Homework1 Program Report

张景浩 PB20010399

2023.3.12

1 问题描述

在 c++ 中搭建一个矩阵模板类，要求有最基本的检索和输出功能，对运算符进行重载，再用 chrono 获取运行时间，并与 Eigen 库进行比较。

2 算法原理

参考 c++ 中 class 类和模板的相关知识。

3 概要设计

```
template<typename T>
class Matrix
{
private:
        int rows, cols;
        T** data;
private:
        void Init();//初始化矩阵
        void Free();//清除矩阵
public:
        Matrix();
        Matrix(int m, int n);
        Matrix(const Matrix& rhs);
        ~Matrix();

        int getRow() const;
        int getCol() const;

        Matrix col(int n);
        Matrix row(int n);
```

```
21          Matrix subMat(int m1, int m2, int n1, int n2);
22
23          void print() const;
24
25          //重载运算符
26          T& operator()(int m, int n) const;
27          T& operator[](int m)const;
28          Matrix operator =(const Matrix& rhs);
29          Matrix operator +(const Matrix& rhs);
30          Matrix operator -(const Matrix& rhs);
31          Matrix operator *(const Matrix& rhs);
32          Matrix operator /(const Matrix& rhs);
33
34          Matrix operator +=(const Matrix& rhs);
35          Matrix operator -=(const Matrix& rhs);
36          Matrix operator *=(const Matrix& rhs);
37          Matrix operator /=(const Matrix& rhs);
38
39          Matrix operator +(T v);
40          Matrix operator -(T v);
41          Matrix operator *(T v);
42          Matrix operator /(T v);
43
44          Matrix operator +=(T v);
45          Matrix operator -=(T v);
46          Matrix operator *=(T v);
47          Matrix operator /=(T v);
48  };
```

4 解决方式

(代码见.cpp 文件)

5 测试结果

```
this matrix has size (12 * 15)
the entries are:
1.041   19.467  7.334   27.5    20.169  16.724  12.478  30.358  27.962  25.464  6.705   29.145  24.281  17.827  10.961
1.491   3.995   12.942  5.827   6.436   33.391  15.604  4.902   1.153   1.292   13.382  18.421  19.716  20.718  20.895
6.447   22.726  15.771  12.538  2.869   20.912  26.667  27.299  18.035  10.894  29.703  24.811  32.322  31.333  18.673
5.664   16.141  8.711   29.253  7.868   26.547  28.644  33.662  33.757  21.037  13.859  9.723   10.741  28.529  1.778
13.316  4.035   23.19   2.842   1.288   31.106  10.04   9.942   20.264  23.648  28.446  24.805  16.89   7.729   25.37
16.35   16.006  32.101  25.393  4.548   20.629  13.623  25.084  20.954  19.756  12.84   5.966   8.376   14.931  27.308
17.944  33.439  25.626  12.323  6.537   22.538  17.118  3.082   23.929  17.541  5.833   32.115  5.639   30.658  23.704
10.93   14.977  3.306   32.673  23.386  6.021   29.745  27.924  20.072  7.27    6.829   27.777  16.573  6.097   17.512
24.986  14.29   10.161  19.636  23.355  25.767  24.655  16.574  5.031   13.052  28.35   2.15    17.941  22.724  14.966
4.43    32.107  31.191  19.007  12.337  16.457  13.287  28.753  11.383  15.945  9.909   33.209  10.758  25.221  19.588
7.422   25.946  28.506  14.03   17.413  30.168  1.9     33.591  19.762  2.655   18.41   7.359   28.624  21.537  22.548
7.483   28.595  5.041   4.602   25.35   11.291  31.836  10.374  12.02   5.596   25.021  28.348  24.199  20.668  25.484

this matrix has size (15 * 4)
the entries are:
9.281   5.734   1.053   2.999
27.418  28.938  7.9     4.788
19.127  1.467   4.728   15.893
25.648  23.483  18.807  3.421
15.31   7.617   23.813  10.514
15.309  8.616   19.935  18.451
21.6    6.249   17.519  32.556
23.798  31.303  7.224   12.008
6.844   33.609  15.989  33.702
4.195   21.485  4.093   15.343
31.523  2.587   30.314  10.503
8.448   26.2    14.458  7.618
21.58   20.796  15.798  16.281
20.589  21.798  29.009  28.157
21.472  24.622  19.538  13.292

this matrix has size (12 * 4)
the entries are:
4827.96 5983.08 4249.17 4266.86
3475.25 2827.52 3348.04 3021.51
5954.53 5617.41 5022.76 4966.3
5054.34 5486.81 4404.95 4923.96
4112.56 4002.64 3735.92 3834.59
4888.48 4773.3  3702.8  4005.26
4732.2  5251.74 4100.2  4393.85
4647.55 5043.13 3899.82 3675.08
5178.09 3921.71 4437.4  3909.5
5327.32 5542.11 4137.67 4069.95
5588.4  5202.16 4412.26 4180.79
5192.32 4712.8  4668.78 4175.47

 Matrix_1 solve time cost = 0.0672418 seconds.
```

```
 7.038 25.179  19.19 30.657  8.958  7.191 20.815 23.888 20.156 12.511 17.202  3.634 25.272 21.055 21.328
23.646 27.362  5.886 19.875 29.433 30.869 21.142 24.844  2.416 22.881 32.998 11.322 19.651 11.021  6.699
 4.557 29.476 28.892 25.389  6.075 11.712   3.6   3.51 22.003 27.869 18.861 15.688 14.401 10.789 16.255
17.423  6.002 11.585 25.182 11.285 28.088 32.426 29.617 24.757 10.832 31.932  5.169  3.154 26.721 18.189
20.976 32.329  3.368 29.692 22.425 11.555  4.434 17.549  8.441 10.512 31.145  19.06 22.718  4.753 17.139
13.423 17.279 26.996 17.687 13.529 23.549 18.437 20.866 13.949  1.193 24.195  4.297 21.416 29.286 17.105
25.488 17.282 13.455 26.734 19.114 12.701 32.316 21.671  6.786 13.263  5.313 25.355 32.185 21.053  1.912
11.808  2.832 21.945  5.313 28.756 29.321 20.558 24.646 28.982  1.481  5.144 24.196 21.222  8.129  3.161
 6.535  21.45 12.173 11.466 13.044 22.659 27.292 27.439 18.253 21.024 27.154  30.51  5.745 21.649 14.186
 9.313  5.474 29.022  3.168 15.018 19.787 10.905 18.958  8.391 11.202  4.625 27.477  5.414 10.314 26.824
30.334 26.874 25.372 21.159 12.833  29.07  8.487 29.297  8.518  9.177 18.773  33.27  2.763  3.668 18.192
14.985  4.102   9.48 30.213  8.627  5.802  5.099 31.527  3.625  2.543  2.924 12.023 30.972 14.061 15.181

32.003 28.432 18.505 28.593
23.725 14.031  9.492  1.142
18.222 32.286 14.064    8.9
20.187   9.36 23.413 31.974
 15.27  30.17  1.235 31.833
20.711  26.76 19.896  5.667
 8.285  13.55   1.14 14.694
 3.695 22.624 29.019  3.125
27.576 22.694 23.658 27.302
18.371 23.466  5.678 23.593
24.851 26.484  2.018 29.464
22.119 24.152    3.8 19.087
 32.06  2.926  10.01  5.757
 33.17 21.315 10.576 31.227
13.043 23.758  8.164  6.109

5417.93 5068.88 3474.15 4442.37
 5910.5 6185.57 3305.62 5187.76
5228.32 5028.49  2878.8 4189.02
5565.83 6188.32 3795.05 5435.41
5513.39 5177.92 3021.74  4659.2
5596.65 5603.85 3383.63 4455.75
5792.06 5303.17 3296.23 4939.13
4740.66 5345.61 3156.54 4001.63
5493.84 6156.48 3182.71 4889.26
3845.67 4979.88 2439.57 3113.03
5527.86 6427.48 3755.62  4468.7
3912.83 3531.13 2938.19 3149.23

 Matrix_2 solve time cost = 0.094003 seconds.
```

6 总结

在 Matrix.h 中的 Matrix 类能够正常运行，并用重载后的运算符完成矩阵的运算，且运行时间与 Eigen 库在一个量级。

关于系数矩阵的类，我参考了三元数组的结构来表达一个稀疏矩阵，并实现了一些相关的功能。