

# Report



교과목 :	공학 수학
학 과 :	컴퓨터공학과
학 번 :	20190850
이 름 :	이 수 진
제출일 :	2020-09-25

## 1. 문제

### 1.1 문제 내용

- 사용자가 PPM 파일을 입력하여 기능을 선택해 편집한 후, 결과를 적절한 이름의 파일에 저장하는 프로그램을 설계 및 구현한다.
- 기본 문제 : 색상 반전, 상하/좌우 반전, 영상 회전, 채널 추출
- 확장 문제 : GUI 구현, 추가 영상 처리 메뉴

### 1.2 주의 사항 및 고려할 점

- 영상 처리 관련 라이브러리 사용이 불가하다. 즉 직접 모든 기능을 코딩해야 한다.

### 1.3 문제 해결 범위(자신이 수행한 범위)

- 파일 열기, 저장, 기본 문제(색상 반전, 상하/좌우 반전/영상 회전/채널 추출), 확장 문제로 그레이 스케일 변환 기능을 추가하였다.

## 2. 기능 및 요구 사항 분석

### 2.1 입출력 정의

출력: 읽기 또는 편집할 PPM 파일 입력하세요



입력: 파일 위치



출력: 기능 선택 1. 반전 2. 회전 3. 채널 추출 4. 저장 5. 종료



입력: 선택할 기능 번호



출력: 이미지 적용



입력: 저장 또는 종료



출력: 파일을 저장 or 프로그램을 종료합니다.

### 2.2 핵심 기능과 그들 간의 관계

- 핵심 기능은 파일 불러오기(읽기), 파일 저장, 색상 반전, 상하/좌우 반전, 영상 회전, 채널 추출, 그레이 스케일 변환이다. 사용자로부터 파일을 읽은 후(파일 불러오기), 원하는 기능을 수행한다. (색상 반전, 상하/좌우 반전, 영상 회전, 채널 추출) 그 후 편집한 파일을 다른 이름으로 저장(파일 저장)한다.

### 2.3 요구 사항 분석

#### 2.3.1 기능적 요구사항

- 색상 반전
- 상하/좌우 반전
- 영상 회전 : 사용자가 원하는 각도로 영상을 회전한다.
- 채널 추출 : 단일 추출, 멀티 추출 중 선택

### 2.3.2 비기능적 요구사항

- 기능의 중복 선택이 가능하다.
- 편집한 후 저장한 파일의 이름은 사용자가 입력한다.
- 저장한 후에도 이어서 편집, 저장이 가능하다.
- 실행 환경 : C++ 콘솔 창에서 수행

### 2.3.3 대체흐름

- 입력 오류 : 오류 처리와 적절한 메시지 출력 후 전 단계로 돌아간다.

ex. 불러온 파일이 비어있는 경우, 또는 PPM 파일이 아닌 경우

PPM 파일의 최대 밝기가 255가 아닌 경우

번호 입력의 오류

: 이 때 메뉴와 번호 입력을 다시 출력한다.

메인 메뉴에서 오류 : 메인 메뉴 다시 출력

편집 메뉴에서 오류 : 편집 메뉴 다시 출력

편집-기능에서 오류 : 편집 메뉴 출력

영상 회전각 입력에서 오류 : 회전각 입력 메시지 출력

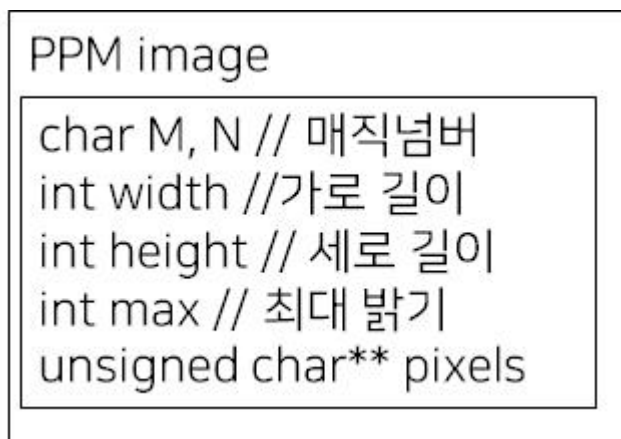
파일을 불러오기 전에 편집 및 저장을 시도하는 경우

회전각이 360보다 크거나 - 360보다 작은 경우

## 3. 설 계

### 3.1 자료 구조 또는 클래스 설계

#### 3.1.1 구조체 - PPM 이미지



PPM 이미지 객체는 다음과 같은 요소를 가진다.

- 매직 넘버는 파일의 유형이 무엇이며 데이터가 어떻게 저장되어있는가를 정의한다. PPM 이미지의 경우 매직 넘버는 P6이다.
- 이미지의 가로, 세로 길이 - 픽셀의 내용을 읽고자 할 때 쓰인다.
- 밝기는 0~255 사이의 값을 가지는데, 이미지 객체에서 최대 밝기는 255로 정의한다.
- 픽셀값을 담기 위한 이중 포인터 변수를 포함한다. 픽셀값의 타입은 unsigned char로 정의한다. unsigned로 해주는 이유는 unsigned의 값이 0~255로 제한되기 때문이다.

### 3.2 알고리즘 설계(pseudo code 또는 flowchart)

#### 3.2.1 파일 읽기

```
Algorithm Read_PPM(Filename, img)
Input : PPM 파일 이름 Filename과 파일을 읽고 저장할 PPM 이미지 객체 img
Output : 성공과 실패 여부(1, 0)
deletePPM(img)
if (filename == NULL)
    Error()
    return 0
read file
if (file == NULL)
    Error()
    return 0
img-M, img-N <- file M, N
if (M != 'P' or N != '6')
    Error()
    return 0
img-width, height, max <- file width, height, max
img-pixels <- file pixels
file close
return 1
```

#### 3.2.2. 편집

##### 3.2.2.1 색상 반전

```
Algorithm color_Invert(img)
Input : PPM 이미지 객체 img
Output : 없음
unsigned char temp
for (int i=0, i<img->height;i++)
    for (int j=0; j<img->width*3;j++)
    {
        temp <- 255 - img->pixels[i][j]
        img->pixels[i][j] <- temp
    }
return
}
```

### 3.2.2.2 좌우/상하 반전

```
Algorithm Filp(img, num)
Input : PPM 이미지 객체 img, 사용자 입력 num
Output : 없음
PPMimage tmp_img = img
int k;
if (num == 1) // 좌우 반전
    for (int i=0;i<img->height;i++)
        for (int j=0; j<img->width+3;j+=3)
        {
            k <- (img->width * 3) - j -3
            img->pixels[i][k] <- tmp_img.pixels[i][j]
            img->pixels[i][k+1] <- tmp_img.pixels[i][j+1]
            img->pixels[i][k+2] <- tmp_img.pixels[i][j+2]
        }
else if (num == 2) // 상하 반전
    for (int i=0;i<img->height;i++)
        k = img->height - i - 1
        for (int j=0; j<img->width*3;j++)
        {
            img->pixels[k][j] <- tmp_img.pixels[i][j]
        }
else
    ErrorPrint()
    return
deletePPM(tmp_img)
return
}
```

### 3.2.2.3 회전

```
Algorithm Rotion(img, degree)
Input : PPM 이미지 객체 img, 사용자 입력 각도
Output : 없음
center_x <- img->width/2.0
center_y <- img->height/2.0
int new_x, new_y
int a
unsigned char **R, R_t, G, G_t, B, B_t
double seta <- degree / 180.0*PI
```

```

R, G, B <- 동적할당, 0으로 초기화
R_t, G_t, B_t <- 동적할당, pixels의 R,G,B값
for (int i=0;i<img->height;i++)
    for (int j=0; j<img->width;j++)
    {
        new_x <- (i-center_y)*sin(seta)+(j-center_x)*cos(seta)+center_x
        new_y <- (i-center_y)*cos(seta)-(j-center_x)*sin(seta)+center_y
        a <- 0
        if ((new_x >= 0 && new_x < img->width && new_y >=0 &&
new_y<img->height)
            a <- R_t[new_y][new_x]
            R[i][j] = a
        }
    }
for (int i=0;i<img->height;i++)
    for (int j=0; j<img->width;j++)
    {
        new_x <- (i-center_y)*sin(seta)+(j-center_x)*cos(seta)+center_x
        new_y <- (i-center_y)*cos(seta)-(j-center_x)*sin(seta)+center_y
        a <- 0
        if ((new_x >= 0 && new_x < img->width && new_y >=0 &&
new_y<img->height)
            a <- G_t[new_y][new_x]
            G[i][j] = a
        }
    }
for (int i=0;i<img->height;i++)
    for (int j=0; j<img->width;j++)
    {
        new_x <- (i-center_y)*sin(seta)+(j-center_x)*cos(seta)+center_x
        new_y <- (i-center_y)*cos(seta)-(j-center_x)*sin(seta)+center_y
        a <- 0
        if ((new_x >= 0 && new_x < img->width && new_y >=0 &&
new_y<img->height)
            a <- B_t[new_y][new_x]
            B[i][j] <- a
        }
    }
for (int i=0;i<img->height;i++)
    for (int j=0; j<img->width;j++)
    {img->pixels[i][3*j] <- R[i][j]
      img->pixels[i][3*j+1] <- G[i][j]
      img->pixels[i][3*j+2] <- B[i][j]
    }
R,G,B,R_t,G_t,B_t 동적할당 해제
return;

```

#### 3.2.2.4 채널 추출

Algorithm channel\_RGB(img, num)

Input : PPM 이미지 객체 img, 사용자 입력 num

Output : 없음

```
if (num == 1) // 빨간색 추출
    for (int i=0;i<img->height;i++)
        for (int j=0; j<img->width*3;j+=3){
            img->pixels[i][j+1] <- 0
            img->pixels[i][j+2] <- 0
        }
if (num == 2) // 초록색 추출
    for (int i=0;i<img->height;i++)
        for (int j=0; j<img->width*3;j+=3){
            img->pixels[i][j] <- 0
            img->pixels[i][j+2] <- 0
        }
if (num == 3) // 파란색 추출
    for (int i=0;i<img->height;i++)
        for (int j=0; j<img->width*3;j+=3){
            img->pixels[i][j] = 0
            img->pixels[i][j+1] = 0
        }
if (num == 4) // 빨간색 + 초록색 추출
    for (int i=0;i<img->height;i++)
        for (int j=0; j<img->width*3;j+=3){
            img->pixels[i][j+2] <- 0
        }
}
if (num == 5) // 초록색 + 파란색 추출
    for (int i=0;i<img->height;i++)
        for (int j=0; j<img->width*3;j+=3){
            img->pixels[i][j] <- 0
        }
if (num == 6) // 빨간색 + 파란색 추출
    for (int i=0;i<img->height;i++)
        for (int j=0; j<img->width*3;j+=3){
            img->pixels[i][j+1] <- 0
        }
else
    ErrorPrint()
return
```

#### 3.2.2.5 그레이 스케일 변환

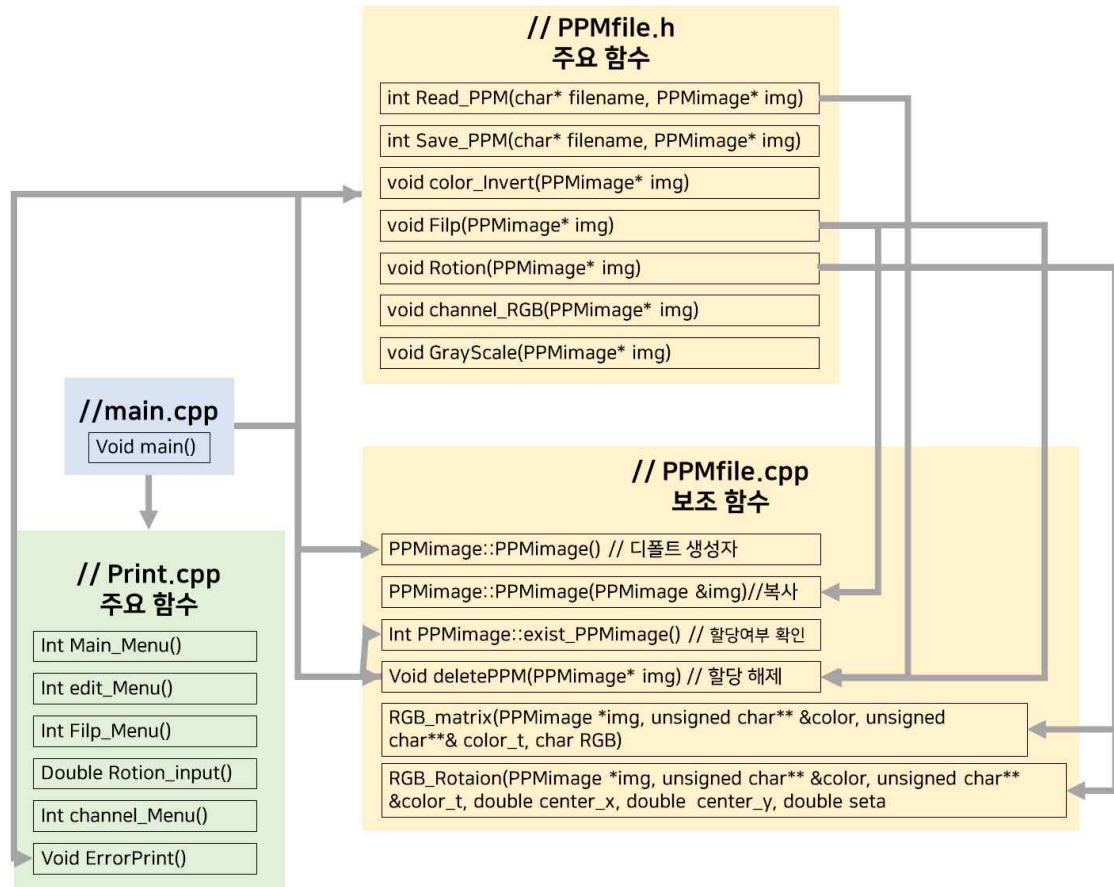
```
Algorithm Grayscale(img)
Input : PPM 이미지 객체 img
Output : 없음
unsigned char Gray
for (int i=0;i<img->height;i++)
    for (int j=0; j<img->width*3;j+=3)
    {
        Gray <- (img->pixels[i][j] + img->pixels[i][j+1] + img->pixels[i][j+2])
        img->pixels[i][j] <- Gray
        img->pixels[i][j+1] <- Gray
        img->pixels[i][j+2] <- Gray
    }
return
```

#### 3.2.3 파일 저장

```
Algorithm Save_PPM(filename, img)
Input : 저장할 파일 이름 filename, PPM 이미지 객체 img
Output : 성공과 실패 여부(0,1)
file write
if (file == NULL)
    ErrorPrint()
    return 0
file <- P6, width, height, pixels(char)
file close
return 1
```



### 3.3 함수들 간의 호출 관계 또는 프로그램 구조(구현된 내용대로 그릴 것)



## 4. 구 현

### 4.1 자료 구조 또는 클래스 코드

#### 4.1.1 PPM 이미지 구조체

```

typedef struct PPMimage{
    char M, N; // 매직넘버
    int width; // 가로 길이
    int height; // 세로 길이
    int max; // 최대 밝기
    unsigned char** pixels; // 픽셀값
}PPMimage;
    
```

- PPM 파일에 저장된 정보를 받는 PPM 이미지 객체이다.
- char M, N은 매직넘버로 PPM 이미지는 P6이어야한다.
- int width, height은 PPM 이미지의 가로 길이와 세로 길이이다.
- int max는 최대 밝기로, 255를 초과할 수 없다.
- unsigned char\*\* pixels;은 픽셀값을 저장할 이중 포인터이다. 타입이 unsigned char인 이유는 R, G, B 값 모두 0~255를 가지기 때문이다.

### 4.2 핵심 알고리즘의 코드 부분

#### 4.2.1 파일 읽기

- (1) 먼저 검사 후, 입력받은 img의 픽셀에 할당된 메모리를 해제한다.

- (2) 입력받은 파일 이름이 비어있는지 확인한다.
- (3) 파일 포인터로 파일을 바이너리 타입으로 연다. 이때 파일 포인터가 비어있는지 확인한다.
- (4) 파일을 읽고, 매직 넘버, 너비, 높이, 최대 밝기 순으로 img 객체에 넣는다. 이때 매직 넘버가 P6 인지, 최대 밝기가 255를 넘지 않는지를 확인한다.
- (5) img의 pixels 메모리를 먼저 할당해준 후, 파일을 읽어 픽셀값을 집어넣는다.
- (6) 파일을 닫는다.

#### 4.2.2 파일 저장

- (1) 파일 포인터로 입력받은 파일명을 아스키코드로 쓴다. 이때 파일 포인터가 NULL인지 확인한다.
- (2) P6, 가로, 세로 길이, 최대 밝기를 순서대로 파일에 쓴다.
- (3) 파일에 픽셀값을 char 형으로 넣는다.
- (4) 파일을 닫는다.

#### 4.2.3 편집

##### 4.2.3.1 색상 반전

- (1) 타입이 unsigned char인 temp 변수 하나를 생성한다.
- (2) 이중 for문으로 객체를 읽어 temp에 pixels에서 255를 뺀 값을 넣는다.
- (3) 기존 pixels 값에 temp를 넣는다.

```
for (int i=0;i<img->height;i++)
    for (int j=0;j<img->width*3;j++)
    {
        temp = 255-img->pixels[i][j];
        img->pixels[i][j] = temp;
    }
```

##### 4.2.3.2 좌우/상하 반전

- (1) int 타입의 변수 k값을 생성한다. 그리고 복사 생성자를 이용하여 img를 tmp\_img에 복사한다.
- (2) 사용자의 입력에 따라 case를 나눈다.
- (3) 좌우 반전인 경우, y좌표는 그대로 두고 y축을 기준으로 x좌표의 값만 좌우를 반전시킨다.

```
for (int i=0;i<img->height;i++)
    for (int j=0;j<img->width*3;j+=3)
    {
        k = (img->width*3) - j - 3;
        img->pixels[i][k] = tmp_img.pixels[i][j];
        img->pixels[i][k+1] = tmp_img.pixels[i][j+1];
        img->pixels[i][k+2] = tmp_img.pixels[i][j+2];
    }
```

- (4) 상하 반전인 경우, x좌표는 그대로 두고 x축을 기준으로 y좌표의 값만 상하를 반전시킨다.

```
for (int i=0;i<img->height;i++)
{
    k = img->height - i - 1;
    for (int j=0;j<img->width*3;j++)
    {
        img->pixels[k][j] = tmp_img.pixels[i][j];
    }
}
```

k의 값에 하나는 -3, 하나는 -1을 해주는 이유는 배열의 시작과 관련이 있다. 기존의 배열의 시작은 1이 아닌 0인데, 좌우 반전의 경우 j의 값(x좌표의 값)을 3씩 더해주므로 -3을 해주었다.

#### 4.2.3.3 영상 회전

- (1) 사용자로부터 degree(각도)를 입력받는다.
- (2) center\_x, center\_y(회전 중심), seta를 설정한다.
- (3) int 타입의 new\_x, new\_y, a를 선언한다. 그리고 R, G, B 매트릭스를 저장할 이중 포인터를 2개씩 선언한다.
- (4) img의 가로 길이, 세로 길이 만큼 R, R\_t, G, G\_t, B, B\_t에 메모리를 할당한다.
- (5) R,B,G 이중 포인터는 비어있는 채로 두고, pixels를 읽어 R\_t, G\_t, B\_t에 각각 R, G, B의 값을 넣는다.

ex. 

255	0	0	0	255	0
0	0	255	255	255	255

255	0
0	255

0	255
0	255

0	0
255	255

원본(예시) - 2\*2 pixels                      R\_t                      G\_t                      B\_t

- (6) R, G, B 이중 포인터에 R\_t, G\_t, B\_t를 이용하여 rotation을 적용한다.

```
// rotation
// x0 : x축의 중점
// y0 : y축의 중점
// x1, y1 : 현재 옮기고자 하는 픽셀의 좌표
// x2, y2 : 이동 후 픽셀의 좌표
// degree : 회전 각도
pi = 3.141592
seta = pi / (180.0/degree)
x2 = (y1 - y0) * sin(seta) + (x1-x0)*cos(seta)+x0
y2 = (y1 - y0) * cos(seta) - (x1-x0)*sin(seta)+y0
```

- (7) 적용한 R, G, B를 이중 포인터를 읽어 img pixels에 합쳐서 집어넣는다.
- (8) 위에서 할당했던 값을 모두 해제한다.

#### 4.2.3.4 채널 추출

- (1) 사용자의 입력에 따라 case를 나눈다.
- (2) R, G, B중 사용자가 추출하고 싶은 색상 외에 다른 색상은 모두 0으로 처리한다.

```
// 예시 - 빨간색 + 초록색 추출
for (int i=0;i<img->height;i++)
    for (int j=0;j<img->width*3;j+=3)
    {
        img->pixels[i][j+2] = 0
    }
```

#### 4.2.3.5 그레이 스케일 변환

- (1) unsigned char 타입의 Gray 변수를 선언한다.
- (2) 이중 for문을 이용하여, 픽셀을 한 개씩 읽어 R, G, B를 모두 더한 후 3으로 나눈 값을 Gray 값에 넣는다.
- (3) R, G, B에 Gray 값을 넣는다



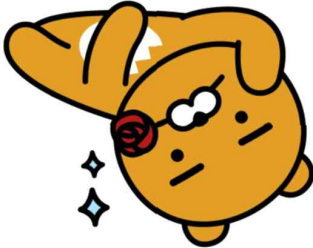
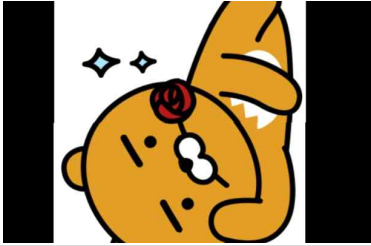







```

for (int i=0;i<img->height;i++)
    for (int j=0;j<img->width*3;j+=3)
    {
        Gray = (img->pixels[i][j] + img->pixels[i][j+1] + img->pixels[i][j+2])/3
        img->pixels[i][j] = Gray;
        img->pixels[i][j+1] = Gray;
        img->pixels[i][j+2] = Gray;
    }

```

#### 4.2.4 핵심 기능 실행 결과

파일 열기		<div>파일 이름을 입력하세요 : C:\Users\LG\Rion.ppm</div> <div>파일을 열었습니다.</div> <div>   Rion </div>	
파일 저장		<div>저장할 파일 이름을 입력하세요 : C:\Users\LG\result.ppm</div> <div>파일을 저장합니다.</div> <div>   result </div>	
편집	원본 사진	 Rion.ppm	
	색상 반전		
	좌우/상하 반전	좌우 반전	

		상하 반전			
	영상 회전	90도 회전			
		180도 회전			
		-90도 회전			
	채널 추출				
		빨간색 추출	초록색 추출	파란색 추출	
					
		빨+초 추출	초+파 추출	빨+파 추출	
	그레이 스케일 변환				

#### 4.3 함수 선언부(또는 함수 원형) 코드

```
// PPMfile.h
#include <iostream>
#include <stdlib.h>
#include <math.h>
#include "Print.h"
Using namespace std;
#define PI 3.14159265
```

```
Typedef struct PPMimage {...} PPMimage;
```

```
PPMimage::PPMimage();
```

```
PPMimage::PPMimage(PPMimage &img);
```

```
int PPMimage::exist_PPMimage();
```

```
void deletePPM(PPMimage* img);
```

```
int Read_PPM(char* filename, PPMimage* img);
```

```
int Save_PPM(char* filename, PPMimage* img);
```

```
void color_Invert(PPMimage* img);
```

```
void Filp(PPMimage* img);
```

```
RGB_matrix(PPMimage *img, unsigned char**
&color, unsigned char**& color_t, char RGB);
```

```
RGB_Rotaion(PPMimage *img, unsigned char**
&color, unsigned char** &color_t, double
center_x, double center_y, double seta);
```

```
void Rotion(PPMimage* img);
```

```
void channel_RGB(PPMimage* img);
```

```
void GrayScale(PPMimage* img);
```

```
//main.cpp
#include <iostream>
#include "PPMfile.h"
#include "Print.h"
```

```
// Print.h
```

```
#pragma once
#include <iostream>
using namespace std;
```

```
int Main_Menu();
```

```
int edit_Menu();
```

```
int Filp_Menu();
```

```
double Rotion_Menu();
```

```
int channel_Menu();
```

```
void ErrorPrint();
```

## 5. 결과 분석 및 검토

### 5.1 문제 해결 과정 검토

- 한 픽셀에 R, G, B라는 데이터 3개가 들어있었기 때문에, img의 pixels값을 변경할 때마다 이를 신경 써야 했다. ex. for문에서 가로길이 \*3, 영상 회전 등
- 메모리를 자주 할당하고 해제하기 때문에 놓친 부분이 없는지 검토했다. ex. 기존 img 객체를 지우고 파일을 새로 열 때, 프로그램을 종료할 때 delete 해주었다.

### 5.2 앞으로 시도할 의미가 있는 기술 요소

#### 5.2.1 확장 기능 추가

- 이동하기 : 사용자에게 얼마만큼 이동할 건지 입력받아 pixels의 좌표를 변경한다.
- 잘라내기 : 사용자에게 영역의 좌표를 입력받아 가로 길이, 세로 길이를 변경한 후 pixels도 새로 할당해 넣어준다.

#### 5.2.2 GUI 구현

- GUI로 구현하여 기존 사진에 영상 처리한 사진을 사용자에게 보여준다.

#### 5.2.3 구조체

- 한 픽셀의 R, G, B 값을 저장하는 pixels 타입을 정의한다.

### 5.3 여러 측면에서의 코드 평가

#### 5.3.1 장점

##### (1) 함수의 간결성

- 보조 함수를 사용하여 회전 함수 외에는 비교적 간결하다.

##### (2) 사용자의 사용 편리함

- 기능마다 돌아가기 키를 추가하였고, 선택될 때마다 적용되었다는 메시지를 출력하도록 하였다.

#### 5.3.2 단점

##### (1) 프로그램 구조의 깔끔함 부족

- 소스는 cpp인데 C를 쓴 경우가 많다. 일정하지 않고 많이 혼합되어 있다.
- 회전 함수는 더 짧게 줄일 수 있을 것 같은 코드인데 그런 방법을 찾지 못해 길어졌다.
- 함수, 또는 프로그램 구조가 일정하지 않다. (Menu 출력 장소, 반환 값 등)

##### (2) 오류 처리의 섬세함 부족

- 상대적으로 main()보다 다른 함수를 신경 쓰는 바람에 각종 입력 오류가 생길 확률이 높다.
- 사용자가 할 만한 오류를 충분히 예측하지 못했다.

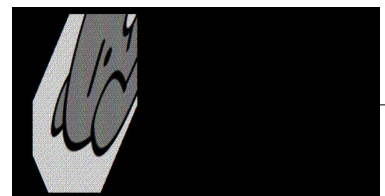
### 5.4 실행이 생각과 달랐을 경우 원인 추측 또는 분석

#### 5.4.1 영상 회전

- 설계 당시 작성한 영상 회전 코드는 R, G, B로 나누지 않고 rotation를 그대로 적용하였는데, 결과를 보니 흑백인 상태로 회전이 되었다. hole도 발생했고, 회전도 제대로 되지 않고 잘렸다. 알고 보니 설계 당시 작성한 영상 회전 코드는 이미지의 bit수가 8bit인, 즉 흑백 영상에만 가능했다.
- 각도가 양의 값이면 시계방향으로, 음의 값이면 반시계방향으로 회전을 줄 알았는데 그 반대였다. 픽셀의 새로운 위치를 계산하는 과정에서 발생한 것으로 보이며, 이는 굳이 고치지 않았다.

설계 때 짰던 영상 회전  
코드를 적용한 결과

→





## 6. 개발 일지 및 후기

### 6.1 개발 일지

날짜	요구 분석	설계	보고서	발표 자료	함수 구현	비고
2020.09.01.~09	○			○		
2020.09.10					○	main 구현
2020.09.11.~16		○	○			
2020.09.18					○	파일 읽기, 저장 구현
2020.09.19					○	색 반전, 좌우/상하 반전 구현
2020.09.20					○	rgb 추출 구현
2020.09.21					○	회전 구현
2020.09.22.~24			○	○		

### 6.2 후기

PPM이란 파일이 있다는 것을 처음 들어봐서 문제를 받았을 땐 매우 당황스러웠다. 올해 1학기까지 프로그래밍 언어를 배우면서 파일 입출력을 어려워하기도 했던 터라 9월 말에 과연 제출할 수 있을지 걱정이었다. 그랬던 만큼 PPM 이미지 정의나 영상 처리 관련해서 자료를 많이 찾아보았다. 자료나 이해되지 않는 알고리즘은 손으로 직접 써가면서 프로그래밍했다. 상대적으로 파이썬이나 자바를 선호했어도 조원 사람들과 토론하고 자료를 찾아보면서 C++ 사용을 결정했지만, 사실 여러모로 기피하는 경향이 있었는데 이번 기회로 C++을 다루는 실력이 향상되었다. 결과적으로 프로그램은 완성되었고, 가장 걱정했던 회전까지 수행이 잘 되어 성취감이 컸다.

마지막으로 가장 아쉬운 것은 프로그램의 구조가 깔끔하지 않다는 것이다. while(), switch()문 등의 기본 문법임에도 잘 활용하지 못했다. 그리고 지난 기간 동안 사용자의 입장을 고려하지 않고 코딩했었기 때문인지 오류 처리 부분에서 스스로 굉장히 미숙하다는 것이 느껴졌다.

2단계에서는 이를 보완하기 위해 요구 사항 분석과 설계 단계에서 좀 더 신경을 써서 작성해보고자 한다. 1단계에서 2단계, 2단계에서 3단계. 단계가 올라갈수록 실력이 향상되는 것을 느끼고 싶다.

## 7. 참고문헌

- [1] 오클라호미호, 여린 개발자 블로그, <https://oneday0012.tistory.com/19>
- [2] EK, (기말고사) 1주차 노트2, [https://blog.naver.com/rose1216\\_/221265850836](https://blog.naver.com/rose1216_/221265850836)
- [3] Mise en Musique, 코딩으로 이미지를 회전시키기, <https://m.blog.naver.com/PostView.nhn?blogId=virapasas&logNo=130185961923&proxyReferer=https:%2F%2Fwww.google.com%2F>