자료구조
실습 레포트

제출일	20.04.15	**전 공**	컴퓨터소프트웨어공학과
과 목	자료구조 1	**학 번**	20184612
담당교수	홍 민 교수님	**이 름**	김동민

| 목 차 |

1. 행렬입력과 행렬의 연산 프로그램

행렬입력과 행렬의 연산 프로그램

- **두개의 행렬 데이터를 data.txt파일에서 입력 받아, 두 행렬의 +, -, * 연산을 수행하는 프로그램을 작성 한다. 단, 동적 할당으로 2차원 배열을 생성하여 작성하여야 한다.**

1.1 문제 분석

조건 1 data.txt파일로부터 행렬을 입력받아온다.
조건 2 2중 포인터를 동적 할당하여 2차원 배열을 생성한다.
조건 3 행렬의 합, 차, 곱에 맞는 조건일 때만 수행한다.

- 이 문제는 먼저 data파일로부터 행렬의 행과 열을 입력받는다. 그리고 입력받은 행과 열만큼 2중 포인터를 동적 할당하여 입력받아야 할 행렬의 2차원 배열을 만들어야할 것이다. 그 다음 이중 for문을 이용해 행과 열만큼 for문을 돌려서 data파일로부터 행렬을 입력받는다. 두 번째 행렬도 위와 같은 방법으로 입력받는다.
그리고 첫 번째 행렬과 두 번째 행렬의 행, 열을 각각 비교해서 행렬의 합, 행렬의 차, 행렬의 곱이 가능하다면 수행하고, 만약 합, 차만 수행할 수 있거나 곱만 수행할 수 있는 조건이라면 가능한 것만 수행한다. 3가지를 모두 수행할 수 없는 조건이라면 아무것도 수행하지 않고 프로그램을 종료한다.

행렬의 곱 공식

A 는 m x p, B는 p x n 행렬.
A와 B의 행렬의 곱 AB는 m x n 행렬.

∴행렬의 곱은 앞 행렬의 열== 뒤 행렬의 행
이를 곱한 값의 행은 앞 행렬의 행의 값
열은 뒤 행렬의 열의 값이 된다.
오른쪽 식은 행렬의 곱 알고리즘을 나타낸다.

```
Algorithm product(A, B, C)
begin
        for i=1 to m
                for j=1 to n
                begin
                        c[i][j] =0
                        for k=1 to p
                        c[i][j] = c[i][j] + a[i][k]*b[k][i]
                        endfor
                end
                endfor
        endfor
end
```

1.2 소스 코드

```c
/*
    작성일: 2021.04.
    학번: 20184612
    작성자: 김동민
    프로그램명: 파일 데이터를 입력받아 2중포인터 동적할당하고
                행렬의 덧셈,뺄셈 곱셈하는 프로그램
*/
#include <stdio.h>
#include <stdlib.h>
#pragma warning(disable:4996)

void Mul_Matrix(int **, int **,int **, int, int, int);      //행렬의 곱   int **첫번째 행렬, int ** 두번째행렬,int **결과 행렬, int 
void Plus_Matrix(int **, int **,int **, int, int);          //행렬의 합
void Sub_Matrix(int **, int **,int **, int, int);           //행렬의 차
void Matrix_Print(int **, int , int);                       //행렬 출력
void Matrix_free(int **, int);

int main(){
    FILE *fp;
    int m_row, m_col;                                       //첫번째 행렬의 행, 열
    int **M_matrix;                                         //첫번째 행렬 2중 포인터
    int n_row, n_col;                                       //두번째 행렬의 행, 열
    int **N_matrix;                                         //두번째 행렬 2중 포인터
    int i,j;                                                //for문 돌릴 변수
    int **res_matrix;                                       //결과 행렬
    int g=0;                                                //결과 행렬을 동적할당 하지 않았을 때 사용

    fp=fopen("data.txt", "r");
    if(fp == NULL){
        printf("파일이 열리지 않았습니다.₩n");
        return 0;
    }
    fscanf(fp, "%d%d", &m_row, &m_col);                 //첫번째 행렬의 행(m_row)과 열(m_col)값을 입력받음
    M_matrix = (int **)malloc(sizeof(int*)*m_row);      //첫번째 행렬 동적할당
    for(i=0; i<m_row; i++){
        M_matrix[i] = (int *)malloc(sizeof(int)*m_col);
    }

    for(i=0; i<m_row; i++){                             //이중 for문을 통과 열값만큼 돌려 첫번째 행렬의 값을 data로부터 입력
        for(j=0; j<m_col; j++){
            fscanf(fp,"%d",&M_matrix[i][j]);
        }
    }

    fscanf(fp, "%d%d", &n_row, &n_col);                 //두번째 행렬의 행(n_row)과 열(n_col)값을 입력받음
    N_matrix = (int **)malloc(sizeof(int*)*n_row);      //두번째 행렬 동적할당
    for(i=0; i<n_row; i++){
        N_matrix[i] = (int *)malloc(sizeof(int)*n_col);
    }
    for(i=0; i<n_row; i++){                             //이중 for문을 통과 열값만큼 돌려 두번째 행렬의 값을 data로부터 입력
        for(j=0; j<n_col; j++){
            fscanf(fp,"%d",&N_matrix[i][j]);
        }
    }
    fclose(fp);             //파일 닫기
```

```c
    printf("행렬1>>%d x %d₩n",m_row, m_col);                        //첫번째 행렬 출력
    Matrix_Print(M_matrix, m_row, m_col);

    printf("행렬2>>%d x %d₩n",n_row, n_col);                        //두번째 행렬 출력
    Matrix_Print(N_matrix, n_row, n_col);

    if((m_row==n_row)&&(m_col==n_col)||(m_col==n_row)){              //결과행렬은 합, 차 곱이 가능할때만 동적할당 하고 가능하지 않을때는 동
        res_matrix = (int **)malloc(sizeof(int*)*m_row);                    //결과 행렬의 행값은 1번행렬의 행값, 열값은 2번행렬의 열값으로
            for(i=0; i<m_row; i++){
                res_matrix[i] = (int *)malloc(sizeof(int)*n_col);
        }
        g=1;
    }
    if((m_row==n_row)&&(m_col==n_col)&&(m_col==n_row)){                  //행과 열의 수가 n으로 같은 정방행렬일때 (n*n)(n*n)  = 모두 계산 가
        Plus_Matrix(M_matrix,N_matrix,res_matrix, m_row, n_col);        //행렬의 합계산 -> 매개변수 1.1번행렬, 2.2번행렬, 3.결과행렬, 4.1번
        Sub_Matrix(M_matrix,N_matrix,res_matrix, m_row, n_col);         //행렬의 차계산
        Mul_Matrix(M_matrix, N_matrix,res_matrix, m_row, n_col, n_col);     //행렬의 곱계산
    }


    else if((m_row==n_row)&&(m_col==n_col)){                            //1번행렬 행과 2번행렬의 행의 값이 같고, 1번행렬 열과 2번행렬의 열

        Plus_Matrix(M_matrix,N_matrix,res_matrix, m_row, n_col);        //행렬의 합계산 -> 매개변수 1.1번행렬, 2.2번행렬, 3.결과행렬, 4.1번
        Sub_Matrix(M_matrix,N_matrix,res_matrix, m_row, n_col);         //행렬의 차계산
        printf("행렬의 곱은 계산 불가₩n");
    }
    else if(m_col==n_row){                                              //1번행렬 열과 2번행렬 행의 값이 같을 때 (m*n)(n*p) -> 곱셈가능(합,

        printf("행렬의 합과 차은 계산불가₩n");                          //m!=n, n!=p이므로 합,차 계산불가
        Mul_Matrix(M_matrix, N_matrix,res_matrix, m_row, n_col, n_col);     //행렬의 곱계산 -> 매개변수 1.1번행렬, 2.2번행렬, 3.결과행렬, 4.1번

    }
    else{
        printf("행렬의 합,차,곱 모두 계산불가₩n");                      //모든 조건을 만족하지 못하면 합,차, 곱 모두 계산불가
        g=0;                                                            //g의 값을 1로설정
    }

    Matrix_free(M_matrix, m_row);                                       //1번행렬 동적할당 해제
    Matrix_free(N_matrix, n_row);                                       //2번행렬 동적할당 해제
    if(g==1) Matrix_free(res_matrix, m_row);                            //g의 값을 결과값 동적할당에서 1으로 설정, 만약 합,차,곱이 모두 불
    return 0;
}
```

함수 구현부

```
void Plus_Matrix(int **a, int **b,int **res, int row, int col){                //이중포인터가 전달인자로 넘어오므로 매개변수로 이중포인터변수를 선언함
    int i,j;
        for(i=0; i<row; i++){                                                  //입력받은 행과 열만큼 for문으로 합 계산
            for(j=0; j<col; j++){
                res[i][j] = a[i][j] + b[i][j];
            }
        }
        printf("행렬의 합>>>%d x %d₩n",row, col);
        Matrix_Print(res, row, col);                                           //행렬값 출력
}
void Sub_Matrix(int **a, int **b,int **res, int row, int col){
    int i,j;
    printf("행렬의 차>>>%d x %d₩n",row, col);
    for(i=0; i<row; i++){                                                  //입력받은 행과 열만큼 for문으로 차 계산
        for(j=0; j<col; j++){
            res[i][j] = a[i][j] - b[i][j];
        }
    }
    Matrix_Print(res, row, col);
}
void Mul_Matrix(int **a, int **b,int **res, int row, int col, int m_col){
    int i,j;
    int k;
    printf("행렬의 곱>>>%d x %d₩n",row, col);
    for(i=0; i<row; i++){
        for(j=0; j<col; j++){
            res[i][j] = 0;                                                 //결과 행렬 원자값을 0으로 설정
            for(k=0; k<m_col; k++){
                res[i][j] += a[i][k]*b[k][j];                              //k를 1번행렬의 열값(2번행렬의 행값)만큼 for문을 돌리며 (1번행렬의 열*2번행
            }
        }
    }
    Matrix_Print(res, row, col);
}
void Matrix_Print(int **matrix, int row, int col){
    int i, j;
    for(i=0; i<row; i++){
        for(j=0; j<col; j++){
            printf("%3d  ", matrix[i][j]);                                 //행렬값 출력
        }
        printf("₩n");
    }
    printf("₩n");
}
void Matrix_free(int **matrix, int row){
    int i;
    for(i=0; i<row; i++) free(matrix[i]);
    free(matrix);
}
```

1.3 소스코드 분석

```
/*
    작성일: 2021.04.14
    학번: 20184612
    작성자: 김동민
    프로그램명: 파일 데이터를 입력받아 2중포인터 동적할당하고
                행렬의 연산수행을 출력하는 프로그램
*/
#include <stdio.h>
#include <stdlib.h>
#pragma warning(disable:4996)

void Mul_Matrix(int **, int **,int **, int, int, int);
void Plus_Matrix(int **, int **,int **, int, int);
void Sub_Matrix(int **, int **,int **, int, int);
void Matrix_Print(int **, int , int);
void Matrix_free(int **, int);

int main(){
    FILE *fp;
    int m_row, m_col;
    int **M_matrix;
    int n_row, n_col;
    int **N_matrix;
    int i,j;
    int **res_matrix;
    int g=0;
```

1. 소스코드를 작성한 날짜, 이름, 프로그램명을 작성한다.
2. 필요한 헤더를 포함한다.
3. 필요한 함수들을 선언한다.

plus, sub, mul은 각각 행렬의 합, 차, 곱을 수행하는 함수이고
행렬을 출력하는 print함수와 동적 메모리 할당을 해제하는 free함수를 선언하였다.

4. main함수 안에 필요한 변수들을 선언한다.

- fp는 FILE구조체를 가리키는 포인터 변수
- M_matrix와 m_row, m_col은 각각 1번 행렬을 나타낼 이중포인터와 행, 열을 나타내고 N_matrix, n_row, n_col또한 2번 행렬에 대하여 나타낸다.
- i, j는 이중for문을 돌리기 위한 변수이다.
- res_matrix는 결과 값을 나타내기 위한 이중포인터이다.
- g는 마지막에 만약 결과행렬을 동적할당 하지 않았을 시에 사용한다.

```
fp=fopen("data.txt", "r");
if(fp == NULL){
    printf("파일이 열리지 않았습니다.₩n");
    return 0;
}
fscanf(fp, "%d%d", &m_row, &m_col);
M_matrix = (int **)malloc(sizeof(int*)*m_row);
for(i=0; i<m_row; i++){
    M_matrix[i] = (int *)malloc(sizeof(int)*m_col);
}

for(i=0; i<m_row; i++){
    for(j=0; j<m_col; j++){
        fscanf(fp,"%d",&M_matrix[i][j]);
    }
}

fscanf(fp, "%d%d", &n_row, &n_col);
N_matrix = (int **)malloc(sizeof(int*)*n_row);
for(i=0; i<n_row; i++){
    N_matrix[i] = (int *)malloc(sizeof(int)*n_col);
}
for(i=0; i<n_row; i++){
    for(j=0; j<n_col; j++){
        fscanf(fp,"%d",&N_matrix[i][j]);
    }
}
fclose(fp);             //파일 닫기
```

5. data파일을 읽기 전용으로 개방한다.

6. data파일로부터 행값과 열값을 입력받고 동적할당한 후 값을 저장 한다.

- 먼저 data로부터 m_row(m행렬의 행), m_col(m행렬의 열)에 행, 열값을 입력받고 그만큼 M_matrix를 동적 할당하여 2차원 배열을 생성한 후, 이중for문으로 data로부터 행렬을 입력받는다.

- data파일로부터 M_matrix의 행렬 값을 모두 입력받았다면 그 다음 2개의 값은 N_matrix의 행, 열값을 나타낸다. 그러면 위에서 했던 것과 같이 n_row(n행렬의 행), n_col(n행렬의 열)에 행, 열값을 입력받고 N_matrix를 동적 할당하고 이중for문으로 data로부터 행렬을 입력받는다.

7. 파일 입력이 모두 끝났으면 사용이 모두 끝났으므로 fclose를 이용해 파일을 닫는다.

```
printf("행렬1>>%d x %d₩n",m_row, m_col);
Matrix_Print(M_matrix, m_row, m_col);

printf("행렬2>>%d x %d₩n",n_row, n_col);
Matrix_Print(N_matrix, n_row, n_col);

if(((m_row==n_row)&&(m_col==n_col))||(m_col==n_row)){
    res_matrix = (int **)malloc(sizeof(int*)*m_row);
        for(i=0; i<m_row; i++){
            res_matrix[i] = (int *)malloc(sizeof(int)*n_col);
    }
    g=1;
}
```

8. 행렬이 정상적으로 입력받았는지 print함수를 이용해 확인한다.
print함수의 매개변수로는 행렬과, 그 행렬의 행, 열값을 매개변수로 전달한다.

9. 행렬의 출력이 모두 끝나면 결과 값 행렬을 동적 할당하는데, 만약 행렬의 덧셈, 뺄셈, 곱셈의 조건에 모두 부합하지 않다면 동적할당을 아예 하지 않는다. 만약 동적 할당을 했는데 조건에 부합하지 않는다면 결과 행렬은 아무 쓸데가 없어서 메모리가 낭비되기 때문이다.
(m_row==n_row)&&(m_col==n_col)은 행렬의 합, 차를 계산할 때의 조건이고 (m_col==n_row)은 행렬의 곱을 할 때의 조건이다.

10. 동적할당은 행렬M의 행 x 행렬N의 열로 동적 할당한다.
만약 덧셈의 경우 (m*n)+(m*n)=(m*n)으로 성립하고,
곱셈의 경우(m*n)*(n*p)=(m*p)로 성립하므로
결과행렬은 행렬M의 행 x 행렬N의 열임을 알 수 있다.

11. g의 값을 1로 설정해 나중에 동적메모리할당 해제 시 결과 행렬이 해제될 수 있도록 한다.

```
if(((m_row==n_row)&&(m_col==n_col))&&(m_col==n_row)){
    Plus_Matrix(M_matrix,N_matrix,res_matrix, m_row, n_col);
    Sub_Matrix(M_matrix,N_matrix,res_matrix, m_row, n_col);
    Mul_Matrix(M_matrix, N_matrix,res_matrix, m_row, n_col, m_col);
}


else if((m_row==n_row)&&(m_col==n_col)){

    Plus_Matrix(M_matrix,N_matrix,res_matrix, m_row, n_col);
    Sub_Matrix(M_matrix,N_matrix,res_matrix, m_row, n_col);
    printf("행렬의 곱은 계산 불가₩n");
}
else if(m_col==n_row){

    printf("행렬의 합과 차는 계산불가₩n");
    Mul_Matrix(M_matrix, N_matrix,res_matrix, m_row, n_col, m_col);

}
else{
    printf("행렬의 합,차,곱 모두 계산불가₩n");
    g=0;
}
```

12. ((m_row==n_row)&&(m_col==n_col))&&(m_col==n_row)의 경우 모든 행렬의 행과 열이 동일한 (m*m)행렬, 따라서 정방행렬임을 알 수 있다. 정방행렬은 행렬의 합, 차, 곱 모두 가능하다. 따라서 덧셈함수, 뺄셈함수, 곱셈함수를 모두 호출한다.

13. (m_row==n_row)&&(m_col==n_col)은 (m*n)+(m*n)의 행렬로 행렬의 덧셈과 뺄셈만 가능하고 M 행렬의 열과 N 행렬의 행이 다르기 때문에 곱은 불가능하다 . 따라서 덧셈함수, 뺄셈함수만 호출하고 행렬의 곱은 계산 불가라는 메시지를 보인다.

14. (m_col==n_row)은 (m*n)*(n*p)의 행렬로 M 행렬의 열과 N 행렬의 행이 같기 때문에 행렬의 곱만 가능하다. 이 경우 각 행렬의 행과 열이 다르기 때문에 덧셈과 뺄셈은 불가능하다. 따라서 행렬의 곱 함수만 호출하고 합과 차는 계산이 불가 한다는 메시지를 보인다.

15. 행렬의 합, 차, 곱은 모두 매개변수로 M행렬, N행렬, 결과 행렬, M의 행값, N의 열값을 공통적으로 전달한다. 다만 곱의 경우, M행렬의 열값 또는 N행렬의 행값이 필요하므로 곱에서는 M행렬의 열값도 전달하였다.

16. 만약 이 조건에 모두 부합하지 않는다면 행렬의 합, 차, 곱이 모두 계산 불가라는 메시지를 내보내고 g의 값을 0으로 설정한다.

```
    Matrix_free(M_matrix, m_row);
    Matrix_free(N_matrix, n_row);
    if(g==1) Matrix_free(res_matrix, m_row);
    return 0;
}
```

16. 동적 할당한 포인터를 모두 메모리 해제시킨다.
만약 결과 행렬의 경우, 덧셈, 뺄셈, 곱셈의 조건에 하나라도 부합했다면 동적할당을 하고 g의 값을 1로 설정했기 때문에 메모리 할당을 해제한다.
만약 행렬의 합, 차, 곱 모두 계산 불가라면 g의 값이 0이므로 메모리 해제를 하지 않는다. 동적할당을 하지 않았는데 메모리 해제를 하게 되면 오류가 발생한다.

17. M행렬과 결과 행렬의 경우 행이 m의 행으로 동적할당 되었으므로 m_row를 전달하고, N행렬의 경우는 n의 행으로 동적할당 되었으므로 n_row를 전달한다.

```
void Plus_Matrix(int **a, int **b,int **res, int row, int col){
    int i,j;
        for(i=0; i<row; i++){
            for(j=0; j<col; j++){
                res[i][j] = a[i][j] + b[i][j];
            }
        }
        printf("행렬의 합>>%d x %d₩n",row, col);
        Matrix_Print(res, row, col);
}
void Sub_Matrix(int **a, int **b,int **res, int row, int col){
    int i,j;
    printf("행렬의 차>>%d x %d₩n",row, col);
    for(i=0; i<row; i++){
        for(j=0; j<col; j++){
            res[i][j] = a[i][j] - b[i][j];
        }
    }
    Matrix_Print(res, row, col);
}
```

18. 행렬의 덧셈과 뺄셈을 수행하는 함수이다. 행렬의 덧셈 뺄셈은 같은 조건에서 수행된다. 이중for문을 이용해 결과 행렬 값에 M행렬에서 N행렬을 더하거나 뺀 값을 저장한다.

19. print함수를 호출하여 결과 행렬 값을 출력한다. print함수의 매개변수로는 M행렬과 N행렬은 필요 없기 때문에 결과행렬과 결과행렬의 행, 열값만을 전달한다.

```
void Mul_Matrix(int **a, int **b,int **res, int row, int col, int m_col){
    int i,j;
    int k;
    printf("행렬의 곱>>%d x %d₩n",row, col);
    for(i=0; i<row; i++){
        for(j=0; j<col; j++){
            res[i][j] = 0;
            for(k=0; k<m_col; k++){
                res[i][j] += a[i][k]*b[k][j];
            }
        }
    }
    Matrix_Print(res, row, col);
}
```

20. 행렬의 곱을 수행하는 함수이다.
시작 for문으로 행을 순환하고 중간 for문으로 열을 순환하고, 마지막 for문으로 연산에서의 인덱스 값을 순환한다.
결과 행렬의 값을 0으로 초기화 한 다음 M행렬의 열만큼 for문을 돌려 결과 값에 (a[i][k] * b[k][i])의 값을 더하며 저장한다.

21. print함수로 곱셈 결과 행렬 값을 출력한다.

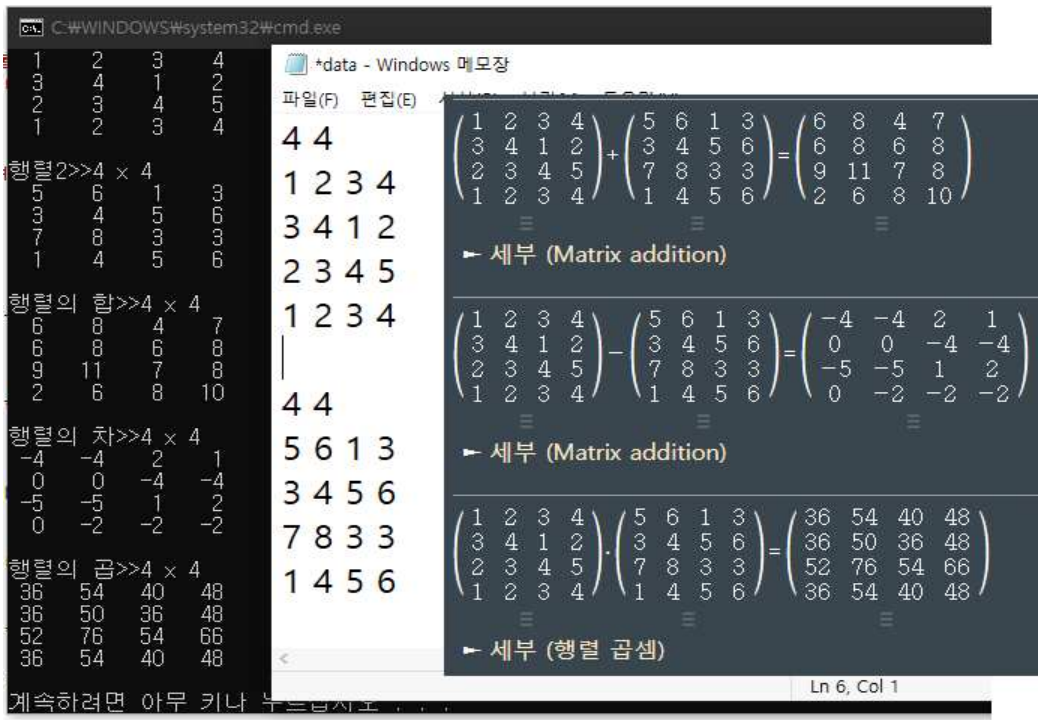
```
void Matrix_Print(int **matrix, int row, int col){
    int i, j;
    for(i=0; i<row; i++){
        for(j=0; j<col; j++){
            printf("%3d  ", matrix[i][j]);
        }
        printf("₩n");
    }
    printf("₩n");
}
void Matrix_free(int **matrix, int row){
    int i;
    for(i=0; i<row; i++) free(matrix[i]);
    free(matrix);
}
```

22. 행렬을 출력하는 print함수이다. 매개변수로는 출력할 행렬의 포인터와 행, 열값을 입력받는다. 이중 for문으로 행과 열만큼 순환하고 행렬 값을 출력하게 된다.

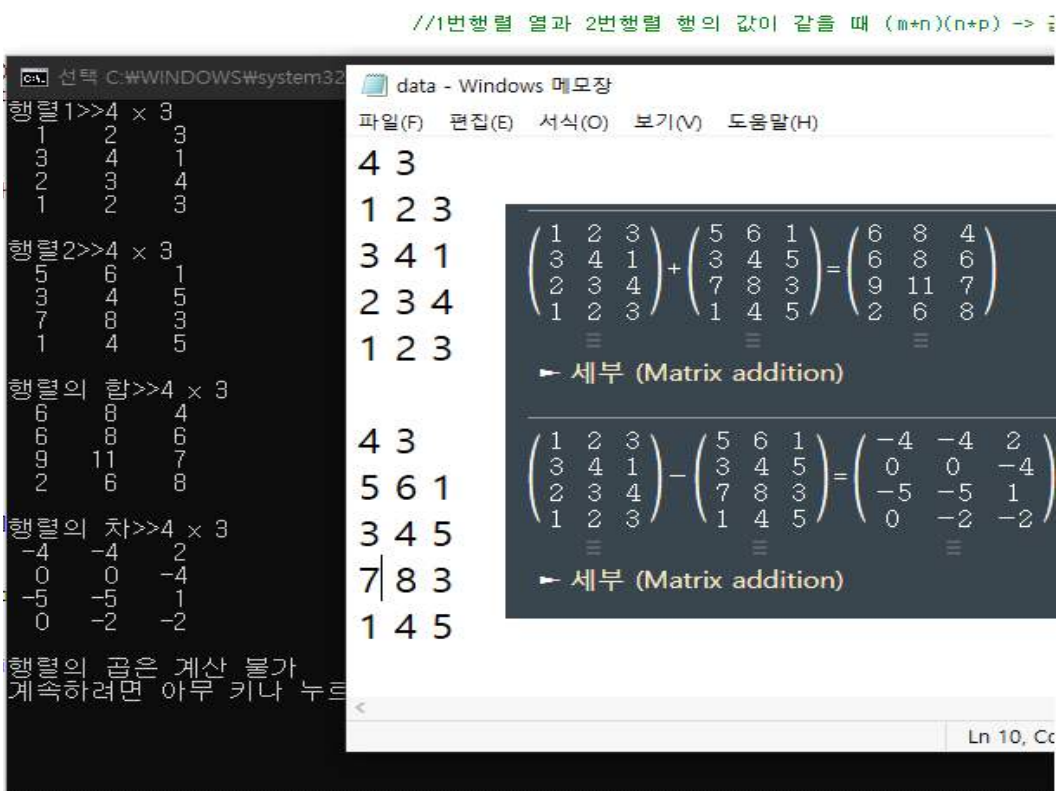
23. 동적 메모리 할당을 해제하는 함수이다. 입력받은 행만큼 for문을 돌며 matrix[i]의 메모리를 해제하고 마지막으로 matrix의 메모리를 해제한다.

1.4 실행 창

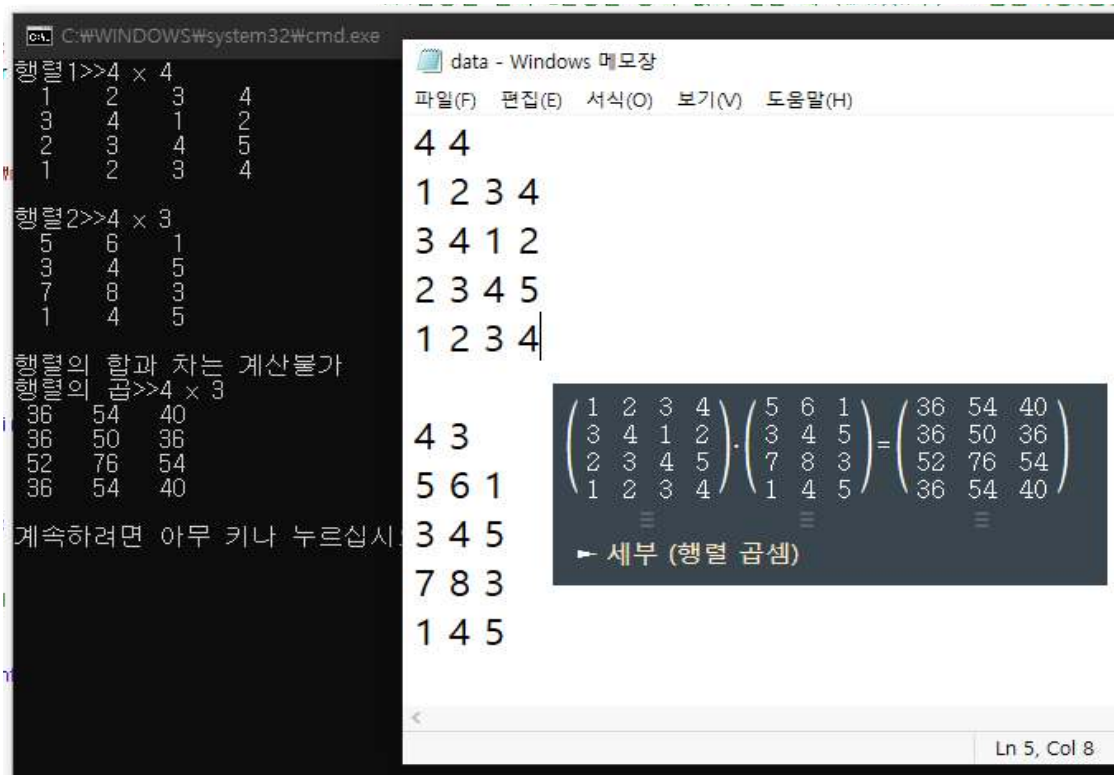
(m*m)형의 정방행렬인 경우(오른쪽은 행렬계산기를 통한 실제 값)



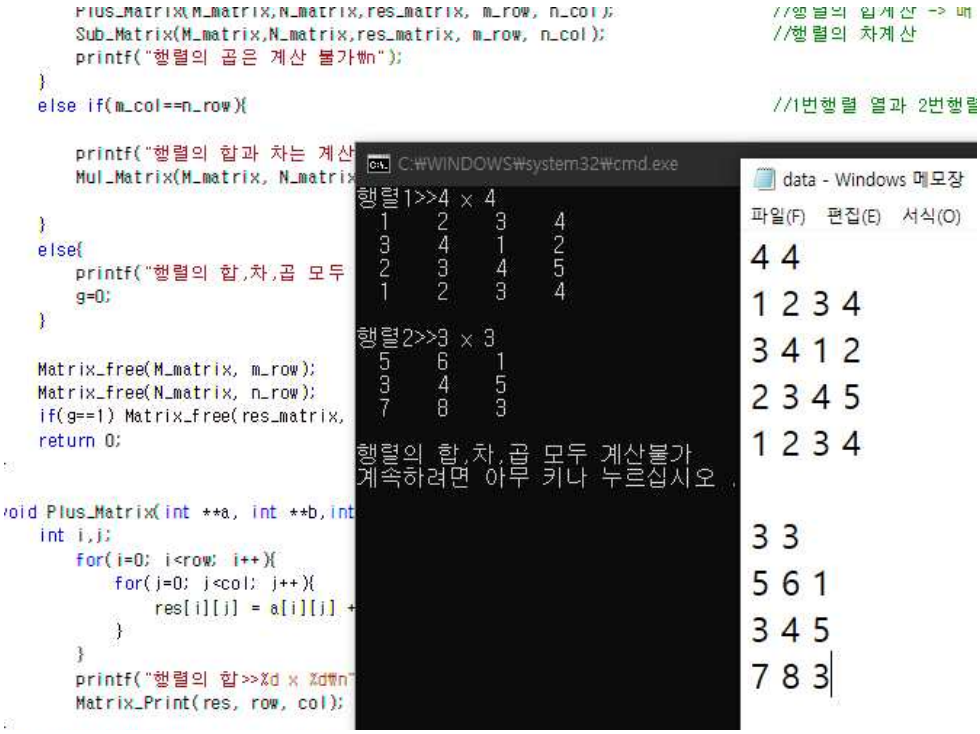
(m*n)의 행렬인 경우(오른쪽은 행렬계산기를 통한 실제 값)



(m*n)*(n*p)의 행렬인 경우(오른쪽은 행렬계산기를 통한 실제 값)



행렬의 행과 열이 모두 달라 계산이 불가능한 경우



1.5 느낀 점

처음에 이 과제를 들었을 때 다행히 이산수학 수업을 들어서 행렬에 대해서는 머릿속에 공식은 있었지만 이걸 어떻게 코드로 풀어낼까 하는 생각이 먼저 들었습니다. 특히 행렬의 합과 차보다는 행렬의 곱을 어떻게 풀어내야 할지 생각을 많이 했습니다. 그래서 행렬의 곱이 진행되는 과정을 다시 한 번 읽어보고 행렬의 곱은 이중for문이 아니라 삼중for문으로 해야겠다는 것을 깨달았습니다.

3년 전에 군대에 가기 전 이차원 배열 동적할당에 대해 조금 공부한 적이 있었습니다. 그런데 이번에 이차원 배열 동적할당을 하려니 잘 생각이 나지 않는 부분들이 많았습니다. 교수님께서 이 부분에 대해 자세히 설명해주셔서 동적할당을 수행하는 데에 많은 도움이 되었던 것 같습니다.

처음 이 코드를 짤 때 저는 거의 모든 코드를 main안에서 해결하려고 했었고 그 결과 코드가 엄청 길어졌던 적이 있었습니다. 그런데 모든 코드를 작성하고 다시 한 번 천천히 봐보니 제가 생각했던 것 보다 중복된 코드들이 많이 있었습니다. 그래서 이 중복된 코드들은 함수로 작성해야겠다고 생각했고, 이 부분들을 모두 함수로 작성했습니다. 이번 레포트를 하며 코드의 길이를 최대한 짧고 간결하게 하려고 노력했고, 효율적으로 작성하려고 했습니다. 그 결과 제가 처음 작성했던 코드보다 훨씬 코드의 길이가 줄었고 가독성이 좋아졌습니다. 이번 과제로 함수와 동적할당에 대해 더 자세하게 알 수 있었던 계기가 되었던 것 같습니다.

이상 “행렬입력과 행렬의 연산 프로그램” 레포트를 마치겠습니다.

- 감사합니다. -