객체지향형 프로그래밍과 자료구조 Lab. 3

3.1 Class Mtrx

1) class Mtrx 정의:

```
/**   Class_Mtrx.h */
#ifndef MTRX_H
#define MTRX_H
#include <iostream>
#include <fstream>
using namespace std;
#define MAX_SIZE 100

class Mtrx {
public:
    Mtrx(int num_row, int num_col);
    Mtrx(double *dA, int num_data, int num_row, int num_col);
    Mtrx(istream& fin);
    ~Mtrx(); // destructor
    int getN_row() { return n_row; }
    int getN_col() { return n_col; }
    void fprintMtrx(ostream& fout);
    void setName(string nm) { name = nm;};
    string getName() { return name;};
    Mtrx add(const Mtrx&);
    Mtrx sub(const Mtrx&);
    Mtrx multiply(const Mtrx&);
private:
    string name;
    int n_row;
    int n_col;
    double **dM;
};
#endif
```

2) 클래스 멤버 함수 구현

- class Mtrx 선언은 "Class_Mtrx.h" header 파일에 class Mtrx 의 멤버함수들은 Class_Mtrx.cpp 파일에 구현할 것.
- Mtrx(istream& fin) 생성자는 주어진 입력 데이터 파일 객체로부터 행렬의 크기 (행의 크기, 열의 크기)와 데이터를 입력 받아, 주어진 행렬의 크기에 맞는 2 차원 double 형 배열을 동적으로 생성.
- add() 멤버 함수는 행렬의 덧셈을 계산. 기준이 되는 객체의 행렬과 인수로 주어진 행렬의 합을 계산하여 반환.
- sub() 멤버 함수는 행렬의 뺄셈을 계산. 기준이 되는 객체의 행렬과 인수로 주어진 행렬의 차을 계산하여 반환.
- multiply() 멤버 함수는 행렬의 곱셈을 계산. 기준이 되는 객체의 행렬과 인수로 주어진 행렬의 곱을 계산하여 반환.
- fprintMtrx(ostream& fout) 함수는 지정된 출력 객체로 class Mtrx 를 출력. 행렬의 출력에서는 행렬의 이름을 출력하고, 확장완성형 코드를 사용하여 행렬 모습인 대괄호 ([,])가 표시되도록 할 것.

3.2 main() 함수

1) 파일 입력, class Mtrx 객체 생성

- 입력파일 ("Matrix_data.txt")로부터 행렬의 크기 (size_m, size_n)와 행렬의 원소 데이터를 파일로부터 입력 받아 동적으로 double 형 2 차원 배열 mtrxA, mtrxB 및 mtrxC 를 생성하고, 행렬의 원소 데이터를 초기화.
- 입력파일 ("Matrix_data.txt")에는 3 개의 행렬에 대한 행렬의 크기 (size_m, size_n)와 해당 행렬의 원소들이 저장되어 있도록 구성할 것.
- mtrxA 와 mtrxB 는 5 x 7 의 크기를 가지며, mtrxC 는 7 x 5 의 크기를 가짐

2) class Mtrx 멤버 함수 실행 및 행렬 계산 결과 출력

- mtrxC 를 생성하고, mtrxA 와 mtrxB 의 행렬 덧셈을 계산하여 결과를 저장한 후, fprintMtrx()를 사용하여 출력하라.
- mtrxD 를 생성하고, mtrxA 와 mtrxB 의 행렬 뺄셈을 계산하여 결과를 저장한 후, fprintMtrx()를 사용하여 출력하라.
- mtrxE 를 생성하고, mtrxA 와 mtrxC 의 행렬 곱셈을 계산하여 결과를 저장한 후, fprintMtrx()를 사용하여 출력하라.

```
/**    main.c */

#include <iostream>
#include <fstream>
#include "Class_Mtrx.h"
using namespace std;

void main()
{
    ifstream fin;
    ofstream fout;

    fin.open("Matrix_data.txt");
    if (fin.fail())
    {
        cout << "Error in opening Matrix_5x5_data.txt !!" << endl;
        exit;
    }

    fout.open("output.txt");
    if (fout.fail())
    {
        cout << "Error in opening Matrix_operations_results.txt !!" << endl;
        exit;
    }

    Mtrx mtrxA(fin);
    mtrxA.setName("MtrxA");
    int n_row = mtrxA.getN_row();
    int n_col = mtrxA.getN_col();
    mtrxA.fprintMtrx(fout);

    Mtrx mtrxB(fin);
    mtrxB.setName("MtrxB");
    mtrxB.fprintMtrx(fout);

    Mtrx mtrxC(fin);
```

```
    mtrxC.setName("MtrxC");
    mtrxC.fprintMtrx(fout);

    Mtrx mtrxD(mtrxA.getN_row(), mtrxB.getN_col());
    mtrxD = mtrxA.add(mtrxB);
    mtrxD.setName("MtrxD = mtrxA.add(mtrxB)");
    mtrxD.fprintMtrx(fout);

    Mtrx mtrxE(mtrxA.getN_row(), mtrxB.getN_col());
    mtrxE = mtrxA.sub(mtrxB);
    mtrxE.setName("MtrxE = mtrxA.sub(mtrxB)");
    mtrxD.fprintMtrx(fout);

    Mtrx mtrxF(mtrxA.getN_row(), mtrxC.getN_col());
    mtrxF = mtrxA.multiply(mtrxC);
    mtrxF.setName("MtrxF = mtrxA.multiply(mtrxC)");
    mtrxF.fprintMtrx(fout);

    fout.close();

} // end of main()
```

3.3 입력 데이터 파일과 실행 결과

```
5  7
1.0 2.0 3.0 4.0 5.0 6.0 7.0
2.0 3.0 4.0 5.0 1.0 7.0 8.0
3.0 2.0 5.0 3.0 2.0 4.0 6.0
4.0 3.0 2.0 7.0 2.0 1.0 9.0
5.0 4.0 3.0 2.0 9.0 6.0 9.0

5  7
1.0 0.0 0.0 0.0 0.0 1.0 2.0
0.0 1.0 0.0 0.0 0.0 2.0 3.0
0.0 0.0 1.0 0.0 0.0 3.0 4.0
0.0 0.0 0.0 1.0 0.0 4.0 5.0
0.0 0.0 0.0 0.0 1.0 5.0 6.0

7  5
1.0 2.0 3.0 4.0 5.0
6.0 7.0 2.0 3.0 4.0
5.0 1.0 7.0 8.0 3.0
2.0 5.0 3.0 2.0 4.0
6.0 4.0 3.0 2.0 7.0
2.0 1.0 9.0 5.0 4.0
3.0 2.0 9.0 6.0 9.0
```

```
MtrxA =
┌                          ┐
|  1.0  2.0  3.0  4.0  5.0  6.0  7.0 |
|  2.0  3.0  4.0  5.0  1.0  7.0  8.0 |
|  3.0  2.0  5.0  3.0  2.0  4.0  6.0 |
|  4.0  3.0  2.0  7.0  2.0  1.0  9.0 |
└  5.0  4.0  3.0  2.0  9.0  6.0  9.0 ┘

MtrxB =
┌                          ┐
|  1.0  0.0  0.0  0.0  0.0  1.0  2.0 |
|  0.0  1.0  0.0  0.0  0.0  2.0  3.0 |
|  0.0  0.0  1.0  0.0  0.0  3.0  4.0 |
|  0.0  0.0  0.0  1.0  0.0  4.0  5.0 |
└  0.0  0.0  0.0  0.0  1.0  5.0  6.0 ┘

MtrxC =
┌                    ┐
|  1.0  2.0  3.0  4.0  5.0 |
|  6.0  7.0  2.0  3.0  4.0 |
|  5.0  1.0  7.0  8.0  3.0 |
|  2.0  5.0  3.0  2.0  4.0 |
|  6.0  4.0  3.0  2.0  7.0 |
|  2.0  1.0  9.0  5.0  4.0 |
└  3.0  2.0  9.0  6.0  9.0 ┘

MtrxD = mtrxA.add(mtrxB) =
┌                          ┐
|  2.0  2.0  3.0  4.0  5.0  7.0  9.0 |
|  2.0  4.0  4.0  5.0  1.0  9.0 11.0 |
|  3.0  2.0  6.0  3.0  2.0  7.0 10.0 |
|  4.0  3.0  2.0  8.0  2.0  5.0 14.0 |
└  5.0  4.0  3.0  2.0 10.0 11.0 15.0 ┘

MtrxD = mtrxA.add(mtrxB) =
┌                          ┐
|  2.0  2.0  3.0  4.0  5.0  7.0  9.0 |
|  2.0  4.0  4.0  5.0  1.0  9.0 11.0 |
|  3.0  2.0  6.0  3.0  2.0  7.0 10.0 |
|  4.0  3.0  2.0  8.0  2.0  5.0 14.0 |
└  5.0  4.0  3.0  2.0 10.0 11.0 15.0 ┘

MtrxF = mtrxA.multiply(mtrxC) =
┌                          ┐
|  99.0  79.0 172.0 124.0 160.0 |
|  94.0  81.0 193.0 144.0 161.0 |
|  84.0  64.0 153.0 124.0 134.0 |
|  87.0  93.0 149.0 118.0 165.0 |
└ 141.0 111.0 212.0 162.0 226.0 ┘
```

<2021-2 객체지향형 프로그래밍과 자료구조 Lab 3 Oral Test>

학번		성명		점수	

(1) C/C++ 프로그래밍에서 사용되는 포인터 (pointer)의 사용 용도에 대하여 설명하라. (핵심포인트: 함수의 호출과 반환에서 사용, 동적 메모리 할당, 자기참조 구조체/클래스)

(2) C/C++ 프로그래밍에서 사용되는 포인터에 +/- 1 의 덧셈/뺄셈 연산이 어떤 의미를 가지는가에 대하여 예를 들어 설명하라. (핵심포인트: 포인터의 자료형 (예: char, int, double, struct Student, class Person)에 따라 포인터가 가리키는 메모리 블록의 크기가 달라짐)

(3) C++ 클래스의 데이터 멤버로 2 차원 동적 배열이 사용되어야 하는 경우, 생성자에서 생성하는 방법과 소멸자에서 2 차원 동적 배열을 어떻게 삭제하는 방법에 대하여 각각 예를 들어 설명하라. (핵심 포인트: C++ 클래스 생성자에서 2 차원 배열의 동적 생성과 삭제)

(4) Windows 환경에서 실행 중인 프로그램 (process)의 가상 메모리 맵에 설명하라. (핵심 포인트: 가상 메모리 멥의 각 구간이 어떤 용도로 사용되는가, 스택과 힙에서의 메모리 할당에 따른 주소 변화에 대한 설명)