

(1) 다음 영상을 histogram stretching하는 프로그램을 작성하라

입력영상: /image/puppy.pgm (10점)

```
#include <stdio.h>

int main(int argc, char * argv[]) //argc는 외부에서 입력한 문자열 개수,
{
    //argv[]는 외부에서 입력한 문자열들
    int image1[600][800]; //800x600 영상까지 처리가능
    int image2[600][800]; //800x600 영상까지 처리가능

    int x, y; //반복문에서 사용할 정수 변수 x, y 선언
    char M, N; //문자 변수 M과 N 선언
    int XX, YY, MAX; //정수 변수 XX, YY, MAX 선언
    FILE *file1, *file2; //파일 변수 file1과 file2 선언
    file1 = fopen(argv[1], "r"); //외부에서 입력한 두 번째 문자열을 읽기모드로 옴
    fscanf(file1, "%c", &M); //헤더의 매직넘버를 읽어 들임 'P'
    fscanf(file1, "%c", &N); //헤더의 매직넘버를 읽어 들임 '2'
    fscanf(file1, "%d", &XX); //입력영상의 가로크기 읽어 들임
    fscanf(file1, "%d", &YY); //입력영상의 세로크기 읽어 들임
    fscanf(file1, "%d", &MAX); //입력영상의 최대 명암도 읽어 들임
```

```

for(y = 0; y < YY; y++)          //반복문을 사용하여 입력영상의 픽셀 값 읽어 들임
    for(x = 0; x < XX; x++)
        fscanf(file1, "%d", &image1[y][x]);

for(y = 0; y < YY; y++)
    for(x = 0; x < XX; x++)
    {
        image2[y][x] = image1[y][x] * 2; //입력 영상의 모든 픽셀에 대비 상수를 곱해줌
        if(image2[y][x] > 255)           //픽셀 값이 255가 넘으면 255로 고정
            image2[y][x] = 255;
        else if(image2[y][x] < 0)       //픽셀 값이 0보다 작으면 0으로 고정
            image2[y][x] = 0;
    }

file2 = fopen(argv[2], "w");          //외부에서 입력한 세 번째 문자열을 쓰기모드로 열
fprintf(file2, "%c", M);              //읽어 들인 두 번째 문자열 파일의 헤더 매직넘버를
fprintf(file2, "%cWn", N);            //세 번째 문자열 파일에 입력
fprintf(file2, "%d %dWn", XX, YY);    //읽어 들인 가로축과 세로축 크기를 입력
fprintf(file2, "%dWn", MAX);          //읽어 들인 최대 명암도를 입력

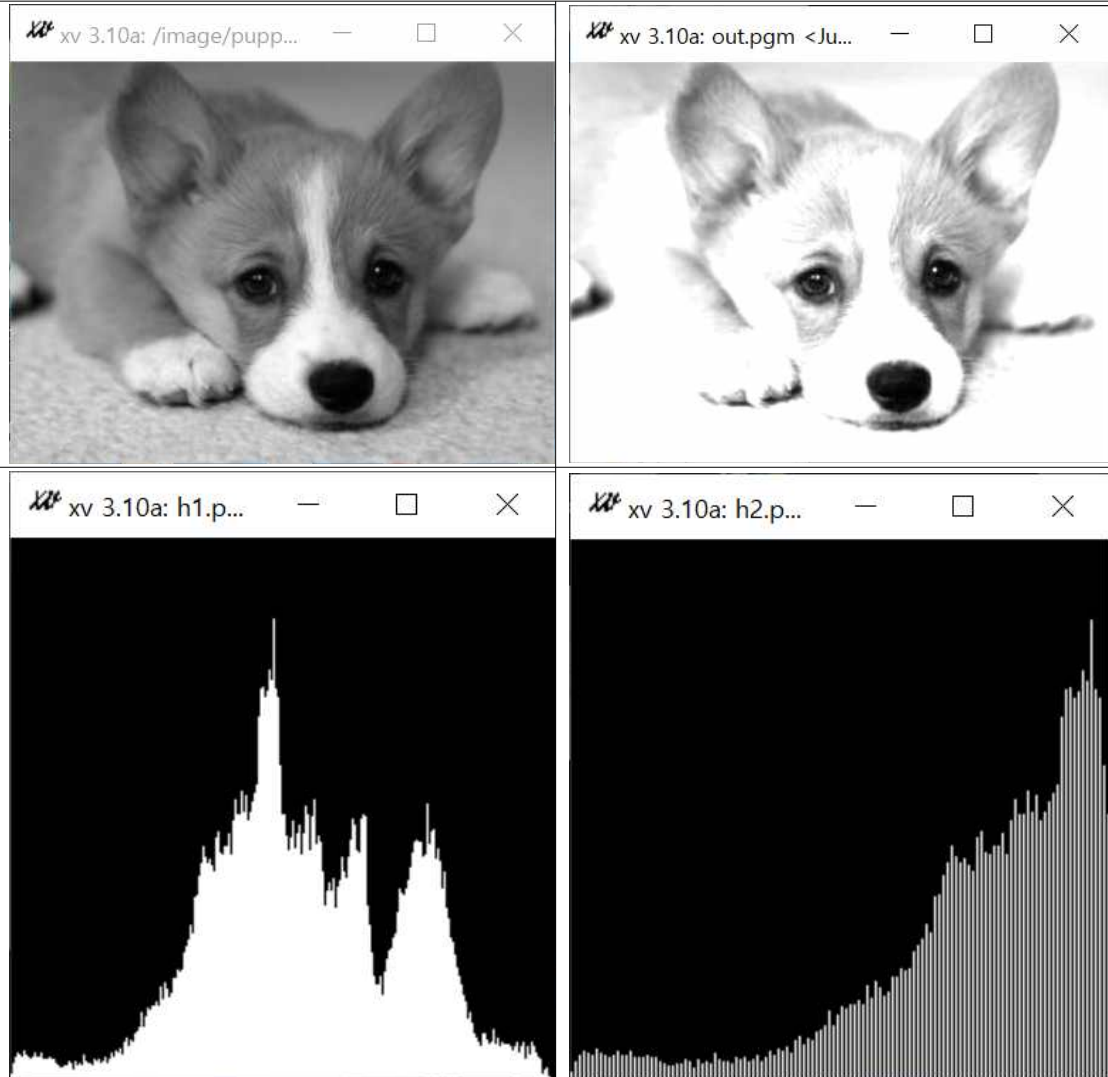
for(y = 0; y < YY; y++){
    for(x = 0; x < XX; x++){          //반복문을 사용하여 contrast가 개선된 영상을
        fprintf(file2,"%4d", image2[y][x]); //세 번째 문자열 파일에 입력
    }
    fprintf(file2, "Wn");              //줄 내림 입력
}

return 0;
}

```

What are you doing?

```
[kimhyungho@raspberrypi4 ~]gcc stretch.c -o stretch
[kimhyungho@raspberrypi4 ~]xv /image/puppy.pgm &
[1] 29422
[kimhyungho@raspberrypi4 ~]stretch /image/puppy.pgm out.pgm
[kimhyungho@raspberrypi4 ~]xv out.pgm &
[2] 29424
[kimhyungho@raspberrypi4 ~]hist3 /image/puppy.pgm h1.pgm
[kimhyungho@raspberrypi4 ~]xv h1.pgm &
[3] 29426
[kimhyungho@raspberrypi4 ~]hist3 out.pgm h2.pgm
[kimhyungho@raspberrypi4 ~]xv h2.pgm &
[4] 29428
[kimhyungho@raspberrypi4 ~]
```



fsacnf로 읽기모드로 연 입력영상 /image/puppy.pgm의 헤더와 데이터 포맷 부분을 읽어 들였습니다. 읽어 들인 영상의 대비가 개선 된 영상을 얻기 위해 입력 영상의 모든 픽셀에 대비 상수(2)를 곱해주었습니다. 그리고 픽셀 값이 255가 넘으면 255로 고정되고 0미만이 되면 0이 되게 조건문을 사용했습니다. 입력 영상에서 읽어 들인 영상의 헤더와 대비상수를 곱해준 영상의 데이터 포맷을 쓰기모드로 연 out.pgm에 fprintf를 사용하여 입력했습니다. 이 후 실행과정에서 gcc와 -o옵션을 사용해서 stretch.c 컴파일하고 실행파일 stretch를 만들어주었습니다. 입력영상에 /image/puppy.pgm을 넣고 출력영상으로 out.pgm을 넣어 작업했고 xv와 후면작업 &를 사용하여 동시에 출력하고 저번 과제에서 만들었던 그레이스케일 영상의 히스토그램을 출력하는 hist3를 사용하여 두 영상의 히스토그램까지 출력해보았습니다. /image/puppy.pgm이 이미 좋은 명암대비를 가지고 있었기 때문에 clamping에 의해 잘려나간 부분도 많아서 과하게 밝아진 느낌을 볼 수 있었습니다.

(2)다음 영상을 contrast stretching하는 프로그램을 작성하라

입력영상: /image/moth.pgm (20점)

/image/statue.pgm

```
#include <stdio.h>
int main(int argc, char * argv[])    //argc는 외부에서 입력한 문자열 개수,
{                                     //argv[]는 외부에서 입력한 문자열들
    int image1[600][800];            //800x600 영상까지 처리가능
    int image2[600][800];            //800x600 영상까지 처리가능
    int x, y;                        //반복문에서 사용할 정수 변수 x, y 선언
    char M, N;                       //문자 변수 M과 N 선언
    int XX, YY, MAX;                 //정수 변수 XX, YY, MAX 선언
    FILE *file1, *file2;             //파일 변수 file1과 file2 선언

    int frequency[256]={0};          //명암도의 빈도를 체크할 변수 선언후 초기화
    int bright, height;               //정수 변수 bright와 정수 변수 height 선언
    int min, max;                    //최소 픽셀 값 정수 변수 min와 최대 픽셀 값 정수 변수 max 선언

    file1 = fopen(argv[1], "r");      //외부에서 입력한 두 번째 문자열을 읽기모드로 열
    fscanf(file1, "%c", &M);          //헤더의 매직넘버를 읽어 들임 'P'
    fscanf(file1, "%c", &N);          //헤더의 매직넘버를 읽어 들임 '2'
    fscanf(file1, "%d", &XX);         //입력영상의 가로크기 읽어 들임
    fscanf(file1, "%d", &YY);         //입력영상의 세로크기 읽어 들임
    fscanf(file1, "%d", &MAX);        //입력영상의 최대 명암도 읽어 들임

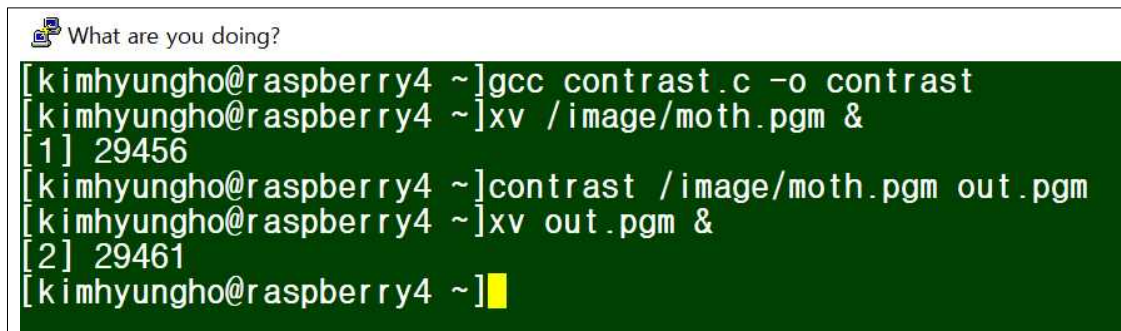
    for(y = 0; y < YY; y++)          //반복문을 사용하여 입력영상의 픽셀 값 읽어 들임
        for(x = 0; x < XX; x++)
            fscanf(file1, "%d", &image1[y][x]);
```

```

for(y = 0; y < YY; y++)
    for(x = 0; x < XX; x++)
    {
        bright = image1[y][x];          //입력 영상의 픽셀 값을 bright에 넣음
        frequency[bright] = frequency[bright] + 1;    //같은 픽셀 값이 나오면
    }                                     //해당 픽셀 값의 빈도수 증가
for(bright=255; bright >= 0;bright--)
    if(frequency[bright])                //히스토그램의 최대값 구하기
    { max = bright;                      //빈도수가 0이 아닌 가장 픽셀 값이 큰 수를 max
      break; }
for(bright=0;bright < 256;bright++)
    if(frequency[bright])                //히스토그램의 최소값 구하기
    { min = bright;                      //빈도수가 0이 아닌 가장 픽셀 값이 작은 수를 min
      break; }
for(y=0;y<YY;y++)
    for(x=0;x<XX;x++)                    //명암대비 스트레칭
        image2[y][x] = (float)(image1[y][x] - min)/(max - min) * 255;
                                                //float는 정수/정수가 0이 되는 것을 방지하기 위한 cast
file2 = fopen(argv[2], "w");             //외부에서 입력한 세 번째 문자열을 쓰기모드로 열
fprintf(file2, "%c", M);                 //읽어 들인 두 번째 문자열 파일의 헤더 매직넘버를
fprintf(file2, "%cWn", N);               //세 번째 문자열 파일에 입력
fprintf(file2, "%d %dWn", XX, YY);       //읽어 들인 가로축과 세로축 크기를 입력
fprintf(file2, "%dWn", MAX);             //읽어 들인 최대 명암도를 입력

for(y = 0; y < YY; y++){
    for(x = 0; x < XX; x++){              //반복문을 이용하여 명암대비 스트레칭한 영상을
        fprintf(file2,"%4d", image2[y][x]); //세 번째 문자열 파일에 입력
    }
    fprintf(file2, "Wn");                  //줄 내림 입력
}
return 0;
}

```

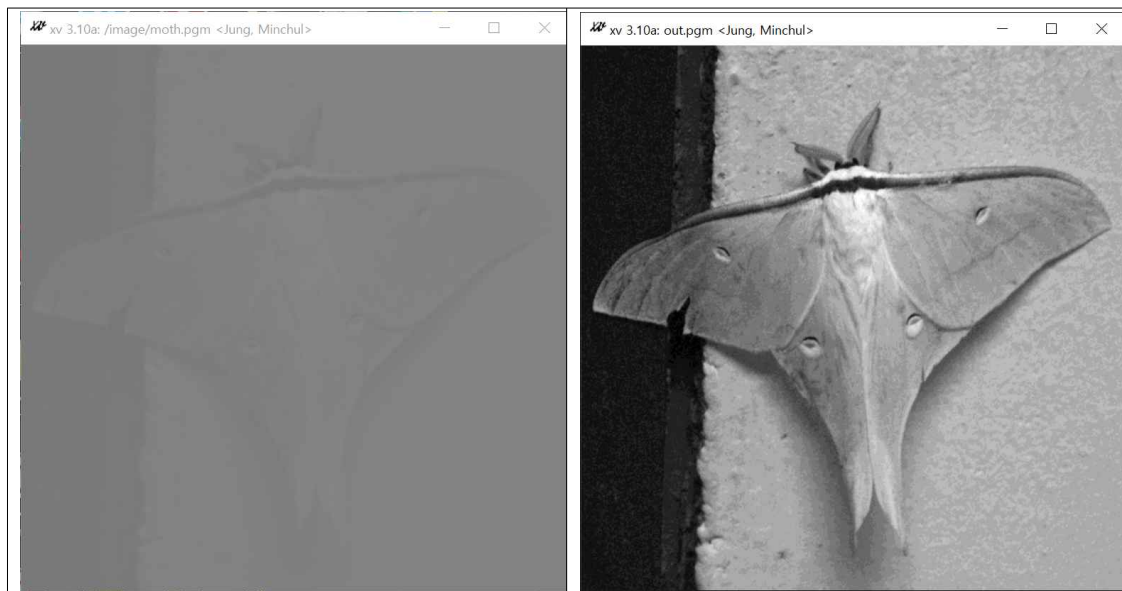


A terminal window titled "What are you doing?" shows the following commands and output on a Raspberry Pi 4:

```

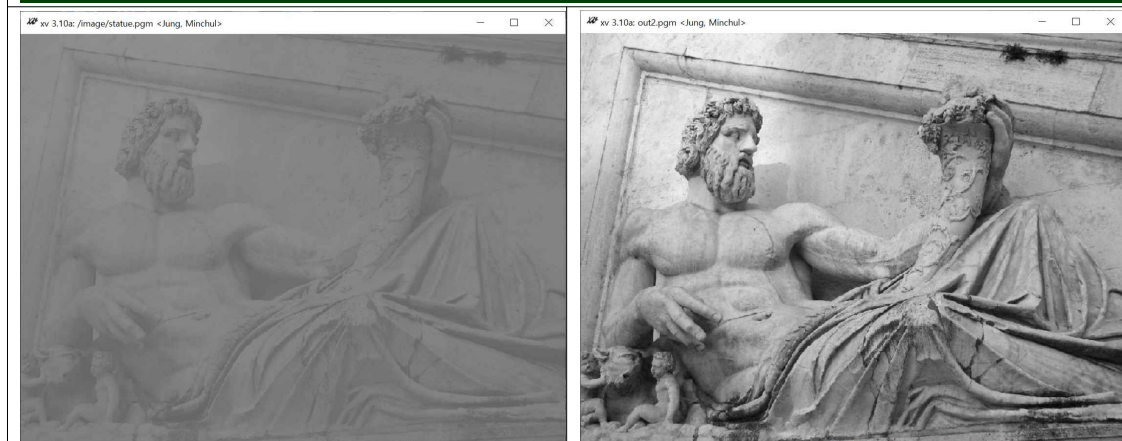
[kimhyungho@raspberrry4 ~]gcc contrast.c -o contrast
[kimhyungho@raspberrry4 ~]xv /image/moth.pgm &
[1] 29456
[kimhyungho@raspberrry4 ~]contrast /image/moth.pgm out.pgm
[kimhyungho@raspberrry4 ~]xv out.pgm &
[2] 29461
[kimhyungho@raspberrry4 ~]

```



What are you doing?

```
[kimhyungho@raspberrry4 ~]xv /image/statue.pgm &  
[1] 29590  
[kimhyungho@raspberrry4 ~]contrast /image/statue.pgm out2.pgm  
[kimhyungho@raspberrry4 ~]xv out2.pgm &  
[2] 29600  
[kimhyungho@raspberrry4 ~]
```



이번에는 명암대비 스트레칭을 사용하여 아무 값이나 대비상수로 곱해주는 것이 아니라 적절한 값을 곱해주었습니다. 적절한 값을 구하는 공식은 (입력영상의 픽셀 값 - 가장 어두운 값) / (가장 밝은 값 - 가장 어두운 값) × 255를 출력영상의 픽셀 값으로 사용하면 됩니다. 즉 $O(x,y) = \frac{I(x,y) - \min}{\max - \min} \times 255$ 의 식이 됩니다. fsacnf로 읽기모드로 연 입력영상 /image/moth.pgm과 statue.pgm의 헤더와 데이터 포맷 부분을 읽어 들였습니다. 읽어 들인 영상의 데이터 포맷에 조건문을 사용하여 빈도수가 0이 아닌 밝기 값이 제일 큰 값 max와 빈도수가 0이 아닌 밝기 값이 제일 작은 값 min을 구하고 위 식을 적용하여 명암대비 스트레칭 시켰습니다. 입력 영상에서 읽어 들인 영상의 헤더와 대비상수를 곱해준 영상의 데이터 포맷을 쓰기모드로 연 out.pgm(moth 영상의 출력영상)과 out2.pgm(statue 영상의 출력영상)에 fprintf를 사용하여 입력했습니다. 이 후 실행과정에서 gcc와 -o 옵션을 사용해서 contrast.c를 컴파일하고 실행파일 contrast를 만들어주었습니다. moth와 statue 각각 입력영상과 출력영상을 xv와 후면작업 &를 사용하여 동시에 출력했습니다.

(3) 다음 컬러 영상을 contrast stretching하는 프로그램을 작성하라

입력영상: /image/crayfish.ppm (50점)

```
#include <stdio.h>

int main(int argc, char * argv[])    //argc는 외부에서 입력한 문자열 개수,
{                                     //argv[]는 외부에서 입력한 문자열들
    int image1[600][800*3];          //800x600 영상까지 처리가능 컬러영상
    int image2[600][800*3];          //800x600 영상까지 처리가능 컬러영상
    int x, y;                         //반복문에서 사용할 정수 변수 x, y 선언
    char M, N;                        //문자 변수 M과 N 선언
    int XX, YY, MAX;                  //정수 변수 XX, YY, MAX 선언
    FILE *file1, *file2;              //파일 변수 file1과 file2 선언

    int frequency[256]={0};           //명암도의 빈도를 체크할 배열 선언 후 초기화
    int bright;                       //정수 변수 bright 선언
    int min, max;                     //최소 픽셀 값 정수 변수 min와 최대 픽셀 값 정수 변수 max 선언

    file1 = fopen(argv[1], "r");      //외부에서 입력한 두 번째 문자열을 읽기모드로 열
    fscanf(file1, "%c", &M);          //헤더의 매직넘버를 읽어 들임 'P'
    fscanf(file1, "%c", &N);          //헤더의 매직넘버를 읽어 들임 '3'
```

```

fscanf(file1, "%d", &XX);    //입력영상의 가로크기 읽어 들임
fscanf(file1, "%d", &YY);    //입력영상의 세로크기 읽어 들임
fscanf(file1, "%d", &MAX);    //입력영상의 최대 명암도 읽어 들임

for(y = 0; y < YY; y++)      //반복문을 사용하여 입력영상의 픽셀 값 읽어 들임
    for(x = 0; x < XX*3; x++) //컬러영상은 가로픽셀 당 값을 3개 가지므로 x3
        fscanf(file1, "%d", &image1[y][x]);

for(y = 0; y < YY; y++)
    for(x = 0; x < XX*3; x++) //컬러영상은 가로픽셀 당 값을 3개 가지므로 x3
    {
        bright = image1[y][x]; //입력 영상의 픽셀 값을 bright에 넣음
        frequency[bright] = frequency[bright] + 1; //같은 픽셀 값이 나오면
    } //해당 픽셀 값의 빈도수 증가

for(bright=255; bright >= 0;bright--)
    if(frequency[bright]) //히스토그램의 최대값 구하기
    { max = bright; //빈도수가 0이 아닌 가장 픽셀 값이 큰 수를 max
      break; }

for(bright=0;bright < 256;bright++)
    if(frequency[bright]) //히스토그램의 최솟값 구하기
    { min = bright; //빈도수가 0이 아닌 가장 픽셀 값이 작은 수를 min
      break; }

for(y=0;y<YY;y++)
    for(x=0;x<XX*3;x++) //명암대비 스트레칭
        image2[y][x] = (float)(image1[y][x] - min)/(max - min) * 255;
                                //float는 정수/정수가 0이 되는 것을 방지하기 위한 cast

file2 = fopen(argv[2], "w"); //외부에서 입력한 세 번째 문자열을 쓰기모드로 엮
fprintf(file2, "%c", M); //읽어 들인 두 번째 문자열 파일의 헤더 매직넘버를
fprintf(file2, "%cWn", N); //세 번째 문자열 파일에 입력
fprintf(file2, "%d %dWn", XX, YY); //읽어 들인 가로축과 세로축 크기를 입력
fprintf(file2, "%dWn", MAX); //읽어 들인 최대 명암도를 입력

for(y = 0; y < YY; y++){
    for(x = 0; x < XX*3; x++){ //반복문을 이용하여 명암대비 스트레칭한 영상을
        fprintf(file2,"%4d", image2[y][x]); //세 번째 문자열 파일에 입력
    }
    fprintf(file2, "Wn"); //줄 내림 입력
}
return 0;
}

```



```

What are you doing?
[kimhyungho@raspberrypi4 ~]$ gcc ccontrast.c -o ccontrast
[kimhyungho@raspberrypi4 ~]$ xv /image/crayfish.ppm &
[1] 29692
[kimhyungho@raspberrypi4 ~]$ ccontrast /image/crayfish.ppm out.ppm
[kimhyungho@raspberrypi4 ~]$ xv out.ppm &
[2] 29700
[kimhyungho@raspberrypi4 ~]$

```

(2)번과 같이 적절한 값을 구해 입력 영상에 곱해서 출력 영상의 값을 구했습니다. 하지만 컬러 영상이기 때문에 한 픽셀에 값을 3개 씩 갖기 때문에 읽어 들이거나 작업하거나 쓸 때 반복문에 x3을 해주었습니다. fscanf로 읽기모드로 연 입력영상 crayfish.ppm의 헤더와 데이터 포맷 부분을 읽어 들였습니다. 읽어 들인 영상의 데이터 포맷에 조건문을 사용하여 빈도수가 0이 아닌 밝기 값이 제일 큰 값 max와 빈도수가 0이 아닌 밝기 값이 제일 작은 값 min을 구하고 위 식을 적용하여 명암대비 스트레칭 시켰습니다. 입력 영상에서 읽어 들인 영상의 헤더와 대비상수를 곱해준 영상의 데이터 포맷을 쓰기모드로 연 out.ppm에 fprintf를 사용하여 입력했습니다. 이 후 실행과정에서 gcc와 -o 옵션을 사용해서 ccontrast.c를 컴파일하고 실행파일 ccontrast를 만들어주었습니다. 입력 영상과 출력영상을 xv와 후면작업 &를 사용하여 동시에 출력했습니다.

(3-1) R, G, B의 밝기 값 각각 최대값과 최소값을 구하여 스트레칭

```
#include <stdio.h>
```

```
int main(int argc, char * argv[]) //argc는 외부에서 입력한 문자열 개수,
```

```
{ //argv[]는 외부에서 입력한 문자열들
```

```
int image1[600][800*3]; //800x600 영상까지 처리가능
```

```
int image2[600][800*3]; //800x600 영상까지 처리가능
```

```

int x, y;                                //반복문에서 사용할 정수 변수 x, y 선언
char M, N;                               //문자 변수 M과 N 선언
int XX, YY, MAX;                         //정수 변수 XX, YY, MAX 선언
FILE *file1, *file2;                    //파일 변수 file1과 file2 선언
//R, G, B 따로 명암도의 빈도를 체크할 배열 선언 후 초기화
int frequency_R[256]={0}, frequency_G[256]={0}, frequency_B[256]={0};
int bright_R, bright_G, bright_B; //R, G, B 각각 밝기 값을 체크할 변수 선언
int min_R, min_G, min_B, max_R, max_G, max_B;
//R, G, B 각각 밝기값의 최대값 최소값 넣을 변수 선언
file1 = fopen(argv[1], "r"); //외부에서 입력한 두 번째 문자열을 읽기모드로 열
fscanf(file1, "%c", &M);    //헤더의 매직넘버를 읽어 들임 'P'
fscanf(file1, "%c", &N);    //헤더의 매직넘버를 읽어 들임 '3'
fscanf(file1, "%d", &XX);    //입력영상의 가로크기 읽어 들임
fscanf(file1, "%d", &YY);    //입력영상의 세로크기 읽어 들임
fscanf(file1, "%d", &MAX);    //입력영상의 최대 명암도 읽어 들임

for(y = 0; y < YY; y++) //반복문을 사용하여 입력영상의 픽셀 값 읽어 들임
    for(x = 0; x < XX*3; x++) //컬러영상은 가로픽셀 당 값을 3개 가지므로 x3
        fscanf(file1, "%d", &image1[y][x]);

for(y = 0; y < YY; y++)
    for(x = 0; x < XX*3; x+=3) //컬러영상은 가로픽셀 당 값을 3개 가지므로 x3
    {
        //R, G, B 각각 구하기 위해 x+=3
        bright_R = image1[y][x]; //입력 영상의 R 밝기 값을 bright에 넣음
        bright_G = image1[y][x+1]; //입력 영상의 G 밝기 값을 bright에 넣음
        bright_B = image1[y][x+2]; //입력 영상의 B 밝기 값을 bright에 넣음
        frequency_R[bright_R] = frequency_R[bright_R] + 1; //R 빈도수 체크
        frequency_G[bright_G] = frequency_G[bright_G] + 1; //G 빈도수 체크
        frequency_B[bright_B] = frequency_B[bright_B] + 1; //B 빈도수 체크
    }

for(bright_R=255; bright_R >= 0; bright_R--)
    if(frequency_R[bright_R]) //R 밝기값 최대값 구하기
    { max_R = bright_R; //빈도수가 0이 아닌 가장 밝기 값이 큰 수를 max_R
      break; }

for(bright_G=255; bright_G >= 0; bright_G--)
    if(frequency_G[bright_G]) //G 밝기값 최대값 구하기
    { max_G = bright_G; //빈도수가 0이 아닌 가장 밝기 값이 큰 수를 max_G
      break; }

for(bright_B=255; bright_B >= 0; bright_B--)
    if(frequency_B[bright_B]) //B 픽셀 값 최대값 구하기
    { max_B = bright_B; //빈도수가 0이 아닌 가장 픽셀 값이 큰 수를 max_G

```

```

        break; }
for(bright_R=0;bright_R < 256;bright_R++)
    if(frequency_R[bright_R])        //R 밝기값 최소값 구하기
        { min_R = bright_R;          //빈도수가 0이 아닌 가장 픽셀 값이 작은 수를 min_R
          break; }
for(bright_G=0;bright_G < 256;bright_G++)
    if(frequency_G[bright_G])        //G 픽셀값 최소값 구하기
        { min_G = bright_G;          //빈도수가 0이 아닌 가장 픽셀 값이 작은 수를 min_G
          break; }
for(bright_B=0;bright_B < 256;bright_B++)
    if(frequency_B[bright_B])        //B 밝기값 최소값 구하기
        { min_B = bright_B;          //빈도수가 0이 아닌 가장 픽셀 값이 작은 수를 min_B
          break; }

for(y=0;y<YY;y++){
    for(x=0;x<XX*3;x+=3){            //R, G, B 각각 명암대비 스트레칭
        image2[y][x] = (float)(image1[y][x] - min_R)/(max_R - min_R) * 255;
        image2[y][x+1] = (float)(image1[y][x+1] - min_G)/(max_G - min_G) * 255;
        image2[y][x+2] = (float)(image1[y][x+2] - min_B)/(max_B - min_B) * 255;
    }
    file2 = fopen(argv[2], "w");      //외부에서 입력한 세 번째 문자열을 쓰기모드로 엮
    fprintf(file2, "%c", M);          //읽어 들인 두 번째 문자열 파일의 헤더 매직넘버를
    fprintf(file2, "%cWn", N);        //세 번째 문자열 파일에 입력
    fprintf(file2, "%d %dWn", XX, YY); //읽어 들인 가로축과 세로축 크기를 입력
    fprintf(file2, "%dWn", MAX);      //읽어 들인 최대 명암도를 입력

    for(y = 0; y < YY; y++){
        for(x = 0; x < XX*3; x++){    //반복문을 이용하여 명암대비 스트레칭한 영상을
            fprintf(file2,"%4d", image2[y][x]); //세 번째 문자열 파일에 입력
        }
        fprintf(file2, "Wn");          //줄 내림 입력
    }

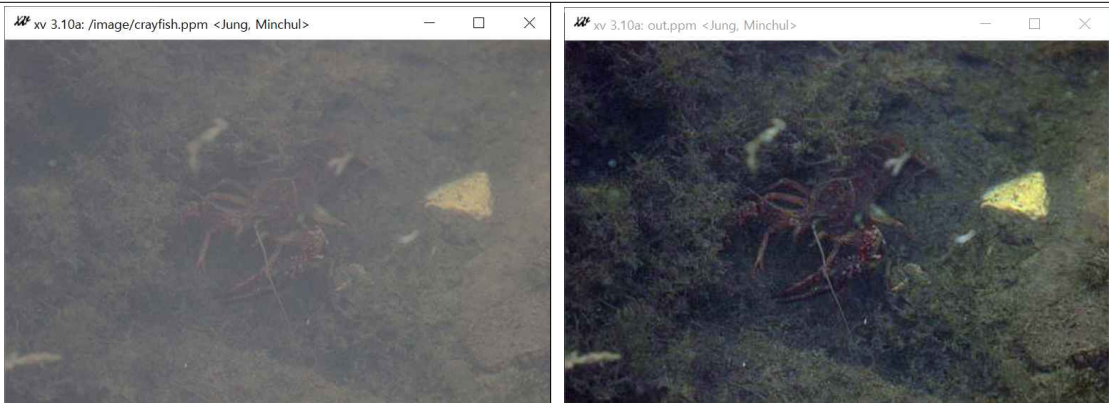
    return 0;
}

```

☞ 워드파일(hwp/doc)로 작성하는 보고서 파일에는

- ① 각각의 줄을 주석 처리한 소스(화면 캡처금지, 텍스트로 작성),
- ② 리눅스 명령 실행 화면, ③ 입력 영상 ④ 출력 영상이 반드시 있어야 하며 과제는 상대 평가 됩니다.

```
What are you doing?  
[kimhyungho@raspberrypi4 ~]$ gcc fcontrast.c -o fcontrast  
[kimhyungho@raspberrypi4 ~]$ xv /image/crayfish.ppm &  
[1] 14723  
[kimhyungho@raspberrypi4 ~]$ fcontrast /image/crayfish.ppm out.ppm  
[kimhyungho@raspberrypi4 ~]$ xv out.ppm &  
[2] 14725  
[kimhyungho@raspberrypi4 ~]$
```



(3)번에서는 R, G, B 값 모두에서 최대값과 최소값을 구해 명암대비 스트레칭을 했다면 이번에는 R, G, B 값 각각의 최대값과 최소값을 구해서 R, G, B 각각 다른 값을 곱하여 명암대비 스트레칭을 해보았습니다. 즉, R이 가지는 밝기 값의 범위와 G가 가지는 밝기 값의 범위, B가 가지는 밝기 값의 범위는 다를 것입니다. 예를 들어 R이 50부터 200까지의 밝기 값을 가지고 있고 G가 20부터 150, B가 70부터 230의 값을 가지고 있다고 생각하면 R의 밝기 값 최소는 50 최대는 200 G는 최소 20 최대 150 B는 최소 70 최대 230의 값을 가져 $O(x,y) = \frac{I(x,y) - \min}{\max - \min} \times 255$ 식이 각각 다른 값이 나오게 될 것입니다. 그렇게 구한 값을 원래 픽셀 값에 곱하여 명암대비 스트레칭 시키니 (3)에서 출력했던 영상과는 좀 다른 스트레칭 된 영상을 얻은 것을 확인 할 수 있었습니다.

예시

