분석 및 설계 보고서

압축된 BMP 이미지 파일에 대한 영상 처리

학 과	컴퓨터공학과
학 번	20120209
이 름	김민성
제작기간	약 2개월

1 문제 개요

8-bit-gray-scale 영상의 BMP 파일을 읽어 영상의 밝기를 반전하고 사용자가 지정한 크기의 사각형을 그린뒤에 새로운 이름으로 젖아한느 프로그램을 작성하시오.단, 입력으로 사용하는 bmp 파일은 run length encoding (RLE) 방식으로 압축(BI_RLE8)되어 있다.

2 제한 요소

- 2개 이상의 소스파일과 1개이상의 헤더파일을 사용한다.
- 압축하지 않은 형식과 RLE 압축한 형식을 모두 지원한다.
- 전역 변수를 사용할 수 없다.
- 입력으로 받는 영상은 1개씩 받아서 처리한다.
- 기능별로 분리(영상 반전, 사각형 그리기 등)하여 메뉴를 제공한다.
- 사용자는 가지고있는 파일에 대한 지식이 없다고 가정한다.즉, 입력 할 때, RLE로압축된파일인지, 그냥 BMP 파일인지 모르므로, 이에 대한 처리가 필요하다.

3 요구 분석

1. 기능 요구사항

가. BMP파일을 입력으로 받아서, 색상을 반전한다.
나. BMP파일을 입력으로 받아서 원하는 위치에 사각형을 그려서 출력한다.
다. RLE로 압축된 BMP파일을 입력으로 받아서 압축을 해제한다.
라. 프로그램의 종료를 원할 경우 종료한다.
마. 영상 처리를 완료한 파일을 사용자가 원하는 새로운이름으로 저장한다.

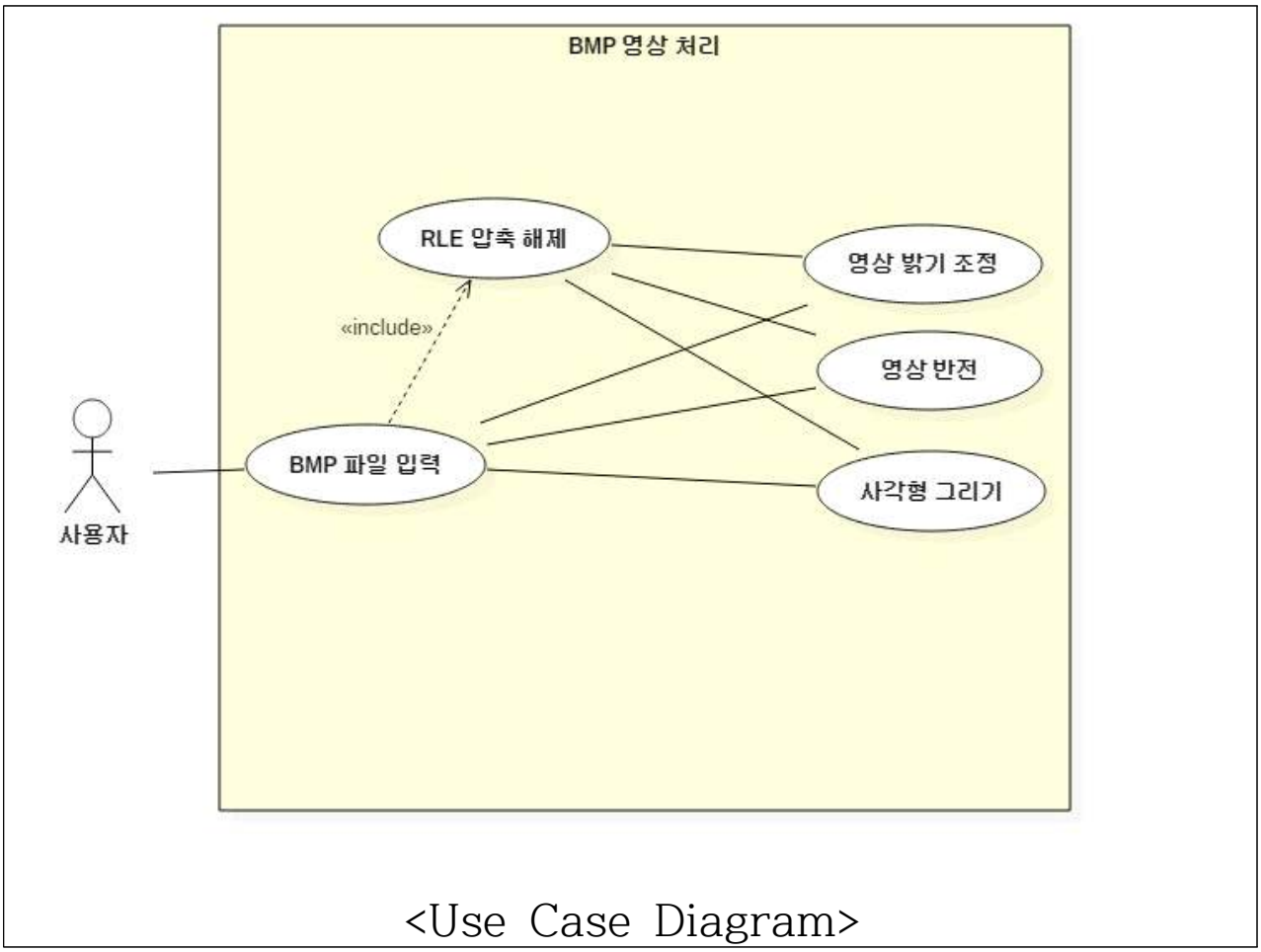
2. 비기능 요구사항

가. BMP(8bit-grayscale,24bit-truecolor)영상 외엔 처리 불가능
나. 한번에 하나의 영상만 처리할수 있다.
다. 처리되지 않은 입력시 오류 메시지를 출력후, 다시 입력받는다.
라. 영상 처리를 할 경우, 압축되지않은 BMP파일에 대해서만 처리한다.

4 유스케이스 명세화

■ 서비스 개요

이 시스템은 사용자가 입력한 파일을 사용자의 선택에 따라, 원하는 영상 처리를 수행하고, 사용자가 원하는 파일명으로 저장하는 프로그램이다.



<Use Case Diagram>

■ BMP파일입력 유스케이스

사용자가 입력하는 BMP파일을 다룬다. 사용자의 입력에 따라서, 영상 처리를 수행하며, 필요데 따라서, BMP파일이 압축된 경우,RLE로 압축된 BMP 파일을 압축을 해제 시킨뒤 영상 처리를 수행한다.

■ RLE압축해제 유스케이스

사용자가 입력한 BMP 파일이 RLE로 압축된 경우에 다룬다.RLE로 압축된 BMP파일을 영상 처리하면, 정상적으로 처리 되지않기 때문에,RLE로 압축된 파일을 압축 해제 시킨다.

■ 영상밝기조정 유스케이스

BMP 영상파일의 밝기를 조정한다. 예를들어,-30의 영상밝기를 주면, 30만큼 밝아지고, +30의 영상밝기를 주면, 30만큼 어두워진다.

■ 영상반전 유스케이스

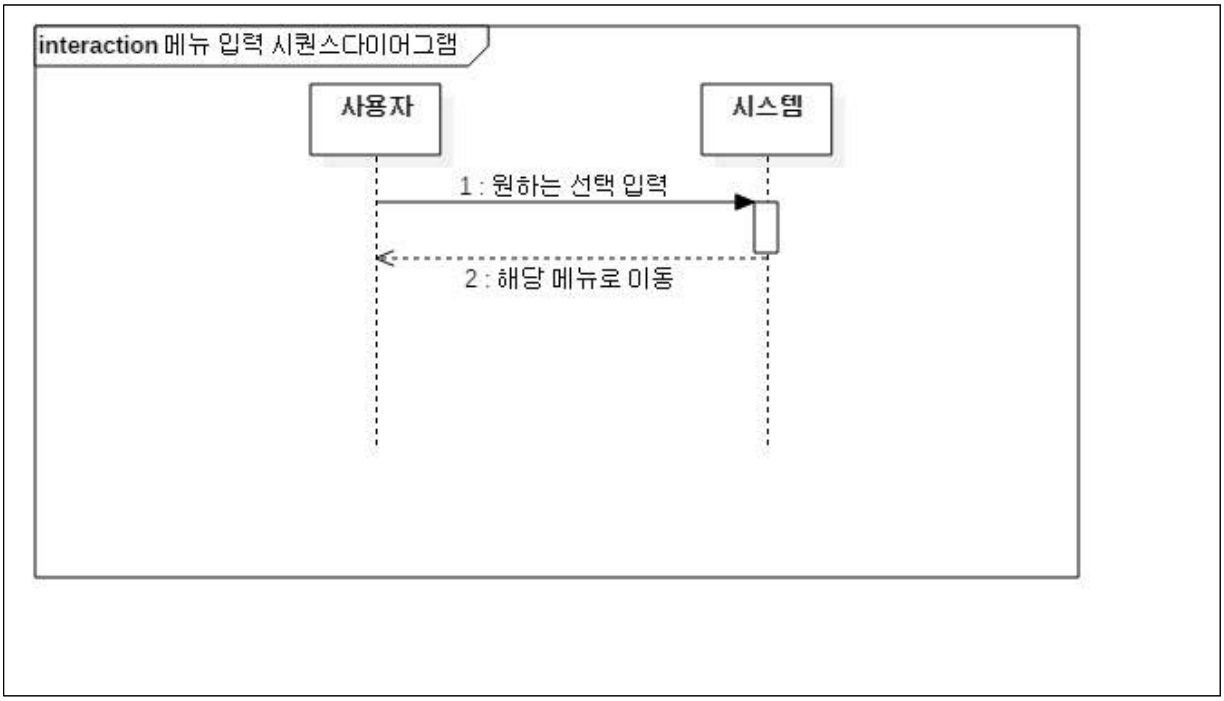
BMP 영상파일의 밝기를 반전시킨다.사용자가 입력한 BMP 파일의 밝기를 반전시킨뒤에, 입력받은 파일명으로 파일을 저장한다.

■ 사각형그리기 유스케이스

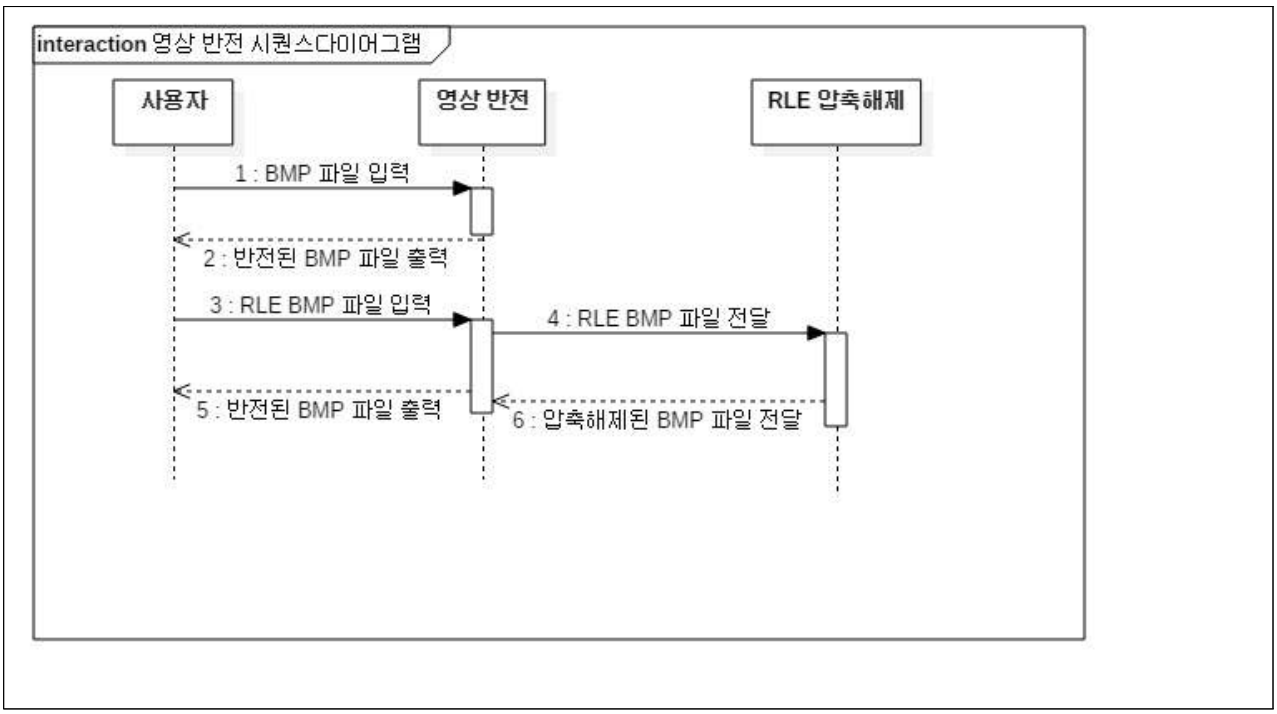
BMP 영상 파일을 입력받은뒤 사용자가 원하는 좌표에 사각형을그린다.사용자는 사각형이 그려질 시작좌표와 나중좌표를 입력한다.사각형을 그리고 난후, 입력받은 파일명으로 파일을 저장한다.

5 유스케이스 실체화

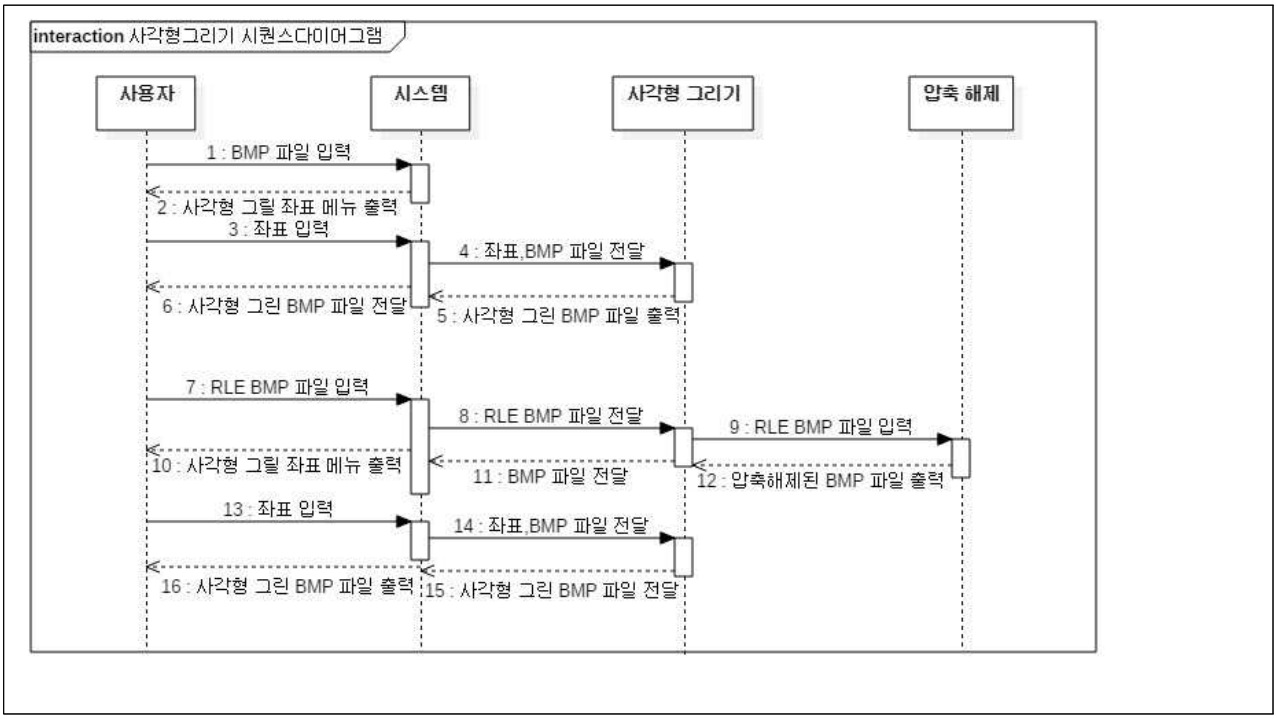
1. 메뉴 입력 시퀀스 다이어그램



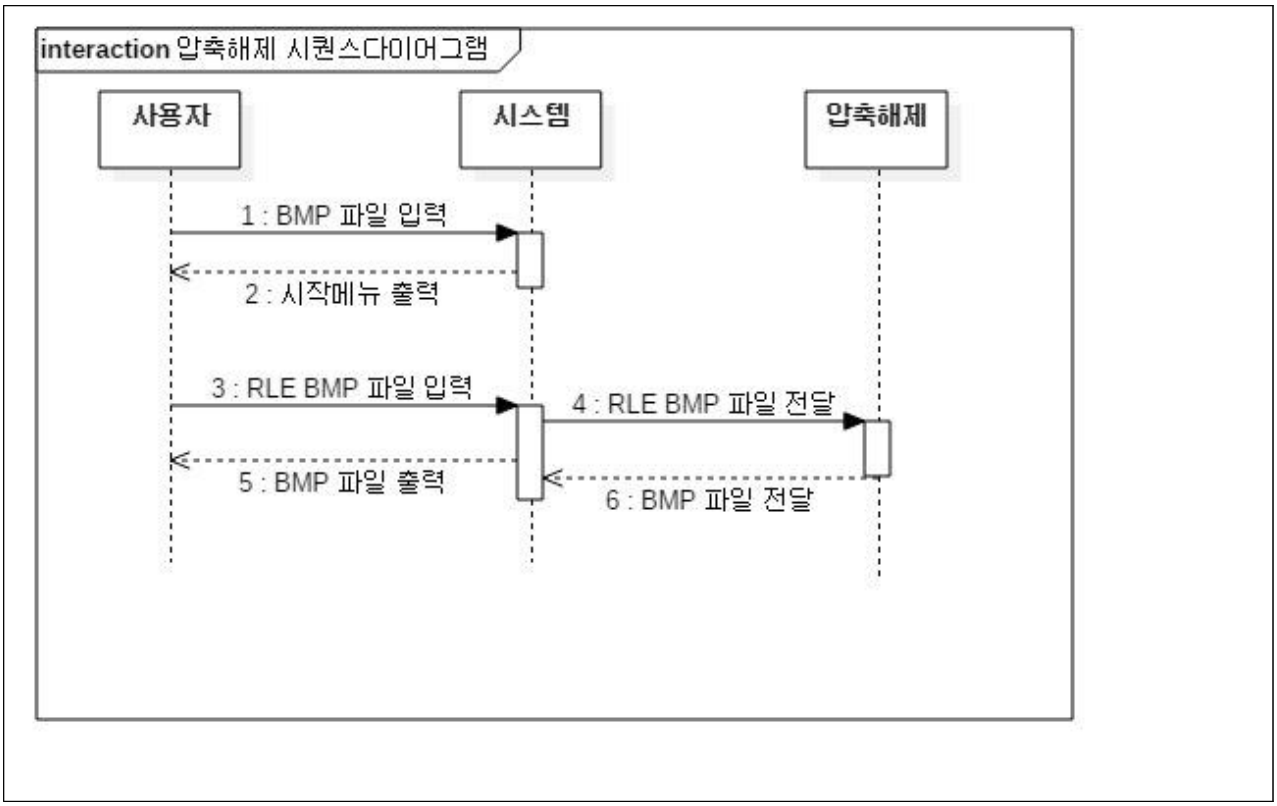
2. 영상 반전 시퀀스 다이어그램



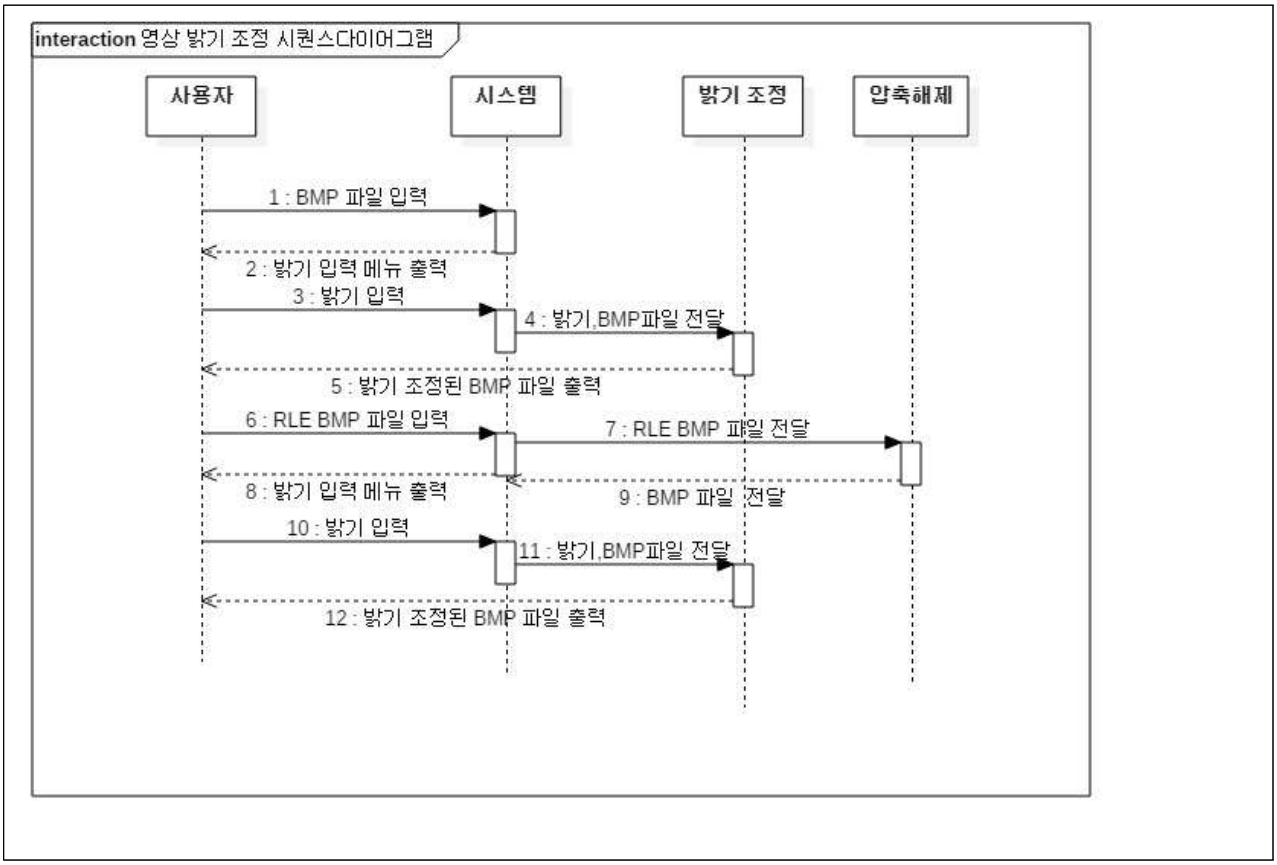
3. 사각형그리기 시퀀스 다이어그램



4.압축해제 시퀀스 다이어그램



5.영상 밝기 조정 시퀀스 다이어그램



6. 대체 흐름

○ 입력 오류

- ▸ 입력으로 받은 BMP파일의 이름이 존재하지않을 경우, 오류 메시지 출력후 다시 입력받도록 한다.
- ▸ 입력으로 받은 파일의 형식이 BMP 파일형식이 아닐 경우, 오류 메시지 출력후 시작 메뉴로 돌아가도록한다.
- ▸ 메뉴 선택 이외의 입력을 할시엔, 경고 메시지 출력후 다시 입력받는다.
- ▸ 영상 처리시 RLE로 압축된 경우 오류 메시지 출력후 압축해제 메뉴 메뉴로 이동한다.

6 입출력 설계(Text-based-UI)

1. 시작 메뉴

```
BMP 이미지 파일 변환 메뉴
-------------------------------------------------
1. BMP영상의 밝기 반전
2. 사각형 그리기
3. 영상 밝기 조정
4. RLE 압축파일 압축 해제
5. 끝내기

입력 >>
```

2. BMP영상의 밝기 반전 메뉴

```
1.BMP영상의 밝기 반전
-------------------------------------------------
영상의 밝기를 반전할 BMP 파일 입력 >>
8bit-grayscale 파일 입력
(BMP파일이 아니라면, 에러 메시지 출력 후 시작메뉴로 돌아감)
(RLE 압축파일 이라면 압축 해제 메뉴로 이동)
(입력된 BMP 영상 반전 시행)
입력파일의 크기 [ 가로 100px 세로 300px]
반전된 파일을 저장할 파일명 입력 >>
저장할 파일 이름 입력
(저장 후 시작 메뉴로 돌아감)
```

3. 사각형 그리기 메뉴

2.사각형 그리기

사각형을 그릴 BMP 파일 입력 >>
8bit-grayscale 파일 입력
(BMP파일이 아니라면, 에러 메시지 출력 후 시작메뉴로 돌아감)
(RLE 압축파일 이라면 압축 해제)
입력파일의 크기 [가로 100px 세로 300px]
사각형을 그릴 좌표를 입력[x1 y1 x2 y2] >>
사각형 그릴 좌표 입력
(입력된 BMP 영상 사각형을 그리기 시행)
사각형을 그린 파일을 저장할 파일명 입력 >>
저장할 파일 이름 입력
(저장 후 시작 메뉴로 돌아감)

4. 영상 밝기 조정 메뉴

3.영상 밝기 조정

밝기를 조정할 BMP 파일 입력 >>
8bit-grayscale 파일 입력
(BMP파일이 아니라면, 에러 메시지 출력 후 시작메뉴로 돌아감)
(RLE 압축파일 이라면 압축 해제)
원하는 밝기를 입력 하세요 [(어둠)-50 ~ +50(밝음)] >>
(입력된 값에따라 BMP 영상의 밝기 조정)
저장할 파일명 입력 >>
저장할 파일 이름 입력
(저장 후 시작 메뉴로 돌아감)

5. RLE 압축파일 압축 해제 메뉴

4.RLE 압축파일 압축 해제

압축해제 할 BMP 파일 입력 >>

RLE로압축된 8bit-grayscale 파일 입력

(압축된 파일이아니라면 에러메세지 출력후 시작메뉴로 돌아감)

입력파일의 크기 [가로 100px 세로 300px]

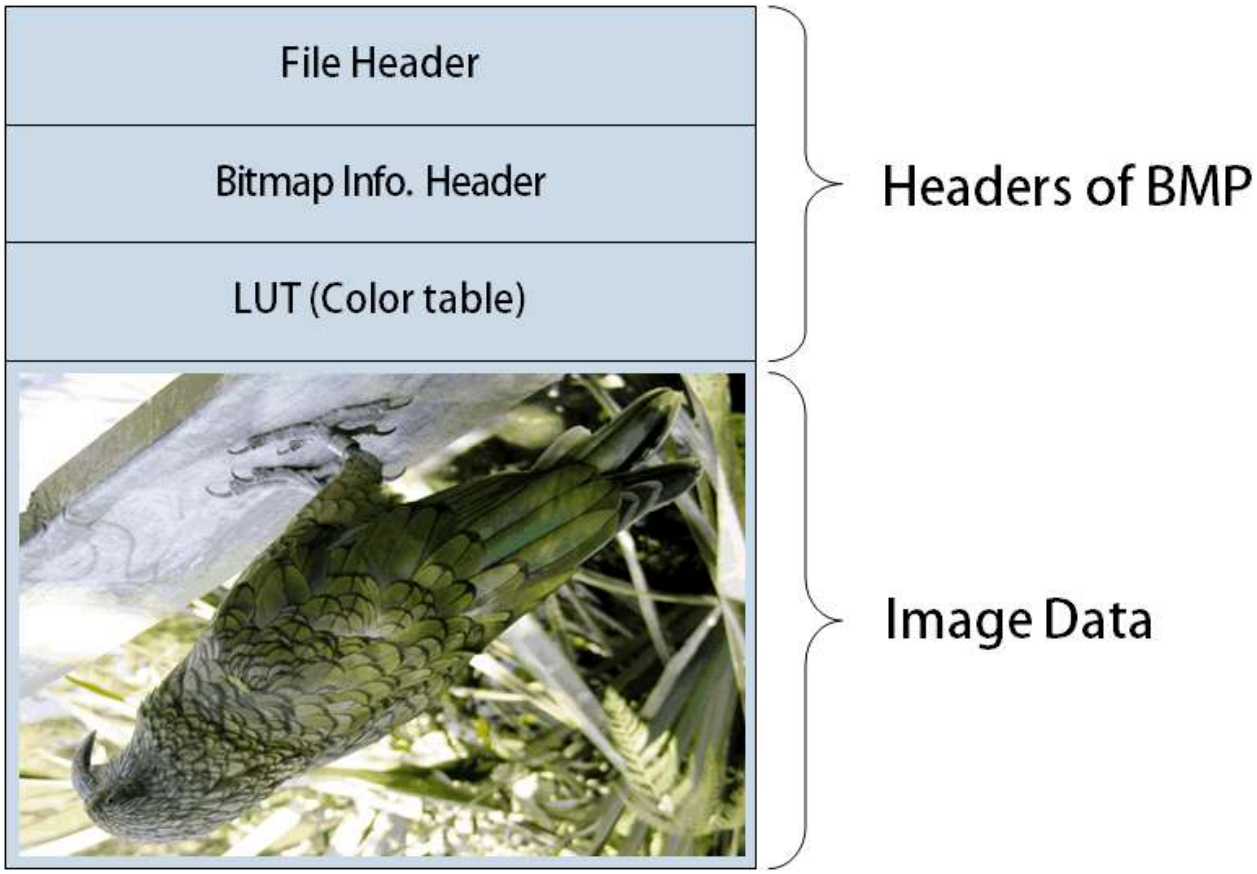
(압축 해제 시행)

압축해제 한 BMP파일을 저장할 파일명 입력 >>

저장할 파일 이름 입력

(저장 후 시작 메뉴로 돌아감)

7 데이터 구조 분석



1. BMP 파일 구조 분석

가. 파일 헤더 구성

```
/* 자료형 재선언 */
typedef unsigned char BYTE;
typedef unsigned short WORD;
typedef unsigned long DWORD;

/* 파일 헤더의 멤버 */
WORD bfType; // BMP file type: "BM" (4D42)
DWORD bfSize; // 전체 파일의 크기 (byte)
WORD bfReserved1; // reserved (항상 0)
WORD bfReserved2; // reserved (항상 0)
DWORD bfOffBits; // 이미지 데이터의 시작 오프셋
```

나. 비트맵 정보 헤더 구성

```
/* 비트맵 정보 헤더의 멤버 */
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, BI_RLE4, BI_RLE8)
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


/* 칼라 테이블 */
BYTE rgbBlue; // 파란색 성분 (B component)
BYTE rgbGreen; // 녹색 성분 (G component)
BYTE rgbRed; // 빨간색 성분 (R component)
BYTE rgbReserved1; // 예약 (reserved)
```

다. BMP 파일 특징

1. 영상 데이터의 각 행은 4Byte로 구성된다.예를 들어, 가로 150px 세로 100px로 구성되는 영상이면 8bit-gray-scale의 경우, 한행의 바이트수는 150byte 이지만, 4의배수가 아니므로, 152byte가 된다.
4Byte의 행을 구하기위한 식은 다음과 같다.

((BMP파일의 픽셀 당 비트수+31)/32*4)

2. BMP 파일은 영상 데이터를 상하 반전되어 파일에 저장한다.따라서, (x,y)좌표에 위치하는 픽셀에 접근하기위해서는 다음의 수식을 사용한다.

$$(x, y) \Rightarrow [widthStep^{*}(height\text{-}y\text{-}1) + x]$$

2. RLE로 압축된 BMP 파일 구조 분석

RLE알고리즘은 반복되어 나타나는 데이터가 많은 경우에, 압축효율이 좋은 알고리즘이다.따라서,연속적으로 반복되는 데이터가 많이 나타나는 이미지에 적합하다. RLE로 압축된 BMP파일의 경우엔 비트맵 정보 헤더의 biCompression가 BI_RLE8 값을 가진다.BMP의 RLE는 두가지 모드로에서 압축을 수행하며, 두 가지 모드 모두 2개의 바이트를 처리 단위로 사용하여 첫 번째 바이트는 발생 횟수, 두번째 바이트는 값을 나타낸다.

Encoded Mode 의 경우엔,

첫 번째 바이트는 발생 횟수, 두 번째 바이트는 값을 의미한다.따라서, 05 0A 의

경우엔 복호화 하면 0A 0A 0A 0A 0A 로 복원된다.

Absolute Mode 의 경우엔,

첫 바이트가 0이고, 두 번째 바이트가 3이상이면(<256) 두 번째 바이트의 값은 데이터 열의 길이이다. 따라서, 00 04 00 01 02 03 04 의 경우, 복호화를 진행하면 00 01 02 03 이 된다.첫 바이트가 0이고, 두 번째 바이트가 0,1,2 중의 한 가지 값이면 마커(Marker)에 해당한다.

■End Of Line (00 00)

이 마커는 다음 코드는 새로운 라인을 의미하며, 현재 라인에는 더 이상의 정보가 없음을 나타낸다.만약 영상의 가로길이보다 픽셀의 개수가 적으면 마지막 값으로 저장한다.

■End Of Bitmap (00 01)

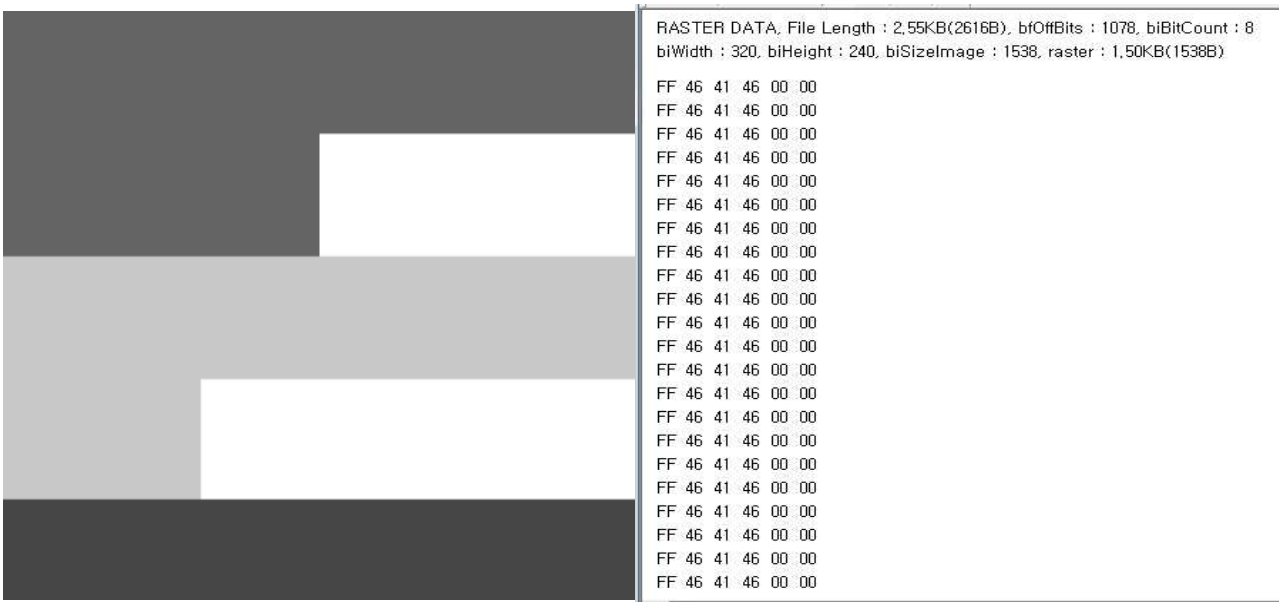
이 마커는 부호화된 데이터의 마지막을 나타낸다. 나머지 영역은 마지막 값으로 저장한다.

■Delta (00 02)

이 마커는 현재 위치로부터 상대적으로 이동할 위치를 나타낸다.즉, 현재 위치에서 새로운 위치 사이에있는 데이터(픽셀)은 부호화를 생략한다.Delta 마커 이후에 등장하는 첫 번째 바이트는 수평방향의 오프셋, 두 번째 바이트는 수직방향의 오프셋을 의미한다. 단, 이 값들은 부호 없는 정수로 처리한다. 오프셋의 다음에 나타나는 코드는 새로운 위치의 픽셀을 위한 것임을 알 수 있다.생략한 픽셀들은 현재 위치의 값으로 채운다.

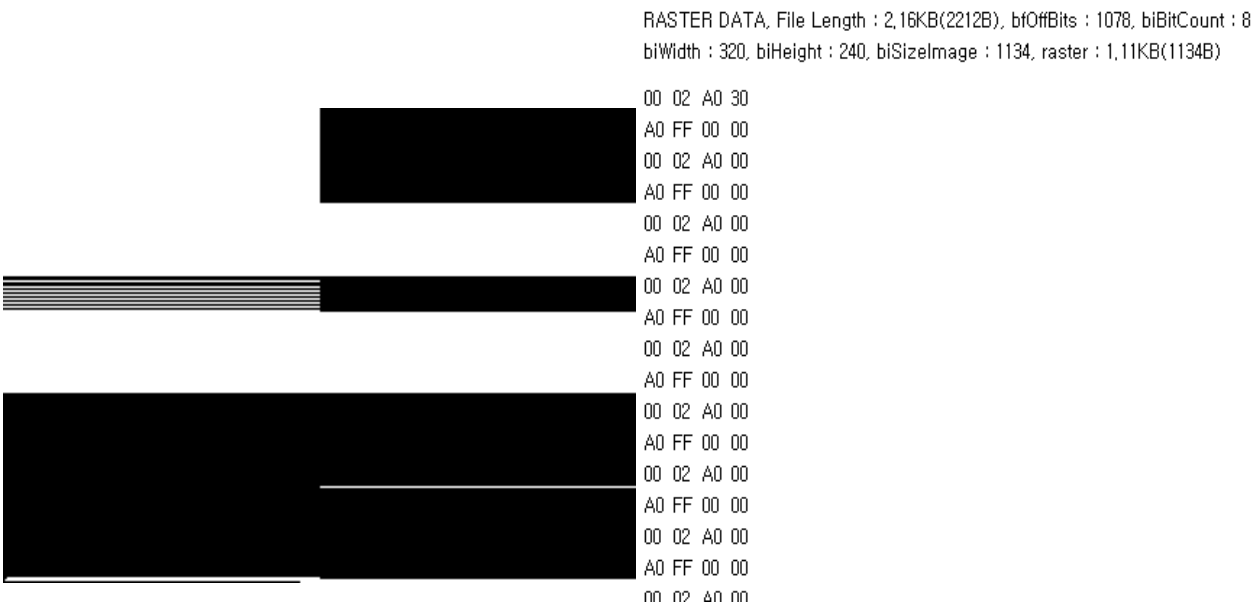
3.다양한 RLE 파일 분석

가. End Of Line Marker를 포함하는 RLE BMP 파일



■해당 파일은 EOL Marker 만 존재하는 파일로, 부호화 했을 경우, 딱 영상 데이터의 가로 * 세로 길이만큼 부호화되므로, EOB Marker는 존재 하지 않는다.

나.Delta Marker를 포함하는 RLE BMP 파일



■Delta Marker가 존재하는 파일로,바로 다음 나오는 xOffset ,yOffset만큼에 대해선, 부호화를 생략하고, 검은색으로 채워넣는다.또한, BMP파일은 영상데이터가 상하 반전 되있으므로, 이또한 고려해서 파일을 분석해야한다.예를들어, (A0 30) 만큼 검은색으로 채워진부분이 아래에서부터 해다 오프셋만큼 검은색으로 채워진 것이다.

다.어떤 영상데이터도 들어있지않고, EndOfBitmap Marker만 들어있는 경우,

94	94	94	00	95	95	95	00	96	96	96	00	97	97	97	00	98	98	98
00	99	99	99	00	9A	9A	9A	00	9B	9B	9B	00	9C	9C	9C	00	9D	9D
9D	00	9E	9E	9E	00	9F	9F	9F	00	A0	A0	A0	00	A1	A1	A1	00	A2
A2	A2	00	A3	A3	A3	00	A4	A4	A4	00	A5	A5	A5	00	A6	A6	A6	00
A7	A7	A7	00	A8	A8	A8	00	A9	A9	A9	00	AA	AA	AA	00	AB	AB	AB
00	AC	AC	AC	00	AD	AD	AD	00	AE	AE	AE	00	AF	AF	AF	00	B0	B0
B0	00	B1	B1	B1	00	B2	B2	B2	00	B3	B3	B3	00	B4	B4	B4	00	B5
B5	B5	00	B6	B6	B6	00	B7	B7	B7	00	B8	B8	B8	00	B9	B9	B9	00
BA	BA	BA	00	BB	BB	BB	00	BC	BC	BC	00	BD	BD	BD	00	BE	BE	BE
00	BF	BF	BF	00	C0	C0	C0	00	C1	C1	C1	00	C2	C2	C2	00	C3	C3
C3	00	C4	C4	C4	00	C5	C5	C5	00	C6	C6	C6	00	C7	C7	C7	00	C8
C8	C8	00	C9	C9	C9	00	CA	CA	CA	00	CB	CB	CB	00	CC	CC	CC	00
CD	CD	CD	00	CE	CE	CE	00	CF	CF	CF	00	D0	D0	D0	00	D1	D1	D1
00	D2	D2	D2	00	D3	D3	D3	00	D4	D4	D4	00	D5	D5	D5	00	D6	D6
D6	00	D7	D7	D7	00	D8	D8	D8	00	D9	D9	D9	00	DA	DA	DA	00	DB
DB	DB	00	DC	DC	DC	00	DD	DD	DD	00	DE	DE	DE	00	DF	DF	DF	00
E0	E0	E0	00	E1	E1	E1	00	E2	E2	E2	00	E3	E3	E3	00	E4	E4	E4
00	E5	E5	E5	00	E6	E6	E6	00	E7	E7	E7	00	E8	E8	E8	00	E9	E9
E9	00	EA	EA	EA	00	EB	EB	EB	00	EC	EC	EC	00	ED	ED	ED	00	EE
EE	EE	00	EF	EF	EF	00	F0	F0	F0	00	F1	F1	F1	00	F2	F2	F2	00
F3	F3	F3	00	F4	F4	F4	00	F5	F5	F5	00	F6	F6	F6	00	F7	F7	F7
00	F8	F8	F8	00	F9	F9	F9	00	FA	FA	FA	00	FB	FB	FB	00	FC	FC
FC	00	FD	FD	FD	00	FE	FE	FE	00	FF	FF	FF	00	00	01			

■이 경우엔, 팔레트 배열 마지막 다음에, 어떤 영상데이터도 존재하지않고, 바로 00 01(EndOfBitmap)이 나온경우이다. 따라서, 해당 나머지 영역은 정의 된 것처럼 모두 검은색으로 채워 넣어야한다.

라.RLE로 압축된 BMP 파일이 원래의 크기보다 큰 경우,

RASTER DATA, File Length : 260,96KB(267224B), bfOffBits : 1078, biBitCount : 8
biWidth : 512, biHeight : 512, biSizeImage : 266146, raster : 259,91KB(266146B)

00 FF 0C 0B 0E 0D 0E 0C 0D 0E 10 0E 0C 0D 0E 0E 18 29 49 7B 8C 8D 8D
00 00 FF 0C 0C 0E 0D 0E 0D 0E 0D 10 0D 0C 0E 0E 0E 18 29 4A 7C 8C 8D
00 00 00 FF 0B 0C 0D 0C 0F 0C 0E 0D 11 0E 0C 0E 0F 11 1A 2B 4F 85 88
27 00 00 00 FF 0C 0C 0D 0D 0E 0E 0D 0D 0F 0F 0F 0E 0E 13 1C 32 57 87
02 25 00 00 00 FF 0F 0E 0C 0C 0E 0E 0D 0F 0F 11 0E 0F 0F 11 22 33 5A
00 01 23 01 21 00 00 00 FF 0D 0D 10 0E 0F 0F 0E 0F 11 0D 0E 10 12 14
1A 23 21 20 23 00 00 00 00 AA 0D 0C 0F 0E 0D 0F 0E 0E 0F 10 0F 0F 10
17 19 1C 1B 21 22 00 00 00 00 FF 0D 0D 0E 0E 0E 0F 0E 0F 0F 13 0F 10
0F 11 16 16 16 00 01 1A 01 1E 00 00 00 FF 0D 0D 0F 0F 10 0D 0D 0E 0F
0E 0F 0D 0D 0D 11 11 15 00 01 19 01 20 00 00 00 FF 0D 0C 0F 12 11 0E
10 10 0F 0E 0F 12 11 10 0F 12 16 1B 00 00 00 FF 0E 0D 0E 0F 0F 0E 0E
0F 10 11 0E 0F 0F 0E 0F 0E 10 00 01 14 01 15 00 00 00 FF 0E 0D 0F 11
10 10 11 11 10 12 10 0F 0F 10 10 0E 0F 00 01 10 01 0F 00 00 00 FF 0E
12 15 11 11 0F 11 11 10 0F 0E 0E 0E 0F 0E 0C 0D 00 01 0E 01 10 00 00
20 1B 19 15 15 11 10 0F 10 0F 13 0F 10 0E 0F 0E 10 0E 0F 00 02 0D 00
25 23 20 1B 16 15 13 11 12 12 11 11 0F 12 11 10 0F 0E 0E 10 00 02 0E
27 28 24 23 1B 1A 15 17 15 13 11 10 10 11 12 13 12 10 10 0F 0F 10 0D
2A 25 22 1F 1B 19 18 17 13 13 11 11 11 12 11 12 12 0F 0F 0E 11 0F 00
2A 27 20 1D 19 17 14 15 14 12 12 12 10 11 13 12 0E 0F 0E 0E 0C 00 00
26 21 1E 1B 17 17 16 11 14 13 12 10 11 11 12 10 0F 0F 0D 0D 00 00 00
26 25 23 27 1F 1B 17 16 14 13 13 12 11 12 13 14 13 14 10 10 0D 0D 00

■RLE로 압축했는데, 영상데이터가 오히려 더 커진경우이다.이것은 RLE 압축 알고리즘이 그리 효율이 좋지않다는 것을 뜻한다.반복적인 데이터가 많이나오는 경우엔 ,효율이 좋지만 그렇지 않은경우엔, 오히려 데이터의 크기가 더 커진다.

따라서, RLE로 압축된 데이터를 복호화를 진행할때는, 가로*세로 길이가 아닌, 해당 데이터의 크기로 복호화를 진행하여야한다.

8 데이터 구조 설계

1. 함수 기능 설계

PrintBMPImageChoiceMenu(choice)

//BMP 이미지 파일 변환 메뉴 출력

BMPImageConversion()

//BMP 파일의 영상 밝기를 반전

BMPImageDrawSquare()

//BMP 파일에 사각형을 그리기

BMPImageAdjustLight()

//BMP 파일 밝기 조정

RLEBMPImageDeCompression()

//RLE로 압축된 BMP파일 압축해제

2. 기능 별 알고리즘 설계

가. 영상 반전 알고리즘(슈도 코드)

```
function BMPImageConversion
        int i , j;
        pImg <- 영상 데이터 입력
        for(i=0 ; i<영상데이터.세로길이 ; i++)
        {
            for( j=0 ; j<영상데이터.가로길이 ; j++)
            {
                pImg[i*rwsize + j] = 255 – pImg[i*가로길이 + j];
                //영상 데이터 반전 처리
            }
        }
end function
```

나. 사각형 그리기 알고리즘(슈도 코드)

```
function BMPImageDrawSquare
        int i , j , x1 , y1 , x2 , y2
        pImg <- 영상 데이터 입력
        [x1 y1 x2 y2] <- 사각형 그릴 좌표 입력
        for(i=x1 ; i < x1 + (x2 – x1) ; i++)
        {
              for( j=y1 ; j < y1 + (y2 – y1) ; j++)
              {
                    pImg[i*가로길이 + j] = 원하는 색깔
                    //지정된 좌표 영역에 사각형 그림(색깔지정 가능)
              }
        }
end function
```

다. RLE로 압축된파일 압축해제 알고리즘(슈도 코드)

```
function RLEBMPImageDeCompression
        int i , j;
        pImgOut 출력 데이터 변수(복호화된 데이터 저장할 변수)
        pImg <- 영상 데이터 입력
        While(부호화된 데이터의 마지막에 도달할 때 까지)
                    firstbyte <- 첫 번째 바이트 읽기
                    secondbyte <- 두 번째 바이트 읽기
                    if(firstbyte == 0) //Absolute Mode
                    {
                          switch(secondbyte)
                          {
                          case 0: //End Of Line(00 00)
```

```
            if 가로길이에 다 도달하지못했다면, then
                pImgOut <- 검은색으로 채워 넣음
            end if
            다음 라인으로 이동
            break
        case 1: //End Of Bitmap(00 01)
            if 아직 채우지 못한 영역이 남았다면 then
                pImgOut <- 검은색으로 채워 넣음
            end if
            break
        case 2: //Delta(00 02)
            xOffset <- 수평 방향 오프셋 읽기
            yOffset <- 수직 방향 오프셋 읽기
            while(상대적인 Offset 위치에 도달할 때 까지)
                pImgOut <- 검은색으로 채워 넣음
            end while
            break
        default:
            length = secondbyte
            for(length 만큼 반복)
                pImgOut <- pImg //현재 위치의 값을 채워 넣음
            end for
            if 길이가 홀수라면 then
                해당 인덱스는 건너뜀(부호화 생략)
            end if
    else //Encoded Mode
    {
        length = firstbyte
        value = secondbyte
        for(length 만큼 반복)
            pImgOut <- value
        end for
    }
```

```
        end While
end function
```

라. 영상 밝기 조절 알고리즘(슈도 코드)

```
function BMPImageAdjustLight
        int i , j;
        pImg <- 영상 데이터 입력
        Input <- 사용자의 입력값(-50 ~ +50)
        for(i=0 ; i<영상데이터.세로길이 ; i++)
        {
            for( j=0 ; j<영상데이터.가로길이 ; j++)
            {
                pImg[i*가로길이 + j] += Input;
                //사용자 입력값이 최대 밝기나 최소밝기를 넘어선다면,
                if(pImg[i*가로길이 + j] > 255)
                    pImg[i*가로길이 + j] = 255;
                if(pImg[i*가로길이 + j] < 0)
                    pImg[i*가로길이 + j] = 0;
            }
        }
end function
```

9 GUI 구현 설계

입출력 화면 설계

■아직 구현에 들어가지 않아, 많이 변경될 여지가 있지만, 기본 화면을 간단하게 설계해 보았습니다.구현은 현재 설계한 알고리즘을 그대로 사용하여,MFC를 사용하여 구현 할 생각입니다.각 버튼에 기능을 연결하여, 구현할것이고, 이후에 변경될 수 있는 부분은 후에, 추가적으로 제출하겠습니다.그리고, 히스토그램 평활화 기능은, 멀티미디어공학에서 배우고나면, 추가적으로 넣어볼 생각입니다.