프로젝트 최종 보고서

압축된 BMP이미지 파일의 변환

작성일	2022.11.13	제출일	2022.11.20
과 목	어드벤처디자인 3분반	전 공	컴퓨터공학
담당교수	김경수	조 명	1조
조 장	송제용	조 원	김현진
조 원	안현진	조 원	정지수

목 차

1. 서론 및 문제정의

본 보고서는 중간레벨 프로젝트 최종 보고서임을 명시한다.

1-1. 서론

이번 프로젝트의 목적으로써는 rle방식으로 압축된 8-bit gray-scale 영상에 대해서 압축 해제하고[(1)]그 결과물을 이용하여 사각형 그리기, 색상 반전을 적용한 뒤[(2)]이 결과를 저장할 수 있는[(3)] 프로그램을 만드는 것이다. 이러한 목표를 위하여 rle 방식의 압축이 어떤 식으로 이루어지는지[(1)], 영상의 밝기 반전은 어떻게 실행할 수 있는지[(2)], 사용자가 지정한 크기만큼 사각형을 그리려면 어떻게 해야 하는지[(3)]에 대해서 구체적으로 조사해보고 생각할 필요가 있었다.

1-2. 문제 정의

이번 프로젝트의 문제는 서론에서 언급했듯 “RLE 방식으로 압축된 8bit gray-scale 영상을 저장하고 있는 BMP파일에 대해서 압축을 해제, 그에 대한 색상 반전, 사각형 그리기를 수행한 후에 그 결과를 저장하라”는 것이다. 이에 대한 설명은 계획 및 설계보고서에 기술하였으니 생략 하도록 하고 문제에서 주어진 기본 요구사항을 만족시키기 위해 우리가 해결해야 했던 각 요구사항의 중요 포인트에 대해 간략히 짚고 넘어가도록 하겠다.

먼저, RLE 방식으로 압축된 8비트 그레이 스케일 파일에 대해 압축해제를 수행을 위해 앱솔루트 모드와 인코디드 모드로 나누어 프로그램을 설계하였고, 특히 앱솔루트 모드에서는 두 번째 인덱스가 3이상일 경우 앱솔루트 모드로, 3 이하일 경우에는 마커로 인식을 하여 처리하도록 구현하였다.

문제에서 요구된 영상처리 부분은 크게 2가지로, 색상반전 및 사각형 그리기를 수행하는 것이다.
색상반전 부분은 255라는 8비트에서 가장 큰 컬러값을 빼서 색을 반전시키도록 설계하고
사각형 그리기 부분은 사용자가 입력한 사각형의 좌표값을 토대로 데이터의 인덱스 값에 GRAY값이 입력되도록 설계 및 구현하도록 하였다.

2. 자료구조 설계

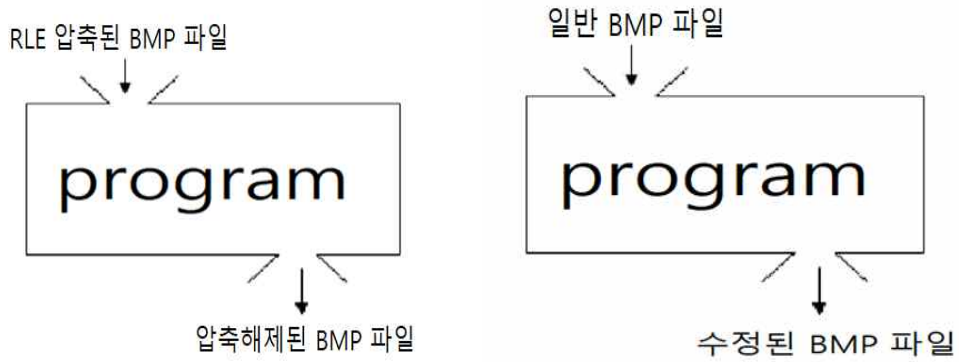
2-1. 입력, 출력 데이터 및 내부 흐름 데이터

입력데이터

-RLE 방식으로 압축된 BMP파일 또는 압축되지 않은 BMP파일

출력데이터

-압축 해제한 BMP 파일 , 사각형 그리기, 밝기 반전하기, 밝기 조절하기, 좌우 반전하기, 상하반전하기, 임계화하기, 레벨 분할하기의 기능을 이용한 유저의 수정을 거친 BMP파일을 새로운 이름으로 저장

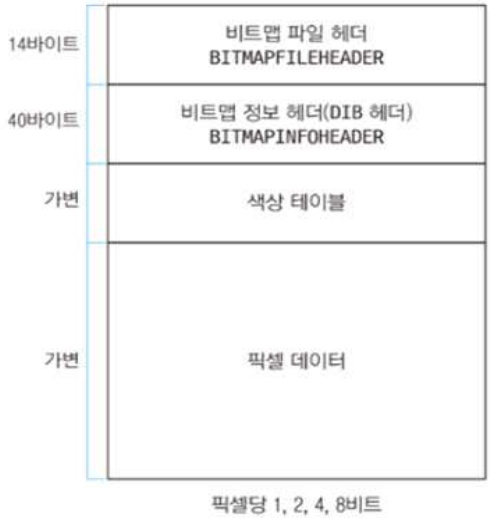


내부 흐름 데이터(위 사진 중 program 부분)

-RLE8로 압축된 8bit gray-scale BMP일 경우 압축해제하기.
-8bit gray-scale BMP 영상을 사용자가 지정한 크기의 사각형을 그리기
-8bit gray-scale BMP 영상을 밝기반전시키기
-8bit gray-scale BMP 영상을 밝기조절하기
-8bit gray-scale BMP 영상을 좌우반전하기
-8bit gray-scale BMP 영상을 상하반전하기
-8bit gray-scale BMP 영상을 임계화하기
-8bit gray-scale BMP 영상을 레벨분할하기

2-2. 구현을 위해 필요한 주요 데이터

bmp파일 내부데이터



자세한 내부데이터 분석은 계획 및 설계보고서에 기술하였음.

아래는 bmp내부데이터 활용을 위해 실제 구현한 소스코드 사진이다. 이해를 돕기 위해 첨부하였다.

```
class ImageProcess {
public:
    FILE* file;
    BITMAPFILEHEADER fileheader;
    BITMAPINFOHEADER infoheader;
    RGBQUAD* RGB;
```

```
fread(&fileheader, sizeof(BITMAPFILEHEADER), 1, file);

fread(&infoheader, sizeof(BITMAPINFOHEADER), 1, file);
```

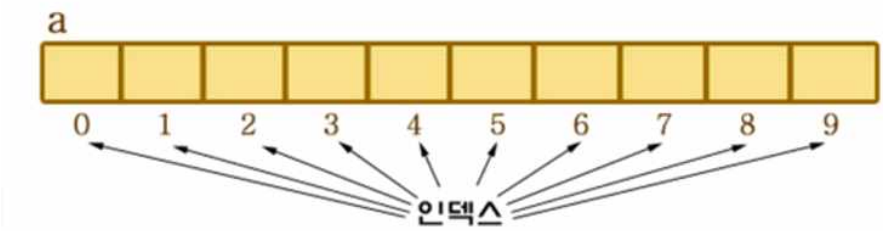
bmp파일 내부데이터에서 주로 사용할 변수들

BITMAPFILEHEADER	BITMAPINFOHEADER

- WORD bfType; // Specifies the file type
- DWORD bfSize; // 파일의 크기 (byte)
	- DWORD biSize; // 구조체의 크기 (bytes)
	- LONG biWidth; // 비트맵의 가로 길이 (pixels)
	- LONG biHeight; // 비트맵의 세로 길이 (pixels)
	- WORD biBitCount; // 픽셀당 비트수 (1,4,8,16,24,32)
	- DWORD biCompression; // 압축 유형
	- DWORD biSizeImage; // 비트맵 데이터의 크기 (bytes)
	- DWORD biClrUsed; // 칼라 인덱스의 개수

2-3. 사용할 주요 자료구조 설계

위에서 본 것과 같이 입력, 출력 데이터 및 내부 흐름 데이터 구현을 위해서는 bmp 파일 구조를 이용하여 bmp 내부 데이터들을 들고 와야 한다. bmp 내부 데이터를 효율적으로 들고 오기 우리는 필요한 자료구조로 배열을 선정했다.



배열은 연속된 메모리 공간에 순차적으로 저장된 데이터 모음이다. 우리가 배열을 선정한 이유는 첫 번째로는 구현하기 쉽다는 이유이다. 우리는 비교적 방대한 데이터인 bmp 내부 데이터를 다루기 때문에 코드 구현에 있어 오류가 날 가능성이 많다. 그래서 최대한 오류가 적게 발생하는 배열을 이용하여 구현하고자 하였다. 두 번째로는 우리는 위에서 기술한 대로 bmp 파일 구조를 이용하여 bmp 데이터들을 들고 오므로 필요한 데이터의 크기가 정해져 있다. 그래서 데이터의 추가적인 삽입이 필요가 없어 데이터의 크기를 바꿀 일이 없다. 그래서 다른 자료구조를 이용하지 않아도 되고 배열이 효율적이라고 판단했다. 세 번째로는 실제 메모리 상에서 물리적으로 데이터가 순차적으로 저장되기 때문에 데이터에 순서가 있으며, index가 존재하여 indexing 및 slicing이 가능하기 때문이다.

```
RGB = new RGBQUAD[256];
fread(RGB, sizeof(RGBQUAD), 256, file);

ReadFile = new BYTE[infoheader.biSizeImage];
fread(ReadFile, sizeof(BYTE), infoheader.biSizeImage, file);
```

위 사진을 우리가 실제 구현한 소스코드 중 일부이다. 이를 이용하여 추가적인 설명을 돕겠다. 사진에서 보는 것과 같이 bmp 파일구조를 이용해서 내부 데이터를 읽어온다. 만약 내부 데이터에 순서가 정해져있어 순차적으로밖에 읽어올 수밖에 없다고 가정해 보자. 그렇다면 사각형 그리기와 같이 전체 데이터가 필요하지 않고 사용자가 지정한 영역의 데이터만 필요한 기능들을 구현하기 굉장히 비효율적일 것이다. 그래서 index를 사용해 특정 요소를 리스트로부터 효율적으로 읽어올 수 있는 indexing 기술과 요소에 특정한 부분을 따로 분리해 조작하는 slicing 기술이 가능한 배열을 이용한다면 효율적으로 구현할 수 있다고 생각했다. 또한 속도에 있어서도 배열은 기본위치 + 오프셋(요소 크기 * 인덱스) 연산으로 모든 요소에 접근 가능하기 때문에 배열의 각 요소에 접근하는 시간은 O(1)로 모두 동일하다. 그래서 빠르게 필요한 bmp 데이터를 읽어와 기능을 수행할 수 있기 때문에 자료구조로 배열을 선정하였다.

3. 프로그램 설계

3-1. 모듈 사용 목적, 기능 및 제약 조건 명세

설계 보고서에서 작성한 모듈 사용 목적, 기능 및 제약 조건 명세를 기술하였지만 실제 구현을 하면서 추가로 구현한 기능이 있어 그 기능을 추가로 기술하겠다. 또한 아래 기능들을 설계 보고서에 기술한 기능들과 다르게 추가적인 기능 설명이 필요한 기능들이라 추가로 기능 설명을 기술하겠음.

1.영상 임계화하기(추가기능)

실행 및 제약조건:

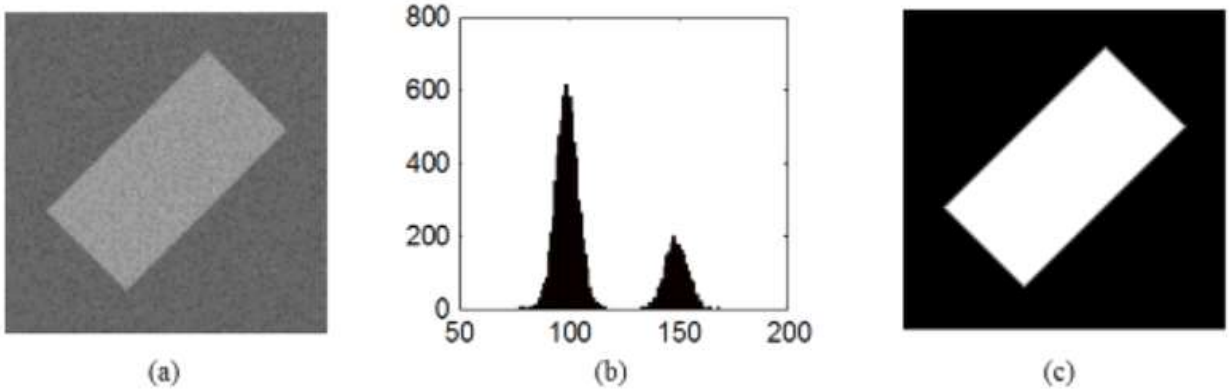
사용자에게 임계값을 입력받아야함. 사용자가 입력하는 임계값의 범위는 0~255이다.

사용 목적과 기능:

영상을 사용자가 지정한 임계값을 기준으로 임계화하여 사용자에게 임계화된 영상을 제공한다.

-임계화에대한 설명

영상처리에서 임계화는 어떤 주어진 임계값(threshold)보다 밝은 픽셀들은 모두 흰색으로, 그렇지 않은 픽셀들은 모두 검은색으로 바꾸는 것을 지칭한다. 아래 사진은 이해를 도와줄 예시이다.



(a)는 원본 영상, (b)는 픽셀들의 밝기값 히스토그램, (c)는 이진화 결과영상이다. 히스토그램에서 볼 수 있듯이 배경영역은 약 100 정도의 밝기값을, 물체는 150 근처의 밝기값 분포를 갖는다. 따라서 임계값을 125 정도로 주고 이진화를 했을 때 (c)처럼 임계화가 수행된다.

2.영상 레벨분할하기(추가기능)

실행 및 제약조건:

사용자에게 레벨분할을 하기위한 밝기 범위를 입력받아야함. 사용자가 입력하는 밝기 범위는 0~255이다.

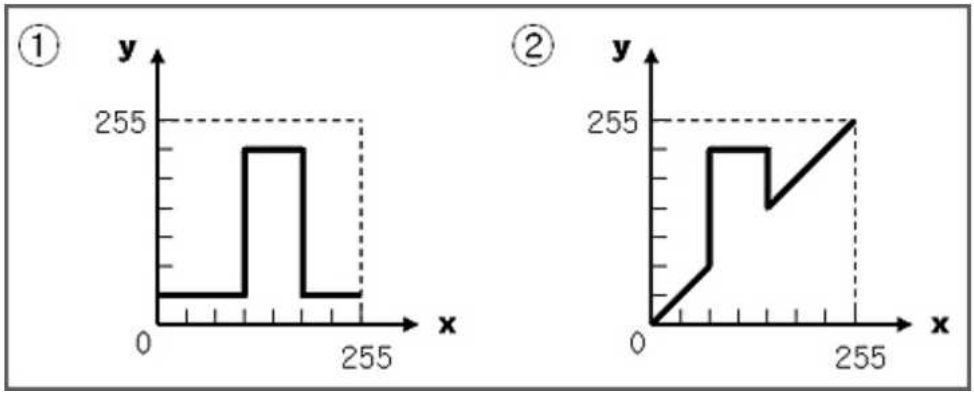
또한 입력한 밝기범위를 기준으로 해당 객체를 강조하기 위해 바꿀 밝기 값을 입력받아야함. 사용자가 입력하는 밝기 값의 범위는 0~255이다.

사용 목적과 기능:

영상을 사용자가 지정한 레벨분할기능을 수행할 밝기 범위를 입력받은 뒤 밝기 값을 입력받고 해당 밝기 값을 기준으로 레벨분할하여 사용자에게 레벨분할된 영상을 제공한다.

-레벨분할에대한 설명

영상 레벨분할은 입력영상에서 관심 영역이나 객체를 강조하거나 분리하기 위한 목적으로 사용한다. 아래는 이해를 도와줄 예시이다.



위 사진에서 보는 것처럼 레벨분할에는 두 가지 종류가 있다.
첫 번째로는 원하는 영역을 높은 값으로 줘서 밝게 만들어주고, 나머지영역은 어둡게 처리한 방법.
두 번째로는 원하는 영역을 높은 값으로 줘서 밝게 만들어주고, 나머지 영역은 원래 그대로 보존하는 방법이다.
우리가 택한 방법은 후자의 방법이다. 하지만 우리가 구현한 코드의 다른 점은 원하는 영역을 높은 값으로 주도록 강제하지않고 원하는 밝기값을 줄 수 있도록 하였다.

3-2. 모듈의 입출력 및 실행 예시 명세

설계보고서에서 작성한 모듈의 입출력 및 실행예시 명세가 구현하면서 바뀐 부분이 있으나 최종보고서의 프로그램 실행예시의 내용을 참고하면 확인 가능하므로 생략하겠음. 설계보고서에 기술하지 못한 기능만 구현하겠음.

1. 영상 임계화하기(추가기능)

INPUT: 압축이 되어있지 않은 8비트 비트맵 영상, 임계값
OUTPUT: 사용자가 지정한 임계값으로 임계화 된 8비트 비트맵 영상

'임계화 기능 수행'
'임계값을 입력해주세요 최소값:0 최대값:255 >>'
INPUT:임계값
#만약 사용자가 입력한 임계값이 범위를 벗어났다면 다시입력받음.
'임계화 기능을 수행하겠습니다'
OUTPUT: 사용자가 지정한 임계값으로 임계화 된 8비트 비트맵 영상

2. 영상 레벨분할하기(추가기능)

INPUT: 압축이 되어있지 않은 8비트 비트맵 영상, 밝기범위, 밝기 값
OUTPUT: 사용자가 지정한 밝기범위와 밝기값으로 레벨분할 된 8비트 비트맵 영상

'레벨분할 기능 수행'
'객체의 밝기 범위를 입력해주세요 최소값:0 최대값:255>>
INPUT:밝기 범위
#만약 사용자가 입력한 밝기범위가 지정 범위를 벗어났다면 다시입력받음.
'객체를 강조하기 위한 밝기 값을 입력해주세요 최소값:0 최대값:255
INPUT:밝기 값
#만약 사용자가 입력한 밝기 값이 지정 범위를 벗어났다면 다시입력받음.
OUTPUT: 사용자가 지정한 밝기범위와 밝기값으로 레벨분할 된 8비트 비트맵 영상

3-3. 모듈의 알고리즘(suedo code)

모듈의 알고리즘은 이미 설계보고서에 기술하였음. 최종보고에서는 설계보고서에 기술하지 못한 추가로 구현한 기능만 기술하겠음.

1. 영상 임계화하기(추가기능)

INPUT: 압축이 되어있지 않은 8비트 비트맵 영상, 임계값
OUTPUT: 사용자가 지정한 임계값으로 임계화 된 8비트 비트맵 영상

```
1. int I ← 영상의 세로길이를 담을 변수  j ← 영상의 가로길이를 담을 변수
2. bitmap_data ← input값으로 받은 비트맵 영상의 영상데이터를 담은 변수
3. user_bit_map ← 임계화 된 비트맵 영상을 담을 변수
4. bit_map_index ← i * (width) +j
5. Thresholding_control ← 사용자가 지정한 임계값
6. for i←0 to i<영상의 세로길이 {
7.      for j←0 to j<영상의 가로길이 {
8.           if(bitmap_data [bit_map_index] <= Thresholding_control )
9.                user_bit_map[bit_map_index] ← 0
10.          elseif(bitmap_data [bit_map_index] > Thresholding_control )
11.               user_bit_map[bit_map_index] ← 255 }
```

알고리즘 핵심 내용 설명

8. if(bitmap_data [bit_map_index] <= Thresholding_control)
9. user_bit_map[bit_map_index] ← 0
10. elseif(bitmap_data [bit_map_index] > Thresholding_control)
11. user_bit_map[bit_map_index] ← 255

=> 사용자에게 입력받은 임계값을 기준으로 그 임계값보다 작다면 검정색(0) 크다면 흰색(255)로 픽셀 데이터의 밝기 값을 바꿔준다.

2. 영상 레벨분할하기(추가기능)

INPUT: 압축이 되어있지 않은 8비트 비트맵 영상, 밝기범위, 밝기 값
OUTPUT: 사용자가 지정한 밝기범위와 밝기값으로 레벨분할 된 8비트 비트맵 영상

```
1. int I ← 영상의 세로길이를 담을 변수  j ← 영상의 가로길이를 담을 변수
2. bitmap_data ← input값으로 받은 비트맵 영상의 영상데이터를 담은 변수
3. user_bit_map ← 레벨분할 된 비트맵 영상을 담을 변수
4. bit_map_index ← i * (width) +j
5. obj_start, obj_end ← 사용자가 강조하고 싶은 객체의 밝기범위
6. Slicing_control ← 사용자가 지정한 밝기값
7. for i←0 to i<영상의 세로길이 {
8.     for j←0 to j<영상의 가로길이 {
8.          if( obj_start <= bitmap_data [bit_map_index] <= obj_end )
9.               user_bit_map[bit_map_index] ← Slicing_control
10.         else
11.              user_bit_map[bit_map_index] ← bitmap_data [bit_map_index] }
```

알고리즘 핵심 내용 설명

```
8.          if( obj_start <= bitmap_data [bit_map_index] <= obj_end )
9.               user_bit_map[bit_map_index] ← Slicing_control
10.         else
11.              user_bit_map[bit_map_index] ← bitmap_data [bit_map_index] }
```

=> 사용자에게 입력받은 밝기 범위와 밝기 값으로 픽셀 데이터의 밝기 값이 범위에 속해있으면 사용자가 입력한 밝기 값으로 바꾸어주고 만약 속해 있지 않다면 원래 픽셀 데이터의 밝기를 유지해 준다.

4. 프로그램 코드

4-1. 소스코드

소스 코드가 길어 보고서에 담기에는 무리가 있다고 판단하여 해당 소스코드는 압축된 파일에 주석을 달아 첨부하였다.

4-2. 구현 내용 분석

프로그램코드에 주석을 달아 프로그램 코드를 이해하는 것에는 무리가 없다고 생각하나 추가로 설명해주면 좋을 것같은 부분들이 있어 이 부분에 대해 추가적인 설명을 기술하겠음.

1. 비트맵영상은 영상의 가로의길이가 4의배수가 아니여도 4의 배수 바이트로 바꾸어 저장을 한다. 이를 해결하기위해 아래사진과 같은 공식을 활용하였다. 아래 사진은 실제 소스코드 사진이다.

```
#define NEW_WIDTH(bits) (((bits)+31)/32*4)
int new_width = NEW_WIDTH(infoheader.biBitCount * infoheader.biWidth);
```

해당 공식을 활용하여서 4의배수가 아닌 숫자를 4의배수로 바꾸어준다.

2. 압축해제된 크기를 직접 입력해주어야하는데 만약 가로의 길이가 4의 배수가 아닌 영상을 입력받는다면 아래사진처럼 입력받으면 안된다.

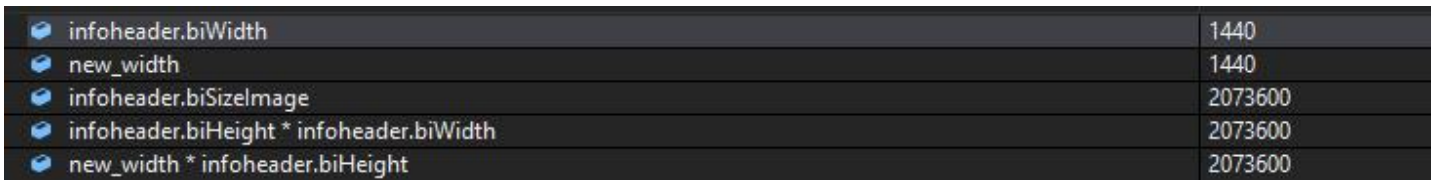
```
infoheader.biSizeImage = infoheader.biWidth * infoheader.biHeight;  //압축해제된 크기
```

왜냐하면 아래 사진에서 보는 것과 같이 biSizeimage의 값은 만약 biWidth값이 4의 배수가 아닐 경우 4의배수로 바꾼 값에 즉 new_width값에 biHeight를 곱한 것에 2를 더한 값과 같다.

infoheader.biWidth	570
new_width	572
infoheader.biSizeImage	240242
infoheader.biHeight * infoheader.biWidth	239400
new_width * infoheader.biHeight	240240

참고로 위에 사진에서 사용한 영상의 크기는 아래 사진과 같다. 이해를 돕기위해 첨부하였다. 보는 것과같이 4의 배수가 아닌 것을 알 수 있다.

반대로 영상의 가로 길이가 4의 배수인 영상이라면 픽셀 데이터의 크기는 new_width 값에 biHeight를 곱한 것과 같다 아래는 이해를 도와줄 사진이다.



위 사진에서 보는 것과 같이 new_width 값에 biHeight를 곱한 값과 biSizeimage의 크기는 같다. 참고로 아래 사진은 위에 사진에서 사용한 영상의 크기이다. 보는 것과 같이 가로의 길이가 4의 배수임을 알 수 있다.

그래서 압축을 푸는 기능을 수행한 뒤 bisizeimage변수 값을 아래 사진과 같이 가로의 길이가 4의 배수일 때와 4의 배수가 아닐 때 다르게 설정해 주어야 한다.

```
if (infoheader.biWidth % 4 == 0)
{
    infoheader.biSizeImage = new_width * infoheader.biHeight;    //압축해제된 크기
}
else
{
    infoheader.biSizeImage = new_width * infoheader.biHeight + 2;    //압축해제된 크기
}
```

3. 사용자가 사각형 그리기를 먼저 수행하고 밝기를 반전하거나 조절하거나 임계화기능을 수행하는 것과 같이 영상의 밝기값을 조절하는 기능을 수행한다면 아래사진처럼 사용자가 그린 사각형의 밝기또한 조절해버리는 문제가 발생한다.

이를 해결하기 위해 아래사진과 같이 bool값으로 Sqare_index값을 메소드 변수로 추가하였다.

```
class ImageProcess {
public:

    bool* Sqare_index;
```

Sqare_index 변수를 이용하여 사용자가 사각형 그리기 기능을 수행할 때 아래 사진과 같이 Sqare_index 또한 1을 저장해 준다.

```
for (int Sqare_height = user_select_y_start; Sqare_height < user_select_y_end; Sqare_height++)
{
    for (int Sqare_width = user_select_x_start; Sqare_width < user_select_x_end; Sqare_width++)
    {
        bit_map_index = Sqare_height * new_width + Sqare_width;
        ReadFile[bit_map_index] = 200;
        Sqare_index[bit_map_index] = 1;
    }
}
```

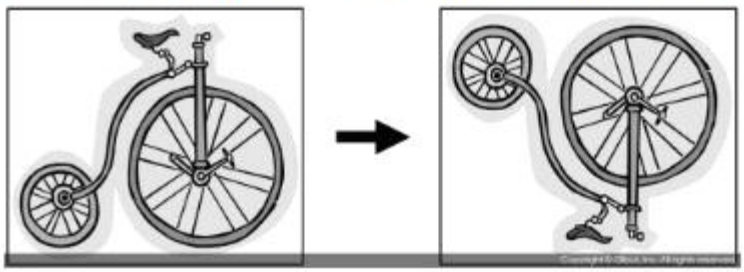
이렇게 저장한 Sqare_index 값을 이용하여 밝기 조절 기능을 수행할 때는 아래 사진과 같이 Sqare_index 값이 1이 아닐 때에만 수행해 준다.

```
if (Sqare_index[bit_map_index] != 1)
```

이러한 코드를 이용하여 아래 사진과 같이 왼쪽 위에 사각형이 그려진 후 밝기 조절 기능을 수행하여도 사각형의 색깔이 유지하도록 하였다.

4. 비트맵은 아래 사진에서 보는 것처럼 상하가 뒤집힌 방식으로 저장하기 때문에 사각형을 그릴 때 이 특징을 고려해야 한다.

▼ 그림 3-2 비트맵에서 픽셀 데이터를 상하가 뒤집힌 방식으로 저장



사각형을 그리는 코드를 짤 때에는 아래 사진과 같이 픽셀 데이터를 반대로 읽어와서 사각형을 그려주어야 한다.

```
for (int Sqare_height = user_select_y_start; Sqare_height < user_select_y_end; Sqare_height++)
{
    for (int Sqare_width = user_select_x_start; Sqare_width < user_select_x_end; Sqare_width++)
    {
        bit_map_index = (infoheader.biHeight - 1 - Sqare_height) * new_width + Sqare_width;
        ReadFile[bit_map_index] = 200;
```

왜냐하면 사용자의 입장에서는 당연히 왼쪽 위가 사각형을 그리는 기준이 되기 때문에 사용자의 기준으로 생각했다.

```
printf("사용자가 입력한 영상은 가로:%d 세로:%d 의 크기를 가지고 있습니다.Wn", new_width, infoheader.biHeight);
printf("사각형을 그릴 위치 값을 입력해주세요Wn영상의 왼쪽 위를 기준으로 X왼쪽: X오른쪽: Y위쪽: Y아래쪽Wn");
```

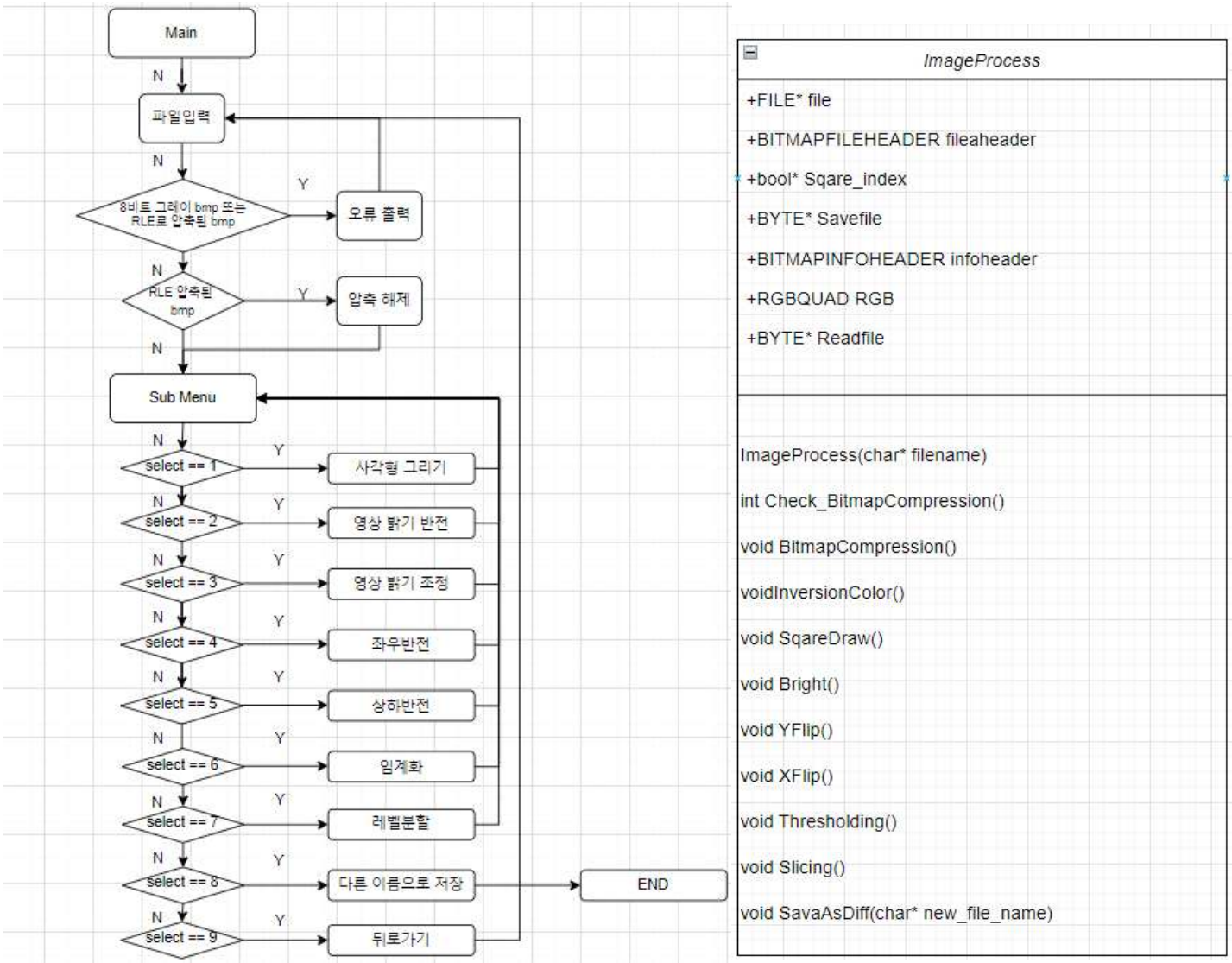
사용자가 이 사실을 알고 프로그램을 잘 사용할 수 있도록 출력해주기도 하였다.

5. 실행 흐름 및 인터페이스 설계

아래는 프로그램의 실행 및 클래스 다이어그램을 나타낸 것이다.
설계보고서 이후 코드 리팩토링 등의 이유로 전체적인 인터페이스의 흐름이 변경되었다. 클래스 다이어그램 또한 설계보고서 이후 새롭게 추가된 두 기능(임계화, 레벨분할)이 포함된 것을 볼 수 있다. 또한 주요 기능들의 실행흐름도를 첨부하여 문제의 핵심 기능(압축해제, 사각형 그리기, 화면반전)들의 플로우차트를 확인할 수 있다.

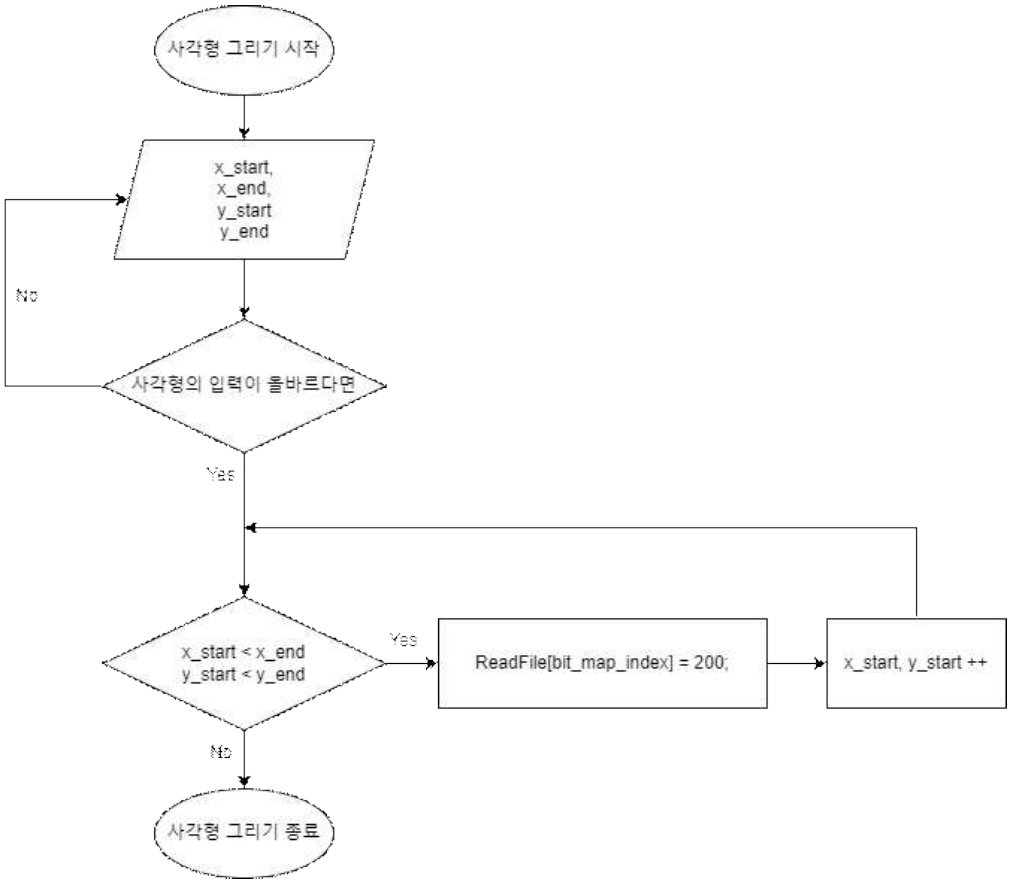
5-1. 인터페이스 플로우차트 및 기능 플로우차트

-전체 인터 페이스의 실행흐름 및 설계

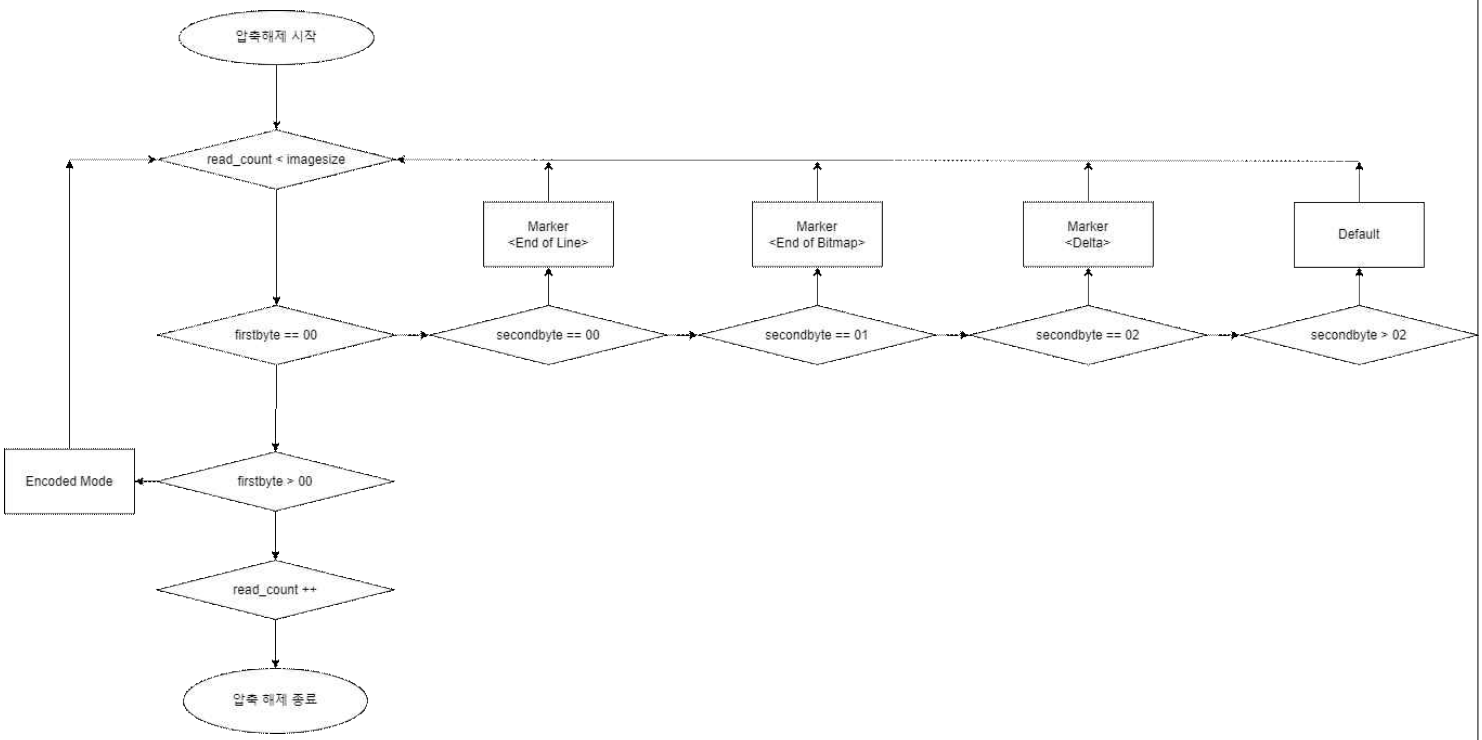


<전체 인터페이스 플로우 차트와 클래스 다이어그램>

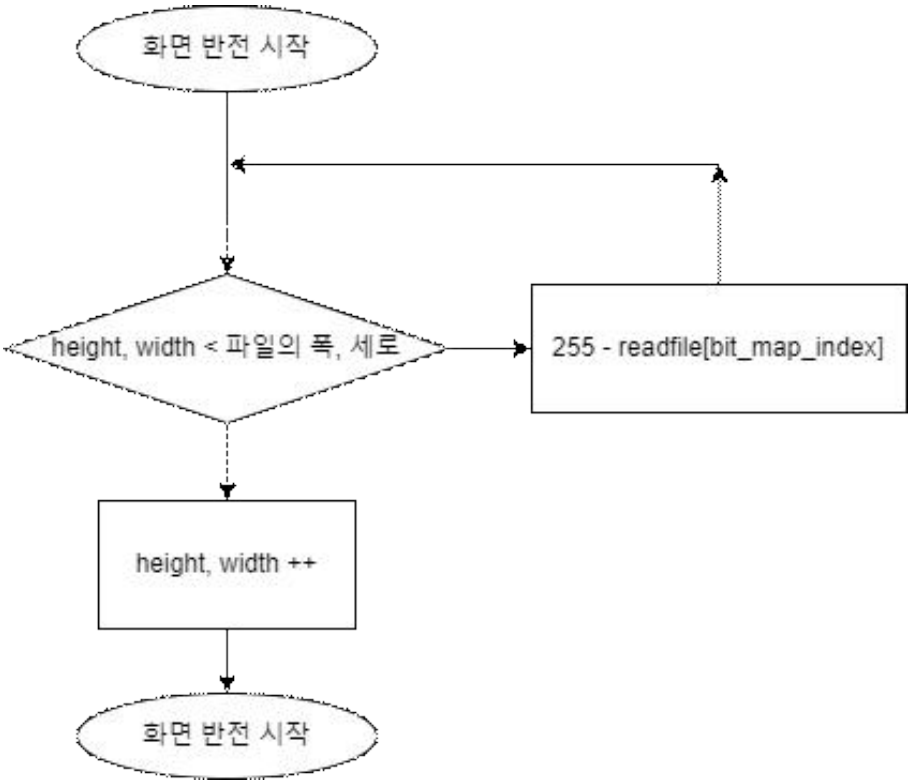
-주요 기능들의 실행흐름 (압축해제, 사각형 그리기, 화면반전)



<사각형 그리기 실행흐름도>



< 압축 해제 실행 흐름도 >



< 화면 반전 실행 흐름도 >

전체 인터페이스 흐름을 보면 메인메뉴에는 파일 입력창이 나오고 만약 8비트 그레이 스케일 파일이라면 바로 서브메뉴로 진입, 만약 RLE로 압축된 8비트 그레이 스케일 파일이라면 압축 해제 후 서브메뉴로 진입하게 된다.
만약 위 두 가지 경우 모두 아닐 때는 오류 메시지를 출력하고 다시 파일을 입력하라는 메시지가 출력된다.

서브메뉴는 총 9가지의 옵션을 가지고 있다. 사각형 그리기, 영상밝기 반전, 영상 밝기 조절, 좌우반전, 상하반전, 임계화, 레벨분할 총 7가지의 파일 수정(영상처리)이 가능하고, 다른이름으로 저장 및 뒤로가기 옵션이 존재한다. 7가지 파일 수정 메뉴 모두 수정이 끝날 때는 다시 서브메뉴로 돌아가도록 설계가 되어있다.

다른 이름으로 저장을 선택한 경우, 어떤 이름으로 수정할 것인지 사용자로부터 받아서 새로운 이름을 가진 파일로 저장한 후 프로그램은 종료된다. 뒤로가기 옵션을 선택한 경우 메인메뉴로 돌아가도록 설계가 되어있다.

압축 해제의 동작 흐름은 첫 번째 바이트를 읽어 그 값이 0인지 아닌지 판단하는 것에서 시작한다. 만약 0이라면 Absoutemode로 진입하고 두 번째 바이트의 값에 따라 Marker, 혹은 불연속적인 값들에 대한 Default 동작을 수행하게 된다. 첫 번째 바이트가 0보다 크다면 Encoded mode로

진입하여 연속적인 값들에 대한 압축해제 동작을 수행하게 된다.

사각형 그리기의 경우 실행 흐름이 RLE 압축방식의 Marker중 하나인 Delta와 유사하며 사용자에게 입력받은 x_start와 y_start 지점을 기준으로 픽셀값을 모두 200으로 바꾸는 동작을 수행하게 된다. 이때 동작은 x_start가 x_end, y_start가 y_end가 될 때까지 진행하게 된다.

밝기 반전은 bmp의 모든 영역에 대해 255-픽셀값을 수행하는 동작을 하며 비트맵의 전체 폭과 세로길이에 대해 동작을 수행하게 된다.

5-2. 인터페이스의 입출력 구조 명세

기능	입력 및 출력
압축 해제	입력: RLE로 압축된 8bit-grayscale BMP 파일
	출력: 압축해제된 8bit-grayscale BMP 파일
색상 반전	입력: 8bit-grayscale BMP파일
	출력: 밝기가 반전된 8bit-grayscale BMP 파일
상하 반전	입력: 8bit-grayscale BMP 파일
	출력: 상하가 반전된 8bit-grayscale BMP 파일
좌우 반전	입력: 8bit-grayscale BMP 파일
	출력: 좌우가 반전된 8bit-grayscale BMP 파일
밝기 조절 입출력	입력 : 8bit-grayscale 파일, 밝기 조절할 값
	출력: 밝기가 조정된 8bit-grayscale 파일
사각형 그리기 입출력	입력: 8bit-grayscale 파일, (x,y)좌표
	출력: 사각형이 그려진 8bit-grayscale 파일
임계화	입력: 8bit-grayscale BMP파일
	출력: 임계화가 완료된 8bit-grayscale BMP파일
레벨분할	입력: 8bit-grayscale BMP파일
	출력: 레벨분할이 완료된 8bit-grayscale BMP파일
다른 이름으로 저장	입력: 8bit-grayscale BMP 파일, 저장할 이름
	출력: 다른 이름으로 저장된 8bit-grayscale BMP 파일

※ 밑줄 친 부분은 구현 보고서 이후 새롭게 추가된 기능임.

5-3. 사용자 인터페이스와 직접적으로 통신하는 내부

- 기존 내부모듈

내부 모듈명	기능 명세
BITMAPFILEHEADER	비트맵 파일 자체에 관한 정보를 가지고 있음
BITMAPFILEHEADER	DIB의 크기(가로 폭, 세로 높이)와 색상 포맷에 관

	한 정보를 가지고 있음
RGBQUAD	밝기 정보를 표현하기 위한 모듈

- 우리가 정의할 내부 모듈

내부 모듈명	기능 명세
BmpCompression()	RLE 파일 압축 해제 기능 수행
squreDraw()	BMP 파일의 원하는 위치에 사각형 출력
inversionColor()	BMP 파일의 밝기를 반전한 결과 제공
Bright()	BMP 파일의 밝기를 조정한 결과 제공
YFlip()	BMP 파일의 좌우 반전된 결과 제공
XFlip()	BMP 파일의 상하 반전된 결과 제공
Threshholding()	BMP 파일의 임계화 결과 제공
Slicing()	BMP 파일의 레벨분할 결과 제공
SaveAsDiff()	영상처리가 완료된 파일을 새로운 이름으로 저장하는 기능 수행

※ 밑줄 친 부분은 구현 보고서 이후 새롭게 추가된 기능임.

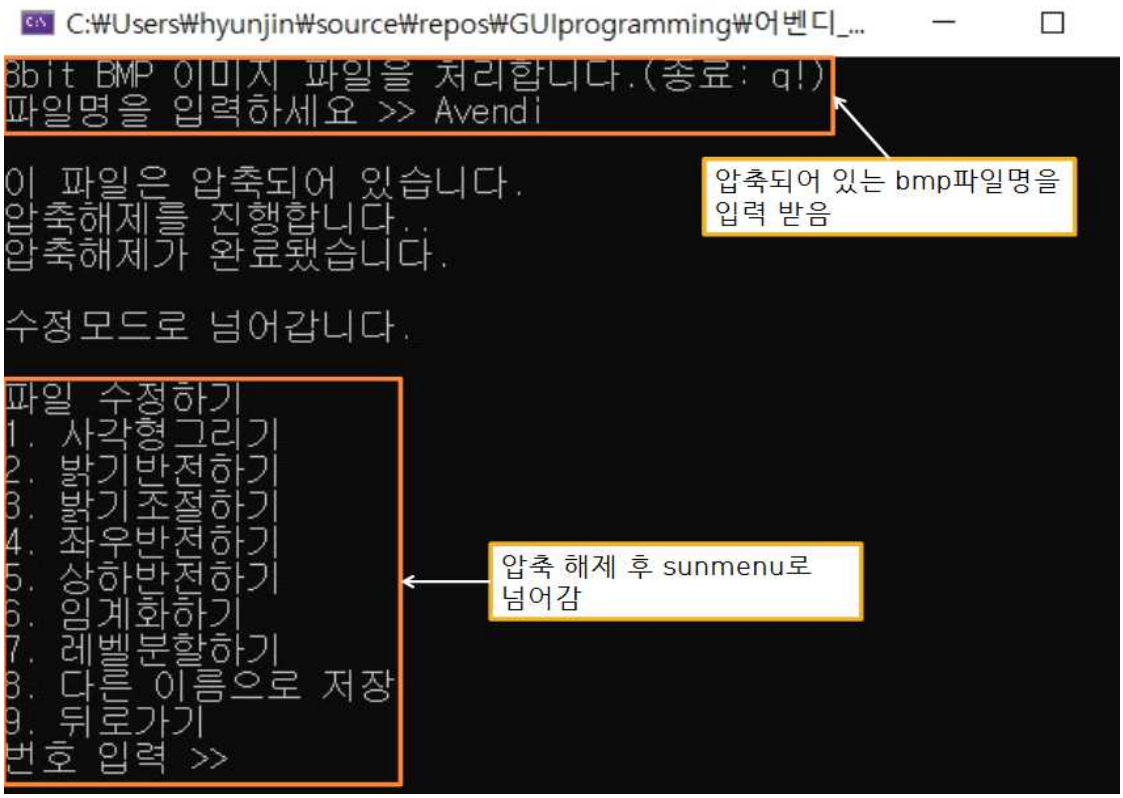
- UI 관점에서 본 내부모듈 명세

1. RLE 파일 압축해제 — **BmpCompression() : RLE 파일 압축해제 기능 수행**
2. 사각형 그리기 — **SqureDraw() : BMP 파일의 원하는 위치에 사각형 출력기능 수행**
3. 영상 밝기 반전 — **InversionColor() : 입력된 bmp 파일의 밝기를 반전한 결과 제공**
4. 영상 밝기 조정 — **Bright() : BMP 파일의 밝기를 조정한 결과 제공**
5. 영상 좌우 반전 — **Yflip() : BMP 파일의 좌우 반전된 결과 제공**
6. 영상 상하 반전 — **Xflip() : BMP 파일의 좌우 반전된 결과 제공**
7. 영상 임계화 처리 — **Threshholding() : BMP 파일의 좌우 반전된 결과 제공**
8. 영상 레벨 분할 — **Slicing() : BMP 파일의 좌우 반전된 결과 제공**
9. 다른 이름으로 저장 — **SaveAsDiff() : BMP 파일의 좌우 반전된 결과 제공**

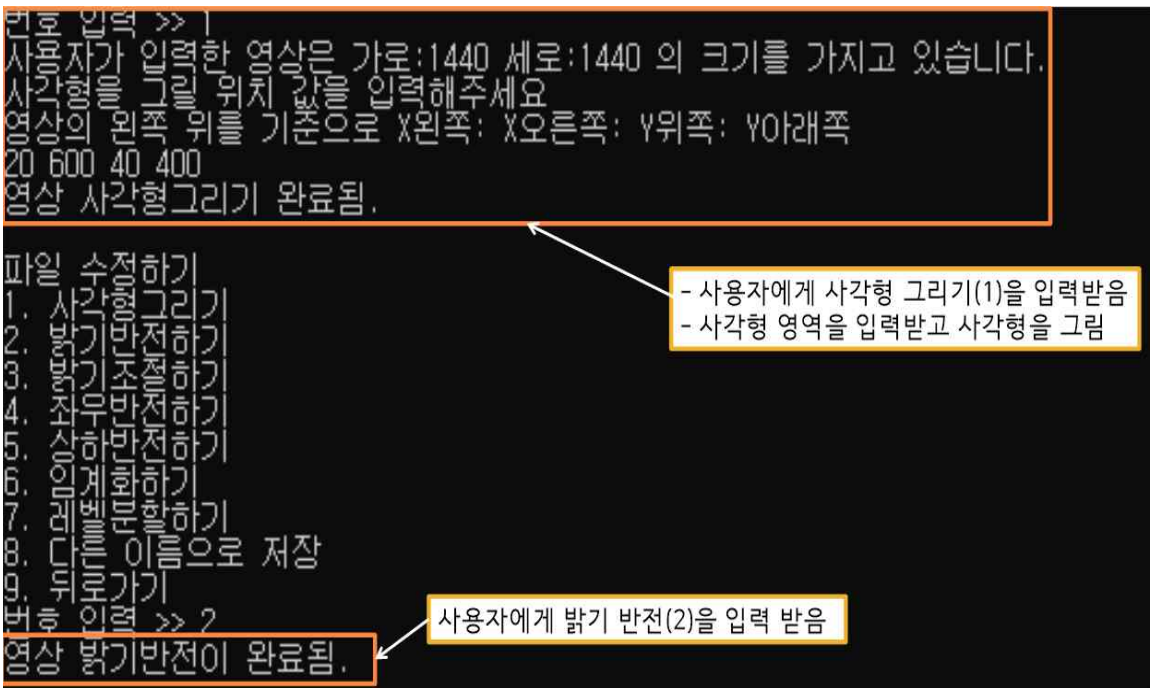
※ 영상 임계화 처리 및 영상 레벨 분할 부분은 구현 보고서 이후 새롭게 추가된 기능임.

6. 실행 화면 및 실행 결과

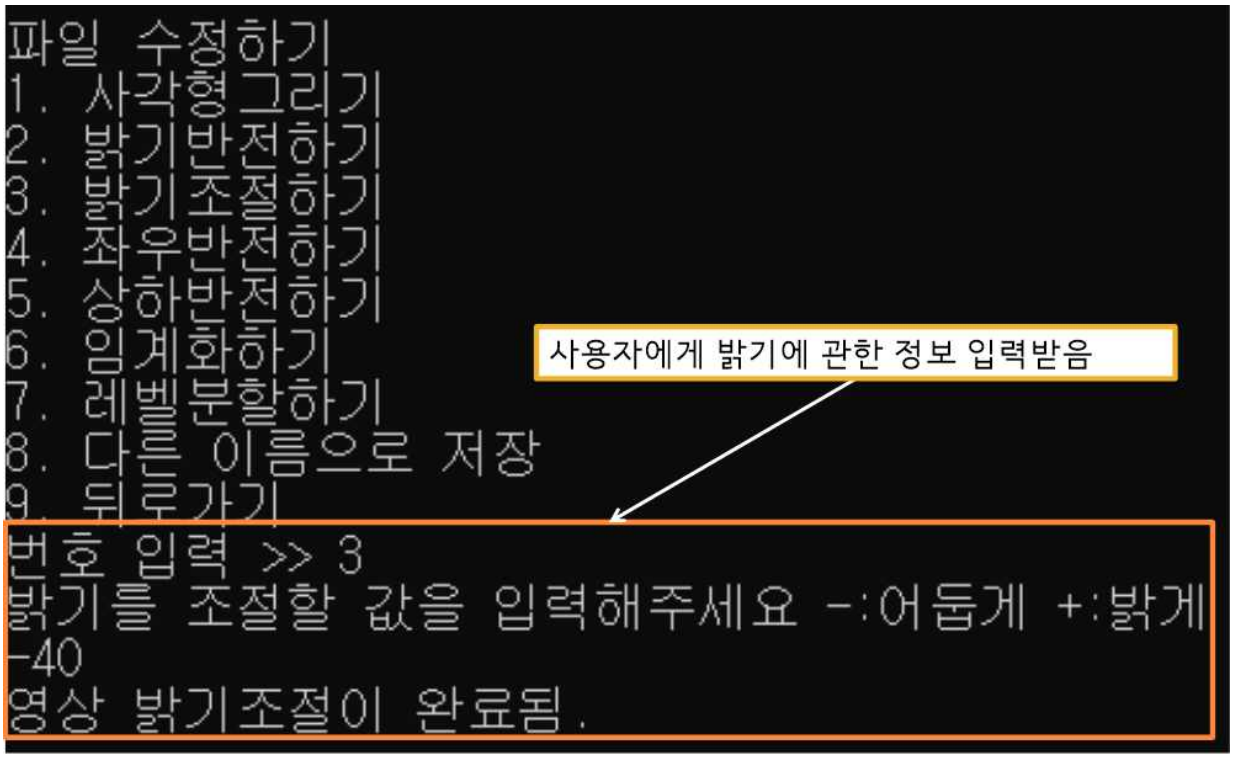
6-1. 실행 화면



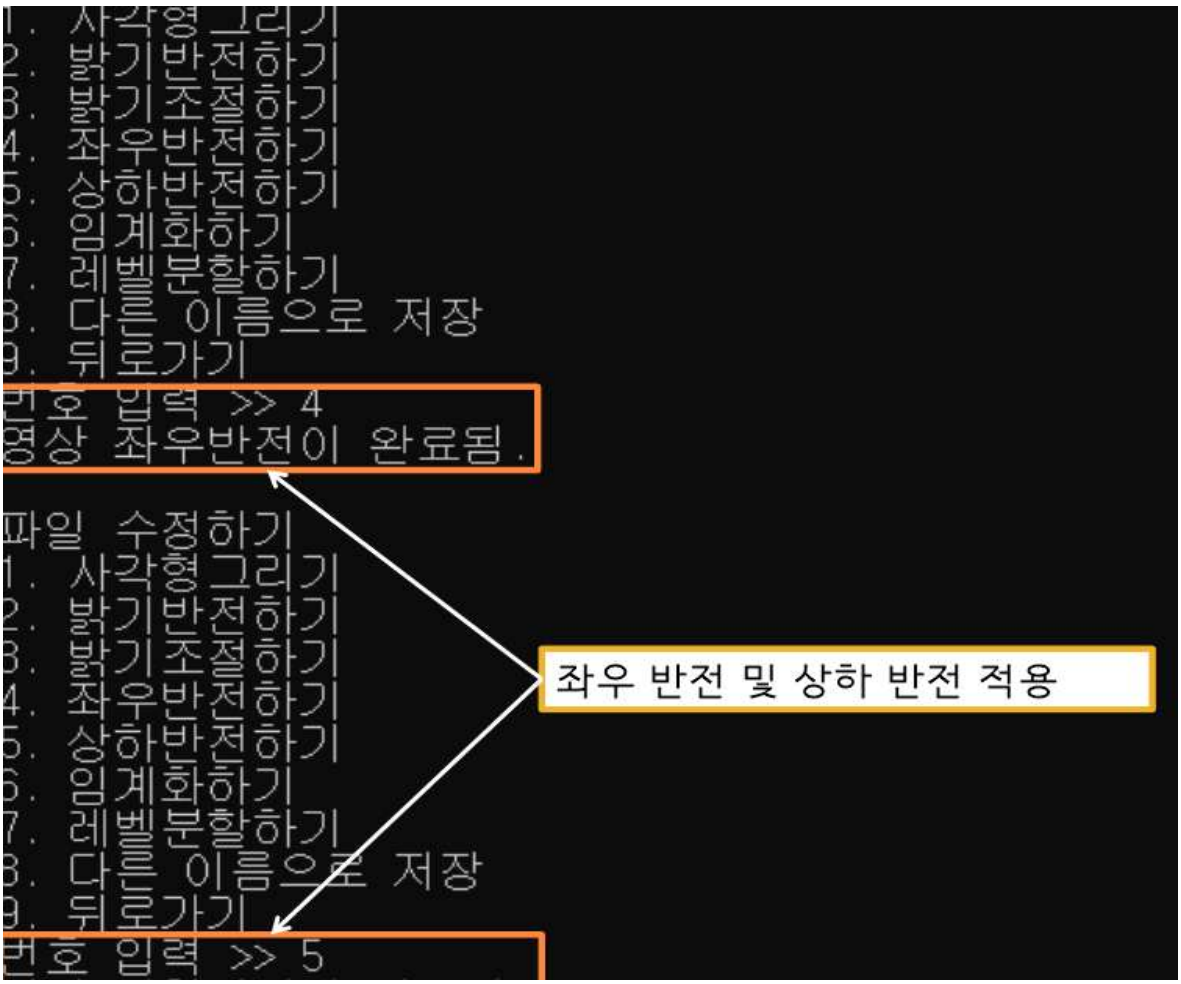
<파일명 입력 후 압축 해제, 수정 모드 진입>



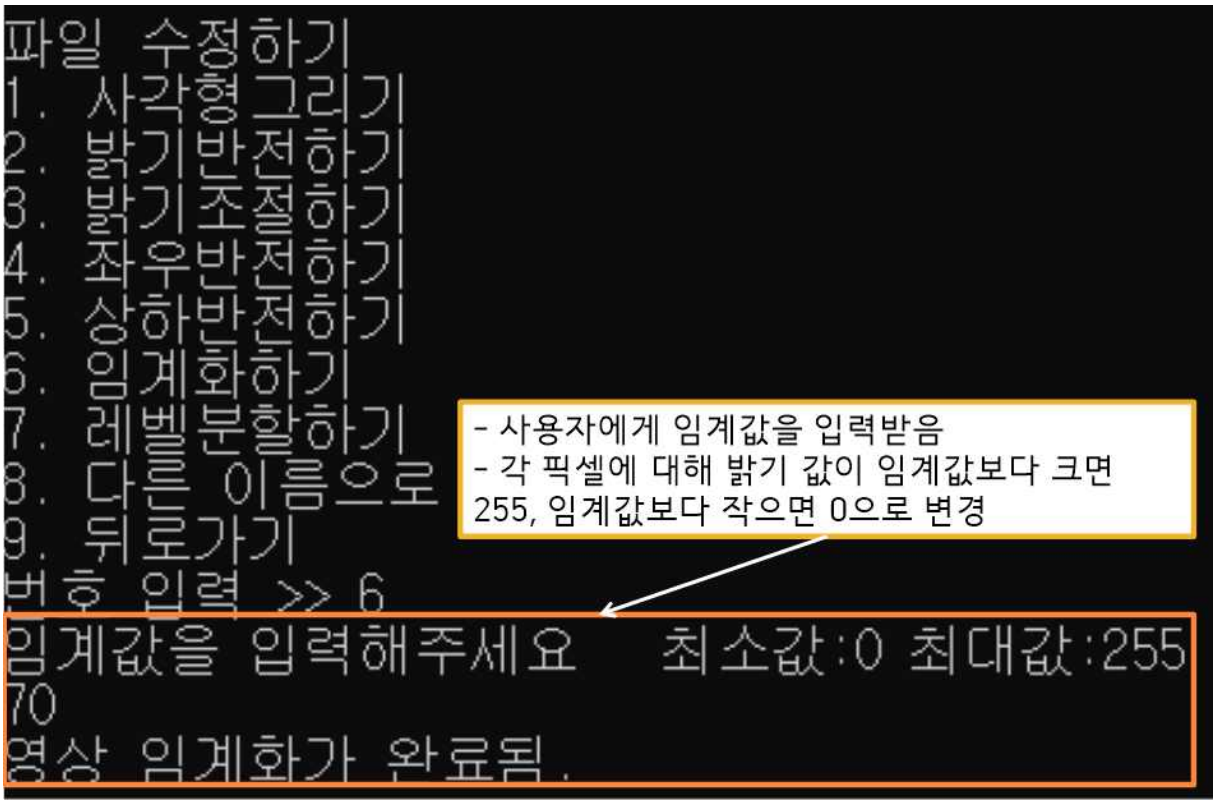
< 사각형 그리기 및 영상 밝기 반전 수행 >



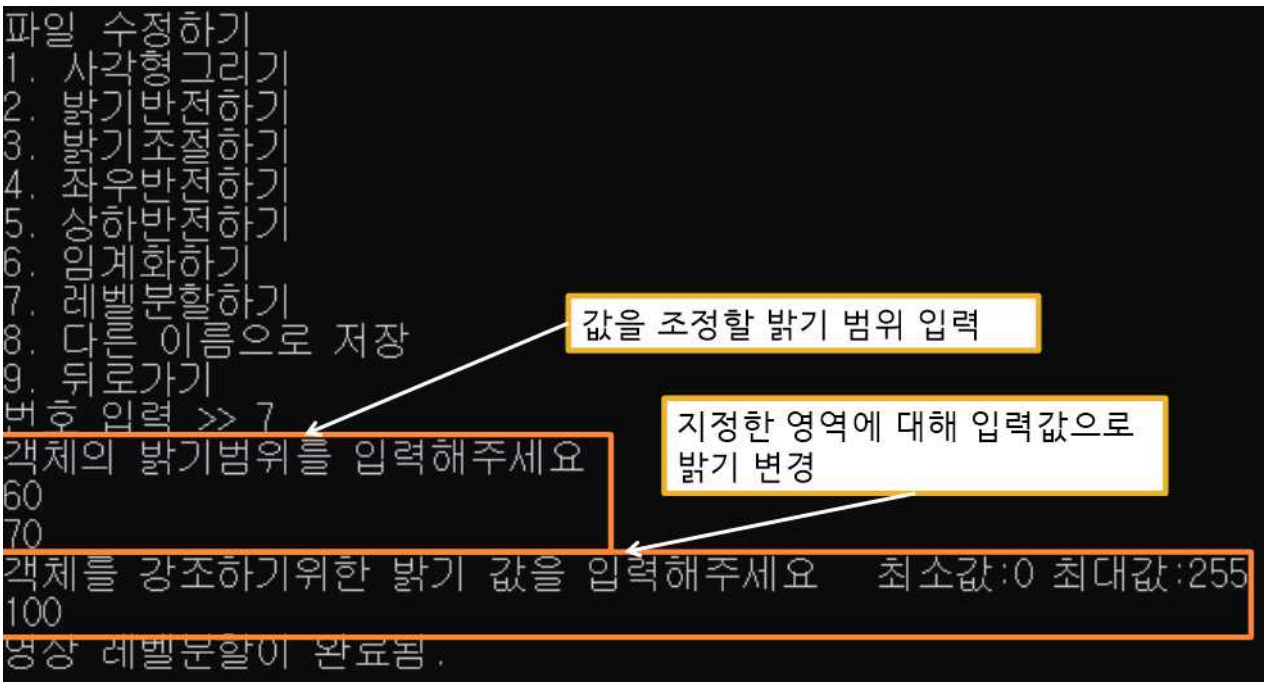
< 사용자에게 입력받은 밝기 조절 값으로 동작 수행 >



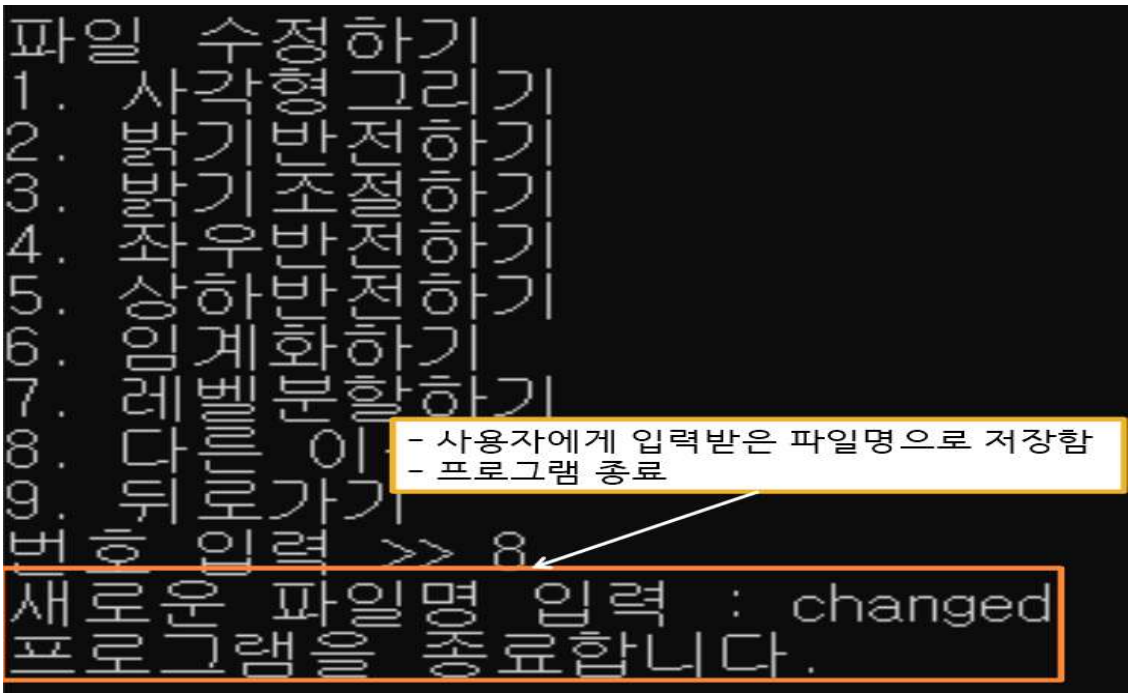
< 좌우 반전 및 상하 반전 수행 >



< 사용자에게 임계값을 입력받은 후 그에 따라 임계화 수행>



< 사용자가 입력한 밝기 레벨 구간에 대해 레벨분할 수행>



<파일명 입력 받고 변경사항 저장>

6-2. 실행 결과

다음은 우리가 제작한 콘솔 입출력문의 예시이다.

(RLE8형식으로 압축된 bmp 파일)

1. 압축된 bmp 파일을 사각형 그리기와 밝기 반전 수행(기본 과제)

추가로 구현한 밝기 조절, 상하좌우 반전, 임계화, 레벨분할도 이와 같은 방법으로 작동하는 것을 알 수 있다.

2. 밝기조절 + 50, 좌우 반전

3. 임계화(임계값 50으로 지정)

4. 레벨 분할(픽셀값 50 - 100 사이, 밝기 60)

7. 설계구성요소 및 설계제한 요소 평가

설계 구성요소				설계 제한요소		
목표설정	분석	구현/제작	시험/평가	성능	규격/표준	테스트
O	O	O	O	▲	O	O

7-1. 설계 구성요소 평가

1. 목표 설정 -

RLE 방식으로 압축된 8bit gray scale bmp 영상을 압축해제 한 후 영상의 밝기반전과 사용자가 원하는 사각형을 그려 새로운 이름으로 저장한다. 이에 추가적으로 밝기 반전과 유사한 영상 좌우, 상하 위치 반전, 또한 원하는 오브젝트와 그 외를 분리하는 임계화, 특정 그레이 레벨 범위를 하이라이트하는 레벨 분할 등을 설계하고 사용자가 콘솔창에서 사용할 수 있도록 설계한다. 따라서 순번으로 표기하면 다음과 같다.

1. RLE8 방식으로 압축된 이미지를 입력으로 받음.
2. 영상 반전, 사각형 그리기 등을 메뉴에서 선택할 수 있게 구현함..
3. 입력으로 받은 압축된 파일을 압축 해제함.
4. 압축 해제한 이미지에 대해 색상을 반전시킬 수 있어야 함.
5. 압축 해제한 이미지 안에 사각형을 그릴 수 있어야 함.
6. 압축 해제한 이미지에 대해 상하/좌우 반전을 수행.
7. 압축 해제한 이미지에 대해 밝기 조절을 수행.
8. 압축 해제한 이미지에 대해 임계화, 레벨분할을 수행.
9. 압축해제와 영상처리의 수행 여부를 사용자가 직접 정하게 유도함.
10. 위 동작들을 수행한 후 다른 이름으로 파일을 저장.
11. 기능별로 분리하여 콘솔창의 메뉴로 제공.

2. 분석 -

우리가 필요한 bmp 파일의 정보는 압축 데이터와 픽셀 데이터인데 이는 구조체로 쉽게 사용 및 변경할 수 있다. 파일입출력을 통해 프로그램에 압축된, 혹은 압축되지 않은 bmp 파일이 입력되는

데 이는 피트맵파일헤더와 비트맵정보헤더에 데이터가 입력된다.
특히 비트맵정보헤더에 biCompression이라는 멤버 변수를 통해 압축방식을 확인 할 수 있는데, 이를 이용하여 현재 입력된 파일의 압축여부를 확인하여 그에 알맞게 동작을 수행할 수 있도록 한다.
영상 처리 후 새로운 이름으로 파일을 따로 저장하는 것은 입력을 받은 비트맵 파일의 정보를 가진 배열(ReadFile)과 똑같은 크기의 배열(SaveFile) 두가지를 두어 압축해제와 영상처리의 최종적인 결과를 두 번째 배열에 삽입하여 영상을 저장하고 출력한다.

3. 구현/제작 -

IDE => microsoft에서 제공하는 VisalStdio 2019를 사용하였다. 또한 절차적 프로그래밍인 C언어로 프로그램으로 구현을 1차례 실시하였으며 프로그램의 유지보수와 업그레이드의 용이함과 신뢰성 있는 소프트웨어 제작할 수 있는 장점을 가진 객체지향프로그래밍 언어인 C++로 리팩토링을 진행하였다.

4. 시험/평가 -

레나 사진에 국한하지 않고, 직접 우리가 촬영한 영상을 영상프로그램으로 bmp 파일로 변환하고 이를 압축한 파일과 일반 bmp 파일로 시험하였다. 또한 bmp파일이 아닌 일반 jpeg파일과 png파일로 입력을 하여 에러와 예외를 처리하는 것을 확인하여 우리가 원하는 프로그램의 방향을 잡음. 이는 뒤에 있는 단위테스트를 통해 수행하였다.
1. 일반 bmp 파일
2. RLE8 방식으로 압축된 bmp 파일
3. bmp 형식이 아닌 파일 (.png .jpeg)

계획한 함수(메소드)	완성 여부	설명
비트맵 압축 여부	o	Check_BitmapCompression()
비트맵 압축해제	o	BitmapCompression()
색상 반전	o	InversionColor()
사각형 회화(그리기)	o	SqareDraw()
밝기 조정	o	Bright()
좌우 반전	o	YFlip()
상하 반전	o	XFlip()
임계화	o	Thresholding()
레벨 분할	o	Slicing()
새로운 이름으로 저장	o	SaveAsDiff(char* new_file_name)

또한 설계 보고서에 명시한 모듈들과 추가로 설계한 두 개의 모듈들 역시 메소드로 별 다른 이슈 없이 정상적으로 수행되게 구현하였다.

7-2. 설계 제한요소 평가

1. 성능 -

어드벤쳐디자인의 목적은 팀원들이 협업하여 프로그램의 기능이 적절하게 수행하는 것 자체가 목표이므로 설계와 구현당시에 프로그램 자체의 성능인 속도와 유저에 친숙한 GUI에 관해 아쉬운 점이 있다. 후에 있을 논의에 대해 이와 관련되어 자세히 살펴볼 예정이다.

2. 규격/ 표준 -

8bit gray scale 비트맵 영상만을 처리한다. => 이 외에는 에러를 출력한다.
이미지의 압축 유무를 판별하여 압축되어있다면 해제한다.
영상처리 전에는 합축 해제만 먼저 수행되어야 한다.
한 번에 하나의 영상만 처리한다.
옳지 않은 입력이 들어오면 오류를 출력하고 다시 입력받는다.

3. 테스트 -

소프트웨어의 테스트 방법 중 하나인 '단위 테스트'를 통하여 유지보수를 진행하였다. 단위테스트는 테스트가 가능한 최소 단위 기준으로 결함을 찾고 개발의 의도대로 하나의 기능만으로 이루어진 메소드(모듈)들을 고립시켜 동작이 원활하게 진행되는 지 확인하고 검증하는 작업을 각 팀원들이 적절히 분배하여 진행하였다.

7-3. 설계 평가

설계의 기본 표준과 제약 조건들을 인지 한 후 목표를 설정하고 분석을 자세히 하였기에 구현과 디버깅에 큰 이슈가 없었다. 이를 통하여 기존에 프로그램은 단순히 코딩이라는 생각을 완전히 탈피할 수 있었던 계기가 되었으며, 다음 고급 레벨에서도 중간 레벨에서 깨달은 사고와 협업등을 이용하여 문제를 해결해 나갈 수 있을 거 같다. 다만 조금 아쉬운 점이 있다면 프로그램 작동에 힘을 쏟아 성능에 관하여 그렇게 많은 시간을 투자하지 않았는데 이 역시 고급 레벨에서 보안해야 할 추가 과제라고도 할 수 있다.

8. 논의

8-1. 개발 내용 및 실험 결과 요약

이번 과제에서 설계하고 구현한 프로그램은 RLE 방식으로 압축되어 있거나 되어있지 않은 8bit gray-scale 영상을 저장하고 있는 BMP 파일을 읽어 여러 가지 영상처리(사각형 그리기, 밝기 반전하기, 밝기 조절하기, 좌우 반전하기, 상하반전하기, 임계화하기, 레벨 분할하기)를 해준 뒤 새로운 이름으로 저장하는 프로그램 이다. 시간 관계상 몇 가지의 추가기능 밖에 삽입을 하지 못 했는데 다음 문단에는 우리 팀원들이 생각한 다른 추가 기능과 개선 과제들을 서술하였다.

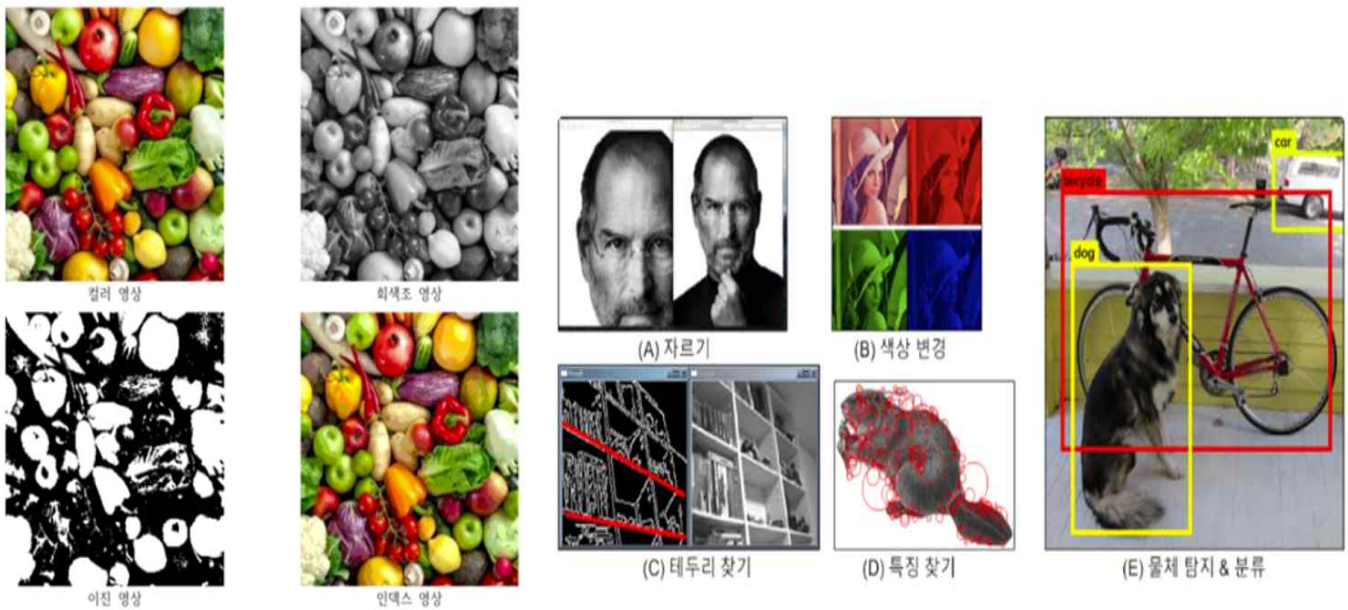
8-2. 향후 개선 과제

현재 우리 프로그램은 콘솔을 통해서 사용자에게 편의성을 제공해 준다. 이는 프로그래머가 아닌 사용자들이 봤을 때는 사용하기 어렵다고 생각한다. 그래서 **MFC를 활용해 GUI 기능** 또한 추가로 구현하여 사용자에게 조금 더 친화적인 인터페이스를 제공하고 싶다. 아래 사진은 직접 그린 GUI 기능 구현시 예상도이다.

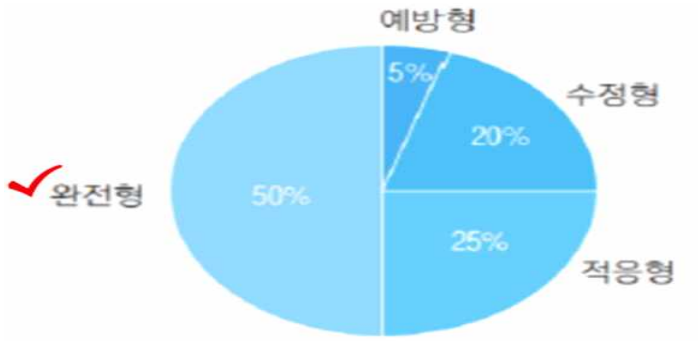
또한 현재 우리가 구현한 프로그램은 압축 해제 기능만 가능하다. 압축 해제 기능만으로는 기능적으로 많이 부족하다고 판단하였다. 그래서 우리는 압축을 푸는 알고리즘을 설계하고 구현하였으면 역설계 역시 가능하므로 **압축하기 기능도 구현**하여 더욱더 활용 범위를 넓히는 모듈을 설계하는 것이 조금 더 유용한 거 같다. 또한 아래 사진에서 보는 것과같이 여러가지 압축 및 압축해제 기능을 추가로 제공한다면 더욱 더 사용범위가 넓어진다고 생각하기 때문에 여러 가지 압축 및 압축해제 기능 또한 구현하는 것 역시 필요하다.

값	식별자	압축 방식	비고
0	BI_RGB	없음	가장 일반적이다
1	BI_RLE8	RLE 8비트/화소	8비트/화소 비트맵에만 사용할 수 있다.
2	BI_RLE4	RLE 4비트/화소	4비트/화소 비트맵에만 사용할 수 있다.
3	BI_BITFIELDS	비트 필드	16, 32비트/화소 비트맵에만 사용할 수 있다.
4	BI_JPEG	JPEG	비트맵은 JPEG 이미지를 포함한다.
5	BI_PNG	PNG	비트맵은 PNG 이미지를 포함한다.

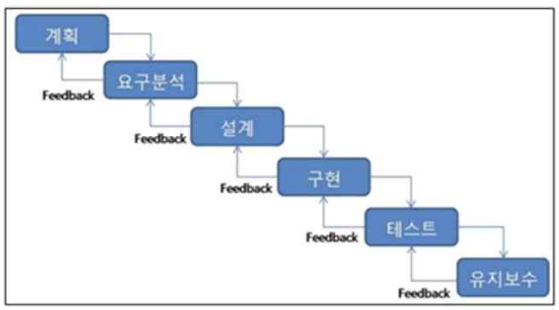
그리고 구현한 프로그램은 그레이스케일 영상만 다루고 있다. 하지만 그레이스케일 영상만 다루는 프로그램으로는 할 수 있는 것들도 많지 않기 때문에 한계가 명확하다고 생각한다. 그래서 **이진 영상, 컬러 영상도 다룰 수 있도록 확장** 역시 고려해 볼 수 있다. 또한 우리가 구현한 프로그램은 간단한 영상처리 기능만 구현하였지만 이진 영상, 그레이스케일 영상. 컬러 영상을 모두 다루게 된다면 당연히 영상처리 기능을 더 추가할 수 있을 것이다. 여러 가지 영상처리 기능 또한 공부하고 추가하고 싶다.



이러한 활동들은 모든 테스트가 완료된 시스템이 사용자에게 최종적으로 인수된 후에 수행하는 활동이기 때문에 유지 보수라고 칭할 수 있다. 유지 보수에 대해서 조사해 보았는데 현재 우리가 구현한 프로그램은 기능적으로 많이 부족하여 상업적으로 배포하기 어렵다고 생각한다. 그래서 우리가 해야 할 유지 보수는 **완전형 유지 보수**이다. 완전형 유지 보수란 시스템의 성능을 지속적으로 향상시키는 유지 보수이다. 우리는 완전형 유지 보수를 통해 지속적으로 향상을 계속해서 시킬 것이다.



앞서 명시했던 완전형 유지보수를 위해서는 우리가 계획보고서에서 계획했던 프로세스를 바꿀 필요가 있다. 저희는 계획단계에서 폭포수모델을 프로세스모델로 잡았지만, **완전형 유지보수를 위해서 Up(통합 프로세스) 모델로 바꾸고자 한다.** Up 프로세스란 소프트웨어를 반복/점진적으로 개발하는 프로세스로 [도입, 정련, 구현, 전이]가 계속 반복되는 프로세스이다. Up 모델이 앞으로 개선해야 할 사항이 많은 저희 프로그램에게는 더욱더 적합하다고 생각한다.



< 폭포수 모델 >

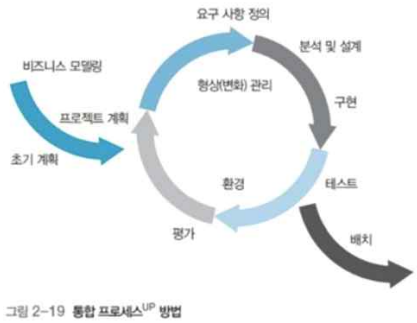


그림 2-19 통합 프로세스UP 방법

< UP (통합 프로세스)모델 >

8-3. 결과물 활용 가능성

우리는 향후 개선과제에서 현재 우리 프로그램은 기능적으로 부족하여 상업적으로 배포하기 어렵다고 판단했기 때문에 완전형 유지 보수를 통해 지속적으로 성능을 향상을 시킨다고 기술한 바 있다. 우리는 지속적으로 성능을 향상시켜 최종적으로는 포토샵, GIMP와 같은 그래픽 프로그램을 만드는 것 목표이다. 물론 아직 우리 프로그램이 포토샵, GIMP와 같은 그래픽 프로그램보다 보다 많이 부족하지만 언젠가는 부족하지 않은 프로그램으로 확장시킨 뒤 제법 쓸만한 그래픽 프로그램으로 만들어 학교 수업 교재(멀티미디어 수업자료)로도 쓰이고 싶고 인터넷에 배포해서 많은 사람들이 사용하도록 하고 싶다.

8-4. 기술적, 사회적, 경제적 파급 효과 및 기대효과

압축을 푸는 알고리즘을 설계하고 구현하였으면 역설계 역시 가능하므로 비트맵 파일 자체를 압축할 수 있는 기능도 한 번 고려해 볼 수 있다고 향후 개선과제에 기술한 바 있다. 우리는 완전형 유지 보수를 하며 압축 해제 기능을 구현을 할 것이다. 이 기능을 활용하여서 이미지 파일의 전송이 필요한 저성능 컴퓨터, 즉 IoT 장비나 이와 결을 같이 하는 임베디드 시스템에서 이미지를 사용하고 전송할 때 우리의 프로그램을 이용하면 압축 기능을 수행한 다음 이미지를 전송하고 받은 다음에 다시 압축을 해제하면 매우 효율적이라고 볼 수 있다.

8-5. 중간레벨 과제 최종 정리

(수행 기간 : 2022년 10월 12일 ~ 2022년 11월 19일)						
기간(주) / 개발내용	1	2	3	4	5	담당 팀원
문제분석 및 자료조사	O					전원
계획 보고서 작성	O					전원
설계 보고서 작성 및 발표자료 제작		O	O			김현진, 안현진
프로그램 구현			O	O	O	송제용, 정지수
결과 보고서 작성 및 발표자료 제작					O	김현진, 안현진

전체 프로그램 제작 소요 시간 그래프에서를 보면 알 수 있듯이 구현 전 설계와 분석 단계에서 많은 시간을 투자 할 수 있었기에 디버깅과 구현 부분, 리팩토링에서 별 다른 이슈 없이 진행 할 수 있었다. 또한 간트 차트에서 볼 수 있듯이 팀원 간의 소통과 협업, 그리고 분업이 상당히 중요한 것을 알 수 있었으며 이는 고급 레벨을 수행할 때도 이와 같이 진행할 예정이다.

9. 역할 분담

프로그램 구현 및 최종 보고서의 역할분담은 아래와 같이 담당하였다.

안현진	김현진	송제용	정지수
각 모듈 별 실행 흐름 분석 실행 화면 자료 제작 프로그램 리팩토링	설계 구성요소 및 설계 제한요소 평가 작성 실행 결과 자료 제작 최종 보고서 추합 및 작성 프로그램 코드 초안 구현	자료구조 설계 작성 프로그램 설계 작성 프로그램 구현내용 분석 논의 작성 프로그램 코드 구체화, 기능 구현 및 주석 작성	서론 및 문제 정의 인터페이스 설계 피피티 제작 프로그램 코드 초안 구현