KWANGWOON UNIVERSITY

2-Dimensional Digital Data Operations

2018.09.04

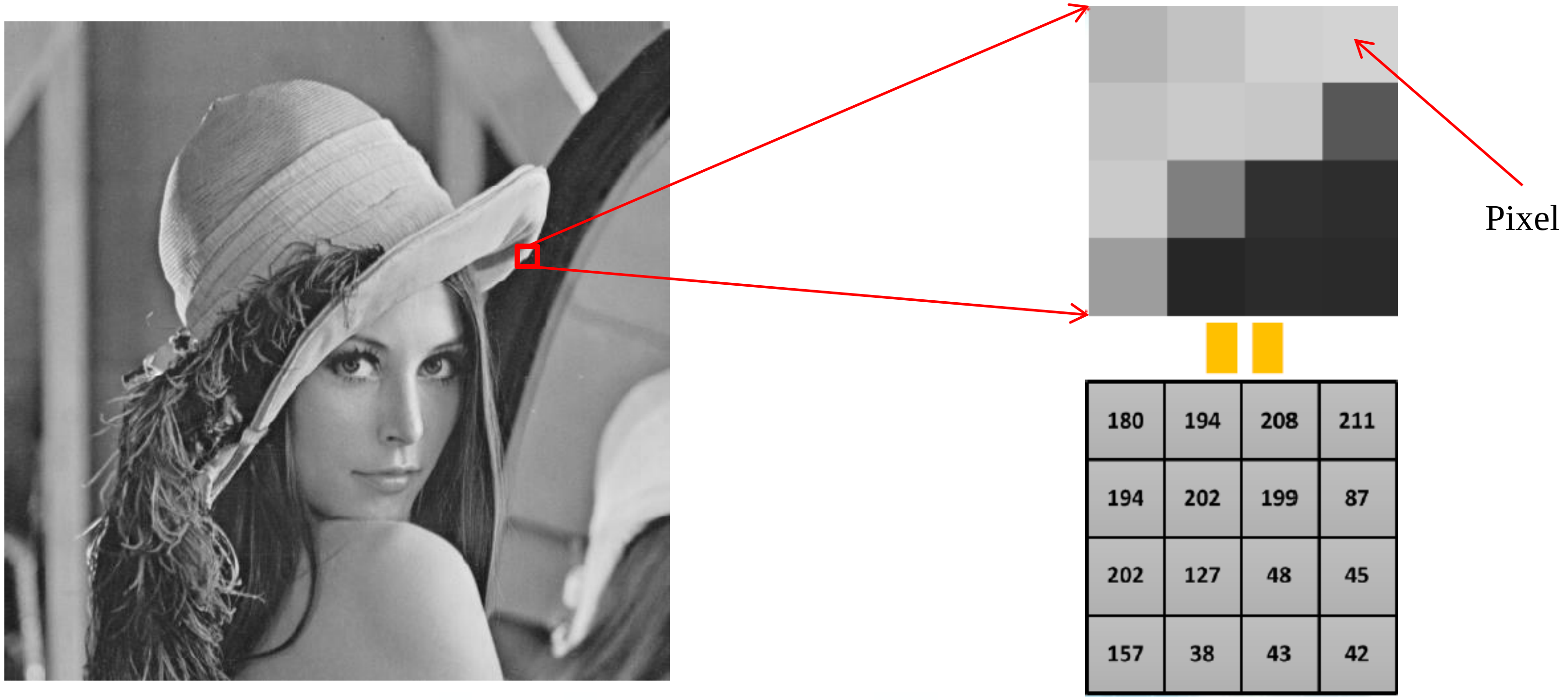
Seoungjun Oh(sjoh@kw.ac.kr)
Wooju Lee (krosea@kw.ac.kr)

Multimedia LAB

VIA-Multimedia Center, Kwangwoon University

Contents

Raw Image

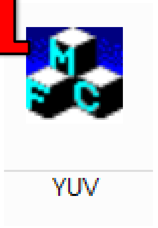


180	194	208	211
194	202	199	87
202	127	48	45
157	38	43	42

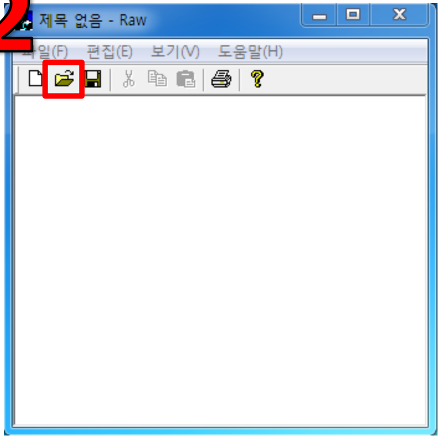
Range : 0 ~ 255

Raw Image

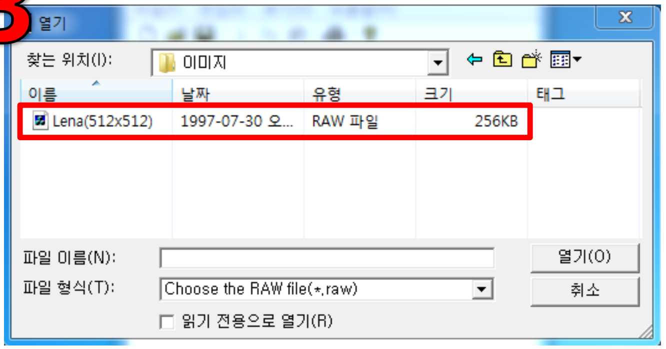
1



2



3

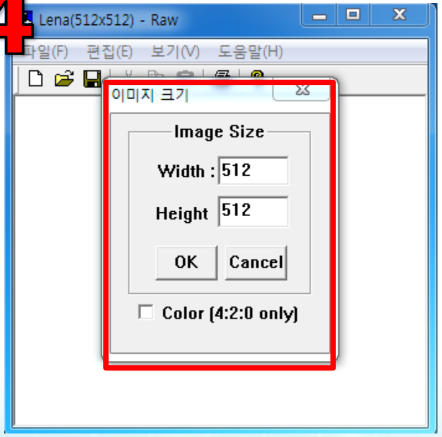


찾는 위치(I): 이미지

이름	날짜	유형	크기	태그
Lena(512x512)	1997-07-30 오...	RAW 파일	256KB	

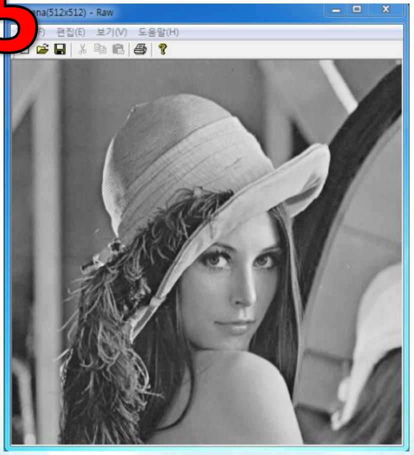
파일 형식(T): Choose the RAW file(*.raw)

4



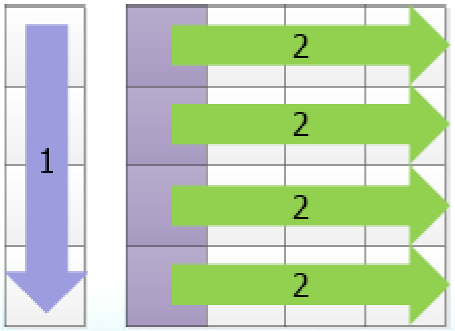
Width : 512
Height 512

5

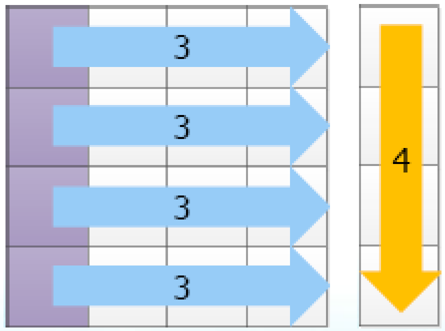


2D Memory Allocation

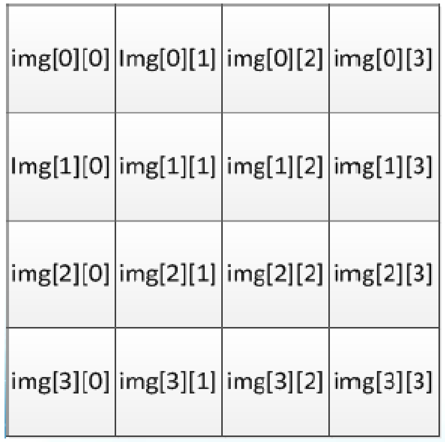
```
//  Memory allocation
unsigned char **img_in = 0;
img_in = (unsigned char **)malloc(sizeof(unsigned char*) * HEIGH);      //  1

for( i = 0 ; i < HEIGH ; i++){                                          //  2
    img_in[i] = (unsigned char *)malloc(sizeof(unsigned char) * WIDTH);
}


// Memory free
for( i = 0 ; i < HEIGH ; i++){       //  3
    free( img_in[i] );
}

free( img_in);                       //  4
```



< Memory allocation >



< Memory free >



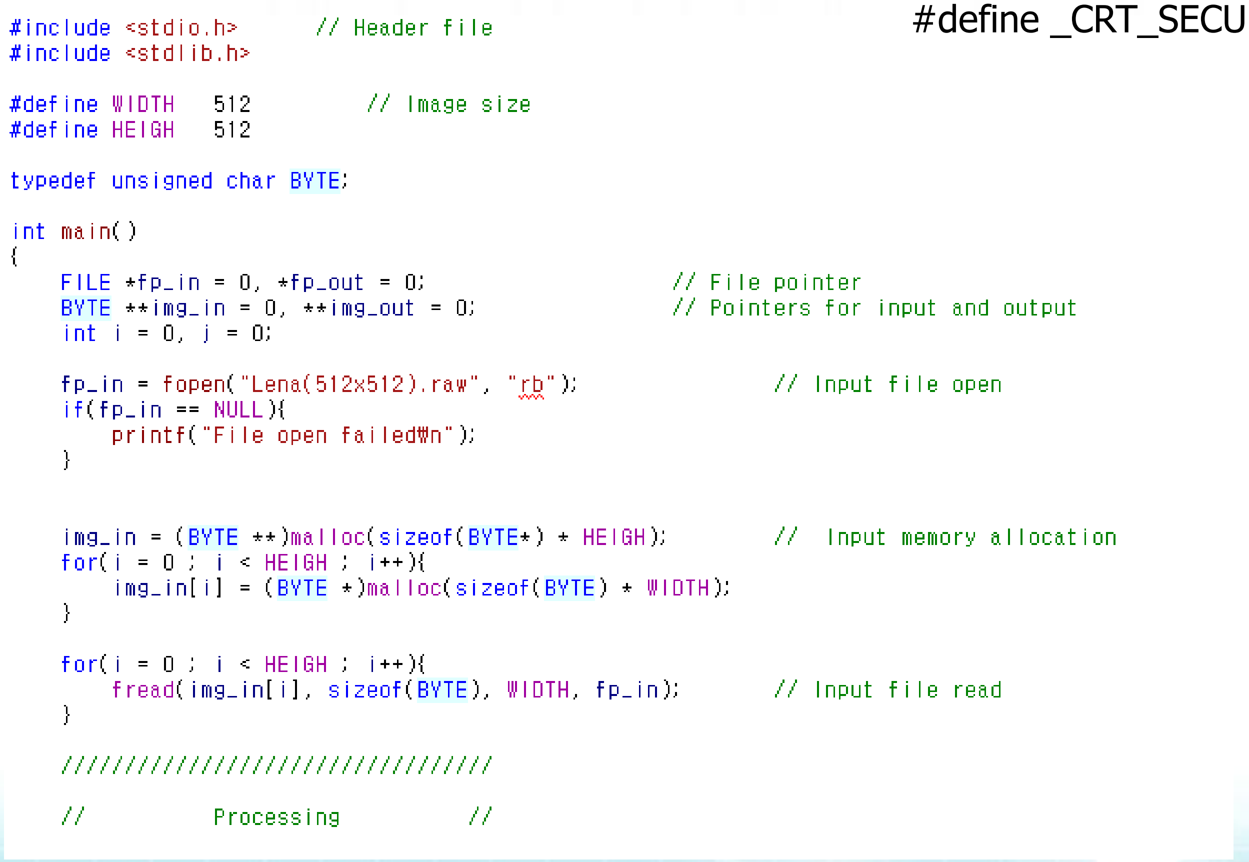
img[0][0]	Img[0][1]	img[0][2]	img[0][3]
Img[1][0]	img[1][1]	img[1][2]	img[1][3]
img[2][0]	img[2][1]	img[2][2]	img[2][3]
img[3][0]	img[3][1]	img[3][2]	img[3][3]

< 2D array >

Example

❖ Image file I/O

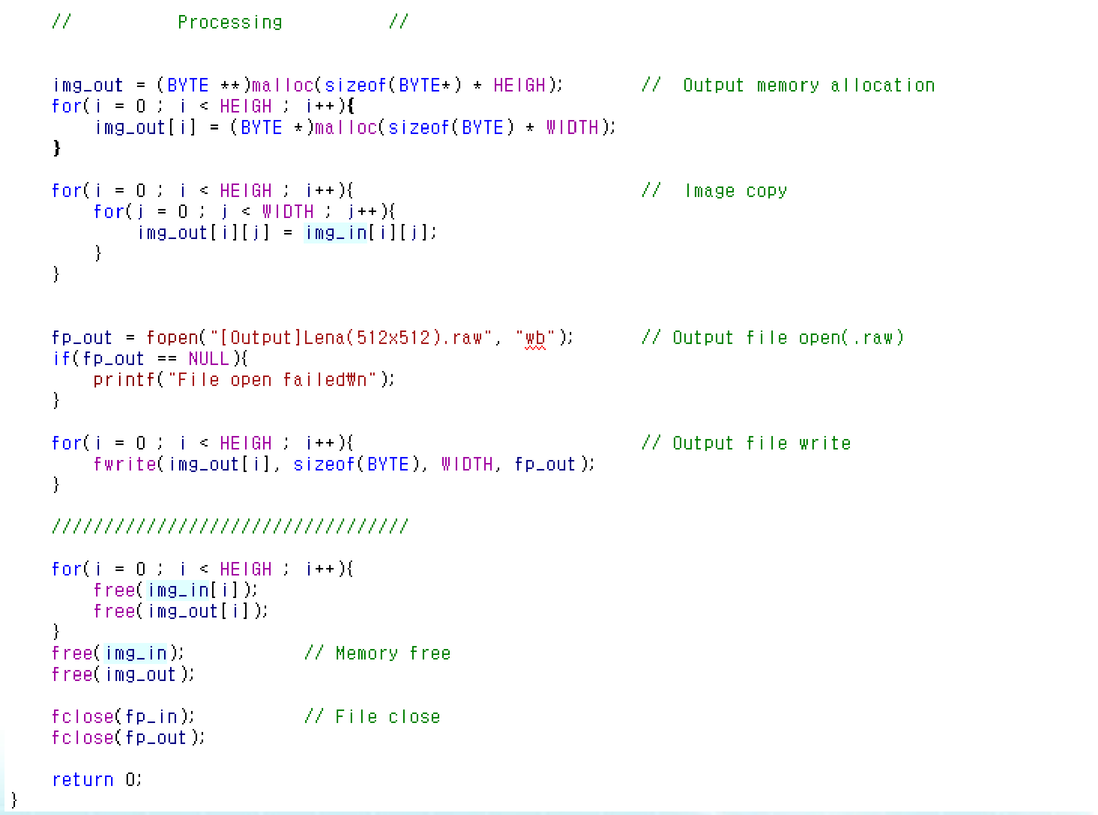
#define _CRT_SECURE_NO_WARNINGS



```
#include <stdio.h>      // Header file
#include <stdlib.h>

#define WIDTH   512         // Image size
#define HEIGH   512

typedef unsigned char BYTE;

int main()
{
    FILE *fp_in = 0, *fp_out = 0;                       // File pointer
    BYTE **img_in = 0, **img_out = 0;                   // Pointers for input and output
    int i = 0, j = 0;

    fp_in = fopen("Lena(512x512).raw", "rb");                   // Input file open
    if(fp_in == NULL){
        printf("File open failed\n");
    }


    img_in = (BYTE **)malloc(sizeof(BYTE*) * HEIGH);          //  Input memory allocation
    for(i = 0 ; i < HEIGH ; i++){
        img_in[i] = (BYTE *)malloc(sizeof(BYTE) * WIDTH);
    }

    for(i = 0 ; i < HEIGH ; i++){
        fread(img_in[i], sizeof(BYTE), WIDTH, fp_in);         // Input file read
    }

    //////////////////////////////////////

    //          Processing          //
```

Example

❖ Image file I/O



```
    //          Processing          //

    img_out = (BYTE **)malloc(sizeof(BYTE*) * HEIGH);        //  Output memory allocation
    for(i = 0 ; i < HEIGH ; i++){
        img_out[i] = (BYTE *)malloc(sizeof(BYTE) * WIDTH);
    }

    for(i = 0 ; i < HEIGH ; i++){                            //  Image copy
        for(j = 0 ; j < WIDTH ; j++){
            img_out[i][j] = img_in[i][j];
        }
    }


    fp_out = fopen("[Output]Lena(512x512).raw", "wb");      // Output file open(.raw)
    if(fp_out == NULL){
        printf("File open failed₩n");
    }

    for(i = 0 ; i < HEIGH ; i++){                            // Output file write
        fwrite(img_out[i], sizeof(BYTE), WIDTH, fp_out);
    }

    //////////////////////////////////

    for(i = 0 ; i < HEIGH ; i++){
        free(img_in[i]);
        free(img_out[i]);
    }
    free(img_in);           // Memory free
    free(img_out);

    fclose(fp_in);          // File close
    fclose(fp_out);

    return 0;
}
```

Example

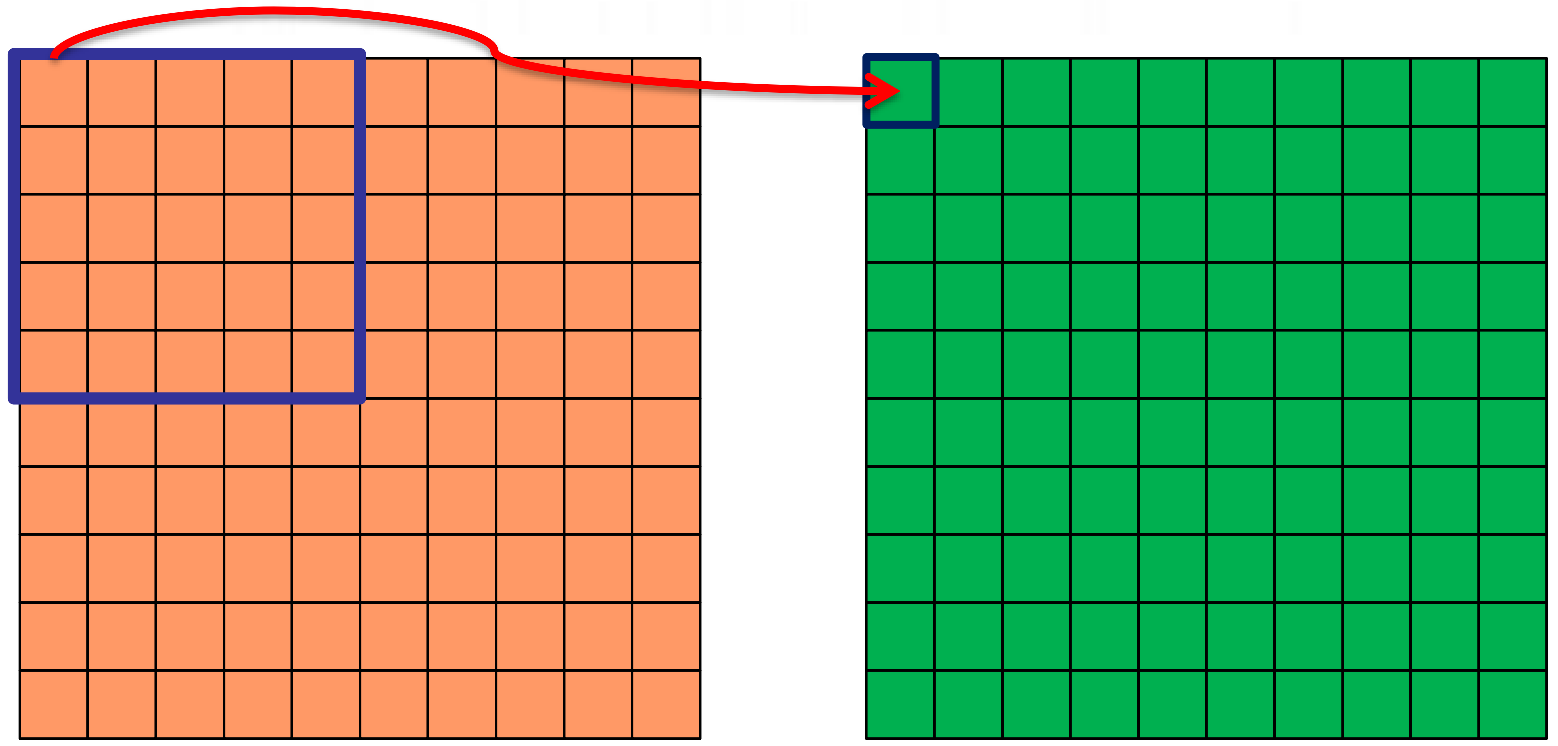
- Image file I/O

< Input image >

< Output image >

Example

- 5x5 average

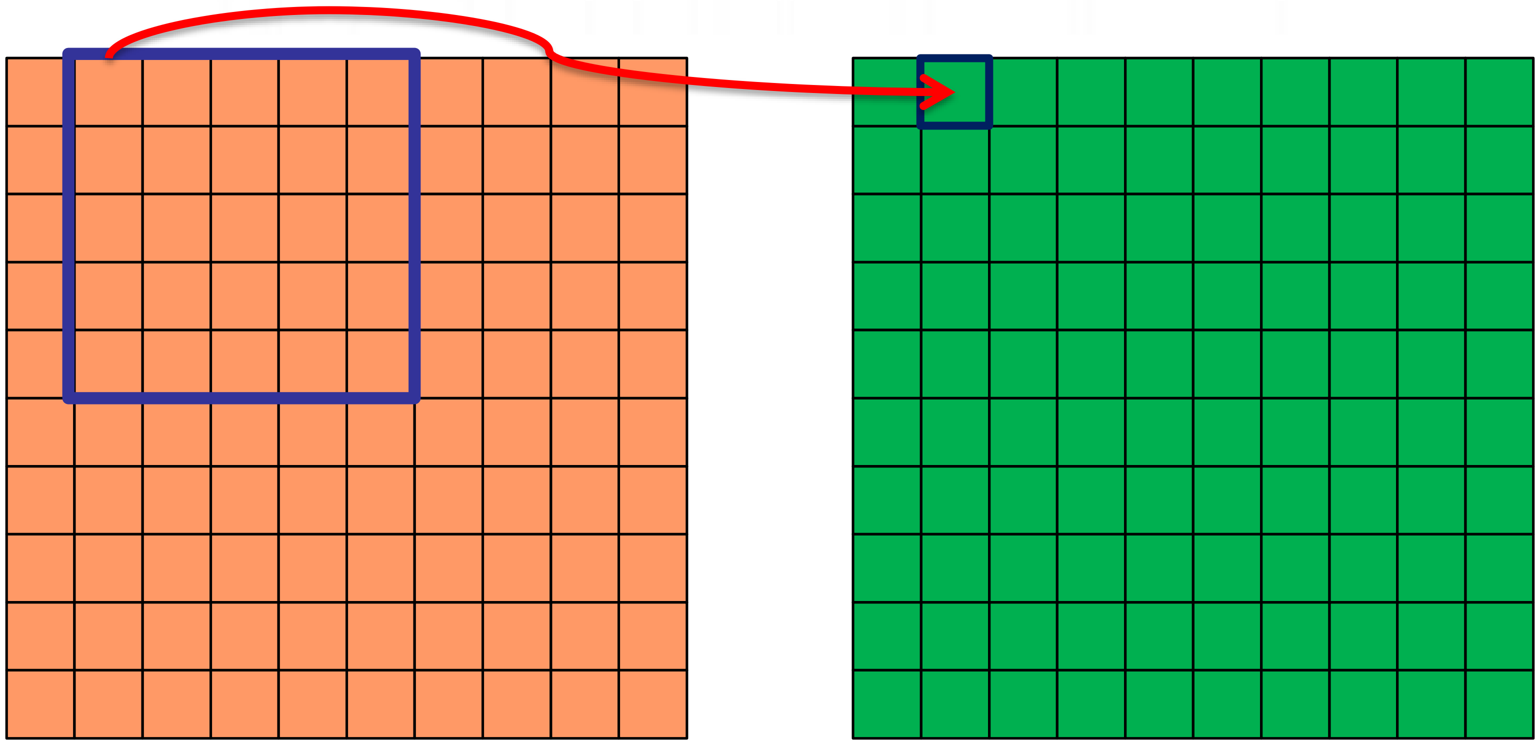


< Input image >

< Output image >

Example

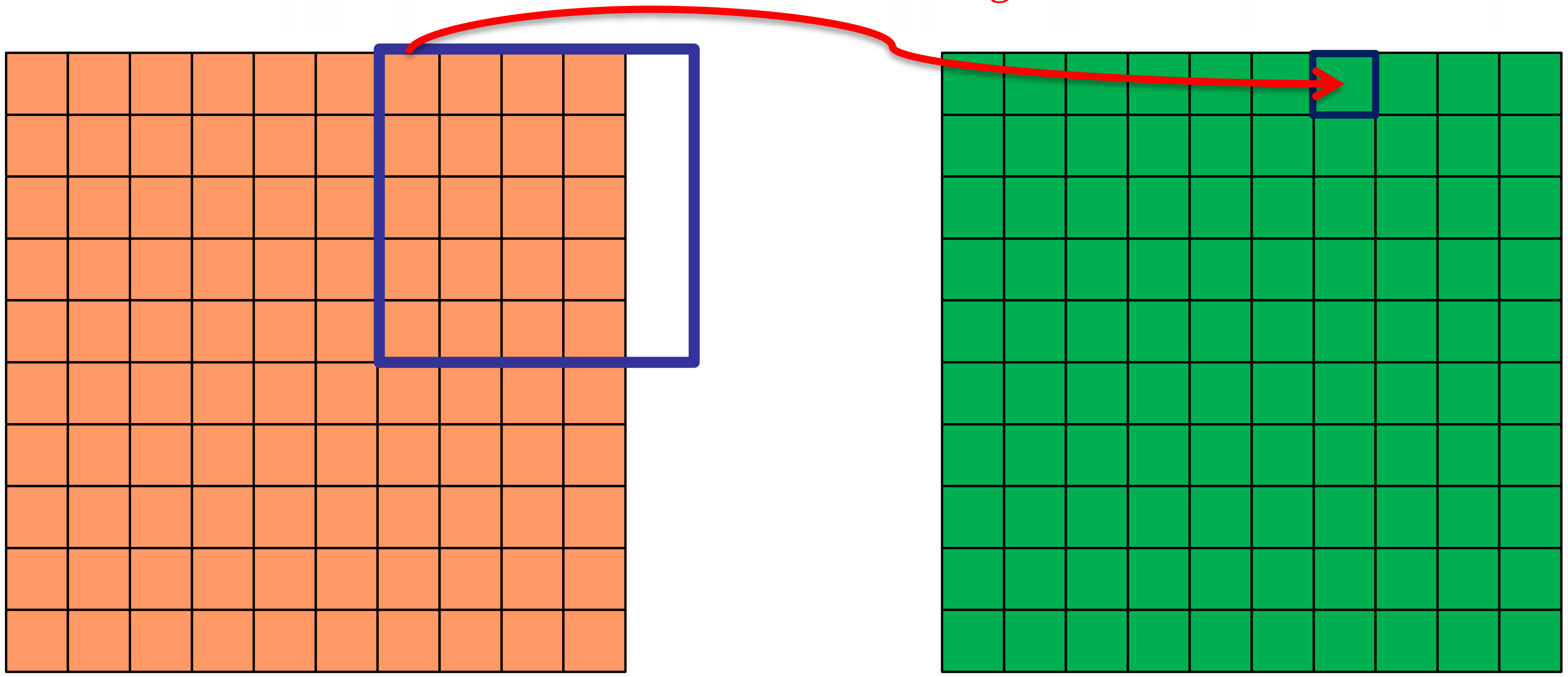
- 5x5 average



< Input image >   < Output image >

Example

- 5x5 average

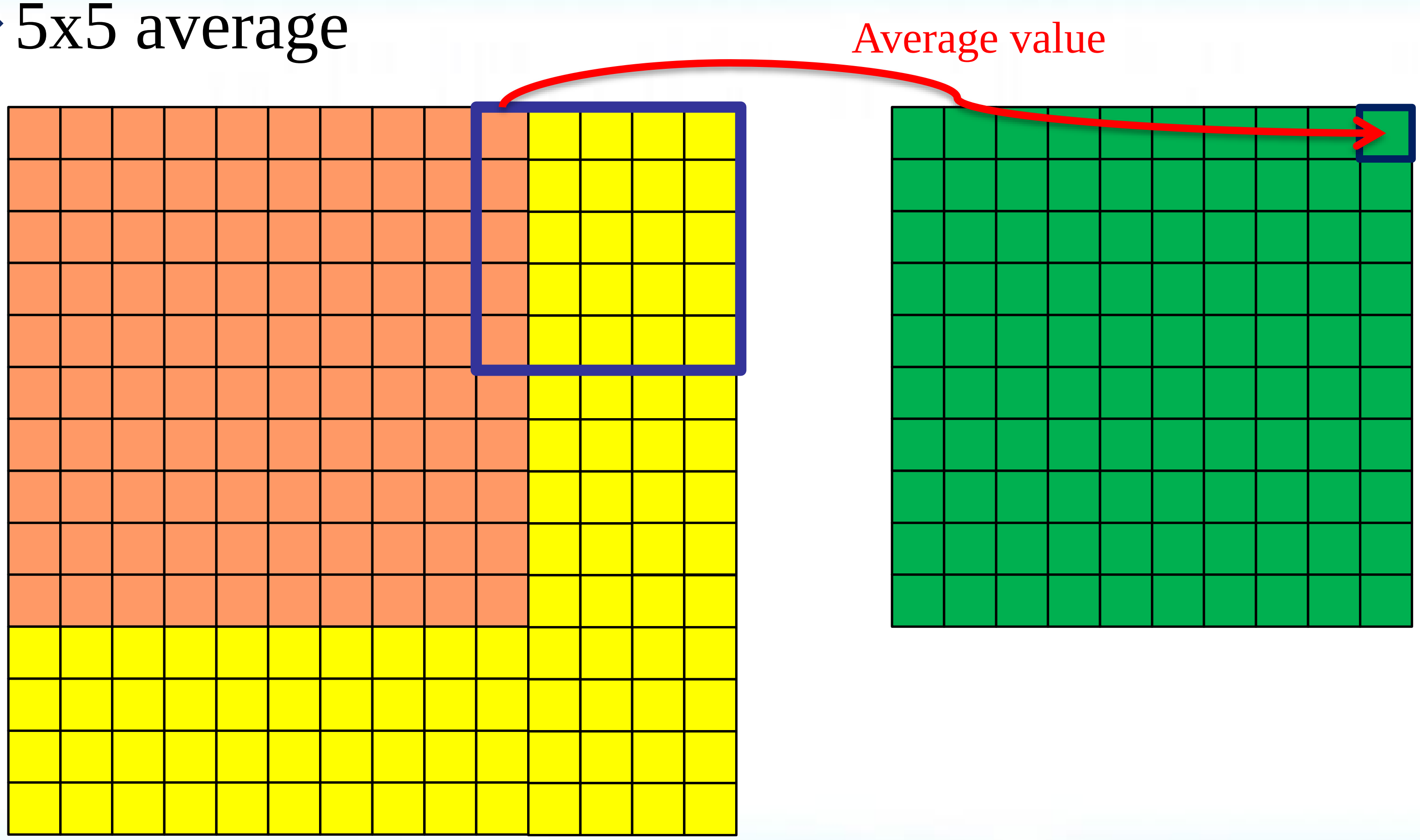


< Input image >

< Output image >

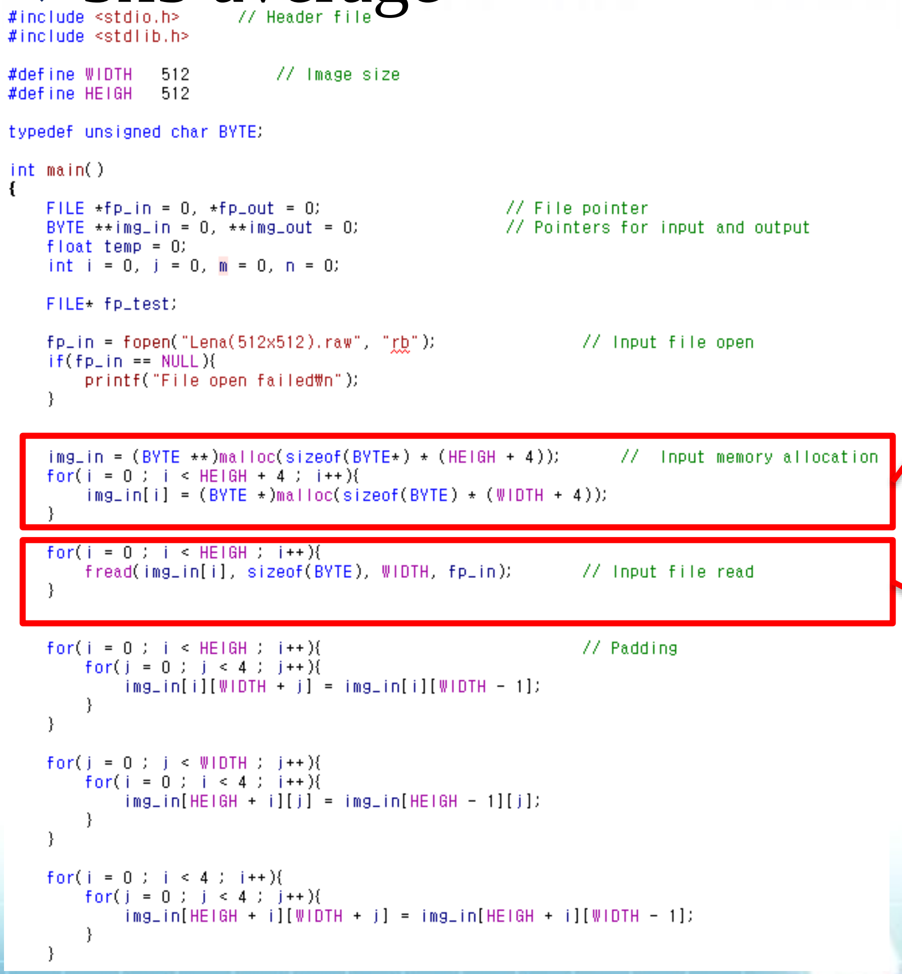
Example

- 5x5 average

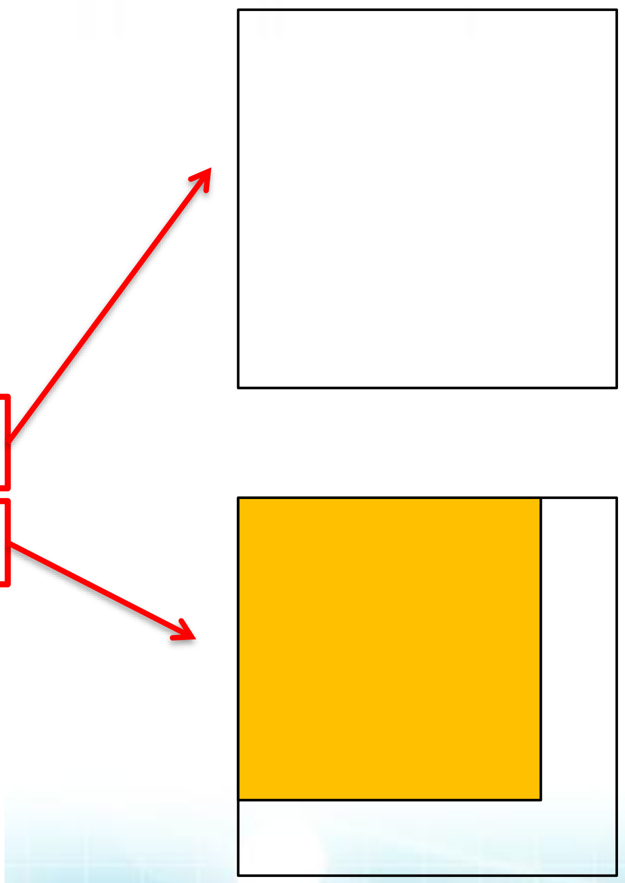


Example

❖ 5x5 average

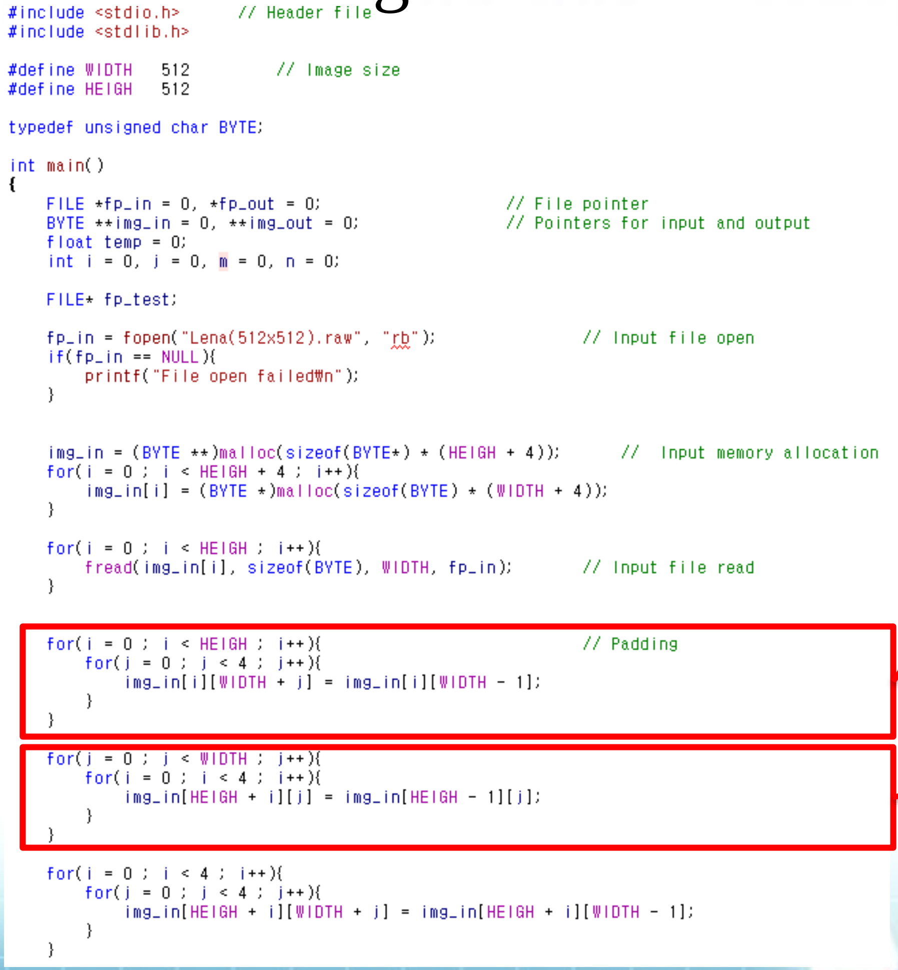


```c
#include <stdio.h>      // Header file
#include <stdlib.h>

#define WIDTH   512         // Image size
#define HEIGH   512

typedef unsigned char BYTE;

int main()
{
    FILE *fp_in = 0, *fp_out = 0;                   // File pointer
    BYTE **img_in = 0, **img_out = 0;               // Pointers for input and output
    float temp = 0;
    int i = 0, j = 0, m = 0, n = 0;

    FILE* fp_test;

    fp_in = fopen("Lena(512x512).raw", "rb");               // Input file open
    if(fp_in == NULL){
        printf("File open failed₩n");
    }

    img_in = (BYTE **)malloc(sizeof(BYTE*) * (HEIGH + 4));      //  Input memory allocation
    for(i = 0 ; i < HEIGH + 4 ; i++){
        img_in[i] = (BYTE *)malloc(sizeof(BYTE) * (WIDTH + 4));
    }

    for(i = 0 ; i < HEIGH ; i++){
        fread(img_in[i], sizeof(BYTE), WIDTH, fp_in);       // Input file read
    }

    for(i = 0 ; i < HEIGH ; i++){                           // Padding
        for(j = 0 ; j < 4 ; j++){
            img_in[i][WIDTH + j] = img_in[i][WIDTH - 1];
        }
    }

    for(j = 0 ; j < WIDTH ; j++){
        for(i = 0 ; i < 4 ; i++){
            img_in[HEIGH + i][j] = img_in[HEIGH - 1][j];
        }
    }

    for(i = 0 ; i < 4 ; i++){
        for(j = 0 ; j < 4 ; j++){
            img_in[HEIGH + i][WIDTH + j] = img_in[HEIGH + i][WIDTH - 1];
        }
    }
```

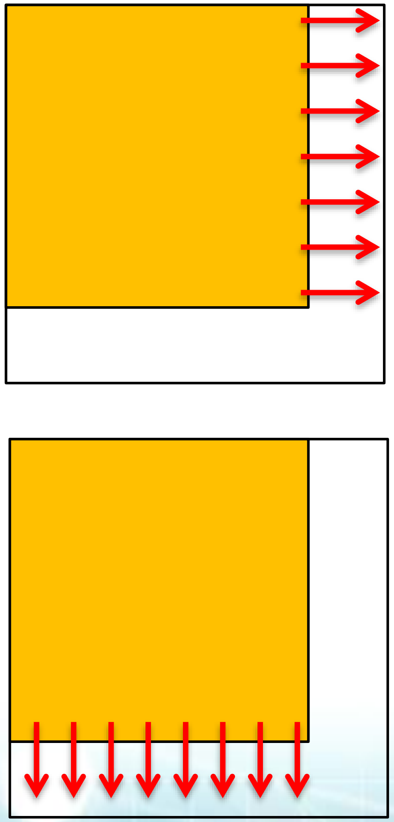


Example

❖ 5x5 average

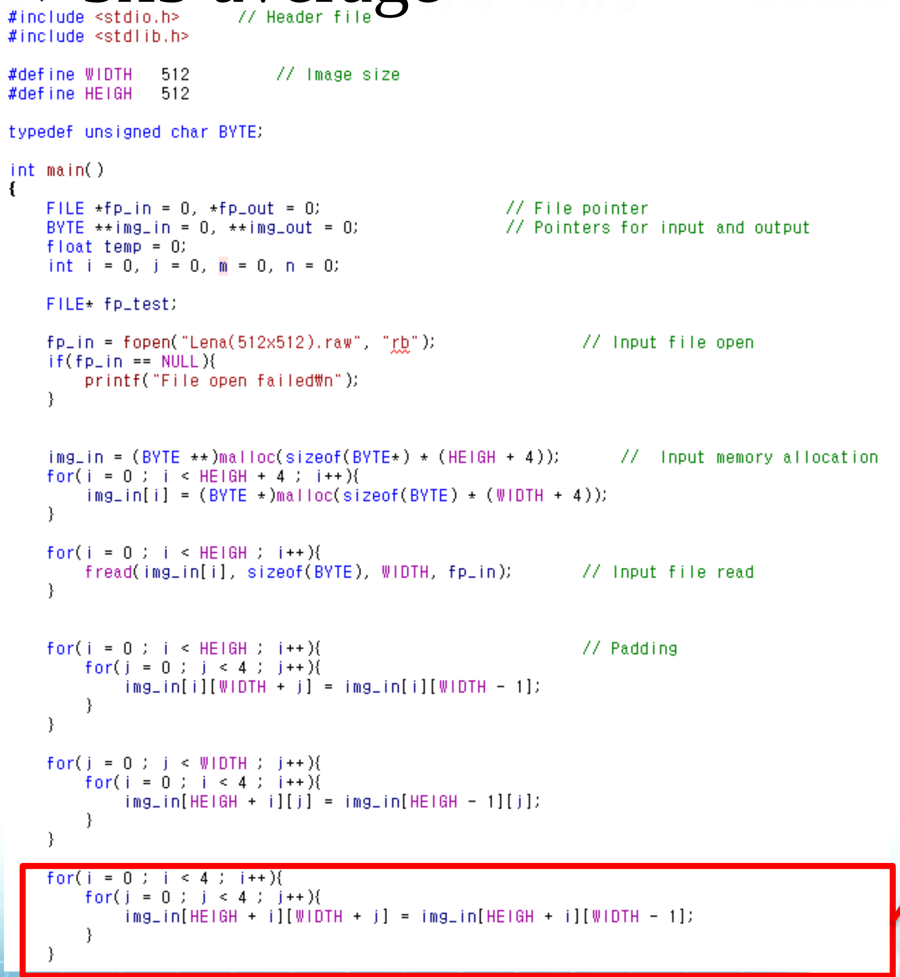


```
#include <stdio.h>      // Header file
#include <stdlib.h>

#define WIDTH   512         // Image size
#define HEIGH   512

typedef unsigned char BYTE;

int main()
{
    FILE *fp_in = 0, *fp_out = 0;                 // File pointer
    BYTE **img_in = 0, **img_out = 0;             // Pointers for input and output
    float temp = 0;
    int i = 0, j = 0, m = 0, n = 0;

    FILE* fp_test;

    fp_in = fopen("Lena(512x512).raw", "rb");             // Input file open
    if(fp_in == NULL){
        printf("File open failed₩n");
    }


    img_in = (BYTE **)malloc(sizeof(BYTE*) * (HEIGH + 4));      //  Input memory allocation
    for(i = 0 ; i < HEIGH + 4 ; i++){
        img_in[i] = (BYTE *)malloc(sizeof(BYTE) * (WIDTH + 4));
    }

    for(i = 0 ; i < HEIGH ; i++){
        fread(img_in[i], sizeof(BYTE), WIDTH, fp_in);       // Input file read
    }

    for(i = 0 ; i < HEIGH ; i++){                         // Padding
        for(j = 0 ; j < 4 ; j++){
            img_in[i][WIDTH + j] = img_in[i][WIDTH - 1];
        }
    }

    for(j = 0 ; j < WIDTH ; j++){
        for(i = 0 ; i < 4 ; i++){
            img_in[HEIGH + i][j] = img_in[HEIGH - 1][j];
        }
    }

    for(i = 0 ; i < 4 ; i++){
        for(j = 0 ; j < 4 ; j++){
            img_in[HEIGH + i][WIDTH + j] = img_in[HEIGH + i][WIDTH - 1];
        }
    }
```

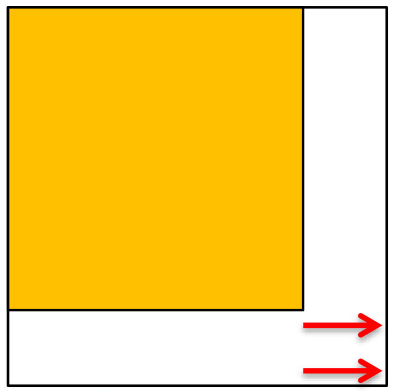


Example

- 5x5 average

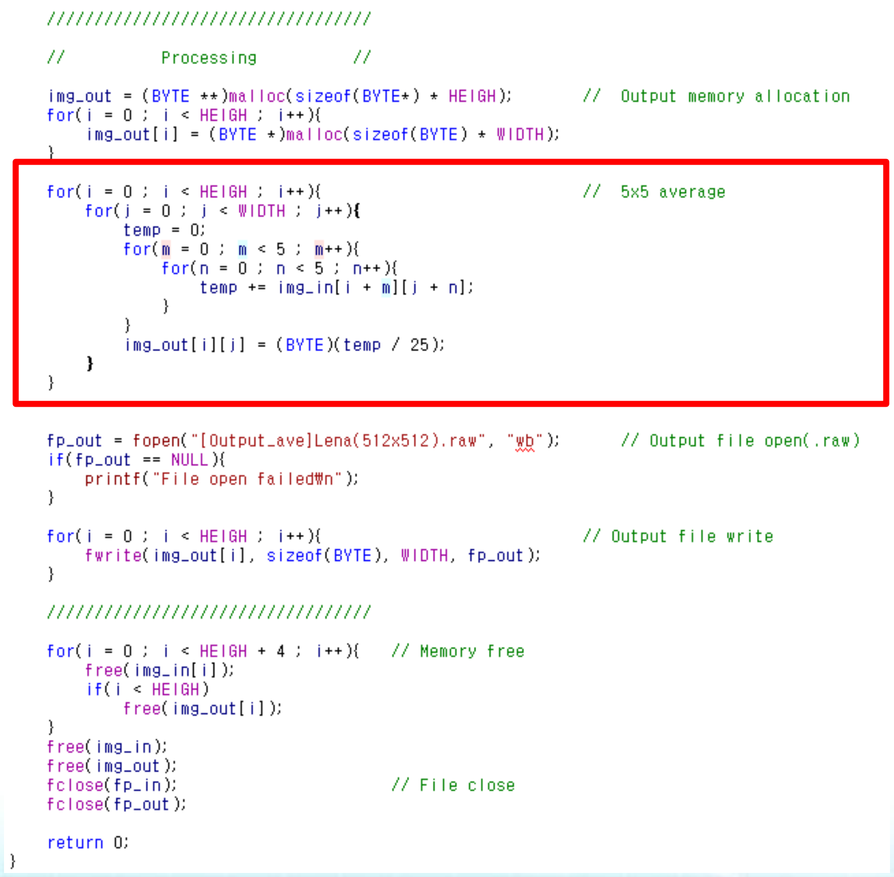


```c
#include <stdio.h>      // Header file
#include <stdlib.h>

#define WIDTH   512         // Image size
#define HEIGH   512

typedef unsigned char BYTE;

int main()
{
    FILE *fp_in = 0, *fp_out = 0;                   // File pointer
    BYTE **img_in = 0, **img_out = 0;               // Pointers for input and output
    float temp = 0;
    int i = 0, j = 0, m = 0, n = 0;

    FILE* fp_test;

    fp_in = fopen("Lena(512x512).raw", "rb");              // Input file open
    if(fp_in == NULL){
        printf("File open failed₩n");
    }


    img_in = (BYTE **)malloc(sizeof(BYTE*) * (HEIGH + 4));      //  Input memory allocation
    for(i = 0 ; i < HEIGH + 4 ; i++){
        img_in[i] = (BYTE *)malloc(sizeof(BYTE) * (WIDTH + 4));
    }

    for(i = 0 ; i < HEIGH ; i++){
        fread(img_in[i], sizeof(BYTE), WIDTH, fp_in);       // Input file read
    }


    for(i = 0 ; i < HEIGH ; i++){                          // Padding
        for(j = 0 ; j < 4 ; j++){
            img_in[i][WIDTH + j] = img_in[i][WIDTH - 1];
        }
    }

    for(j = 0 ; j < WIDTH ; j++){
        for(i = 0 ; i < 4 ; i++){
            img_in[HEIGH + i][j] = img_in[HEIGH - 1][j];
        }
    }

    for(i = 0 ; i < 4 ; i++){
        for(j = 0 ; j < 4 ; j++){
            img_in[HEIGH + i][WIDTH + j] = img_in[HEIGH + i][WIDTH - 1];
        }
    }
```



Example

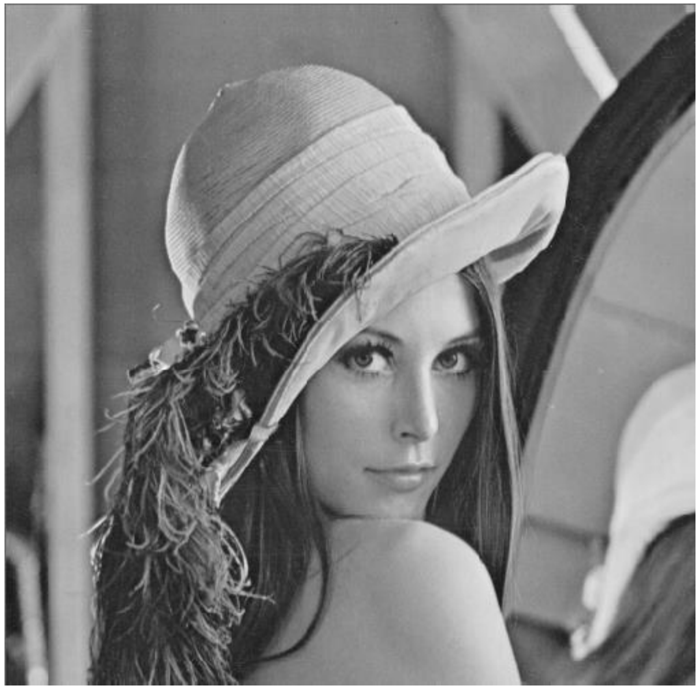
❖ 5x5 average

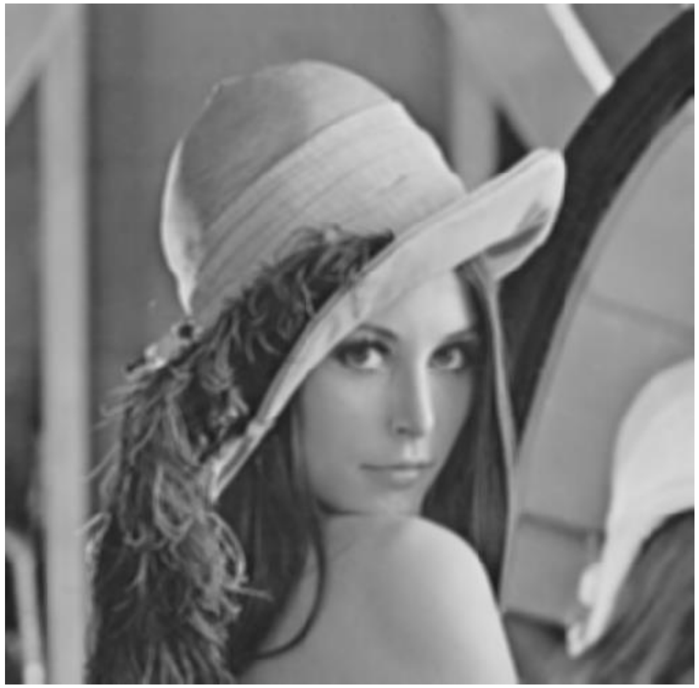


```
    ////////////////////////////////////

    //          Processing          //

    img_out = (BYTE **)malloc(sizeof(BYTE*) * HEIGH);       //  Output memory allocation
    for(i = 0 ; i < HEIGH ; i++){
        img_out[i] = (BYTE *)malloc(sizeof(BYTE) * WIDTH);
    }

    for(i = 0 ; i < HEIGH ; i++){                           //  5x5 average
        for(j = 0 ; j < WIDTH ; j++){
            temp = 0;
            for(m = 0 ; m < 5 ; m++){
                for(n = 0 ; n < 5 ; n++){
                    temp += img_in[i + m][j + n];
                }
            }
            img_out[i][j] = (BYTE)(temp / 25);
        }
    }

    fp_out = fopen("[Output_ave]Lena(512x512).raw", "wb");      // Output file open(.raw)
    if(fp_out == NULL){
        printf("File open failed₩n");
    }

    for(i = 0 ; i < HEIGH ; i++){                           // Output file write
        fwrite(img_out[i], sizeof(BYTE), WIDTH, fp_out);
    }

    ////////////////////////////////////

    for(i = 0 ; i < HEIGH + 4 ; i++){    // Memory free
        free(img_in[i]);
        if(i < HEIGH)
            free(img_out[i]);
    }
    free(img_in);
    free(img_out);
    fclose(fp_in);                      // File close
    fclose(fp_out);

    return 0;
}
```

Example

- 5x5 average

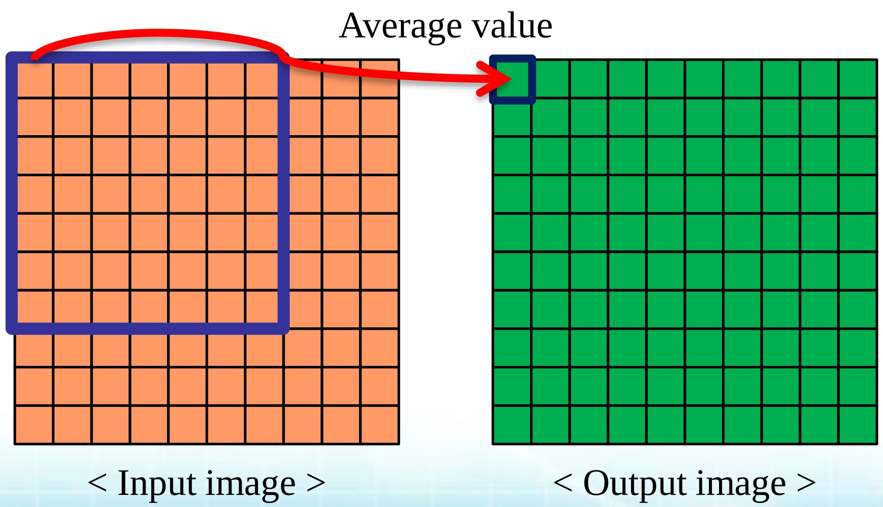

< Input image >


< Output image >

Assignment

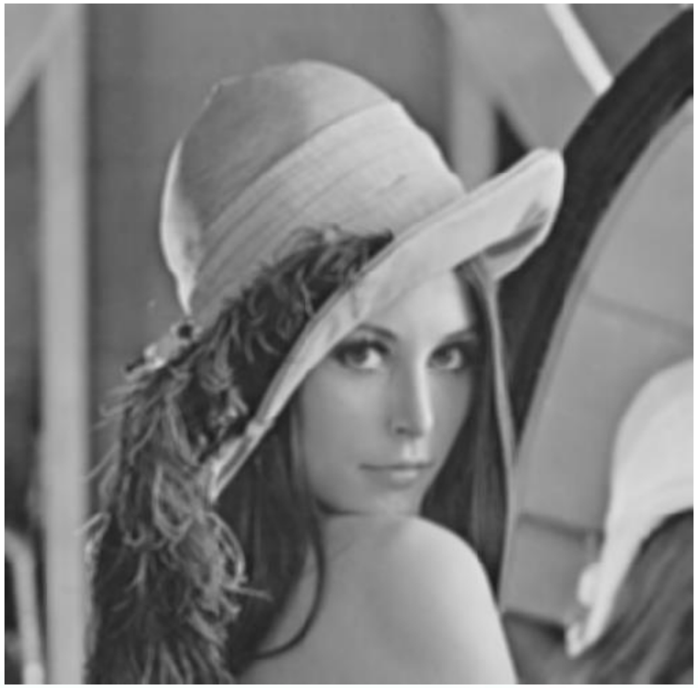
Assignment

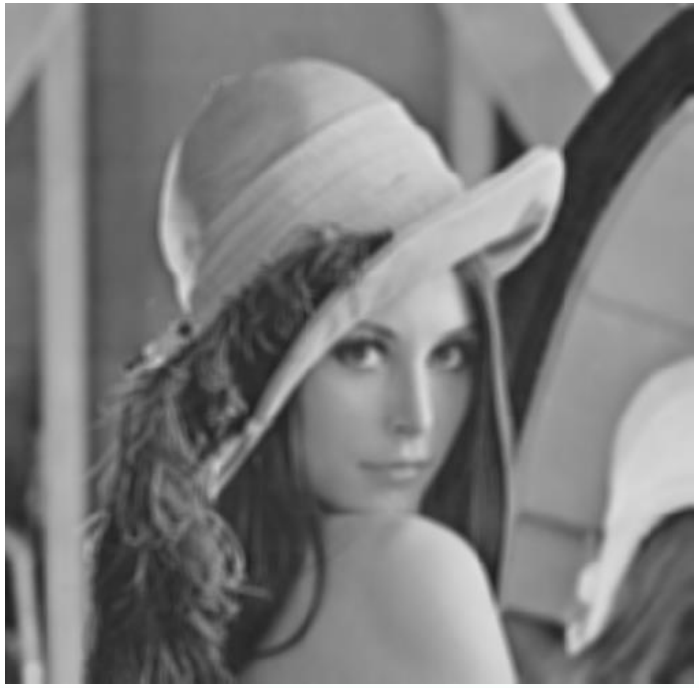
- Assignment #1
  - Image file I/O
    - Same as the example
- Assignment #2
  - 7x7 average



Assignment

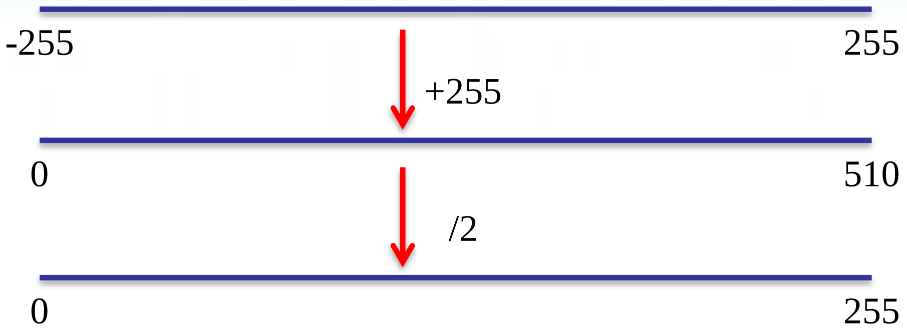
- Assignment #2
  - 5x5 average vs. 7x7 average

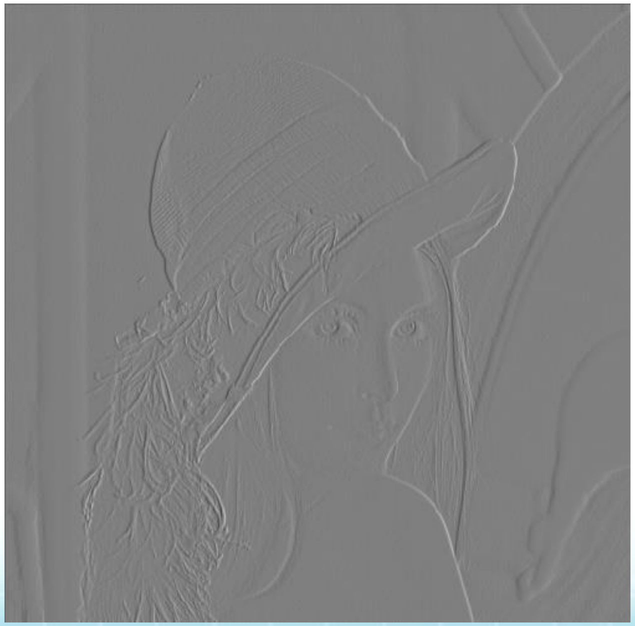

< 5x5 average >

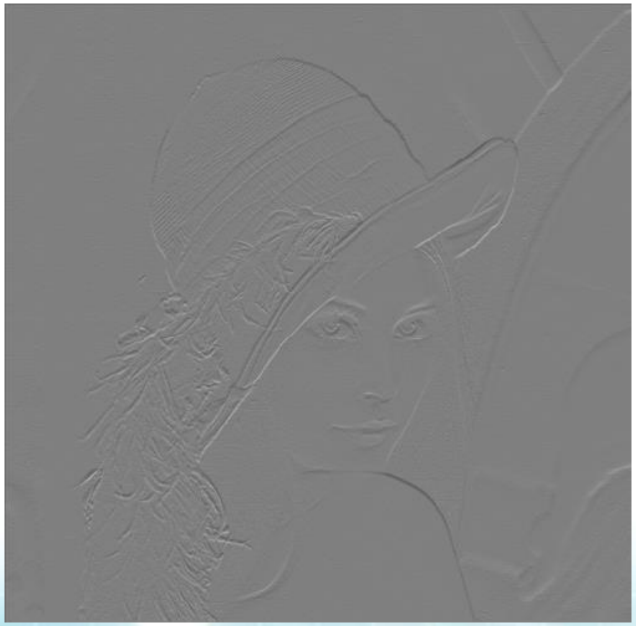

< 7x7 average >

Assignment

- Assignment #3
  - $y(i,j) = x(i,j) - x(i,j+1)$
  - $y(i,j) = x(i,j) - x(i+1,j)$

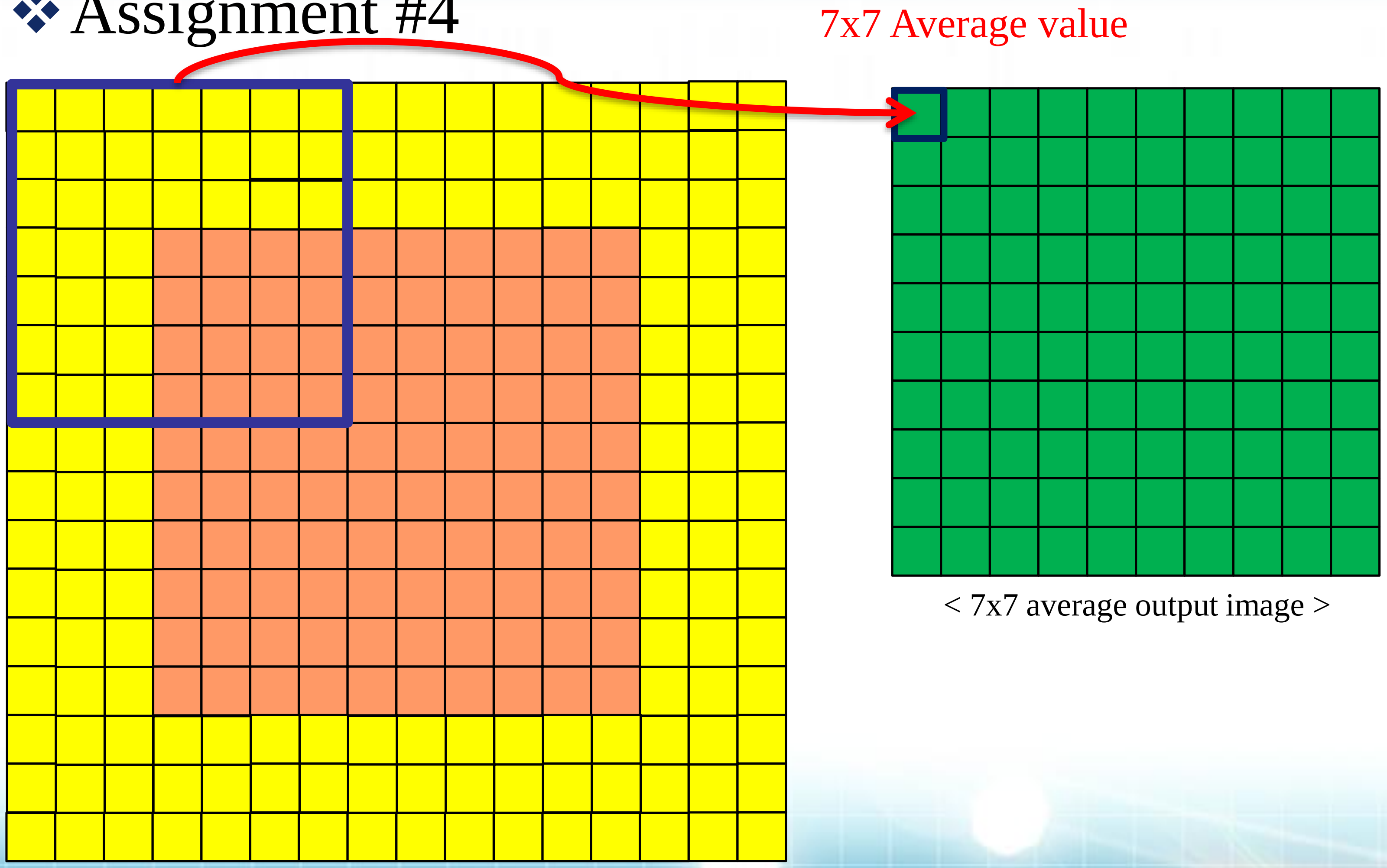



< $y(i,j) = x(i,j) - x(i,j+1)$ >


< $y(i,j) = x(i,j) - x(i+1,j)$ >

Assignment

- Assignment #4



< Padded input image >

< 7x7 average output image >