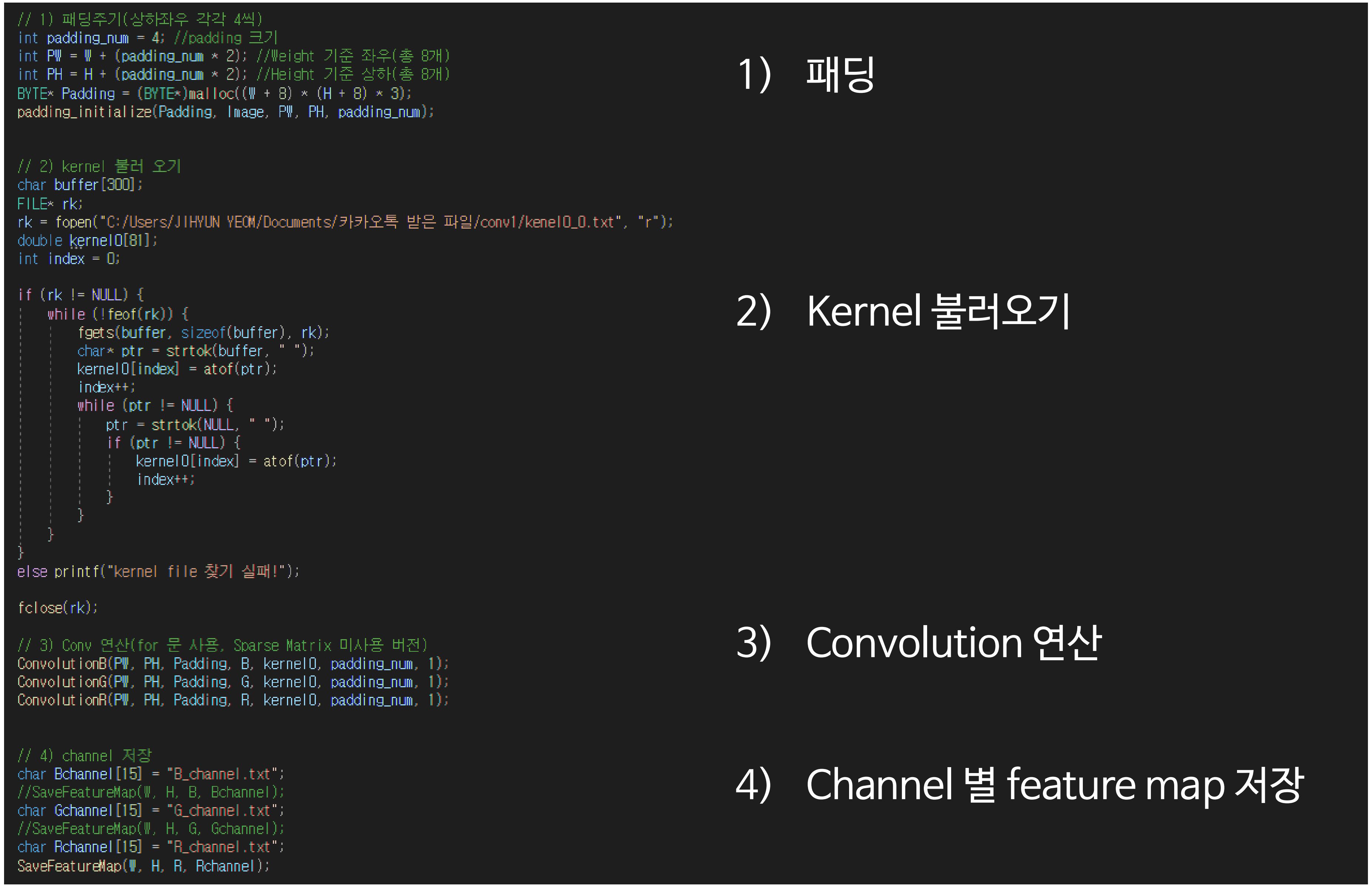
Convolution 연산

염지현

Convolution 연산 수행



```c
// 1) 패딩주기(상하좌우 각각 4씩)
int padding_num = 4; //padding 크기
int PW = W + (padding_num * 2); //Weight 기준 좌우(총 8개)
int PH = H + (padding_num * 2); //Height 기준 상하(총 8개)
BYTE* Padding = (BYTE*)malloc((W + 8) * (H + 8) * 3);
padding_initialize(Padding, Image, PW, PH, padding_num);


// 2) kernel 불러 오기
char buffer[300];
FILE* rk;
rk = fopen("C:/Users/JIHYUN YEOM/Documents/카카오톡 받은 파일/conv1/kenel0_0.txt", "r");
double kernel0[81];
int index = 0;

if (rk != NULL) {
    while (!feof(rk)) {
        fgets(buffer, sizeof(buffer), rk);
        char* ptr = strtok(buffer, " ");
        kernel0[index] = atof(ptr);
        index++;
        while (ptr != NULL) {
            ptr = strtok(NULL, " ");
            if (ptr != NULL) {
                kernel0[index] = atof(ptr);
                index++;
            }
        }
    }
}
else printf("kernel file 찾기 실패!");

fclose(rk);

// 3) Conv 연산(for 문 사용, Sparse Matrix 미사용 버전)
ConvolutionB(PW, PH, Padding, B, kernel0, padding_num, 1);
ConvolutionG(PW, PH, Padding, G, kernel0, padding_num, 1);
ConvolutionR(PW, PH, Padding, R, kernel0, padding_num, 1);


// 4) channel 저장
char Bchannel[15] = "B_channel.txt";
//SaveFeatureMap(W, H, B, Bchannel);
char Gchannel[15] = "G_channel.txt";
//SaveFeatureMap(W, H, G, Gchannel);
char Rchannel[15] = "R_channel.txt";
SaveFeatureMap(W, H, R, Rchannel);
```

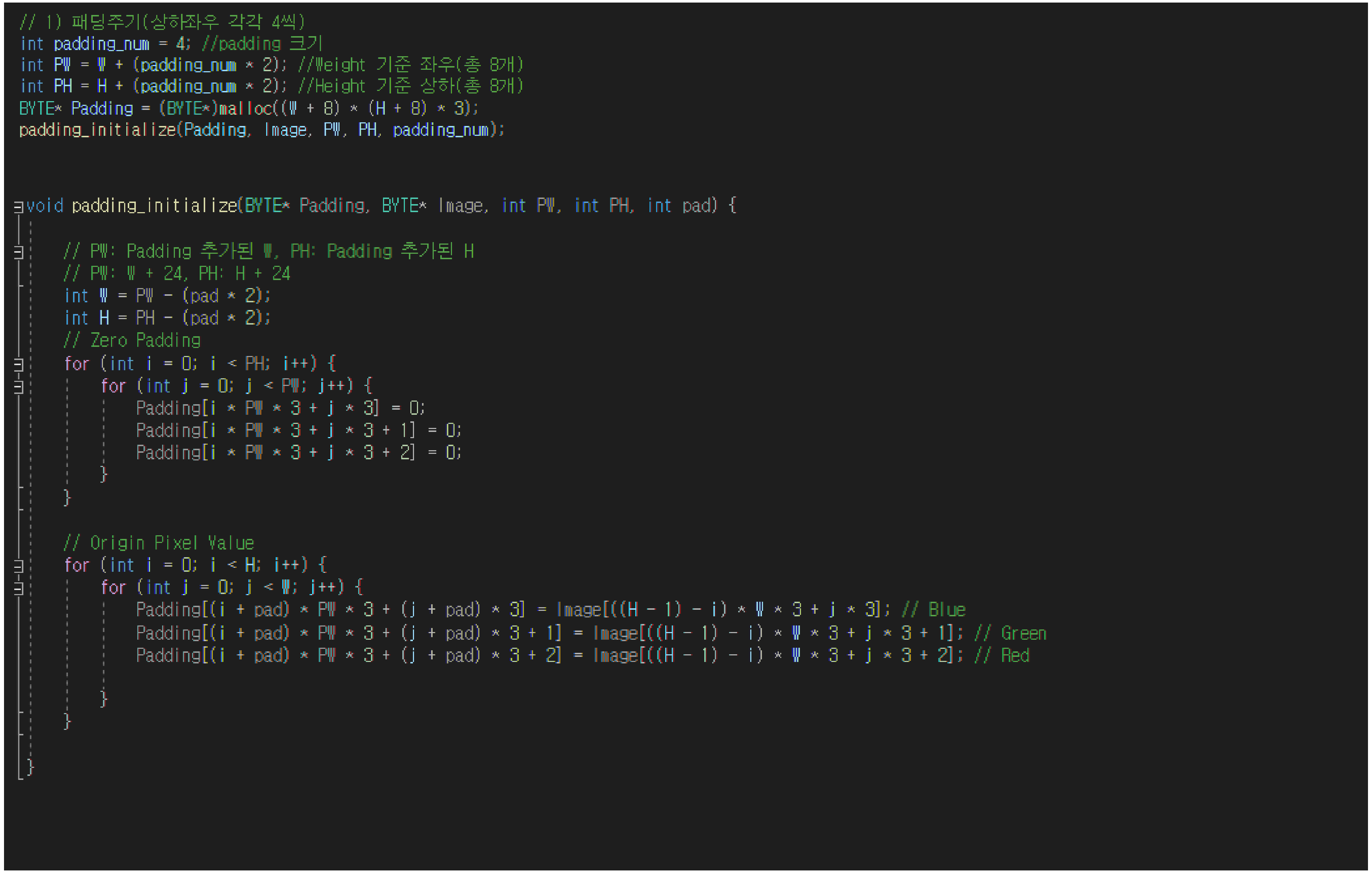
1) 패딩

2) Kernel 불러오기

3) Convolution 연산

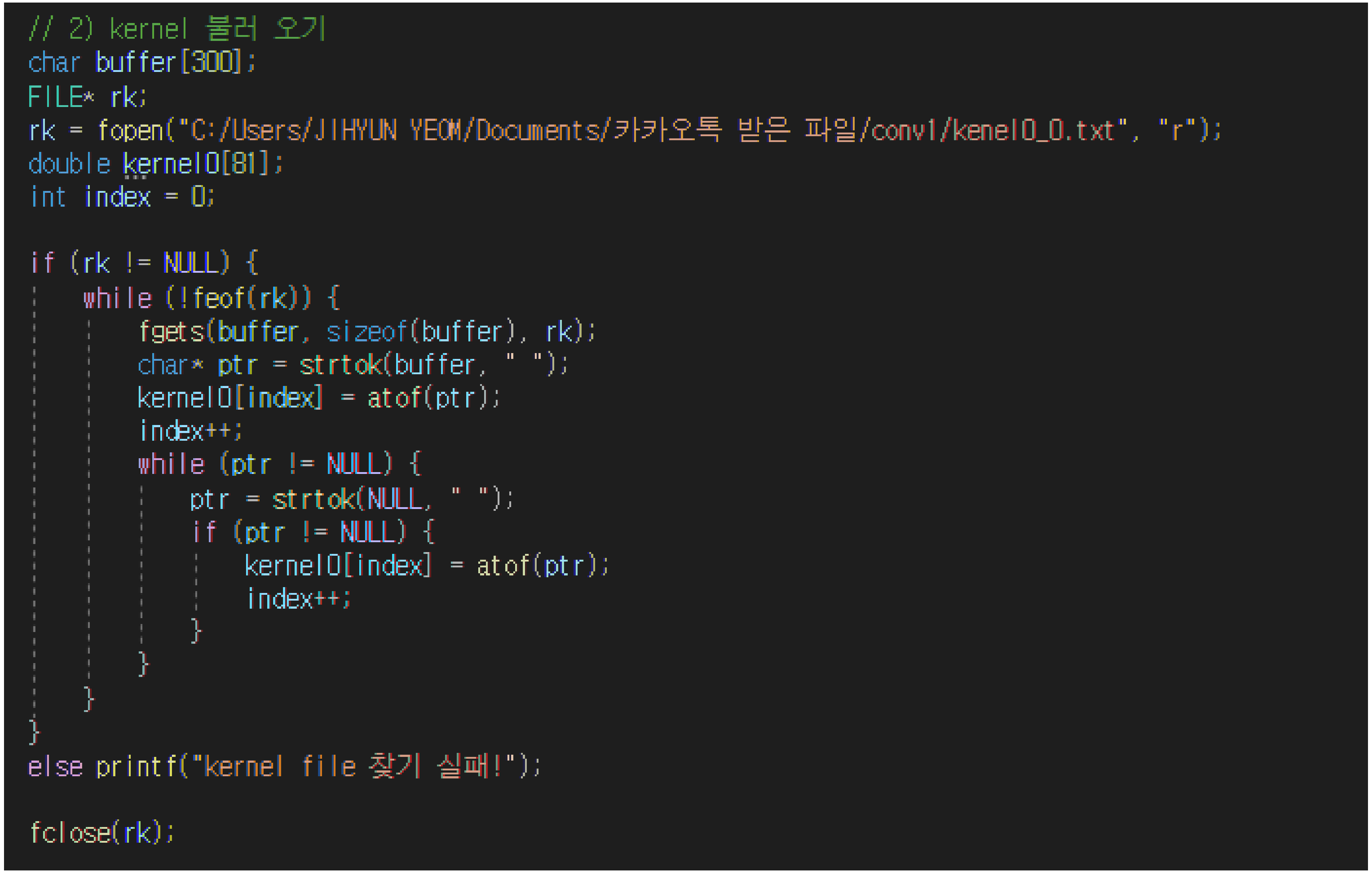
4) Channel 별 feature map 저장

1) 패딩



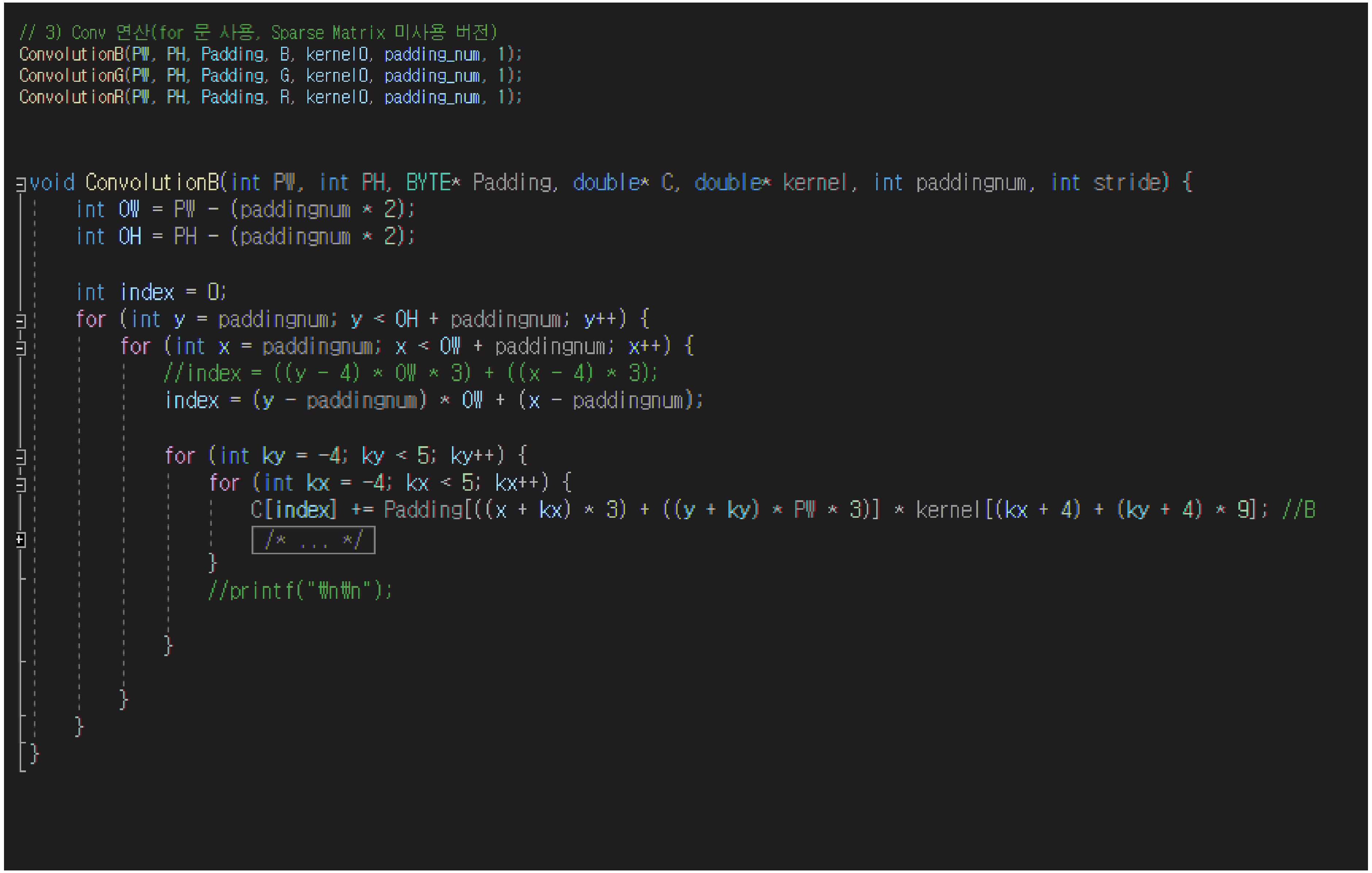
```c
// 1) 패딩주기(상하좌우 각각 4씩)
int padding_num = 4; //padding 크기
int PW = W + (padding_num * 2); //Weight 기준 좌우(총 8개)
int PH = H + (padding_num * 2); //Height 기준 상하(총 8개)
BYTE* Padding = (BYTE*)malloc((W + 8) * (H + 8) * 3);
padding_initialize(Padding, Image, PW, PH, padding_num);


void padding_initialize(BYTE* Padding, BYTE* Image, int PW, int PH, int pad) {

    // PW: Padding 추가된 W, PH: Padding 추가된 H
    // PW: W + 24, PH: H + 24
    int W = PW - (pad * 2);
    int H = PH - (pad * 2);
    // Zero Padding
    for (int i = 0; i < PH; i++) {
        for (int j = 0; j < PW; j++) {
            Padding[i * PW * 3 + j * 3] = 0;
            Padding[i * PW * 3 + j * 3 + 1] = 0;
            Padding[i * PW * 3 + j * 3 + 2] = 0;
        }
    }

    // Origin Pixel Value
    for (int i = 0; i < H; i++) {
        for (int j = 0; j < W; j++) {
            Padding[(i + pad) * PW * 3 + (j + pad) * 3] = Image[((H - 1) - i) * W * 3 + j * 3]; // Blue
            Padding[(i + pad) * PW * 3 + (j + pad) * 3 + 1] = Image[((H - 1) - i) * W * 3 + j * 3 + 1]; // Green
            Padding[(i + pad) * PW * 3 + (j + pad) * 3 + 2] = Image[((H - 1) - i) * W * 3 + j * 3 + 2]; // Red

        }
    }

}
```

2) Kernel 입출력



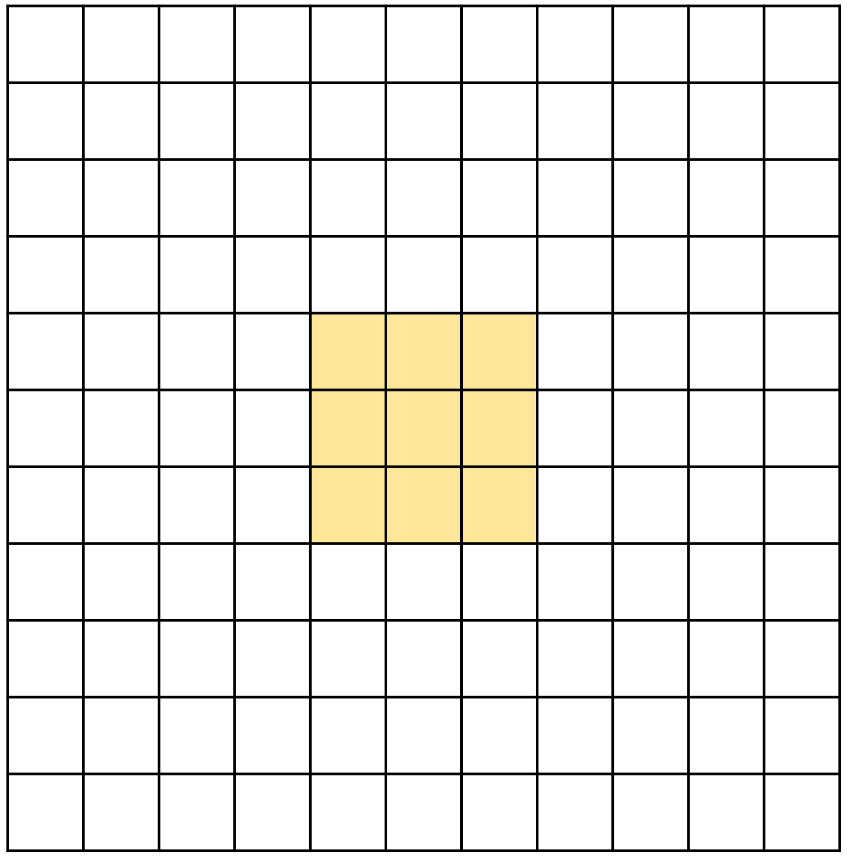
```
// 2) kernel 불러 오기
char buffer[300];
FILE* rk;
rk = fopen("C:/Users/JIHYUN YEOM/Documents/카카오톡 받은 파일/conv1/kenel0_0.txt", "r");
double kernel0[81];
int index = 0;

if (rk != NULL) {
    while (!feof(rk)) {
        fgets(buffer, sizeof(buffer), rk);
        char* ptr = strtok(buffer, " ");
        kernel0[index] = atof(ptr);
        index++;
        while (ptr != NULL) {
            ptr = strtok(NULL, " ");
            if (ptr != NULL) {
                kernel0[index] = atof(ptr);
                index++;
            }
        }
    }
}
else printf("kernel file 찾기 실패!");

fclose(rk);
```

3) Convolution 연산



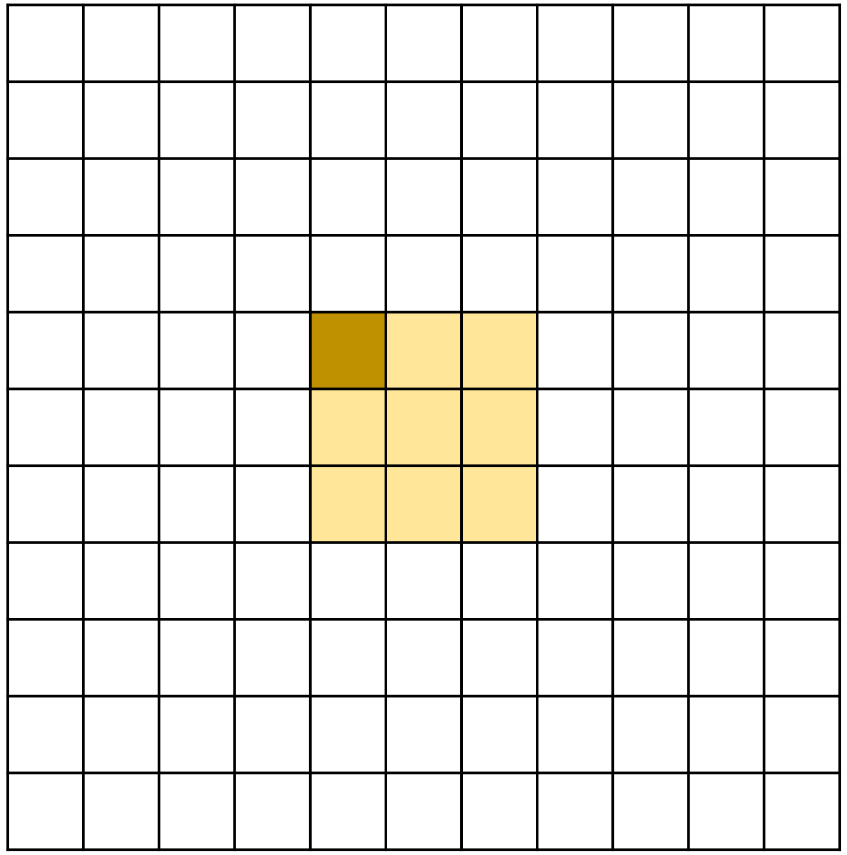
```
// 3) Conv 연산(for 문 사용, Sparse Matrix 미사용 버전)
ConvolutionB(PW, PH, Padding, B, kernel0, padding_num, 1);
ConvolutionG(PW, PH, Padding, G, kernel0, padding_num, 1);
ConvolutionR(PW, PH, Padding, R, kernel0, padding_num, 1);


void ConvolutionB(int PW, int PH, BYTE* Padding, double* C, double* kernel, int paddingnum, int stride) {
    int OW = PW - (paddingnum * 2);
    int OH = PH - (paddingnum * 2);

    int index = 0;
    for (int y = paddingnum; y < OH + paddingnum; y++) {
        for (int x = paddingnum; x < OW + paddingnum; x++) {
            //index = ((y - 4) * OW * 3) + ((x - 4) * 3);
            index = (y - paddingnum) * OW + (x - paddingnum);

            for (int ky = -4; ky < 5; ky++) {
                for (int kx = -4; kx < 5; kx++) {
                    C[index] += Padding[((x + kx) * 3) + ((y + ky) * PW * 3)] * kernel[(kx + 4) + (ky + 4) * 9]; //B
                    /* ... */
                }
                //printf("₩n₩n");

            }


        }
    }
}
```

3) Convolution 연산



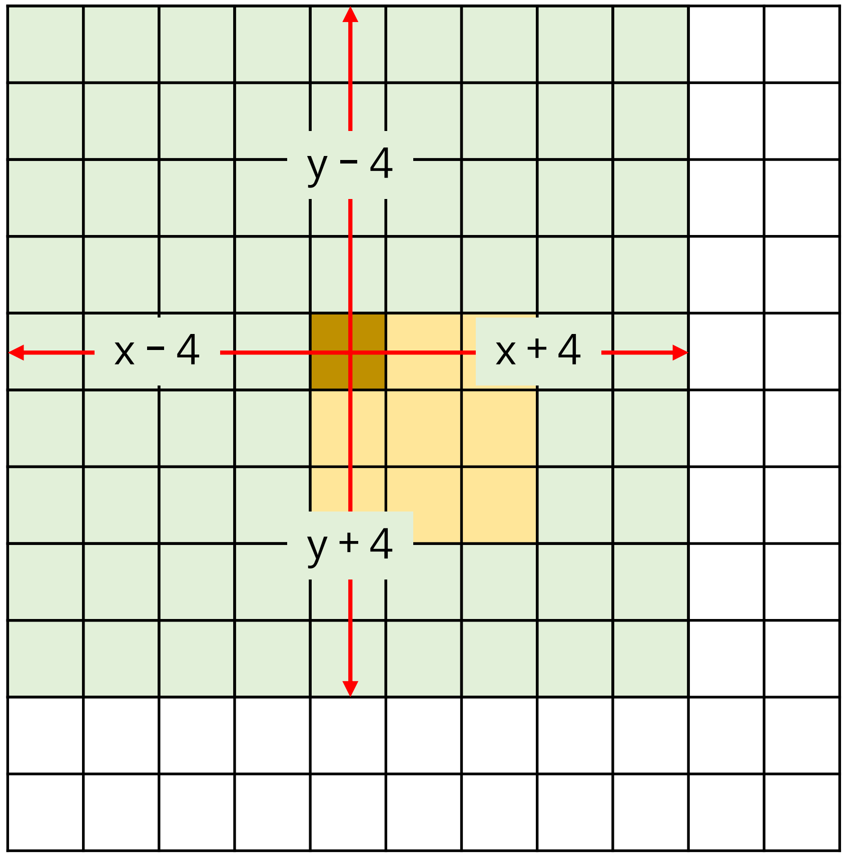
Output 기준

3) Convolution 연산



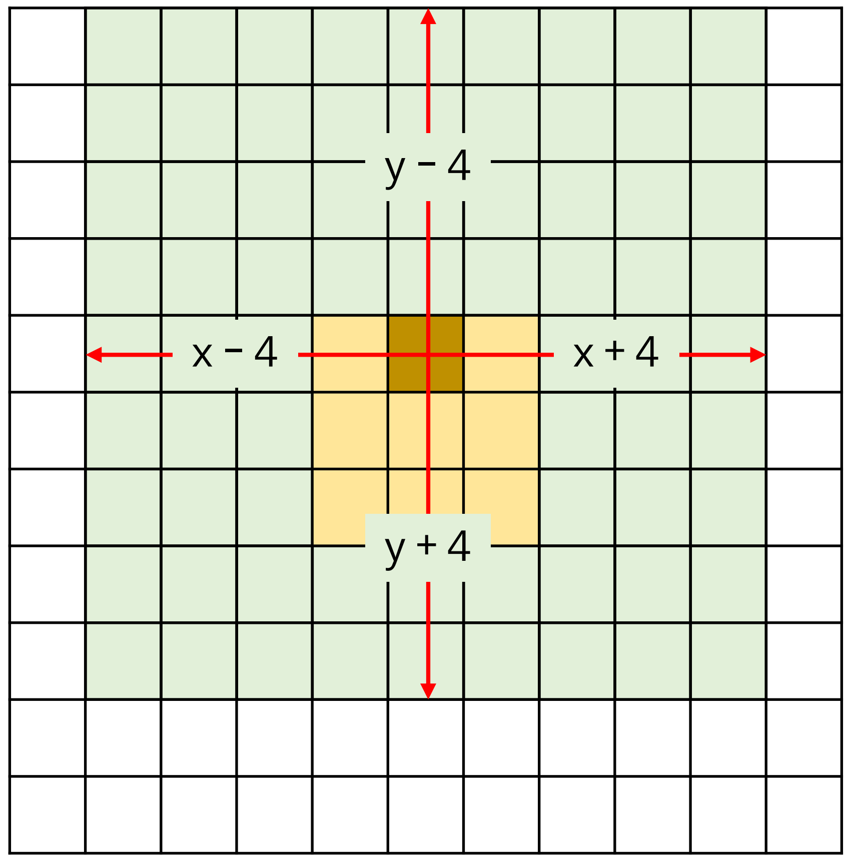
Output 기준

3) Convolution 연산



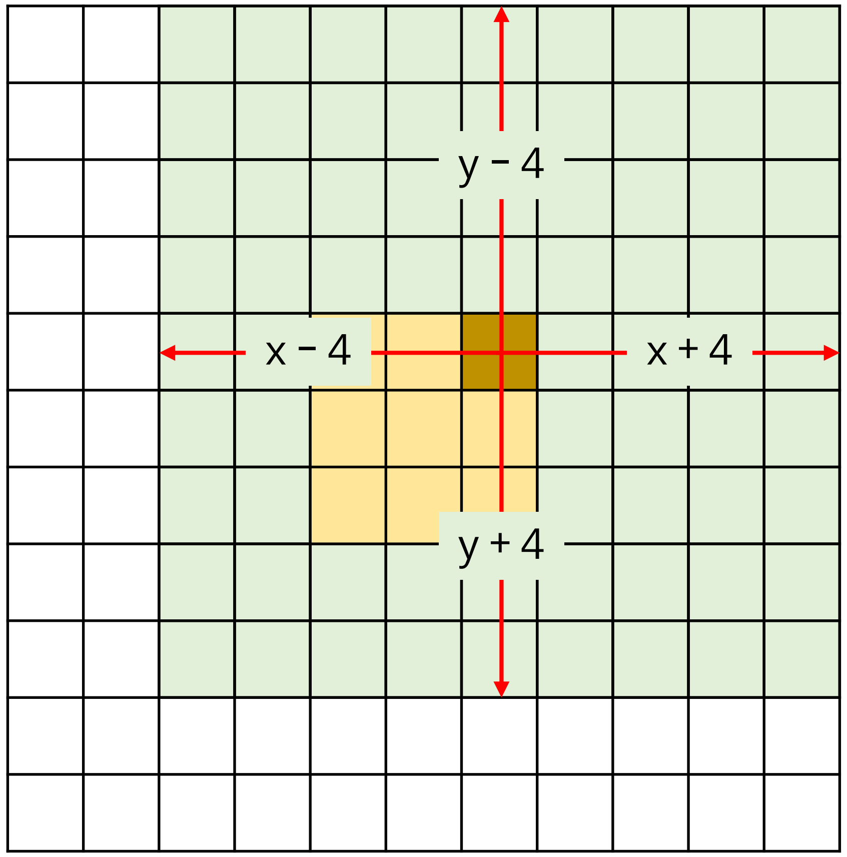
Output 기준

3) Convolution 연산



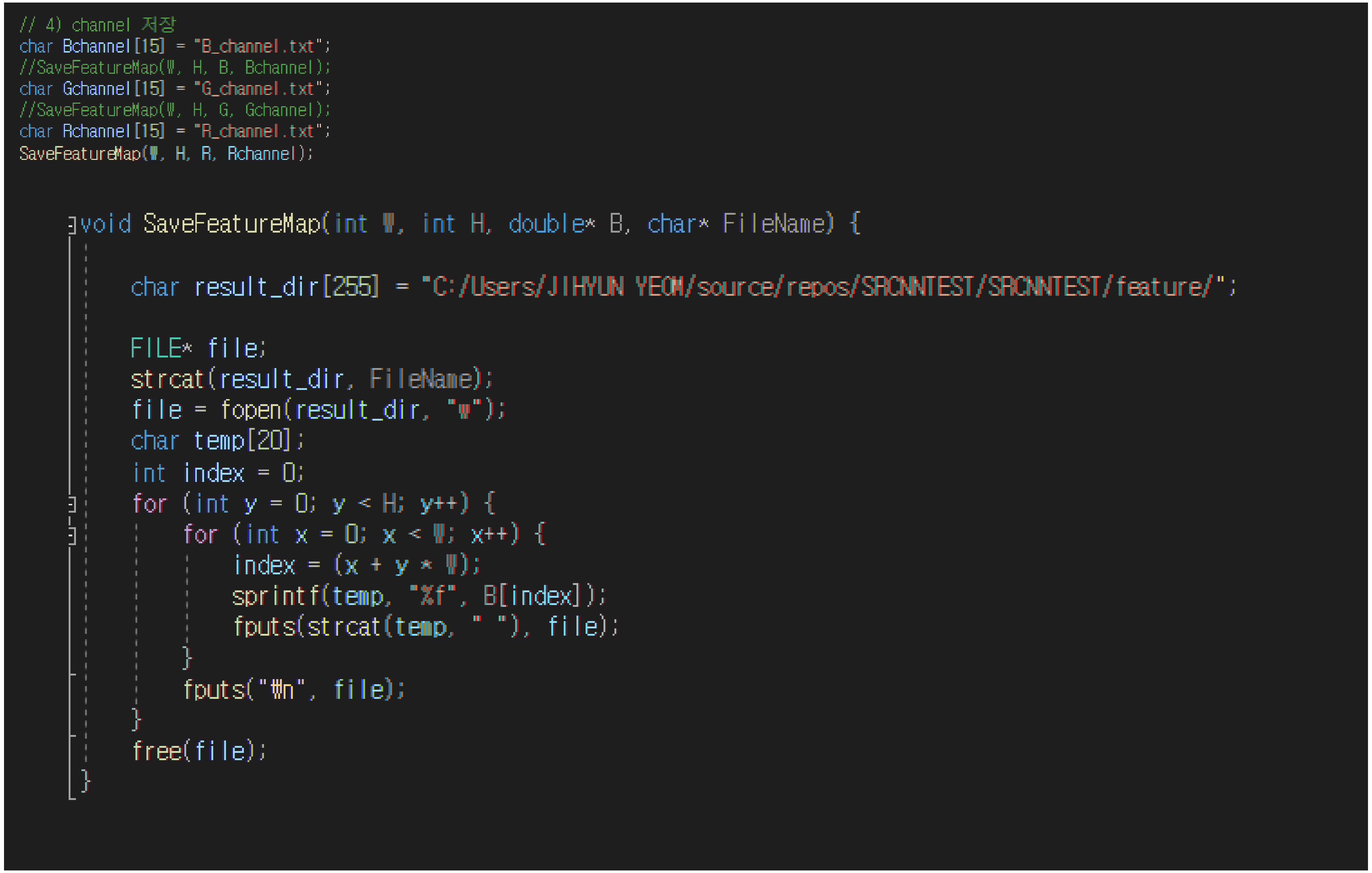
Output 기준

3) Convolution 연산



Output 기준

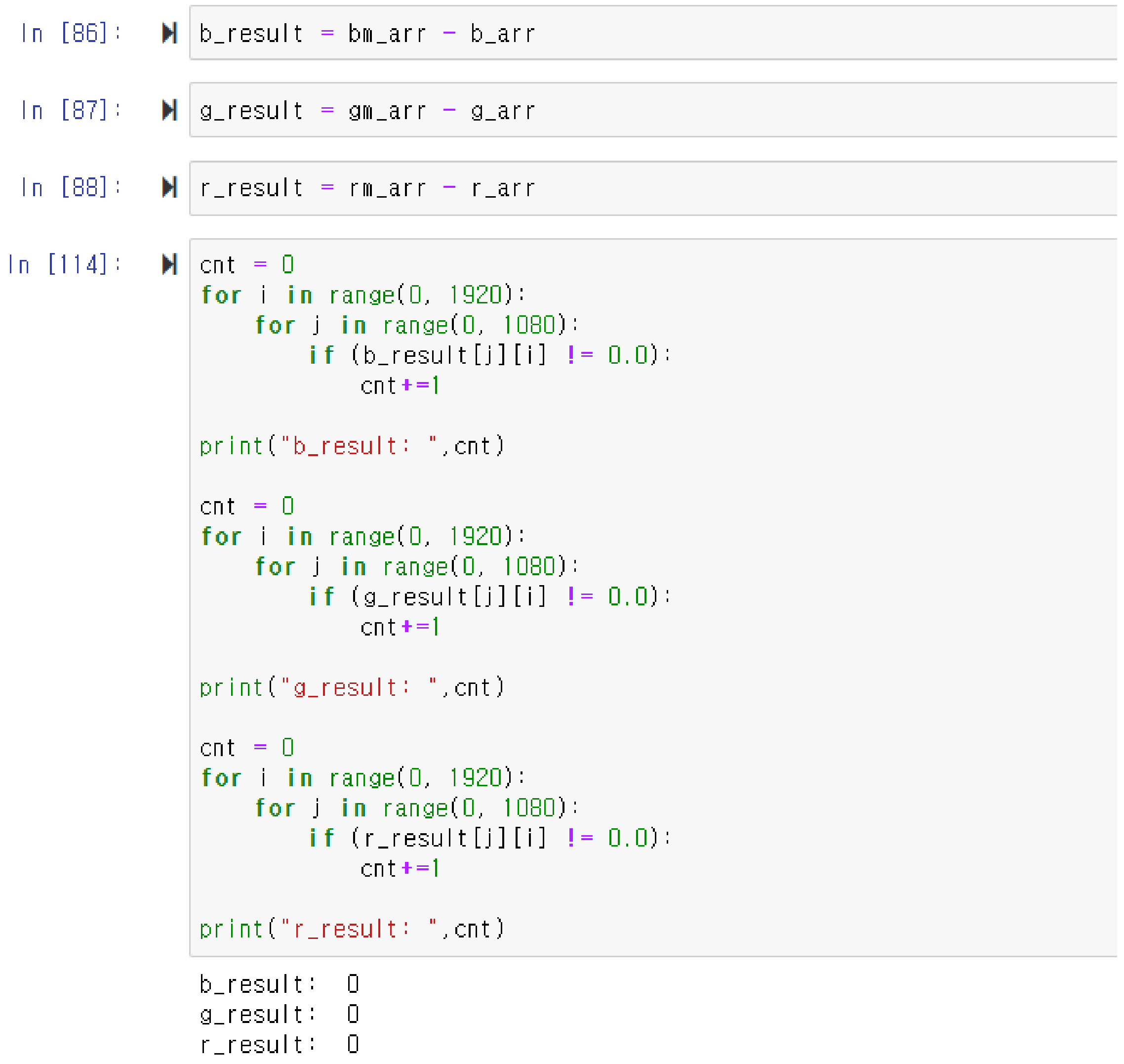
4) Channel 별 feature map 저장



```c
// 4) channel 저장
char Bchannel[15] = "B_channel.txt";
//SaveFeatureMap(W, H, B, Bchannel);
char Gchannel[15] = "G_channel.txt";
//SaveFeatureMap(W, H, G, Gchannel);
char Rchannel[15] = "R_channel.txt";
SaveFeatureMap(W, H, R, Rchannel);


void SaveFeatureMap(int W, int H, double* B, char* FileName) {

    char result_dir[255] = "C:/Users/JIHYUN YEOM/source/repos/SRCNNTEST/SRCNNTEST/feature/";

    FILE* file;
    strcat(result_dir, FileName);
    file = fopen(result_dir, "w");
    char temp[20];
    int index = 0;
    for (int y = 0; y < H; y++) {
        for (int x = 0; x < W; x++) {
            index = (x + y * W);
            sprintf(temp, "%f", B[index]);
            fputs(strcat(temp, " "), file);
        }
        fputs("₩n", file);
    }
    free(file);
}
```

오차 및 소요 시간

* (1928 x 1088) --(9 x 9)--> (1920 x 1080)



```
In [86]:  b_result = bm_arr - b_arr

In [87]:  g_result = gm_arr - g_arr

In [88]:  r_result = rm_arr - r_arr

In [114]: cnt = 0
          for i in range(0, 1920):
              for j in range(0, 1080):
                  if (b_result[j][i] != 0.0):
                      cnt+=1

          print("b_result: ",cnt)

          cnt = 0
          for i in range(0, 1920):
              for j in range(0, 1080):
                  if (g_result[j][i] != 0.0):
                      cnt+=1

          print("g_result: ",cnt)

          cnt = 0
          for i in range(0, 1920):
              for j in range(0, 1080):
                  if (r_result[j][i] != 0.0):
                      cnt+=1

          print("r_result: ",cnt)

          b_result:  0
          g_result:  0
          r_result:  0
```

Input
1928 * 1088
Output
1920 * 1080
time
0.636

Sparse Matrix 크기

염지현

Bilinear & Bicubic

* (1920 x 1080) → (3840 * 2160)

Memory 계산: (int 크기) * 3(x, y, w) * (가중치 수) * W * H * scale * scale

	Bilinear	Bicubic
Weight	4	16
Memory	398,131,200 B = 398.1 MB	1,592,524,800 B = 1.6 GB
Sparse Matrix 생성 시간	0.35	5.626
결과 연산 시간	0.225	0.723