BMP 파일 구조

김성영교수
금오공과대학교
컴퓨터공학부

학습 목표

- BMP 파일의 구조 및 그 특징을 설명할 수 있다.
- 파일 헤더 및 비트맵 정보 헤더의 주요 필드를 구분하고 그 역할을 설명할 수 있다.
- C언어를 사용하여 BMP 파일을 처리할 수 있다.

BMP 파일 구조

File Header (BITMAPFILEHEADER)

Bitmap Info. Header (BITMAPINFOHEADER)

LUT (RGBQUAD)

Headers of BMP

Image Data

파일 헤더 (File Header)

```
typedef struct tagBITMAPFILEHEADER
{
  WORD        bfType;       // Specifies the file type
                            // It must be “BM” (4D42)
  DWORD       bfSize;       // 파일의 크기 (byte)
  WORD        bfReserved1; // reserved (항상 0)
  WORD        bfReserved2; // reserved (항상 0)
  DWORD       bfOffBits;    // 픽셀 데이터의 시작 오프셋
} BITMAPFILEHEADER;
```

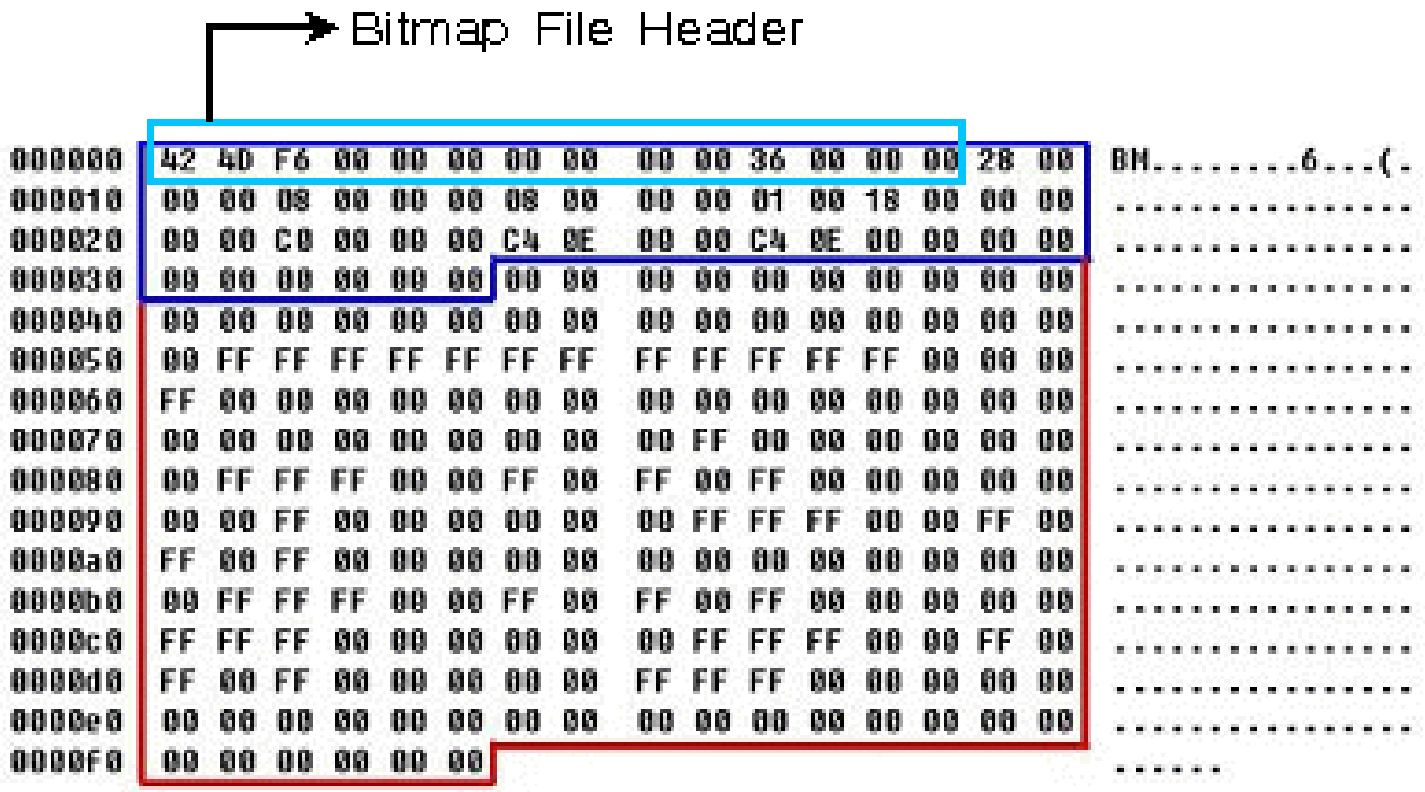
비트맵 정보 헤더 (Bitmap Info Header)

```
typedef struct tagBITMAPINFOHEADER {
   DWORD biSize;              // 구조체의 크기 (bytes)
   LONG  biWidth;             // 비트맵의 가로 길이 (pixels)
   LONG  biHeight;            // 비트맵의 세로 길이 (pixels)
   WORD  biPlanes;            // 비트 플레인 수 (항상 1)
   WORD  biBitCount;          // 픽셀당 비트수 (1,4,8,16,24,32)
   DWORD biCompression;       // 압축 유형 (BI_RGB)
   DWORD biSizeImage;         // 비트맵 데이터의 크기 (bytes)
   LONG  biXPelsPerMeter;     // 수평 해상도 (pixels/meter)
   LONG  biYPelsPerMeter;     // 수직 해상도 (pixels/meter)
   DWORD biClrUsed;           // LUT에 포함된 칼라 인덱스의 개수
   DWORD biClrImportant;      // 비트맵을 화면에 출력하기 위해
                              // 사용된 칼라 인덱스의 개수
} BITMAPINFOHEADER;
```

팔레트 (Palette)

```
typedef struct tagRGBQUAD
{
  BYTE        rgbBlue;          // B component
  BYTE        rgbGreen;         // G component
  BYTE        rgbRed;           // R component
  BYTE        rgbReserved1;     // reserved
} RGBQUAD;
```

BMP 파일 예제 (1)



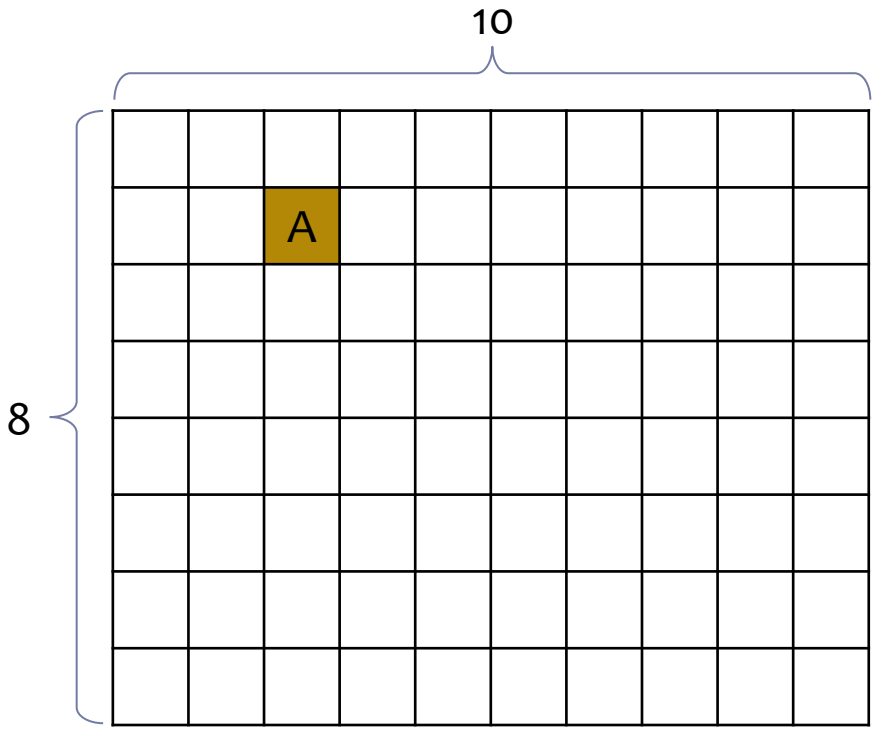
비트맵 파일의 내부 값

BMP 파일 구조 확인 실습 → sample.bmp

BMP 파일 예제 (2)

항목	값	항목	값
영상 데이터 오프셋		픽셀당 비트수	
영상 가로 길이		비트맵 데이터 크기	
영상 세로 길이		참조표의 인덱스 개수	

BMP 파일 특징 (1)



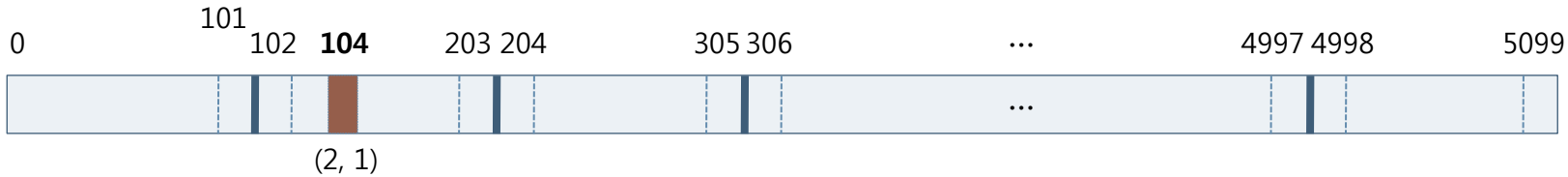
2차원 배열(data)

A ⇒ data[1][2]

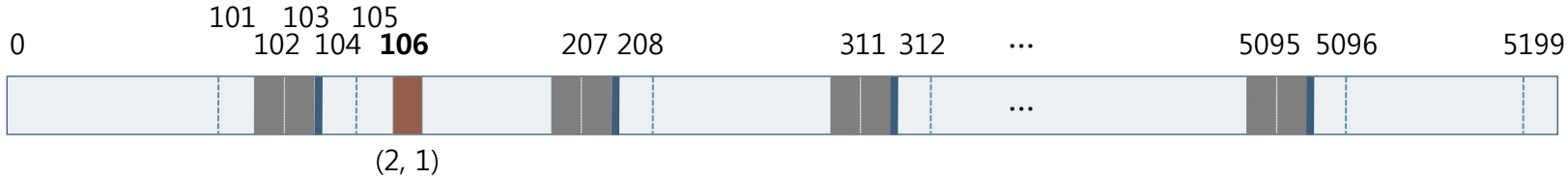
1차원 배열(data)

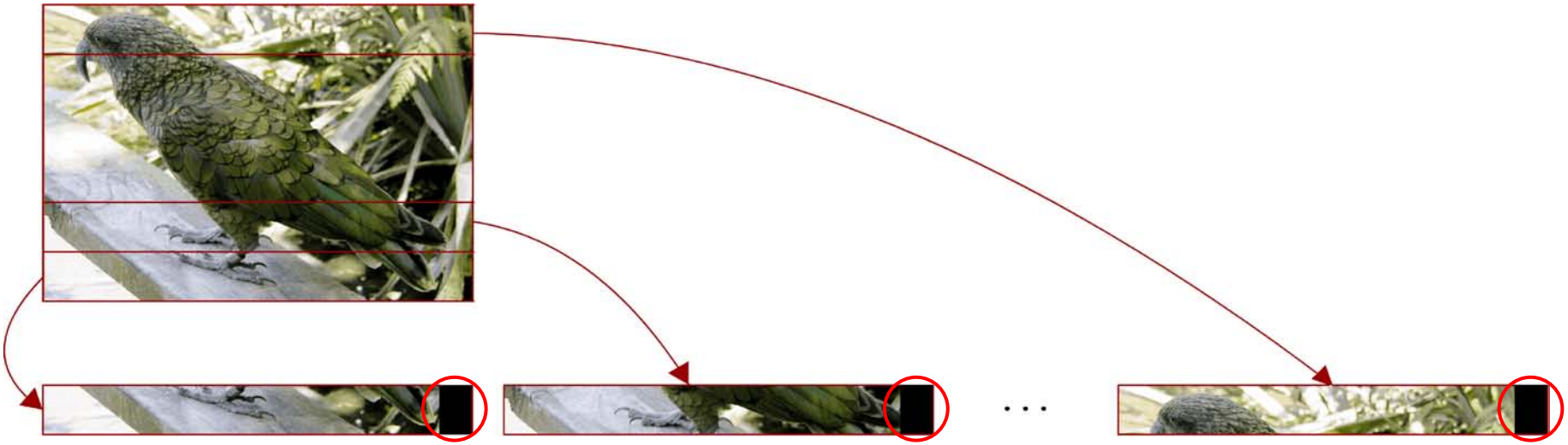
A ⇒ data[12]

BMP 파일 특징 (2)



102 x 50 8bit gray-scale image **(2, 1) ?**





BMP 파일 특징 (3)

- 비트맵 데이터는 파일 내부에서 **수직으로 반전되어** 저장

```
for(y=0; y<biHeight; y++) {
    for(x=0; x<biWidth; x++) {
        GrayImg[y][x] = Mem[(biHeight-y-1)*rwsize+x];
    }
}
```

- 비트맵 데이터의 **각 행은 반드시 4 bytes의 배수**임

```
#define WIDTHBYTES(bits) (((bits)+31)/32*4)
rwsize = WIDTHBYTES(biBitCount*biWidth);
```

예) 가로길이가 102인 8bit 그레이스케일 영상
((8*102)+31)/32*4 = (816+31)/32*4 = 847/32*4 = 26*4 = 104

Example – Inverting Image

```
/*
   이 프로그램은 8bit 그레이스케일 영상의 픽셀값을 반전한 후
   트루칼라 포맷의 영상으로 저장한다.
*/

#include <stdio.h>
#include <windows.h>

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

typedef unsigned char BYTE;

int main() {
    FILE *file;                 // file pointer
    BITMAPFILEHEADER hf;        // 파일헤더 (bmp file header)
    BITMAPINFOHEADER hInfo;     // 비트맵 정보헤더 (bitmap information header)
    int rwsize, rwsize2;        // 라인 당 바이트수 (bytes per a line)
    BYTE *lpImg;              // 입력 데이터 포인터 (pointer for input image data)
    BYTE *lpOutImg;           // 출력 데이터 포인터 (pointer for output image data)
    int x, y;
```

Cont'd

```
// 입력 영상 파일을 연다
file=fopen ("input.bmp", "rb");
if(file==NULL) {
    printf("There is no file!!!\n");
    return -1;
}

fread (&hf, sizeof(BITMAPFILEHEADER), 1, file); // 파일 헤더 읽음
if(hf.bfType!=0x4D42) // BMP 포맷 ('BM') 인지를 확인
    return -1;

fread (&hInfo, sizeof(BITMAPINFOHEADER), 1, file); // 비트맵 정보 헤더 읽음
printf("Image Size: (%3dx%3d)\n", hInfo.biWidth, hInfo.biHeight);

// 8 bit 회색톤 영상만을 입력으로 받음
if(hInfo.biBitCount != 8 || hInfo.biClrUsed != 0) {
    printf("Bad File format!!\n");
    return -1;
}

// 입출력 데이터를 위한 라인 당 바이트 수 계산
rwsize = WIDTHBYTES(hInfo.biBitCount*hInfo.biWidth);  // 입력 영상
rwsize2 = WIDTHBYTES(24*hInfo.biWidth);   // 출력 영상
```

Cont'd

```
fseek (file, hf.bfOffBits, SEEK_SET);   // 비트맵 데이터가 시작되는 위치로
                                        // 파일 포인터를 이동

// 입력 영상 데이터를 위한 메모리 할당
lpImg = (BYTE *)malloc(rwsize*hInfo.biHeight);

// 영상 데이터를 입력 영상으로 부터 읽음
fread (lpImg, sizeof(char), rwsize*hInfo.biHeight, file);

fclose (file);

// 출력 영상 데이터를 위한 메모리 할당
lpOutImg = (BYTE *)malloc(rwsize2*hInfo.biHeight);


for(y=0; y<hInfo.biHeight; y++) {
    for(x=0; x<hInfo.biWidth; x++) {
        lpOutImg[y*rwsize2+3*x+2] = 255-lpImg[y*rwsize+x];  /* R */
        lpOutImg[y*rwsize2+3*x+1] = 255-lpImg[y*rwsize+x];  /* G */
        lpOutImg[y*rwsize2+3*x+0] = 255-lpImg[y*rwsize+x];  /* B */
    }
}
```

Cont'd

```
    // 트루칼라 포맷으로 변환 영상을 저장
    hInfo.biBitCount = 24;
    hInfo.biSizeImage = rwsize2*hInfo.biHeight;
    hInfo.biClrUsed = hInfo.biClrImportant = 0;
    hf.bfOffBits = 54; // There is no palette
    hf.bfSize = hf.bfOffBits + hInfo.biSizeImage;

    file = fopen("output.bmp", "wb");

    fwrite (&hf, sizeof(char), sizeof(BITMAPFILEHEADER), file);
    fwrite (&hInfo, sizeof(char), sizeof(BITMAPINFOHEADER), file);
    fwrite (lpOutImg, sizeof(char), rwsize2*hInfo.biHeight, file);

    fclose(file);

    // 메모리 해제
    free(lpOutImg);
    free(lpImg);

    return 0;
}
```

Cont’d

Lab.

- 이전 예제에 대해 다음과 같이 기능을 확장하라.
  - 트루 칼라 (true color) 포맷을 지원
    - 입력 가능 포맷: 8bit gray-scale, **true color** images
    - 출력 포맷: true color
  - 영상 좌측 상단에 50x50 크기의 사각형을 그림
    - 회색톤 영상은 밝기값 128, 칼라 영상은 빨간색(255,0,0)을 사용
  - 수업용 게시판에 있는 테스트 영상 사용
    - Gray-scale image: testG.bmp, TVG.bmp
    - True color image: test.bmp, TV.bmp
  - Character Set : 멀티바이트 문자 집합 사용

Cont’d

학습정리 (1)

1. BMP 파일 구조

- File Header (BITMAPFILEHEADER), Bitmap Information Header (BITMAPINFOHEADER), LUT (RGBQUAD) 및 Image Data로 구성

2. BMP 파일 구조의 특징

- 영상 데이터는 상하가 반전되어 있음
- 영상의 각 행의 데이터양은 4바이트의 배수이어야 함

학습정리 (2)

3. File Header

- BMP 식별자 ("BM"), 파일 크기 및 픽셀 데이터의 시작 위치 등의 정보 포함

4. Bitmap Information Header

- 영상 가로 및 세로 길이, 픽셀당 비트수, 영상 데이터 크기 및 LUT에 포함된 엔트리 개수 등의 정보 포함