myMatrix.h

```
#include "../../include/myMatrix.h"
```

자신의 폴더에 위치에 있는 myMatrix.h라는 헤더파일 가져오는 방법

```
#include <iostream>
#include <string>
#include <fstream>
```

```
typedef struct {
    double** at;//2d array, 더블 포인터
    int rows, cols;//dimension 정보 저장
}Matrix;
```

행렬을 표현할 구조체 선언해주기, 앞으로 이런 형식으로 행렬을 선언해 줄 것이다.

행렬을 가져올 때 기본적으로 지켜야 될 규칙들

```
#define ASGN        999     // enter your assignment number
#define EVAL        0       // [¡Ø DO NOT EDIT !!!]

#include "../../include/myMatrix.h"
int main(int argc, char* argv[])
{
    /*   [¡Ø DO NOT EDIT !!!]   Resources file path setting for evaluation  */
    std::string path = "C:/NP_data/Assignment" + std::to_string(ASGN) + "/";


#if EVAL
    path += "eval/";
#endif
    return 0;
}
```

curve fitting

arr2Mat()

1차원 배열을 행렬로 만들때 사용한다.

```
Matrix  arr2Mat(double* _1Darray, int _rows, int _cols);
```

Parameters

> double* _1Darray: 행렬로 만들 배열
>
> _rows: 만들고 싶은 행의 개수
>
> _cols: 만들고 싶은 열의 개수

Example code

```
#include "../../include/myMatrix.h"
int main(int argc, char* argv[]){
    int M = 6;
    double T_array[] = { 30, 40, 50, 60, 70, 80 };
    Matrix T = arr2Mat(T_array, M, 1);
    printMat(T, "T");
    freeMat(T);
    return 0;
}
```

output

```
T =
      30.000000
      40.000000
      50.000000
      60.000000
      70.000000
      80.000000
```

Warning

- _1Darray는 배열이어야 한다.
- _rows, _cols는 int 형식이어야 한다.

Error Handling

- 0으로 나눌 시 오류가 발생한다.

linearFit()

1차 방정식꼴로 근사하고 싶을 때 사용한다.

$$\hat{y} = a_0 + a_1 x$$

$$\hat{z} = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}$$

$$a1 = \frac{nS_{xx} - S_x S_y}{nS_{xx} - S_x S_x}, a0 = \frac{S_{xx} S_y - S_{xy} S_x}{nS_{xx} - S_x S_x}$$

```
Matrix  linearFit(Matrix _X, Matrix _Y);
```

Parameters

> _X: 그래프를 그릴 때 x축에 들어가는 값들이 모인 행렬
>
> _Y: 그래프를 그릴 때 y축에 들어가는 값들이 모인 행렬

Example code

```
#include "../../include/myMatrix.h"
int main(int argc, char* argv[]){
    int M = 6;
    double T_array[] = { 30, 40, 50, 60, 70, 80 };
    double P_array[] = { 1.05, 1.07, 1.09, 1.14, 1.17, 1.21 };

    Matrix T = arr2Mat(T_array, M, 1);
    Matrix P = arr2Mat(P_array, M, 1);

    Matrix z = linearFit(T, P);
    printMat(z, "z");
    freeMat(T);
    freeMat(P);
    freeMat(z);
    return 0;
}
```

output

```
z =
       0.940952
       0.003286
```

Warning

- 1차 근사만 가능한다.
- linear형태만 가능하다.

Error Handling

- _X와 _Y의 열의 개수가 같지 않으면 오류가 발생한다.
- _X와 _Y의 행의 개수가 같지 않으면 오류가 발생한다.
- _X와 _Y의 열의 개수가 1이 아니면 오류가 발생한다.

polyFit()

n차 방정식꼴로 근사하고 싶을 때 사용한다.

$$\hat{y} = a_0 + a_1 x \cdots + a_n x^n$$

$$A = \begin{bmatrix} 1 & x_1 & \dots & x_1^n \\ \vdots & & & \vdots \\ 1 & x_m & \dots & x_m^n \end{bmatrix}$$

$$A^T A = \begin{bmatrix} m & \sum_{k=1}^{m} x_k^1 & \dots & \sum_{k=1}^{m} x_1^n \\ \vdots & & & \vdots \\ \sum_{k=1}^{m} x_k^n & \sum_{k=1}^{m} x_k^{n+1} & \dots & \sum_{k=1}^{m} x_1^{2n} \end{bmatrix}$$

$$A^T y = \begin{bmatrix} \sum_{k=1}^{m} y_k \\ \vdots \\ \sum_{k=1}^{m} y_k x_k^n \end{bmatrix}$$

$$z = (A^T A)^{-1} A^T y$$

$$\hat{z} = \begin{bmatrix} a_0 \\ \vdots \\ a_n \end{bmatrix}$$

```
Matrix  polyFit(Matrix _X, Matrix _Y, int n);
```

Parameters

> _X: 그래프를 그릴 때 x축에 들어가는 값들이 모인 행렬
>
> _Y: 그래프를 그릴 때 y축에 들어가는 값들이 모인 행렬
>
> n: 만들고 싶은 차수

Example code

```
#include "../../include/myMatrix.h"
int main(int argc, char* argv[]){
    int n = (6 - 0) / 0.4 + 1;
    double strain_array[16] = { 0 };
    for (int i = 0; i < n - 1; i++) {
        strain_array[i + 1] =strain_array[i]+ 0.4;
    }

    double stress_array[16] = { 0, 3, 4.5, 5.8,5.9,5.8, 6.2, 7.4, 9.6, 15.6,
20.7, 26.7, 31.1, 35.6, 39.3, 41.5 };
    int order = 3;//차수를 의미
    Matrix strain = arr2Mat(strain_array, n, 1);
    Matrix stress = arr2Mat(stress_array, n, 1);
    Matrix z = polyFit(strain, stress,order);

    printMat(z, "z");
    freeMat(strain);
    freeMat(stress);
    freeMat(z);
    return 0;
}
```

output

```
z is  =
       2.892982
      -1.988223
       1.803028
      -0.054117
```

Warning

- 차수는 행렬의 행의 개수보다 작아야 한다.
- A는 m x (n+1) 행렬이다.
- ATA는 (n+1) x (n+1) 행렬이다.
- ATy는 (n+1) x 1행렬이다.

Error Handling

- _X와 _Y의 열의 개수가 같지 않으면 오류가 발생한다.
- _X와 _Y의 행의 개수가 같지 않으면 오류가 발생한다.
- _X와 _Y의 열의 개수가 1이 아니면 오류가 발생한다.
- 차수가 행렬의 행의 개수보다 많으면 오류가 발생한다.

expFit()

exp꼴로 근사하고 싶을 때 사용한다. 1차 방정식꼴로 근사하는 방법과 유사하다.

$$y = c_0 e^{c_1 x}$$
$$ln(y) = c_1 x + ln(c_0)$$
$$\hat{y} = a_0 + a_1 x$$
$$a_0 = ln(c_0), a_1 = c_1$$

```
Matrix  expFit(Matrix _X, Matrix _Y);
```

Parameters

> _X: 그래프를 그릴 때 x축에 들어가는 값들이 모인 행렬
>
> _Y: 그래프를 그릴 때 y축에 들어가는 값들이 모인 행렬

Example code

```
#include "../../include/myMatrix.h"
int main(int argc, char* argv[]){
    int n = (30 - 2) / 2 + 1;
    double time_array[15] = { 2};
    for (int i = 0; i < n - 1; i++) {
        time_array[i + 1] = time_array[i] + 2.0;
    }

    double Voltage_array[15] = {9.7, 8.1,6.6,5.1,4.4,3.7,2.8,2.4,2.0,1.6, 1.4,
1.1,0.85,0.69,0.6  };

    Matrix time = arr2Mat(time_array, n, 1);
    Matrix Voltage = arr2Mat(Voltage_array, n, 1);

    Matrix z = expFit(time, Voltage)
```

```
    printMat(z, "z");
    freeMat(time);
    freeMat(Voltage);
    freeMat(z);
    return 0;
}
```

output

```
z =
      11.913118
      -0.100161
```

Warning

- 1차 근사만 가능한다.
- 지수함수꼴로 근사를 한다.

Error Handling

- _X와 _Y의 열의 개수가 같지 않으면 오류가 발생한다.
- _X와 _Y의 행의 개수가 같지 않으면 오류가 발생한다.
- _X와 _Y의 열의 개수가 1이 아니면 오류가 발생한다.