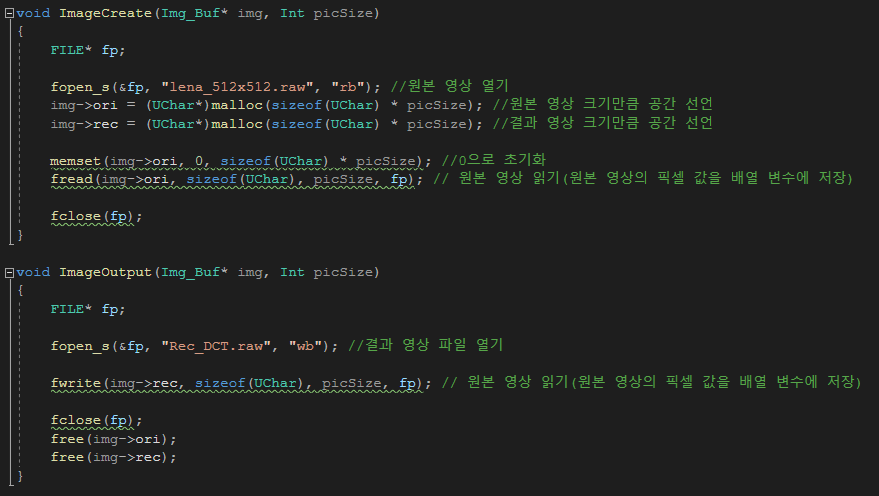
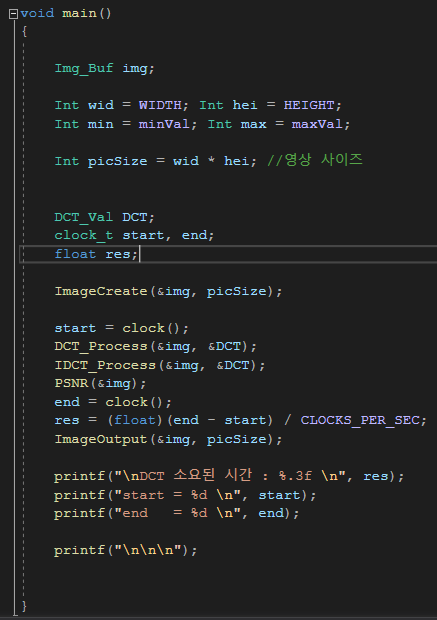
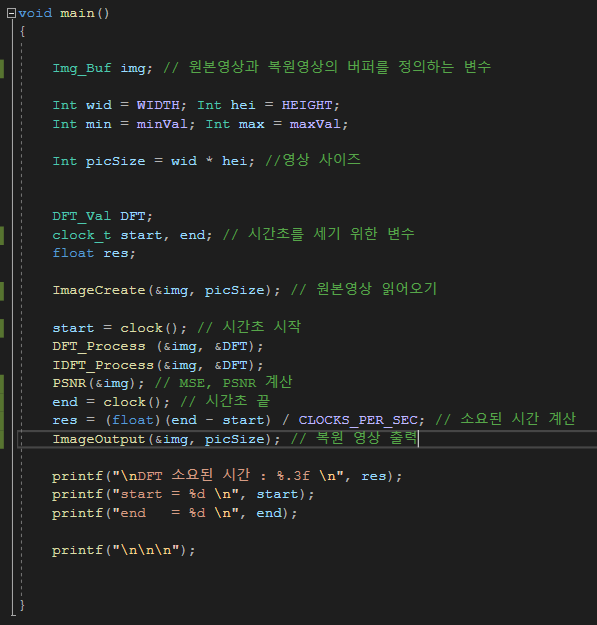
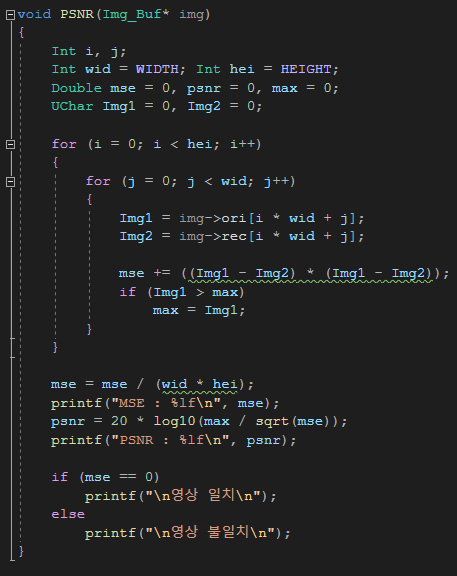
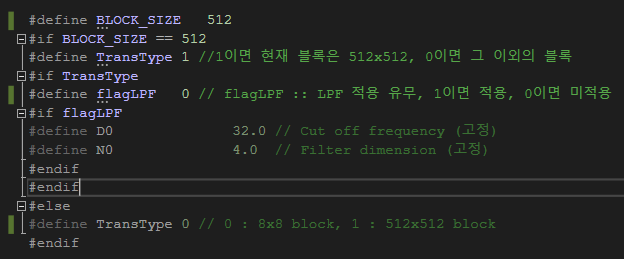
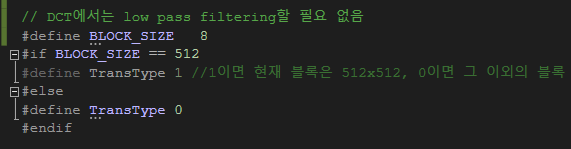
**4. Frequency-domain processing**

**:** Frequency-domain processing은 영상처리를 수행할 때 영상 데이터를 주파수 공간으로 변환하는 과정이 선행됩니다. Fourier Transform을 통해 영상의 변환계수를 얻고, 그 변환계수(spectrum)를 통해 영상처리를 수행한 다음(filtered spectrum) 다시 Inverse Fourier Transform을 통해 다시 주파수 도메인에서 시간 도메인으로 변환합니다.



* 전반적인 실습 과정을 위 코드와 같습니다. 원본 영상을 읽어오고 DFT 및 DCT를 수행한 후 다시 IDFT 및 IDCT를 수행하여 영상을 복원합니다. 실습 과정에서 소요되는 시간을 측정하여 출력하고, 원본 영상과 복원 영상의 일치를 비교하기 위해 MSE와 PSNR을 사용합니다.

* MSE는 평균 제곱 오차로, 두 영상의 픽셀값 간의 차이를 계산하는 함수입니다. 따라서 MSE가 0이면 두 영상을 일치하다는 것을 뜻합니다. PSNR은 최대 신호 대 잡음비로, 손실이 적을수록 높은 값을 가집니다. 따라서 영상을 주파수 공간으로 변환한 후 영상처리를 수행하지 않고 다시 복원한 경우, MSE = 0, PSNR = 무한대 가 되어야 하고, 실습에서 Low Pass Filter을 적용한 경우 MSE가 0이 아니고, PSNR도 무한대가 아니어야 합니다. (가장 왼쪽 코드에 해당) DFT와 DCT를 수행할 때 변환 블록의 크기를 지정하여 수행합니다. 또한 DFT 실습에서는 주파수 영역으로 변환 후 Low pass Filter를 수행하는 실습을 진행하기 때문에 Low Pass Filter를 사용 여부를 나타내는 변수를 선언합니다. (중간 코드 – DFT, 오른쪽 코드 – DCT)

**실습 1) DFT(Discrete Fourier Transform)**

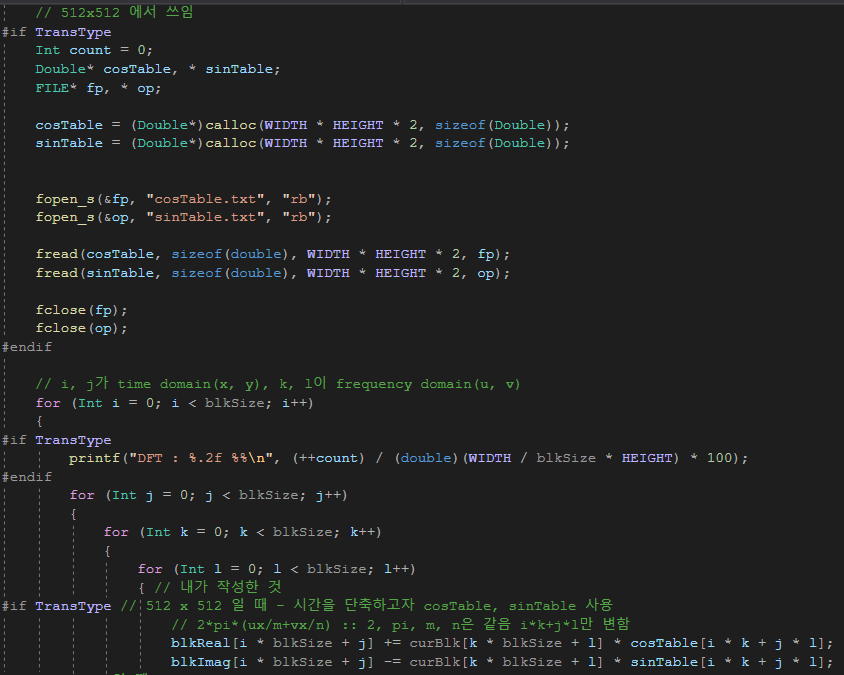
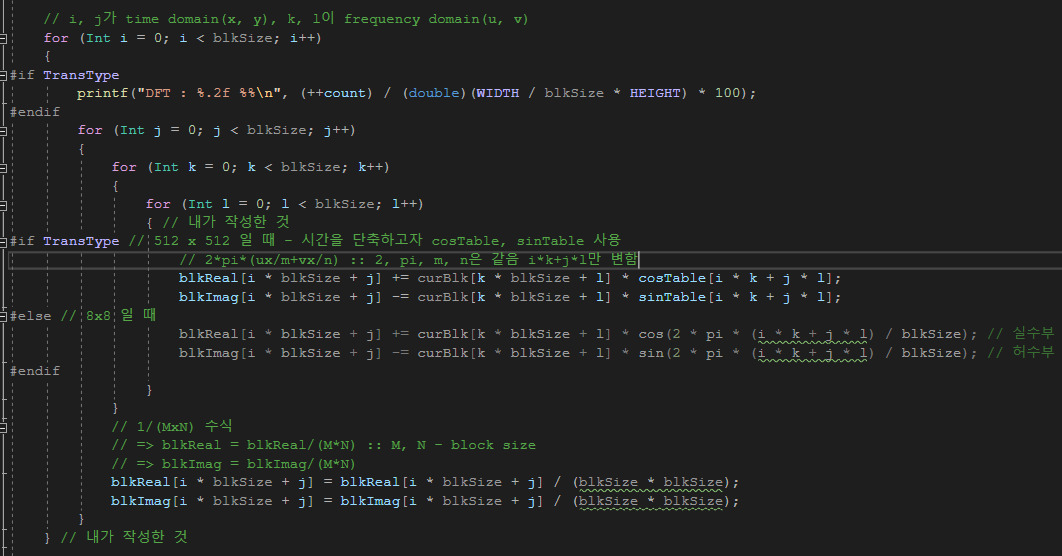
* Time domain인 영상 데이터를 Frequency domain으로 변환할 때 Fourier Transform을 수행합니다. 이 때 영상 처리를 수행할 디지털 영상 데이터는 유한한 개수의 discrete한 신호이기 때문에 주기가 있고, discrete한 신호에서의 변환인 Discrete Fourier Transform(DFT)를 이용하여 Frequency-domain processing을 수행합니다

1. **DFT**

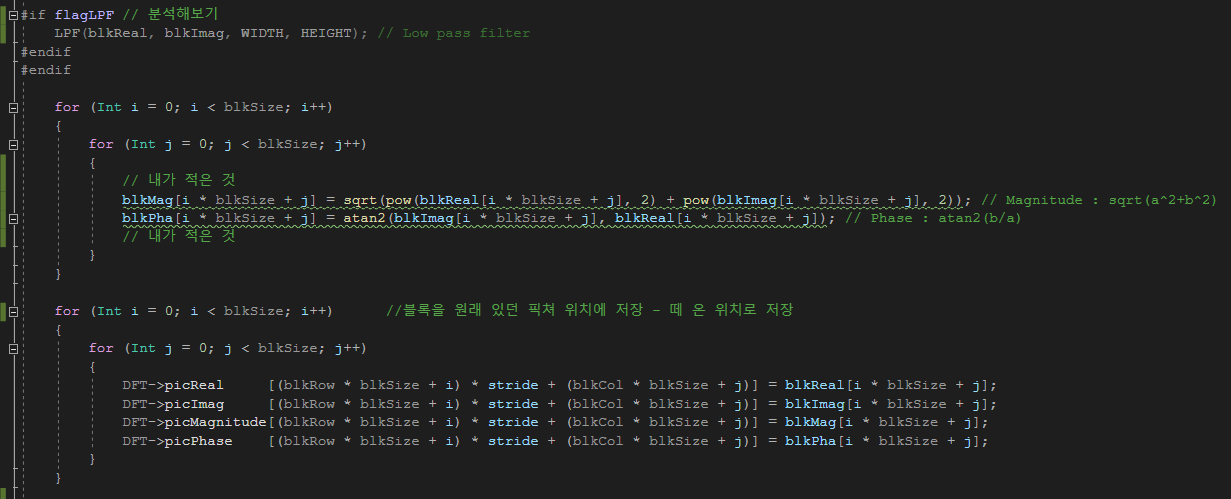
* **Forward DFT**



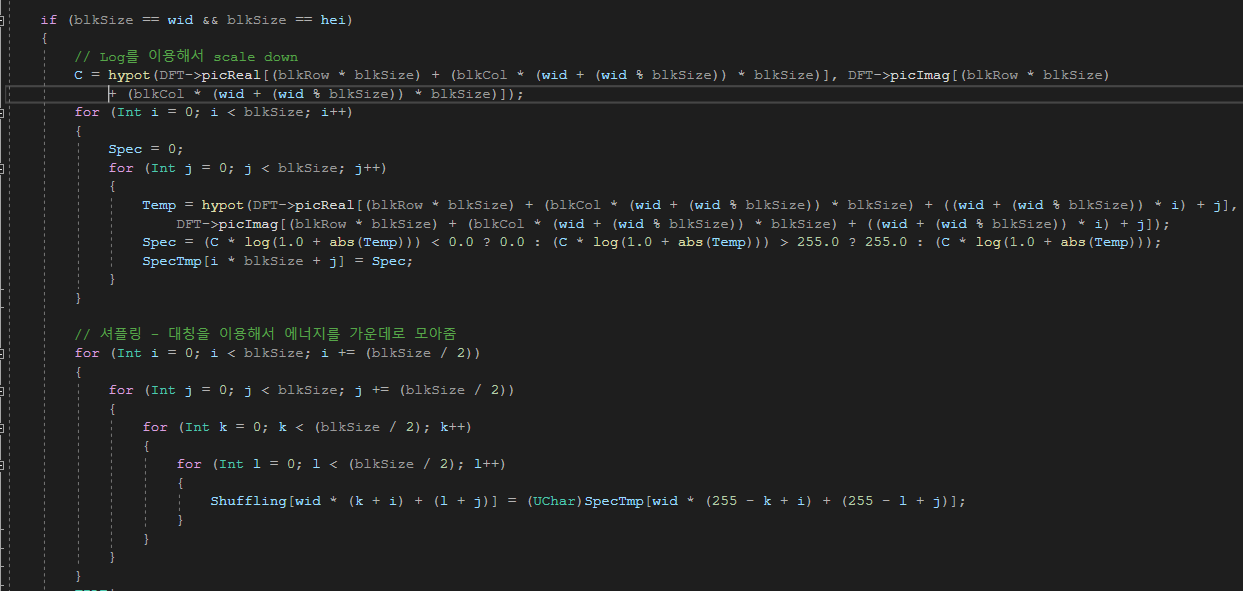
* Forward DFT를 수행할 때는 변환 블록 크기만큼 블록을 가져오기 위해 현재 block의 왼쪽 위 좌표를 찾는 과정이 필요합니다. 이 과정에 해당하는 DFT Process 코드입니다. 현재 block의 왼쪽 위 좌표를 찾기 위해 원본 영상의 첫 번째 공간(ori[0])을 기준으로 떨어져 있는 거리를 계산합니다. 예를 들어 변환 블록의 크기가 8x8이면 (0, 0), (8, 0), (16, 0), … , (0, 8), (8, 8), (16, 8), … 으로 일반화 하면 (blkCol \* blkSize, blkRow \* blkSize)가 됩니다. 이 2차원 좌표계를 블록 단위로 1차원 좌표계로 바꾸면 blkRow \* blkSize \* stride(영상의 width) + blkCol \* blkCol 가 되고, 이를 첫 번째 공간을 기준으로 좌표계를 설정하기 위해 img->ori를 더합니다. 이렇게 블록의 왼쪽 위 좌표를 기준으로 블록을 떼온 변수가 curBlk입니다. 변환 블록을 떼온 후, DFT\_Func을 호출하여 DFT를 수행합니다.

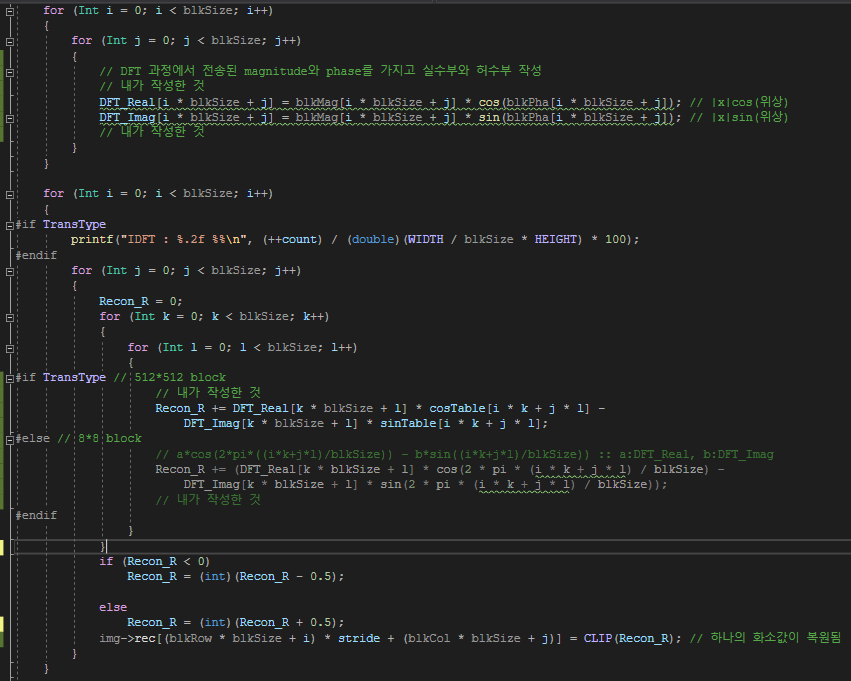
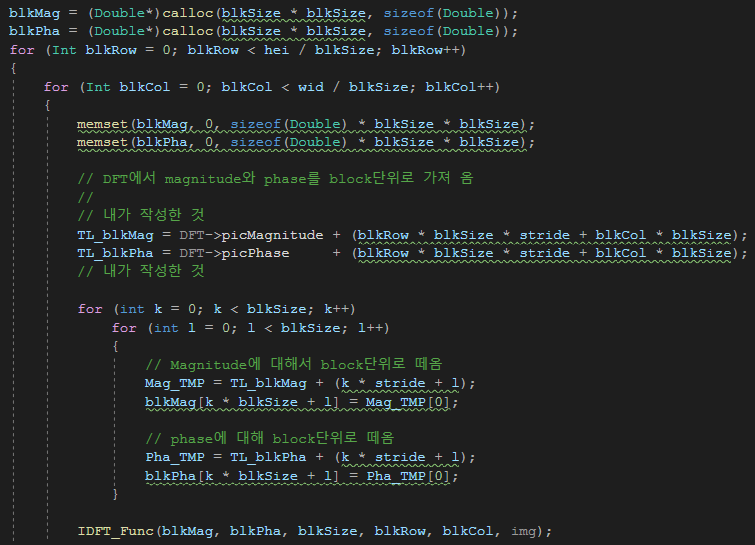
* 512x512변환 블록을 사용할 때 실행 시간이 오래 걸리기 때문에 시간을 단축하고자 값에 따른 cosine과 sine의 결과값을 저장해 놓은 cosTable과 sinTable을 이용합니다.(왼쪽 코드 – cosine/sin table 불러오기) Forward DFT의 식을 그대로 변환 블록 내부의 화소에 적용합니다. Forward DFT 식의 복소수는 오일러 공식(e^(i\*seta) = cos(seta) + i\*sin(seta))에 의해 curBlk \* (cos(2pi\*n\*k/N) – i\*sin(2pi\*n\*k/N))으로 작성할 수 있고, 실수부와 허수부를 각각 저장합니다. 여기에서 N은 변환 블록의 크기이고, k와 l은 time domain에서의 x, y이며 I, j가 frequency domain에서의 u, v입니다. 마지막으로 forward DFT 식의 1/(M\*N)을 구하기 위해 구한 실수부와 허수부에 블록 size의 제곱을 나누어 줍니다. 실습에서는 변환 블록의 width와 height가 같기 때문에 변환 블록 size의 제곱을 나누는 것입니다.



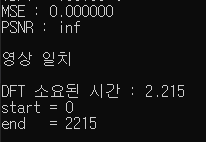
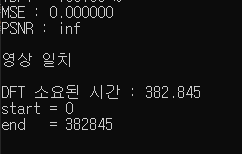
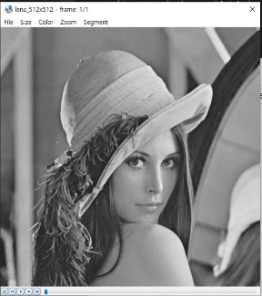
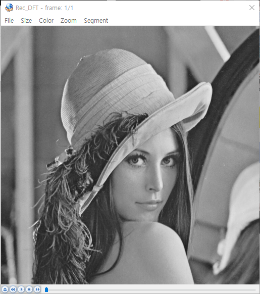
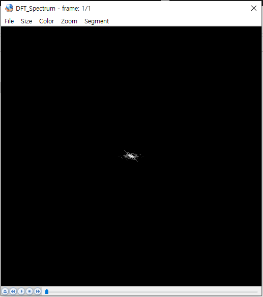
* Forward DFT에서 magnitude와 phase를 구합니다. Magnitude는 실수부와 허수부 각각의 제곱을 더한 값에 루트를 씌운 값이고, phase는 arctans(b/a)입니다. 이 때 구한 magnitude와 phase는 Inverse DFT에서 사용됩니다.
* **Spectrum**



* DFT를 구현한 후 에너지 집중 현상이 발생합니다. 에너지 집중 현상을 눈으로 보기 위해 spectrum을 만드는 코드입니다. 스펙트럼 영상을 나타낼 때 구한 주파수 데이터의 범위가 너무 넓어서 잘 관찰할 수 없기 때문에 먼저 log를 취합니다. 그 다음으로 DC성분이 구석으로 몰리기 때문에 대칭을 이용하여 에너지를 가운데로 모아주는 셔플링 과정을 수행합니다.
* **Inverse DFT**



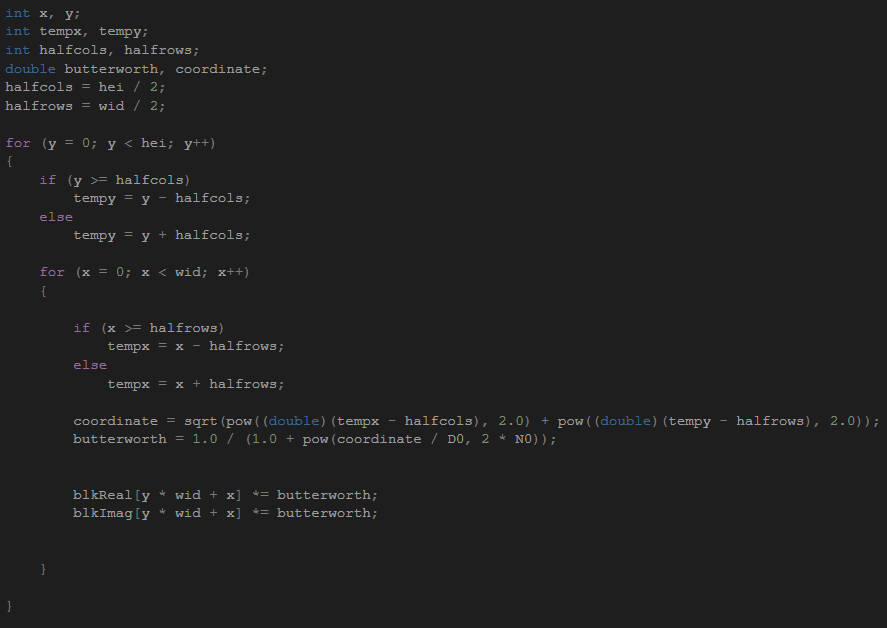
* 앞에서 구한 DFT에서 magnitude와 phase를 변환 block단위로 가져오는 과정을 수행합니다. Block단위로 가져오는 방법은 DFT에서 변환 블록의 왼쪽 상단의 좌표를 가져오는 과정과 동일합니다. (왼쪽 코드에 해당) 다음으로 변환 블록으로 떼어진 DFT과정에서 얻어진 magnitude와 phase를 이용하여 실수부와 허수부를 작성합니다. Magnitude \* cos(phase)가 실수부, Magnitude \* sin(phase)가 허수부입니다. (오른쪽 코드의 위 부분에 해당) 다음으로 inverse DFT(실수부(=magnitude\*cos(phase))\*cos(2pi \* x\*y/N – 허수부(=magnitude\*sin(phase))\*sin(2pi\*x\*y/n) 를 수행합니다. DFT와 동일하게 n은 변환 블록의 크기이고, k와 l은 time domain에서의 x, y이며 i, j가 frequency domain에서의 u, v입니다. 512x512 변환 블록에서의 IDFT또한 시간을 줄이기 위해 sinTable, cosTable을 이용합니다.
* **결과 영상**

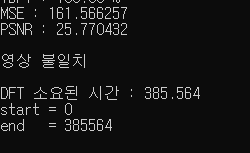
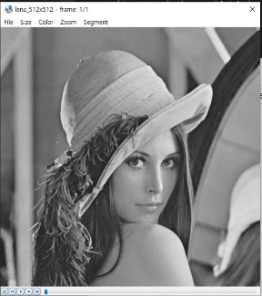
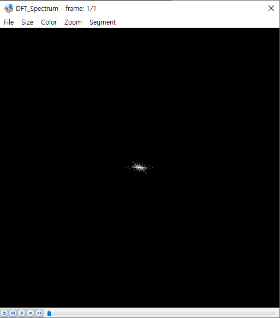
* 첫번째 사진은 8x8 변환 블록을 이용하여 DFT->IDFT을 수행한 결과이고, 두번째 사진은 512x512 변환 블록을 이용하여 수행한 결과입니다. LPF와 같은 별도의 영상처리를 수행하지 않았기 때문에 원본 영상과 복원된 영상이 동일하고, 따라서 MSE는 0, PSNR은 무한대가 출력됨을 확인할 수 있습니다. 결과 영상을 비교해 보았을 때도 원본 영상(세번째 사진)과 8x8 변환 블록을 이용한 복원 영상(네번째 사진), 512x512 변환 블록을 이용한 복원 영상(다섯 번째 사진)이 동일합니다. 여섯 번째 사진은 512x512 변환 블록을 이용하여 DFT를 수행한 주파수 domain의 주파수 스펙트럼입니다. 셔플링하여 DC가 가운데로 존재하게 끔 한 Spectrum 또한 알맞게 출력되었음을 확인할 수 있습니다.

1. **Low pass filtered DFT**

* Frequency-domain filtering은 원본 영상을 Fourier Transform을 수행하여 주파수 영역으로 변환하고, 이 과정으로 얻은 영상의 변환계수를 filter mask와 곱하여 filtered 변환계수를 얻습니다. 그 후 역변환을 통해 filtered image를 얻는 과정을 거칩니다. 실습에서 영상에 Low pass filter를 적용하기 위해 DFT를 수행하여 영상을 frequency-domain으로 변환하고, 영상 처리 기법 중 Low pass filter의 하나인 Butterworth Low pass filter를 적용하는 과정을 수행해야 합니다.



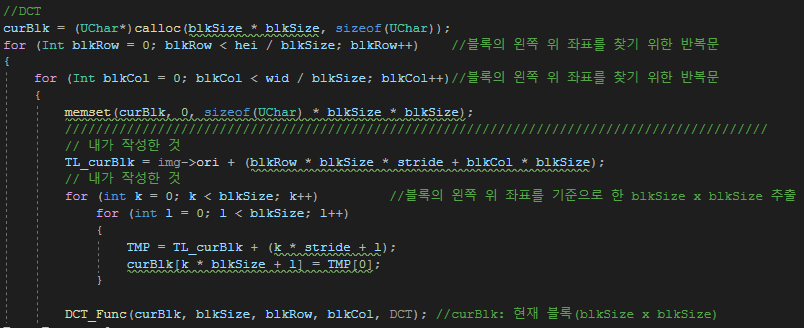
* LPF를 적용하기 전에 DFT를 수행하는 과정과 LPF를 적용한 후 IDFT를 수행하는 과정을 동일합니다.
* main.h 파일에서 선언한 Cutoff freq = 32.0, Filter dimension = 4.0에 따라 LPF을 수행합니다. 저주파 통과 필터는 1/(1+(w/wc)^2N)이며, w는 sqrt(u^2+v^2), wc는 Cutoff freq, N은 Filter dimension입니다. 코드에서 w는 coordinate로, Butterworth LPF는 butterworth 변수로 선언되었습니다.
* **결과 영상**

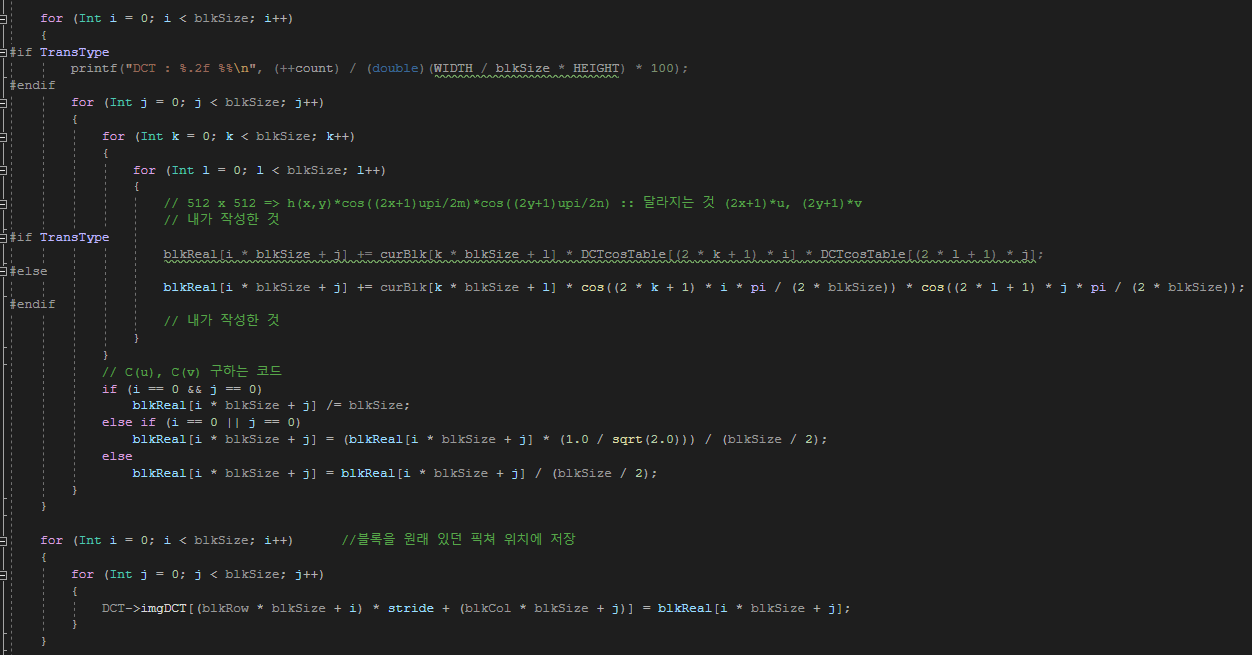
* + - * 첫번째 사진은 512x512 변환 블록을 이용하여 DFT->LPF->IDFT을 수행한 결과입니다. LPF 영상처리를 수행했기 때문에 원본 영상에 비해 복원된 영상이 흐릿하게 출력되었고, 따라서 MSE는 0이 아니며 PSNR은 무한대가 아니도록 출력됨을 확인할 수 있습니다. 네번째 사진은 512x512 변환 블록을 이용하여 DFT를 수행하고 LPF도 수행한 주파수 domain의 주파수 스펙트럼입니다. 셔플링하여 DC가 가운데로 존재하게 끔 한 Spectrum 또한 알맞게 출력되었음을 확인할 수 있습니다

**실습 2) DCT(Discrete Cosine Transform)**

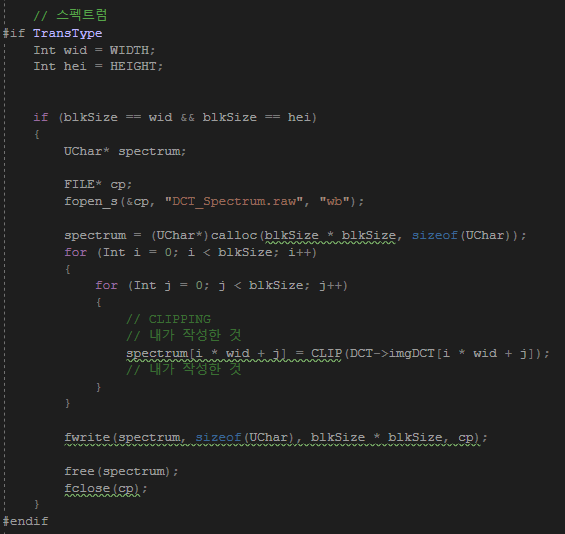
* **Forward DCT**



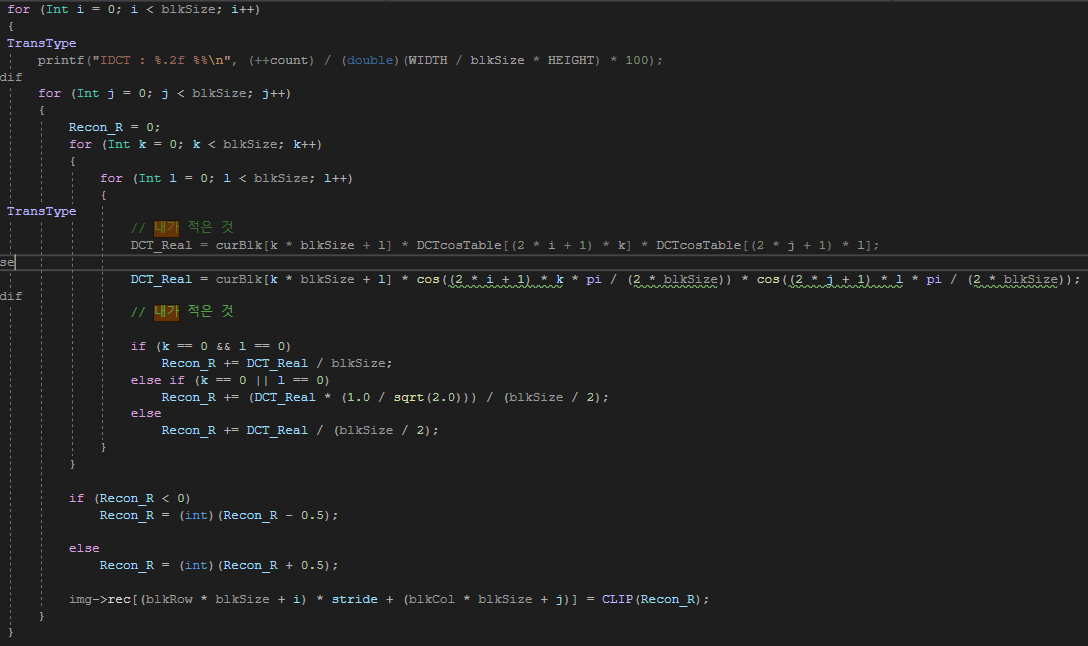
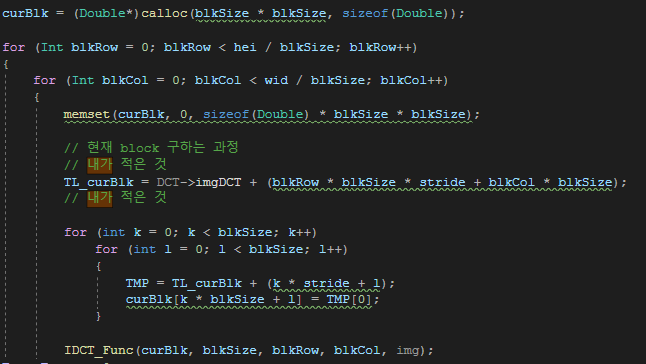
* + - * DCT는 DFT와 유사한 변환으로, DFT에서 발생하는 복소수(실수부와 허수부) 중 실수부 만을 이용합니다. DCT를 수행할 때에도 DFT와 동일한 방법으로 변환 블록의 크기에 따라 블록의 왼쪽 위 좌표를 기준으로 한 blkSize x blkSize를 추출합니다.



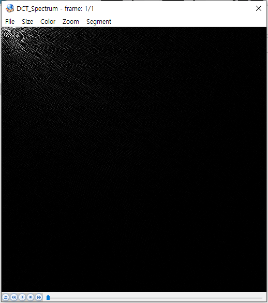
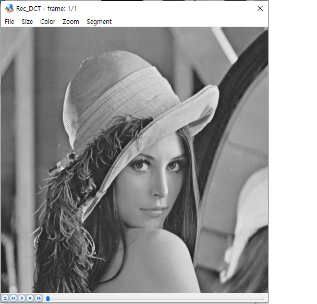
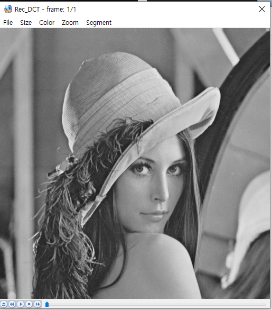
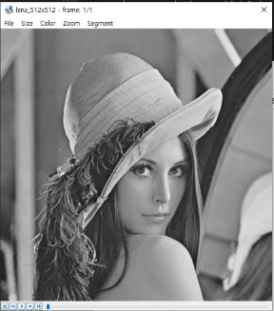
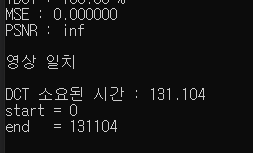
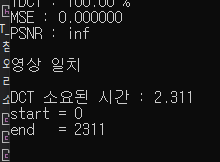
* + - * DCT에서도 512x512변환 블록을 사용할 때 실행 시간이 오래 걸리기 때문에 시간을 단축하고자 값에 따른 cosine과 sine의 결과값을 저장해 놓은 cosTable과 sinTable을 이용합니다. Forward DCT의 식을 그대로 변환 블록 내부의 화소에 적용합니다. DCT는 실수부만을 이용하기 때문에 허수부를 정의하지 않습니다. Forward DCT 식은 Forward DFT식에서 cos내부가 달라집니다. 4개의 반복문 내부에 있는 식이 Forward DCT에서의 cos((2x+1)\*upi/2M)\*cos((2y+1)\*vpi/2N)을 코드에 작성한 부분입니다. 여기에서 N은 변환 블록의 크기이고, k와 l은 time domain에서의 x, y이며 i, j가 frequency domain에서의 u, v입니다. 다음으로 DCT식에서의 C(u)와 C(v)를 구하고, 이렇게 구한 block을 원래의 위치에 저장합니다.
* **Spectrum**



* + - * DCT에서의 주파수 스펙트럼을 구하는 코드입니다. DCT에서의 DC 성분은 DFT와 달리 모서리에 존재하지 않으므로 대칭을 이용하여 중간으로 DC성분을 옮겨주는 셔플링 과정이 필요하지 않고, 영상의 화소 값의 범위를 초과한 화소를 영상의 화소 범위 내로 바꿔주는 clipping과정이 필요하므로 clipping을 수행합니다.
* **Inverse DCT**



* + - * 앞에서 구한 DCT를 변환 block단위로 가져오는 과정을 수행합니다. Block단위로 가져오는 방법은 DCT에서 변환 블록의 왼쪽 상단의 좌표를 가져오는 과정과 동일합니다. (왼쪽 코드에 해당) 다음으로 변환 블록으로 떼어진 DCT에서 inverse DCT를 수행합니다. (오른쪽 코드의 위 부분에 해당) IDCT에서의 cos함수 내부(cos((2x+1)\*upi/2M)\*cos((2y+1)\*upi/2N))를 작성한 것이 반복문 내부의 코드입니다. n은 변환 블록의 크기이고, k와 l은 time domain에서의 x, y이며 i, j가 frequency domain에서의 u, v입니다. 512x512 변환 블록에서의 IDCT또한 시간을 줄이기 위해 sinTable, cosTable을 이용합니다
* **결과 영상**



* 첫번째 사진은 8x8 변환 블록을 이용하여 DCT->IDCT을 수행한 결과이고, 두번째 사진은 512x512 변환 블록을 이용하여 수행한 결과입니다. LPF와 같은 별도의 영상처리를 수행하지 않았기 때문에 원본 영상과 복원된 영상이 동일하고, 따라서 MSE는 0, PSNR은 무한대가 출력됨을 확인할 수 있습니다. 결과 영상을 비교해 보았을 때도 원본 영상(세번째 사진)과 8x8 변환 블록을 이용한 복원 영상(네번째 사진), 512x512 변환 블록을 이용한 복원 영상(다섯 번째 사진)이 동일합니다. 여섯 번째 사진은 512x512 변환 블록을 이용하여 DCT를 수행한 주파수 domain의 주파수 스펙트럼입니다. DC성분이 모서리 쪽에 존재하지 않기 때문에 셔플링을 별도로 적용하지 않아도 됩니다.