# Data Structure Program Assignment #4
# (Due: PM: 9:00, March 24, 2022)

## Sparse Matrix Class and Operations

*Instructor: Jiann-Jone Chen*

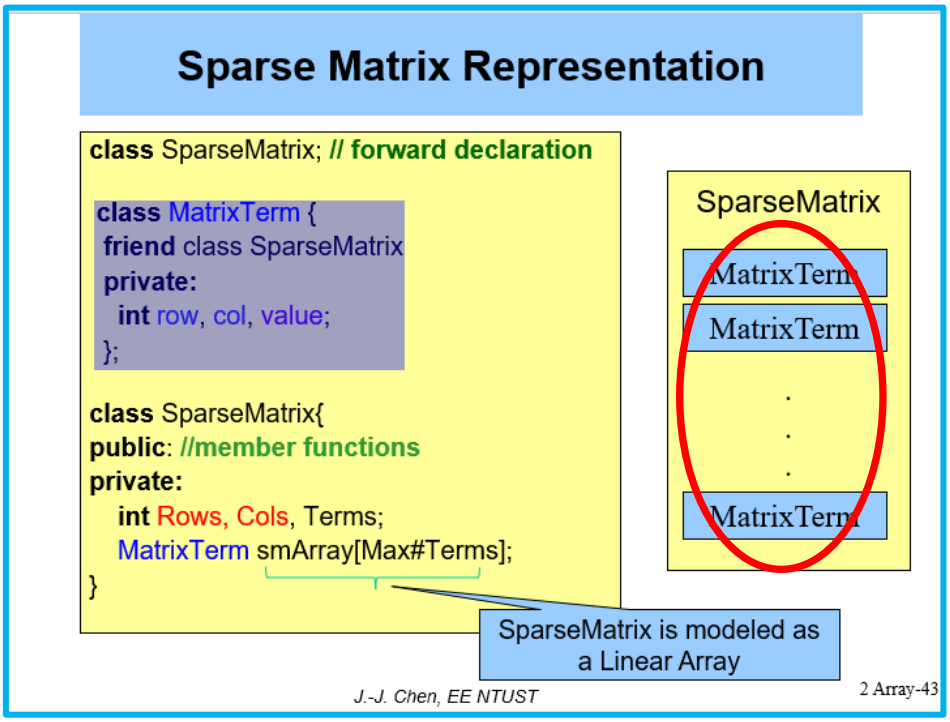
- **Introduction**

A sparse matrix is a matrix in which most elements are zero, as shown on the right figure. In lecture 2, we introduced a matrix term class and a sparse matrix class to store and represent a sparse matrix by a one-dimensional array, as shown in the below figure.

**Sparse Matrix**

Is a matrix in which most elements are zero

$$\begin{bmatrix} -27 & 3 & 4 \\ 6 & 82 & -2 \\ 109 & -64 & 11 \\ 12 & 8 & 9 \\ 48 & 27 & 47 \end{bmatrix} \qquad \begin{bmatrix} 15 & 0 & 0 & 22 & 0 & -15 \\ 0 & 11 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 91 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 & 0 \end{bmatrix}$$




In this homework, you are asked to design a complete sparse matrix based on the template we had introduced in class. The matrix term class and sparse matrix classes are shown below:

MatrixTerm class

```
class MatrixTerm{
    friend ostream & operator<<(ostream & os, SparseMatrix& m);
    friend istream & operator>>(istream & is, SparseMatrix& m);
    friend class SparseMatrix;
public:
    void set(int r, int c, int v) {
        row = r, col = c, value = v;
    };
    void set(MatrixTerm& m) {
        row = m.row; col = m.col; value = m.value;
    };
private: int row,col,value;
};
```

SparseMatrix Class

```
class SparseMatrix
{
   friend ostream & operator<<(ostream & os, SparseMatrix& m);
   friend istream & operator>>(istream & is, SparseMatrix& m);
public:
   SparseMatrix(int ncol = 0, int nrow = 0, int nterm = 20) :
        Rows(nrow), Cols(ncol), Terms(nterm) {
        for (int i = 0; i < Terms; i++) smArray[i].set(0, 0, 0);
    };
   SparseMatrix Transpose();
   SparseMatrix FastTranspose();
   SparseMatrix Add(SparseMatrix b);
   SparseMatrix Multiply(SparseMatrix b);
   SparseMatrix & operator+(SparseMatrix& b);
   SparseMatrix & operator*(SparseMatrix& b);  // polynomial
multiplication

   private:
    int Rows, Cols, Terms;
    MatrixTerm smArray[20];
};
```

Two sparse matrix data files, i.e., “sparseA.txt” and “sparse5000.txt”, are provided for you to verify your program. If your program is correct, the execution results should look like the following figure.

Example1:

main.cpp* | SparseMatirx.cpp | SparseMatrix.h

sparsematrix | (全域範圍) | main(int argc, char * ar

```cpp
#include "SparseMatrix.h"
#include <fstream>
#include <iostream>

int main(int argc, char* argv[])
{
    SparseMatrix A, B, C, D;
    ifstream fin;
    if (argc == 1) fin.open("sparseA.txt");  // open the default sparsematrix data
    else fin.open(argv[1]);
    if (!fin) { // check whether fin is correct or not
        cout << "the input file [" << "sparse2.txt" << "] open error\n"; exit(1);
    }
    else {
        fin >> A >> B>> C >> D;
    }
    fin.close();
    cout << "Matrix A\n" << A << "Matrix B\n" << B;
    cout << "Matrix A + B\n" <<  A + B;
    cout << "Matrix C\n" << C << "Matrix D\n" << D;
    cout << "Matrix C * D\n" <<  C * D;
    cout << "C + D" << C + D;
    cout << "B * C" << B * C << endl ;
    system("PAUSE");A
    return 0;
}
```

```
Matrix A
   2   0   0   0
   4   0   0   3
   8   0   0   1
   0   0   6   0

Matrix B
   0   3   0   6
   0  12   0   7
   2  11   9   0
   1   0   0   8

Matrix A + B
   2   3   0   6
   4  12   0  10
  10  11   9   1
   1   0   6   8

Matrix C
   0   3   0   5
   0   0   7   9
  11   0   3   0

Matrix D
   4   0   9
   0   2   0
   0   0   3
   0   0   6

Matrix C * D
   0   6  30
   0   0  75
  44   0 108


 Can't Addition, invalid matrix dimensions for C + D
 Can't Multiply, invalid matrix dimensions for B * C
```

sparseA.txt - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
4       4       6       // matrix A 4x
0       0       2
1       0       4
1       3       3
2       0       8
2       3       1
3       2       6

4       4       9       // matrix B 4x
0       1       3
0       3       6
1       1       12
1       3       7
2       0       2
2       1       11
2       2       9
3       0       1
3       3       8

3       4       6       // matrix C 3x
0       1       3
0       3       5
1       2       7
1       3       9
2       0       11
2       2       3

4       3       5       // matrix D 4x
0       0       4
0       2       9
1       1       2
2       2       3
3       2       6
```

Example 2: sparsematrix with dimensions 5000x5000.

main.cpp | SparseMatirx.cpp | SparseMatrix.h

sparsematrix | (全域範圍) | main(int argc, char * a

```
#include "SparseMatrix.h"
#include <fstream>
#include <iostream>

int main(int argc, char* argv[])
{
    SparseMatrix A, B, C, D;
    ifstream fin;
    if (argc == 1) fin.open("sparse5000.txt");  // open the default sparsematrix data
    else fin.open(argv[1]);
    if (!fin) { // check whether fin is correct or not
        cout << "the input file [" << "sparse2.txt" << "] open error\n"; exit(1);
    }
    else {
        fin >> A >> B>> C >> D;
    }
    fin.close();
    cout << "Matrix A\n" << A << "Matrix B\n" << B;
    cout << "Matrix A + B\n" <<  A + B;
    cout << "Matrix C\n" << C << "Matrix D\n" << D;
    cout << "Matrix C * D\n" <<  C * D;
    cout << "C + D" << C + D;
    cout << "B * C" << B * C << endl ;
    system("PAUSE");
    return 0;
}
```

```
Matrix A
(    0,   10,   80)
(    0,   30,  100)
(    2,    0,   20)
(    3,    0,  100)
(  100,   60,   60)
(  100,  300,   30)
(2000,2000,   50)
(3000,3000,    1)

Matrix B
(    0,   10,   20)
(    2,    0,   80)
(  100,   60,   40)
(  100,  300,   70)
(2000,2000,   50)
(3000,3000,   99)

Matrix A + B
(    0,   10,  100)
(    0,   30,  100)
(    2,    0,  100)
(    3,    0,  100)
(  100,   60,  100)
(  100,  300,  100)
(2000,2000,  100)
(3000,3000,  100)

Matrix C
(    0,    1,    4)
(    0,    3,    1)
(  100,  200,    4)
(  100,  300,   20)
(  200,  100,    2)
(  222,  200,    4)

Matrix D
(    0,    2,    9)
(    1,    0,   20)
(    2,    2,    3)
(    3,    0,   20)
(  100,  200,   50)
(  200,  222,   25)

Matrix C * D
(    0,    0,  100)
(  100,  222,  100)
(  200,  200,  100)
(  222,  222,  100)


Can't Addition, invalid matrix dimensions for C + D
Can't Multiply, invalid matrix dimensions for B * C
```

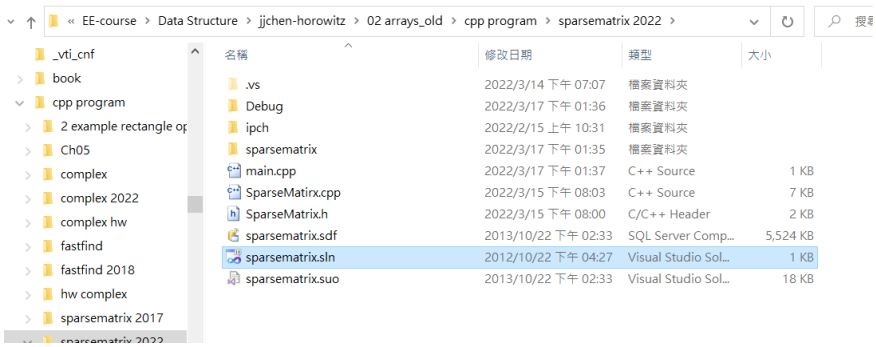
*sparse5000.txt - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
5000    5000    8       // matrix A 4x4 with 8 nc
0       10      80
0       30      100
2       0       20
3       0       100
100     60      60
100     300     30
2000    2000    50
3000    3000    1

5000    5000    6       // matrix B 4x4 with 6 nor
0       10      20
2       0       80
100     60      40
100     300     70
2000    2000    50
3000    3000    99



300     400     6       // matrix C 3x4 with 6 nor
0       1       4
0       3       1
100     200     4
100     300     20
200     100     2
222     200     4

400     300     6       // matrix D 4x3 with 5 non-
0       2       9
1       0       20
2       2       3
3       0       20
100     200     50
200     222     25
```

- Steps:

1. Un-compress the homework zip file. Click the sparsematrix.sln file and open it with visual studio 2022.

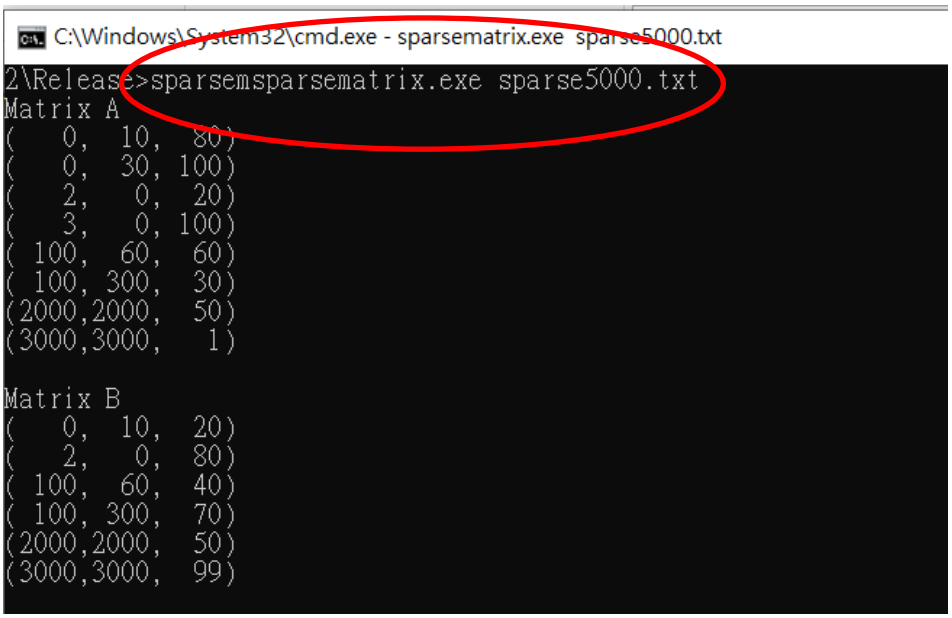



2. This demo project is provided for you to quickly start the design work. The main.cpp is shown below. In the debug mode, the default data file is “sparse.txt.”

```
#include "SparseMatrix.h"
#include <fstream>
#include <iostream>

int main(int argc, char* argv[])
{
     SparseMatrix A, B, C, D;
     ifstream fin;
     if (argc == 1) fin.open("sparse.txt");  // open the default sparsematrix data
     else fin.open(argv[1]);
     if (!fin) { // check whether fin is correct or not
          cout << "the input file [" << "sparse2.txt" << "] open error\n";
exit(1);
     }
     else {
          fin >> A >> B>> C >> D;
     }
     fin.close();
     cout << "Matrix A\n" << A << "Matrix B\n" << B;
     cout << "Matrix A + B\n" <<  A + B;
     cout << "Matrix C\n" << C << "Matrix D\n" << D;
     cout << "Matrix C * D\n" <<  C * D;
     cout << "C + D" << C + D;
     cout << "A * D" << A * D << endl ;
     system("PAUSE");
     return 0;
}
```

3. Design the sparsematrix class such that it is function complete, i.e., it can perform matrix transposition, addition and multiplication through operator overloading methods. For the two Member functions, **transpose()** and **fasttranspose()**, you can copy the code directly from the lecture notes from pages 45, and 47-58, respectively.

4. After finishing your program, you can also execute the program in Release mode and select a different data file, i.e., sparse5000.txt, as shown below




- **Requirement (80%)**
  1. You had to submit the complete project such that the TA can recompile your programs to verify correctness.
  2. Write a short report to describe
     (1) What is all about the program?
     (2) Describe your program by writing notes for each instruction.
     (3) How do you improve this program? List your contributions.

- **Bonus: (20%)**
  1. In this demo program, we set up a fixed-length array to store non-zero elements for easy debugging. You can modify the program such that all sparse matrixes can share one common array for efficient memory utilization.
  2. You can list how do you improve the program efficiency in your design for getting bonuses.

- Note: For this homework, you can discuss with other classmates about the program design instead of copying programs. If you finished the project very early, don't share your program with others. Otherwise, the credits will also be shared by students who submit the same program contents.