2-1. 영상파일의 구조

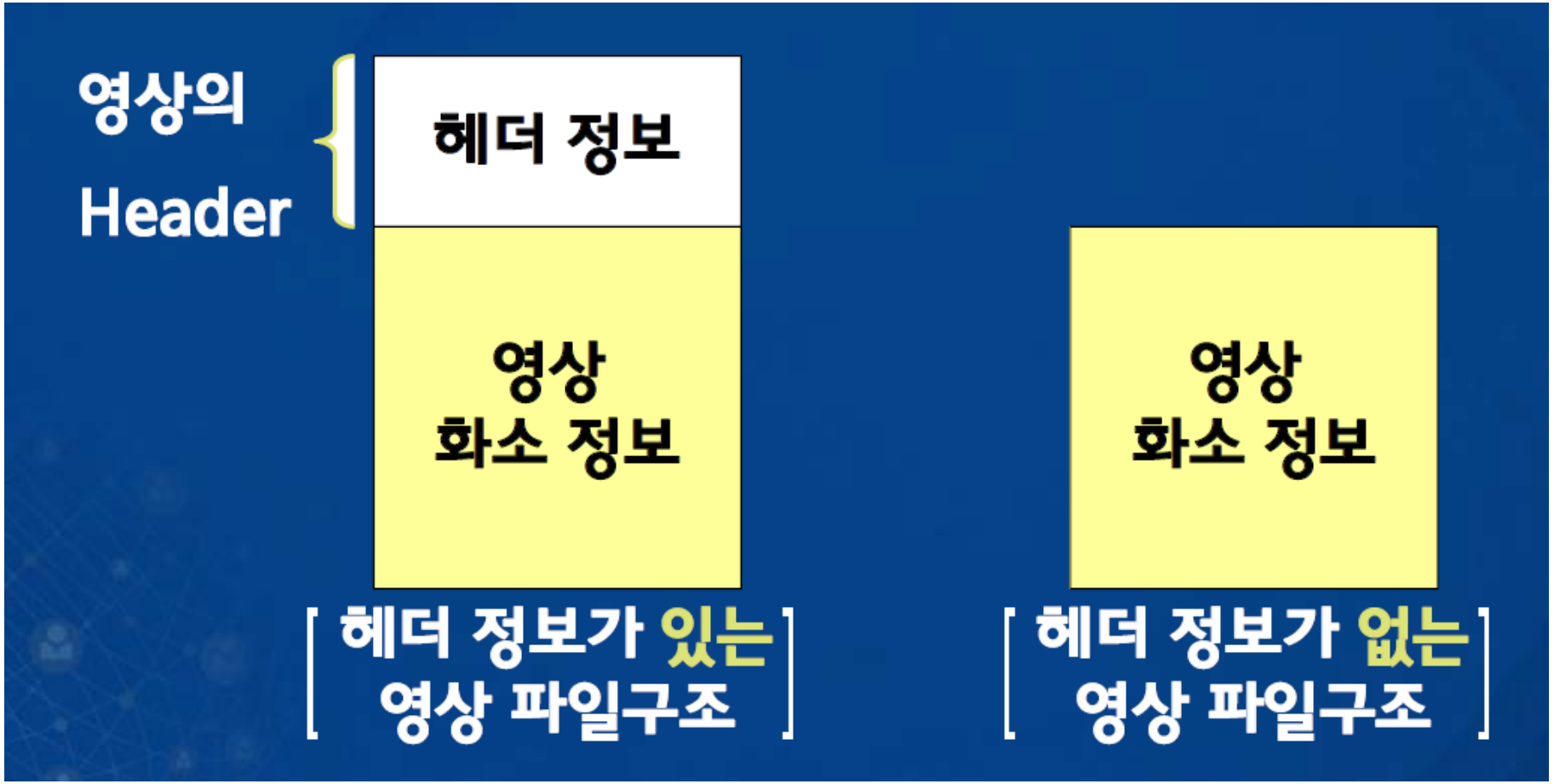
영상파일 형식

[Macromedia Director File Format]

File Import					File Export		Native
Image	Palette	Sound	Video	Anim.	Image	Video	
.BMP, .DIB, .GIF, .JPG, .PICT, .PNG, .PNT, .PSD, TGA, .TIFF, .WMF	.PAL .ACT	.AIFF .AU .MP3 .WAV	.AVI .MOV	.DIR .FLA .FLC .FLI .GIF .PPT	.BMP	.AVI .MOV	.DIR .DXR .EXE

BMP : 어떤 압축도 하지 않고 영상의 화소 정보를 그대로 저장, 주로 사용

영상처리 파일 형식

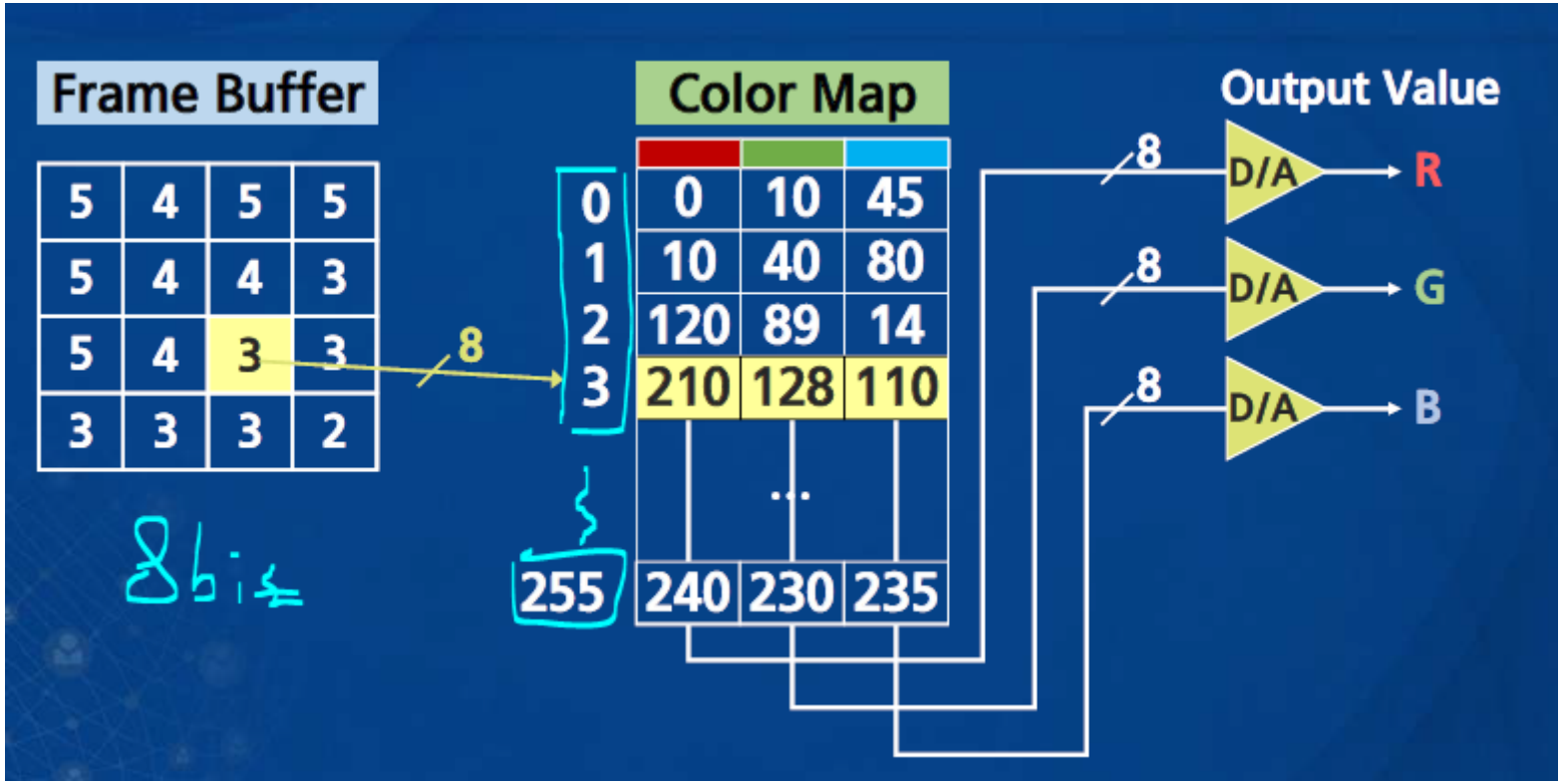


RAW 파일은 헤더 정보가 없다

트루컬러 vs인덱스컬러 영상

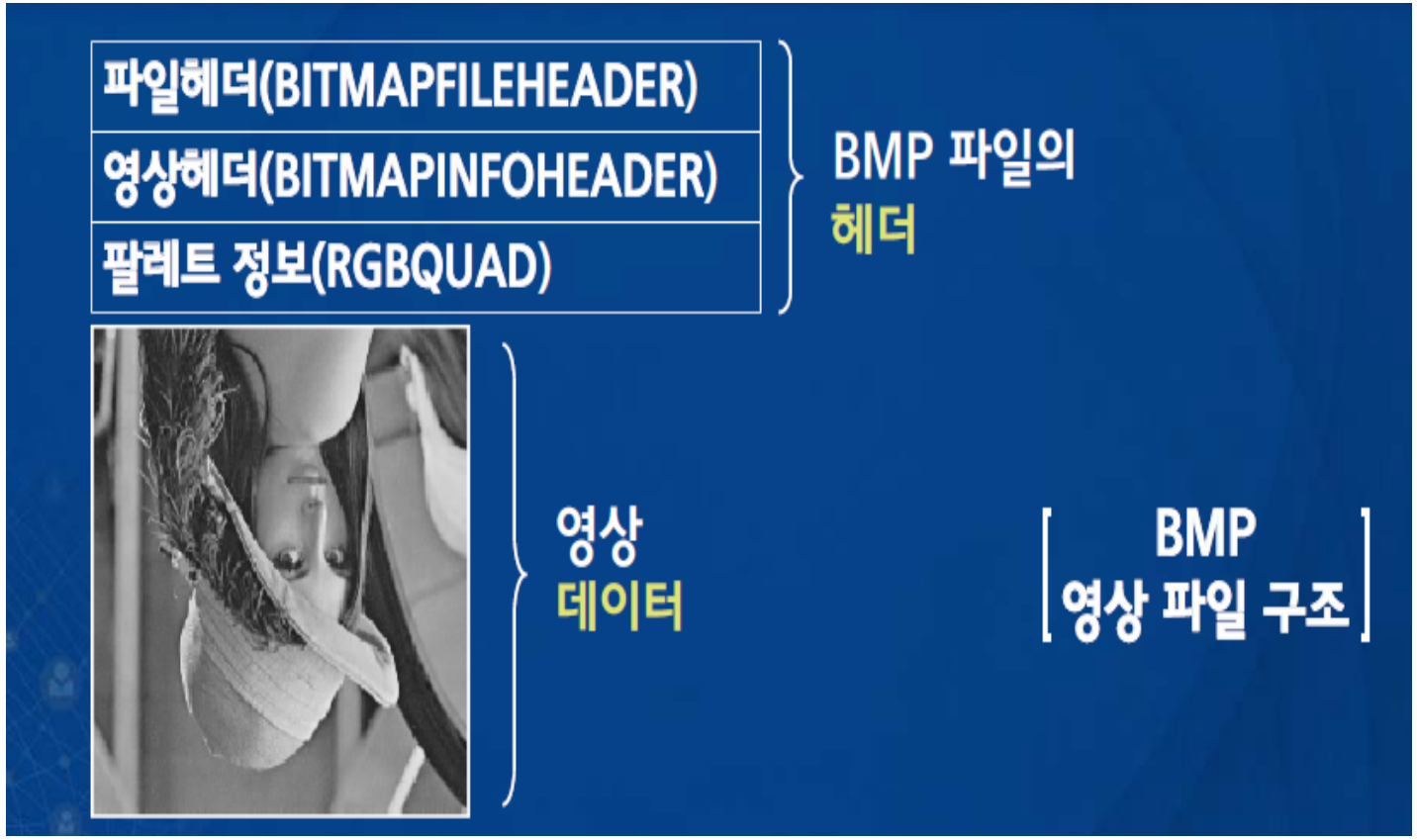
- 트루컬러 영상
  - RGB를 각 8bits 씩 다 써서 한 픽셀의 색상을 표현
  - 2^24승 만큼 컬러 표현 가능
- 인덱스 컬러 영상
  - 트루 컬러 중 영상에서 사용하는 컬러만 256개 뽑아서 사용

Color Map



Color Map은 팔렛트 (=LUT)에 물감을 짜놓는 것과 같다.
R,G,B 값을 지정해놓음

BMP



파일헤더 : 14Bytes
영상헤더 : 40Bytes
팔레트 : 1024Bytes
= 14+40+1024

영상 y좌표 반대로 고려해야함

파일헤더

WORD : 2Bytes
DWORD : 4Bytes
총 14Bytes

WORD bfType에서 bf변수가 bm인 경우 BMP형식임을 의미

영상헤더

영상헤더정보(BITMAPINFOHEADER)

```
typedef struct tagBITMAPINFOHEADER
{
    DWORD   biSize;          // 이 구조체의 크기
    LONG    biWidth;         // 영상의  가로 크기
    LONG    biHeight;        // 영상의 세로 크기
    WORD    biplanes;        // 비트플레인 수(항상 1)
    WORD    biBitCount;      // 화소당 비트수(컬러, 흑백구별)
    DWORD   biCompression;   // 압축유무
    DWORD   biSizeImage;     // 영상의 크기(바이트 단위)
    LONG    biXPelsPerMeter; // 가로 해상도
    LONG    biYPelsPerMeter; // 세로 해상도
    DWORD   biClrUsed;       // 실제 사용 색상수
    DWORD   biClrImportant;  // 중요한 색상인덱스
} BITMAPINFOHEADER;
```

DWORD : 4Bytes
LONG : 4Bytes
총 40Bytes

biSzize : 40Bytes
biWidth, biHeight
: 640 x 480
biBitcount
: gray color – 8
: true color – 24

팔레트 정보

팔레트 정보(RGBQUAD)

- 인덱스에 의한 컬러값을 저장하기 위한 구조체

```
typedef struct tagRGBQUAD
{
  BYTE  rgbBlue;          // B 성분(파란색)
  BYTE  rgbGreen;         // G 성분(녹색)
  BYTE  rgbRed;           // R 성분(빨간색)
  BYTE  rgbReserved1;     // 예약된 변수, 사용하지 않음
} RGBQUAD;
```

4Bytes x 256개
= 1024Bytes

Reserved : 투명도

2-2. C언어를 통한 영상파일 입출력

프로세스

1. 영상 입력
2. 영상 처리
3. 처리된 결과 출력

소스 코드

```c
#include <stdio.h>
#include <stdlib.h>
#include <Windows.h>
void main()
{
    BITMAPFILEHEADER hf; // BMP 파일헤더 14Bytes
    BITMAPINFOHEADER hInfo; // BMP 인포헤더 40Bytes
    RGBQUAD hRGB[256]; // 팔레트 (256 * 4Bytes)
    FILE *fp;
    fp = fopen("lenna.bmp", "rb");
    if(fp == NULL) return;
    fread(&hf, sizeof(BITMAPFILEHEADER), 1, fp);
    fread(&hInfo, sizeof(BITMAPINFOHEADER), 1, fp);
    fread(hRGB, sizeof(RGBQUAD), 256, fp);
    int ImgSize = hInfo.biWidth * hInfo.biHeight;
    BYTE * Image = (BYTE *)malloc(ImgSize);
    BYTE * Output = (BYTE *)malloc(ImgSize);
    fread(Image, sizeof(BYTE), ImgSize, fp);
    fclose(fp);
```

```c
/* ... */
for(int i=0; i<ImgSize; i++)
    Output[i] = 255 - Image[i];
```

```c
fp = fopen("output.bmp", "wb");
fwrite(&hf, sizeof(BYTE), sizeof(BITMAPFILEHEADER), fp);
fwrite(&hInfo, sizeof(BYTE), sizeof(BITMAPINFOHEADER), fp);
fwrite(hRGB, sizeof(RGBQUAD), 256, fp);
fwrite(Output, sizeof(BYTE), ImgSize, fp);
fclose(fp);
free(Image);
free(Output);
}
```

코드 내용

- <stdio.h>, <stdlib.h> 파일 입출력
- <Windows.h>파일헤더, 영상헤더, 팔레트
- Read Binary = “rb”
- 파일포인터 fp는 “lenna.bmp” 파일의 처음 위치
- 파일헤더, 영상헤더, 팔레트 사이즈 할당
- 영상 화소 읽어(width, height) 동적할당 image(원래영상) output(처리된영상)
- 영상을 반전시키는 코드
- 메모리 leak을 막기 위한 free

실습

```c
#include<stdio.h>
#include<stdlib.h>
#include<Windows.h>

void main()
{
    /* 1. 영상 읽기 */
    BITMAPFILEHEADER hf1; // BMP 파일헤더 14Bytes
    BITMAPINFOHEADER hInfo1; //BMP 인포헤더 40Bytes
    RGBQUAD hRGB1[256]; //팔레트 (256 * 4Bytes)
    BITMAPFILEHEADER hf2; // BMP 파일헤더 14Bytes
    BITMAPINFOHEADER hInfo2; //BMP 인포헤더 40Bytes
    RGBQUAD hRGB2[256]; //팔레트 (256 * 4Bytes)
    FILE* fp1;
    FILE* fp2;
    fp1 = fopen("lenna.bmp", "rb"); //파일 열기
    fp2 = fopen("lenna.bmp", "rb"); //파일 열기
    if (fp1 == NULL) return;
    if (fp2 == NULL) return;

    fread(&hf1, sizeof(BITMAPFILEHEADER), 1, fp1); //lenna.bmp의 파일헤더 읽기
    fread(&hInfo1, sizeof(BITMAPINFOHEADER), 1, fp1); //lenna.bmp의 인포헤더 읽기
    fread(hRGB1, sizeof(RGBQUAD), 256, fp1); //lenna.bmp의 팔레트 읽기
    fread(&hf2, sizeof(BITMAPFILEHEADER), 1, fp2); //lenna.bmp의 파일헤더 읽기
    fread(&hInfo2, sizeof(BITMAPINFOHEADER), 1, fp2); //lenna.bmp의 인포헤더 읽기
    fread(hRGB2, sizeof(RGBQUAD), 256, fp2); //lenna.bmp의 팔레트 읽기

    int ImgSize = hInfo1.biWidth * hInfo1.biHeight;//동일한 원본이미지 사용
    BYTE* Image = (BYTE*)malloc(ImgSize);
    BYTE* Output1 = (BYTE*)malloc(ImgSize); //Output1 : 원본보다 밝은 이미지
    BYTE* Output2 = (BYTE*)malloc(ImgSize); //Output2 : 원본보다 어두운 이미지
    fread(Image, sizeof(BYTE), ImgSize, fp1);
    fread(Image, sizeof(BYTE), ImgSize, fp2);
    fclose(fp1);
    fclose(fp2);
```

실습

```
/* 2. 영상 처리 */
int bright = 0;
printf("값을 입력하시오 : ");
scanf("%d", &bright);

//클리핑 : 픽셀 값이 255 보다 높으면 255로, 0보다 작으면 0으로 고정시키는 작업
for (int i = 0; i < ImgSize; i++) {
    Output1[i] = Image[i] + bright;
    if (Image[i]+bright >= 255) // 밝아지는 경우
        Output1[i] = 255;
    else Output1[i] = Output1[i];

    Output2[i] = Image[i] - bright;
    if (Image[i] - bright <= 0) // 어두워지는 경우
        Output2[i] = 0;
    else Output2[i] = Output2[i];
}
```

```
/* 3. 영상 처리 결과 저장 */
fp1 = fopen("output1.bmp", "wb");
fp2 = fopen("output2.bmp", "wb");
fwrite(&hf1, sizeof(BYTE), sizeof(BITMAPFILEHEADER), fp1); //처리한 파일헤더 쓰기
fwrite(&hInfo1, sizeof(BYTE), sizeof(BITMAPINFOHEADER), fp1); //처리한 인포헤더 쓰기
fwrite(hRGB1, sizeof(RGBQUAD), 256, fp1); //처리한 팔레트 쓰기
fwrite(Output1, sizeof(BYTE), ImgSize, fp1); //사이즈 할당(원본이미지 사이즈와 동일)
fwrite(&hf2, sizeof(BYTE), sizeof(BITMAPFILEHEADER), fp2); //처리한 파일헤더 쓰기
fwrite(&hInfo2, sizeof(BYTE), sizeof(BITMAPINFOHEADER), fp2); //처리한 인포헤더 쓰기
fwrite(hRGB2, sizeof(RGBQUAD), 256, fp2); //처리한 팔레트 쓰기
fwrite(Output2, sizeof(BYTE), ImgSize, fp2); //사이즈 할당(원본이미지 사이즈와 동일)
fclose(fp1);
fclose(fp2);
free(Image);
free(Output1);
free(Output2);
```

값을 입력 받아 원 영상의 밝기를 증가 및 감소하여 2개의 bmp결과 파일 생성하기