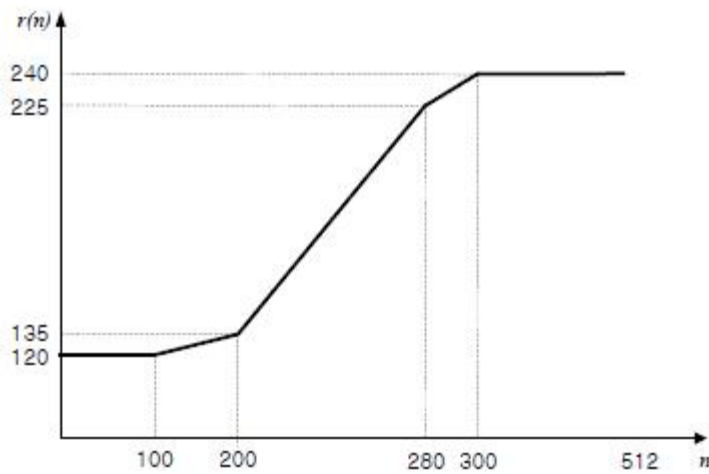


# <영상처리 과제 1> 컴퓨터정보통신공학부

## 1번 문제

1. Generate a 512x512 raw image, each row of which is a smooth ramp  $r(n)$  as shown in the figure below. Display the raw image on a PC monitor using Paint Shop Pro or other relevant image viewer such as Photoshop and designate the location of two Mach bands.



## 상처리과제1

(전역 범위)

```

1 // 2019253084 전유성 영상처리 과제 1-1번
2 #define _CRT_SECURE_NO_WARNINGS
3 #include <stdio.h>
4 #include <stdint.h>
5 #include <stdlib.h>
6 #include <windows.h>
7
8 //사진크기를 정의한다
9 // 512*512사진이기때문에 미리 정의해준다.
10 #define WIDTH 512
11 #define HEIGHT 512
12
13 //이미지에 ramp값을 채워주는 함수 선언
14 void Ramp(uint8_t** image);
15
16 //메인 함수
17 int main(void) {
18     //메모리에 이미지를 할당한다.
19     uint8_t** image = (uint8_t**)malloc(HEIGHT * sizeof(uint8_t*));
20     if (image == NULL) { //만약에 배열 "자체"를 할당 못 하는 경우
21         printf("배열이 메모리에 적재하지 못하여 프로그램이 실패하였습니다.\n");
22         return 1;
23     }
24     for (int i = 0; i < HEIGHT; i++) {
25         image[i] = (uint8_t*)malloc(WIDTH * sizeof(uint8_t));
26         if (image[i] == NULL) { //포인터가 할당된 메모리를 가르키지 못하는 경우
27             printf("포인터가 할당된 메모리를 가르키지 못하여 실패하였습니다.\n");
28             return 1;
29         }
30     }
31     //램프함수를 불러와서 이미지를 만든다.
32     Ramp(image);
33 }

```

## 처리과제1

(전역 범위)

```

34 // 만들어진 ramp파일을 저장한다.
35 FILE* file = fopen("1번문제 RAMP0이미지.raw", "wb");
36 if (file != NULL) {
37     for (int y = 0; y < HEIGHT; y++) {
38         fwrite(image[y], sizeof(uint8_t), WIDTH, file);
39     }
40     fclose(file);
41     printf("RAW 이미지가 성공적으로 만들어졌습니다.\n");
42 }
43 else {
44     printf("이미지 파일을 만들지 못 했습니다.\n 시스템을 종료합니다.\n");
45 }
46
47 // 할당된 메모리를 빼준다
48 for (int i = 0; i < HEIGHT; i++) {
49     free(image[i]);
50 }
51 free(image);
52
53 return 0;
54 }
55

```

```

//Ramp 함수의 내용
void Ramp(uint8_t** image) {
    // 문제 나온 그래프처럼 각 구간에 맞게 ramp값을 할당 해준다.
    for (int y = 0; y < HEIGHT; y++) {
        for (int x = 0; x < WIDTH; x++) {
            // x값 즉 n의 값이 100일 때, y값 r(n)의 값은 120으로 유지되어야한다.
            if (x <= 100) {
                image[y][x] = 120;
            }
            else if (x > 100 && x <= 200) { // x값 즉 n의 값이 100-200사이 일 때, y값 r(n)의 값은 135까지 증가되어야한다.
                image[y][x] = 120 + (uint8_t)((x - 100) * (135 - 120) / 100);
            }
            else if (x > 200 && x <= 280) { // x값 즉 n의 값이 200 ~ 280사이 일 때, y값 r(n)의 값은 225까지 증가되어야한다.
                image[y][x] = 135 + (uint8_t)((x - 200) * (225 - 135) / 80);
            }
            else if (x > 280 && x <= 300) { // x값 즉 n의 값이 280-300일 일 때, y값 r(n)의 값은 240까지 증가되어야한다.
                image[y][x] = 225 + (uint8_t)((x - 280) * (240 - 225) / 20);
            }
            else { // x값 즉 n의 값이 300-512사이 일 때, y값 r(n)의 값은 240으로 유지되어야한다.
                image[y][x] = 240;
            }
        }
    }
}

```

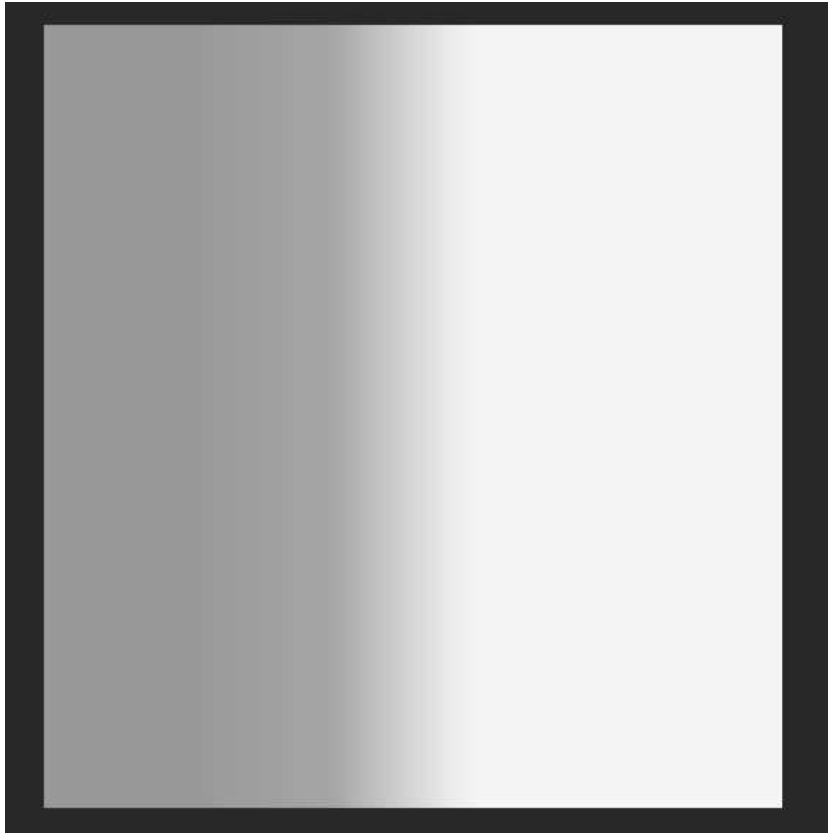
## 설명

512\*512 이미지를 가진 raw파일 만들기이다. 1번 문제에서 나온 그래프를 따라 값을 알맞게 대입한다 그러면 이미지가 선형으로 증가하지 않고 계단형으로 증가하는걸 알 수 있다. 그렇게 계단 형식의 포인터 배열을 완성한 뒤에 Raw파일 형식으로 저장하면 된다.

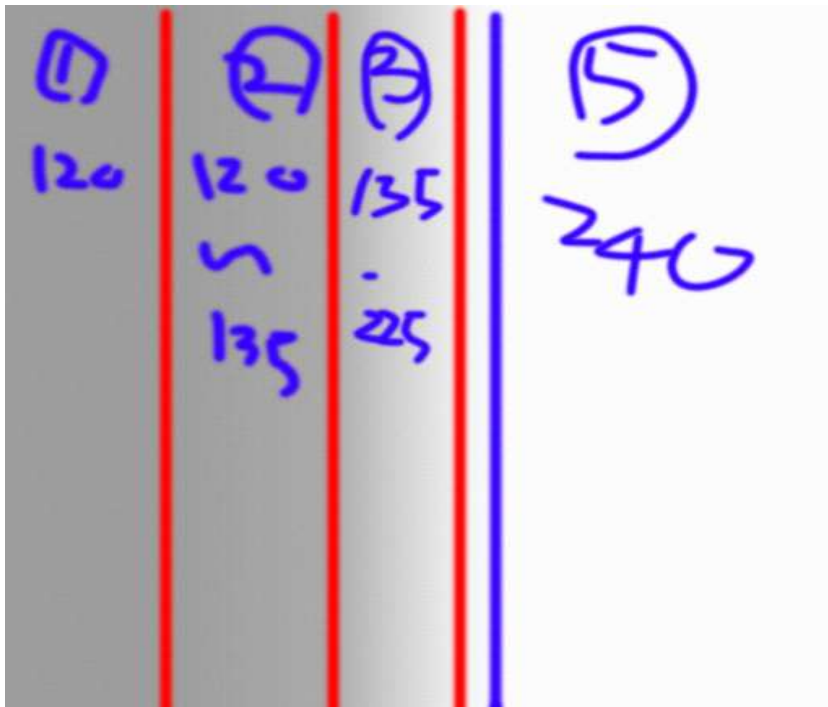
저장하면 왼쪽에서 오른쪽으로 갈수록 하얀색에 가까운 회색의 모습이 나오는 이미지 파일을 확인 할 수 있다. 컴퓨터 사양이나 환경에 따라 다르겠지만, 특정한 값의 경계선 부분을 보면 색이 더 진하다는 걸 알 수 있는데 이를 Mach band effect라고 한다.

raw형식은 이미지를 원본 그 자체로 저장하기 때문에 파일의 크기가 크고, 이러한 파일들은 일반적인 방법으로는 볼 수 없고, 어도비의 포토샵 같은 프로그램 툴을 통해 파일을 열 수 있다. 특히 선언한 ramp함수 부분에서 조건에 따라 색상을 출력하게 했는데 그레이스케일 스타일로 명암을 구분 할 수 있게 하였다. x축이 0 ~ 100까지이면 회색의 밝기는 120, x축이 101~200이면 밝기가 135까지 서서히 증가하도록, x축이 200~280사이이라면 밝기는 135에서 225까지 x축이 281~300이면 밝기는 225에서 240으로 서서히 증가하고 x축이 300~512까지면 밝기는 240으로 유지될 수 있게 하였다. 그러면 밑에 있는 사진처럼 결과가 출력된다.

1번 문제의 결과



1번문제의 그레이스케일 구분



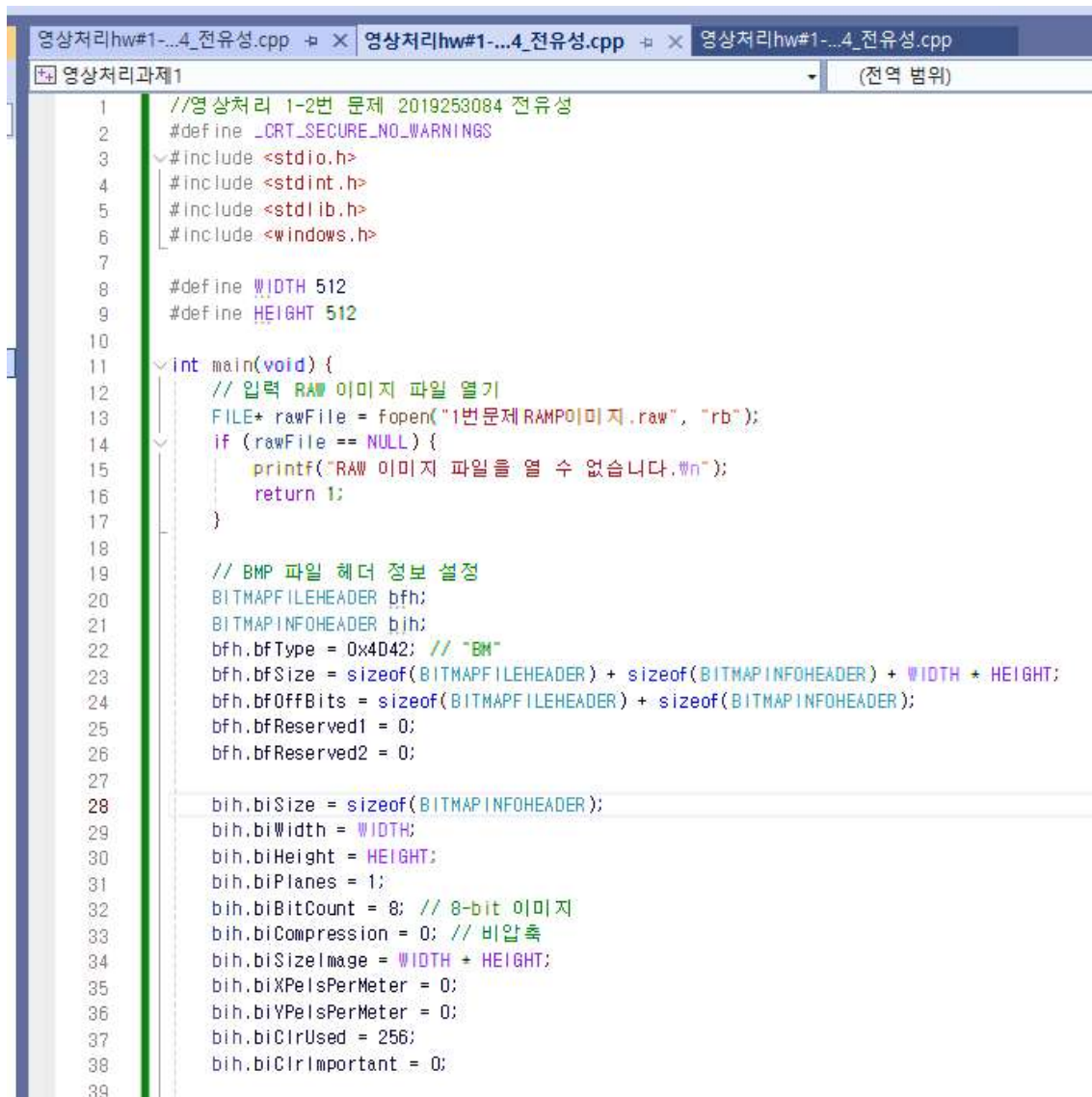
그레이스케일로만 보면 구분 힘들 수도 있으니 포토샵을 통해서 경계선 마다 임의로

색을 그어 구분하였다. 경계선의 기점으로 그레이스케일이 점점 밝아지는 것을 확인할 수 있다.

## 2번 문제

2. Using the raw image generated in the problem 1, generate a BMP (bitmap) image file which can be directly displayed on your PC monitor without executing Paint Shop Pro or Photoshop software.

1번 문제에서 생성한 이미지를 페인트 샵이나 포토샵 같은 소프트웨어 툴 없이 RAW파일에서 BMP파일로 변환 시키는 문제입니다.



```
1 //영상처리 1-2번 문제 2019253084 전유성
2 #define _CRT_SECURE_NO_WARNINGS
3 #include <stdio.h>
4 #include <stdint.h>
5 #include <stdlib.h>
6 #include <windows.h>
7
8 #define WIDTH 512
9 #define HEIGHT 512
10
11 int main(void) {
12     // 입력 RAW 이미지 파일 열기
13     FILE* rawFile = fopen("1번문제 RAW이미지.raw", "rb");
14     if (rawFile == NULL) {
15         printf("RAW 이미지 파일을 열 수 없습니다.\n");
16         return 1;
17     }
18
19     // BMP 파일 헤더 정보 설정
20     BITMAPFILEHEADER bfh;
21     BITMAPINFOHEADER bih;
22     bfh.bfType = 0x4D42; // "BM"
23     bfh.bfSize = sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER) + WIDTH * HEIGHT;
24     bfh.bfOffBits = sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER);
25     bfh.bfReserved1 = 0;
26     bfh.bfReserved2 = 0;
27
28     bih.biSize = sizeof(BITMAPINFOHEADER);
29     bih.biWidth = WIDTH;
30     bih.biHeight = HEIGHT;
31     bih.biPlanes = 1;
32     bih.biBitCount = 8; // 8-bit 이미지
33     bih.biCompression = 0; // 비압축
34     bih.biSizeImage = WIDTH * HEIGHT;
35     bih.biXPelsPerMeter = 0;
36     bih.biYPelsPerMeter = 0;
37     bih.biClrUsed = 256;
38     bih.biClrImportant = 0;
39 }
```



```
영상처리hw#1...4_전유성.cpp* X 영상처리hw#1...4_전유성.cpp 영상처리hw#1...4_전유성.cpp
영상처리과제1 (전역 범위)
38 BITMAPFILEHEADER bfh;
39
40 // BMP 파일 생성 및 헤더 정보 쓰기
41 FILE* bmpFile = fopen("1번문제 RAW이미지를 BMP로 변환한 2번문제 이미지.bmp", "wb");
42 if (bmpFile == NULL) {
43     printf("BMP 파일을 생성할 수 없습니다.\n");
44     fclose(rawFile);
45     return 1;
46 }
47 fwrite(&bfh, sizeof(BITMAPFILEHEADER), 1, bmpFile);
48 fwrite(&bih, sizeof(BITMAPINFOHEADER), 1, bmpFile);
49
50 // 팔레트 정보 설정 (그레이스케일)
51 RGBQUAD palette[256];
52 for (int i = 0; i < 256; i++) {
53     palette[i].rgbBlue = i;
54     palette[i].rgbGreen = i;
55     palette[i].rgbRed = i;
56     palette[i].rgbReserved = 0;
57 }
58 fwrite(palette, sizeof(RGBQUAD), 256, bmpFile);
59
60 // raw 이미지를 뒤집어서 bmp 파일로 출력하기
61 uint8_t* imageData = (uint8_t*)malloc(WIDTH * HEIGHT);
62 fread(imageData, sizeof(uint8_t), WIDTH * HEIGHT, rawFile);
63 for (int y = HEIGHT - 1; y >= 0; y--) {
64     fwrite(&imageData[y * WIDTH], sizeof(uint8_t), WIDTH, bmpFile);
65 }
66
67 // 파일 닫기 및 메모리 해제
68 fclose(rawFile);
69 fclose(bmpFile);
70 free(imageData);
71
72 printf("BMP 파일이 성공적으로 생성되었습니다.\n");
73
74 return 0;
75 }
```

## 설명

BMP파일은 압축하지 않은 상태의 이미지 파일로 압축하지 않아 상대적으로 파일의 크기가 큰 편이다.

RAW파일을 BMP파일로 변환하기 위해서는 먼저 BMP 파일의 헤더들을 추가해야 한다. BITMAPFILEHEADER, BITMAPINFOHEADER, RGBQUAD 헤더를 추가하였다. BITMAPFILEHEADER에서 bfh.bfType는 bmp 파일인지 구분하는 것이며, bfh.bfSize는 bmp 파일의 크기이며, bfh.bfOffBits는 BITMAPFILEHEADER의 처음부터 이미지 정보가 담긴 곳까지의 크기, 비트맵 데이터의 시작 위치를 알 수 있다. bfh.bfReserved1 예약된 위치1번으로 0이어야 하며, bfh.bfReserved2 예약된 위치2번으로 0이어야 한다.

BITMAPINFOHEADER에서 bih.biSize는 파일크기, bih.biWidth는 이미지의 가로

길이, bih.biHeight는 이미지의 세로길이, bih.biPlanes는 planes의 수 1로 설정, bih.biBitCount는 픽셀당 비트로 8비트 256색상을 사용, bih.biCompression는 압축 방식으로 0으로 비압축, bih.biSizeImage는 이미지의 크기로 512\*512, bih.biXPelsPerMeter는 미터 당 픽셀로 가로 해상도를 표현, bih.biYPelsPerMeter는 미터 당 픽셀로 세로 해상도를 표현, bih.biClrUsed는 비트맵에서 사용한 컬러 테이블의 컬러 인덱스 수, bih.biClrImportant는 비트맵에서 표현하는데 중요한 컬러의 인덱스의 수이며, 0으로 입력하면 모두 중요한 것으로 처리 할수 있다.

RGBQUAD에서는 rgbBlue는 파란색의 세기, rgbGreen는 초록색의 세기, rgbRed는 빨간색의 세기, rgbReserved은 예약된 위치로 0으로 설정하였다. 이렇게 헤더파일을 이용해서 BMP형식에 대한 것을 사전에 만들어 놓고 BMP파일을 만든다. BMP파일 만들 때에는 뒤집어 생성 해야한다.

1번문제RAMP이미지	2024-03-29 오후 9:18	RAW 파일	256KB
1번문제RAW이미지를BMP로변환2번문...	2024-03-30 오전 8:47	BMP 파일	258KB
2번문제이미지	2024-03-30 오후 10:40	BMP 파일	250KB

프로그램을 실행하면 정상적으로 만들어 진 것을 확인할수 있다.

2번 문제의 결과



2번문제 또한 BMP 파일을 통해 1번 문제의 RAW처럼 MACH BAND EFFECT를

느낄 수 있다.

### 3번 문제

3. Rotate the image obtained in number 1 90 degrees to the clockwise and store it properly in BMP format as you did in 2. Display the stored BMP image on your PC monitor.

앞선 문제들을 통해 만든 이미지를 시계 방향으로 90도 회전하여 BMP로 저장하는 문제이다. 문제에서는 1번에서 얻어낸 이미지라고 했지만 1번 문제는 RAW임으로 2번 문제에서 뒤집어서 출력된 BMP 이미지를 통해 작동하는 문제이다.



```
1 //영상처리 1-3번 문제 2019253084 전유성
2 #define _CRT_SECURE_NO_WARNINGS
3 #include <stdio.h>
4 #include <stdint.h>
5 #include <stdlib.h>
6 #include <windows.h>
7
8 #define WIDTHBYTES(bits)    (((bits) + 31) / 32 * 4 )
9 #define BYTE unsigned char
10
11 int main(void) {
12     FILE* infile;
13     infile = fopen("1번문제 RAW이미지를 BMP로 변환한 2번문제 이미지.bmp", "rb");
14
15     if (infile == NULL)
16     {
17         printf("영상파일이 없음 ");
18         return 1;
19     }
20
21     // BMP 파일 헤더 정보 입력
22     BITMAPFILEHEADER hf;
23     BITMAPINFOHEADER hInfo;
24
25     fread(&hf, sizeof(BITMAPFILEHEADER), 1, infile);
26     if (hf.bfType != 0x4D42) exit(1);
27
28     fread(&hInfo, sizeof(BITMAPINFOHEADER), 1, infile);
29     if (hInfo.biBitCount != 8)
30     {
31         printf("Bad File format!");
32         return 1;
33     }
34
35     // 팔레트 정보의 입력
36     RGBQUAD hRGB[256];
37     fread(hRGB, sizeof(RGBQUAD), 256, infile);
38
39 }
```



```

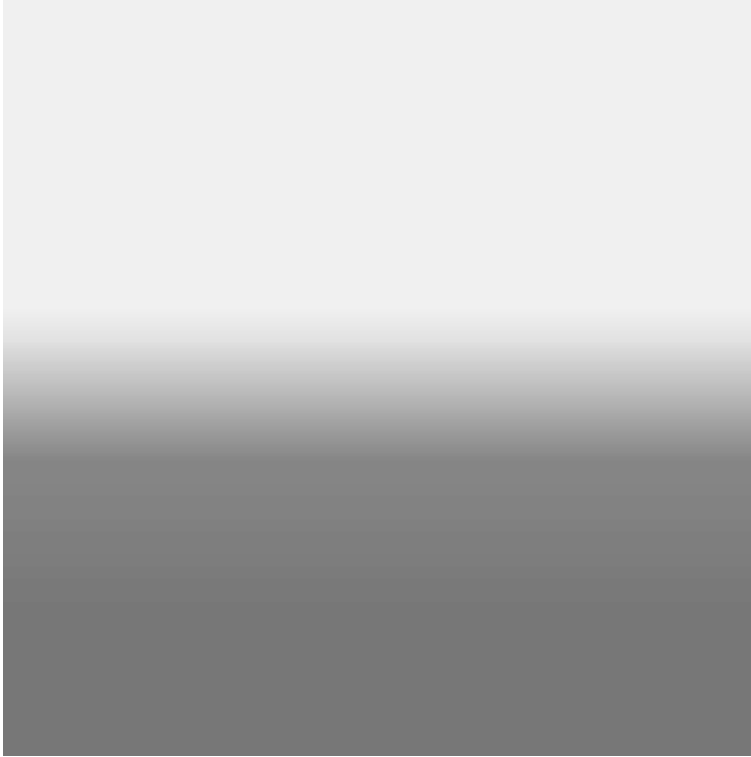
38 fread(hRGB, sizeof(RGBQUAD), 256, infile);
39
40 // 메모리 할당
41 BYTE* lpimg = new BYTE[hInfo.biSizeImage];
42 fread(lpimg, sizeof(char), hInfo.biSizeImage, infile);
43 fclose(infile);
44
45 int rwsz = WIDTHBYTES(hInfo.biBitCount * hInfo.biWidth);
46 int newWidth = hInfo.biHeight; // 새로운 이미지의 너비는 원본 이미지의 높이와 같습니다.
47 int newHeight = hInfo.biWidth; // 새로운 이미지의 높이는 원본 이미지의 너비와 같습니다.
48
49 // 새로운 이미지 배열을 할당
50 BYTE* rotatedImg = new BYTE[hInfo.biSizeImage];
51
52 // 이미지를 90도 시계방향으로 회전하고 뒤집습니다.
53 for (int i = 0; i < newHeight; i++)
54 {
55     for (int j = 0; j < newWidth; j++)
56     {
57         rotatedImg[i * rwsz + j] = lpimg[(newWidth - j - 1) * rwsz + i];
58     }
59 }
60
61 // 회전된 이미지를 새로운 BMP 파일로 저장
62 FILE* outfile = fopen("3번문제 이미지.bmp", "wb");
63 fwrite(&hInfo, sizeof(char), sizeof(BITMAPFILEHEADER), outfile);
64 fwrite(&hInfo, sizeof(char), sizeof(BITMAPINFOHEADER), outfile);
65 fwrite(hRGB, sizeof(RGBQUAD), 256, outfile);
66 fwrite(rotatedImg, sizeof(char), hInfo.biSizeImage, outfile);
67 fclose(outfile);
68 printf("3번문제 이 이미지가 정상적으로 출력되었습니다 \n");
69 // 메모리 해제
70 delete[] lpimg;
71 delete[] rotatedImg;
72
73 return 0;
74 }

```

## 설명

3번 문제는 2번 문제와 대부분 코드가 대체적으로 유사하지만 핵심은 52~59줄에 있는 FOR문이다. FOR를 통해 새로 생성된 이미지를 90도를 회전하고 뒤집으면 원래 이미지에서 90도 회전이랑 같다는걸 알 수 있다.

### 3번 문제 결과



### 과제 후기

1번 문제는 특별하게 큰 문제가 없었다. 코드를 통해 RAW파일이 제대로 생성되었는지, RAW파일을 포토샵을 통해 제대로 열려서 파일이 출력되는지를 확인할 수 있었다.

2번 문제는 처음에 제공 받은 Lena.raw파일을 통해 프로그램을 작성하는 줄 알았으나 문제를 다시 읽어 보니 1번에서 추출된 이미지로 만드는 것이었다. 1번 문제는 단순히 그레이스케일로 출력되기 때문에 2번 문제에서 뒤집어서 출력되어도 큰 차이가 없다는 걸 알 수 있다. 다만 raw파일을 bmp파일을 변환하는 과정에서, bmp파일에 대한 헤더 정보를 직접 정의해야 한다. 그리고 8비트 이미지로 생성 및 비압축으로 생성되기 때문에 파일 용량이 크다는 걸 알 수 있었다.

3번 문제는 2번 문제에서 출력한걸 회전하고 뒤집어서 출력만 하면 되기 때문에 큰 어려움이 없었다.

### 참고

<https://learn.microsoft.com/ko-kr/windows/win32/gdi/bitmap-header-types>

<MS비트맵형식>