<Assignment #4>

201420907

소프트웨어학과

안우일

다음 함수 및 코드를 작성하시오.

(1) printMatrix 함수

(2) C = A*B 행렬 계산 코드

Ⅰ. 문제 파악 및 사용되는 개념

(1) 문제 파악

행렬 연산 프로그램을 작성하시오. 입력은 A matrix, B matrix를 받아들이고, 출력은 C matrix 값을 출력하시오.

- **행렬이란 매트릭스라고도 불리는데, 행렬의 가로줄을 행, 세로줄을 열이라고 한다.**
- **요번 과제에서 작성해야 할 프로그램은 두 행렬 A, B를 받아들이고, 두 행렬의 곱셈 값을 C matrix를 통해 출력하는 프로그램이다.**

  (ex) 행렬 3x1, 1x3의 곱은 1x1의 행렬로 값이 출력될 것이다.

  행렬 3x3, 3x2의 곱은 3x2의 행렬로 값이 출력될 것이다.

(2) 사용되는 개념

- **'printMatrix' 함수는 매개변수로 'int (*Mat)[255]', 'int sy', 'int sx'가 선언되어 있다.**
- **여기서 Mat배열에 있는 것은 Matrix의 각 index 숫자들이고, sy는 열의 개수, sx는 행의 개수이다.**
- **따라서 2중 for문을 이용하여 첫 번째 for문은 열의 개수를 증가시키고, 두 번째 for문은 행의 개수를 증가시키며, 2차원 배열형태로 모니터에 출력하면 입력 받은 Matrix를 출력하는 함수를 작성할 수 있을 것이다.**

- 행렬의 곱은 행렬의 덧셈, 뺄셈과 달리 조금 복잡한 방식을 취하고 있다.

- 아래와 같이 첫 번째 행렬은 열로, 두 번째 행렬은 행으로 1씩 증가하면서 그 index값에 있는 값들을 곱해주어 더해주는 방식이다. (아래 그림 참고)

$$A=\begin{pmatrix} a & b \\ c & d \end{pmatrix},\ B=\begin{pmatrix} e & f \\ g & h \end{pmatrix}\text{일 때,}\quad AB=\begin{pmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{pmatrix}$$

$$\begin{pmatrix} 2 & 4 \\ 1 & 3 \end{pmatrix}\begin{pmatrix} 3 & 1 \\ 2 & 1 \end{pmatrix}=\begin{pmatrix} 2\times3+4\times2 & 2\times1+4\times1 \\ 1\times3+3\times2 & 1\times1+3\times1 \end{pmatrix}=\begin{pmatrix} 14 & 6 \\ 9 & 4 \end{pmatrix}$$

- 두 행렬 A*B가 가능 하려면 A의 행의 개수와 B의 열의 개수가 같아야 한다. (아래그림 참고.)

  만약 같지않다면, 행렬의 곱은 불가능하다.

  (ex) 4x4와 4x3의 행렬은 곱셈 가능. 5x2행렬과 4x4의 행렬은 곱셈 불가.

$$A=\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix},\ B=\begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix}$$

- 두 행렬 A*B의 계산결과를 행렬 C라고 한다면, C의 행과 열의 개수는 A와 B의 행, 열의 개수 중에서 작은 값이다.

(ex) 3x5와 5x2의 행렬의 곱으로 3x2의 행렬이 생긴다.

1x2와 2x1의 행렬의 곱으로 1x1의 행렬이 생긴다.

- 행렬의 곱은 교환법칙이 성립되지 않는다. 따라서 A*B는 B*A와 같지 않다. (아래 그림 참고.)

$$A=\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix},\ B=\begin{pmatrix} 5 & 0 \\ 6 & 7 \end{pmatrix}\ \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$$

$$\Rightarrow A\times B=\begin{pmatrix} 23 & 21 \\ 34 & 28 \end{pmatrix}\text{이고, } B\times A=\begin{pmatrix} 5 & 15 \\ 20 & 46 \end{pmatrix}$$

$$\therefore AB\neq BA$$

Ⅱ. 개념을 C코드로 구현하기

① printMatrix 함수

```
int printMatrix(int (* Mat)[255], int sy, int sx)
{
    // write your code here
    for (int y = 0; y < sy; y++) {
        for (int x = 0; x < sx; x++) {
            printf("%d ", Mat[y][x]);
        }
        printf("₩n");
    }
    printf("₩n");
    return 1;
}
```

- %d를 사용하여 Mat[y][x]를 출력하고 있으며 "%d "로 한 칸 띄워줌으로써

칸마다 정확하게 숫자가 보일 수 있도록 하였다.

- 두번째 반복문이 끝난 후에 printf("₩n")을 넣어주어 열이 분리되게끔 하였다.

② C = A*B 행렬 계산 코드

```
dimC[0] = MIN(dimA[0], dimB[0]);
dimC[1] = MIN(dimA[1], dimB[1]);
end     = MAX(dimA[0], dimB[0]);

if (dimA[1] >= dimB[1])
{
    for (i = 0; i < dimC[0]; i++)
    {
        for (j = 0; j < dimC[1]; j++)
        {
            sum = 0;
            for (k = 0; k < end; k++)
                sum += A[i][k] * B[k][j];
            C[i][j] = sum;
        }

    }
}
```

```
else
{
    for (i = 0; i < dimC[0]; i++)
    {
        for (j = 0; j < dimC[1]; j++)
        {
            sum = 0;
            for (k = 0; k < end; k++)
                sum += B[i][k] * A[k][j];
            C[i][j] = sum;
        }

    }
}
```

- #define MIN(A>=B ? A:B), #define MAX(A>=B ? B:A)를 위에서 정의해 주었다.
- C행렬의 열의 개수 dimC[0]에는 A, B 행렬의 열 중 작은 값을,
- C행렬의 행의 개수 dimC[1]에는 A, B 행렬의 행 중 작은 값을 넣어줬다.
- end에는 dimA[0]과 dimB[0]중 큰 값을 넣어주어 행렬의 곱셈을 할 수 있게 해주었다.
- 행렬의 곱셈을 진행할 때에 행의 개수가 더 많은 행렬이 앞에 진행되어야 하므로 if문을 통해 A의 행의 개수가 더 많을 때와 B의 행의 개수가 더 많을 때를 구분해주었다.
- sum이라는 변수를 통해 행렬의 곱셈 계산 값을 임시적으로 넣어놓고, 행렬의 곱셈을 계산하는 반복문이 끝난 후에 C행렬에 값을 넣어주고 있다.

Ⅲ. 원본 코드 (가로로 정렬)

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#define MAX(A,B) (A >= B ? A : B)
#define MIN(A,B) (A >= B ? B : A)

int A[255][255];
int B[255][255];
int C[255][255];
int dimA[2];
int dimB[2];
int dimC[2];

int sum = 0;
int end = 0;
int i = 0, j = 0, k = 0;
```

```c
int printMatrix(int (* Mat)[255], int sy, int sx)
{
    // write your code here
    for (int y = 0; y < sy; y++) {
        for (int x = 0; x < sx; x++) {
            printf("%d ", Mat[y][x]);
        }
        printf("₩n");
    }
    printf("₩n");
    return 1;
}
```

```c
int readMatrix(int(*Mat)[255], int sy, int sx)
{
    for (int y = 0; y < sy; y++) {
        for (int x = 0; x < sx; x++) {
            scanf("%d", &Mat[y][x]);
        }
    }

    return 1;
}

int main(void)
{
    FILE *fp = freopen("input.txt", "rt", stdin);
    if (fp == NULL) {
        fprintf(stderr, "File not found₩n");
        return 0;
    }
```

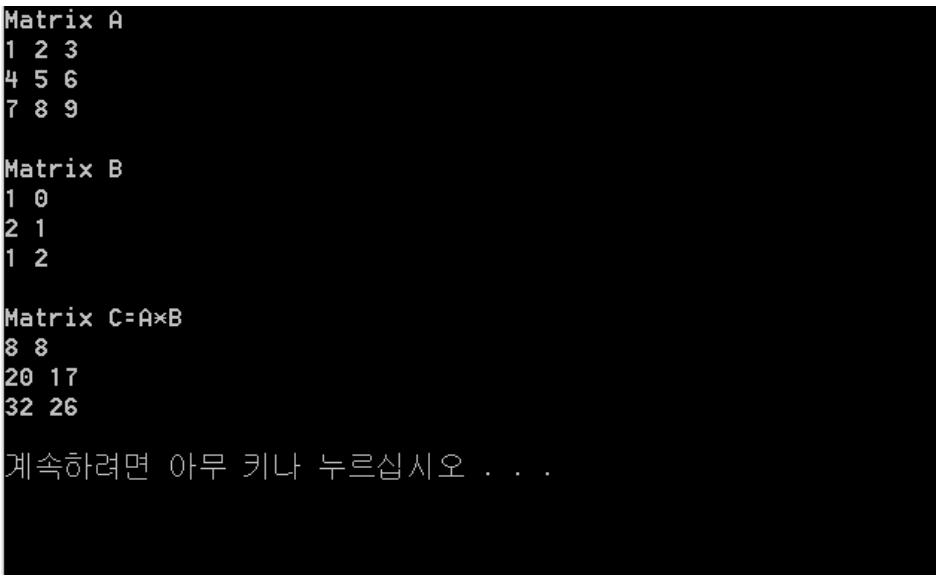
```c
    scanf("%d %d", dimA, dimA+1); // Height x Width
    readMatrix(A, dimA[0], dimA[1]);
    printf("Matrix A₩n");
    printMatrix(A, dimA[0], dimA[1]);

    scanf("%d %d", dimB, dimB + 1);
    readMatrix(B, dimB[0], dimB[1]);
    printf("Matrix B₩n");
    printMatrix(B, dimB[0], dimB[1]);

    printf("Matrix C=A*B₩n");

    // write your code here

    dimC[0] = MIN(dimA[0], dimB[0]);
    dimC[1] = MIN(dimA[1], dimB[1]);
    end     = MAX(dimA[0], dimB[0]);
```

```
if (dimA[1] >= dimB[1])
{
    for (i = 0; i < dimC[0]; i++)
    {
        for (j = 0; j < dimC[1]; j++)
        {
            sum = 0;
            for (k = 0; k < end; k++)
                sum += A[i][k] * B[k][j];
            C[i][j] = sum;
        }

    }
}
```

```
    else
    {
        for (i = 0; i < dimC[0]; i++)
        {
            for (j = 0; j < dimC[1]; j++)
            {
                sum = 0;
                for (k = 0; k < end; k++)
                    sum += B[i][k] * A[k][j];
                C[i][j] = sum;
            }

        }
    }

    printMatrix(C, dimC[0], dimC[1]);
    fclose(fp);

    return 1;
}
```

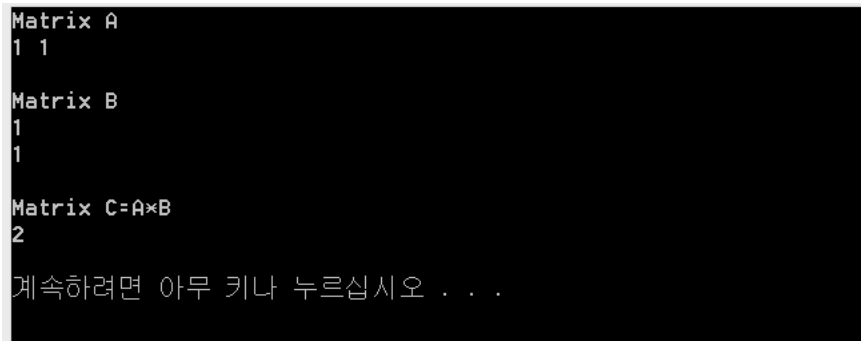
IV. 출력된 화면

1. Assignment #4에 예시로 있는 input



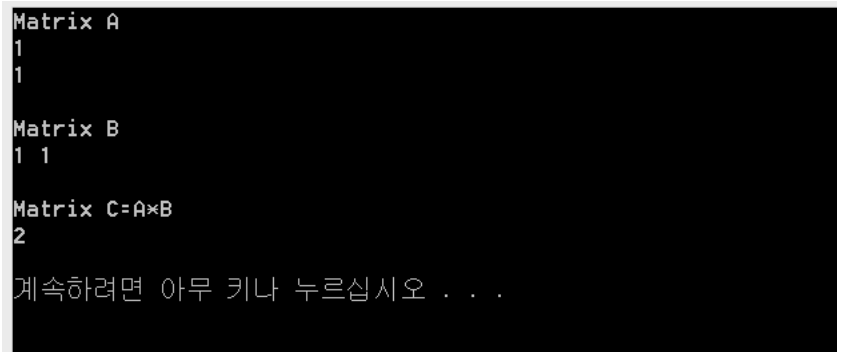
- 3x3과 3x2의 곱으로써 3x2가 나옴을 알 수 있다.

2. 1x2과 2x1 행렬의 곱



- 1x2와 2x1 행렬의 곱으로서 1x1이 나옴을 알 수 있다.

3. 2x1과 1x2 행렬의 곱

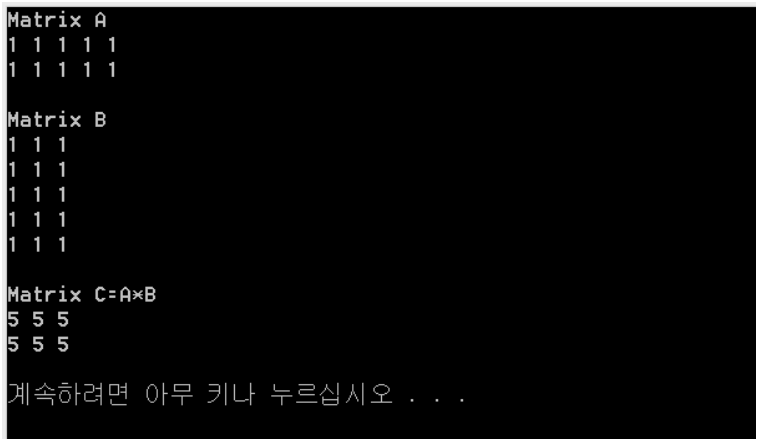


- 2x1과 1x2는 앞의 행렬의 행의 개수가 뒤의 개수보다 작다.

  따라서 else문을 통해 1x2와 2x1의 행렬의 곱으로 바뀌어서 계산되므로

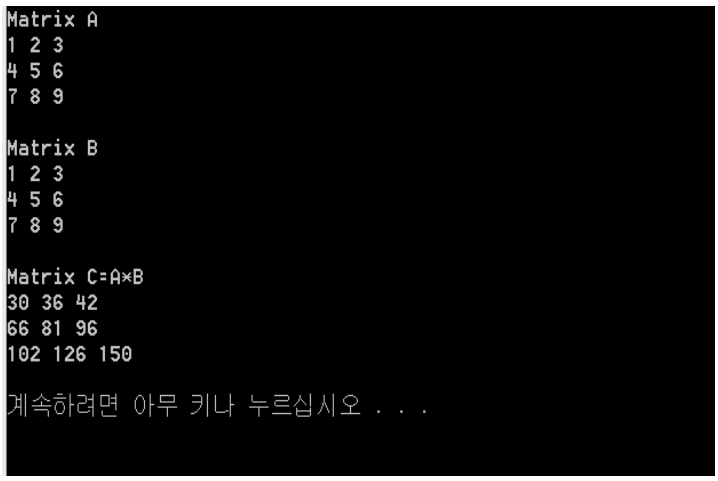
  1x1이 나옴을 알 수 있다.

4. 2x5와 5x3 행렬의 곱



- 2x5 와 5x3 의 행렬의 곱으로 2x3 이 나옴을 알 수 있다.

5. 3x3와 3x3 행렬의 곱



- 3x3 와 3x3 의 행렬의 곱으로 3x3 이 나옴을 알 수 있다.

  행과 열의 개수가 같은 행렬에서도 행렬의 곱셈이 올바르게 출력됨을 알 수 있다.

  출처

  2p http://navercast.naver.com/contents.nhn?rid=22&contents_id=2858

  2p http://cafe.naver.com/differentialintegral/38