프로젝트 설계 보고서

압축된 BMP이미지 파일의 변환

작성일	2022.11.03	제출일	2022.11.06
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. 요구사항 분석

본 보고서는 중간레벨 프로젝트 설계 보고서임을 명시한다.

본격적인 프로그램의 설계에 앞서 요구사항에 대해서 분석해보려고 한다. 소프트웨어의 요구사항에 대해서는 FURPS+모델에 의해 기능적 요구사항과 비기능적 요구사항으로 분류할 수 있다. 기능적 요구사항은 단어의 뜻 그대로 사용자에게 제공해야 하는 기능에 대한 요구사항을 의미한다. 기능적 요구사항은 다음과 같다.

1-1. 기능적 요구사항 및 비기능적 요구사항

기능적 요구사항

1. RLE방식으로 압축된 이미지를 입력으로 받는다.
2. 영상 반전, 사각형 그리기 등을 메뉴에서 선택할 수 있게 구현한다.
3. 입력으로 받은 압축된 파일을 압축 해제한다.
4. 압축 해제한 이미지에 대해 색상을 반전시킬 수 있어야 된다.
5. 압축 해제한 이미지 안에 사각형을 그릴 수 있어야 한다.
6. 압축 해제한 이미지에 대해 상하/좌우 반전을 할 수 있어야 한다.
7. 압축 해제한 이미지에 대해 밝기 조절을 할 수 있어야 한다.
8. 3번 요구사항과 4번 요구사항의 수행 여부를 정할 수 있어야 한다.
9. 위 동작들을 수행한 후 다른 이름으로 파일을 저장한다.
10. 기능별로 분리하여 메뉴를 제공한다.

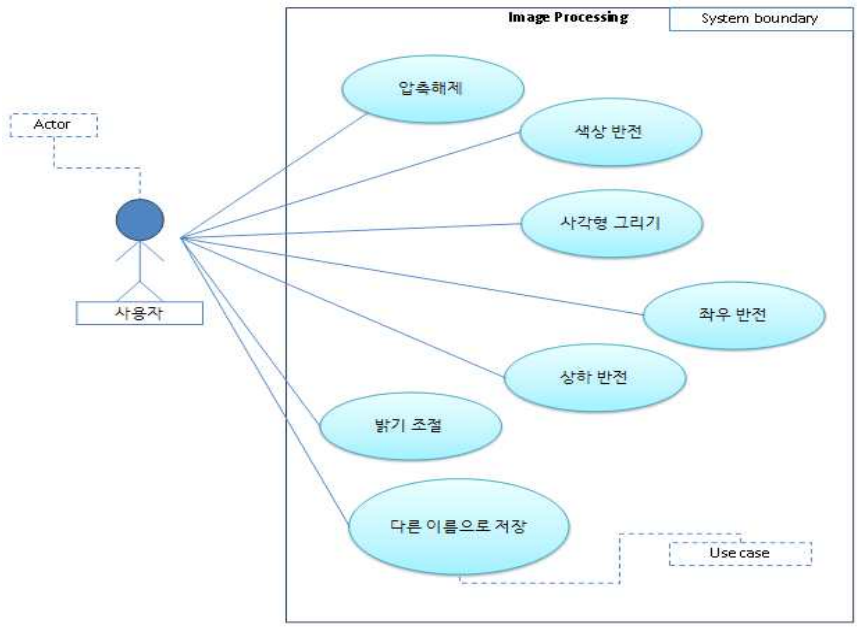
비기능적 요구사항은 흔히 기능적 요구사항이 아닌 모든 요구사항을 의미하며 주로 사용성, 신뢰성, 성능, 지원성 등이 포함되어있다. 이러한 사실을 고려하였을 때 이번 프로젝트에서 고려할 수 있는 비기능적 요구사항은 다음과 같다.

비기능적 요구사항

1. 8bit-grayscale 비트맵이 아니거나 존재하지 않는 파일을 입력받으면 에러를 출력한다.
2. 영상처리 전에 압축 해제가 최초로 먼저 수행되어야 한다.
3. 한 번에 하나의 영상만 처리할 수 있다.
4. 올바르지 않은 입력을 받는다면 오류를 출력하고 다시 입력받는다
5. 이미지의 압축 유무를 판별해서 만약 이미지가 압축되어있다면 압축을 해제하도록 한다.

1-2. UseCase 다이어그램 및 명세

Usecase는 시스템의 동작을 사용자의 입장에서 표현한 시나리오이며, 시스템에 관련된 요구사항을 알아내는 과정이다.
Usecase는 시스템을 사용하거나 보조하는 Actor, 시스템의 경계를 지정하는 System boundary, 그리고 각각의 Usecase들로 구성되어있으며 다음 그림은 이러한 정보를 기반으로 UseCase 다이어그램으로 나타낸 것이다.



UseCase 다이어그램

1-3. Usecase 시나리오

위의 그림을 봤을 때 각각의 Usecase들이 프로젝트의 기능적 요구사항임을 확인할 수 있다. 실제로 기능적 요구사항을 찾기 위하여 Usecase를 많이 활용하게 된다. 다음은 각각의 Usecase들에 대한 성공시나리오를 텍스트로 작성한 것이다.

Usecase : 압축 해제
1. 사용자가 rle방식으로 압축된 8bit-grayscale이미지를 입력으로 전달한다.
2. 압축을 해제하고 시스템 내의 파일 객체에 저장한다.

Usecase : 상하 반전
1. 사용자가 상하반전에 대한 옵션을 선택한다.
2. 상하 반전이 적용된 데이터가 파일 객체에 저장된다.

Usecase : 사각형 그리기
1. 사용자가 사각형 그리기에 대한 옵션을 콘솔창을 통해 입력한다.2. 만약 사용자가 사각형 그리기 옵션을 선택한다면 사각형의 좌표와 넓이를 입력받는다.3.사각형 그림이 적용된 파일이 파일객체에 저장된다.

Usecase : 다른 이름으로 저장
1. 사용자가 콘솔을 통해서 파일명을 입력한다.
2. 현재 프로그램이 존재하는 디렉토리에 입력받은 이름으로 파일을 저장한다.

Usecase : 색상반전
1. 사용자가 콘솔을 통해 이미지에 대한 색상 반전 옵션을 선택한다.
2. 만약 사용자가 색상 반전 옵션을 선택했다면 색상 반전을 적용시킨다.
3. 색상 반전이 적용된 파일을 파일 객체에 저장한다.

Usecase : 좌우 반전
1. 사용자가 좌우 반전에 대한 옵션을 선택한다.
2. 좌우 반전이 적용된 데이터가 파일 객체에 저장된다.

Usecase : 밝기 조절
1. 사용자가 밝기 조절에 대한 옵션을 선택한다.
2. 사용자가 밝기 조절 수치를 입력한다.
3. 밝기 조절된 수치가 적용되어 저장된다.

1-4. 정리

이렇게 본격적인 설계에 앞서 FURPS+모델을 고려하여 기능적 요구사항과 비기능적 요구사항을 나누고 이를 토대로 UseCase 다이어그램와 각각의 시나리오에 대해서 생각해보았다. 이번 프로젝트의 경우 영상처리라는 큰 범위의 요구사항이 존재하고 이를 토대로 그에 대항하는 다양한 기능들을 제공하는 프로그램을 만드는 것이기 때문에 모든 요구사항이 명확하다. 다음 목차에서는 이번에 분석한 요구사항들을 바탕으로 시스템을 설계해보려고 한다.

2. 시스템 설계

요구사항에서 UseCase의 시나리오 대로 우리의 목표를 조금 더 구체적으로 잡기 위해 시스템 설계에 들어간다. 시스템 설계는 전체적인 모듈과 알고리즘 설계 전 기본적인 입출력과 제약조건 등등 “틀”을 잡는 거라고 이해하면 되겠다.

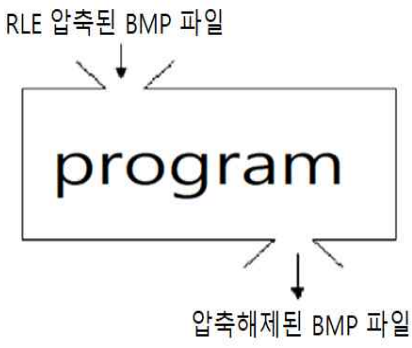
2-1. 프로그램 전체 기능 및 목표 명세

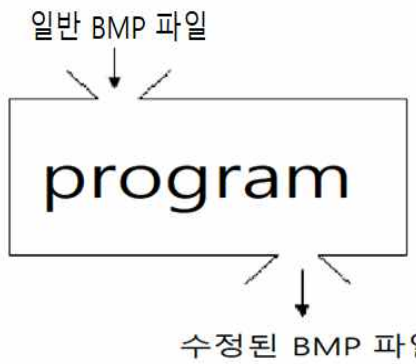
프로그램의 최종목표

‘RLE 방식으로 압축된 8bit grey-scale bmp 파일의 변환’을 목표로 한 소프트웨어이다. 최종적인 목표는 Actor의 수정을 거친 bmp 파일을 출력을 가지는 것이므로 다음과 같은 우리 팀의 프로그램의 최종적인 목표를 살펴볼 수 있다.

INPUT : RLE 방식으로 압축된 BMP파일
또는 압축되지 않은 BMP파일

OUTPUT : 압축 해제한 BMP 파일
색상반전과 사각형그리기와 같은 유저의 수정을 거친 BMP파일을 새로운 이름으로 저장





다음 장에 소개될 알고리즘이나 모듈, 인터페이스의 설계등이 이러한 최종적인 목표를 고려하여 설계된다. 추가적으로 후에 설계될 모듈이나 알고리즘들을 보다 수월하게 구현하고 목표를 상기시키기 위해 프로그램의 전체 기능을 명세한다.

구체화(프로그램 전체 기능)

1.파일 입출력을 통한 압축된 BMP파일(또는 일반 BMP파일)을 입력.

2-1. RLE 압축 BMP 파일인 경우 압축을 해제.

2-2. 파일을 수정
- 2-2.1. 색상 반전 (압축해제한 BMP 파일)
- 2-2.2. 사각형 그리기 (압축해제한 BMP 파일)
- 2-2.3. 좌우, 상하 반전 (압축해제한 BMP 파일)
- 2-2.4. 색상 수정 (압축해제한 BMP 파일)

3. 다른 이름으로 저장 및 종료(출력)

이러한 프로그램 전체 기능을 console로 입출력을 받을 예정이다. 사용자가 유연하게 기능을 사용하기 위해 압축(1)과 수정(2)부분을 인터페이스에서 아예기능을 나눠 기능을 실행하게 시스템 설계를 구성하였다. 순차적으로 설계해도 구현에 큰 영향은 가지 않지만 추가적인 우리의 컨셉인 '일반 파일도 입력할 수 있다.'를 고려하여 기능을 분리하여 설계한다.

2-2. 프로그램 실행 및 제약 조건 명세

프로그램 실행

콘솔창에 메뉴를 설정해 프로그램 전체 기능을 토대로 사용자가 메뉴의 번호와 파일을 입력하는 방식으로 프로그램을 설계한다. 전에 명세하였듯이 압축해제와 파일의 수정 기능은 순차적이지 않고 따로 독립적인 기능으로 고려하여 설계한다. 또한 각 메뉴와 모듈마다 예외처리와 각종 경우를 고려하여야 하는데 각 상황마다 자세한 사항은 프로그램 설계부분에서 자세히 명세한다.

다음은 우리가 예상한 콘솔 입출력문의 예시이다.

```
압축된 BMP 파일 처리(중간 레벨 5번)
1. BMP 파일 압축해제(RLE)
2. BMP 파일 수정
3. 종료
번호를 입력해주세요 >>> 1

BMP 파일 압축해제
------------------
압축된 파일명 입력해주세요(현재 디렉터리에서 선택) >>>  file.bmp
압축해제된 파일의 이름을 입력해주세요 >>> new.bmp
완료!

압축된 BMP 파일 처리(중간 레벨 5번)
1. BMP 파일 압축해제(RLE)
2. BMP 파일 수정
3. 종료
번호를 입력해주세요 >>> 2

BMP 파일 수정
------------------
1. 색상 반전   2. 좌우반전  3. 상하반전 4. 밝기 조절   5. 사각형그리기   6. 뒤로가기
번호를 입력해주세요 >>>  //4 5 6번을 제외한 나머지를 선택

파일을 입력해주세요(현제 디렉터리에서 선택) >>> new.bmp
실행중...
수정 완료!


//4번 선택(밝기 조절)

파일을 입력해주세요(현제 디렉터리에서 선택) >>> new.bmp
밝기를 선택해주세요(어둡게 : - , 밝게 : +) >>> + 30
실행중...
수정 완료!
```

//5번 선택(사각형 그리기)

파일을 입력해주세요(현재 디렉터리에서 선택) >>> new.bmp
현재 파일의 크기는 300, 200 입니다. 범위 내로 지정을 해주세요.
x y 좌표 1 >>> 10, 30
x y 좌표 2 >>> 40, 20
실행중...
수정완료!

//예외 처리

압축된 파일명 입력해주세요(현재 디렉터리에서 선택) >>> 현진잘생김.bmp
!!!파일이 현재 존재하지 않습니다. 첫 화면으로 돌아갑니다.

압축된 BMP 파일 처리(중간 레벨 5번)
1. BMP 파일 압축해제(RLE)
2. BMP 파일 수정
3. 종료
번호를 입력해주세요 >>>

제약 조건 명세

본격적인 모듈, 알고리즘, 인터페이스를 설계하기 전, 강의 자료에 명세되어 있던 제약 조건과 추가적으로 고려해본 조건(약속)들을 나열하였다.

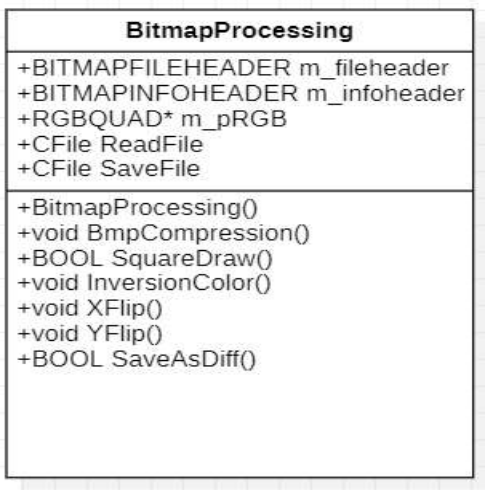
1. 전역 변수의 사용을 최소화한다. 메모리의 효율성을 높이고 적절한 모듈화를 이루어 기존 코드의 재사용과 유지보수의 간결함을 높인다.
2. 사용자가 가지고 있는 파일은 RLE BMP파일이나 일반 BMP 파일이라고 한정하지 않는다. 다시 말해 두 파일이 아니라 전혀 다른 파일이 들어 올 수도 있는 경우도 존재한다는 것이다. 이에 따른 오류와 예외를 따로 처리해주어야 한다.
3. defalut 배경색은 흰색(255)로 설정한다.
4. 마커 0,1,2에 대한 처리과정에서 나머지 영역이 존재할 경우는 거의 발생하지 않으므로 구현 단계에서 따로 고려하지 않는다.

5. RLE 방식으로 압축한 BMP 파일의 제작은 포토샵으로 저장하여 사용한다.
6. BMP 파일의 영상 데이터는 각 행이 4바이트의 배수로 저장되어야 한다.
7. 기능별로 분리하여 메뉴(인터페이스)를 제공한다. 큰 틀로 압축과 수정으로 나누고, 수정 부분은 영상 반전, 사각형 그리기 등을 메뉴에서 선택할 수 있게 구현한다.

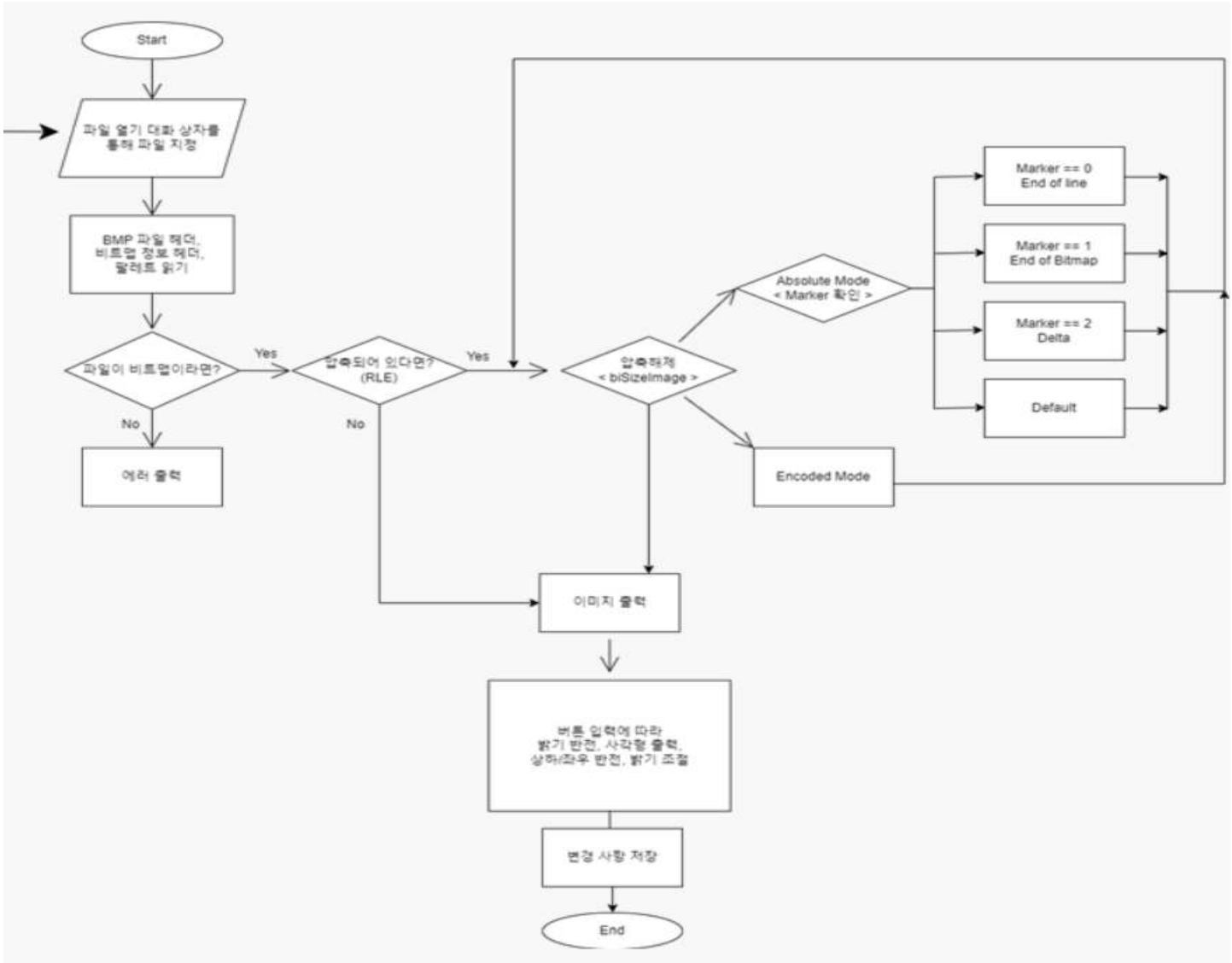
2-3. 모듈 간 상호작용 정의

프로그램의 최종 목표와 실행, 제약 조건을 모두 살펴보았다. 이 세가지를 종합하여 모듈의 틀을 설계해본다. 먼저 설계할 모듈들을 명세하고 이를 클래스 다이어그램과 플로우 차트를 제작해보았다.

클래스 다이어그램.



플로우차트.



플로우차트에선, 입력된 BMP파일이 압축여부를 검사하고 만약 압축이 되어있다면 이를 해제한다. 그리고 이러한 파일을 수정으로 들어가며 사용자가 원하는 방식대로 수정을 거친 뒤 변경 사항을 저장하고 새로운 이름을 입력하면 프로그램은 종료한다.

클래스 다이어그램은 사실 그림이 살짝 단촐해보인다. 주어진 과제와 조건들을 모듈로 명세하고 이를 메소드(함수)로도 구현이 충분하다는 결론이 나와 다이어그램이 이렇게 귀결되었다. 이제부턴 클래스 다이어그램 속 메소드, 즉 모듈을 한 번 살펴보자.

1. RLE 파일 압축 해제 : RLE 파일 압축 해제 기능 수행.
2. 사각형 그리기 : Bmp 파일의 원하는 위치에 사각형 출력 기능 수행.
3. 영상 밝기 반전 : 입력된 bmp 파일의 밝기를 반전한 결과 제공.
4. 영상 밝기 조정 : bmp 파일의 밝기를 조정한 결과 제공.
5. 영상 좌우 반전 : bmp 파일의 좌우 반전된 결과 제공.

6. 영상 상하 반전 : bmp 파일의 상하 반전된 결과 제공.
7. 파일 저장 : 영상 처리가 완료된 파일을 새로운 이름으로 저장하는 기능 수행.

이러한 모듈들은 명확한 요구사항에 각 기능을 알맞게 수행한다. 이를 자세히 보기 위해 다음장을 살펴보면 되겠다.

2-4. 내부데이터 분석

잠시 처음으로 돌아가 우리가 수정할 압축된 BMP 파일 내부 데이터를 분석해보기로 한다. 이를 위해 XVL32 라는 프로그램을 사용하였다.

압축되지 않은 일반 8bit grey scale 레나 사진.

-비트맵파일헤더(BITMAPFILEHEADER), 비트맵정보헤더(BITMAPINFOHEADER)

0	42	4D	36	04	04	00	00	00	00	00	36	04	00	00	28	00
10	00	00	00	02	00	00	00	02	00	00	01	00	08	00	00	00
20	00	00	00	00	04	00	00	00	00	00	00	00	00	00	00	01
30	00	00	00	00	00	00	00	00	00	00	01	01	01	00	02	02

-픽셀데이터
(비트맵 데이터를 찾을 수 있는 시작 오프셋이 1078이고 이는 10진수이므로 16진수 변환시 436부터 비트맵 데이터가 시작된다.)

430	FE	00	FF	FF	FF	00	2C	2C	37	42	36	2F	32	33	3B	33
440	2F	33	35	35	4F	6D	91	B8	C3	C4	C4	C8	C6	C6	C7	C1
450	C0	BF	B7	B2	9F	78	69	6E	77	83	87	8E	9B	9F	A5	A3
460	A7	B2	AE	B6	B4	B3	B4	B5	B2	B6	B4	B7	B6	B7	AF	B3
470	AD	A3	7F	4E	5F	5E	45	38	35	47	30	2B	27	2B	28	29
480	36	3E	46	3C	22	26	36	49	2B	2F	4B	47	3F	55	38	40
490	3E	48	46	39	46	48	45	54	41	39	35	3A	49	68	61	46
4A0	88	A2	5C	3A	35	36	4C	43	25	33	3B	3A	48	72	71	71

RLE8형식으로 압축된 8bit grey scale 레나 사진.

-비트맵파일헤더(BITMAPFILEHEADER), 비트맵정보헤더(BITMAPINFOHEADER)

0	42	4D	CC	15	04	00	00	00	00	00	36	04	00	00	28	00
10	00	00	00	02	00	00	00	02	00	00	01	00	08	00	01	00
20	00	00	96	11	04	00	12	0B	00	00	12	0B	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	01	01	01	00	02	02

-픽셀데이터

(비트맵 데이터를 찾을 수 있는 시작 오프셋이 1078이고 이는 10진수이므로 16진수 변환시 436부터 비트맵 데이터가 시작된다.)

430	FE	00	FF	FF	FF	00	00	FF	2C	2C	37	42	36	2F	32	33
440	3B	33	2F	33	35	35	4F	6D	91	B8	C3	C4	C4	C8	C6	C6
450	C7	C1	C0	BF	B7	B2	9F	78	69	6E	77	83	87	8E	9B	9F
460	A5	A3	A7	B2	AE	B6	B4	B3	B4	B5	B2	B6	B4	B7	B6	B7
470	AF	B3	AD	A3	7F	4E	5F	5E	45	38	35	47	30	2B	27	2B
480	28	29	36	3E	46	3C	22	26	36	49	2B	2F	4B	47	3F	55
490	38	40	3E	48	46	39	46	48	45	54	41	39	35	3A	49	68
4A0	61	46	88	A2	5C	3A	35	36	4C	43	25	33	3B	3A	48	72

RLE8형식으로 압축된 8bit gray scale 레나사진의 분석은 계획보고서의 레나사진을 분석한 결과와 동일하므로 이는 생략한다. 우리가 중요하게 봐야하는 것은 이 압축과 비압축의 차이점이다.

먼저 비트맵파일헤더부분에서는 2-5번째 BMP 파일의 크기를 나타내는 정보가 변환되었다. 그리고 비트맵정보헤더 부분에서는 30-33번째 압축방식을 나타내는 정보가 바뀌었다.(0000 -> 0100. 이 정보는 리틀엔디언 방식이므로 0 -> 1.)

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.

픽셀데이터부분 역시 압축된 모습을 볼 수 있는데 압축된 부분의 픽셀데이터에서는 00FF로 시작하는 모습을 볼 수 있다. 숫자를 보면 직관적으로 이해가 가겠지만 굉장히 비효율적으로 압축된 것이다. 이에 대한 이유를 분석해본다면 만약 픽셀데이터가 반복적인 부분이 빈번하게 존재한다면 효율적이겠지만 레나사진처럼 여러 가지의 색의 밝기 값이 쓰여진 영상에서는 픽셀데이터의 값이 반복적이지 않기 때문에 비효율적이다. 이는 RLE8형식의 압축방식이 효율이 좋지 않다는 것을 의미한다.

사진에서 보면 알 수 있듯 파일의 크기도 증가하였다.

2-5. 정리

지금까지 시스템 정리를 정리해보았다. 요구사항 분석에서 각 UseCase를 분석한 것을 토대로 시스템의 최종적인 입출력값을 정리하고, 이를 위한 전체적인 기능, 프로그램의 실행과 제약 조건을 명세하였다.이러한 결과물들을 거쳐 플로우차트와 클래스 다이어그램, 내부데이터를 살펴봤고, 이에 필요한 모듈들의 윤곽을 제작하였다. 조금 더 구체적으로 명세될 모듈들의 자세한 사항들은 다음 장에서 살펴보도록 한다.

3. 프로그램 설계

시스템 설계에서 초안을 만든 모듈을 이제 구체화하는 작업을 거친다. 자세한 모듈의 입출력과 제약 조건, 주요 변수와 그 목적을 명세하고 이에 대한 알고리즘을 작성한다. 설계의 꽃이라고도 볼 수 있다. 유지보수를 위해선 프로그램 설계 부분을 먼저 건드려야 하기에 세부적인 설계가 필요하다.

3-1. 모듈 사용 목적,기능 및 제약 조건 명세

시스템 설계에서 제작한 모듈을 본격적으로 쓰임새와 그 기능을 명세한다. 명세하였듯이 7개의 모듈 모두 메소드로 구현할 예정이며 이는 슈도코드로 기술할 수 있다. 첫 번째로 목적, 기능과 제약조건을 살펴보자.

1.RLE 압축파일 압축 해제

실행 및 제약조건:

어떤 파일을 압축해제 할 것인지 파일의 이름을 입력 받아야함. 압축방식이 RLE8이 아닐 경우 오류출력. 이미 압축해제 된 영상이 이 기능을 수행하려고 하는 경우 알려줄 것.

사용 목적과 기능:

만약 영상이 RLE8로 압축되었을 경우 압축해제를 하여 압축 해제된 영상을 제공한다.

2.BMP영상의 밝기 반전

실행 및 제약조건:

X

사용 목적과 기능:

영상의 밝기를 반전하여 사용자에게 밝기가 반전된 영상을 제공한다.

3.사각형 그리기

실행 및 제약조건:

사용자가 영상에서 사각형을 그리고 싶은 좌표 값을 입력 받아야함. 사각형의 좌표를 제공 받을 때 영상의 크기를 벗어나게 받으면 안됨.

사용 목적과 기능:

사용자에게 좌표를 입력 받아 사각형을 그려주고 사각형이 그려져 있는 영상을 제공한다.

4.영상의 밝기를 더 밝게 하거나 어둡게 한 영상제공 (추가기능)

실행 및 제약조건:

사용자가 밝기를 조정하고 정도의 값을 입력 받아야함. 사용자가 밝기를 선택할 때 8비트의 범위를 벗어난다면 0 또는 255로 설정하여 최소한도 또는 최대한도를 맞춰준다.

사용 목적과 기능:

사용자에게 영상의 밝기를 조절할 값을 입력 받은 후 해당 값을 이용하여 밝기를 조절한 영상 제공

5.상하반전(추가기능)

실행 및 제약조건:

X

사용 목적과 기능:

영상을 상하 반전하여 상하 반전된 영상을 제공한다.

6.좌우반전(추가기능)

실행 및 제약조건:

X

사용 목적과 기능:

영상을 좌우 반전하여 사용자에게 좌우 반전된 영상을 제공한다.

7.BMP파일 저장

실행 및 제약조건:

사용자가 지정한 이름으로 영상을 저장하여야 하기 때문에 이름을 입력 받아야함.

사용 목적과 기능:

사용자가 지정한 이름을 입력 받은 뒤 해당 값을 이용하여 사용자가 지정한 이름으로 영상처리한 BMP영상 저장

3-2. 모듈의 입출력 및 실행예시 명세

1. RLE 압축파일 압축 해제

INPUT: RLE8형식으로 압축 되어있는 8비트 비트맵 영상
OUTPUT: 사용자가 지정한 이름으로 바뀌고 압축이 풀린 비트맵 영상

'압축해제 기능 수행'
'압축해제하고 싶은 파일을 입력하세요>>'
INPUT: RLE8형식으로 압축 되어있는 8비트 비트맵 영상
#만약 RLE8형식으로 압축된 8비트 비트맵이 아닐 경우 오류 출력 후 다시 입력 받음.
'압축해제하고 싶은 파일을 확인했습니다 압축해제기능을 수행하겠습니다'
OUTPUT: 압축이 풀린 비트맵 영상

2. BMP영상의 밝기 반전

INPUT: 압축이 되어있지 않은 8비트 비트맵 영상
OUTPUT: 밝기가 반전된 8비트 비트맵 영상

'밝기 반전 기능 수행'
'밝기 반전하고 싶은 파일을 입력하세요>>'
INPUT: 압축 되어있지 않은 8비트 비트맵 영상
#만약 RLE8형식으로 압축된 8비트 비트맵일 경우 오류 출력 후 다시 사용자에게 알려줌.
'밝기 반전하고 싶은 파일을 확인했습니다 밝기 반전기능을 수행하겠습니다'
OUTPUT: 밝기가 반전된 8비트 비트맵 영상

3. 사각형 그리기

INPUT: 압축이 되어있지 않은 8비트 비트맵 영상, 사각형을 그릴 좌표들
OUTPUT: 사각형이 그려진 8비트 비트맵 영상

'사각형 그리기 기능 수행'
'사각형을 그리고 싶은 파일을 입력하세요>>'
INPUT: 압축 되어있지 않은 8비트 비트맵 영상
#만약 RLE8형식으로 압축된 8비트 비트맵일 경우 오류 출력 후 다시 사용자에게 알려줌.
'입력된 영상의 크기는 가로:?? 세로:?? 의 크기를 가지고 있습니다'
'사각형을 그릴 좌표들을 입력해주세요 영상의 크기를 벗어나면 안됩니다.>>'
#만약 사용자가 입력한 좌표의 크기가 영상의 크기를 벗어날 경우 오류 출력 후 다시 사용자에게 알려줌.
'사각형을 그리고 싶은 파일을 확인했습니다 사각형 그리기 기능을 수행하겠습니다'
OUTPUT: 사각형이 그려진 8비트 비트맵 영상

4. 영상의 밝기를 더 밝게하거나 어둡게한 영상제공 (추가기능)

INPUT: 압축이 되어있지 않은 8비트 비트맵 영상, 밝기를 조절할 값
OUTPUT: 사용자가 지정한 밝기가 조절된 8비트 비트맵 영상

'밝기 조절 기능 수행'
'밝기를 조절하고 싶은 파일을 입력하세요>>'
INPUT: 압축 되어있지 않은 8비트 비트맵 영상
#만약 RLE8형식으로 압축된 8비트 비트맵일 경우 오류 출력 후 다시 사용자에게 알려줌.
'밝기를 조절하고 싶은 값을 입력해주세요 -:어둡게 +:밝게 >>'
#만약 사용자가 입력한 밝기 조절 값을 더했는데 밝기 값이 8비트으 범위를 초과한다면 최대값이거나 최소값인 255, 0 으로 설정해줌.
'밝기를 조절 할 파일을 확인했습니다. 밝기 조절 기능을 수행하겠습니다'
OUTPUT: 사용자가 지정한 밝기가 조절된 8비트 비트맵 영상

5. 상하반전(추가기능)

INPUT: 압축이 되어있지 않은 8비트 비트맵 영상
OUTPUT: 상하 반전된 8비트 비트맵 영상

'상하 반전 기능 수행'
'상하 반전하고 싶은 파일을 입력하세요>>'
INPUT: 압축 되어있지 않은 8비트 비트맵 영상
#만약 RLE8형식으로 압축된 8비트 비트맵일 경우 오류 출력 후 다시 사용자에게 알려줌.
'상하 반전하고 싶은 파일을 확인했습니다 상하 반전기능을 수행하겠습니다'
OUTPUT: 상하 반전된 8비트 비트맵 영상

6. 좌우반전(추가기능)

INPUT: 압축이 되어있지 않은 8비트 비트맵 영상
OUTPUT: 좌우 반전된 8비트 비트맵 영상

'좌우 반전 기능 수행'
'좌우 반전하고 싶은 파일을 입력하세요>>'
INPUT: 압축 되어있지 않은 8비트 비트맵 영상
#만약 RLE8형식으로 압축된 8비트 비트맵일 경우 오류 출력 후 다시 사용자에게 알려줌.
'좌우 반전하고 싶은 파일을 확인했습니다 좌우 반전기능을 수행하겠습니다'
OUTPUT: 상하 반전된 8비트 비트맵 영상

7. BMP파일 저장

INPUT: 압축이 되어있지 않은 8비트 비트맵 영상, 사용자가 변경하고 싶은 이름
OUTPUT: 사용자가 지정한 이름으로 바뀐 영상

'영상 저장 기능 수행'
'사용자가 원하는 이름을 입력해주세요>>'
INPUT: 사용자가 변경하고 싶은 이름
'사용자가 변경하고 싶은 이름을 확인했습니다. 영상 저장을 수행하겠습니다.'
OUTPUT: 사용자가 지정한 이름으로 바뀐 영상

3-3. 모듈의 알고리즘 (seudo code) 및 중요 변수 기술

요구사항 분석, 시스템 설계를 거쳐 모듈의 정보들을 추합해 직접 코드로 옮겨지는 작업이 수행된다. 이 작업을 보다 수월하게 가이드 라인을 제공하는 seudo code를 기술한다. 구현과 가장 밀접한 관계를 지녔으므로 설계 보고서 중 가장 빈번하게 보는 파트가 아닐까 싶다. 추가적으로 7개 모듈 중 마지막인 "BMP 파일 저장"은 C언어의 라이브러리 fwrite() 함수를 쓰는 것이 주를 이루므로 이는 기술하지 않는다.

window.h 헤더파일을 이용해서 알고리즘 구현에 활용할 멤버함수를 알고리즘의 이해를 돕기 위해 기술 하겠다.

BITMAPFILEHEADER	BITMAPINFOHEADER

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

1.RLE 압축파일 압축 해제

INPUT: RLE8형식으로 압축 되어있는 8비트 비트맵 영상 (rle8_bitmap_file)
OUTPUT: 사용자가 지정한 이름으로 바뀌고 압축이 풀린 비트맵 영상

1. rle8_bitmap_data ← 압축된 8비트 비트맵 영상의 영상데이터

```
2. user_bit_map ← 압축이 풀린 비트맵 영상을 담을 변수
3. int i ← 첫번째 인덱스  j ← 두번째 인덱스
4. while (압축된 비트맵의 영상데이터가 끝날 때 까지) {
5.      if(rle8_bitmap_data[i]==0) {                                    //absolute mode
6.          if(rle8_bitmap_data[j]>=3) {
7.              len ← rle8_bitmap_data[j]
8.              while(len의 크기만큼 반복)
9.                  user_bit_map ← rle8_bitmap_data
10.             if(len%2==1)
11.                 해당 값은 넘어가기
12.         elseif(rle8_bitmap_data[j]==0) {                            //마커(marker)
13.             while(현재 가로의 길이가 부족하다면)
14.                 user_bit_map ← 255
15.             다음라인으로넘어감.
16.         elseif(rle8_bitmap_data[j]==1) {                            //마커(marker)
17.             while(현재 가로의 길이가 부족하다면)
18.                 user_bit_map ← 255
19.             break
20.         elseif(rle8_bitmap_data[j]==2) {                            //마커(marker)
21.             x= rle8_bitmap_data[j+1] , y= rle8_bitmap_data[j+2]
22.              while(현재위치가 x+가로길이*y의 값과 같을때까지)
23.                 user_bit_map ← 255
24.     elseif(rle8_bitmap_data[i]!=0) {                                //Encoded mode
25.         len ← rle8_bitmap_data[i]
26.         data ← rle8_bitmap_data[j]
27.         while(len의 크기만큼 반복)
28.             user_bit_map ← rle8_bitmap_data
```

- 각 모드는 슈도코드에 주석으로 표현해 놓았다. 이해를 돕기 위해 해당 모드가 무슨 모드인지 설명하겠다.

i, j ← 영상데이터의 첫번째와 두번째 인덱스를 지정할 변수이다. 해당 인덱스를 이용하여 압축해제를 수행한다.

absolute mode는 횟수를 나타내는 첫 번째 바이트가 0이고 두 번째 바이트가 3이상(<256) 값이다. 이 모드는 값이 계속해서 변하는 경우에 사용하고 최소 길이는 3, 최대 길이는 255이다. 이 두 번째 값은 뒤에 따르는 값의 길이이고, 이 값만큼 부호화 되지 않은 값이 나타낸다. 만약 길이가 홀수이면 부호화 코드에는 0을 추가한다. 그래서 absolute mode를 처리하여 줄 때에는 두번째 바이트의 크기만큼 그 뒤에 나오는 데이터를 저장하여 주고 만약 두번째 바이트의 크기가 홀 수수일 때 그 자리에 위치하는 데이터는 건너 뛴다.

마커(marker)는 횟수를 나타내는 첫 번째 바이트가 0이고 두 번째 바이트가 0, 1, 2 중의 한 가지 값이다.

▪ rle8_bitmap_data[i, j] ← (00 00))

End of Line (00 00) 이 마커는 다음 코드는 새로운 라인을 위한 것이며 현재 라인을 위한 더 이상은 정보는 없음을 의미한다. 만약 영상의 가로길이보다 픽셀의 개수가 적으면 디폴트 배경색으로 저장한다.

▪ rle8_bitmap_data[i, j] ← (00 01)

End of Bitmap (00 01) 이 마커는 부호화 된 데이터의 마지막을 나타낸다. 나머지 영역은 디폴트 배경색으로 저장한다.

▪ rle8_bitmap_data[i, j] ← (00 02)

Delta (00 02) 이 마커는 현재 위치로부터 상대적으로 이동할 위치를 나타낸다. 연속하는 첫 번 째 바이트는 수평방향의 오프셋, 두 번째 바이트는 수직방향의 오프셋을 의미한다. 단, 이 값들은 부호 없는 정수로 처리한다. 따라서 다음에 나타나는 코드는 새로운 위치의 픽셀을 위한 것이다. 사이의 값은 디폴트 배경색으로 채운다.

참고로 여기서 나오는 디폴트 배경색은 문제에서 제시한대로 흰색(255)이다.

Encoded mode는 일반적인 RLE 압축 방식으로 동작한다. 첫 번째 바이트는 횟수, 두 번째 바이트는 반복되는 값을 나타낸다. 그래서 Encoded mode라면 첫번째 바이트의 크기만큼 두번째 바이트의 값을 반복해준다.

-RLE8형식으로 압축된 8비트 비트맵이 아닐 경우를 판단해주기 위해서는 비트맵정보헤더의 biCompression변수를 이용한다. 해당 변수가 1이라면 RLE8형식으로 압축된 것이다.

-8비트의 영상인 것을 판단해주기 위해서 비트맵정보헤더의 biBitcount 변수를 이용한다. 해당 값이 8이라면 8비트영상이다.

-비트맵영상의 가로의 길이와 세로의 길이는 비트맵정보헤더의 biWidth와 biHeight 값을 활용할 것이다.

2.BMP영상의 밝기 반전

INPUT: 압축이 되어있지 않은 8비트 비트맵 영상
OUTPUT: 밝기가 반전된 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 ← 밝기가 반전된 비트맵 영상을 담을 변수
4. bit_map_index ← i * (width) +j
5. for i←0 to i<영상의 세로길이 {
6.      for j←0 to j<영상의 가로길이 {
7.            user_bit_map[bit_map_index] ← 255- bitmap_data[bit_map_index] } }
```

- **i * (width) +j** 라고 표현한 이유는 높이 x 세로인덱스 + 가로인덱스의 값이 2차원 배열을 1차원 배열의 인덱스값으로 표현한 수이기 때문이다.

- user_bit_map[bit_map_index]=255- user_bit_map[bit_map_index]

=> 해당 코드는 255라는 8비트에서 가장 큰 컬러값을 이용해서 색을 반전 시키는 것이다.

3.사각형 그리기

INPUT: 압축이 되어있지 않은 8 비트 비트맵 영상, 사각형을 그릴 좌표들
OUTPUT: 사각형이 그려진 8 비트 비트맵 영상

1. int i ← 사각형의 세로길이를 담을 변수 j ← 사각형의의 가로길이를 담을 변수
2. bitmap_data ← input 값으로 받은 비트맵 영상의 영상데이터를 담은 변수
3. user_bit_map ← 사각형이 그려진 비트맵 영상을 담을 변수
4. bit_map_index ← i * (width) +j
5. user_select_x1, user_select_x2, user_select_y1, user_select_y2 ← 사용자가 지정한 사각형 위치
6. for i←user_select_y1 to i<user_select_y2 {
7. for j←user_select_x1 to j< user_select_x2 {
8. bitmap_data [bit_map_index] ← 200(GRAY) } }
9, user_bit_map ← bitmap_data

- bitmap_data [bit_map_index]= GRAY

=> 해당 코드는 지정한 위치를 회색으로 색을 변경해 주어 사각형을 그리는 것이다.

- for i←user_select_y1 to i<user_select_y2 { //line 6-8
  for j←user_select_x1 to j< user_select_x2 {
  bitmap_data [bit_map_index] ← 200(GRAY) } }

=> 유저가 입력한 사각형의 좌표값을 토대로 데이터 인덱스 값에 GRAY 값을 입력한다.

4.영상의 밝기를 더 밝게하거나 어둡게한 영상제공 (추가기능)

INPUT: 압축이 되어있지 않은 8비트 비트맵 영상, 밝기를 조절할 값
OUTPUT: 사용자가 지정한 밝기가 조절된 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 ← 사각형이 그려진 비트맵 영상을 담을 변수
4. bit_map_index ← i * (width) +j
5. bright_control ← 사용자가 지정한 밝기 조절값 (어둡게하거나 밝게하거나)
6. for i←0 to i<영상의 세로길이 {

```
7.      for j←0 to j<영상의 가로길이 {
8.          if(bitmap_data [bit_map_index]+ bright_control <0)
9.              user_bit_map[bit_map_index] ← 0
10.         elseif(bitmap_data [bit_map_index]+ bright_control >255)
11.             user_bit_map[bit_map_index] ← 255
11.         else
12.             user_bit_map[bit_map_index] ← user_bit_map[bit_map_index] + bright_control }
```

- If(bitmap_data [bit_map_index]+ bright_control <0) //line 8 - 12
  user_bit_map[bit_map_index] ← 0
  elseif(bitmap_data [bit_map_index]+ bright_control >255)
  user_bit_map[bit_map_index] ← 255
  else
  user_bit_map[bit_map_index] ← user_bit_map[bit_map_index] + bright_control

=> 해당 코드는 사용자가 지정한 밝기조절 값과 현재 영상에서 밝기의 값의 합이 8비트의 범위를 넘는다면 0또는 255로 처리해주고 만약 범위를 넘지 않는다면 해당 밝기를 넣어주는 코드이다.

5.상하반전(추가기능)

INPUT: 압축이 되어있지 않은 8비트 비트맵 영상
OUTPUT: 상하 반전된 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 ← 상하가 반전된 비트맵 영상을 담을 변수
4. bit_map_index ← i * (width) +j
5. for i←0 to i<영상의 세로길이 {
6.      for j←0 to j<영상의 가로길이 {
7.          user_bit_map[bit_map_index]  ← bitmap_data[ ((height - 1 - i) * width) +j ]
```

- user_bit_map[bit_map_index] ← bitmap_data[((height - 1 - i) * width) +j]

=> 해당코드는 상하반전을 위해서 반대로 저장해주는 코드이다.

6.좌우반전(추가기능)
INPUT: 압축이 되어있지 않은 8비트 비트맵 영상
OUTPUT: 좌우반전된 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 ← 좌우가 반전된 비트맵 영상을 담을 변수
4. bit_map_index ← i * (width) +j
5. for i←0 to i<영상의 세로길이 {
6. for j←0 to j<영상의 가로길이 {
7. user_bit_map[bit_map_index] ← bitmap_data[((i * width) + width -1 - j]

- **user_bit_map[bit_map_index] ← bitmap_data[((i * width) + width -1 - j]**

=> 해당 코드는 좌우반전을 위해서 반대로 저장해주는 코드이다.

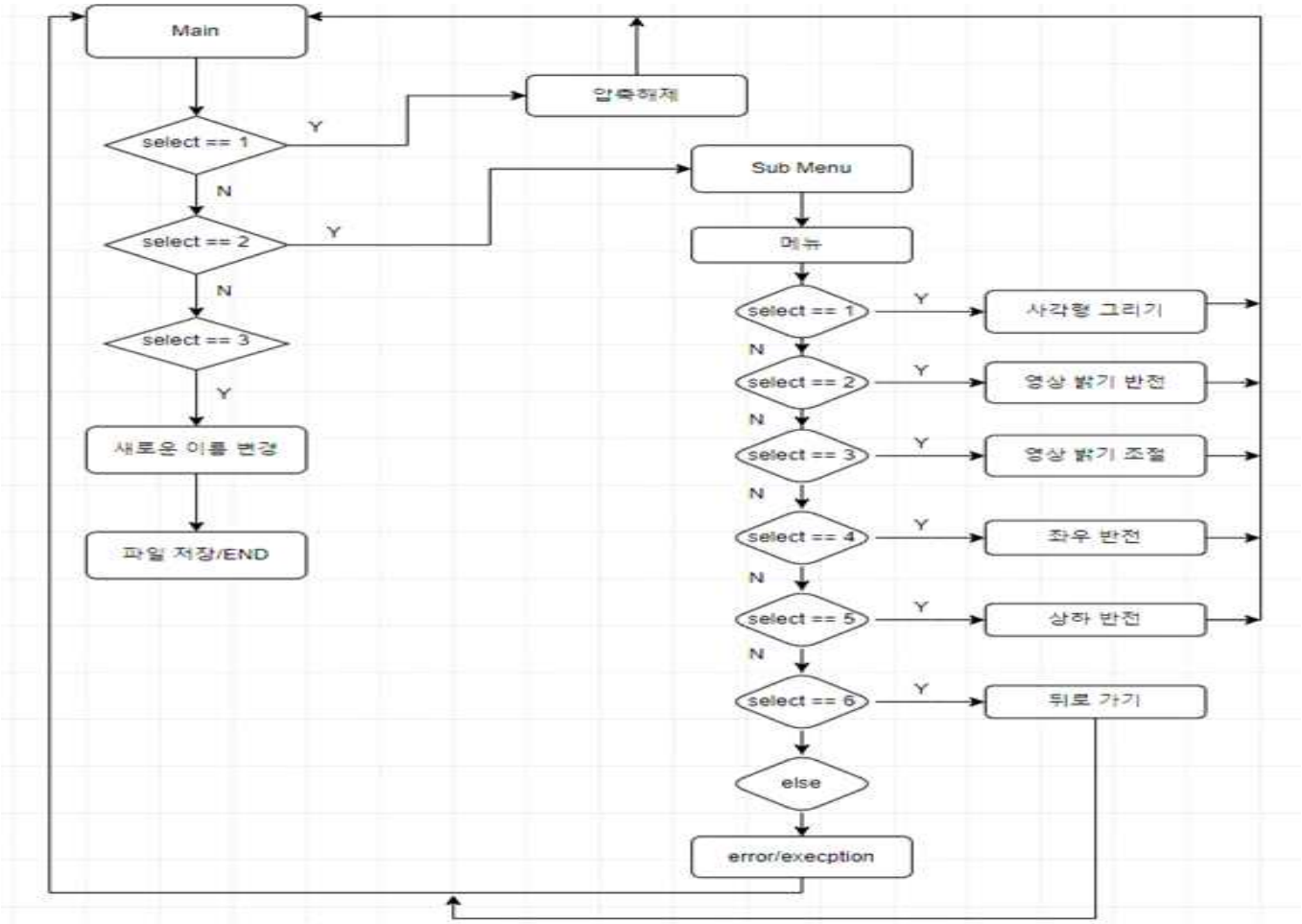
3-4. 정리

지금까지 프로그램 설계, 즉 시스템 설계에서 명시한 7개의 모듈들의 입출력과 기능, 제약조건, 주요변수 및 슈도코드를 명세하고 이를 설명하였다. 가장 핵심적인 부분인 만큼 정교하고 자세한 설계가 필요하여 가장 많은 시간을 투자하지 않았나 싶다. 다음파트에선 각 모듈들 사이의 정보처리, 신호를 주고받는 접점, 즉 인터페이스를 기술하려고 한다.

4. 인터페이스 설계

프로그램 실행 시 사용자 측면에서 볼 수 있는 UI와 인터페이스의 실행 흐름을 살필 수 있는 인터페이스 구조도, 사용자와 상호작용 하는 과정에서 사용되는 모듈에 대해 살펴본다.

4-1. 인터페이스의 전체적인 구조 도식화 및 설명



메인메뉴에는 압축해제, 파일수정, 다른 이름으로 저장을 해주는 3가지 기능이 있다.

메인메뉴에서 압축해제를 선택할 경우, RLE 파일을 입력하여 해당파일 압축해제를 진행한다. 이후 메인메뉴로 돌아간다.
파일 수정을 선택할 경우 프로그램의 서브메뉴로 들어가게 되고, 사각형 그리기, 영상밝기 반전,

영상 밝기 조절, 좌우반전, 상하반전의 총 5가지의 파일 수정(영상처리)이 가능하다. 이전 메뉴로 돌아갈 수 있는 기능도 제공한다. 모든 파일 수정이 끝날 때는 다시 메인메뉴로 돌아가도록 설계가 되어 있다.

메인메뉴에서 다른 이름으로 저장하는 부분은 새로운 이름을 입력받아 해당 파일을 새로운 이름으로 저장하게 된다.

위 전 과정에서 에러가 발생할 경우 예외처리를 통해 프로그램이 정상 실행이 가능하도록 설계한다.

4-2. 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

4-3. 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 관한 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
SaveAsDiff()	영상처리가 완료된 파일을 새로운 이름으로 저장하는 기능 수행

- UI 관점에서 본 내부모듈 명세

5. 테스트 및 유지보수 방법 설계

소프트웨어를 제작하고 보수하는데 있어 테스트는 가히 필수적이라고 할 수 있다. 그에 따라 소프트웨어나 프로그램의 특성에 따라 많은 테스트 방법이 존재하지만, 우리가 채택한 테스트 방법은 “단위테스트”이다.

5-1. 단위테스트

단위 테스트란 소프트웨어 개발 후 테스트 가능한 최소 단위 기준으로 결함을 찾고, 기능을 검증하는 테스트 활동을 말하며 컴퓨터 프로그래밍에서 소스코드의 특정 모듈이 의도된 대로 정확히 작동하는지 검증하는 절차를 의미한다. 여러 테스트 중에서 단위테스트를 선택한 이유는 다음곽 같다.

5-2선정이유

1. 우리 프로그램의 기능은 각각의 메소드로 독립적으로 구성될 수 있다.
2. 이에 따라 각 메소드를 고립시켜 동작이 원활하게 진행되는지 확인할 수 있다.
3. 팀원들 간의 협업에서 업무분담(테스트)도 원활하게 이루어질 수 있다.

5-3 테스트 방법

1. 각각의 모듈에 대해서 RLE로 압축되지 않은 bitmap파일을 입력으로 사용한다.
2. 테스트하고자 하는 기능을 별도의 파일로 수행한다.
3. 정상적으로 동작한다면 프로그램의 코드에 합쳐 작동을 확인한다.
4. 제약 조건을 추가하여 기능을 완성한다.

6. 추진 일정 및 역할 분담

6-1. 추진 일정

(수행 기간 : 2022년 10월 12일 ~ 2022년 11월 16일)						
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

모든 개발내용은 팀원 함께 다룰 예정이나 중점적으로 담당할 사람을 표에 기술하였다
담당할 팀원은 프로젝트를 진행하며 추진 일정 및 담당할 포지션은 유동적으로 조정이 가능하며
보고서 바다 각 역할 분담 내용을 다시 기술할 예정이다.

6-2. 역할 분담

설계 보고서의 역할분담은 아래와 같이 담당하였다.

안현진	김현진	송제용	정지수
요구사항 분석 및 작성 - Usecase 분석 - Class Diagram 제작 - FlowChart 제작	시스템 설계 분석 및 작성 -프로그램 전체 기능 및 목표 명세 -실행 및 시스템 설계 제약조건	프로그램 설계분석 및 작성 - 모듈의 입출력 명세 - 제약조건 명세 - 알고리즘 설계	인터페이스 분석 및 작성 - Interface Flowchart 및 UI 제작 - 내부 모듈 입출력 명세 - PPT 제작