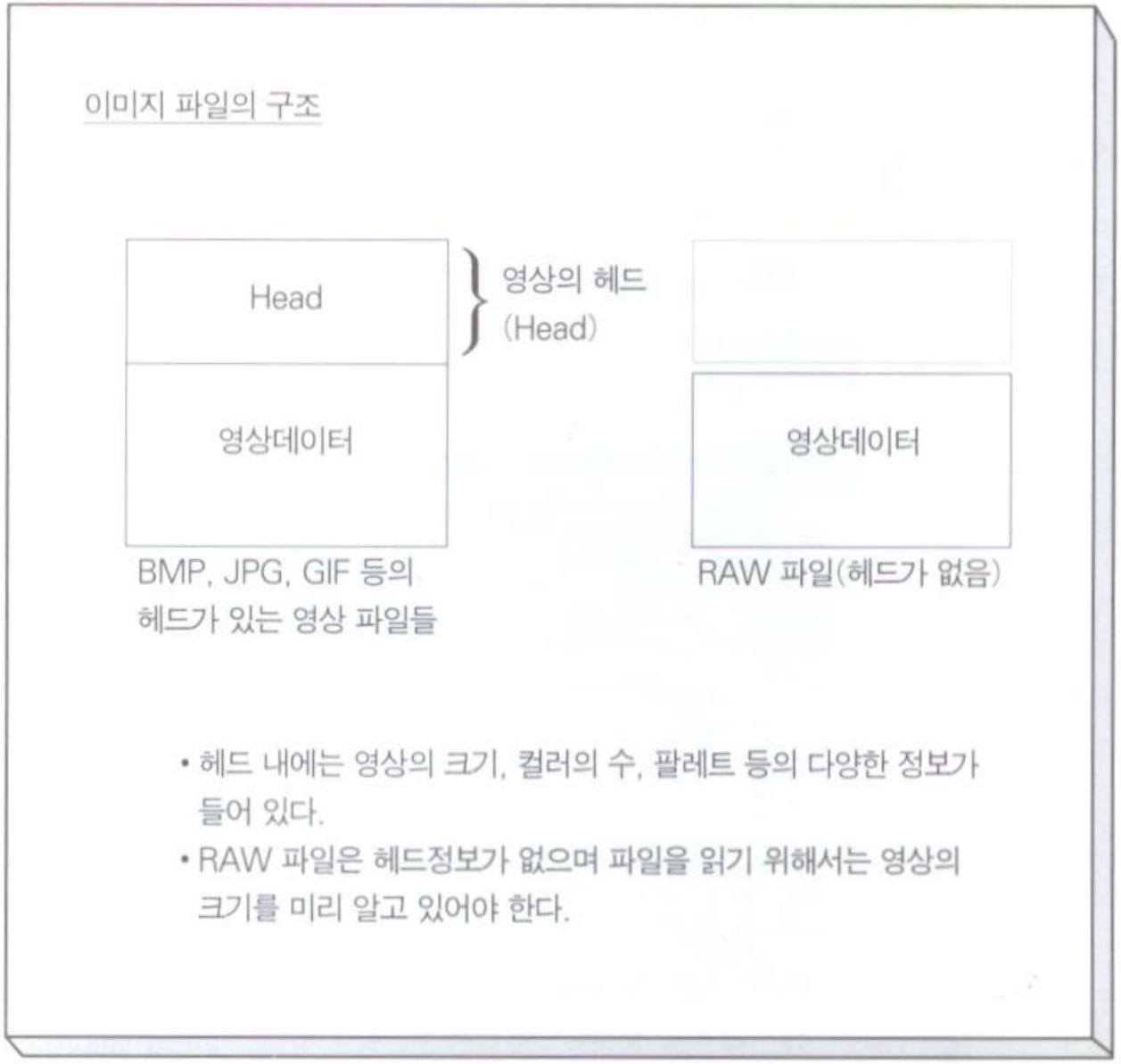
BMP File (비트맵 파일)

MediaLab.
Dowan Kwon

Contents of Table

Image File

- Non Compression
  - RAW / BMP
- Compression
  - JPG / GIF / PCX …



❖ BMP File

File Header

Bitmap Info Header

LUT(Palette)

File 의 정보

Bitmap 영상의 정보

(필요에 따라)
미리 정의된 색상 테이블

Pixel Data

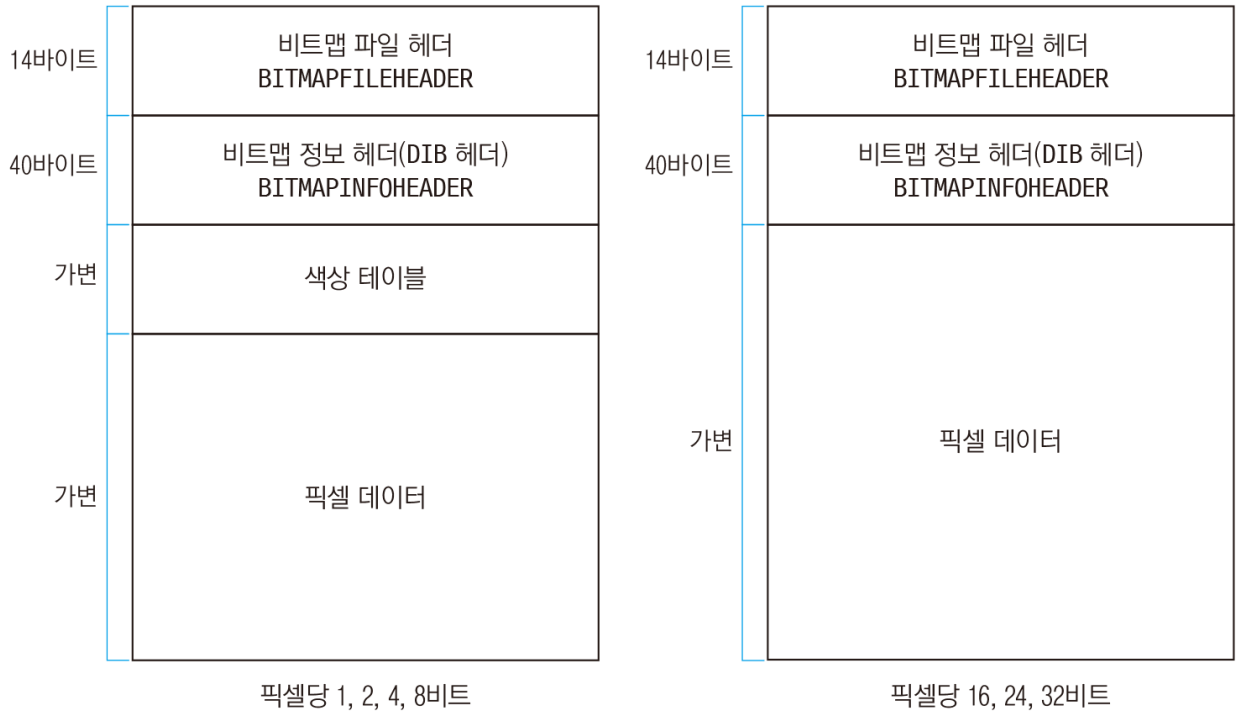
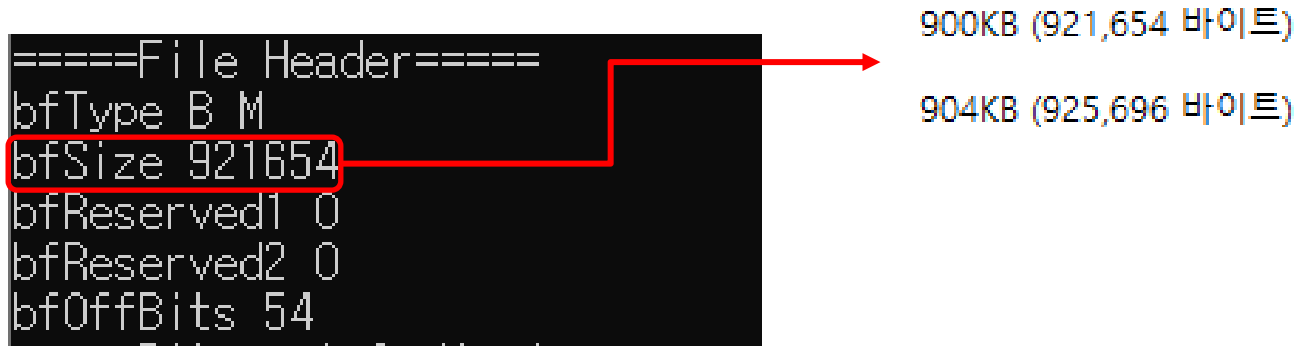


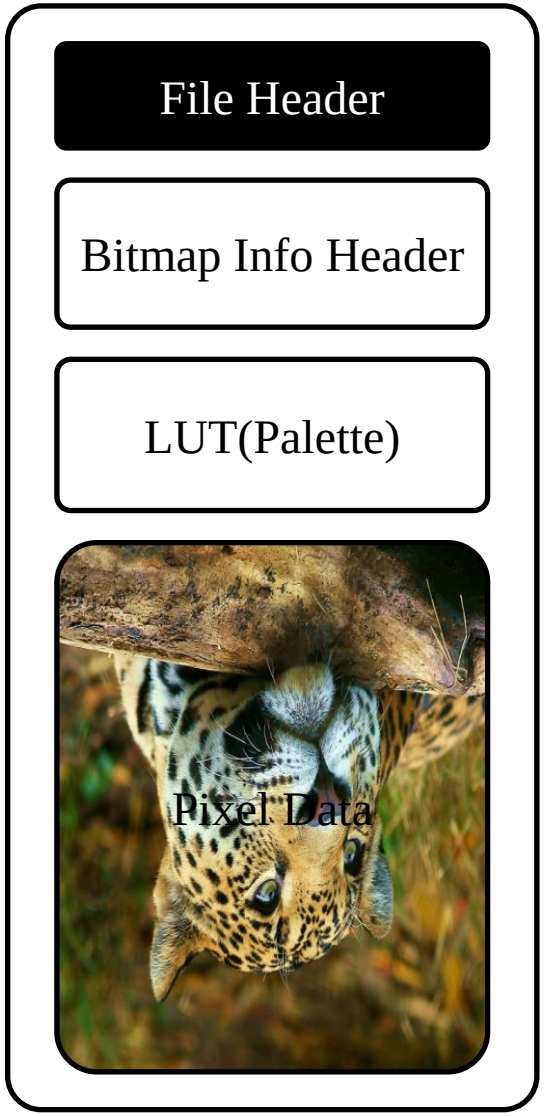
Fig1. 각 비트 BMP File의 구조

❖ BMP File 구조

멤버	크기(byte)	설명
bfType	2	BMP File 매직 넘버. ASCII 코드로 0x42(B), 0x4D(M)가 저장
bfSize	4	파일 크기(바이트)
bfReserved1	2	현재는 사용하지 않으며 미래를 위해 예약된 공간
bfReserved2	2	현재는 사용하지 않으며 미래를 위해 예약된 공간
bfOffBits	4	비트맵 데이터의 시작 위치

Fig2. File Header의 구조





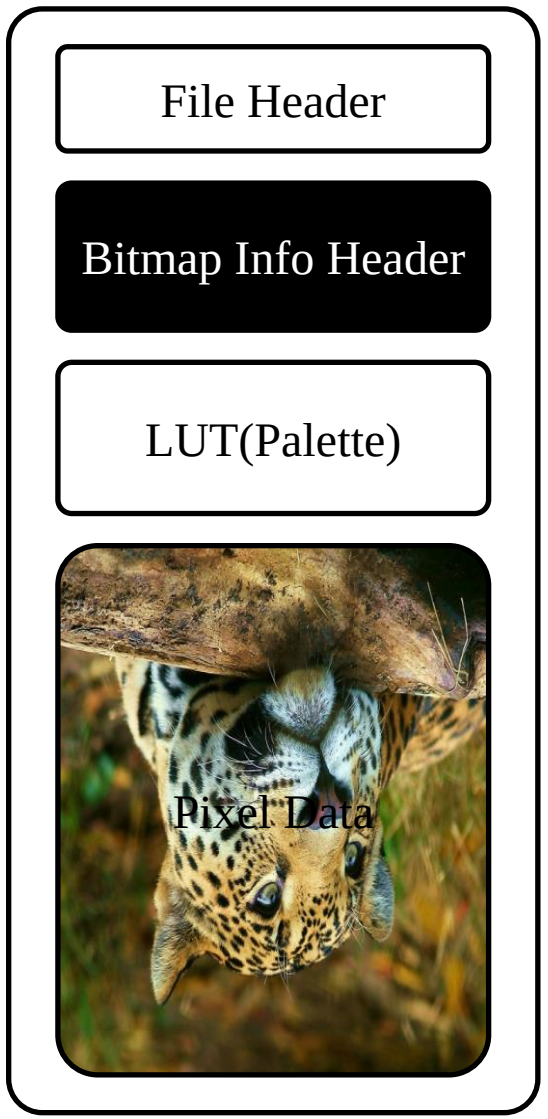
❖ BMP File 구조

멤버	크기(byte)	설명
biSize	4	현재 비트맵 정보 헤더(BITMAPINFOHEADER)의 크기
biWidth	4	비트맵 이미지의 가로 크기(픽셀)
biHeight	4	비트맵 이미지의 세로 크기(픽셀)
biPlanes	2	사용하는 색상판의 수. 항상 1
biBitCount	2	픽셀 당 비트 수 (1, 4, 8, 16, 24, 32 중 하나)
biCompression	4	압축 방식. 보통 BMP는 압축을 하지 않으므로 0
biSizeImage	4	비트맵 이미지의 픽셀 데이터 크기(압축 되지 않은 크기)
biXPelsPerMeter	4	그림의 가로 해상도(미터당 픽셀)
biYPelsPerMeter	4	그림의 세로 해상도(미터당 픽셀)
biClrUsed	4	색상 테이블에서 실제 사용되는 색상 수
biClrImportant	4	중요한 색의 수, 일반적으로 무시

Fig3. Bitmap Info Header의 구조

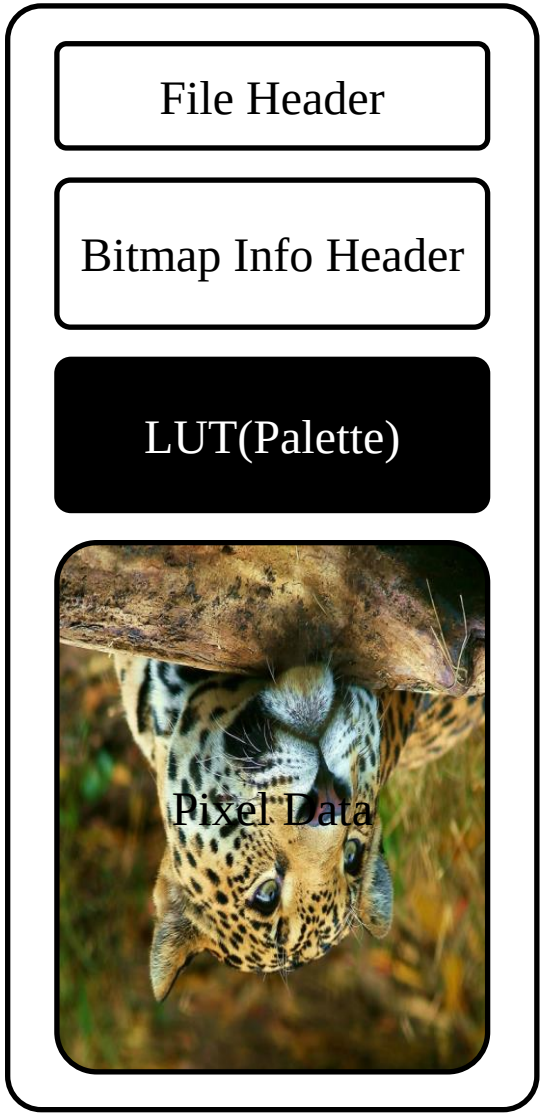
```
=====Bitmap Info Header=====
biSize 40
biWidth 640
biHeight 480
biPlanes 1
biBitCount 24
biCompression 0
biSizeImage 0
biXPelsPerMeter 3780
biYPelsPerMeter 3780
biClrUsed 0
biClrImportant 0
```

한픽셀이 24비트다



❖ BMP File 구조

- Look up table 방식
- 미리 정해진 색상의 집합
- 영상의 색상 수만큼 기록됨
- 영상의 픽셀은 색상 값을 가지는 것이 아닌
  그 픽셀이 사용하는 Palette 색상의 번호를 가지게 됨

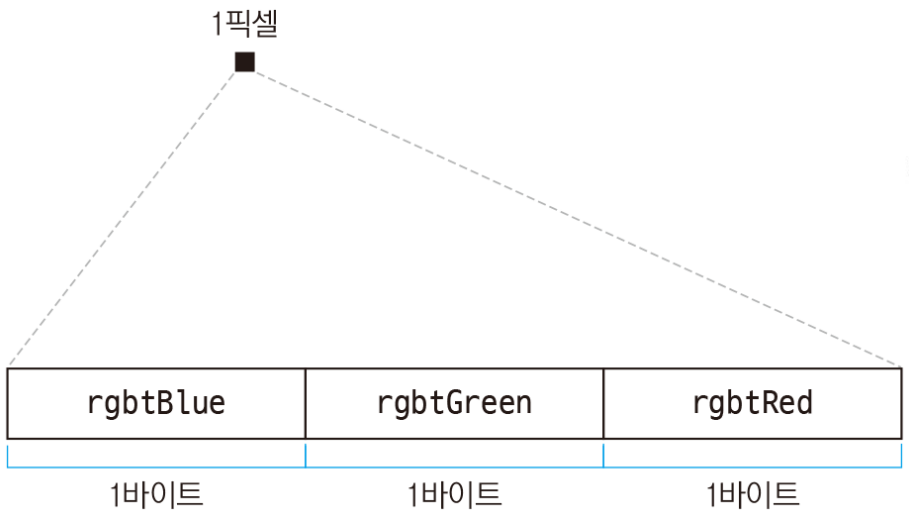


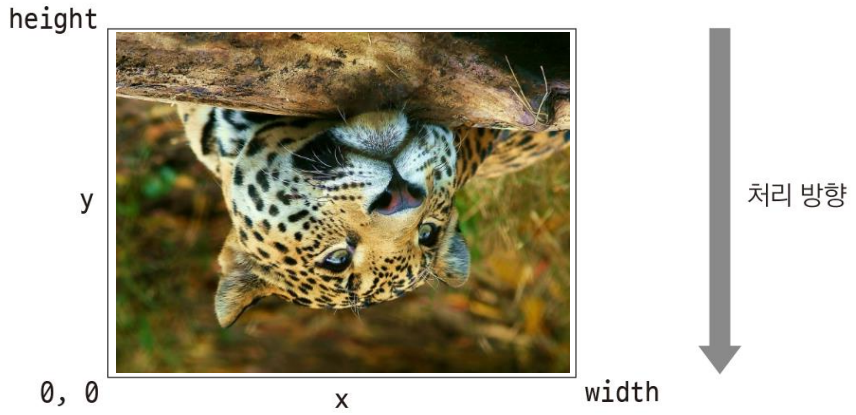
❖ BMP File 구조 (24bit)

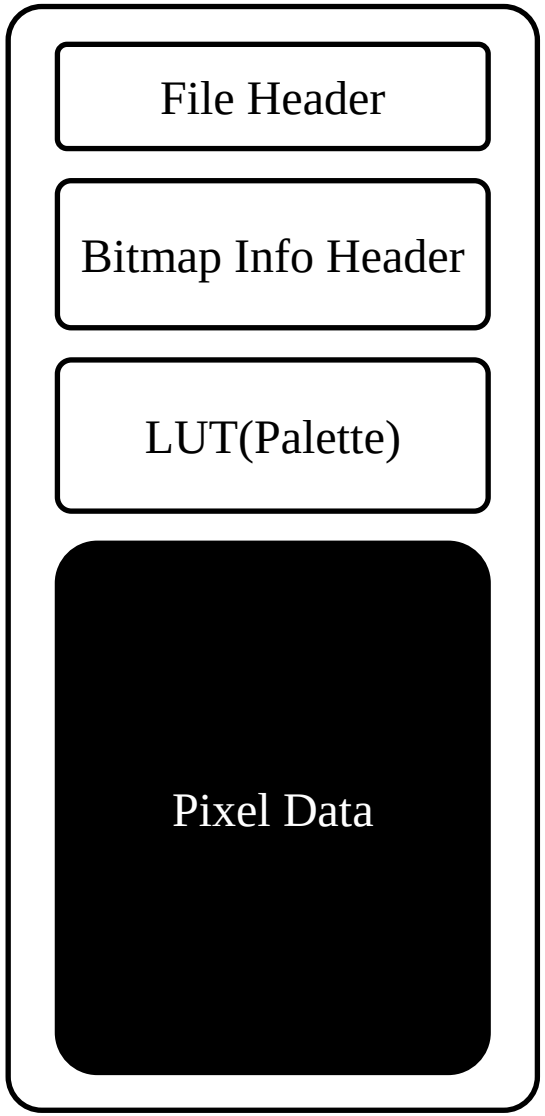
- R, G, B * 8bit = 24bit $(2^8 * 2^8 * 2^8 = 2^{24} = 1{,}600만)$
- 한 행(가로)의 길이는 **4바이트의 배수**
  - 4바이트의 배수가 되도록 0으로 Padding
  - ex) 가로가 9 Pixel 이라면 실제 저장되는 가로 크기는 27바이트가 아닌 28바이트
- **상하반전**되어 저장됨

멤버	크기(byte)	설명
rgbtBlue	1	파랑
rgbtGreen	1	초록
rgbtRed	1	빨강

Fig4. 24비트 비트맵의 픽셀 구조







❖ Code

```
#define _CRT_SECURE_NO_WARNINGS //fopen 오류 해결
#include <stdio.h> // FILE*, fseek, fread 등 사용
#define DATA_OFFSET_OFFSET 0x000A
#define WIDTH_OFFSET 0x0012
#define HEIGHT_OFFSET 0x0016
#define BITS_PER_PIXEL_OFFSET 0x001C
#define HEADER_SIZE 14
#define INFO_HEADER_SIZE 40
#define NO_COMPRESION 0
#define MAX_NUMBER_OF_COLORS 0
#define ALL_COLORS_REQUIRED 0
```

보안문제 해결

16진수 숫자 10

BMP 파일 읽을때 필요한 정보를 미리 기록해둠

❖ Code

```
typedef unsigned int int32;
typedef short int16;
typedef unsigned char byte;
```

Unsigned int 를 int32라 하겠다.
그냥 단순히 별칭을 만들어줌.

```
void ReadImage(const char* fileName, const char* outfileName, byte** pixels, int32* width, int32* height,
int32* bytesPerPixel, FILE*& imageFile, FILE*& OUT)
```

Const char* 포인터 2개와 byte 자료형 이중포인터 한 개, 포인터의 참조로 구성되어짐

❖ Code

```
void ReadImage(const char* fileName, const char* outfileName, byte** pixels, int32* width, int32* height,
int32* bytesPerPixel, FILE*& imageFile, FILE*& OUT)
```

```
void InitializePointer(int*& ptrRef) {
    ptrRef = new int(5); // ptrRef는 'int' 타입의 새로운 객체를 가리킴.
}

int main() {
    int* myPtr = nullptr; // 초기에는 nullptr을 가리키는 포인터
    InitializePointer(myPtr);// 이제 myPtr는 값 5를 가진 int 타입의 메모리를 가리킴.
    delete myPtr; // 할당된 메모리 해제
    return 0;
}
```

❖ Code

```
imageFile = fopen(fileName, "rb");//파일을 바이너리 모드로 열기
int32 data0ffset; //데이터 시작 위치 주소값
int32 LookUpTable=0;
fseek(imageFile, HEADER_SIZE + INFO_HEADER_SIZE-8, SEEK_SET); //fseek(파일변수,이동byte,기준위치)
fread(&LookUpTable, 4, 1, imageFile); //fread(메모리주소,크기,갯수,파일변수)
fseek(imageFile, 0, SEEK_SET);

OUT = fopen(outfileName, "wb");
```

1. 이미지 파일로 들어가서 파일처음에서부터
2. 헤더사이즈 (14) + 인포헤더사이즈(40) 에서 -8 만큼 한 위치에 포인터를 위치시킨다.
3. 그리고 그 뒤 4바이트만큼을 읽어오겠다. 그리고 그 값을 Lookuptable에 저장.
4. 그런데, 보통 컬러이미지 (24비트)는 이 lookuptable 값이 '0'임.
5. 즉, 색상테이블이 따로 존재하는게 아니라 픽셀이 직접적으로 (R,G,B)를 가지고 있음.

멤버	크기 (byte)	설명
biSize	4	현재 비트맵 정보 헤더(BITMAPINFOHEADER)의 크기
biWidth	4	비트맵 이미지의 가로 크기(픽셀)
biHeight	4	비트맵 이미지의 세로 크기(픽셀)
biPlanes	2	사용하는 색상판의 수. 항상 1
biBitCount	2	픽셀 당 비트 수 (1, 4, 8, 16, 24, 32 중 하나)
biCompression	4	압축 방식. 보통 BMP는 압축을 하지 않으므로 0
biSizeImage	4	비트맵 이미지의 픽셀 데이터 크기(압축 되지 않은 크기)
biXPelsPerMeter	4	그림의 가로 해상도(미터당 픽셀)
biYPelsPerMeter	4	그림의 세로 해상도(미터당 픽셀)
biClrUsed	4	색상 테이블에서 실제 사용되는 색상 수
biClrImportant	4	중요한 색의 수, 일반적으로 무시

❖ Code

```
int header = 0;
if (LookUpTable)
    header = HEADER_SIZE + INFO_HEADER_SIZE + 1024;
else
    header = HEADER_SIZE + INFO_HEADER_SIZE;
for (int i = 0; i < header; i++) // 원본 BMP 파일에
서 헤더와 테이블 뽑아서 새로운 BMP 파일의 헤더에 써
줌
{
    int get = getc(imageFile);
    putc(get, OUT);
}
```

색상테이블이 있을때, 즉 (8비트 일때)
256가지의 색상을 표현 가능하며,
이때 픽셀 하나가 256가지색만 표현가능.
그래서 색상 테이블이 존재 그리고 크기는 1024

보통 컬러 이미지(24bit)면 없음.
따라서 else로 빠짐

이미지 파일의 헤더정보를 가져와 out에 입력

❖ Code

```
fseek(imageFile, DATA_OFFSET_OFFSET, SEEK_SET); //fseek(파일변수,이동byte,기준위치)
fread(&dataOffset, 4, 1, imageFile); //fread(메모리주소,크기,갯수,파일변수)
fseek(imageFile, WIDTH_OFFSET, SEEK_SET);
fread(width, 4, 1, imageFile);
fseek(imageFile, HEIGHT_OFFSET, SEEK_SET);
fread(height, 4, 1, imageFile);
int16 bitsPerPixel;
fseek(imageFile, BITS_PER_PIXEL_OFFSET, SEEK_SET);
fread(&bitsPerPixel, 2, 1, imageFile);
*bytesPerPixel = ((int32)bitsPerPixel) / 8; //3 bytes per pixel
```

1. 파일 처음에서 데이터옵셋(10) 바이트 만큼의 위치로 포인터 이동
2. 이곳에서 4바이트만큼 읽어들임.
3. 그리고 너비, 높이 정보와 픽셀당 몇 비트 인지 읽어드림.

4. 픽셀당 비트수는 bitsperpixel 에 저장되며
5. 이 값을 8로 나눠 '픽셀당 바이트 ' 변수에 저장

멤버	크기(byte)	설명
bfType	2	BMP File 매직 넘버. ASCII 코드로 0x42(B), 0x4D(M)가 저장
bfSize	4	파일 크기(바이트)
bfReserved 1	2	현재는 사용하지 않으며 미래를 위해 예약된 공간
bfReserved 2	2	현재는 사용하지 않으며 미래를 위해 예약된 공간
bfOffBits	4	비트맵 데이터의 시작 위치

❖ Code

```
int paddedRowSize = (int)(((*width * bytesPerPixel) + 3) / 4) * 4; //4의 배수로 만들어주는 과정
int unpaddedRowSize = (*width) * (*bytesPerPixel);
int totalSize = unpaddedRowSize * (*height);
```

BMP 파일은 각 행이 4바이트의 배수가 되어야함.
그래서 4의 배수로 만들어주는 과정을 거친 후,

실제 데이터의 사이즈, 총사이즈등을 구함

```
*pixels = new byte[totalSize];
    int i = 0;
    byte* currentRowPointer = *pixels + ((*height - 1) * unpaddedRowSize);
    for (i = 0; i < *height; i++)
    {
        fseek(imageFile, dataOffset + (i * paddedRowSize), SEEK_SET);
        fread(currentRowPointer, 1, unpaddedRowSize, imageFile);
        currentRowPointer -= unpaddedRowSize;
    }
```

Pixels는 byte** 이중포인터이므로,
Pixels 는 byte 와 동일.

Byte * 에 동적할당.

Fseek 를 통해 픽셀의 첫번째 행 (즉 이미지상 마지막 줄)을 totalSize 배열의 마지막 행으로 집어 넣음.

totalSize 안에선 우리가 보는 이미지와 동일하게 윗줄부터 값들이 나열되어있음.

imageFile

totalSize

❖ Code

```
void WriteImage(byte* pixels, int32 width, int32 height, int32 bytesPerPixel, FILE*& outputFile)
{
    int paddedRowSize = (int)(((width * bytesPerPixel) + 3) / 4.0f) * 4;
    int unpaddedRowSize = width * bytesPerPixel;
    for (int i = 0; i < height; i++)
    {
        int pixelOffset = ((height - i) - 1) * unpaddedRowSize;
        fwrite(&pixels[pixelOffset], 1, paddedRowSize, outputFile);
    }
    fclose(outputFile);


}
```

Pixeloffset = 어떠한 데이터의 마지막 행
Fwrite(&pixels[pixelOffset],1,paddedRowSize,outputFile)

Pixels 배열(new byte [totalSize])의 pixelOffset 인덱스의 주소값으로부터, 1만큼의 크기로, paddedRowSize 만큼 읽어서 outputFile에 입력한다.)

그 이후 pixeloffset 은 마지막 행의 전 행으로 바뀜. (ex 10 → 9 행)

❖ Code

```
int main()
{
    byte* pixels;
    int32 width;
    int32 height;
    int32 bytesPerPixel;
    FILE* imageFile; //파일 포인터
    FILE* outputFile;
    ReadImage("Lena.bmp", "Lena_out.bmp", &pixels, &width, &height, &bytesPerPixel, imageFile, outputFile);
    WriteImage(pixels, width, height, bytesPerPixel, outputFile);
    delete[] pixels;

    return 0;
}
```

Byte형 픽셀 포인터 선언 후 이 포인터의 주소값을 ReadImage에 전달.
마찬가지로 각각 파일 포인터또한 ReadImage 함수의 인자로 전달.

마지막 동적할당 해제.

Coding Homework

❖ Q. 입력된 BMP(24bit) File을 R, G, B 3개의 BMP(24bit) File로 분리하세요.

Input

BMP File

R

G

B

Output

Coding Homework

Lab 1.

제출일		학번	
전공		이름	

① 문제	
② 주요 변수	
③ 알고리즘	
④ 결과	

1. 파일명 : "실감미디어_1st_학번_이름.zip"
   - EX) 실감미디어_1st_2023000100_홍길동.zip
2. 압축 파일 구성
   - 소스 파일, .hwp(.doc) 2개 파일 (Python 의 경우)
   - 소스 파일, .hwp(.doc), .exe 3개 파일 (C/C++ 의 경우)
3. 파일 내용
   - 보고서 : 이름, 학번, 문제, 주요 변수, 알고리즘, 결과
   - (.py, .ipynb)파이썬 소스코드 or (.cpp, .c) C/C++ 소스 코드
   - .exe : 실행파일 (C/C++ 의 경우만 해당)
4. 주의 사항
   - 제출이 늦을 경우 감점은 있지만, 학기 종료까지 제출하시면 점수가 있습니다.

조　교 : 권도완
이메일 : kdwys97@khu.ac.kr
연구실 : Media Lab.3 (전자정보대학 567호)


Thank
you