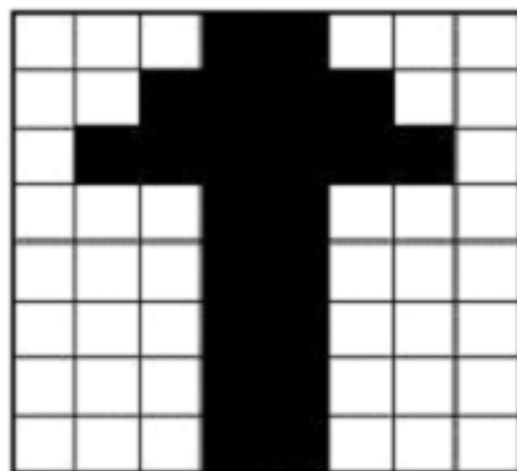
BYTE *image_in =
{
    {255, 255, 255, 0, 0, 255, 255, 255},
    {255, 255, 0, 0, 0, 0, 255, 255},
    {255, 0, 0, 0, 0, 0, 0, 255},
    {255, 255, 255, 0, 0, 255, 255, 255},
    {255, 255, 255, 0, 0, 255, 255, 255},
    {255, 255, 255, 0, 0, 255, 255, 255},
    {255, 255, 255, 0, 0, 255, 255, 255},
    {255, 255, 255, 0, 0, 255, 255, 255}
};

for (j = 0; j < 8; j++)
    for (i = 0; i < 8; i++)
        *(image_out + j*8+i) =
            *(image_in + (8-1-j)*8+i);

```



处理后图像

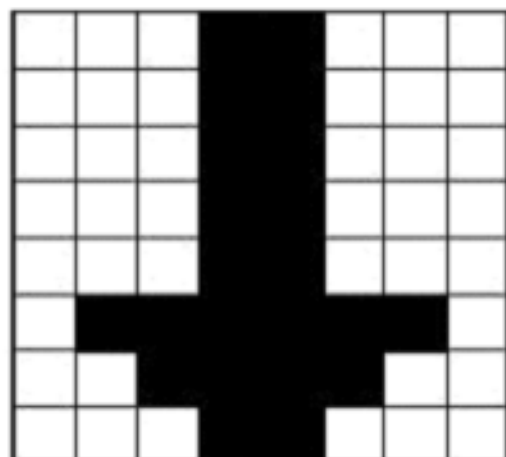


原图像

```

BYTE *image_in =
{
  {255, 255, 255, 0, 0, 255, 255, 255},
  {255, 255, 0, 0, 0, 0, 255, 255},
  {255, 0, 0, 0, 0, 0, 0, 255},
  {255, 255, 255, 0, 0, 255, 255, 255},
  {255, 255, 255, 0, 0, 255, 255, 255},
  {255, 255, 255, 0, 0, 255, 255, 255},
  {255, 255, 255, 0, 0, 255, 255, 255},
  {255, 255, 255, 0, 0, 255, 255, 255}
}

```

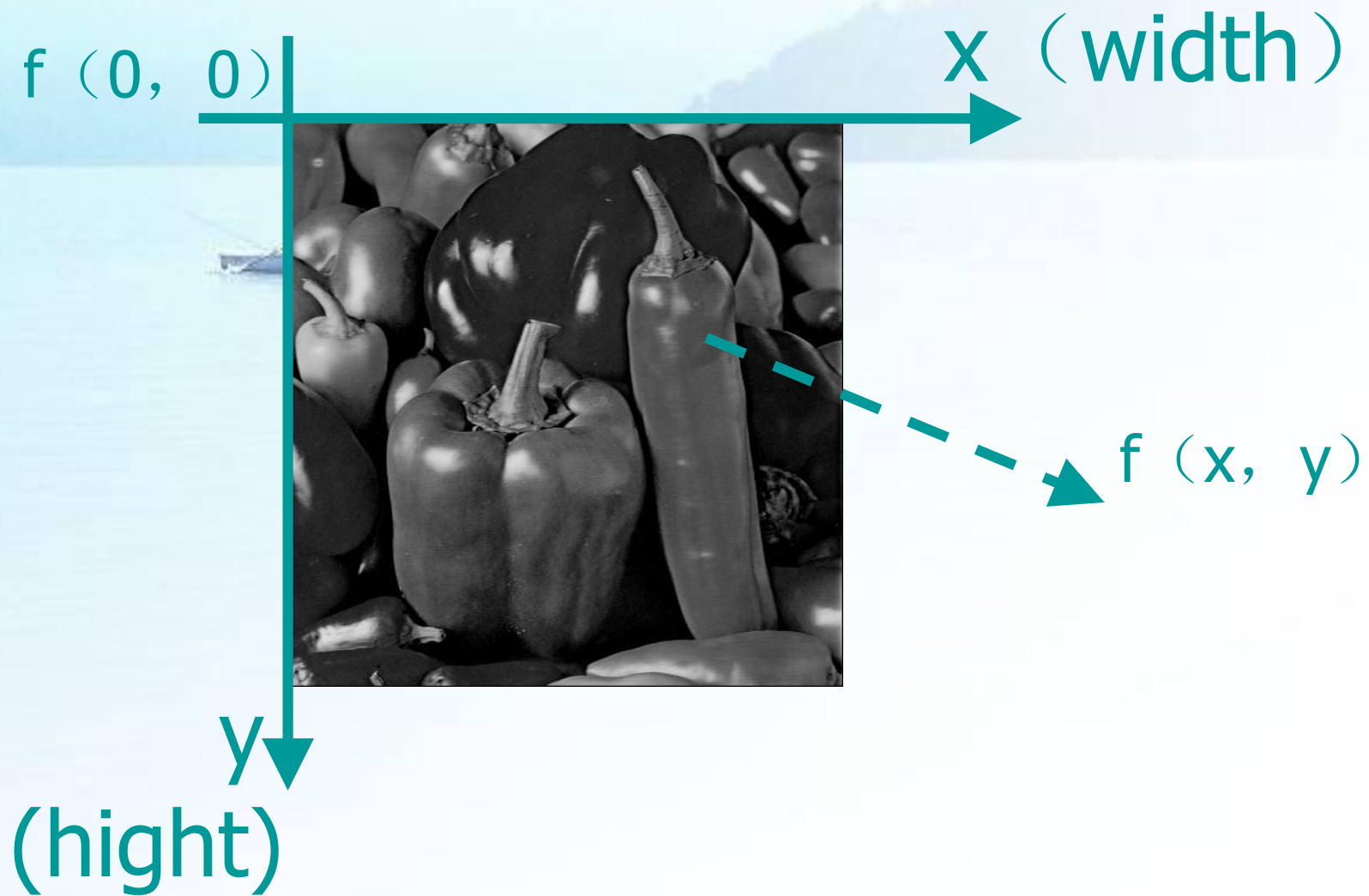


处理后图像

```

};
for (j = 0; j < 8; j++)
  for (i = 0; i < 8; i++)
    *(image_out + j*8+i) =
      *(image_in + (8-1-j)*8+i);

```







Disk

RAM

# 彩色图像BMP文件格式

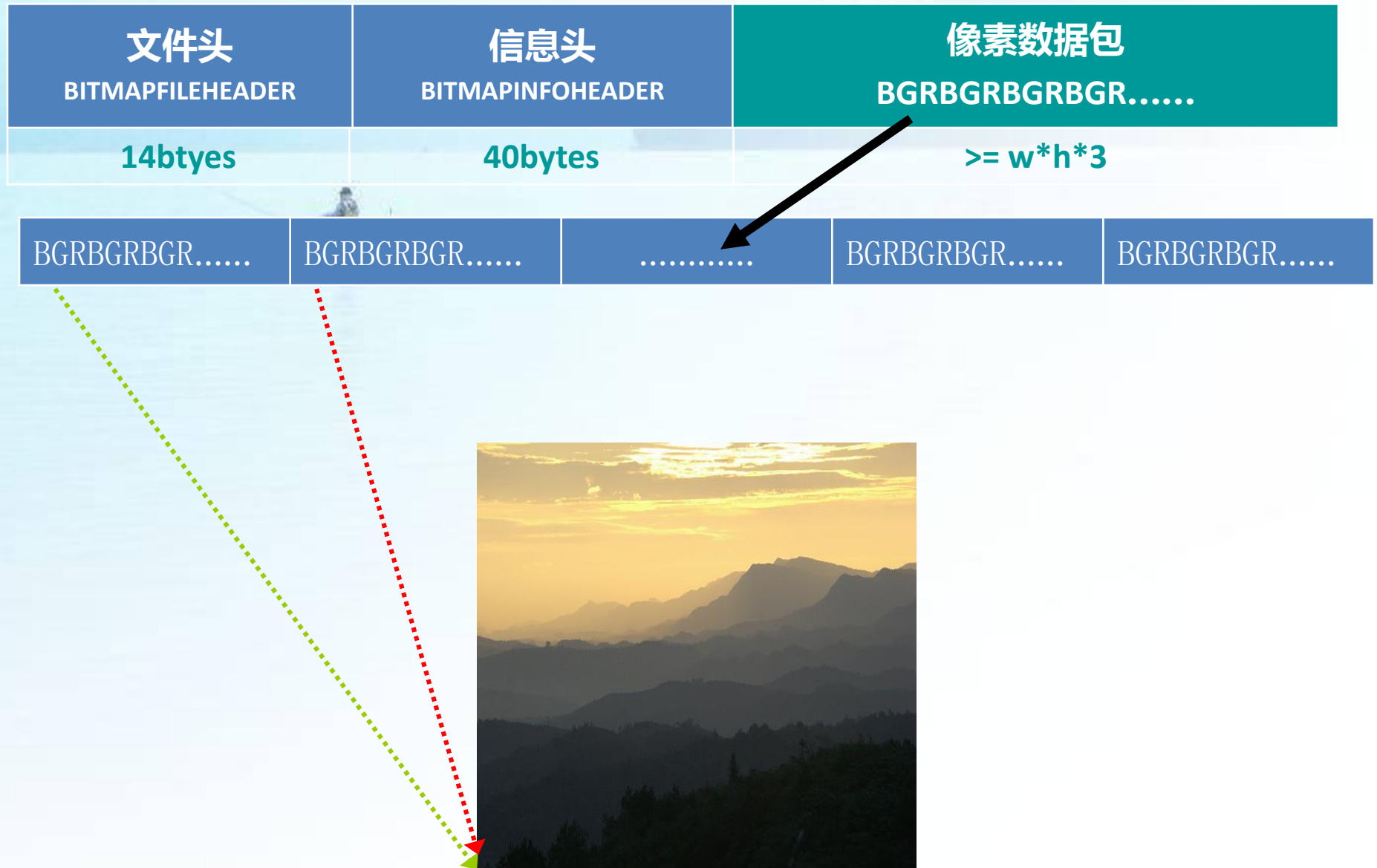


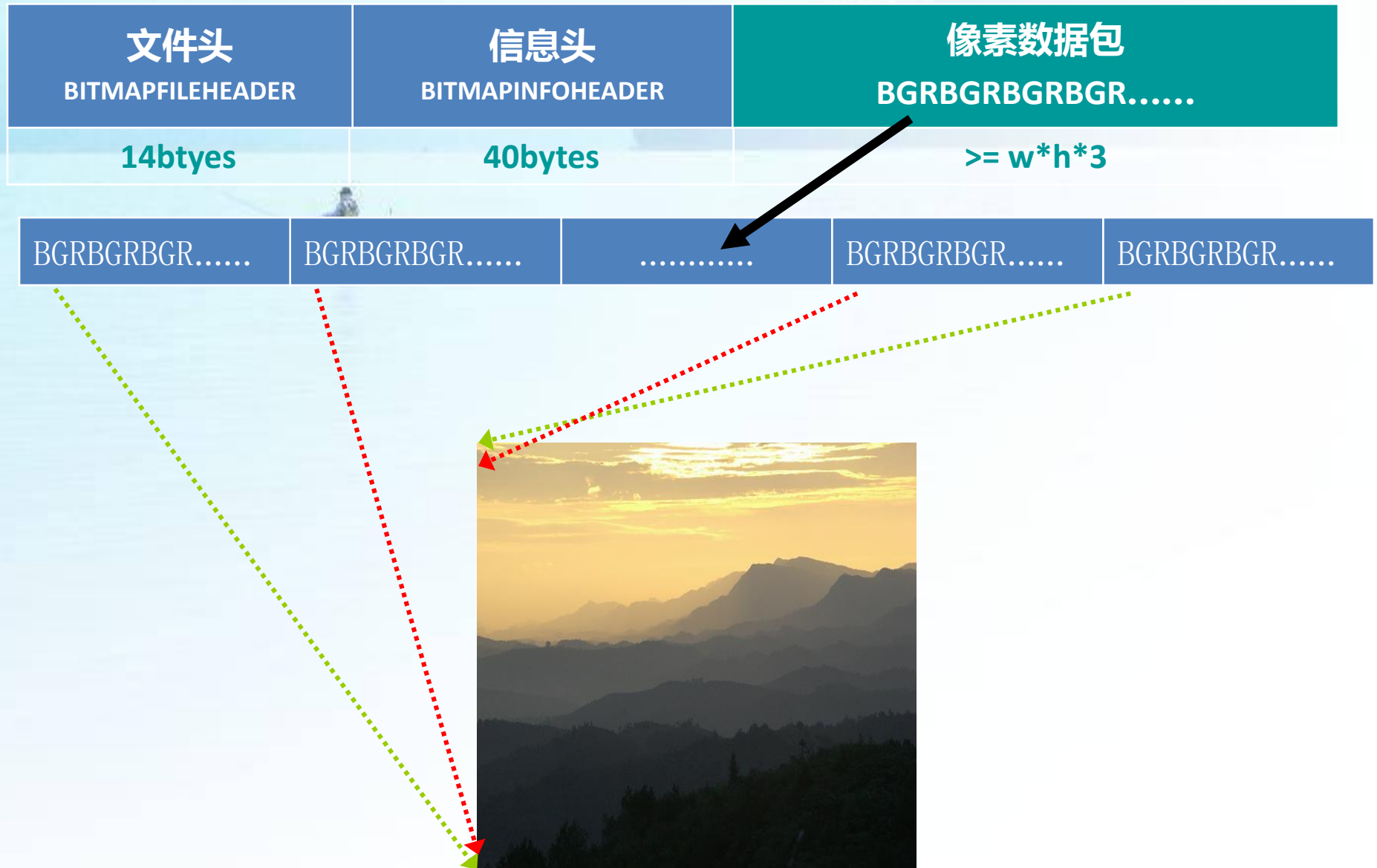
Disk

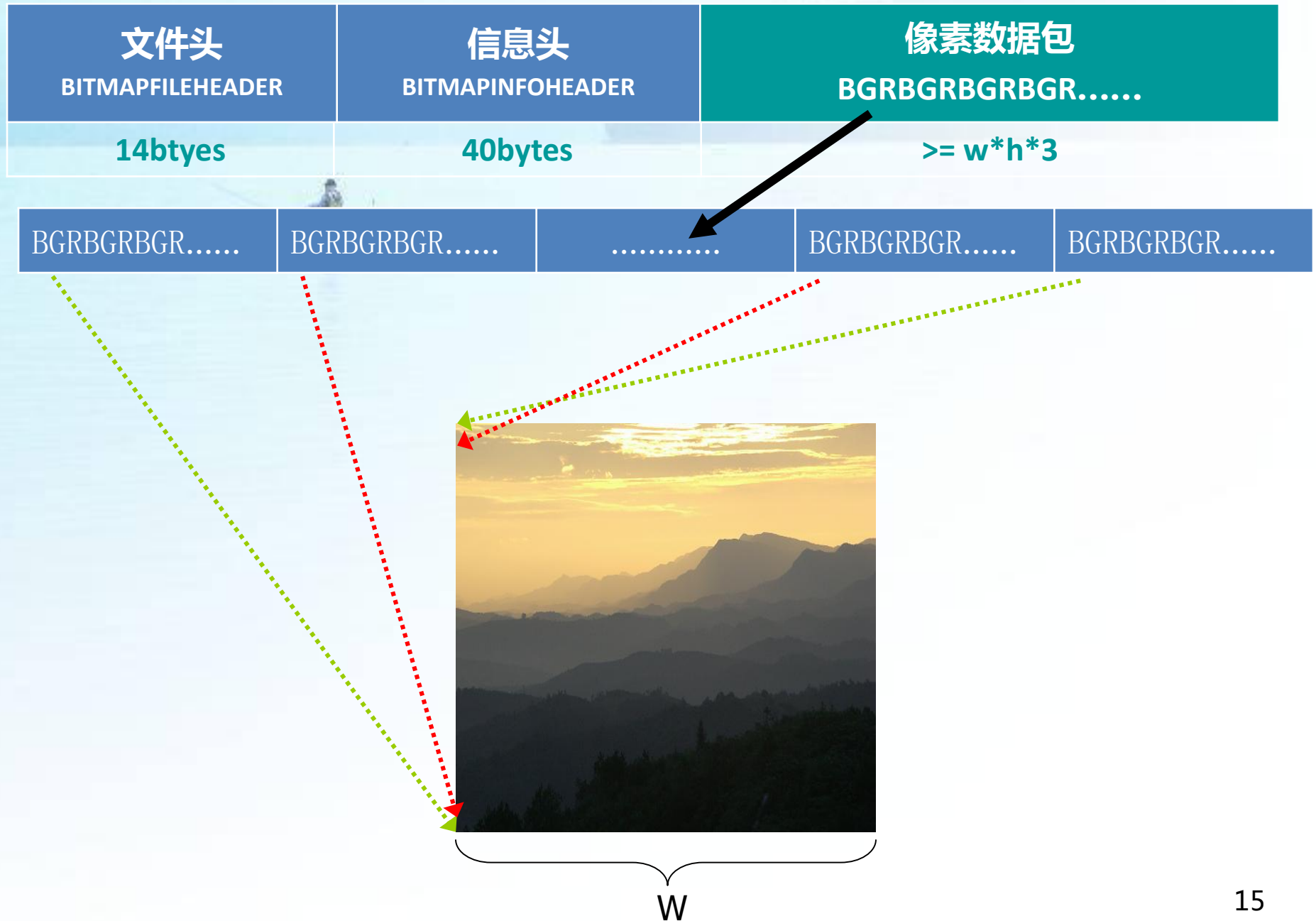
文件头 BITMAPFILEHEADER	信息头 BITMAPINFOHEADER	像素数据包 BGRBGRBGRBGR.....
14bytes	40bytes	$\geq w \times h \times 3$

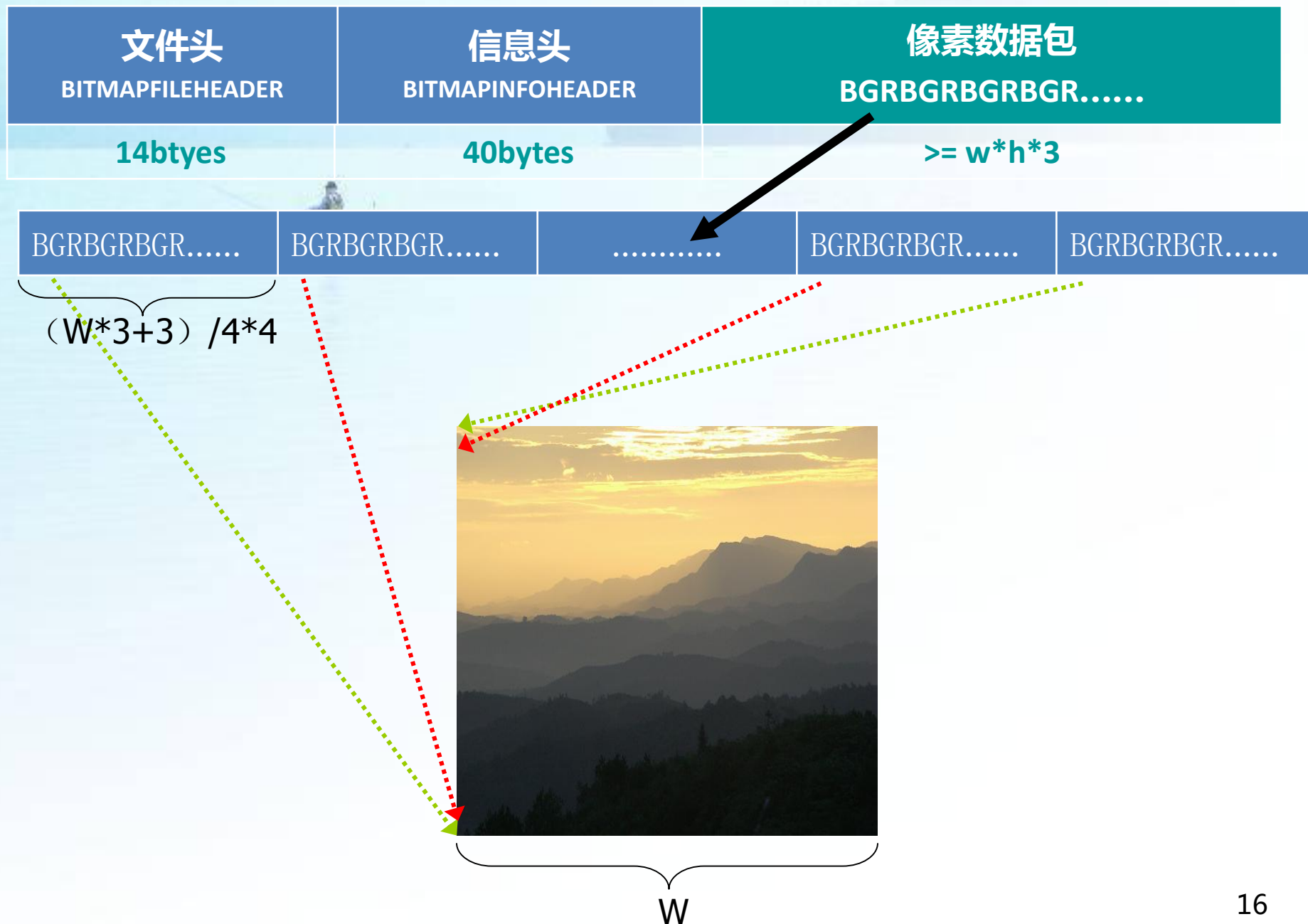




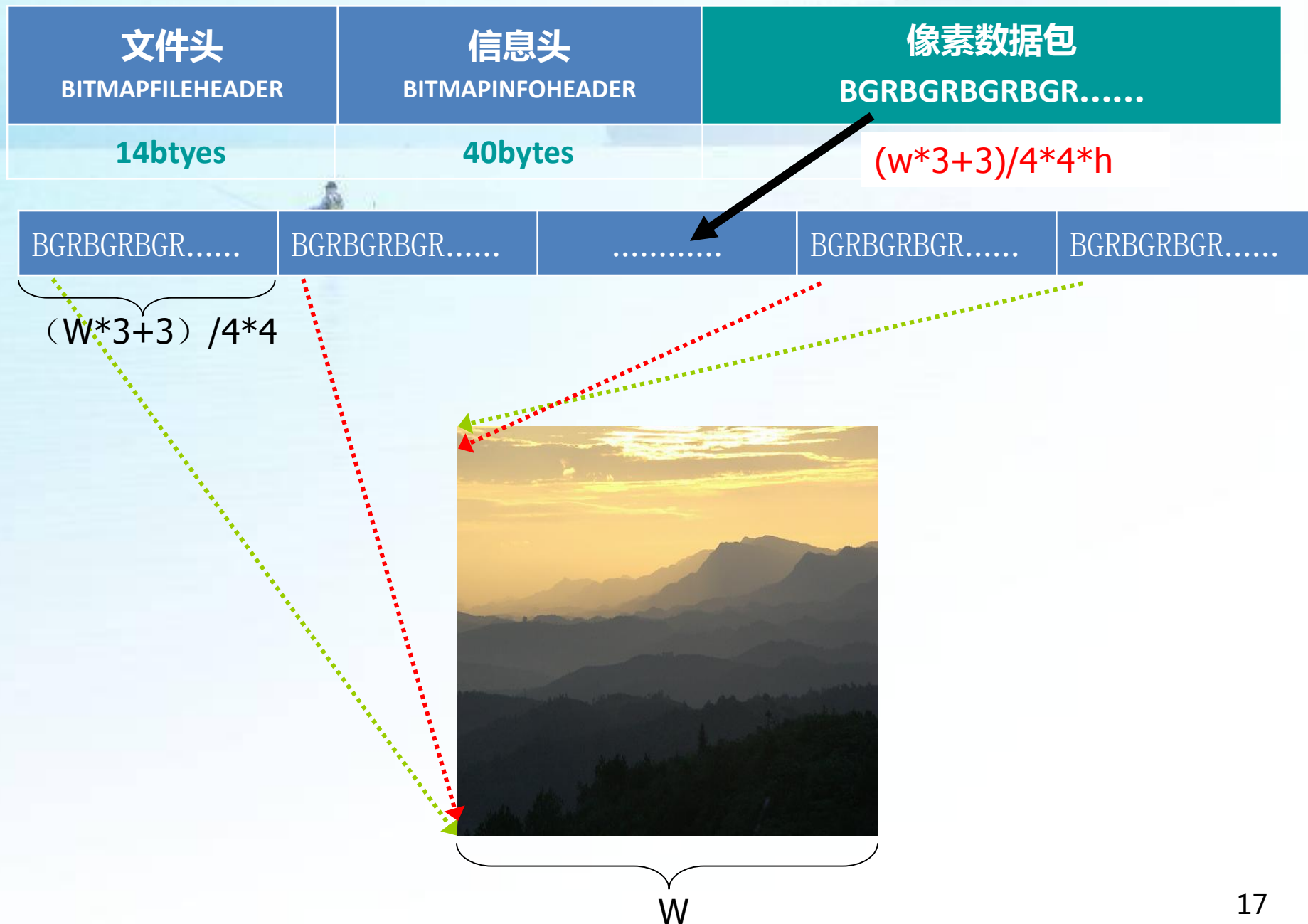


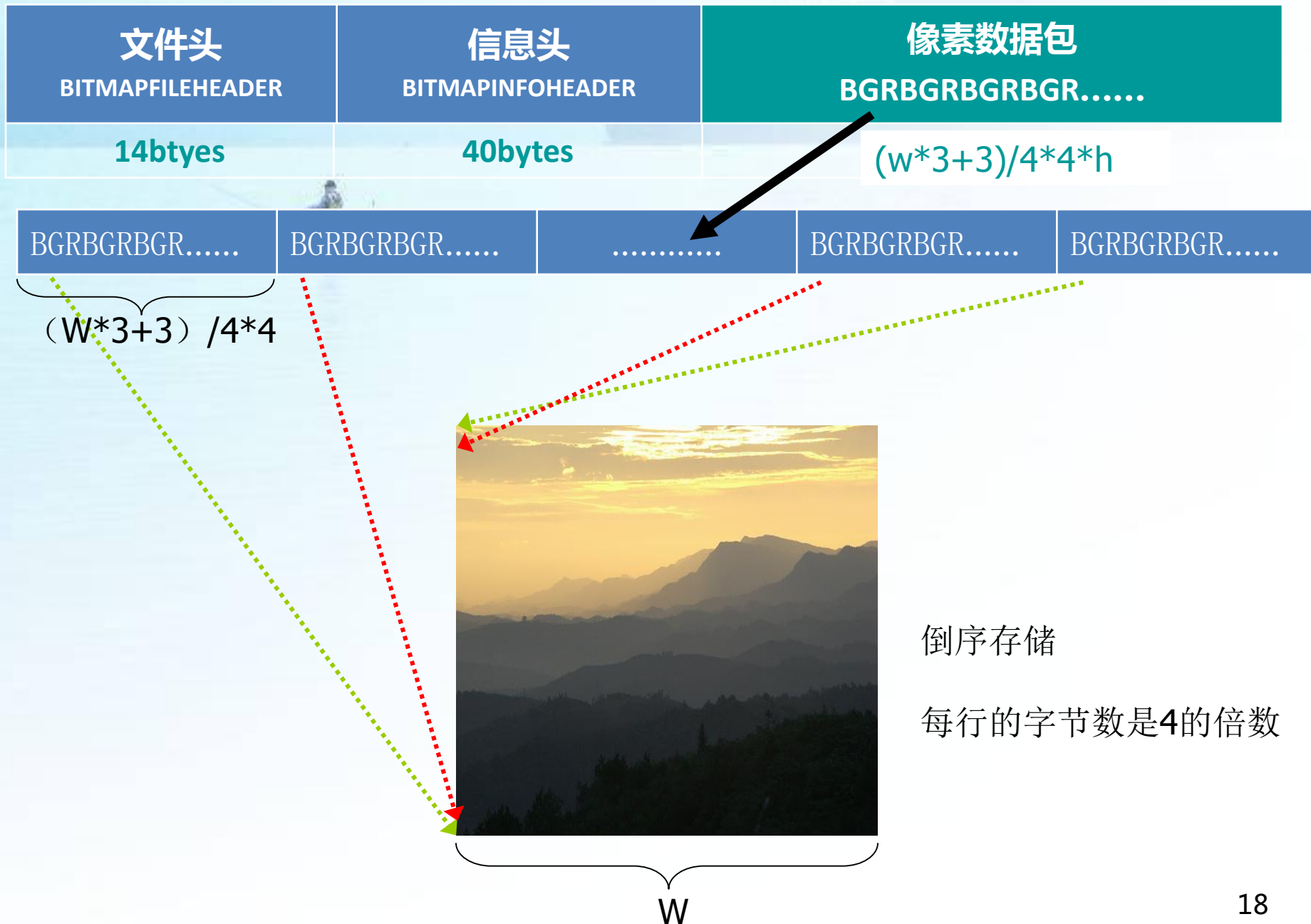












文件头 BITMAPFILEHEADER	信息头 BITMAPINFOHEADER	像素数据包 BGRBGRBGRBGR.....
14bytes	40bytes	$(w*3+3)/4*4*h$



- typedef struct tagBITMAPFILEHEADER {
- WORD   bfType;
- //指文件类型，必须是0x4D42，即字符串“BM”。
- DWORD   bfSize;
- //整个文件大小 = 文件头 + 信息头 + 像素数据包
- WORD   bfReserved1;//保留位
- WORD   bfReserved2;//保留位
- DWORD   bfOffBits;
- //指从开始到像素数据位置的偏移字节数=文件头+信息头
- } **BITMAPFILEHEADER;**



- `typedef struct tagBITMAPFILEHEADER {`
- `WORD bfType;`
- `//指文件类型，必须是0x4D42，即字符串“BM”。`
- `DWORD bfSize;`
- `//整个文件大小 = 文件头 + 信息头 + 像素数据包`
- `WORD bfReserved1; //保留位`
- `WORD bfReserved2; //保留位`
- `DWORD bfOffBits;`
- `//指从开始到像素数据位置的偏移字节数 = 文件头 + 信息头`
- `} BITMAPFILEHEADER;`

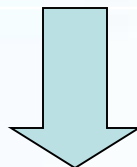
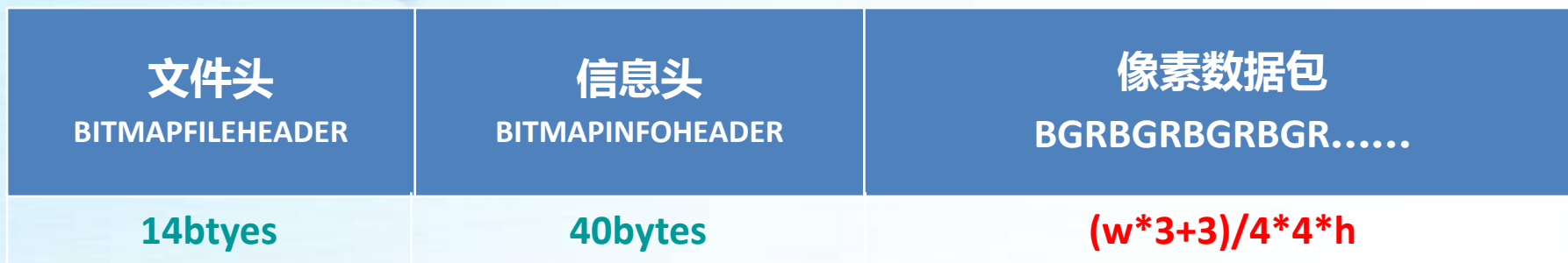
文件头 BITMAPFILEHEADER	信息头 BITMAPINFOHEADER	像素数据包 BGRBGRBGRBGR.....
14bytes	40bytes	$(w*3+3)/4*4*h$



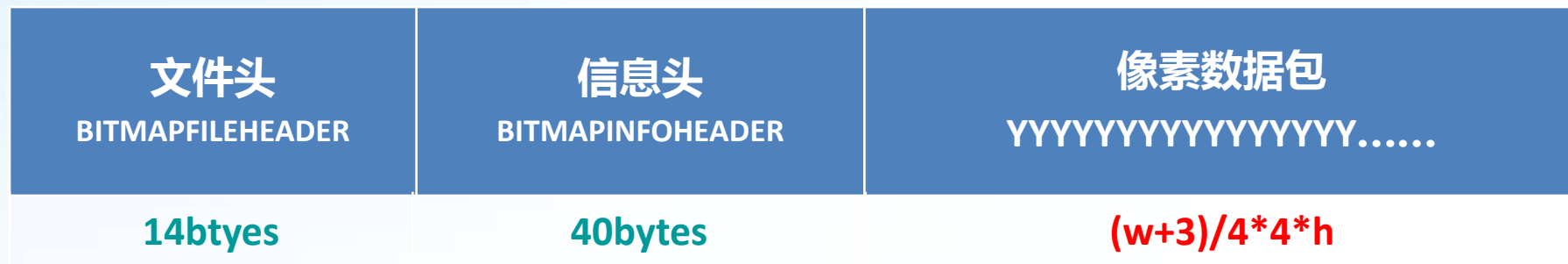
- typedef struct tagBITMAPINFOHEADER{
- DWORD     biSize; //指该结构体的大小，一般为40个字节
- LONG      biWidth; //指图像的宽度，单位：像素
- LONG      biHeight; //指图像的高度，单位：像素
- WORD      biPlanes; //必须是1
- WORD      biBitCount; //指图像数据位数。彩色图像24，灰度图像为8、4、2、1等
- DWORD     biCompression; //指图像是否压缩。一般为BI\_RGB表示非压缩格式。
- DWORD     biSizeImage; //指像素数据包的大小。
- LONG      biXPelsPerMeter; //指目标设备的水平分辨率，单位是每米的像素个数。
- LONG      biYPelsPerMeter; //指目标设备的垂直分辨率，单位是每米的像素个数。
- DWORD     biClrUsed; //指图像用到的颜色数，一般为0，用到的颜色数为 $2^{biBitCount}$ 。
- DWORD     biClrImportant; //指图像中重要的颜色数，一般为0，所有颜色都是重要。
- } **BITMAPINFOHEADER;**

# 灰度图像BMP文件格式

Disk

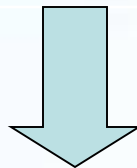
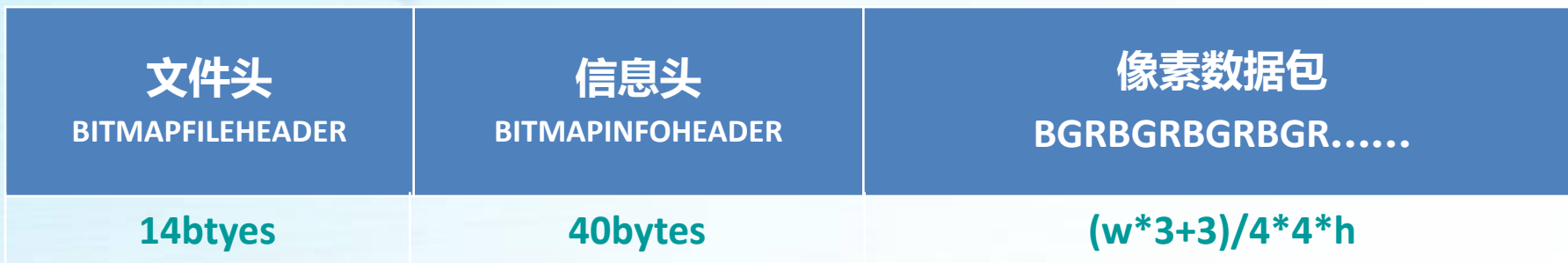


Disk

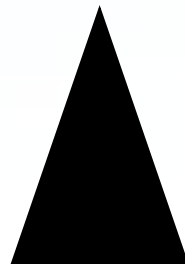


# 灰度图像BMP文件格式

Disk



Disk

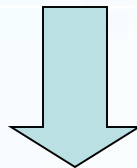


# 灰度图像BMP文件格式

Disk



文件头 BITMAPFILEHEADER	信息头 BITMAPINFOHEADER	像素数据包 BGRBGRBGRBGR.....
14bytes	40bytes	$(w*3+3)/4*4*h$



Disk

文件头 BITMAPFILEHEADER	信息头 BITMAPINFOHEADER	调色板 palette	像素数据包 YYYYYYYYYYYYYYYY.....
14bytes	40bytes	1024bytes	$(w+3)/4*4*h$



YYYYYYYY.....

YYYYYYYY.....

.....

YYYYYYYY.....

YYYYYYYY.....

## 灰度图像数据包





Y=n

## 灰度图像数据包





Y=n

灰度图像数据包





Y=n

灰度图像数据包



BYTE rgbBlue;  
BYTE rgbGreen;  
BYTE rgbRed;  
BYTE gbReserved;



Y=n

灰度图像数据包



```
BYTE rgbBlue;  
BYTE rgbGreen;  
BYTE rgbRed;  
BYTE gbReserved;
```



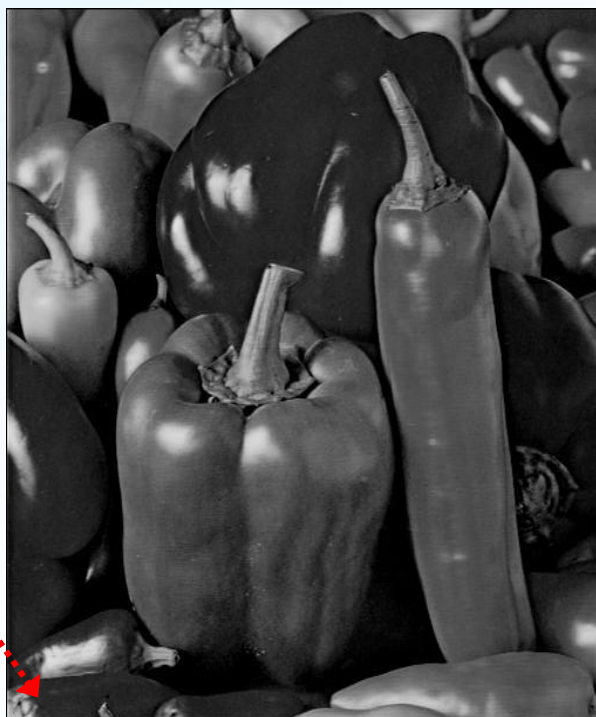
Y=n

灰度图像数据包



```

BYTE rgbBlue;
BYTE rgbGreen;
BYTE rgbRed;
BYTE gbReserved;
  
```





Y=n

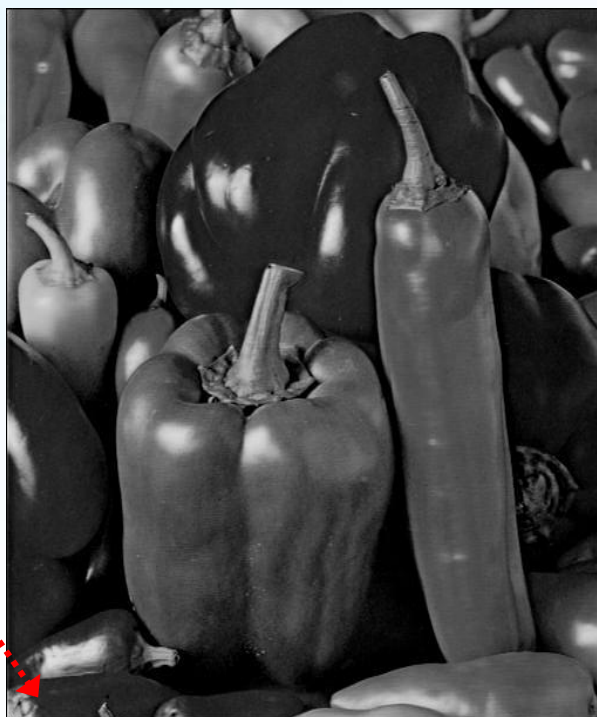
灰度图像数据包



能表示彩色图像吗？

```

BYTE rgbBlue;
BYTE rgbGreen;
BYTE rgbRed;
BYTE gbReserved;
  
```





- typedef struct tagBITMAPFILEHEADER {
- WORD bfType;
- //指文件类型，必须是0x4D42，即字符串“BM”。
- DWORD bfSize;
- //整个文件大小 = 文件头 + 信息头 + 像素数据包 + 调色板
- WORD bfReserved1; //保留位
- WORD bfReserved2; //保留位
- DWORD bfOffBits;
- //指从开始到像素数据位置的偏移字节数 = 文件头 + 信息头 + 调色板
- } BITMAPFILEHEADER;



文件头 BITMAPFILEHEADER	信息头 BITMAPINFOHEADER	调色板 Palette	像素数据包 彩色/灰度
14bytes	40bytes	1024	( 行的字节数+3 ) / 4 * 4 * h



- typedef struct tagBITMAPINFOHEADER{
- DWORD     biSize; //指该结构体的大小，一般为40个字节
- LONG      biWidth; //指图像的宽度，单位：像素
- LONG      biHeight; //指图像的高度，单位：像素
- WORD      biPlanes; //必须是1
- WORD      biBitCount; //指图像数据位数。彩色图像24，灰度图像为8、4、2、1等
- DWORD     biCompression; //指图像是否压缩。一般为BI\_RGB表示非压缩格式。
- DWORD     biSizeImage; //指像素数据包的大小。
- LONG      biXPelsPerMeter; //指目标设备的水平分辨率，单位是每米的像素个数。
- LONG      biYPelsPerMeter; //指目标设备的垂直分辨率，单位是每米的像素个数。
- DWORD     biClrUsed; //指图像用到的颜色数，一般为0，用到的颜色数为 $2^{biBitCount}$ 。
- DWORD     biClrImportant; //指图像中重要的颜色数，一般为0，所有颜色都是重要。
- } BITMAPINFOHEADER;

## • 灰度图像与彩色图像的转换

### – 彩色 变灰度

- $Y = 0.299R + 0.587G + 0.114B$

### – 灰度变彩色

- 人为变化
- 改调色板的值
- 同一幅图像中彩色数超不过  $2^{\text{biBitCount}}$



Disk

RAM



Disk



RAM

## Disk

BITMAPFILEHEADER	BITMAPINFOHEADER	RGBQUAD[256]	$n: (w+3)*4/4$	..... ....	$1: (w+3)*4/4$
------------------	------------------	--------------	----------------	------------	----------------

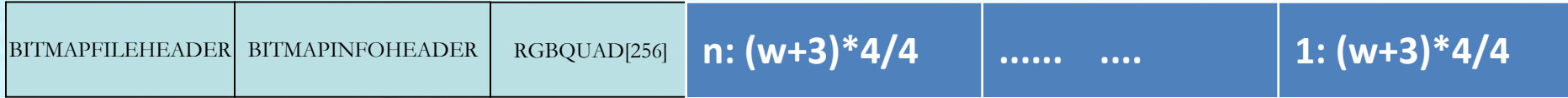
## Memory



```
int imagew, imageh;
int iRGBnum; //1: 灰度
              //3: 彩色
```

```
RGBQUAD palette[256];
```

Disk



```
BYTE *Imagedata [imagew*imageh]
```

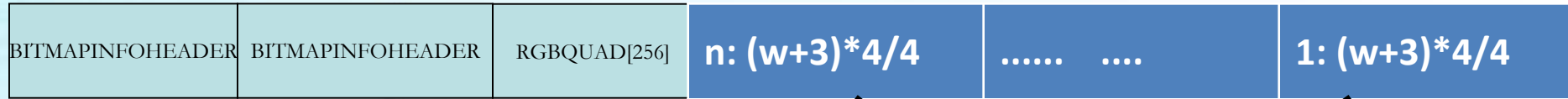
Memory



```
int imagew, imageh;
int iRGBnum; //1: 灰度
              //3: 彩色
```

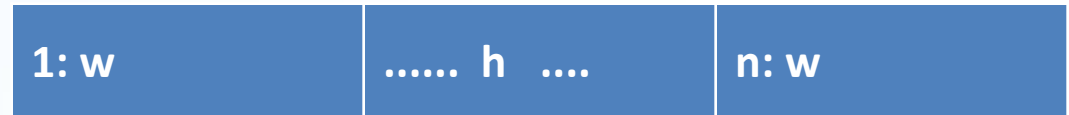
```
RGBQUAD palette[256];
```

Disk



```
BYTE *Imagedata [imagew*imageh]
```

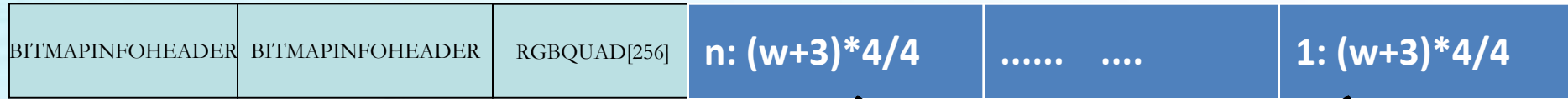
Memory



```
int imagew, imageh;
int iRGBnum; //1: 灰度
              //3: 彩色
```

```
RGBQUAD palette[256];
```

Disk



```
BYTE *Imagedata [imagew*imageh]
```

Memory



```
BYTE *pDataAt(int h,int Y0R0G1B2 = 0)
```





## Disk

BITMAPFILEHEADER	BITMAPINFOHEADER	n: $(w*3+3)*4/4$	..... ..	1: BGRBGR
------------------	------------------	------------------	----------	-----------

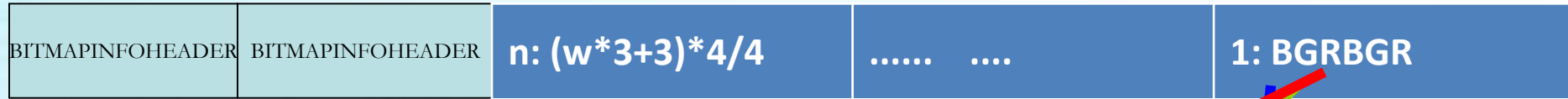
## Memory

R: $w*h$	G: $w*h$	B: $w*h$
----------	----------	----------



```
int imagew, imageh;  
int iRGBnum; //1: 灰度  
             //3: 彩色
```

Disk



```
BYTE *Imagedata [imagew*imageh*3]
```

Memory



0 1 2  
`BYTE *pDataAt(int h,int Y0R0G1B2)`



# 如何存取图像文件

- BITMAPFILEHEADER
  - bfType ?= 0x4D42
    - bfSize ?= all file size
- BITMAPINFOHEADER
  - biBitcount?=24
    - true color image
      - without palette
      - pixel data : BGRBGR...
    - other image
      - palette size =  $2^{\text{biBitcount}}$
      - pixel data: YYYYYY.....

- Get Pixel Data
  - BITMAPFILEHEADER
    - bfOffBits
  - first row in image
    - last row in file
  - Bytes of row in image
    - $w * \text{biBitcount} / 8$
  - Bytes of row in file
    - $(w * \text{biBitcount} + 31) / 32 * 4$

- #include "stdio.h"
- #include "math.h"
- #include "windows.h"
- #ifndef HXLBMPPFILEH
- #define HXLBMPPFILEH
- class HXLBMPPFILE
- {
- BYTE \*Imagedata;
- public:
- int imagew,imageh;
- int iYRGBnum;//1 : 灰度 , 3 : 彩色
- RGBQUAD palette[256];
- BYTE \*pDataAt(int h,int Y0R0G1B2 = 0);
- BOOL AllocateMem();
- BOOL LoadBMPPFILE(char \*fname);
- BOOL SaveBMPPFILE(char \*fname);
- HXLBMPPFILE();
- ~HXLBMPPFILE();
- };
- #endif

- HXLBMPPFILE::HXLBMPPFILE()
- {
- Imagedata=NULL;
- for (int i = 0; i < 256;i++)
- {
- palette [i].rgbBlue =
- palette [i].rgbGreen =
- palette [i].rgbRed = i;
- palette [i].rgbReserved = 0;
- }
- iYRGBnum = 0;
- imagew = imageh = 0;
- }
- HXLBMPPFILE::~~HXLBMPPFILE()
- {
- if (Imagedata) delete[] Imagedata;
- }

```

• #include "stdio.h"
• #include "math.h"
• #include "windows.h"

• #ifndef HXLBMPPFILEH
• #define HXLBMPPFILEH

• class HXLBMPPFILE
• {
•     BYTE *Imagedata;

•     public:
•     int imagew,imageh;
•     int iYRGBnum;//1 : 灰度 , 3 : 彩色
•     RGBQUAD palette[256];

•     BYTE *pDataAt(int h,int Y0R0G1B2 = 0);
•     BOOL AllocateMem();

•     BOOL LoadBMPPFILE(char *fname);
•     BOOL SaveBMPPFILE(char *fname);

•     HXLBMPPFILE();
•     ~HXLBMPPFILE();
• };
• #endif

```

```

■ BYTE *HXLBMPPFILE::pDataAt(int h, int Y0R0G1B2)
■ {
■     if (iYRGBnum <= Y0R0G1B2) return NULL;

■     int w=imagew *h + Y0R0G1B2 * imagew * imageh ;

■     return Imagedata+w;
■ }

■ BOOL HXLBMPPFILE::AllocateMem()
■ {
■     int w = imagew * imageh * iYRGBnum;

■     if (Imagedata)
■     {
■         delete[] Imagedata;Imagedata=NULL;
■     }

■     Imagedata=new BYTE [w];

■     if (Imagedata) memset(Imagedata,0,w);
■     return (Imagedata!=NULL);
■ }

```

```

• #include "stdio.h"
• #include "math.h"
• #include "windows.h"

• #ifndef HXLBMPPFILEH
• #define HXLBMPPFILEH

• class HXLBMPPFILE
• {
•     BYTE *Imagedata;

• public:
•     int imagew,imageh;
•     int iYRGBnum;//1 : 灰度 , 3 : 彩色
•     RGBQUAD palette[256];

•     BYTE *pDataAt(int h,int Y0R0G1B2 = 0);
•     BOOL AllocateMem();

•     BOOL LoadBMPFILE(char *fname);
•     BOOL SaveBMPFILE(char *fname);

•     HXLBMPPFILE();
•     ~HXLBMPPFILE();

• };
• #endif

```

```

■ BOOL HXLBMPPFILE::LoadBMPFILE (char *cFilename)
■ {
■     FILE *f;
■     if (strlen(cFilename)<1) return FALSE;

■     f=fopen(cFilename,"r+b");
■     if (f==NULL) return  FALSE;

■     BITMAPFILEHEADER fh;
■     BITMAPINFOHEADER ih;

■     fread(&fh,sizeof(BITMAPFILEHEADER),1,f);
■     if (fh.bfType!=0x4d42) {fclose(f);return FALSE;}//"BM"

■     fread (&ih,sizeof(BITMAPINFOHEADER),1,f);
■     if ( (ih.biBitCount != 8)&&(ih.biBitCount != 24) )
■     {
■         fclose (f);
■         return FALSE;
■     }

■     iYRGBnum = ih.biBitCount/8;
■     imagew = ih.biWidth ;
■     imageh = ih.biHeight ;

■     if(!AllocateMem()) {fclose (f);return FALSE;}

■     if ( iYRGBnum==1) fread (palette,sizeof(RGBQUAD),256,f);
■     fseek(f,fh.bfOffBits,SEEK_SET);

```

```

• #include "stdio.h"
• #include "math.h"
• #include "windows.h"

• #ifndef HXLBMPPFILEH
• #define HXLBMPPFILEH

• class HXLBMPPFILE
• {
•     BYTE *Imagedata;

• public:
•     int imagew,imageh;
•     int iYRGBnum;//1 : 灰度 , 3 : 彩色
•     RGBQUAD palette[256];

•     BYTE *pDataAt(int h,int Y0R0G1B2 = 0);
•     BOOL AllocateMem();

•     BOOL LoadBMPFILE(char *fname);
•     BOOL SaveBMPFILE(char *fname);

•     HXLBMPPFILE();
•     ~HXLBMPPFILE();

• };
• #endif

```

```

int w4b = (imagew * iRGBnum + 3 )/4 *4, i, j;
BYTE *ptr;

ptr = new BYTE [ w4b];
if (ptr==null) {fclose(f);return FALSE;}

if (iYRGBnum == 1)
{
    for (i=imageh -1; i>=0; i--)
    {
        fread(ptr,w4b,1,f);
        memmove(pDataAt (i),ptr, imagew);
    }
}
if ( iYRGBnum == 3)
{
    for ( i = imageh - 1; i >= 0; i--)
    {
        fread(ptr,w4b,1,f);
        for ( j = 0; j < imagew; j++)
        {
            pDataAt (i,0)+j = ptr+j*3 + 2;
            pDataAt (i,1)+j = ptr+j*3 + 1;
            pDataAt (i,2)+j = ptr+j*3 + 0;
        }
    }
}
delete[] ptr;
fclose(f);
return TRUE;
}

```

```

• #include "stdio.h"
• #include "math.h"
• #include "windows.h"

• #ifndef HXLBMPPFILEH
• #define HXLBMPPFILEH

• class HXLBMPPFILE
• {
•     BYTE *Imagedata;

• public:
•     int imagew,imageh;
•     int iYRGBnum;//1 : 灰度 , 3 : 彩色
•     RGBQUAD palette[256];

•     BYTE *pDataAt(int h,int Y0R0G1B2 = 0);
•     BOOL AllocateMem();

•     BOOL LoadBMPFILE(char *fname);
•     BOOL SaveBMPFILE(char *fname);

•     HXLBMPPFILE();
•     ~HXLBMPPFILE();

• };
• #endif

```



```

• int w4b = (imagew * iRGBnum + 3 )/4 *4, i, j;
• BYTE *ptr;

• ptr = new BYTE [ w4b];
• if (ptr==null) {fclose(f);return FALSE;}

• if (iYRGBnum == 1)
• {
•     for (i=imageh -1; i>=0; i--)
•     {
•         fread(ptr,w4b,1,f);
•         memmove(pDataAt (i),ptr, imagew);
•     }
• }
• if ( iYRGBnum == 3)
• {
•     for ( i = imageh - 1; i >= 0; i--)
•     {
•         fread(ptr,w4b,1,f);
•         for ( j = 0; j < imagew; j++)
•         {
•             pDataAt (i,0)+j = ptr+j*3 + 2;
•             pDataAt (i,1)+j = ptr+j*3 + 1;
•             pDataAt (i,2)+j = ptr+j*3 + 0;
•         }
•     }
• }
• delete[] ptr;
• fclose(f);
• return TRUE;
• }

```

?  
 pDataAt (i,0)+j = ptr+j\*3 + 2;  
 pDataAt (i,1)+j = ptr+j\*3 + 1;  
 pDataAt (i,2)+j = ptr+j\*3 + 0;



```

• #include "stdio.h"
• #include "math.h"
• #include "windows.h"

• #ifndef HXLBMPPFILEH
• #define HXLBMPPFILEH

• class HXLBMPPFILE
• {
•     BYTE *Imagedata;

• public:
•     int imagew,imageh;
•     int iYRGBnum;//1 : 灰度 , 3 : 彩色
•     RGBQUAD palette[256];

•     BYTE *pDataAt(int h,int Y0R0G1B2 = 0);
•     BOOL AllocateMem();

•     BOOL LoadBMPFILE(char *fname);
•     BOOL SaveBMPFILE(char *fname);

•     HXLBMPPFILE();
•     ~HXLBMPPFILE();

• };
• #endif

```



```

BOOL HXLBMPPFILE::SaveBMPFILE (char *cFilename)
{
    if (!Imagedata) return FALSE;
    FILE *f;
    if (strlen(cFilename)<1) return  FALSE;
    f=fopen(cFilename,"w+b"); if (f==NULL) return  FALSE;

    BITMAPFILEHEADER fh; BITMAPINFOHEADER ih;
    memset(&ih,0,sizeof(BITMAPINFOHEADER));

    fh.bfType = 0x4d42;
    fh.bfReserved1 = fh.bfReserved2 = 0;
    fh.bfOffBits = sizeof(BITMAPFILEHEADER)+
        sizeof(BITMAPINFOHEADER) +
        ( (iYRGBnum == 1)?256*sizeof(RGBQUAD):0);

    ih.biSize = 40; ih.biPlanes = 1;
    ih.biWidth = imagew; ih.biHeight = imageh;
    ih.biBitCount = 8 * iYRGBnum;

    int w4b = (imagew*iYRGBnum +3)/4*4;
    ih.biSizeImage = ih.biHeight *w4b;
    fh.bfSize = fh.bfOffBits + ih.biSizeImage;

    fwrite(&fh,sizeof(BITMAPFILEHEADER),1,f);
    fwrite(&ih,sizeof(BITMAPINFOHEADER),1,f);
    if ( iYRGBnum == 1) fwrite(palette,sizeof(RGBQUAD),256,f);
}

```

```

• #include "stdio.h"
• #include "math.h"
• #include "windows.h"

• #ifndef HXLBMPPFILEH
• #define HXLBMPPFILEH

• class HXLBMPPFILE
• {
•     BYTE *Imagedata;

• public:
•     int imagew,imageh;
•     int iYRGBnum;//1 : 灰度 , 3 : 彩色
•     RGBQUAD palette[256];

•     BYTE *pDataAt(int h,int Y0R0G1B2 = 0);
•     BOOL AllocateMem();

•     BOOL LoadBMPFILE(char *fname);
•     BOOL SaveBMPFILE(char *fname);

•     HXLBMPPFILE();
•     ~HXLBMPPFILE();

• };
• #endif

```

```

BYTE* ptr; int i,j;
ptr = new BYTE [w4b];
if (ptr==null){fclose(f); return FALSE;}
memset(ptr,0,w4b);
if (iYRGBnum == 1)
{
    for ( i=ih.biHeight -1;i>=0;i--)
    {
        memmove(ptr, pDataAt(i),ih.biWidth);
        fwrite(ptr,w4b,1,f);
    }
}
if (iYRGBnum == 3)
{
    for ( i=ih.biHeight -1;i>=0;i--)
    {
        for ( j = 0; j < ih.biWidth ; j++)
        {
            ptr+j*3 +0 = pDataAt(i,2)+j;
            ptr+j*3 +1 = pDataAt(i,1)+j;
            ptr+j*3 +2 = pDataAt(i,0)+j;
        }
        fwrite(ptr,w4b,1,f);
    }
}
delete[] ptr;
fclose(f);
return TRUE;
}

```

```

• #include "stdio.h"
• #include "math.h"
• #include "windows.h"

• #ifndef HXLBMPPFILEH
• #define HXLBMPPFILEH

• class HXLBMPPFILE
• {
•     BYTE *Imagedata;

• public:
•     int imagew,imageh;
•     int iYRGBnum;//1 : 灰度 , 3 : 彩色
•     RGBQUAD palette[256];

•     BYTE *pDataAt(int h,int Y0R0G1B2 = 0);
•     BOOL AllocateMem();

•     BOOL LoadBMPFILE(char *fname);
•     BOOL SaveBMPFILE(char *fname);

•     HXLBMPPFILE();
•     ~HXLBMPPFILE();

• };
• #endif

```



```

BYTE* ptr; int i,j;
ptr = new BYTE [w4b];
if (ptr==null){fclose(f); return FALSE;}
memset(ptr,0,w4b);
if (iYRGBnum == 1)
{
    for ( i=ih.biHeight -1;i>=0;i--)
    {
        memmove(ptr, pDataAt(i),ih.biWidth);
        fwrite(ptr,w4b,1,f);
    }
}
if (iYRGBnum == 3)
{
    for ( i=ih.biHeight -1;i>=0;i--)
    {
        for ( j = 0; j < ih.biWidth ; j++)
        {
            ptr+j*3 +0 = pDataAt(i,2)+j;
            ptr+j*3 +1 = pDataAt(i,1)+j;
            ptr+j*3 +2 = pDataAt(i,0)+j;
        }
        fwrite(ptr,w4b,1,f);
    }
}
delete[] ptr;
fclose(f);
return TRUE;
}

```

ptr+j\*3 +0 = pDataAt(i,2)+j;  
ptr+j\*3 +1 = pDataAt(i,1)+j;  
ptr+j\*3 +2 = pDataAt(i,0)+j;

```

• #include "stdio.h"
• #include "math.h"
• #include "windows.h"

• #ifndef HXLBMPPFILEH
• #define HXLBMPPFILEH

• class HXLBMPPFILE
• {
•     BYTE *Imagedata;

•     public:
•     int imagew,imageh;
•     int iYRGBnum;//1 : 灰度 , 3 : 彩色
•     RGBQUAD palette[256];

•     BYTE *pDataAt(int h,int YOROG1B2 = 0);
•     BOOL AllocateMem();

•     BOOL LoadBMPFILE(char *fname);
•     BOOL SaveBMPFILE(char *fname);

•     HXLBMPPFILE();
•     ~HXLBMPPFILE();
• };
• #endif

```



```

■ int main(int argc, char* argv[])
■ {
■     HXLBMPPFILE bmpfile; int i,j,v;


■     if (!bmpfile.LoadBMPFILE ("c.bmp")) return 1;
■     // do other processing with the imagedata//
■     HXLBMPPFILE bf;
■     bf.imagew = bmpfile.imagew;
■     bf.imageh = bmpfile.imageh;
■     bf.iRGBnum = bmpfile.iRGBnum
■     if (!bf.AllocateMem()) return 1;

■     for (i=0;i<bmpfile.imageh;i++)
■     for (j=0;j<bmpfile.imagew;j++)
■     {
■         bf.pDataAt(i)[j]= 255-bmpfile.pDataAt (i)[j];
■     }

■     bf.SaveBMPFILE ("2.bmp");
■     printf("program ends!\n");
■     return 0;
■ }

```

# 任务

- 在VC环境
  - 建立动态库工程，包含HXLBMPPFILE类
  - 建立执行程序，包含main函数和上述动态库，调试该程序
- 编写部分程序
  - 彩色图像变为灰度图像
  - 灰度图像变为彩色图像