Report for project 5

刘一郴 12112609

一、成员参数

```
template<typename T>
class Matrix{
    private:
    size_t row;           //row number
    size_t col;           //column number
    shared_ptr<T *> entry;//store all entries in wich (*entry)[(i - 1)* step + (j - 1)] is entry in (i, j)
    size_t step;          //step length
```

使用了 shared_ptr 来复用内存，利用其特性，便捷省事。

step 用来应对 ROI。

综上取元素的操作为（*entry）[i * step + j]

二、代码细节解释

```
14      public:
15      //not allowed.
16      Matrix() = delete;
17      //set entry to be shared_ptr
18      Matrix(size_t row, size_t col, size_t step = col):row(row), col(col), step(step){
19          entry = make_shared<T *>( (T *) malloc(col * row * sizeof(T) ) );
20      }
21      //delete function. need special care for shared_ptr
22      ~Matrix(){
23          entry = nullptr;
24      }
```

由于要防止内存泄漏，与及时维护复用内存的指向数。在 class 的声明与释放时要对 shared_ptr 特殊照顾。

三、功能函数实现

1.赋值操作：由于 shared_ptr 的特性，直接使用系统默认生成的赋值函数即可。

2.全等操作

```
25      //return true IFF all the entry value are equal
26      bool operator == (const Matrix & o){
27          if(*this == o){
28              return true;
29          }
30          if(col != o.col || row != o.row){
31              return false;
32          }
33          for(size_t i = 0;i< row;i++){
34              for(size_t j = 0;j< col;j++){
35                  if((*entry)[i * step + j] != (*o.entry)[i * o.step + j]){
36                      return false;
37                  }
38              }
39          }
40          return true;
41      }
```

如果两个矩阵元素都相同，直接返回 true。若大小不匹配，直接返回 false。否则元素逐项比对。O（row * col）

3.加减法操作

```cpp
    //return the matrix equal to the sum of two matrix; throw error IFF size not match
    Matrix operator + (const Matrix & o){
        Matrix ans(row, col);
        if(row != o.row || col != o.col){
            throw("erro size do not match in +");
        }
        for(size_t i = 0;i< row;i++){
            for(size_t j = 0;j< col;j++){
                (*ans.entry)[i * ans.step + j] = (*entry)[i * step + j] + (*o.entry)[i * o.step + j];
            }
        }
        return ans;
    }
    //return the matrix equal to the sub of two matrix; throw error IFF size not match
    Matrix operator - (const Matrix & o){
        Matrix ans(row, col);
        if(row != o.row || col != o.col){
            throw("erro size do not match in -");
        }
        for(size_t i = 0;i< row;i++){
            for(size_t j = 0;j< col;j++){
                (*ans.entry)[i * ans.step + j] = (*entry)[i * step + j] - (*o.entry)[i * o.step + j];
            }
        }
        return ans;
    }
```

逐项操作。无亮点。O(row*col)

4.乘法操作

```cpp
    //return the matrix equal to the mul of two matrix; throw error IFF size not match
    Matrix operator * (const Matrix & o){
        Matrix ans(row, o.col);
        if(col != o.row){
            throw("erro size do not match in *");
        }
        for(size_t i = 0;i< row;i++){
            for(size_t j = 0;j< o.col;j++){
                (*ans.entry)[i * ans.step + j] = 0;
            }
        }
        for(size_t i = 0;i< row;i++){
            for(size_t k = 0;k< col;k++){
                for(size_t j = 0;j< o.col;j++){
                    (*ans.entry)[i * ans.step + j] += (*entry)[i * step + k] * (*o.entry)[k * o.step + j];
                }
            }
        }
        return ans;
    }
```

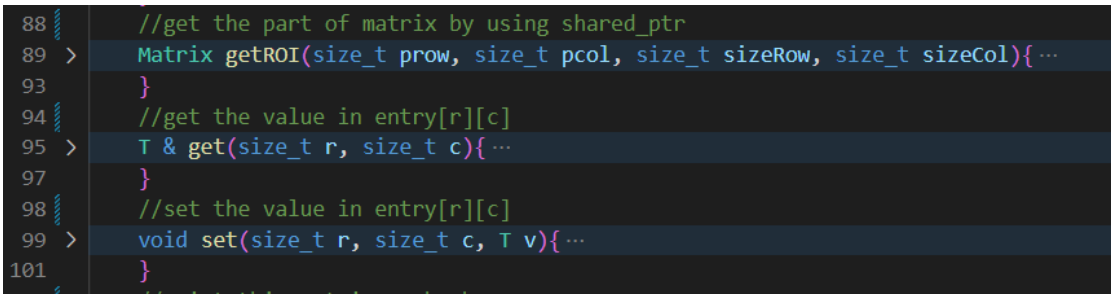
稍微优化乘法顺序。尽可能让读取内存连续。O(row*col*new col)

四、ROI

```cpp
    //get the part of matrix by using shared_ptr
    Matrix getROI(size_t prow, size_t pcol, size_t sizeRow, size_t sizeCol){
        Matrix ans(sizeRow, sizeCol, step);
        ans.entry = make_shared<T *> ((*entry)+ prow * step + pcol);
        return ans;
    }
```

新映射的矩阵的 step 要与前矩阵相同，然后利用 shared_ptr 的特性，将正确的首位置赋值给新矩阵的 shared_ptr。

五、工具函数



print() set() get()

六、代码:

https://github.com/cunlidaniang/coding-workshop/tree/master/c%2B%2B%20workshop/project%205