2015-11923 최한결 컴구 PA5 convolution

```
#include "conv.h"
void convolution(const int M, const int N, const int *input, int *output, const in
{
    register int f1 = filter[0][0];
    register int f2 = filter[0][1];
    register int f3 = filter[0][2];
    register int f4 = filter[1][0];
    register int f5 = filter[1][1];
    register int f6 = filter[1][2];
    register int f7 = filter[2][0];       You, 4 days ago • 12/4 loop unrolling 1/
    register int f8 = filter[2][1];
    register int f9 = filter[2][2];
    register int j = 0;
    register int next_row = N;
    register int pre_row_input;
    register int j_input;
    register int next_row_input;
    register int pre_o;
    register int o;
    register int next_o;
    j_input = input[j];
    next_row_input = input[next_row];
    output[j] += j_input * f5 + next_row_input * f8;
    output[j + 1] += j_input * f4 + next_row_input * f7;
    ++j;
    ++next_row;
```

pre_row, j, next_row 세 값이 있고 j값은 프로그램이 실행되는 동안 0부터 1씩 증가해서 M*N -1 즉, input의 길이까지 증가한다. pre_row와 next_row는 각각 j보다 N만큼 작은 값과 N만큼 큰 값이다.

j가 하나씩 증가하면서 j-N, j, j+N 세 위치의 input값을 읽고 각 값에 filter[0][0],filter[1][0],filter[2][0]을 곱한 값을 output[j-1]에 더하고 filter[0][1],filter[1][1],filter[2][1]을 곱한 값과 filter[0][2],filter[1][2],filter[2][2]를 곱한 값을 각각 output[j]와 output[j+1]에 더한다.

이 때 j가 input의 테두리에 있을 경우에는 예외처리를 해준다.
위의 코드에서 j가 0일 때는 input의 꼭지점에 있을 경우 이므로 j-1,next_row-1,pre_row-1,pre_row, pre_row+1 위치의 input이 모두 0값이므로 제외시켜준다.

```
while (j < N - 1)
{
    j_input = input[j];
    next_row_input = input[next_row];
    output[j - 1] += j_input * f6 + next_row_input * f9;
    output[j] += j_input * f5 + next_row_input * f8;
    output[j + 1] += j_input * f4 + next_row_input * f7;
    ++j;
    ++next_row;
}
j_input = input[j];
next_row_input = input[next_row];
output[j - 1] += j_input * f6 + next_row_input * f9;
output[j] += j_input * f5 + next_row_input * f8;
++j;
++next_row;
```

위에 말한 것처럼 j가 input의 첫 줄을 가르킬 때 (j<N-1) pre_row가 가르키는 input의 값이 없으므로 고려하지 않는다. 메모리 접근을 최소화하기 위해 register공간 j_input과 next_row_input에 j위치의 input값과 next_row의 input값을 저장해 둔 후 output[j-1], output[j], output[j+1] 값을 업데이트할 때 사용한다.

j가 N-1일 때 input의 꼭지점을 가르키므로 예외로 처리해 output[j+1]을 계산하지 않는다.

```
register int pre_row = 0;
int row_last = N - 1;
const int last = M * N;
while (next_row < last)
{
    pre_row_input = input[pre_row];
    j_input = input[j];
    next_row_input = input[next_row];
    o = pre_row_input * f2 + j_input * f5 + next_row_input * f8;
    next_o = pre_row_input * f1 + j_input * f4 + next_row_input * f7;
    ++j;
    ++pre_row;
    ++next_row;
    while (pre_row < row_last)
    {
        pre_row_input = input[pre_row];
        j_input = input[j];
        next_row_input = input[next_row];
        pre_o = o;
        o = next_o;
        pre_o += pre_row_input * f3 + j_input * f6 + next_row_input * f9;
        o += pre_row_input * f2 + j_input * f5 + next_row_input * f8;
        next_o = pre_row_input * f1 + j_input * f4 + next_row_input * f7;
        output[j - 1] = pre_o;
        ++j;
        ++pre_row;
        ++next_row;
    }
    pre_row_input = input[pre_row];
    j_input = input[j];
    next_row_input = input[next_row];
    pre_o = o;
    o = next_o;
    pre_o += pre_row_input * f3 + j_input * f6 + next_row_input * f9;
    o += pre_row_input * f2 + j_input * f5 + next_row_input * f8;
    output[j - 1] = pre_o;
    output[j] = o;
    ++j;
    ++pre_row;
    ++next_row;
    row_last += N;
}
```

두 번째 열부터 M-1번째 열까지 첫 행과 마지막 행을 제외한 위치는 j-N-1, j-N, j-N+1, j-1, j, j+1, j+N-1, j+N, j+N+1위치가 모두 고려되어야 하므로 output[j-1], output[j], output[j+1]을 모두 계산해 준다.
이 때 메모리 접근을 최소화 하기 위해 register에 저장된 pre_o와 o, next_o 값을 업데이트 후 pre_o값에 3번의 덧셈이 끝난 후 output[j-1]에 pre_o값을 저장한다.
각 열의 첫 행과 마지막 행에서는 각각 output[j-1]과 output[j+1]값을 제외하고 구해준다.

```
pre_row_input = input[pre_row];
j_input = inp  int pre_input
output[j] += pre_row_input * f2 + j_input * f5;
output[j + 1] += pre_row_input * f1 + j_input * f4;
++j;
++pre_row;
while (j < last - 1)
{
    pre_row_input = input[pre_row];
    j_input = input[j];
    output[j - 1] += pre_row_input * f3 + j_input * f6;
    output[j] += pre_row_input * f2 + j_input * f5;
    output[j + 1] += pre_row_input * f1 + j_input * f4;
    ++j;
    ++pre_row;
}
pre_row_input = input[pre_row];
j_input = input[j];
output[j - 1] += pre_row_input * f3 + j_input * f6;
output[j] += pre_row_input * f2 + j_input * f5;
```

ㄷ

마지막 열에서는 첫 열에서 한 것과 마찬가지로 next_row를 제외하고 연산 후 output을 업데이트 해준다.