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;
}
```

eigenvalue, eigenvector

- QRHousehold()
- eig()
- eigvec()

QRHousehold()

QR분해를 할때 Household 방식을 이용해서 QR분해를 실시한다.

출력을 U=RQ행렬을 출력한다. 출력해서 나온 행렬 U는 위삼각행렬이 아닐 수 있다.

$$R^0 = A, Q^0 = I$$
$$c = R^0_{i1}$$
$$H^{(1)} = I - \frac{2}{v^t v} v v^t$$
$$v = \vec{c} + ||c||\vec{e}$$
$$Q^{(1)} = Q^{(0)} H^{(1)}$$
$$R^{(1)} = H^{(1)} R^{(0)}$$

이 반복을 n-1번 실시한다.

```
Matrix QRHousehold(Matrix A);
```

Parameters

> A: QR분해를 할 행렬

Example code

```
#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
    Matrix matA = txt2Mat(path, "prob1_matA");
    matA = QRHousehold(matA);
    printMat(matA, "matU");
    return 0;
}
```

output

```
matA =
      30.000000       15.000000       20.000000
      15.000000       22.000000       26.000000
      20.000000       26.000000       40.000000

matU =
      66.262295       18.275830       -3.090015
      18.275830       21.698143       -3.245957
      -3.090015       -3.245957        4.039562
```

Warning

- A: n x n 행렬이어야 한다.
- 계산결과 나온 U가 위삼각 행렬이 아니라면 similar matrix가 아니다.
- 출력으로 나올 행렬의 크기는 n x n 이다.

Error Handling

- 0으로 나눌 시 오류가 발생한다.

eig()

eigenvalue를 구하기 위해 사용을 한다. 이때 QRHousehold를 이용해서 eigenvalue를 구한다.

U가 위삼각행렬이 될 때까지 QRHousehold를 반복한다.

```
Matrix eig(Matrix A);
```

Parameters

> A: eigenvalue를 구하고 싶은 행렬

Example code

```
#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
    Matrix matA = txt2Mat(path, "prob1_matA");
    Matrix eigVals=zeros(matA.rows,1)
    eigVals = eig(matA);
    printMat(eigVals,"eigenvalue");
    return 0;
}
```

output

```
eigenvalue =
      73.031016
      15.507761
       3.461223
```

Warning

- A: n x n 행렬이어야 한다.
- 출력으로 나올 eigenvalue 행렬은 n x 1 행렬이어야 한다.

eigvec()

eigenvector를 구하기 위해 사용을 한다.

A행렬에서 eigenvalue를 이용해서 eigenvector를 구한다.

$$(A-I)v = Bv = 0$$

$$V_1 = \begin{bmatrix} 1 \\ v_{12} \\ v_{13} \end{bmatrix} \quad V_2 = \begin{bmatrix} v_{22} \\ 1 \\ v_{23} \end{bmatrix} \quad V_3 = \begin{bmatrix} v_{31} \\ v_{32} \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} b_{22} & b_{23} \\ b_{32} & b_{33} \end{bmatrix} \begin{bmatrix} v_{12} \\ v_{13} \end{bmatrix} = \begin{bmatrix} -b_{21} \\ -b_{31} \end{bmatrix}$$

$$\begin{bmatrix} v_{12} \\ v_{13} \end{bmatrix} = \begin{bmatrix} b_{22} & b_{23} \\ b_{32} & b_{33} \end{bmatrix}^{-1} \begin{bmatrix} -b_{21} \\ -b_{31} \end{bmatrix}$$

$$V_1 = \frac{V_1}{||V_1||}$$

이러한 방식을 이용하여 eigenvector를 구한다.

```
Matrix eigvec(Matrix A);
```

Parameters

> A: eigenvector를 구하고 싶은 행렬

Example code

```
#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
    Matrix matK = txt2Mat(path, "prob1_matK");
    Matrix matU = zeros(matK.rows, matK.cols);
    Matrix matL = eye(matK.rows, matK.cols);
    Matrix matP= eye(matK.rows, matK.cols);
    LUdecomp(matK, matL, matU,matP);

    printMat(matK, "matK");
    printMat(matL, "matL");
    printMat(matU, "matU")
    return 0;
}
```

output

```
eigen vector =
       0.746757       -0.863751        0.040641
      -0.662056        0.254981       -0.824479
      -0.063526        0.434649        0.564432
```

Warning

- A: n x n 행렬이어야 한다.
- eigenvector 행렬로 n x n 행렬이어야 한다.

Error Handling

- A 행렬의 사이즈가 2 x 2, 3 x 3이 아닌 행렬인 경우에는 오류가 발생한다.
- 0으로 나눈 경우 오류가 발생한다.