

미니 프로젝트 version 1.0

디지털 영상 처리 w. C언어

[Intel] 엣지 AI SW 아카데미 - 절차지향 프로그래밍

Contents

- 프로젝트의 목표
- 개발 환경
- 화면 구성 및 기능
- 부가 기능
- 마치며

프로젝트의 목표



비전

영상 처리에 대해 학습하여
추후에 진행될
컴퓨터 비전 등에 활용하기



미션

영상처리를 학습하고
이를 라이브러리 없이
구현해 보는 것

개발 환경

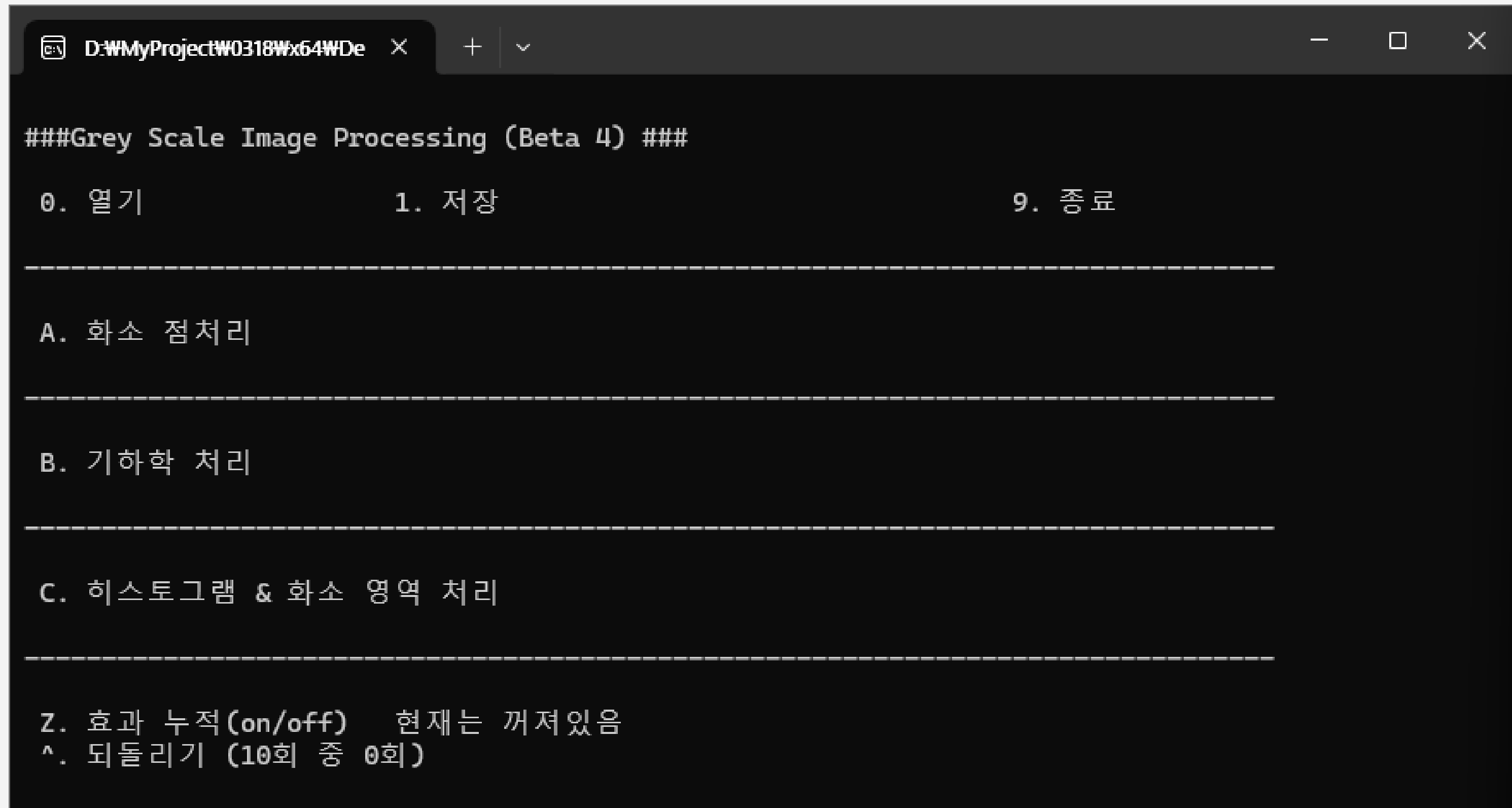


Visual Studio 2022



Visual C++ 2022

화면 구성 및 기능 초기 화면

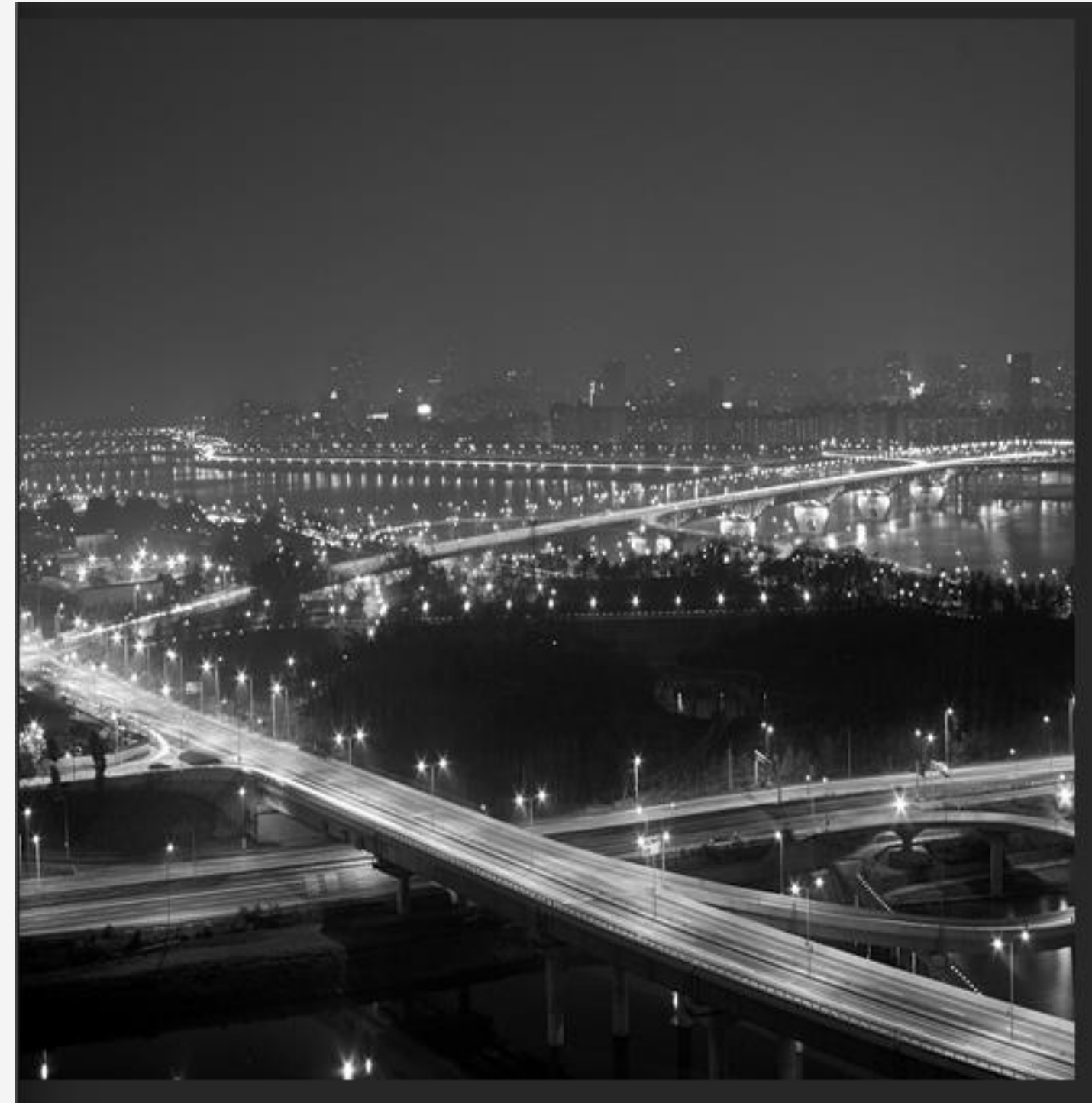


화면 구성 및 기능

0. 지정된 경로에 있는 사진 열기

D:/RAW/Etc_Raw(squire)

파일명 --> nightview512
















화면 구성 및 기능

1. 지정된 경로에 사진 저장하기

D:/RAW/Etc_Raw(squire)

파일명 --> forPPTsave
저장 완료

이름	날짜
 AVE256.RAW	2018-03-12 오전 8:34
 bt512.raw	2024-03-14 오후 3:19
 bt1024.raw	2024-03-14 오후 3:28
 btbt512.raw	2024-03-14 오후 3:21
 circle512.raw	2018-03-12 오전 8:34
 Citrus256.raw	2018-03-12 오전 8:34
 flower512.raw	2018-03-12 오전 8:34
 forPPTsave.raw	2024-03-19 오전 10:31
 ky256.raw	2024-03-14 오후 3:26
 LENNA512.raw	2018-03-12 오전 8:34
 Line512.raw	2018-03-12 오전 8:34
 lotusflower512.raw	2018-03-12 오전 8:34
 lotusflowerbinaryv512.raw	2018-03-12 오전 8:34

화면 구성 및 기능

A 화소점 처리 메뉴

D:\MyProject\W0318Wx64WDe

+

▼

—

□

×

0. 동일영상 알고리즘

A. 밝게 하기

B. 어둡게 하기

C. 반전

D. 감마변환

E. 파라볼라 변환

F. 흑백(127)

G. 흑백(평균값)

H. 흑백(중앙값)

I. 더 밝게 하기 (배수) J. 더 어둡게 하기 (배수) K. 더 밝게 하기 (배수) + 랩핑기법

L. 원형 마스크(AND)

M. 원형 마스크(OR)

N. 원형 마스크(XOR)

O. NOT 반전

P. 명암 대비 스트레칭

Q. 명암 대비 압축

R. 포스터라이징

S. 범위 강조

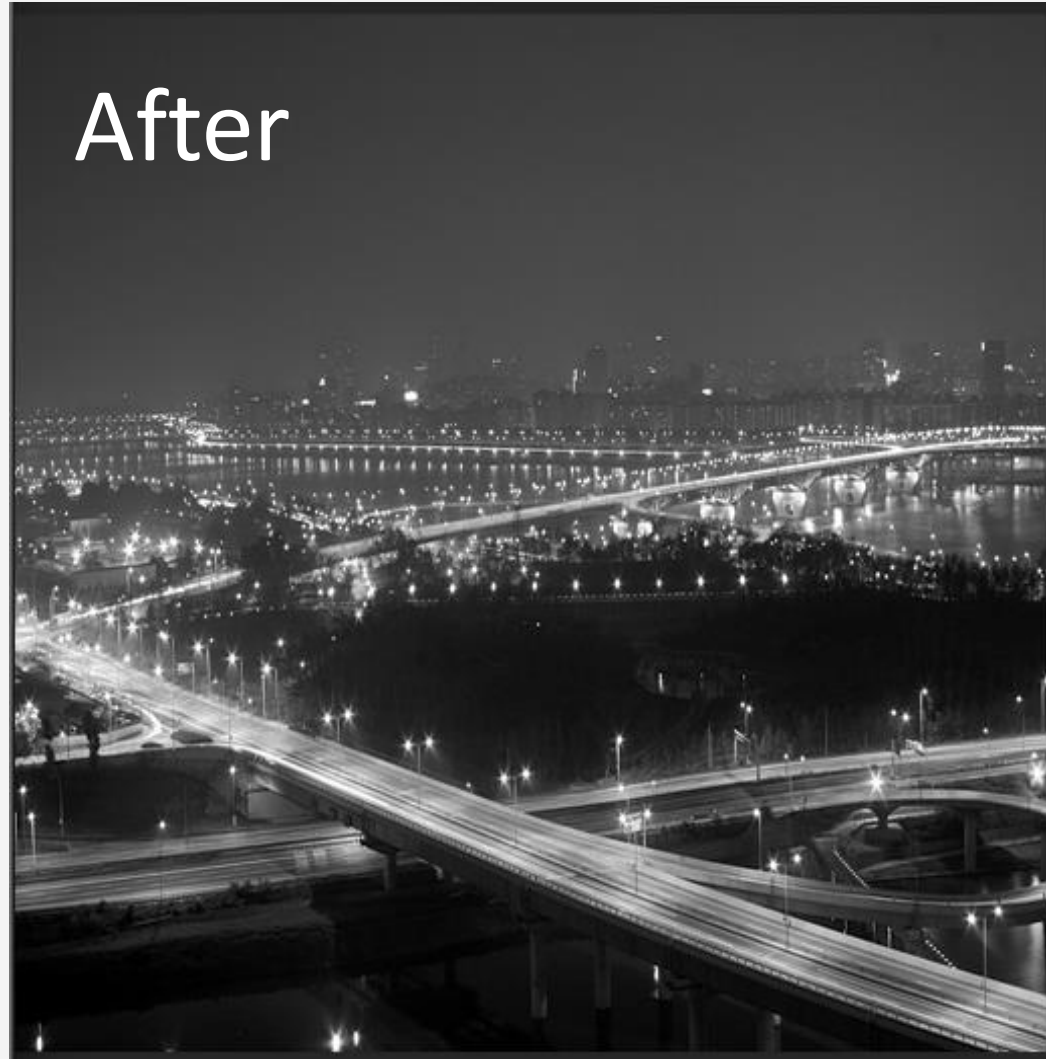
화면 구성 및 기능

A 화소점처리 > 0 동일영상 알고리즘

Before



After

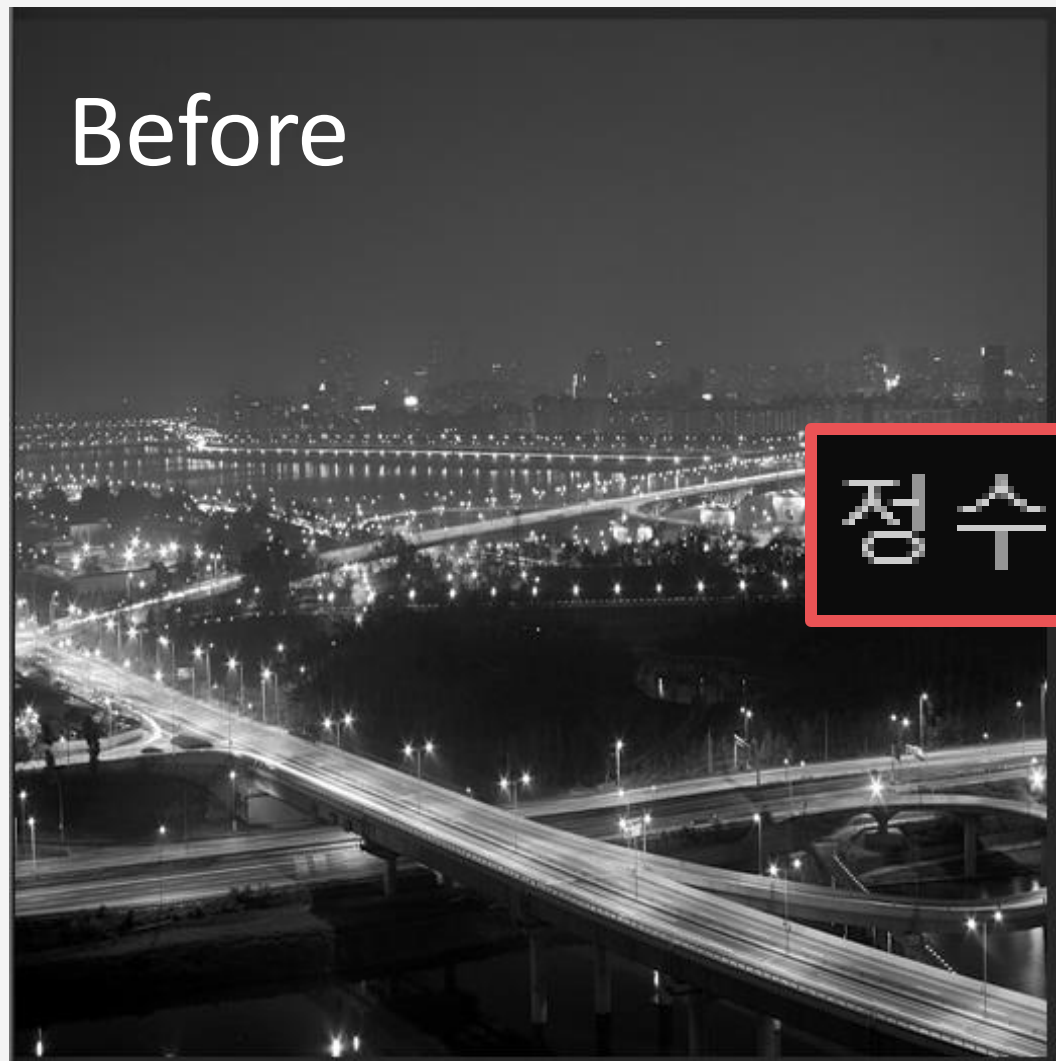


```
outImage[i][k] = inImage[i][k];
```

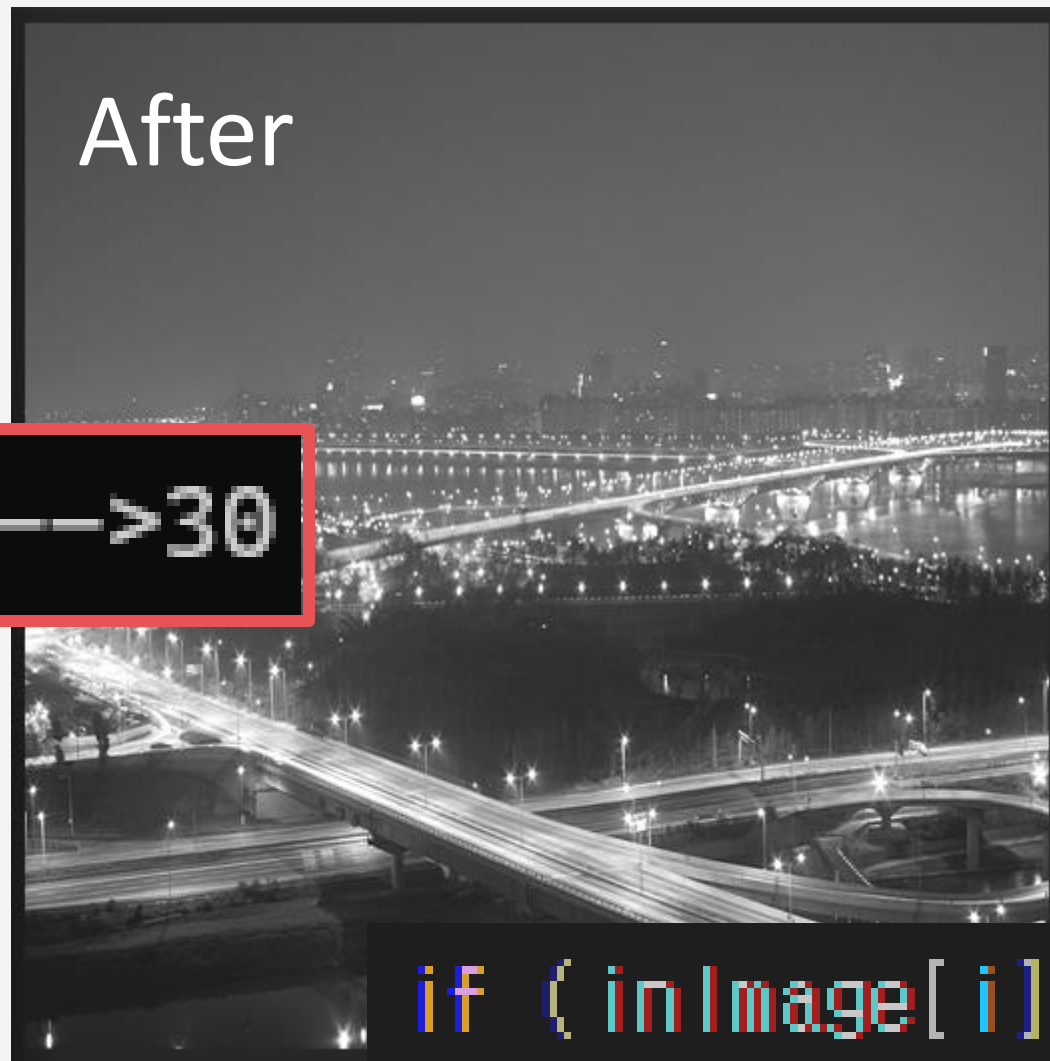
화면 구성 및 기능

A 화소점처리 > A 밝게 하기

Before



After



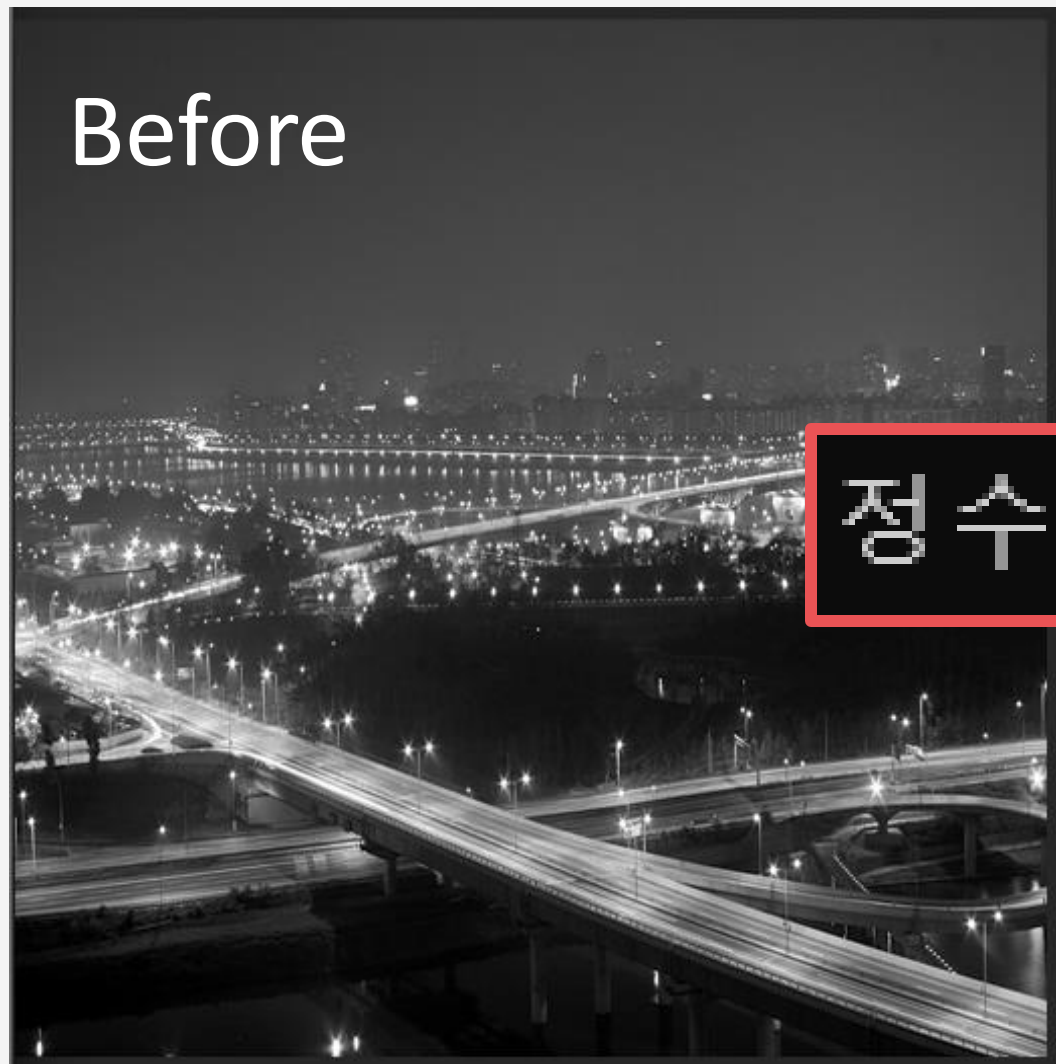
정수값 --> 30

밝기가 최대치인 255를
넘어가지 않도록 방지

```
if ( inImage[i][k] + val < 255 )  
    outImage[i][k] = inImage[i][k] + val;  
else  
    outImage[i][k] = 255;
```

화면 구성 및 기능

A 화소점처리 > B 어둡게 하기

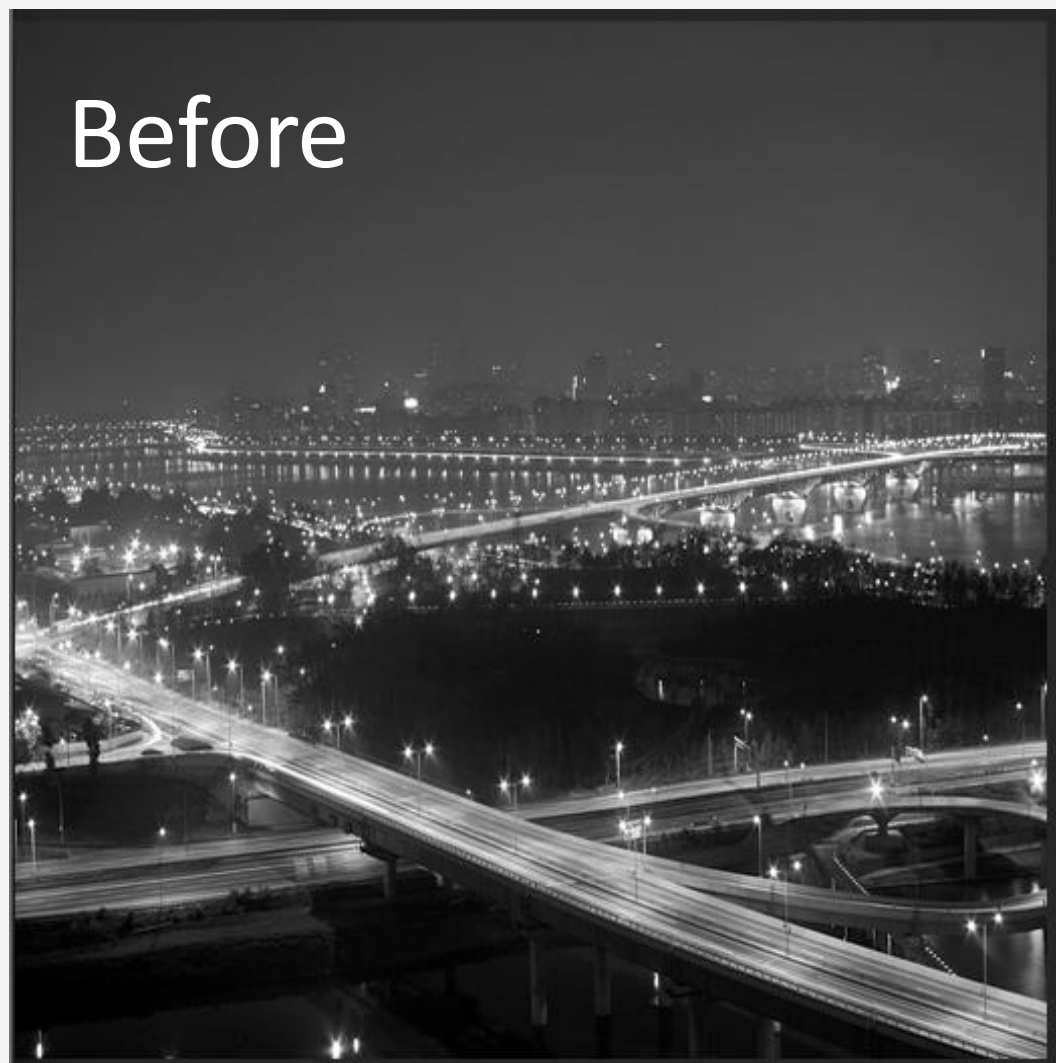


정수 값 --> 30

밝기가 최저치인 0보다
작아지지 않도록 방지

```
if ( inImage[i][k] - val < 0 ) outImage[i][k] = 0;  
else outImage[i][k] = inImage[i][k] - val;
```

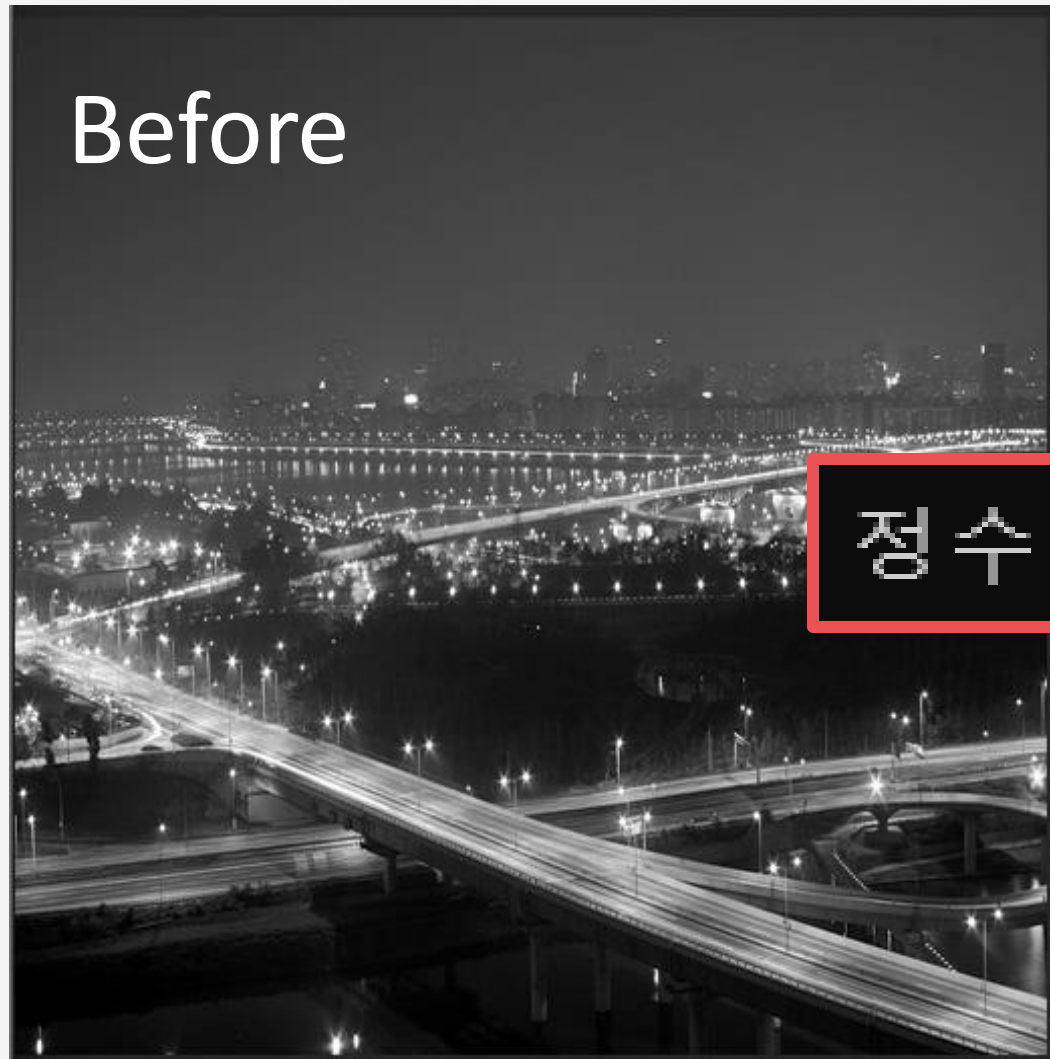
화면 구성 및 기능 A 화소점처리 > C 반전



```
outImage[i][k] = 255 - inImage[i][k];
```


화면 구성 및 기능

A 화소점처리 > D 감마변환



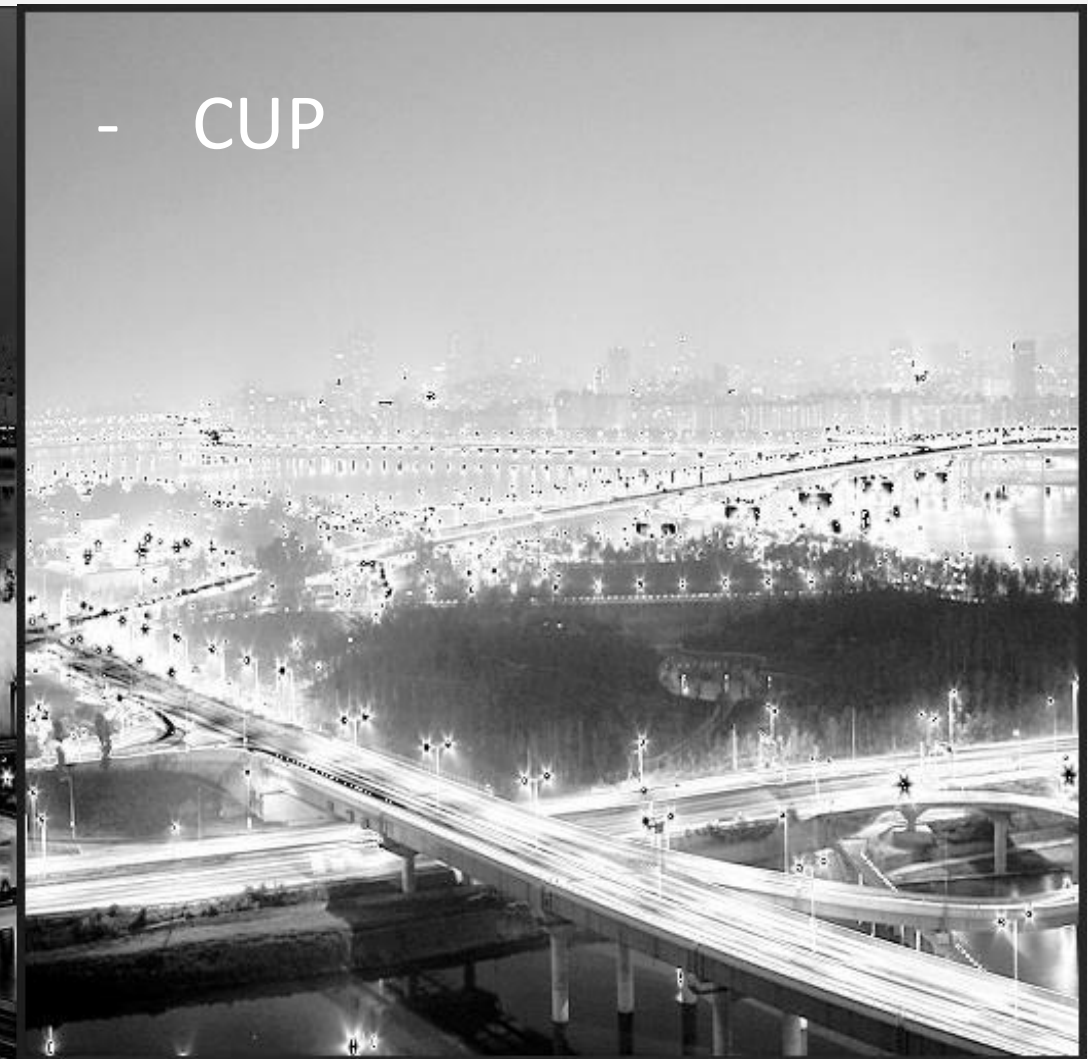
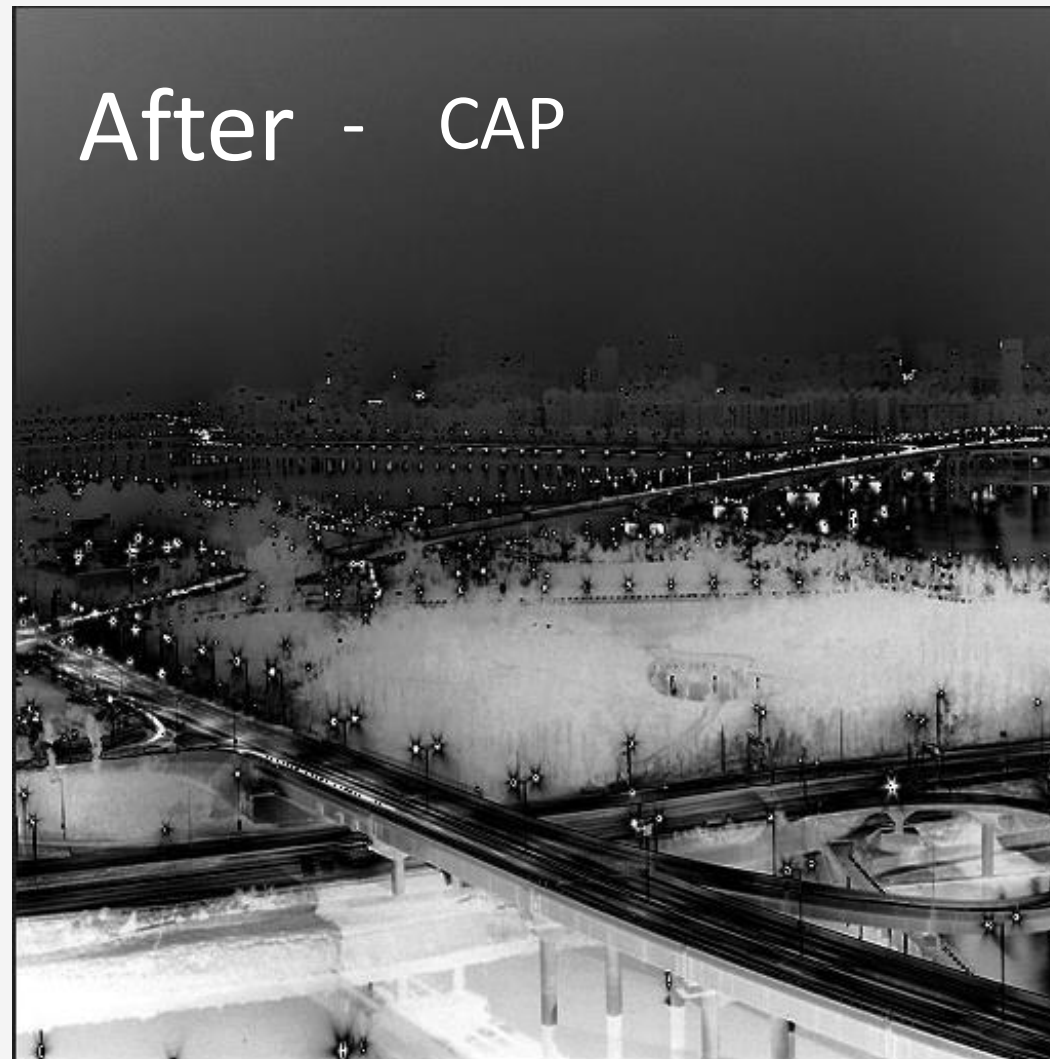
정수 값 --> 2



```
outImage[i][k] = pow((float)inImage[i][k], 1.f / val);
```

화면 구성 및 기능

A 화소점처리 > E 파라볼라 변환



CAP

```
255.f * pow(( inImage[i][k] / 127.f) - 1, 2.f);
```

CUP

```
255.f - 255.f * pow(( inImage[i][k] / 127.f) - 1, 2.f);
```

화면 구성 및 기능

A 화소점처리 > F 흑백 변환(127)

Before



After



```
if ( inImage[i][k] < 128 )  
    outImage[i][k] = 0;  
else outImage[i][k] = 255;
```


화면 구성 및 기능

A 화소점처리 > G 흑백 변환(평균값)

Before



평균 값 : 68

After



```
cMid += inImage[i][k];  
cMid = cMid / (inH * inW);  
if (inImage[i][k] < cMid)  
    outImage[i][k] = 0;  
else outImage[i][k] = 255;
```


화면 구성 및 기능

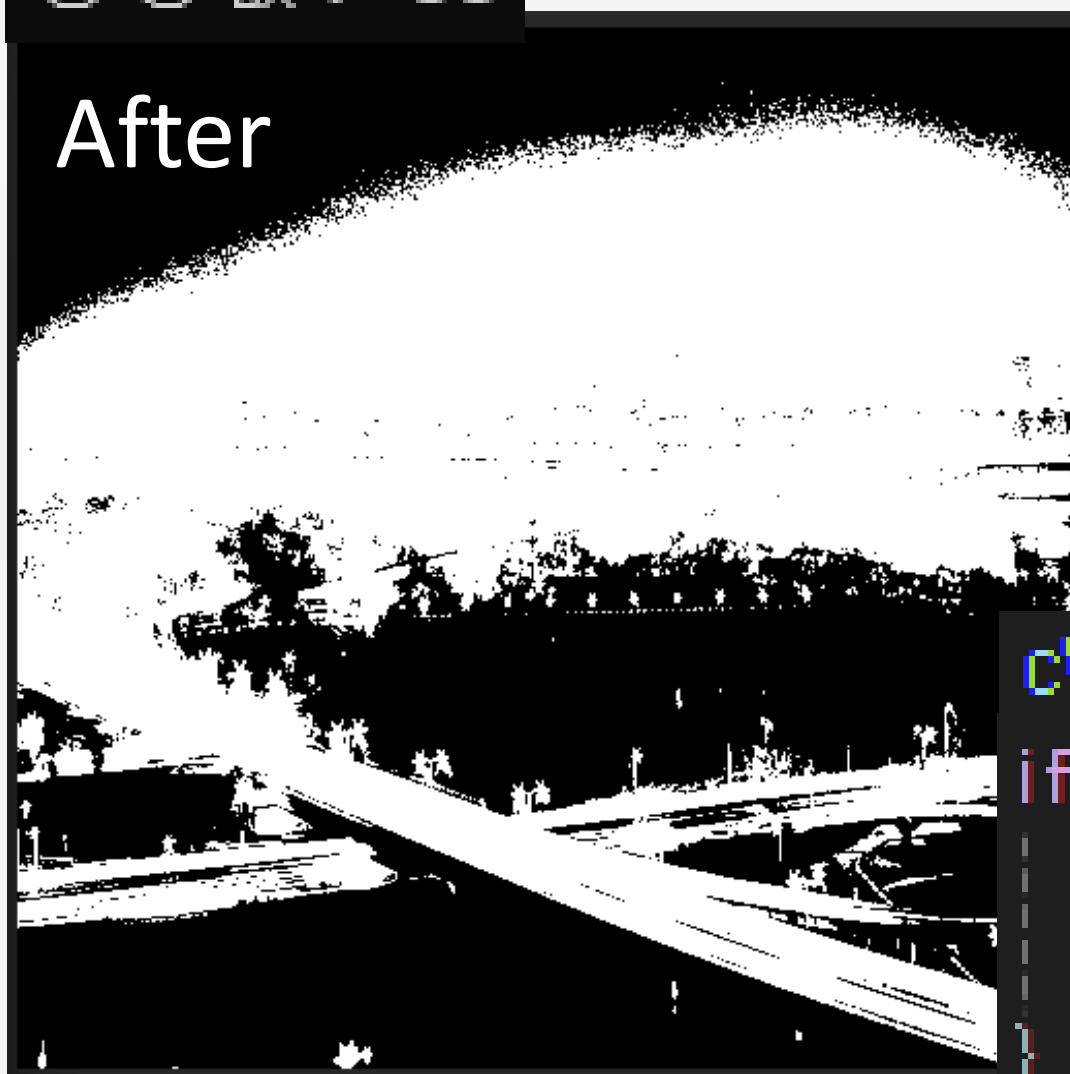
A 화소점처리 > H 흑백 변환(중앙값)

Before



중앙값 : 65

After

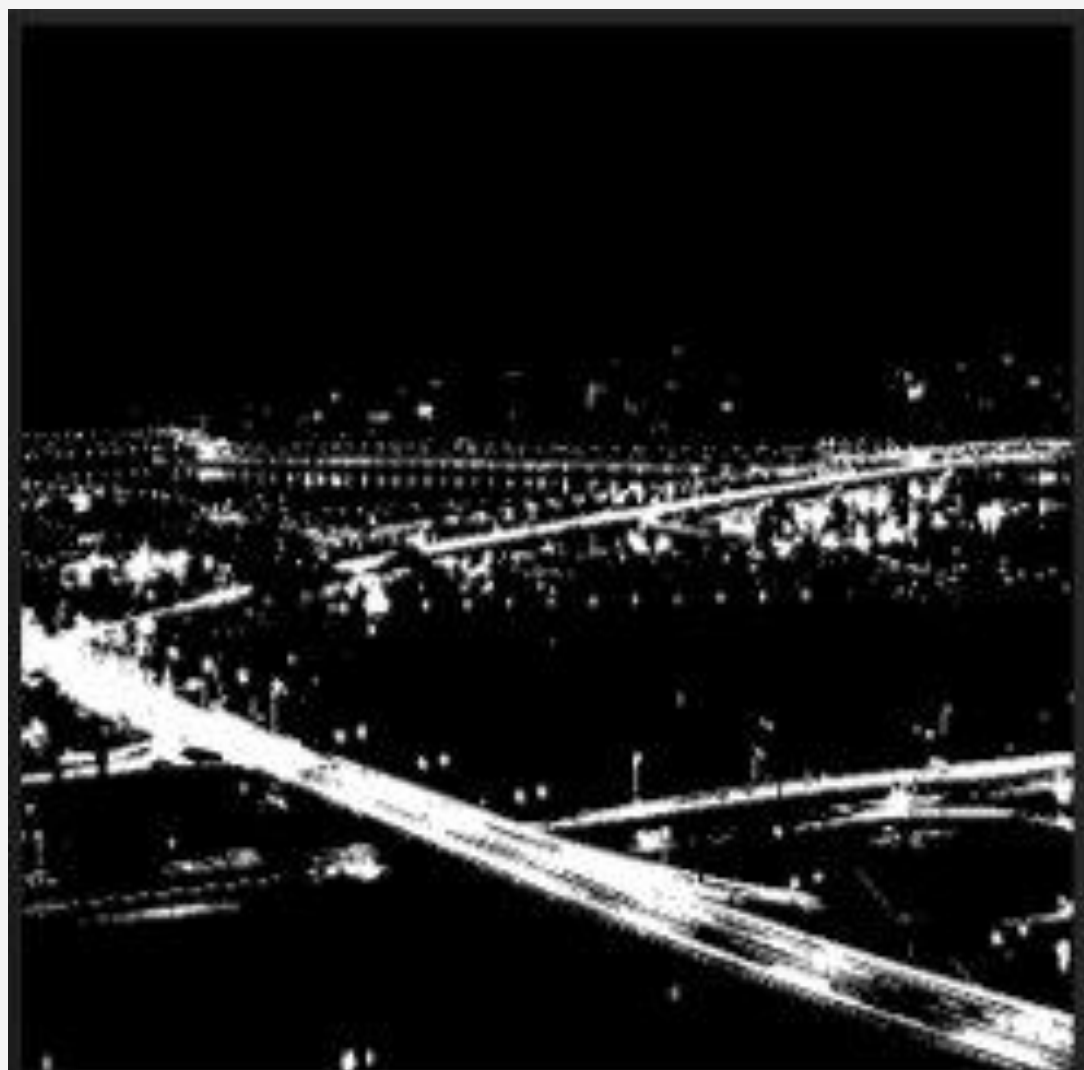


```
cValCount[inImage[i][k]] += 1;  
if (count > (inW * inH) / 2) {  
    cMid = i;  
    break;  
}
```

```
if (inImage[i][k] < cMid)  
    outImage[i][k] = 0;  
else outImage[i][k] = 255;
```

화면 구성 및 기능

A 화소점처리 中 흑백 변환 차이



F (기준: 127)



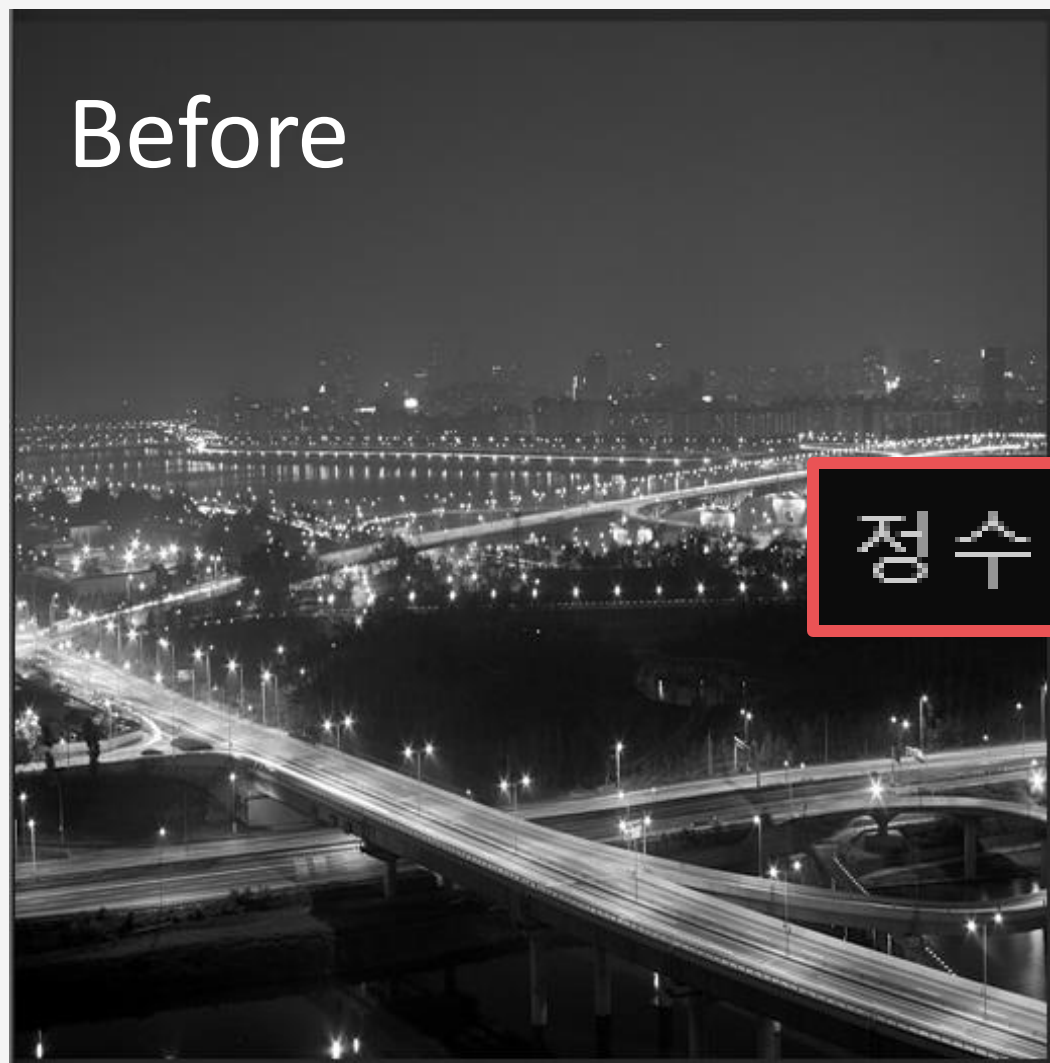
G (기준: 평균값)



H (기준: 중앙값)

화면 구성 및 기능

A 화소점처리 > 1 더 밝게 하기(배수)



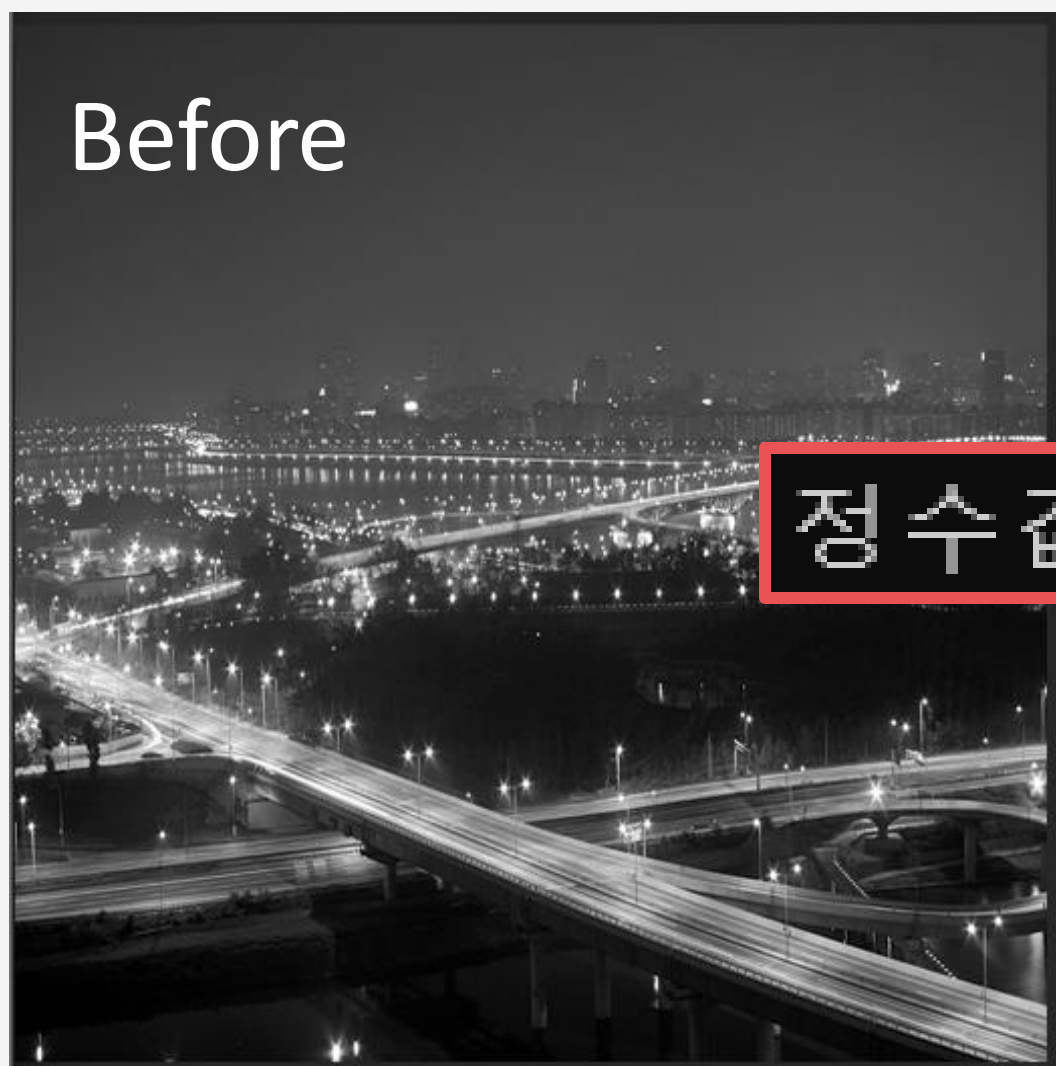
정수값 --> 2



```
if (inImage[i][k] * val > 255) outImage[i][k] = 255;  
else outImage[i][k] = inImage[i][k] * val;
```

화면 구성 및 기능

A 화소점처리 > J 더 어둡게 하기(배수)



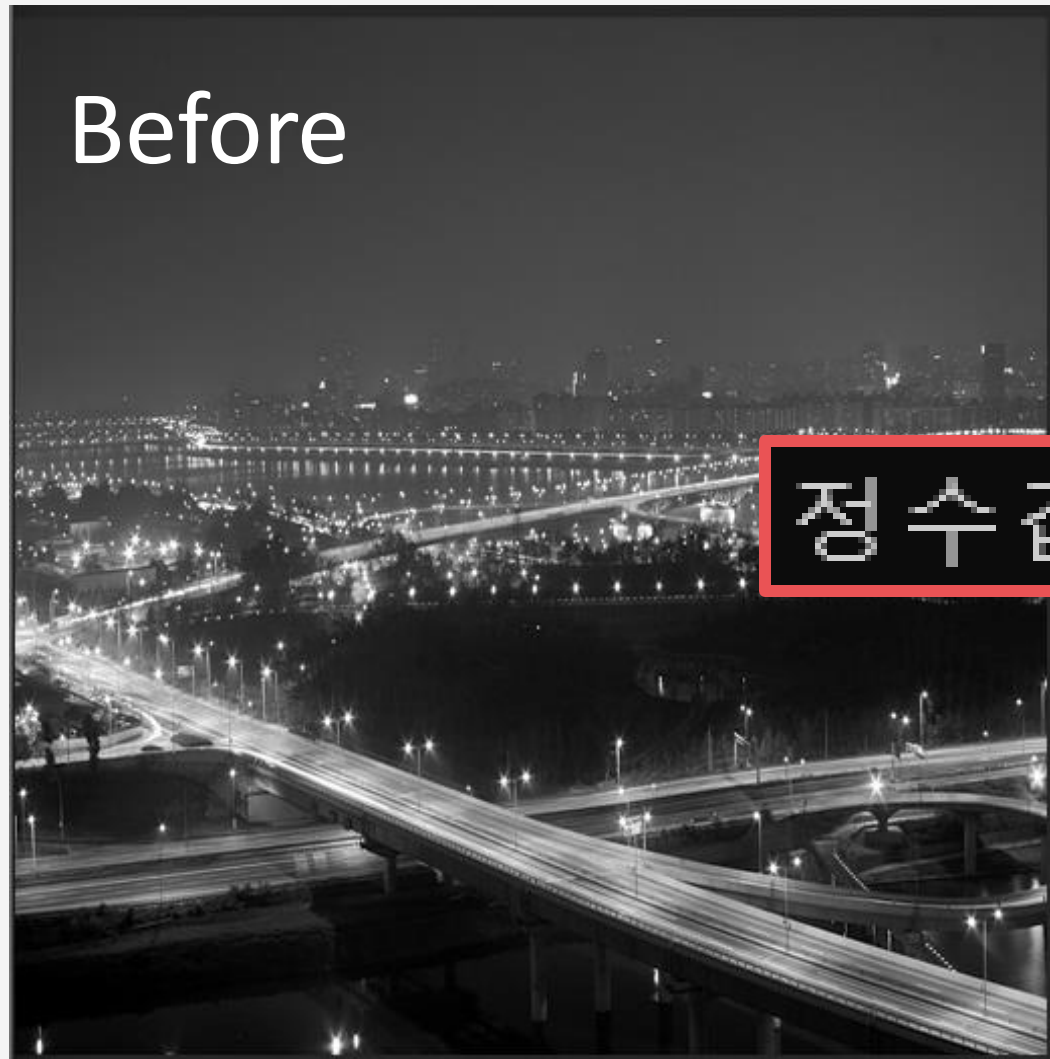
정수값 --> 3



```
in Image[i][k] / val;
```


화면 구성 및 기능

A 화소점처리 > K 더 밝게(배수)+랩핑기법



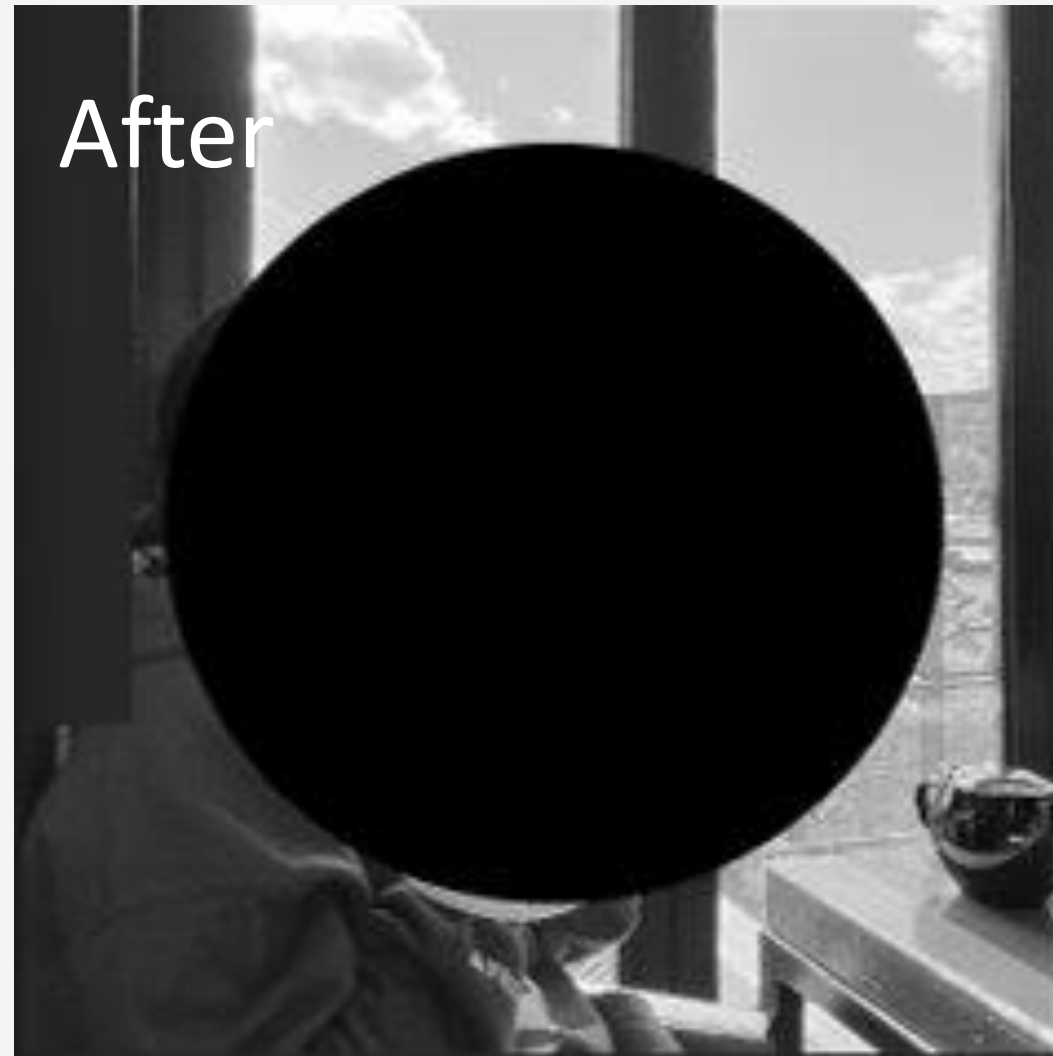
정수값 --> 3



```
outImage[i][k] = (inImage[i][k] + val) % 256;
```

화면 구성 및 기능

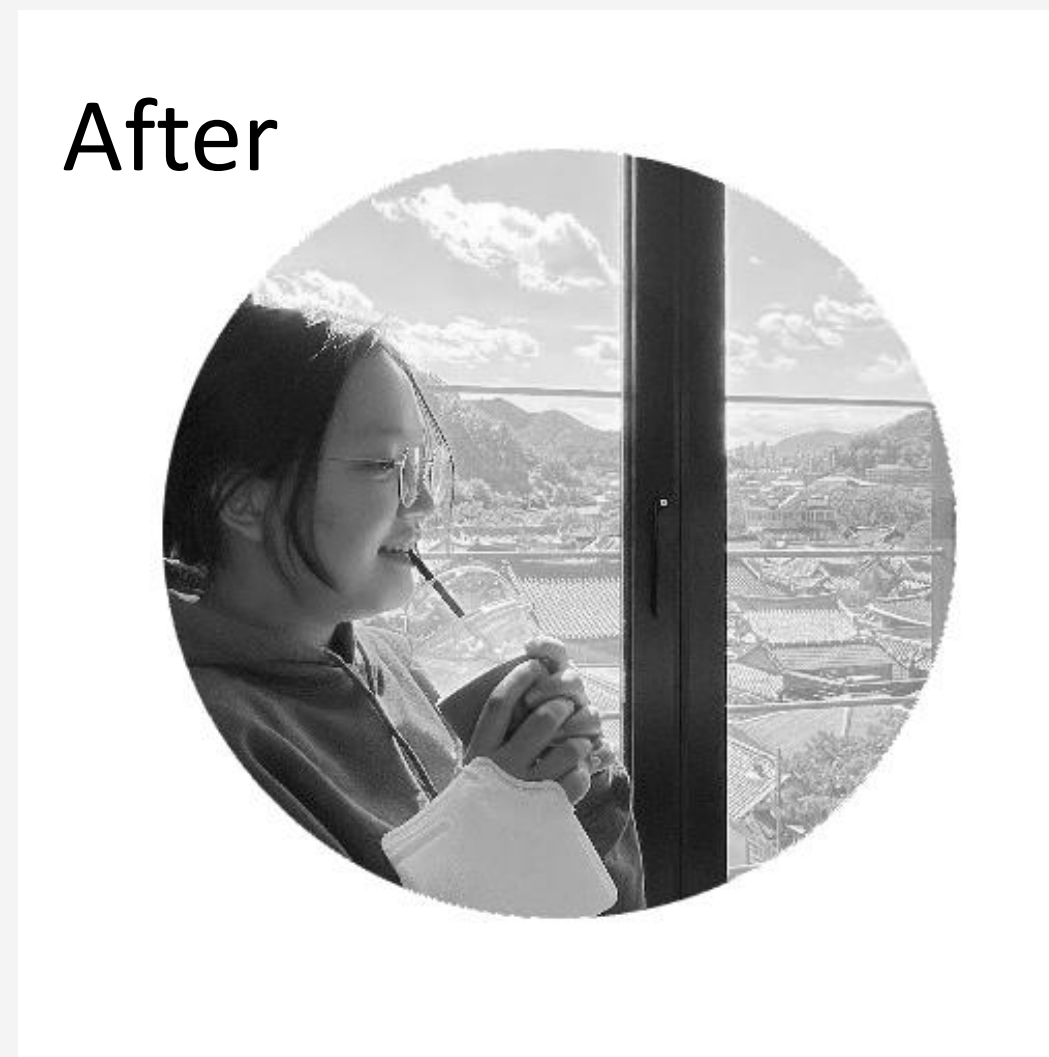
A 화소점처리 > L 원형 마스크(AND)



```
inImage[i][k] & mask[i][k];
```

| 화면 구성 및 기능

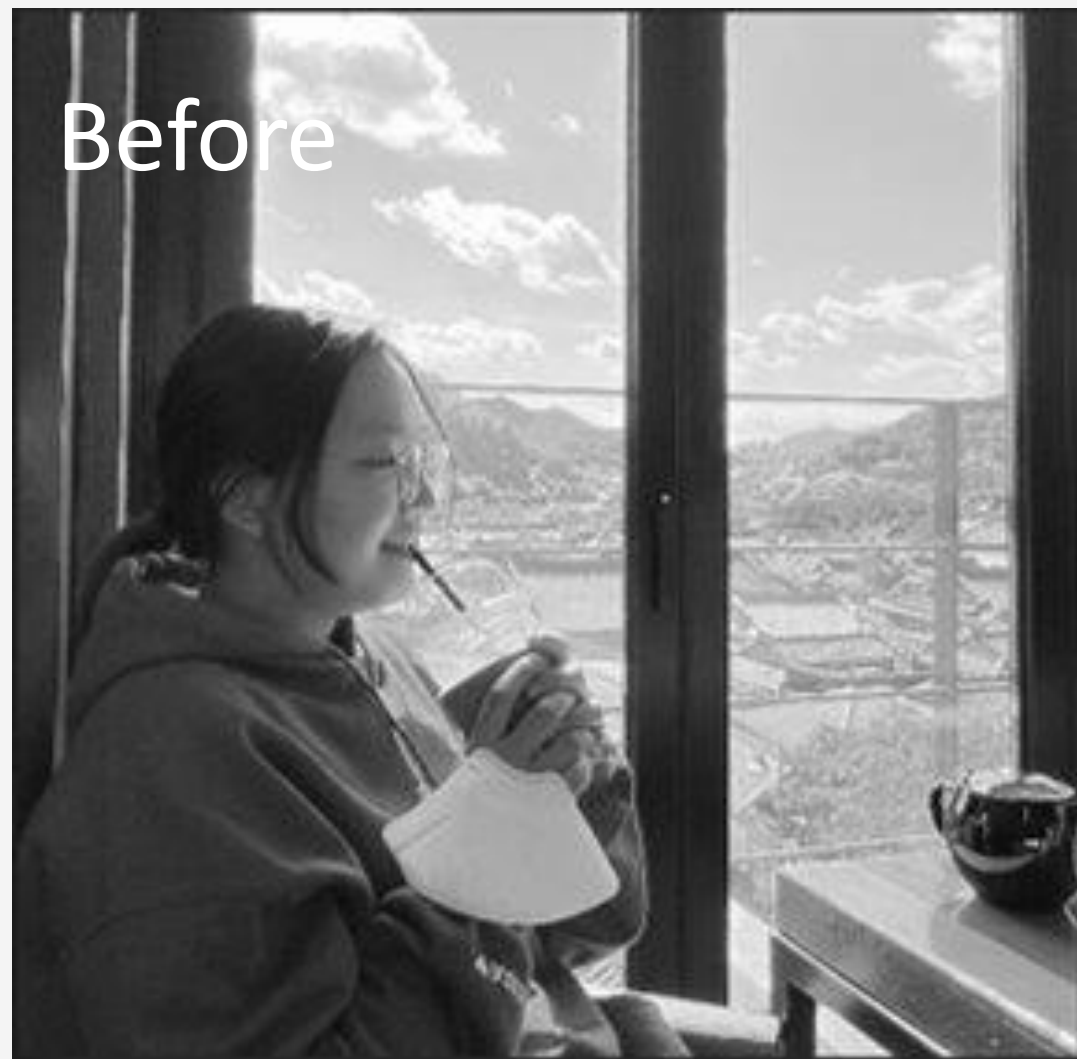
A 화소점처리 > M 원형 마스크(OR)



```
in Image[i][k] | mask[i][k];
```

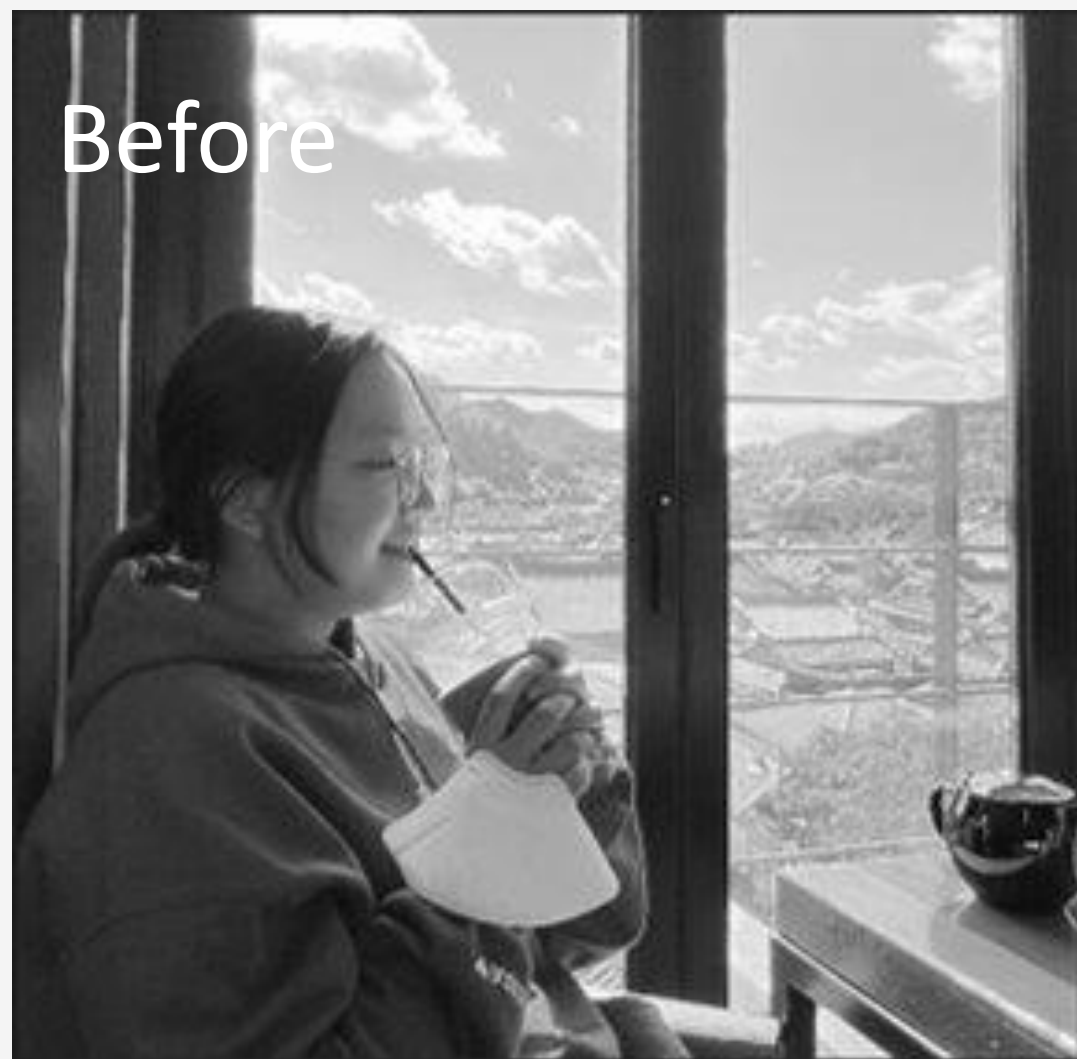
화면 구성 및 기능


A 화소점처리 > N 원형 마스크(XOR)



```
inImage[i][k] ^ mask[i][k]
```

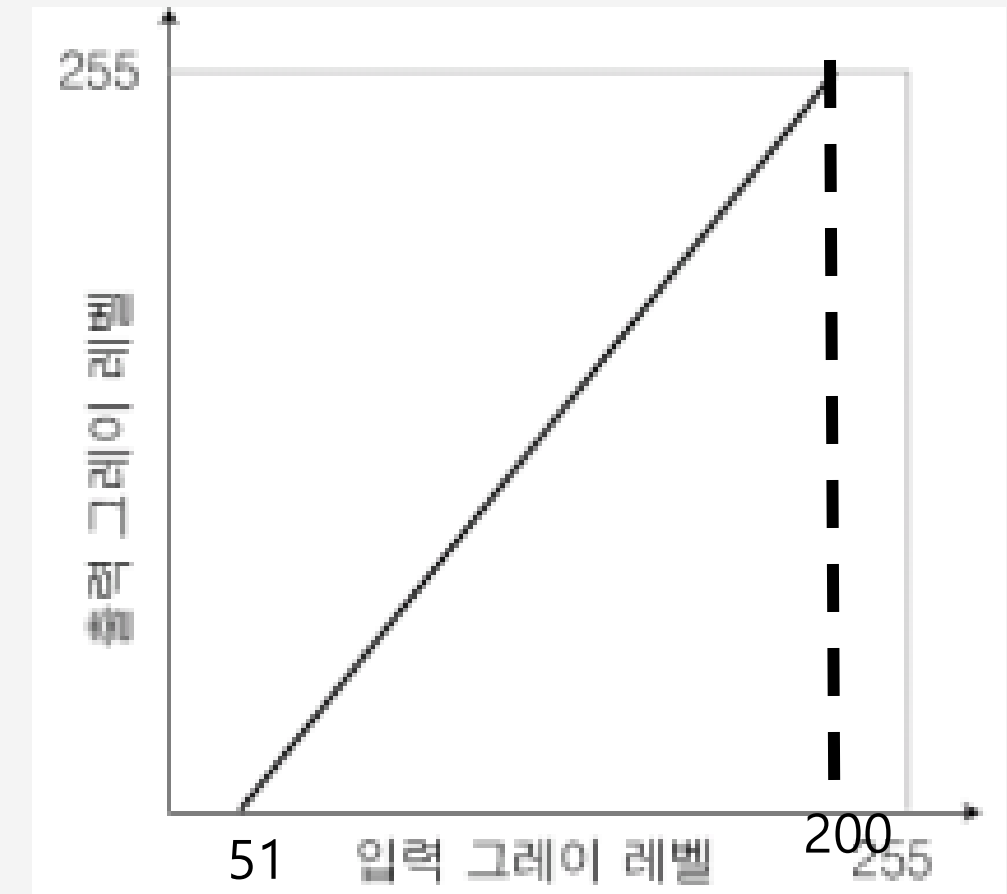

화면 구성 및 기능 A 화소점처리 > O NOT 반전



inImage[i][k].

화면 구성 및 기능

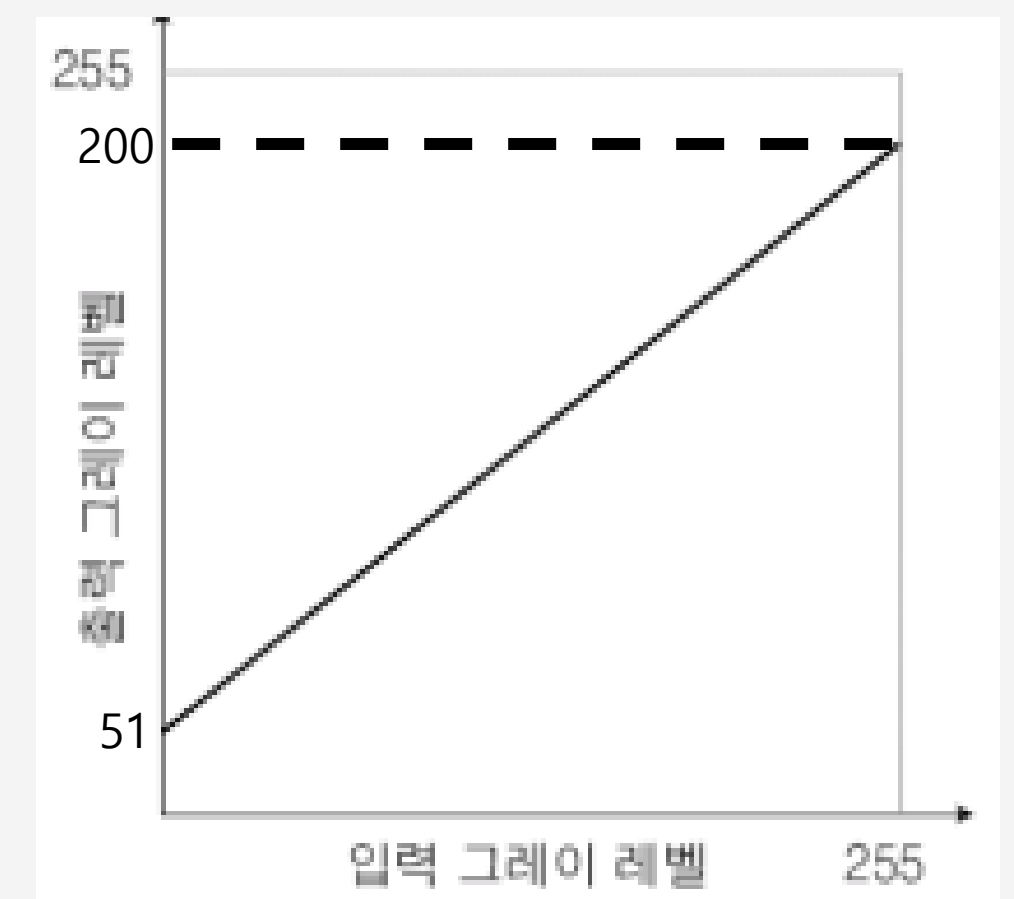
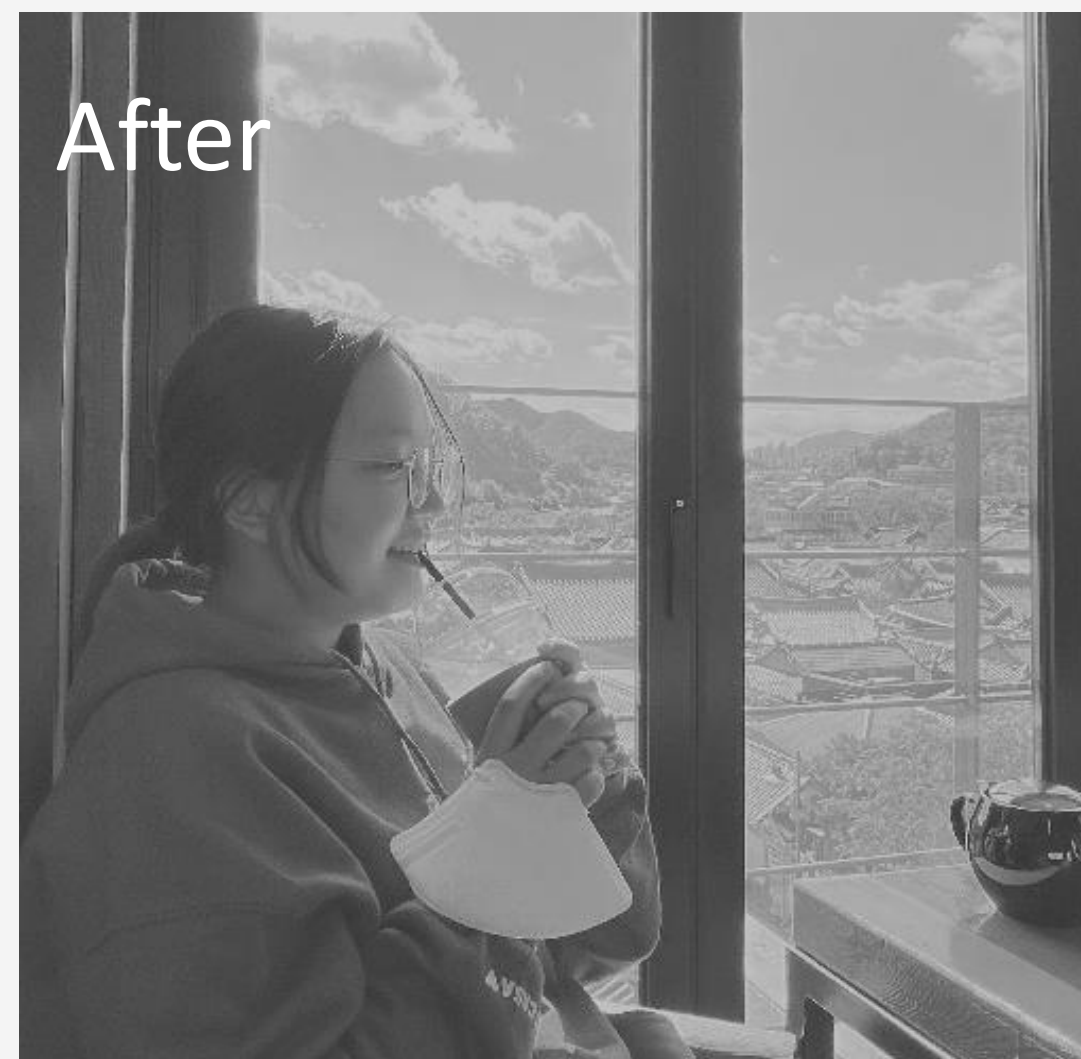
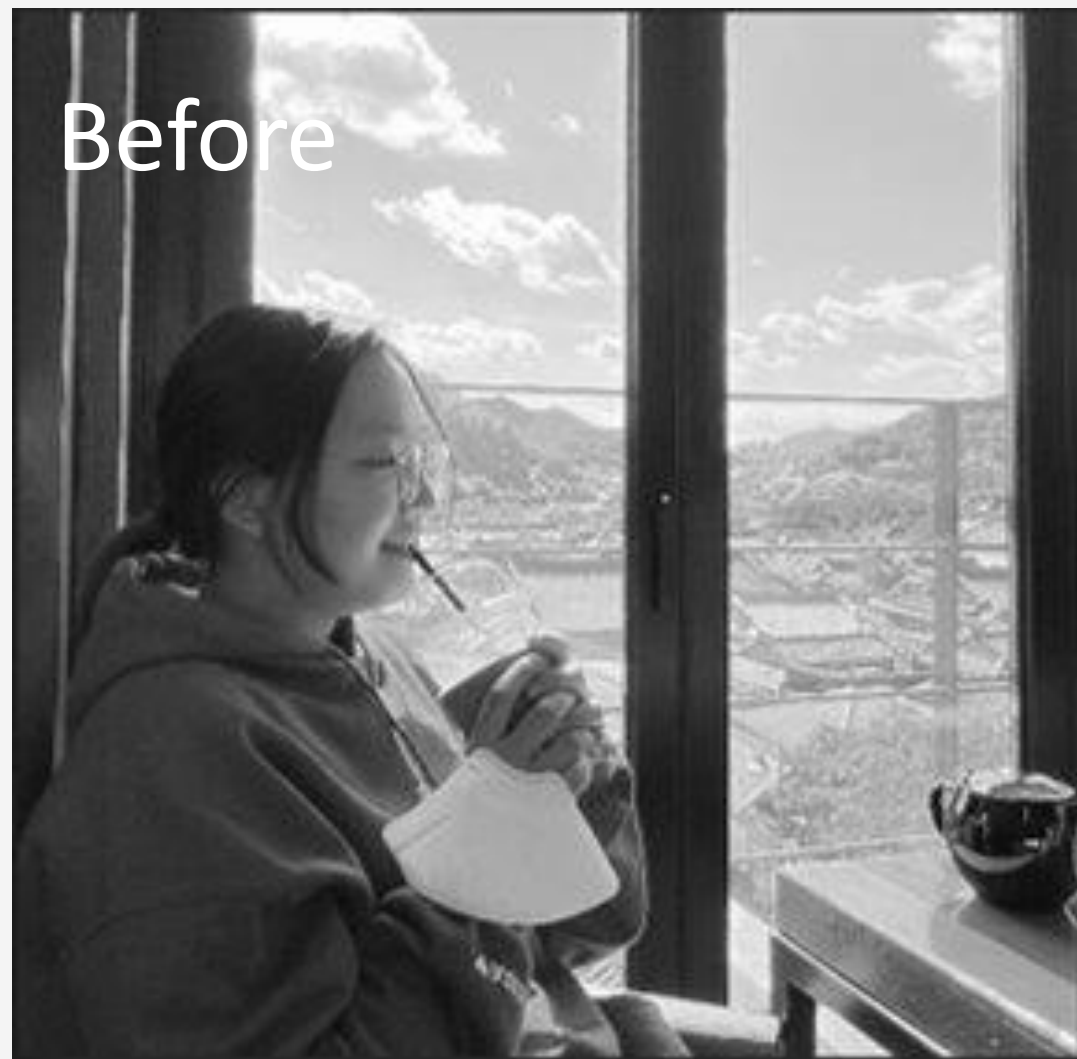
A 화소점처리 > P 명함 대비 스트레칭



```
if (inImage[i][k] < 50) outImage[i][k] = 0;  
else if (inImage[i][k] > 200) outImage[i][k] = 255;  
else outImage[i][k] = inImage[i][k] * 51 / 30 - 85;
```

화면 구성 및 기능

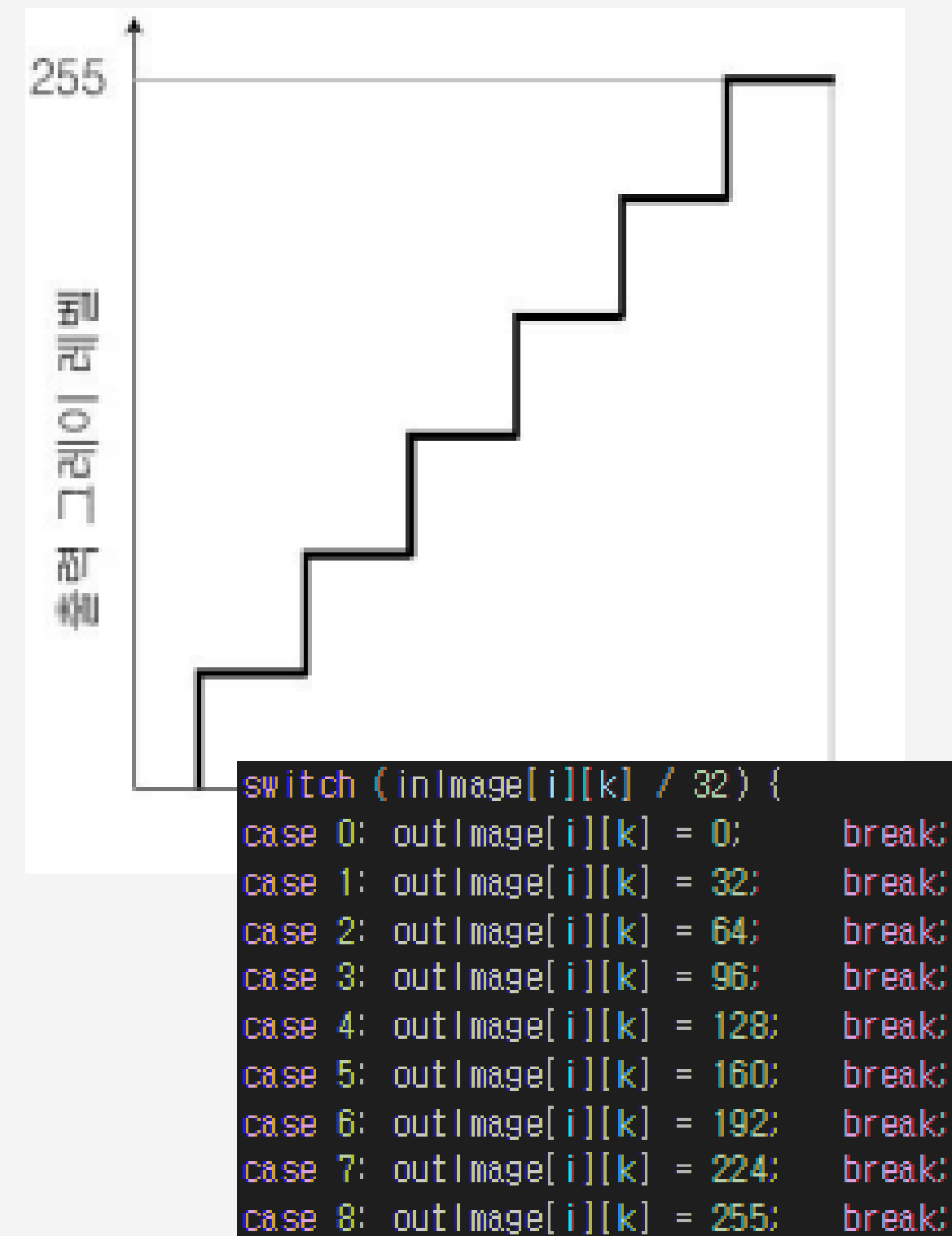
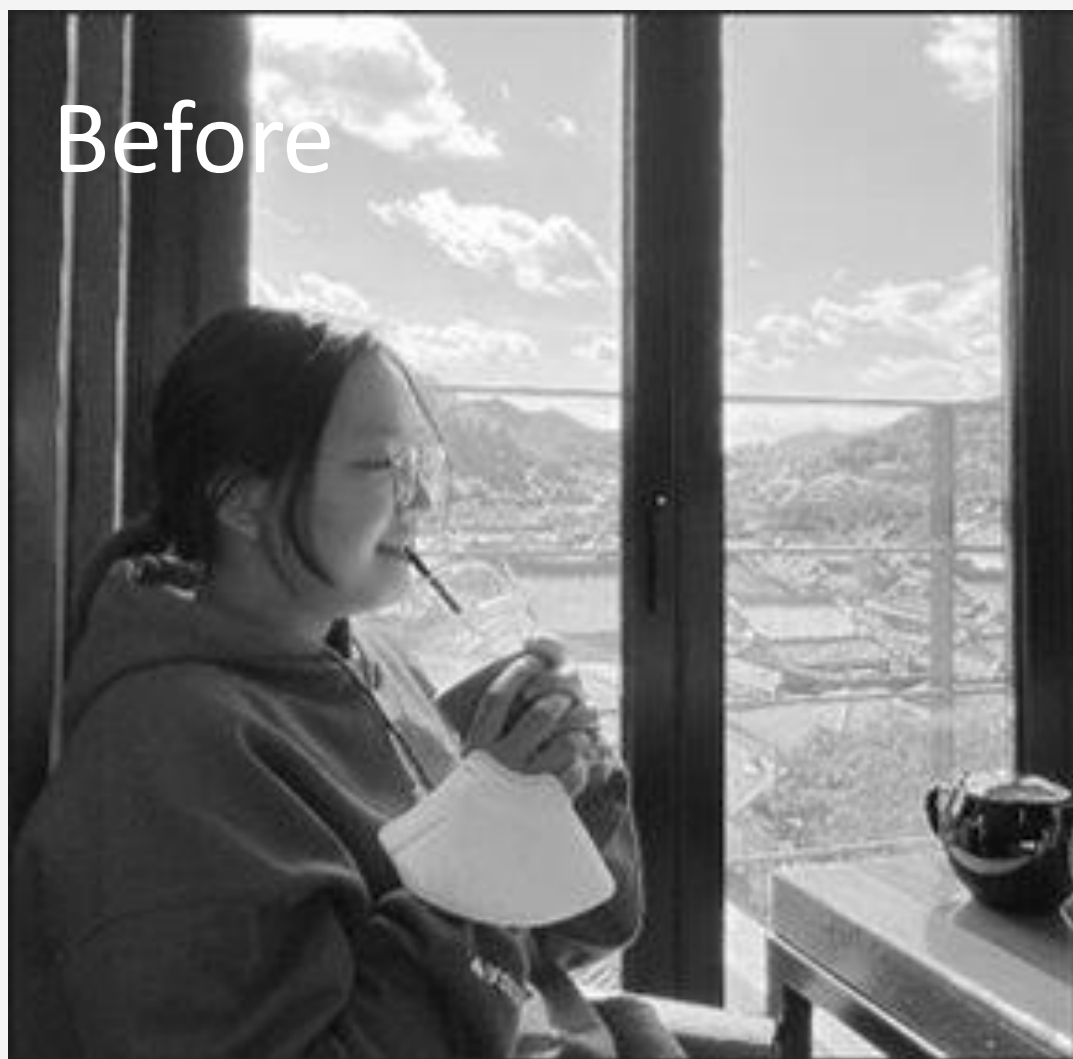
A 화소점처리 > Q 명함 대비 압축



```
inImage[i][k] = 30 / 51 + 50;
```

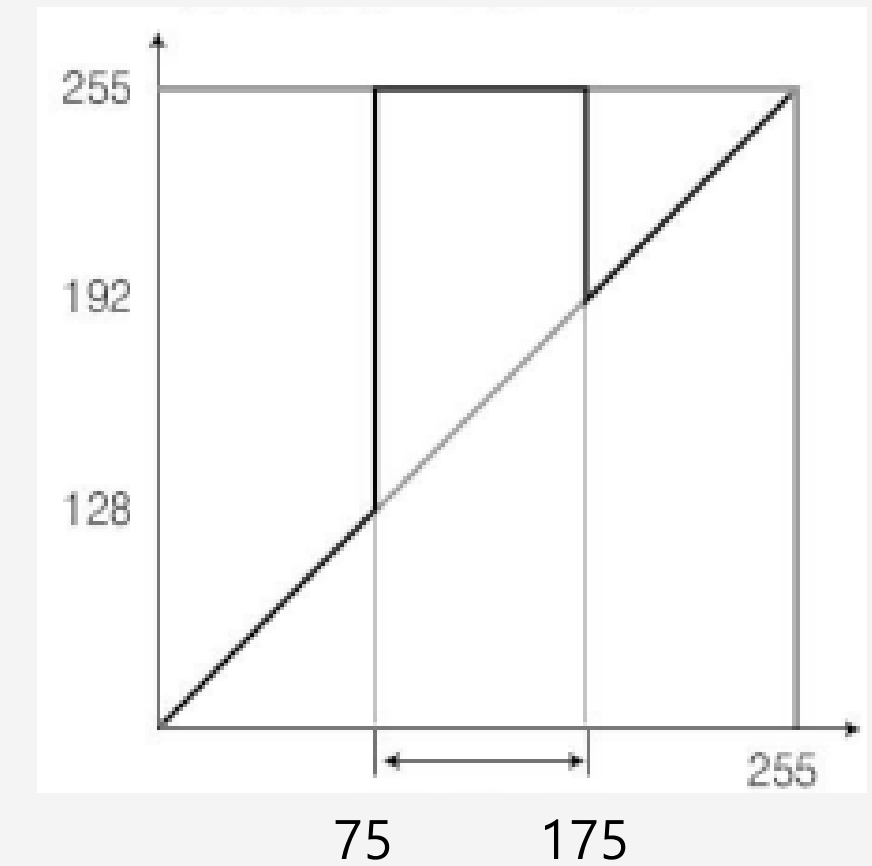
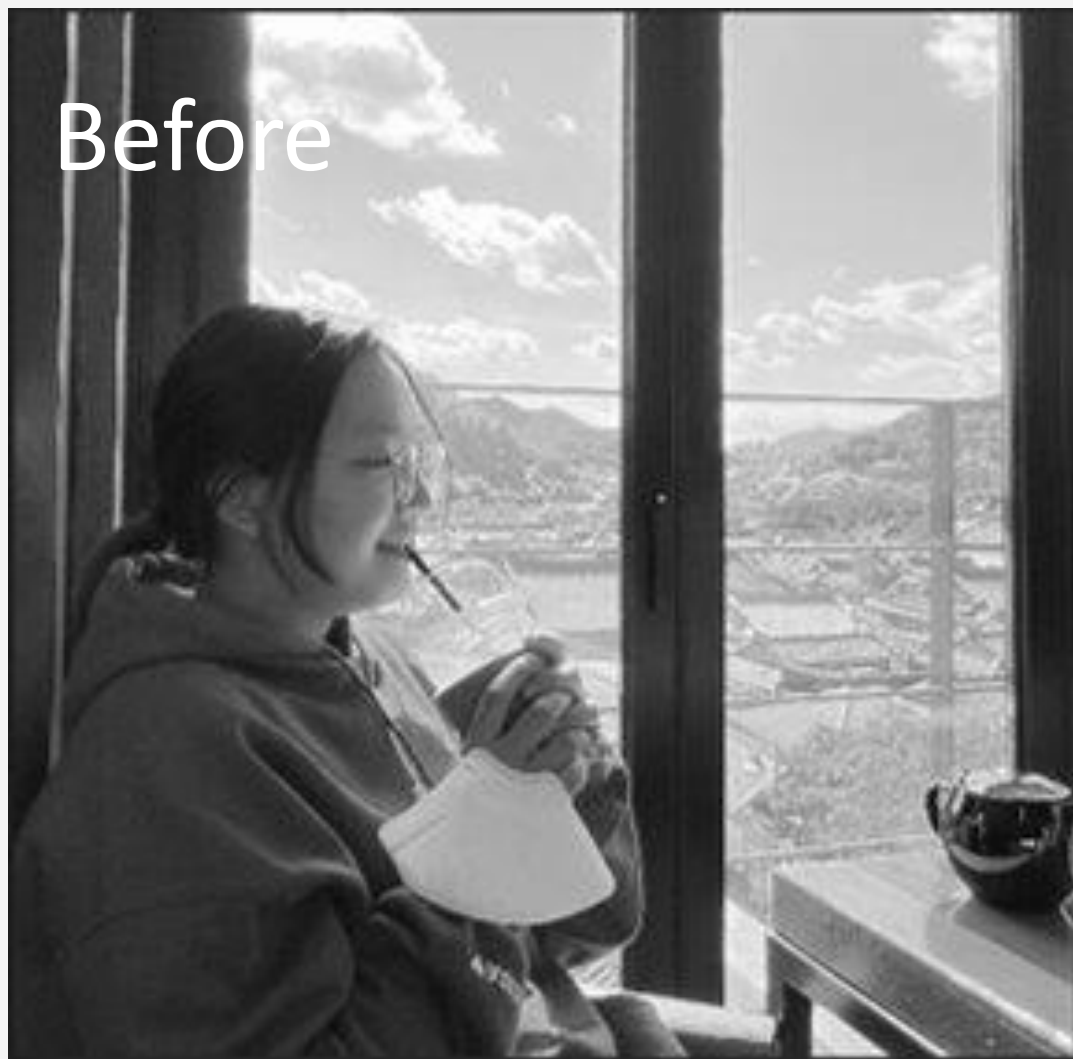
화면 구성 및 기능

A 화소점처리 > R 명함 대비 압축



화면 구성 및 기능

A 화소점처리 > S 범위 강조



```
if (inImage[i][k] < 75) outImage[i][k] = inImage[i][k];  
else if (inImage[i][k] > 175) outImage[i][k] = inImage[i][k];  
else outImage[i][k] = 255;
```

화면 구성 및 기능

B 기하학 처리 메뉴

D:\MyProject\W0318\Wx64\De

+

▼

—

□

×

A. 줌 인 (전 방향)

B. 줌 인 (이웃 화소 보간법)

C. 줌 인 (선형 보간법)

D. 줌 아웃

E. 줌 아웃 (평균)

F. 줌 아웃 (중앙값)

G. 회전 (깨짐)

H. 회전 (안 깨짐)

I. 회전 (안 깨짐 + 안 찢림)

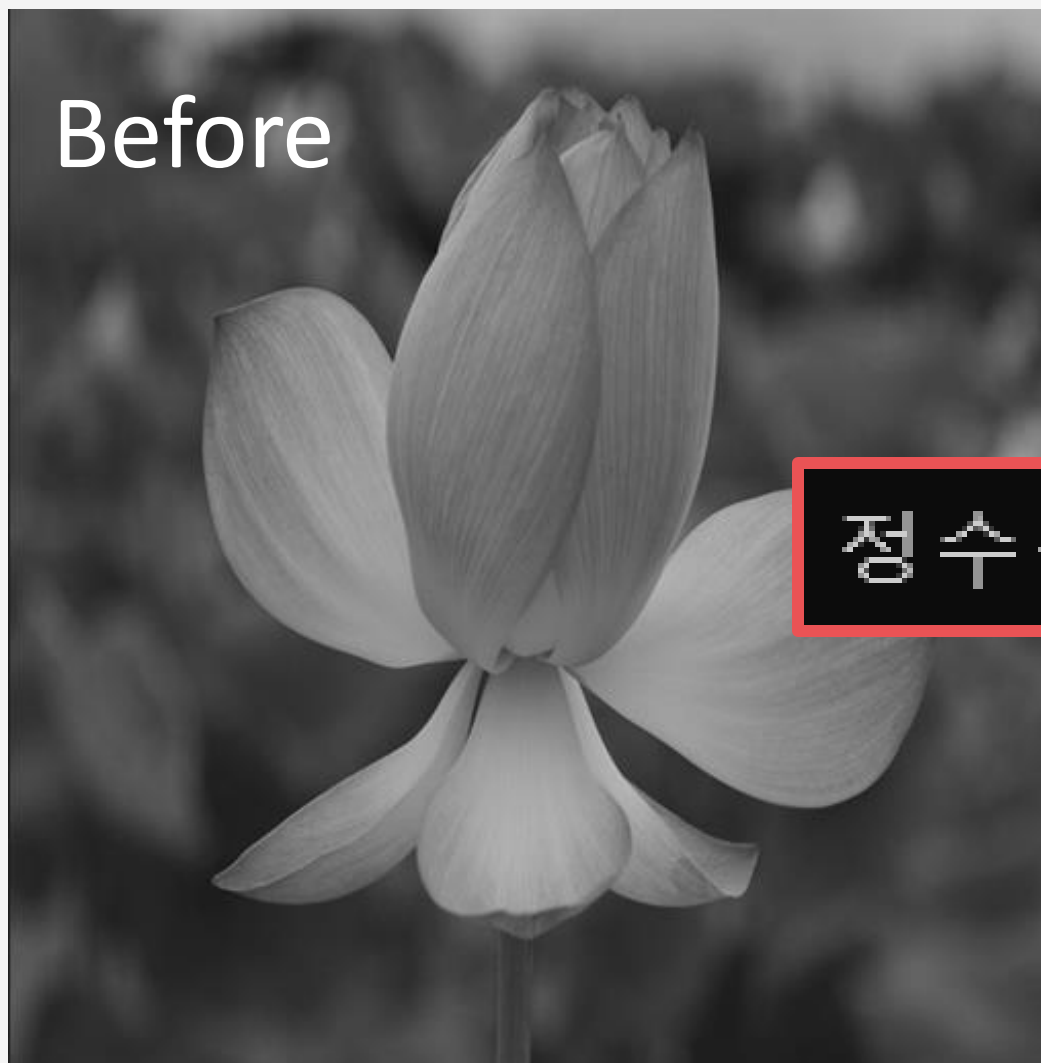
J. 세로 이동

K. 가로 이동 L. 미러링

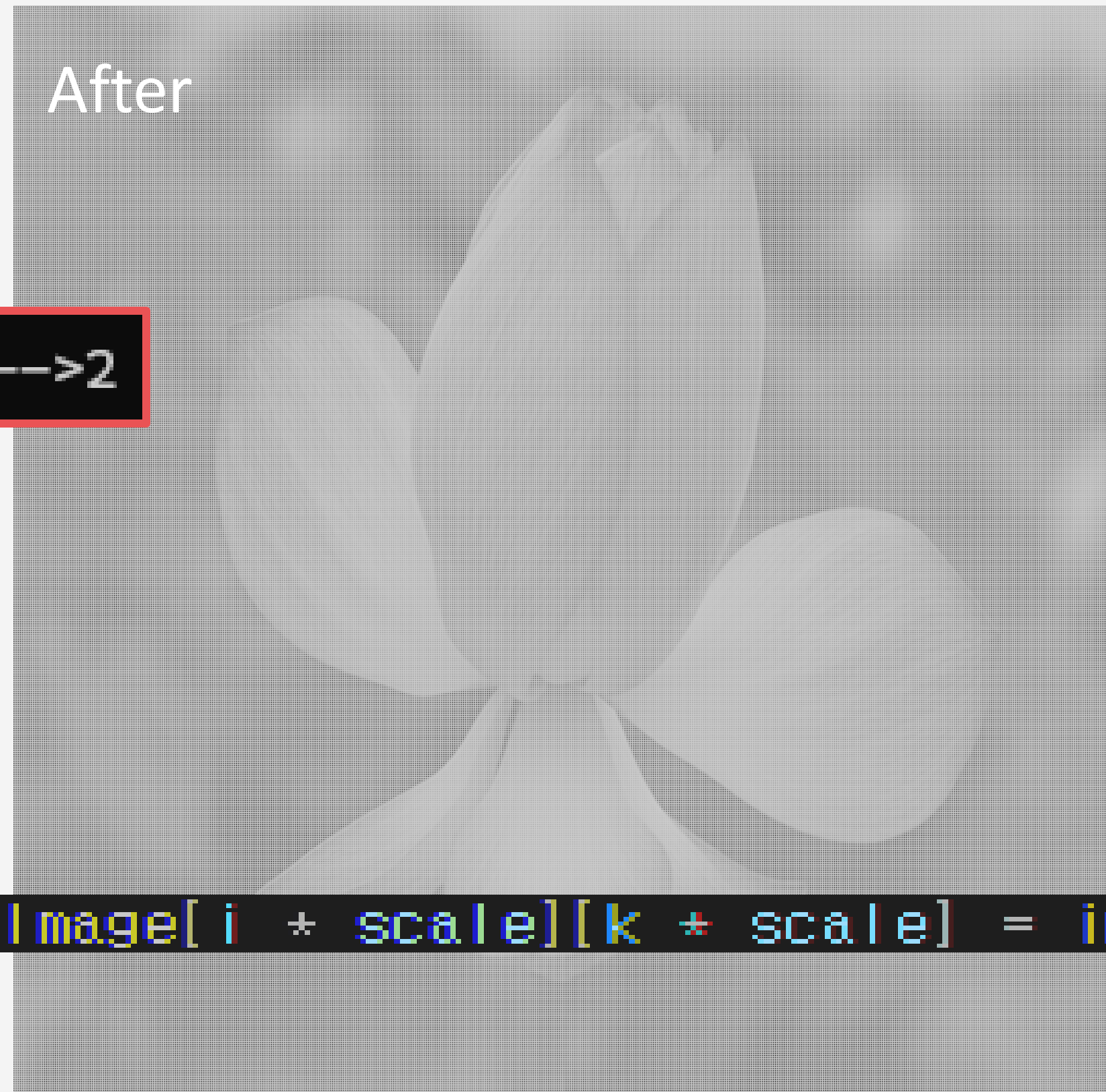
M. 모핑

화면 구성 및 기능

B 기하학처리 > A 줌인(전방향)



정수값 --> 2

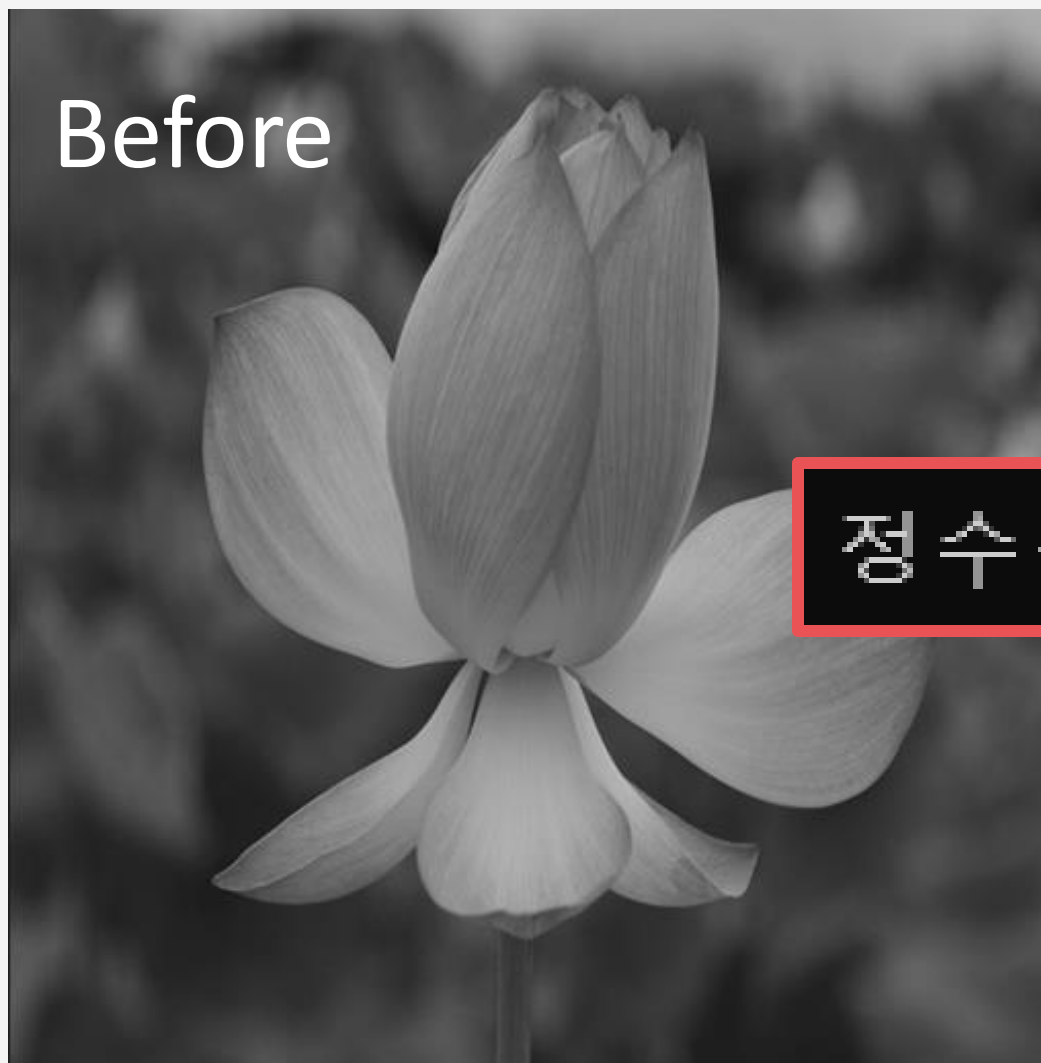


2배로 커졌지만
이미지가 흐려지는 특징

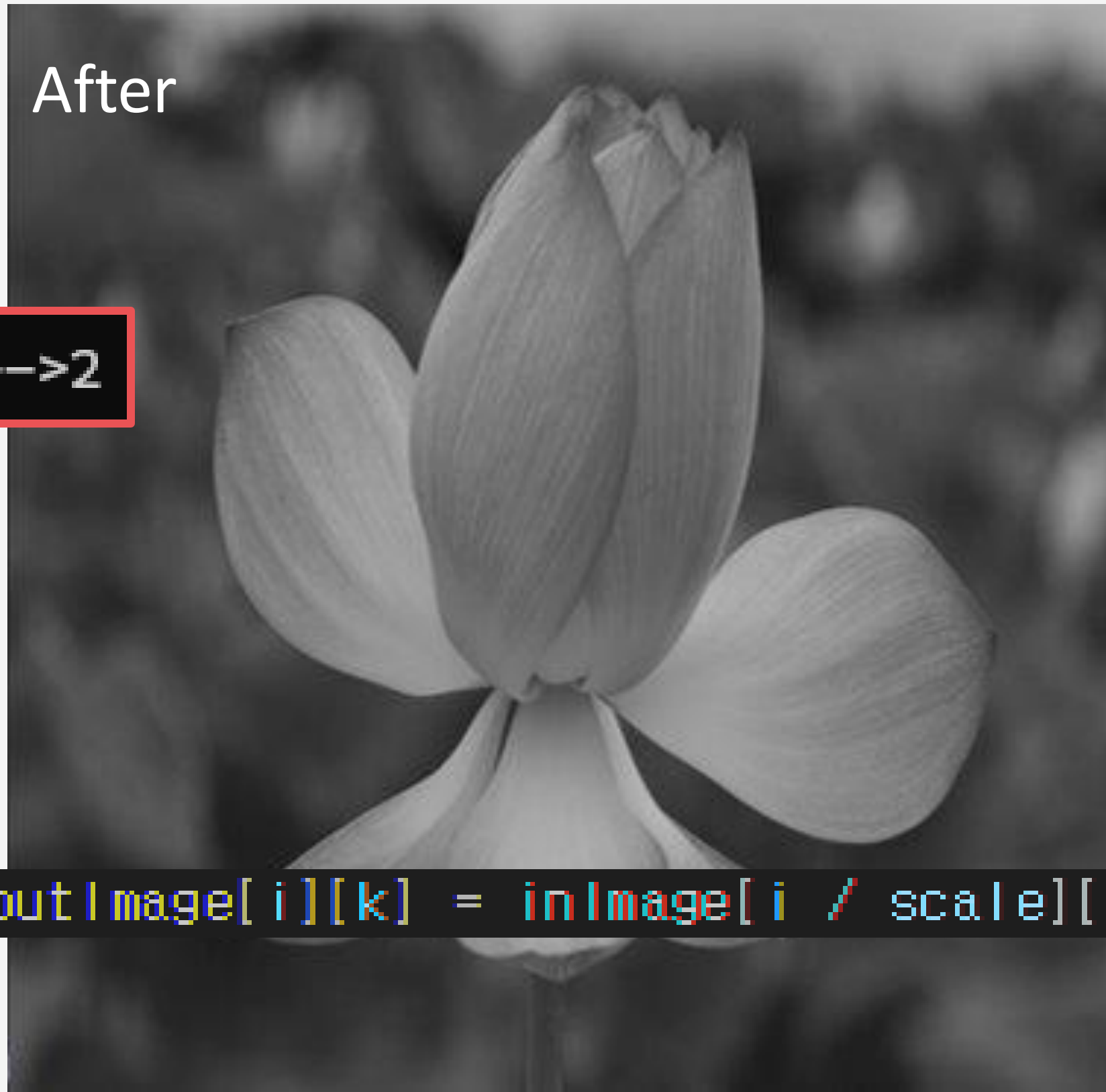
```
outImage[i + scale][k + scale] = inImage[i][k];
```


화면 구성 및 기능

B 기하학처리 > B 줌인(이웃화소보간법)



정수값-->2

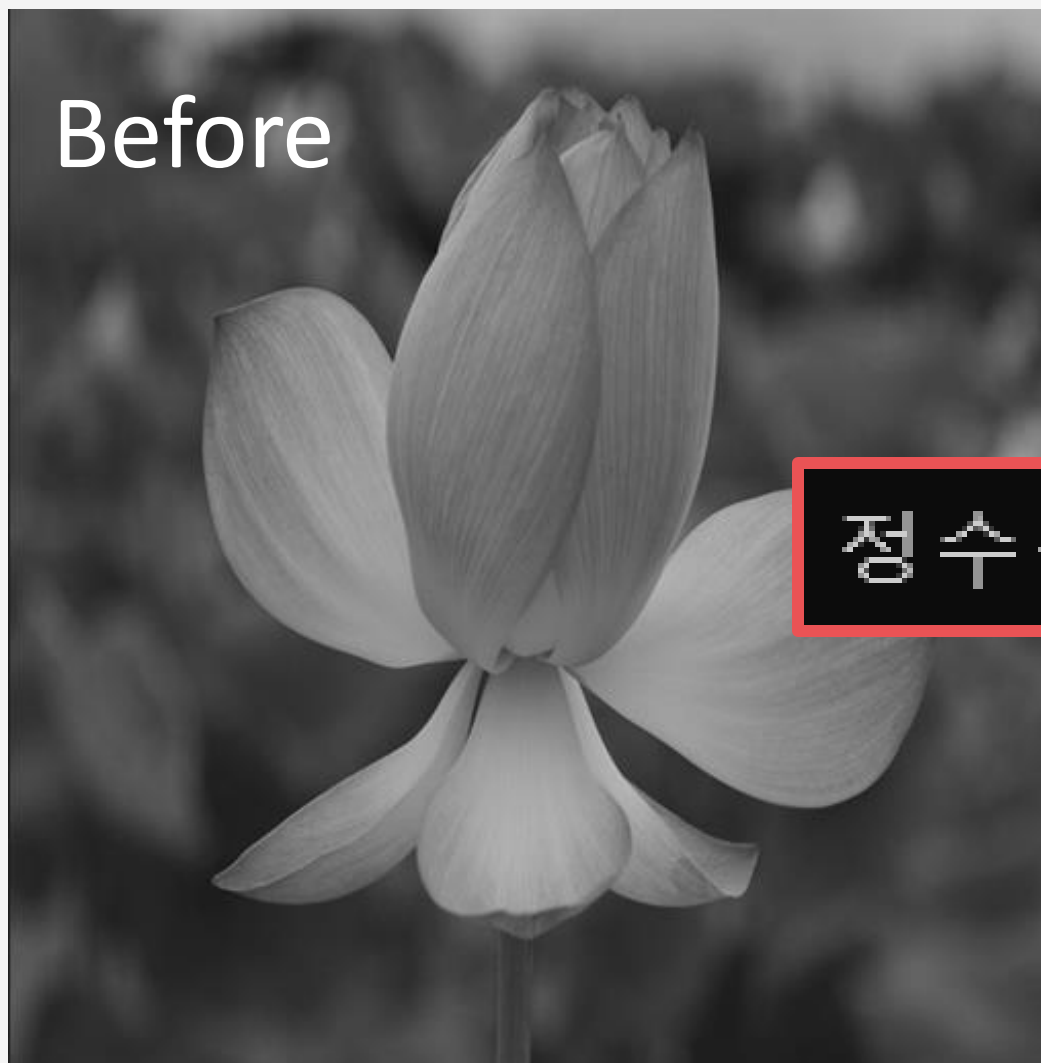


2배로 커졌지만
모서리가 각져 보이는 특징

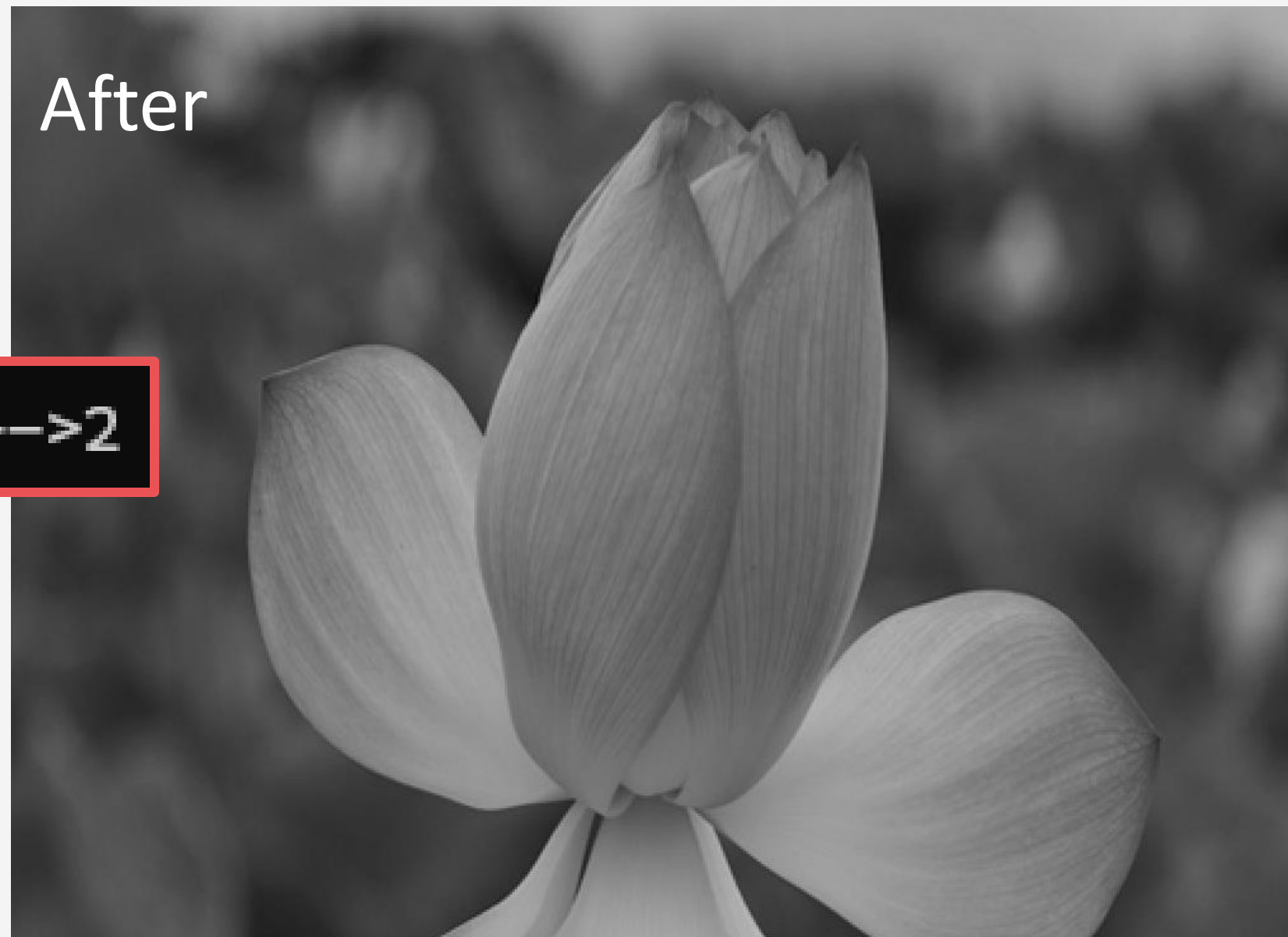
```
outImage[i][k] = inImage[i / scale][k / scale];
```


화면 구성 및 기능

B 기하학처리 > C 줌인(선형 보간법)



정수 값 --> 2



확대 했을 때
가장 자연스러움

```
// 각 점들과의 위치 비율을 반영해서 점의 색 지정
outImage[i][k] = (unsigned char)(C1 * (1 - s_H) * (1 - s_W)
    + C2 * s_W * (1 - s_H) + C3 * s_W * s_H + C4 * (1 - s_W) * s_H);
```

| 화면 구성 및 기능

B 기하학처리 中 줌아웃



Before

정수 값 --> 2



D 줌아웃

이미지가 각져짐



E 줌아웃(평균)



F 줌아웃(중앙값)

둘 모두 손실을 최소화한 방법으로
내부 손실값 처리 방식의 차이

D 줌아웃

```
outImage[(int)(i / scale)][(int)(k / scale)]  
= inImage[i][k];
```

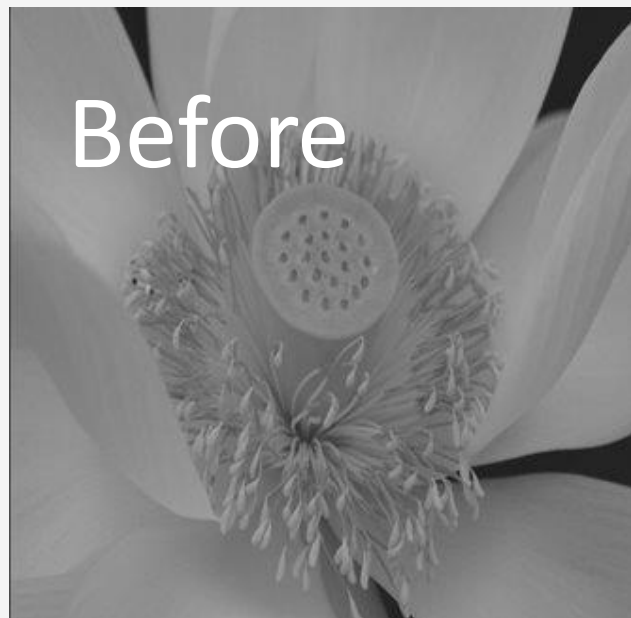
E 줌아웃(평균)

```
tmp[(int)(i / scale)][(int)(k / scale)] += inImage[i][k];  
outImage[i][k] = (int)(tmp[i][k] / scale);
```

F 줌아웃(중앙값)

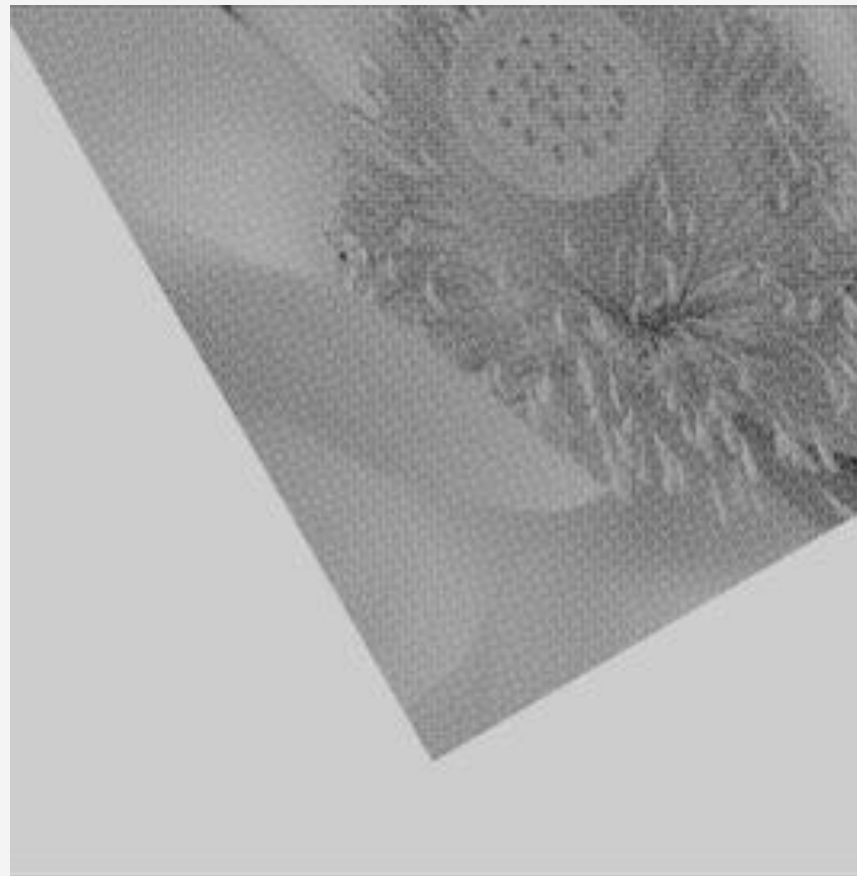
```
qsort(tmp, (sizeof(tmp) / sizeof(tmp[0])), sizeof(tmp[0]), compare);  
outImage[i][k] = tmp[scale * scale / 2];
```

| 화면 구성 및 기능 B 기하학처리 中 회전



Before

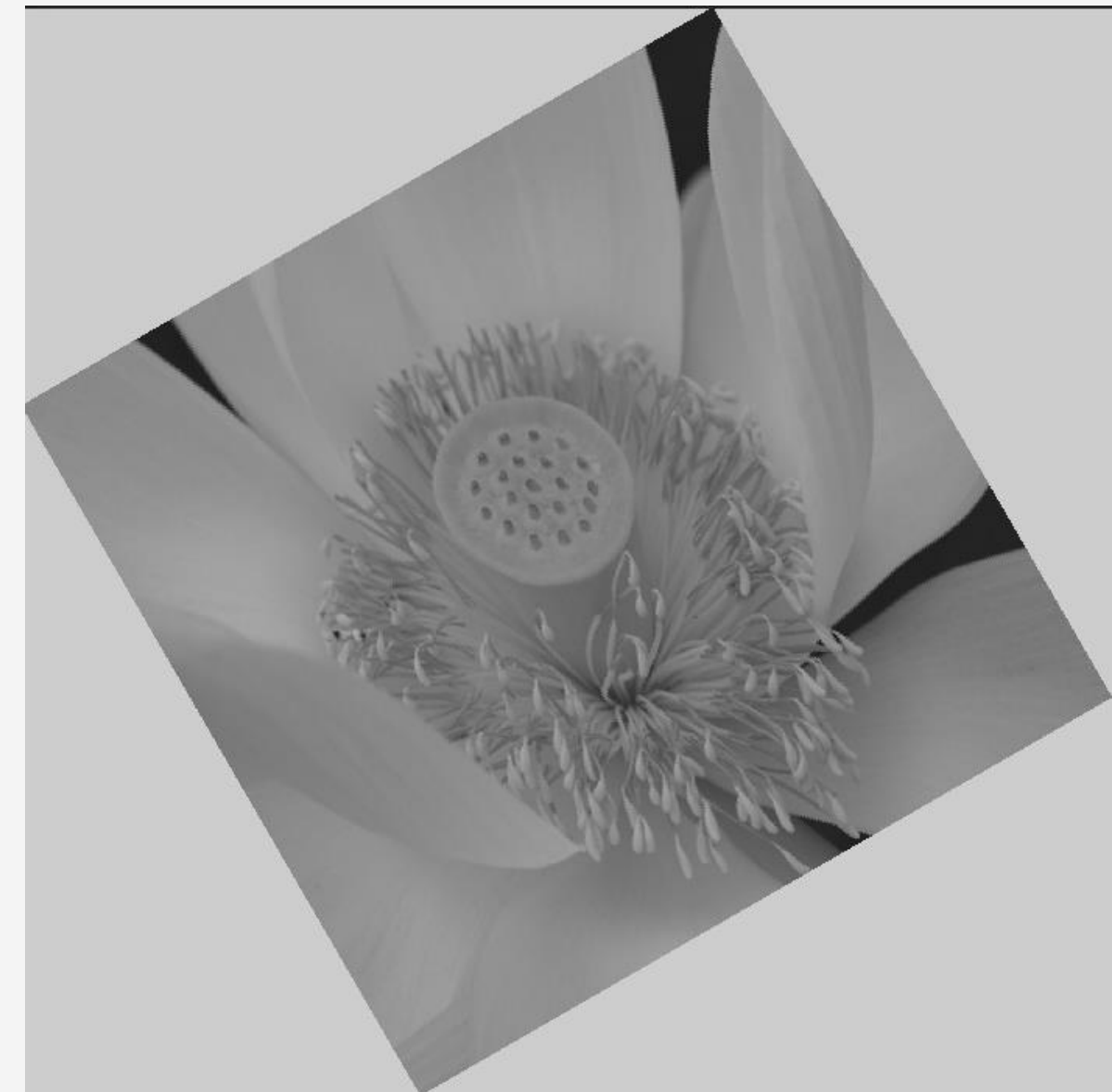
정수값 --> 30



G 회전(깨짐)



H 회전(안깨짐)



I 회전(안깨짐 + 안잘림)

G 회전

```
(int)(cos(radian) * xs - sin(radian) * ys);
(int)(sin(radian) * xs + cos(radian) * ys);
```

H 회전(안깨짐)

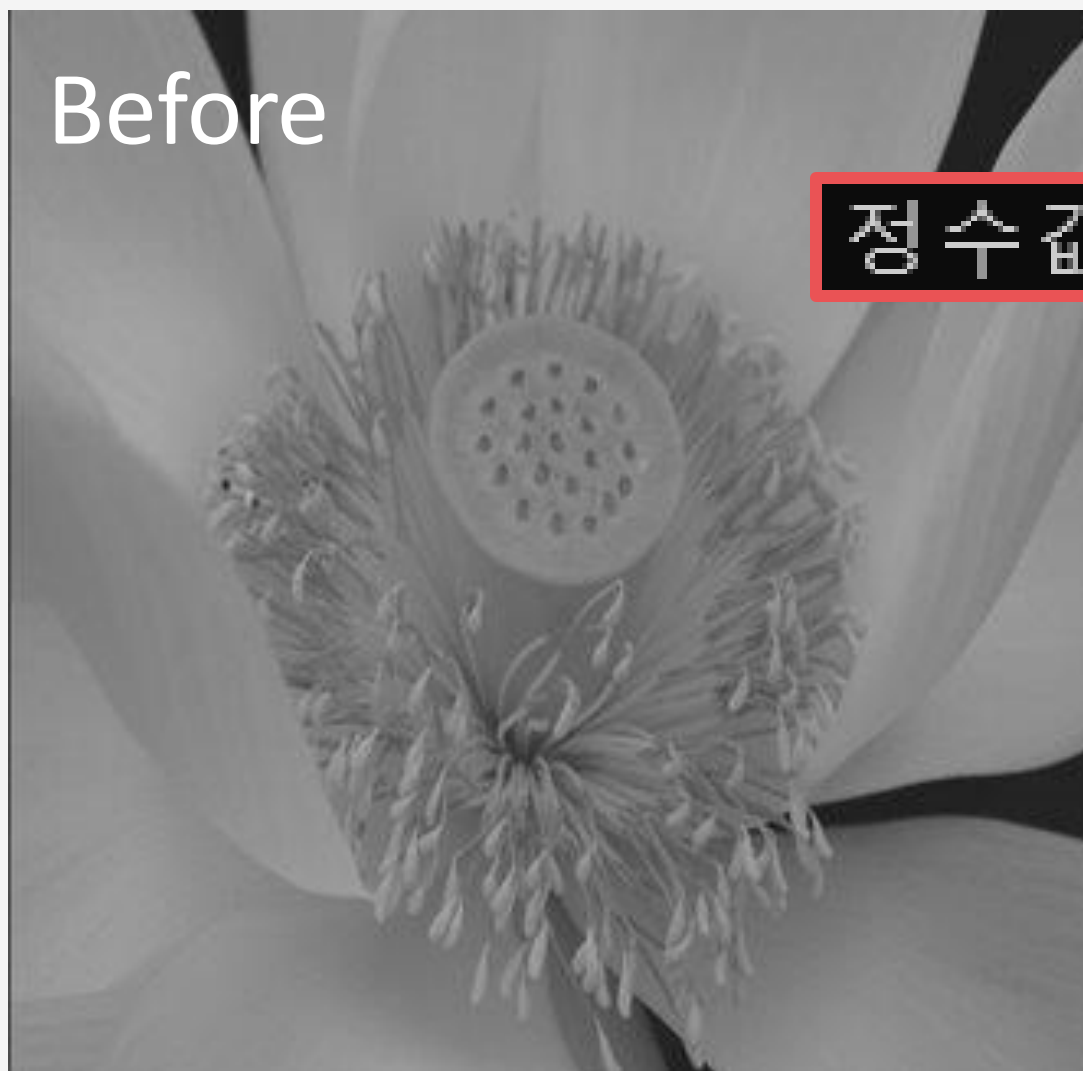
```
(int)(cos(radian) * (xd - cx) + sin(radian) * (yd - cy));
(int)(-sin(radian) * (xd - cx) + cos(radian) * (yd - cy));
```

F 회전(안깨짐+안잘림)

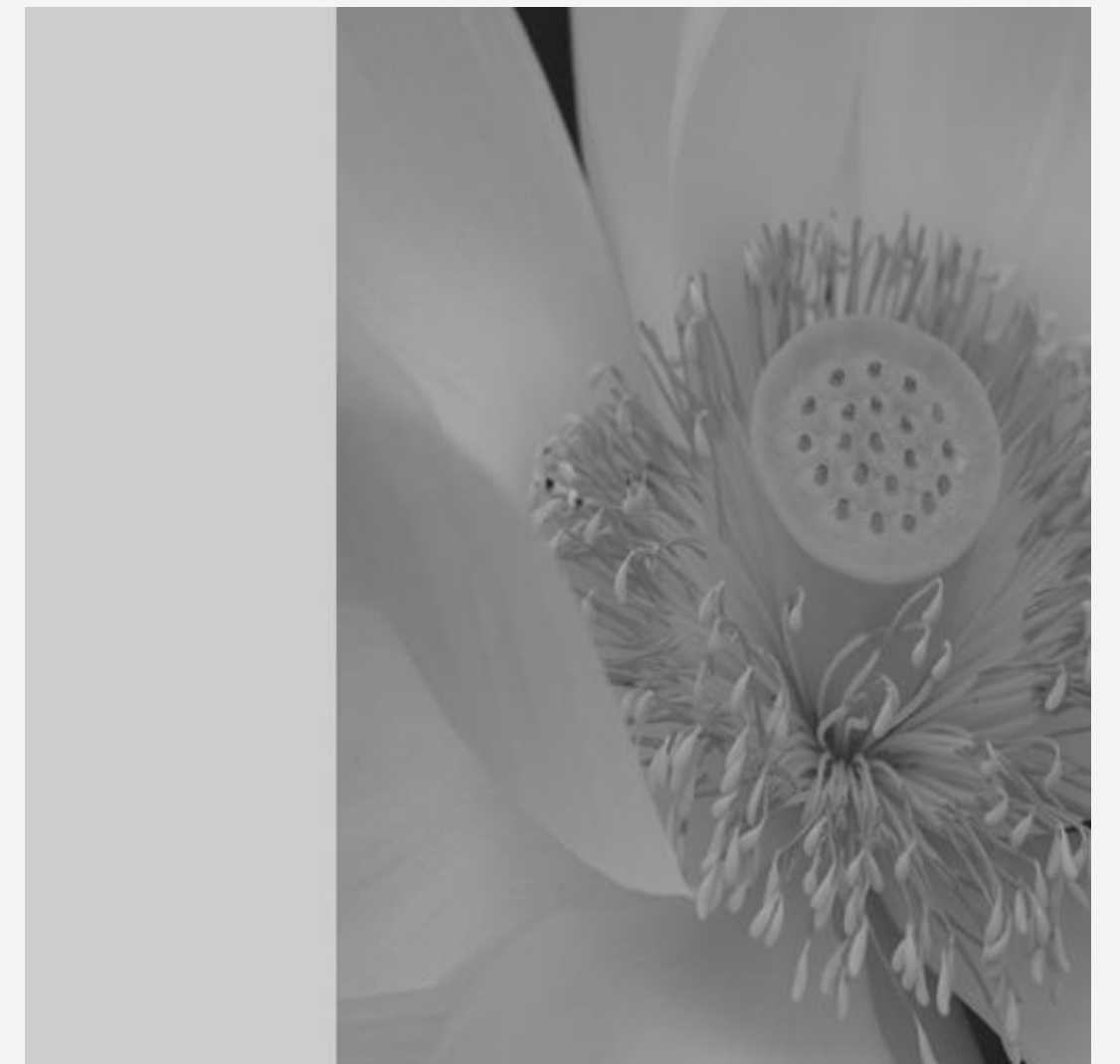
```
(int)(cos(radian) * (xd - (outW/2)) + sin(radian) * (yd - (outH / 2)));
(int)(-sin(radian) * (xd - (outW / 2)) + cos(radian) * (yd - (outH / 2)));
```

화면 구성 및 기능

B 기하학처리 中 이동



J 세로 이동



K 가로 이동

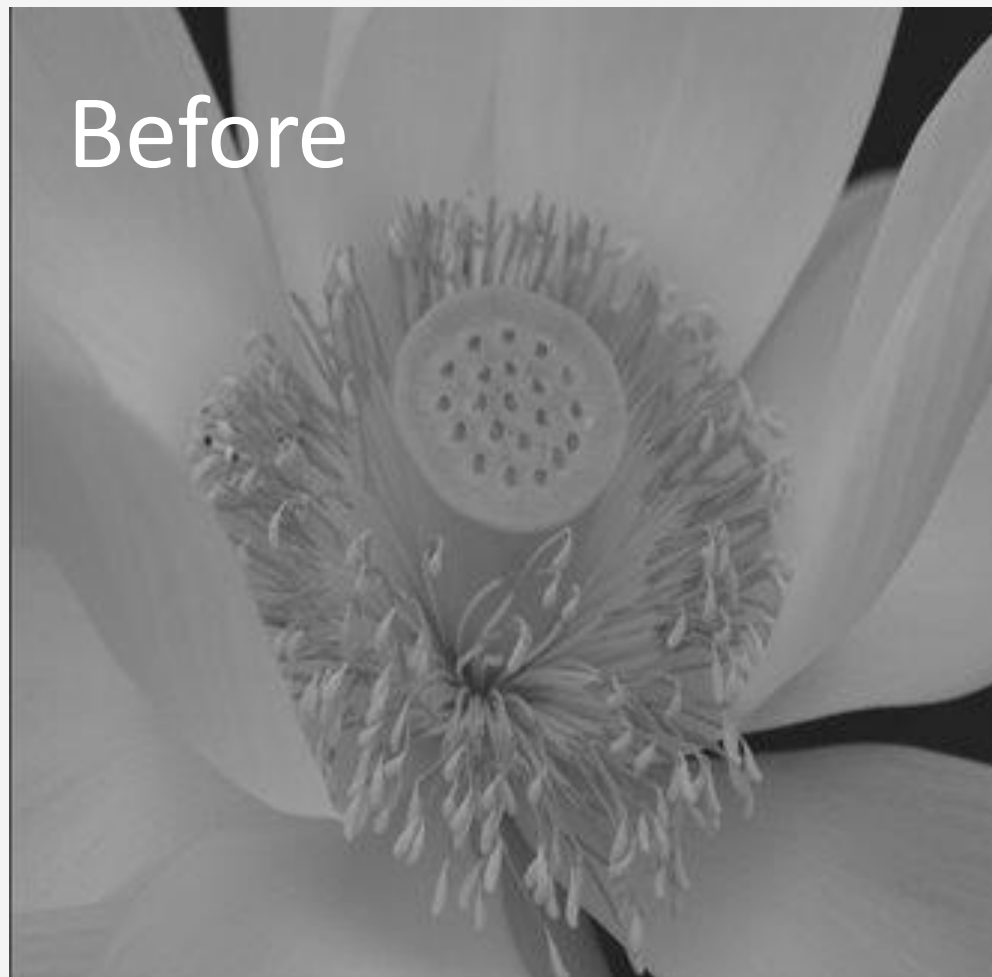
J 세로 이동

```
if (i + val >= outH) break;
if (i + val < 0) break;
outImage[i + val][k] = inImage[i][k];
```

K 가로 이동

```
if (k + val >= outW) break;
if (k + val < 0) break;
outImage[i][k + val] = inImage[i][k];
```


화면 구성 및 기능 B 기하학처리 > L 미러링



```
outImage[i][k] = inImage[i][k];  
outImage[2 * inH - i - 1][k] = inImage[i][k];  
outImage[i][2 * inW - k - 1] = inImage[i][k];  
outImage[2 * inH - i - 1][2 * inW - k - 1] = inImage[i][k];
```

| 화면 구성 및 기능

B 기하학처리 > M 모핑



키를 누를 때마다 점점 다른 사진으로 변화

```
(unsigned char)((4. - n) / 4 * inImage[i][k] + n / 4. * moImage[i][k]);
```

화면 구성 및 기능

C 히스토그램 & 화소 영역 처리 메뉴

D:\MyProject\W0318\Wx64\De

+

▼

—

□

×

A. 히스토 스트레칭

B. 엔드인

C. 히스토그램 평활화

D. 엠보싱

E. 블러링 (3 x 3)

F. 블러링 (N x N)

G. 가우시안 스무딩

H. 샤프닝 1

I. 샤프닝 2

J. 세로 엣지 마스크

K. 유사 연산자

화면 구성 및 기능

C 히스토그램 > A 히스토스트레치



전체적으로 어둡거나
밝은 색감의 사진에 있는
기존 색을 참고하여
부드럽게 밝히거나 어둡게
만드는 처리

기존 사진의 가장 밝고 어두운 수치를 찾음

```
if (inImage[i][k] < low)    low = inImage[i][k];  
if (inImage[i][k] > high)   high = inImage[i][k];
```



찾은 수치를 참고해 색감을 변경

```
new = (int)((double)old - low) / (high - low) * 255.0);
```


화면 구성 및 기능 C 히스토그램 > B 엔드인



구조는 히스토스크래치와
동일하지만,
가장 밝은 값과
어두운 값을 조정하여
사진을 일부 손실하더라도
더욱 선명한
사진을 만드는 처리

기존 사진의 가장 밝고
어두운 수치를 찾음



```
high -= 50;  
low += 50;
```



찾은 수치를 참고해
색감을 변경

화면 구성 및 기능

C 히스토그램 > C 히스토그램 평활화



각 밝기의 빈도를 일정하게
만드는 처리

밝은 사진은
전체적으로 어두워짐

어두운 사진은
전체적으로 밝아짐

히스토그램 생성

밝기의 빈도를 셈



누적 히스토그램 생성

히스토그램의
누적합 배열을 생성



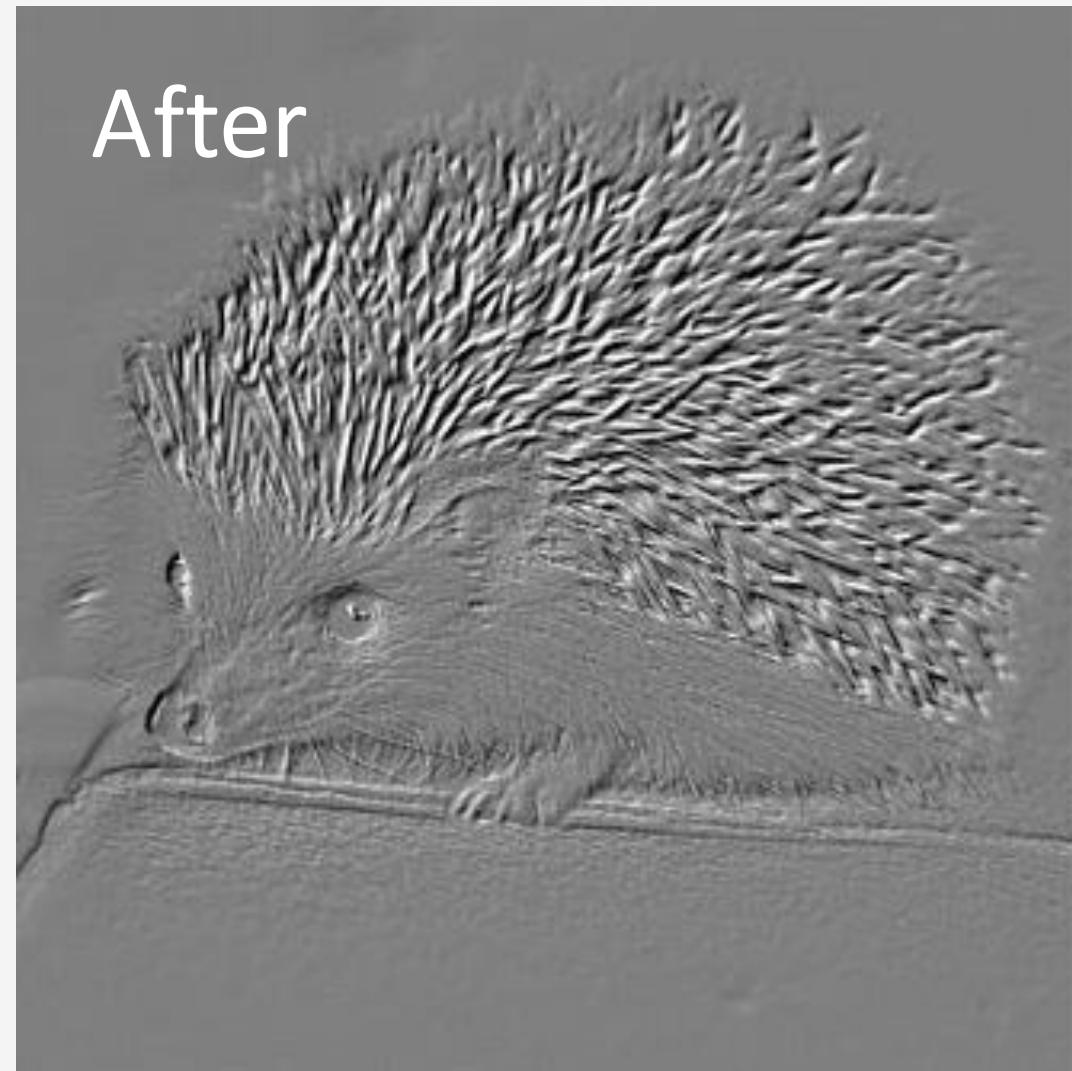
히스토그램 정규화

누적합 배열을 평균냄

화면 구성 및 기능

C 화소영역처리 > D 엠보싱

```
{ {-1.0, 0.0, 0.0},  
  {0.0, 0.0, 0.0},  
  {0.0, 0.0, 1.0} };
```



사진에 일부만 튀어나오게 하는
마스크를 씌움

임시배열의 (1, 1)부터
출력배열에 복사



필터의 합이 0이면
임시배열에서 127을 더함



임시 저장 배열 생성

사진 크기보다 높이와 너비가
(필터크기-1)만큼 크게 생성



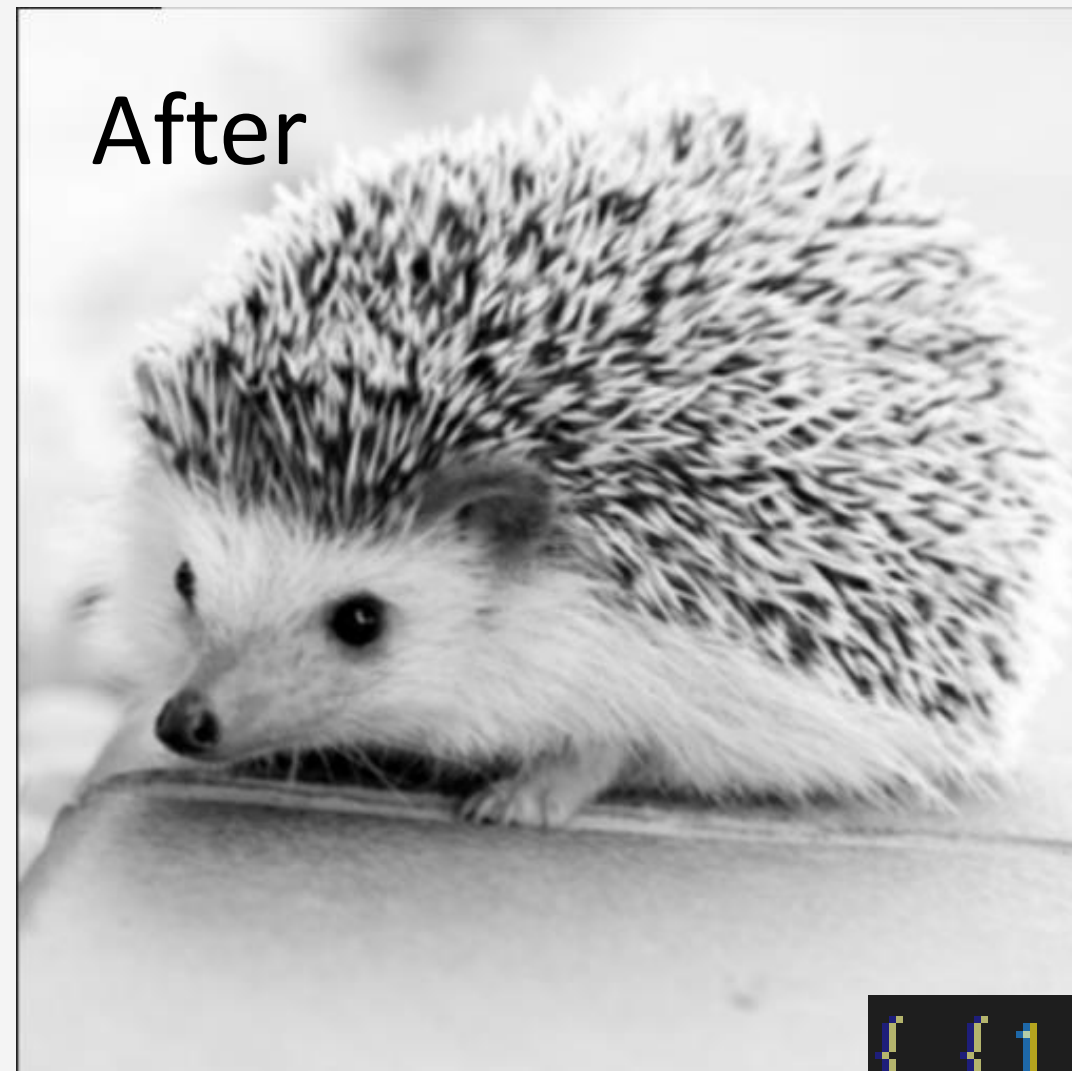
127으로 초기화
입력 사진 복사
(1, 1)부터 시작



임시배열의 (1, 1)부터 마스크를 씌움

마스크와 겹쳐진 곳끼리는 곱하고
곱해진 값의 평균으로 임시배열을 변경

화면 구성 및 기능 C 화소영역처리 > E 블러링 (3*3)



사진에 블러 처리를 하는
마스크를 씌움

```
{ {1.0 / 9, 1.0 / 9, 1.0 / 9},  
  {1.0 / 9, 1.0 / 9, 1.0 / 9},  
  {1.0 / 9, 1.0 / 9, 1.0 / 9} };
```


화면 구성 및 기능 C 화소영역처리 > F 블러링 (n*n)



정수 값 --> 5

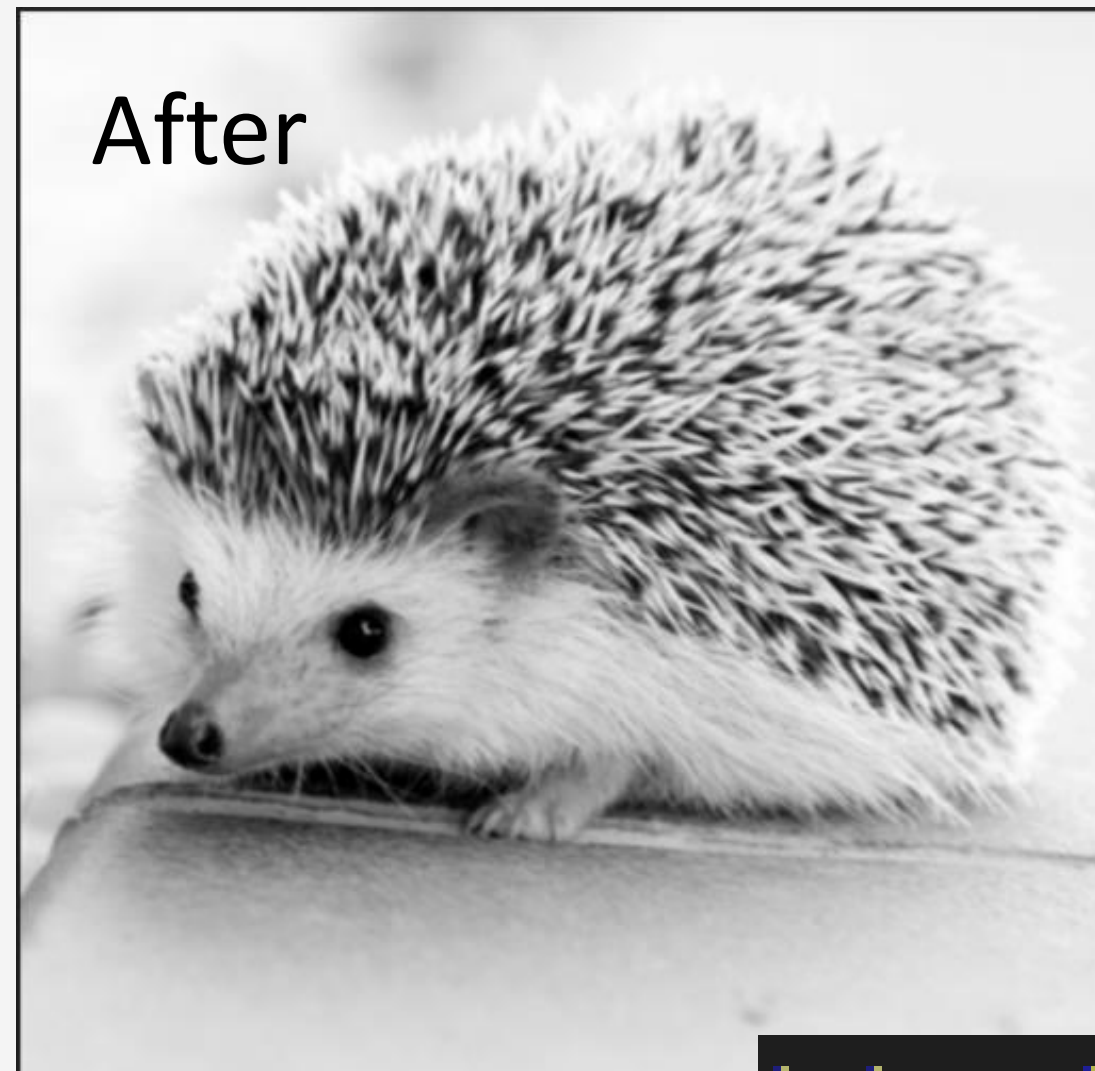
사진에 입력한 수치에 따라
블러 처리를 하는
마스크를 씌움

입력한 수치에 따라 마스크의 크기와
채우는 수를 다르게 처리

```
int lenScale = scale * 2 + 1;  
mask = mallocDoubleMemory(lenScale, lenScale);  
mask[i][k] = 1. / pow(lenScale, 2);
```

화면 구성 및 기능

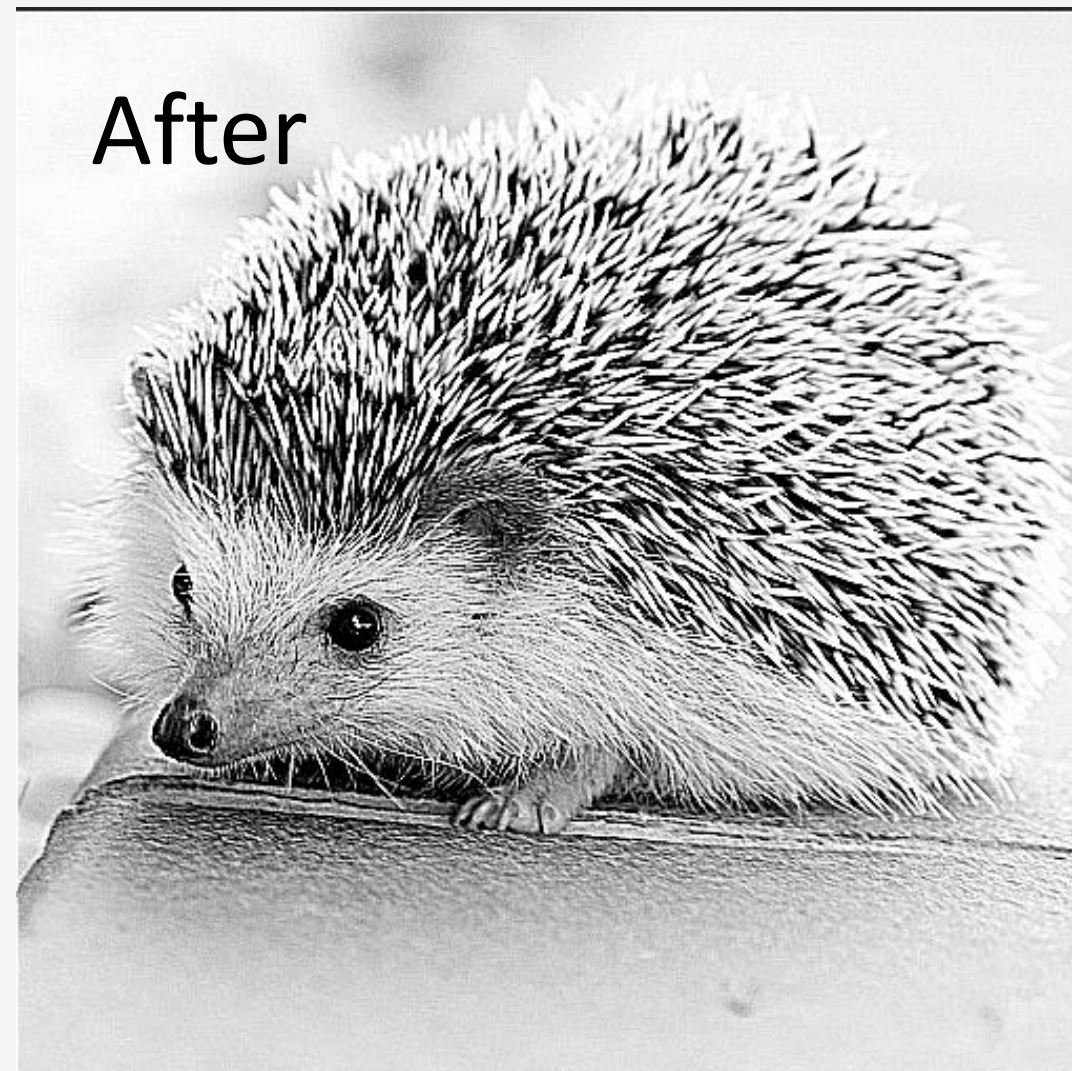
C 화소영역처리 > G 가우시안 스무딩



세세한 부분을 제거해
부드럽게 보이도록 하는
마스크를 씌움
(마스크의 영감을
가우시안 함수에서 받음)

```
{ {1. / 16 , 1. / 8, 1. / 16},  
  {1. / 8 , 1. / 4, 1. / 8},  
  {1. / 16 , 1. / 8, 1. / 16} };
```


화면 구성 및 기능 C 화소영역처리 中 샤프닝



상세한 부분을
더욱 잘 보이도록 하는
마스크를 씌움

H 샤프닝1

```
{ {-1., -1., -1.},  
  {-1., 9 , -1.},  
  {-1., -1., -1.} };
```

I 샤프닝2

```
{ {0., -1., 0.},  
  {-1., 5 , -1.},  
  {0., -1., 0.} };
```


화면 구성 및 기능 C 화소영역처리 > J 세로 엣지 마스크



세로 모서리 부분을
잘 보이게 하는
마스크를 씌움

```
{ {0., 0., 0.},  
  {-1., 1., 0.},  
  {0., 0., 0.} };
```


화면 구성 및 기능 C 화소영역처리 > K 유사 연산자



주변 픽셀에서
현재 계산중의 픽셀의 값을
뺀 값 중 가장 큰 값을
사용하는 방법

계산을 위한 초기 값은
첫번째 값으로 선택

```
S = abs(tmpInImage[i + 1][k + 1] - tmpInImage[i][k]);
```

```
S = max(abs(tmpInImage[i + 1][k + 1] - tmpInImage[i + m][k + n]), S);
```

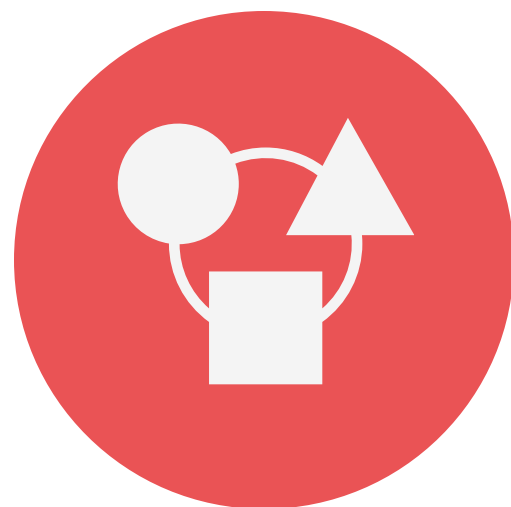
부가 기능



되돌리기

전역 변수와 함수를 통해
돌아 갈 파일과 위치를 저장합니다.

```
int backNum = 0, nowBack = 0;  
case '^':      backToBefore();  
saveCache();
```



효과 중첩

전역 변수를 통해 효과 중첩을 관리합니다.

```
bool accumulation = false;
```

부가 기능



Linux 전환

Before

```
0
파일명-->flower512
137 137 138 137 131 122 112 107 107 105
136 137 138 139 137 127 115 111 111 114
134 138 139 140 138 136 129 124 126 131
136 138 140 139 138 138 138 137 137 135
136 137 138 138 138 136 136 135 136
135 135 137 136 138 138 138 137 136
134 134 135 134 136 138 138 138 137 137
136 135 134 134 135 136 136 135 136 135
136 134 136 134 134 134 132 133 133 134
134 135 135 135 134 132 131 130 132 131
```

A. 히스토 스트레칭 B. 엔드인 C. 히스토그램 평활화

D. 엠보싱 E. 블러링 (3 x 3) F. 블러링 (N x N)

G. 가우시안 스무딩 H. 샤프닝 1 I. 샤프닝 2 J. 세로 엣지 마스크

K. 유사 연산자

```
131 131 137 131 104 71 39 29 29 26
127 131 137 143 131 89 47 37 37 44
117 137 143 148 137 127 96 78 86 104
127 137 148 143 137 137 137 131 131 122
127 131 137 137 137 137 127 127 122 127
122 122 131 127 137 137 137 137 131 127
117 117 122 117 127 137 137 137 131 131
127 122 117 117 122 127 127 122 127 122
127 117 127 117 117 117 108 112 112 117
117 122 122 122 117 108 104 100 108 104
```

After
- 히스토그램 평활화

마치며

좋았던 점

영상처리에 대한 학습을 하고 직접 코딩을 해본 점

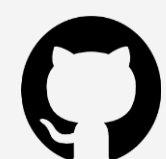
아쉬운 점

감마 알고리즘에서 정수로 입력 받도록 잘못 설정한 점

추후 계획

필터를 좀 더 찾아보고 학습해보자

장혜원 .



https://github.com/Jang-HW/Intel_Edge_AI_SW_Academy



hw11515@naver.com