# 矩陣計算機

## 一、設計動機:

矩陣在線性代數中扮演極為重要的角色，但光是算三階方陣的反矩陣就很花時間了，更何況是更高維的方陣，這讓我萌生想以自己所學寫個矩陣計算機

## 二、輸入與輸出:

### A.主功能表:

```
C:\Users\蔡弘祥\Desktop\計程\個人報告3\final2\final2.exe

Below are the function provided by this program
 1.enter p to print all matrices
 2.enter a to add a new matrix
 3.enter m to choose and modify a matrix
 4.enter d to delete a matrix
 5.enter + to print the addition of the two matrices
 6.enter - to print the subtraction of the two matrices
 7.enter * to print the multiplication of the two matrices or a matrix and a real number
 8.enter / to print the division of a matrix and a real number
 9.enter T to print the transpostion matrix
10.enter D to print the determinant
11.enter A to print the adjugate matrix
12.enter I to print the inverse matrix
13.enter e to end the loop

************************************************************
note that m is for ordinary matrix and s is for square matrix
************************************************************
```

如上，共有十二種功能外加結束，其中因為我把一般矩陣(即行數和列數不等)和方陣分為兩類，故以下在輸入時以 m 代表一般矩陣，s 代表方陣

### B.分項說明:

```
enter somthing here :p

Matrix 1:
  1.00  1.00  1.00
  1.00  1.00  1.00

Matrix 2:
  1.00  1.00
  1.00  1.00
  1.00  1.00

Square Matrix 1:
  1.00  1.00
  1.00  1.00

Square Matrix 2:
  1.00  1.00  1.00
  1.00  1.00  1.00
  1.00  1.00  1.00
```

1.輸入 p，印出所有矩陣

```
enter somthing here :p

no data! add some matrix
```

```
enter somthing here :a

enter column and row :2 3
input the matrix
1 1 1
1 1 1
```

新增一般矩陣

```
enter somthing here :a

enter column and row :2 2
input the square matrix
1 1
1 1
```

新增方陣

2.輸入 a，新增矩陣

```
enter somthing here :m

Matrix 1:
  1.00  1.00  1.00
  1.00  1.00  1.00

Matrix 2:
  1.00  1.00
  1.00  1.00
  1.00  1.00

Square Matrix 1:
  1.00  1.00
  1.00  1.00

Square Matrix 2:
  1.00  1.00  1.00
  1.00  1.00  1.00
  1.00  1.00  1.00

enter type and number for the matrices :s 2
now
  1.00  1.00  1.00
  1.00  1.00  1.00
  1.00  1.00  1.00

input (i,j). (i,j) is element that needs modifying
2 2
input the number you want to replace the element in (2,2)
10
after modifying
  1.00  1.00  1.00
  1.00 10.00  1.00
  1.00  1.00  1.00
```

```
enter somthing here :d

Matrix 1:
  1.00  1.00  1.00
  1.00  1.00  1.00

Matrix 2:
  1.00  1.00
  1.00  1.00
  1.00  1.00

Square Matrix 1:
  1.00  1.00
  1.00  1.00

Square Matrix 2:
  1.00  1.00  1.00
  1.00 10.00  1.00
  1.00  1.00  1.00

enter type and number for the matrix :s 2
```

before

```
enter somthing here :p

Matrix 1:
  1.00  1.00  1.00
  1.00  1.00  1.00

Matrix 2:
  1.00  1.00
  1.00  1.00
  1.00  1.00

Square Matrix 1:
  1.00  1.00
  1.00  1.00
```

after

## 3.輸入 m，修改其中矩陣的其中一個元素值

```
enter somthing here :+

Matrix 1:
  1.00  1.00  1.00
  1.00  1.00  1.00

Matrix 2:
  1.00  2.00  3.00
  4.00  5.00  6.00

enter type and number for the matrices :m 1 2

answer :
  2.00  3.00  4.00
  5.00  6.00  7.00
```

正常輸入

```
enter somthing here :+

Square Matrix 1:
  1.00  2.00
  3.00  4.00

Square Matrix 2:
  1.00  2.00  3.00
  4.00  5.00  6.00
  7.00  8.00  9.00

enter type and number for the matrices :s 1 2
error they cannot be added together
```

無法相加

## 4.輸入 d，刪除矩陣

```
enter somthing here :-

Square Matrix 1:
  1.00  2.00
  3.00  4.00

Square Matrix 2:
 -1.00 -1.00
 -1.00 -1.00

enter type and number for the matrix :s 1 2

answer :
  2.00  3.00
  4.00  5.00
```

正常輸入

```
enter somthing here :-

Square Matrix 1:
  1.00  2.00
  3.00  4.00

Square Matrix 2:
  1.00  2.00  3.00
  4.00  5.00  6.00
  7.00  8.00  9.00

enter type and number for the matrix :s 1 2
error the first subtracted the second
```

無法相減

## 5.輸入+，進行矩陣加法

```
enter somthing here :*

enter m to do matrices multiplication
enter r to do real number multiplication
your choice :r
Matrix 1:
  1.00  1.00  1.00
  1.00  1.00  1.00

Matrix 2:
  1.00  1.00
  1.00  1.00
  1.00  1.00

Square Matrix 1:
  1.00  1.00
  1.00  1.00

Square Matrix 2:
  1.00  1.00  1.00
  1.00  1.00  1.00
  1.00  1.00  1.00

enter type and number for the matrix :m 2
enter the real number
5

answer :
  5.00  5.00
  5.00  5.00
  5.00  5.00
```

```
enter somthing here :*

enter m to do matrices multiplication
enter r to do real number multiplication
your choice :m
Matrix 1:
  1.00  1.00  1.00
  1.00  1.00  1.00

Matrix 2:
  1.00  1.00
  1.00  1.00
  1.00  1.00

Square Matrix 1:
  1.00  1.00
  1.00  1.00

Square Matrix 2:
  1.00  1.00  1.00
  1.00  1.00  1.00
  1.00  1.00  1.00

enter type and number for the first matrix :m 1
Matrix 2
  1.00  1.00
  1.00  1.00
  1.00  1.00

Square Matrix 2
  1.00  1.00  1.00
  1.00  1.00  1.00
  1.00  1.00  1.00

above are the metrices availible for multiplication
enter type and number for the second matrix :m 2

answer :
  3.00  3.00
  3.00  3.00
```

## 6.輸入-，進行矩陣減法

```
enter somthing here :/

Matrix 1:
  2.00  4.00  6.00
  8.00 10.00 12.00

enter type and number for the matrix :m 1
enter the real number :2

answer :
  1.00  2.00  3.00
  4.00  5.00  6.00
```

正常輸入

```
enter somthing here :/

Matrix 1:
  2.00  4.00  6.00
  8.00 10.00 12.00

enter type and number for the matrix :m 1
enter the real number :0
you cannot divide 0
```

除數為 0

## 7.輸入*，進行矩陣乘法或係數積(所有元素*k)

```
enter somthing here :T

Matrix 1:
  1.00  2.00  3.00
  4.00  5.00  6.00

enter type and number for the matrix :m 1
transposition
  1.00  4.00
  2.00  5.00
  3.00  6.00
```

## 8.輸入/，進行矩陣係數積(所有元素/k)

```
enter somthing here :D

Square Matrix 1:
  7.00 11.00
  5.00  8.00

enter the number for the matrix :1
Determinant = 1.00
```

## 9.輸入 T，印出轉置矩陣

```
enter somthing here :A

Square Matrix 1:
  7.00 11.00
  5.00  8.00

enter the number for the matrix :1
adjugate
  8.00-11.00
 -5.00  7.00
```

## 10.輸入 D，印出行列式值

```
enter somthing here :I

Square Matrix 1:
  7.00 11.00
  5.00  8.00

enter the number for the matrix :1
inverse
  8.00-11.00
 -5.00  7.00
```

## 11.輸入 A，印出伴隨矩陣

## 12.輸入 I，印出反矩陣

## 三、程式碼解說:

**A.大綱:**在這支程式中使用到兩個 class，分別為 Matrix(public and protected)和 SquareMatrix(public and private)，其中 SquareMatrix 公開繼承 Matrix 的所有變數與函數。在主函數的部分主要是以 while 迴圈撰寫。

**B.分項說明:**

**甲.matrix class**

一、標頭檔宣告部分

```
class Matrix{
    public:
        //constructor and setup
        Matrix(){}                  //ordinary
        Matrix(int,int);            //construct with (column,row)
        void setup();               //setup
        void modify();              //modify print out the matrix and choose the element to reset

        //function for printing information
        void printRowSize();        //print row size ,which is n
        void printColumnSize();     //print column size ,which is m
        void printTrans();          //print A^T
        void print();               //print the matrix

        //function for getting information
        int getRow();
        int getColumn();
        vector<vector <double> > getData();

        //operator
        Matrix operator+(Matrix&);  //A+B
        Matrix operator-(Matrix&);  //A-B
        Matrix operator*(double);   //k*A
        Matrix operator/(double);   //(1/k)*A
        Matrix operator*(Matrix&);  //AB
        Matrix operator=(Matrix&);  //A=B B assign to A
        bool operator==(Matrix&);   //A=B?

    protected:
        vector <vector <double> > data;     //based on 2D vector

        Matrix *trans;                      //transposition:A^T
        void setTrans();                    //set A^T
};
```

二、較難實現的部分

```
void Matrix::setTrans(){
    Matrix *B = new Matrix;
    for(int i=0;i<getRow();i++){
        vector <double> temp;
        for(int j=0;j<getColumn();j++){
            temp.push_back(data[j][i]);
        }
        B->data.push_back(temp);
    }
    trans=B;
}
```

轉置矩陣:

以 new 宣告一個新的 Matrix，使 $a'_{ij} = a_{ji}$，其中 $a_{ij}$ 為原矩陣之元素，$a'_{ij}$ 為新矩陣之元素，最後再把新宣告的 Matrix 指標 assign 給 trans

乙.squarematrix class

一、標頭檔宣告部分

```
class SquareMatrix : public Matrix{
    friend double Det(vector < vector <double> >);
    public:
        //constructor and setup
        SquareMatrix(){}
        SquareMatrix(int);
        SquareMatrix(Matrix);
        void setup();                   //setup

        //function for printing information
        void printDeterminant();        //print det(B)
        void printAdjugate();           //print adj(B)
        void printInverse();            //print B^(-1)
//      void printCharPoly();

        //function for getting information
        double getDet();
    private:
        double det;                     //determinant:det(B)
        void setDet();                  //set det(B)

        SquareMatrix *adjugate;         //adjugate:adj(B)
        void setAdjugate();             //set adj(B)

        SquareMatrix *inverse;          //B^(-1)
        void setInverse();              //set B^(-1)

//      vector <double> CharPoly;       //characteristic polynomial f(x)
//      void setCharPoly();             //set f(x)
};
```

p.s.原計畫最終要算出特徵方程式，但難度實在太高所以先放棄

二、較難實現的部分

```
double Det(vector < vector <double> > input){
    int n=input.size();
    double output=0;
    if(n==1){
        return input[0][0];
    }else if(n==2){
        return input[0][0]*input[1][1]-input[0][1]*input[1][0];
    }else{
        for(int k=0;k<n;k++){
            vector < vector <double> >next;
            for(int i=0;i<n;i++){
                if(i!=k){
                    vector <double> tmp;
                    for(int j=1;j<n;j++){
                        tmp.push_back(input[i][j]);
                    }
                    next.push_back(tmp);
                }
            }
            if(k%2==0){
                output+=input[k][0]*Det(next);
            }else{
                output-=input[k][0]*Det(next);
            }
        }
        return output;
    }
}
```

行列式的計算:
如果要算行列式值，首先想到的就是降階法。對於一階和二階方陣可以直接計算(雖然三階也可以，但為了簡化程式碼就不特別寫)。在降階的時候我都預設提出第一行當係數，剩下的餘因子矩陣的行列式值再用遞迴表達，最後正負號直接用元素位置判斷

```
void SquareMatrix::setAdjugate(){
    SquareMatrix *C=new SquareMatrix;
    for(int i=0;i<getRow();i++){
        vector <double> c_temp;
        for(int j=0;j<getRow();j++){
            vector < vector <double> >next;
            for(int p=0;p<getRow();p++){
                if(p!=i){
                    vector <double> tmp;
                    for(int q=0;q<getRow();q++){
                        if(q!=j){
                            tmp.push_back(data[p][q]);
                        }
                    }
                    next.push_back(tmp);
                }
            }
            if((i+j)%2==0){
                c_temp.push_back(Det(next));
            }else{
                c_temp.push_back(-Det(next));
            }
        }
        C->data.push_back(c_temp);
    }
    C->setTrans();
    adjugate=(SquareMatrix*)(C->trans);
}
```

伴隨矩陣:
為了計算 adjA ，我們必須先計算每個元素的餘因子矩陣行列式值，再進行轉置，所以才會要用到四層 for 迴圈，最後因為 C->trans 是個指向 Matrix 的指標，所以直接讓他型別轉換升級成 SquareMatrix 的指標

p.s.會要算 adjA 是因為 $A^{-1} = \frac{1}{\det A}\text{adjA}$

丙.main.cpp

```
void printM(vector <Matrix> X){
    for(int i=0;i<X.size();i++){
        cout<<"Matrix "<<i+1<<":"<<endl;
        X[i].print();
    }
}

void printS(vector <SquareMatrix> X){
    for(int i=0;i<X.size();i++){
        cout<<"Square Matrix "<<i+1<<":"<<endl;
        X[i].print();
    }
}
```

printM()和 printS()專門用來印出 M[i]和 S[i]，其中 M[i]是 Matrix 陣列、S[i]是 SquareMatrix 陣列

```
void printInfo(){
    cout<<endl
        <<"Below are the function provided by this program"<<endl
        <<" 1.enter p to print all matrices"<<endl
        <<" 2.enter a to add a new matrix"<<endl
        <<" 3.enter m to choose and modify a matrix"<<endl
        <<" 4.enter d to delete a matrix"<<endl
        <<" 5.enter + to print the addition of the two matrices"<<endl
        <<" 6.enter - to print the subtraction of the two matrices"<<endl
        <<" 7.enter * to print the multiplication of the two matrices or a matrix and a real number "<<endl
        <<" 8.enter / to print the division of a matrix and a real number"<<endl
        <<" 9.enter T to print the transpostion matrix"<<endl
        <<"10.enter D to print the determinant"<<endl
        <<"11.enter A to print the adjugate matrix"<<endl
        <<"12.enter I to print the inverse matrix"<<endl
        <<"13.enter e to end the loop"<<endl<<endl
        <<"************************************************************"<<endl
        <<"note that m is for ordinary matrix and s is for square matrix"<<endl
        <<"************************************************************"<<endl
        <<"enter somthing here :";
}
```

printInfo()專門印出每次 while 迴圈的初始功能表

p.s.main.cpp 會寫到接近四百行是因為要讓輸出好看