CS205 C/C++ Programming - Project 5

Name: 钟元吉(Zhong Yuanji)

SID: 12012613

CS205 C/C++ Programming - Project 5

- Part 1 - Analysis
  - 错误检查
  - 内存管理
  - 矩阵的隐式转换
- Part 2 - Code
  - 文件说明
  - 部分代码展示
- Part 3 - Result & Verification
  - Test case #1: 基本要求的实现
  - Test case #2: 对错误类型的检查
  - Test case #3: 其他特殊功能
- Part 4 - Difficulties & Solutions
  - 问题：在比较矩阵时遇到的问题
- Part 5 - Summary

Part 1 - Analysis

> Design a class for matrices, and the class should contain the data of a matrix and related information such the number of rows, the number of columns, the number of channels, etc. The Matrix can support different data types, use soft or hard copy to manage memory and support operators =, ==, !=, <, >, <=, >=, +, -, *, /, ^.

本次Project要求我们仅通过C++语言建立对不同数据格式的、支持行列与通道数的矩阵类，实现矩阵软拷贝和硬拷贝的内存管理，实现运算符重载和矩阵类型转换。

错误检查

在本次Project中，考虑到使用者在使用矩阵类时容易传入空数据或错误的数据类型，对空矩阵进行操作，对大小不匹配的矩阵进行运算等错误，我们的程序进行以下的错误检查：

错误类型	相关函数
使用Matrix<_T>中不允许的模板数据类型(编译报错)	错误类名
构造指针为0的矩阵	传入行、列、通道(=1)数和数据指针的构造函数
通过错误格式的字符串构造矩阵	传入字符串的构造函数
传入矩阵为空矩阵	括号索引、+-*/^运算、sub()、subCopy()、cofactorMatrix()
传入矩阵为空矩阵	det()、inv()、min()、max()、transpose()、rotate90()、getChannelMat()

错误类型	相关函数
传入行、列、通道数越界	括号索引、sub()、subCopy()、cofactorMatrix()、getChannelMat()
运算中矩阵行数和列数不匹配	+-*/运算
运算中矩阵不是方阵	^运算、det()、inv()
运算中矩阵不可逆	inv()

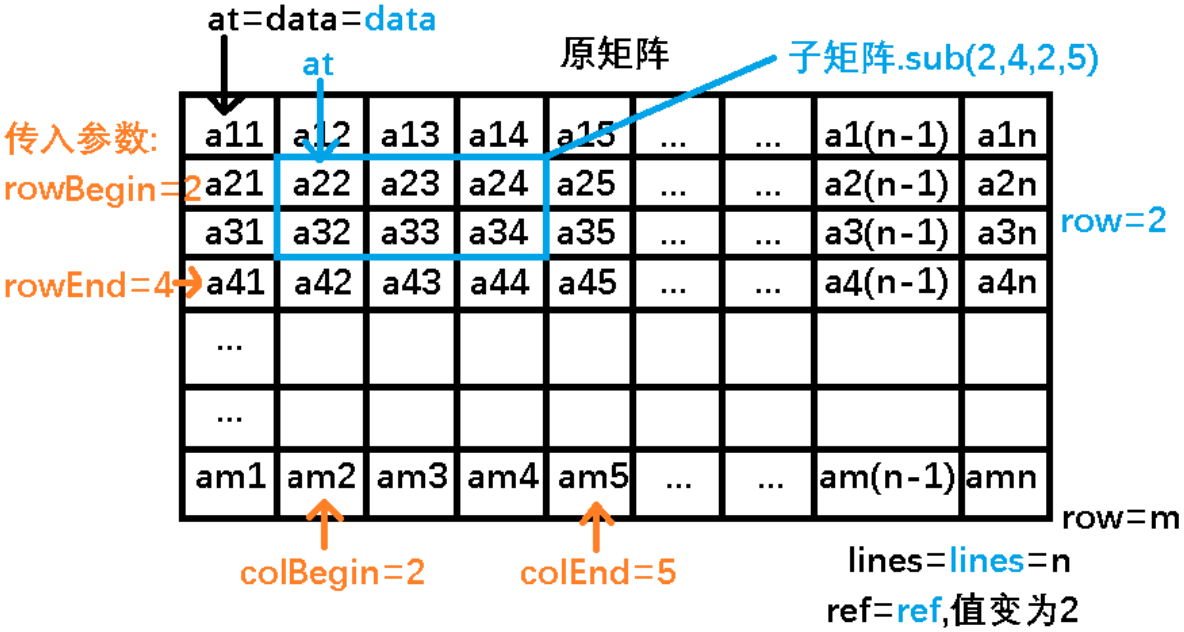
- 首先，不是所有类型都可以成为矩阵的数据。程序中的矩阵类只可以构造为`bool, char, unsigned char, short, unsigned short, int, unsigned int, long, unsigned long, long long, unsigned long long, float, double, long double`类型的矩阵，如果传入不在以上范围的数据类型，为了避免在与字符串的互相转换、不同类型矩阵转换、矩阵运算与求逆过程中出现错误的情况，程序采用`static_assert`静态断言的方式限制矩阵的数据类型。当传入错误类型构造模板类时，**编译器**识别到`static_assert`静态断言中出现`false`，则会立即**报错**，终止程序，防止错误类型的模板类形成。
- 其次，针对两种不同的构造方法进行传入参数的检验：
    1. 通过直接给定行数、列数、(`new`开辟的、无需手动释放)数据指针、通道数(默认为1)，来构造矩阵时，程序会检查行数、列数、通道数是否为零(因为使用`size_t`类型没有负整数)，检查指针是否为空；若为零或空指针，**返回不含数据的空矩阵**，并进行错误提示，否则进行正常的矩阵构造。其中空矩阵通过`Matrix<_T>()`进行构造，此时不会进行检查。
    2. 另一种则为通过字符串来构造矩阵，程序要求传入的字符串必须使用`[ ,;]`作为分割符。程序会**对空格进行智能分析**：当两数据之间仅使用若干个空格分割时，空格会转换为列分隔符，即`,`，若两数据之间已有列分隔符`,`或行分隔符`;`时，其间的空格均忽略。例如2x2矩阵`[1,2;3,4]`也可以传入为`[1, 2 ; 3 4]`。而对不满足上面规则的传入参数或每行的列数不同的参数，如：`"[1;2,3]"`返回空矩阵，并进行错误提示。
- 在函数调用与运算过程中，程序将对所涉及的矩阵检查是否为空矩阵。若为空矩阵将**返回空矩阵**(如取子矩阵)或**提前设置的静态变量0**(如取行列式)，(这是考虑到对空矩阵使用圆括号进行索引时应返回引用类型，但常值0无法被引用，而通过new产生0变量则有可能将不会被释放)，并进行错误提示。
- 程序对其他错误返回值与上一点相同，并进行错误提示。其中矩阵除法`A/B = A * B.inv()`对右边矩阵要求为非空且行数等于列数、对左边矩阵要求为非空且列数与右边矩阵相同，这个运算暂时不支持矩阵`B`不可逆或行数不等于列数的情况(否则求解时可能出现未知收敛性的迭代过程，这是非常危险的)。
- 对于以上错误，程序采用`cerr`进行报错，报错的同时返回空矩阵或0值，**不会中断程序**，无需使用`try{...}catch(...){...}`捕获，对报错提示、错误主程序文件、报错文件及行位置、报错函数输出，例如：

```
Error: The fisrt Matrix is an empty Matrix when using 'operator()'.
    /mnt/d/VScodeProjects/CppClass/Project5/src/Test1.cpp
    /mnt/d/VScodeProjects/CppClass/Project5/src/Matrix.cpp:243 : operator()
```

通过不中断程序输出错误，可以减少或避免错误操作对后续正确操作的影响，同时也有难定位的弊端（由于gcc编译器只支持主函数文件宏`__BASE_FILE__`、所在文件宏`__FILE__`、所在行数宏`__LINE__`、所在函数宏`__func__`），通过提示错误使用的函数一般可以反推出具体的错误代码位置。

内存管理

- 如果矩阵通过数组存储数据，虽然能避免内存管理的问题，但是在矩阵拷贝、取子矩阵、各种函数返回矩阵(如求矩阵的逆，不能返回引用及指针类型，否则会产生新的内存管理问题)时会直接复制所有数据，当矩阵大小较大时，这将浪费较多内存与运行时间。
- 因此我们考虑使用一维指针来存储数据，同时建立一个`int`指针记录'引用'次数，当使用直接赋值到另一个矩阵或通过构造函数拷贝矩阵时，**数据内容将不会被拷贝**，行列通道数与指针直接拷贝，共用引用次数，并将引用次数加1。通过直接调用`sub(开始行数,结束行数,开始列数,结束列数)`取得的子矩阵也会共享同一片数据，改变矩阵数据指针开始位置，引用次数加1，原理如图：



- 当索引子矩阵的第二行第二个元素`a33`时，对应为子矩阵的`at[1*lines+1]`，因此我们为了将软拷贝引入取子矩阵的方法中，**需要额外增加两个变量**：数据指针`at`和邻行间隔`lines`，虽然这将引入更多的数据复制与更繁琐的初始化，但相对于矩阵数据的复制还是相对更高效的。
- 与此同时，改变软拷贝的矩阵数据将意味着同时改变原矩阵的数据，为了提供不影响原矩阵的拷贝方式，程序使用`copy()`和`subCopy(开始行数,结束行数,开始列数,结束列数)`作为矩阵硬拷贝函数，在硬拷贝过程中，所有数据都会被复制，由于数据类型为C++(或C)中的基础数据类型，在复制过程中使用`memcopy`将加快数据整体拷贝。
- 矩阵的析构：通过函数`clear()`手动将矩阵的数据释放并转化为空矩阵，或当离开矩阵作用域时，程序自动调用析构函数。在析构过程中，如果引用次数超过1，则**引用次数减1**，并将**矩阵所有参数置0**(即转换为空矩阵)；如果引用次数为1，除此之外还将释放矩阵的数据、索引次数所在内存。由于矩阵类的索引次数指针、数据指针均为**私有变量**，通过以上几点即可实现矩阵类的内存管理。

矩阵的隐式转换

- 考虑到不同参数类型的矩阵模板类并不是同一个类，如果不做处理，在使用上存在非常差的兼容性，例如整数矩阵与浮点数矩阵的相加可能就无法实现。为了实现不同矩阵类的加、减、乘、除(乘逆矩阵)与比较运算，程序中使用**构造函数模板**将各种其他数据类型的矩阵模板类对象隐式转换为该类对象，其中的数据为**硬拷贝**(受限于指针类型，只能进行硬拷贝)。

- 这与相同类对象的构造赋值方法并不矛盾，当通过**相同类对象构造**或操作符等号赋值时，编译器会优先匹配同类的构造方法和操作符等号方法，**不会生成模板构造函数**，因此为**软拷贝**；通过**不同类对象构造或隐式转换**时，编译器会**生成模板构造函数**，进行**硬拷贝**。
- 在进行加、减、乘、除(乘逆矩阵)与比较运算等操作符运算时，计算只支持相同数据类型的矩阵类，因此程序遇到不同类型矩阵对象运算时，将后者隐式转换为与前者相同类的对象，再进行运

算。特别注意，**为了保证逆矩阵计算的精确度，通过求逆计算返回的矩阵均为`Matrix<long double>`对象。**

- 虽然通过前面的内存管理，隐式转换并不会导致内存泄露或二次释放，但为了减少程序中的隐式转换，在使用运算操作符时，请使用不指定数据类型的矩阵类名`Matrix`或`auto`来**自动匹配对应的矩阵类**，避免隐式转换。例如：

```
// 程序自动将字符串隐式构造为所需矩阵类对象
Matrix mat1 = (Matrix<int>("[1,2;3,4]") == "[1,3;2,4]"); // 判断符右边隐式构造为int矩阵，返回Matrix<bool>
auto mat2 = Matrix<int>("[1,2;3,5]").inv() * Matrix<int>("[1,2;3,4]"); // 判断符右边隐式转换为long double矩阵，返回Matrix<long double>，建议此处乘号右边直接改为字符串，避免隐式转换
auto mat3 = Matrix<int>("[1,2;3,5]").inv() * "[1,2;3,4]"; // 判断符右边隐式构造为long double矩阵，返回Matrix<long double>
```

Part 2 - Code

文件说明

文件名	内容解释
Matrix.hpp	矩阵类及其函数头、友元函数头
Matrix.cpp	矩阵类函数定义、友元函数定义
Test0.cpp	测试文件 - 基本要求的实现
Test1.cpp	测试文件 - 对错误类型的检查
Test2.cpp	测试文件 - 其他特殊功能

部分代码展示

由于代码较长，这里只展示**通过字符串构造矩阵**和**计算矩阵的逆**的部分。

```
// Create a matrix from string
_TP _MAT::Matrix(const char *strOrg)
{
    // Check if string is empty
    if (!strOrg)
    {
        __PRINT_ERROR("The input string is NULL when initializing a matrix.");
        return;
    }
    // Copy the string
    size_t orgLen = strlen(strOrg);
    char str[orgLen];
    strcpy(str, strOrg);
    // Replace the blanks into ','
    for (size_t i = 0, j = 0; j <= orgLen; ++j)
        if ((str[j] == ',' || str[j] == ';') && str[i - 1] == ',')
            str[i - 1] = str[j];
```

```
        else if (str[j] != ' ')
            str[i++] = str[j];
        else if (j != 0 && ((str[j - 1] >= '0' && str[j - 1] <= '9') || str[j -
1] == '.'))
            str[i++] = ',';
    size_t len = strlen(str);
    // Check format error
    if (len < 3 || str[0] != '[' || str[1] == ',' || str[1] == ';' || str[1] ==
']' || str[len - 1] != ']')
    {
        __PRINT_ERROR("The input string must be with the form [ ,;] when
initializing a matrix.");
        return;
    }
    while (str[len - 1] == ',' || str[len - 1] == ';' || str[len - 1] == ']')
        str[--len] = '\0';
    // Count rows and cols
    size_t countD = 0, countF = 0;
    for (size_t i = 1; i < len; ++i)
        if (str[i] == ';')
            ++countF;
        else if (str[i] == ',' && !countF)
            ++countD;
    // Read from string
    size_t At = 1;
    row = countF + 1, col = countD + 1;
    lines = col, channel = 1;
    ref = new int(1);
    data = new _T[row * col], at = data;
    for (size_t i = 0; i <= countF; ++i)
        for (size_t j = 0; j <= countD; ++j, ++At)
        {
            double temp;
            sscanf(&str[At], "%lf", &temp);
            at[i * lines + j] = (_T)temp;
            for (; At < len; ++At)
            {
                if (str[At] == ',')
                {
                    if (str[At + 1] == ';')
                        ++At;
                    break;
                }
                else if (str[At] == ';')
                {
                    if (j != countD)
                        goto RETURN_Error;
                    break;
                }
                else if ((str[At] < '0' || str[At] > '9') && str[At] != '.' &&
str[At] != '-' && str[At] != 'e' && str[At] != 'E')
                    goto RETURN_Error;
            }
        }
    if (At < len)
```

```
        goto RETURN_Error;
    return;
RETURN_Error:
    __PRINT_ERROR("The input string is not valid when initializing a matrix.");
    clear();
}
// ============================================================================
// Compute the inverse
_TP Matrix<long double> _MAT::inv(size_t chaAt) const
{
    // Check if it is empty
    if (!row || !col || !channel || !ref)
    {
        __PRINT_ERROR("Empty Matrix has no inverse.");
        return Matrix<long double>();
    }
    if (row != col)
    {
        __PRINT_ERROR("Only square Matrix has inverse.");
        return Matrix<long double>();
    }
    // If the size of matrix is 1x1
    if (row == 1)
    {
        if (!at[0])
        {
            __PRINT_ERROR("The Matrix is not invertible.");
            return Matrix<long double>();
        }
        long double ret = 1.L / at[chaAt * row * lines];
        return Matrix<long double>(1, 1, new long double[1]{ret}, 1);
    }
    // If the size of matrix is not 1x1
    long double *data_ = new long double[row * col], matDet = det(chaAt);
    if (!matDet)
    {
        __PRINT_ERROR("The Matrix is not invertible.");
        return Matrix<long double>();
    }
#pragma omp parallel for
    for (size_t i = 0; i < row; ++i)
        for (size_t j = 0; j < col; ++j)
        {
            Matrix tmp = cofactorMatrix(j, i);
            data_[i * col + j] = ((i + j) % 2 ? -tmp.det() : tmp.det()) / matDet;
        }
    return Matrix<long double>(row, col, data_, 1);
}
```

Part 3 - Result & Verification

Test case #1: 基本要求的实现

直接编译并运行`Test0.cpp`，或使用`makefile`编译并运行：

```
make
```

```
g++ ./src/Test0.cpp -o Test0 -w & ./Test0
```

1. 构造int空矩阵a、double单通道矩阵b、size_t双通道矩阵c；
2. 通过改变赋值后矩阵d=c(软拷贝)，构造复制矩阵e(c)(软拷贝)，硬拷贝f=c.copy()、子矩阵g=c.sub(...)(软拷贝)、硬拷贝子矩阵h=c.subCopy()的元素，查看是否同时改变了c来检查软硬拷贝；
3. 展示计算功能，其中`A/B = A * B.inv()`，`A^2 = A * A`，`B^-1 = B.inv()`，`C^0 = Id`，完全不对称矩阵有`A==A.transpose() -> Id`

问题 5　调试控制台　终端

```
====================== 1.Construction of Null Matrix, normal matrix, multi-channel matrix:
Mat a is: Matrix 0x0: []

Mat b is: Matrix 2x2:
[
       1.100000       2.200000
       3.300000       4.400000
]

Mat c is: Channel 0:
Matrix 1x4:
[
              4              3              2              1
]
Channel 1:
Matrix 1x4:
[
              8              7              6              5
]

====================== 3.Copy:
Mat c is: Channel 0:
Matrix 1x4:
[
           1000           2000              2              1
]
Channel 1:
Matrix 1x4:
[
              8              7              6              5
]

Mat g is: Channel 0:
Matrix 1x2:
[
           2000              2
]
Channel 1:
Matrix 1x2:
[
              7              6
]

Mat c is: Channel 0:
Matrix 1x4:
[
           1000           2000              2              1
]
Channel 1:
Matrix 1x4:
[
              8           4000              6              5
]

====================== 4.Calculate:
Mat i is: Matrix 2x2:
[
             11             22
             33             44
]

Mat j is: Matrix 2x2:
[
             70            100
            150            220
]

Mat k is: Matrix 2x2:
[
       1.000000       0.000000
       0.000000       1.000000
]

Mat l is: Matrix 2x2:
[
              7             10
             15             22
]

Mat (l == l.transpose()) is: Matrix 2x2:
[
              1              0
              0              1
```

```
]
```

Test case #2: 对错误类型的检查

直接编译并运行`Test1.cpp`：

```
g++ ./src/Test1.cpp -o Test1 -w & ./Test1
```

```
问题 2    调试控制台    终端


==================== 1.Unallowed type of Matrix (it will cause a compile error)
==================== 2.Matrix is empty (with different tips)
Error: The input row/col/channel are zero or data is empty when initializing a matrix.
        /mnt/d/VScodeProjects/CppClass/Project5/src/Test1.cpp
        /mnt/d/VScodeProjects/CppClass/Project5/src/Matrix.cpp:46 : Matrix
Error: The fisrt Matrix is an empty Matrix when using 'operator()'.
        /mnt/d/VScodeProjects/CppClass/Project5/src/Test1.cpp
        /mnt/d/VScodeProjects/CppClass/Project5/src/Matrix.cpp:250 : operator()
Error: Empty Matrix cannot add a number.
        /mnt/d/VScodeProjects/CppClass/Project5/src/Test1.cpp
        /mnt/d/VScodeProjects/CppClass/Project5/src/Matrix.cpp:408 : operator+
Error: Empty Matrix cannot substract a matrix.
        /mnt/d/VScodeProjects/CppClass/Project5/src/Test1.cpp
        /mnt/d/VScodeProjects/CppClass/Project5/src/Matrix.cpp:411 : operator-
Error: Empty Matrix cannot multiply a number.
        /mnt/d/VScodeProjects/CppClass/Project5/src/Test1.cpp
        /mnt/d/VScodeProjects/CppClass/Project5/src/Matrix.cpp:412 : operator*
Error: Empty Matrix cannot join power.
        /mnt/d/VScodeProjects/CppClass/Project5/src/Test1.cpp
        /mnt/d/VScodeProjects/CppClass/Project5/src/Matrix.cpp:459 : operator^
Error: Empty Matrix has no determinant.
        /mnt/d/VScodeProjects/CppClass/Project5/src/Test1.cpp
        /mnt/d/VScodeProjects/CppClass/Project5/src/Matrix.cpp:521 : det
Error: Empty Matrix have no maximial value.
        /mnt/d/VScodeProjects/CppClass/Project5/src/Test1.cpp
        /mnt/d/VScodeProjects/CppClass/Project5/src/Matrix.cpp:504 : max
==================== 3.Initializing with invalid string
Error: The input string is not valid when initializing a matrix.
        /mnt/d/VScodeProjects/CppClass/Project5/src/Test1.cpp
        /mnt/d/VScodeProjects/CppClass/Project5/src/Matrix.cpp:119 : Matrix
==================== 4.The index was out of range of Matrix
Error: The index was out of range of Matrix when using 'getChannelMat()'.
        /mnt/d/VScodeProjects/CppClass/Project5/src/Test1.cpp
        /mnt/d/VScodeProjects/CppClass/Project5/src/Matrix.cpp:237 : getChannelMat
Error: The index was out of range of Matrix when using 'operator()'.
        /mnt/d/VScodeProjects/CppClass/Project5/src/Test1.cpp
        /mnt/d/VScodeProjects/CppClass/Project5/src/Matrix.cpp:253 : operator()
Error: The index was out of range of Matrix when using 'sub()'.
        /mnt/d/VScodeProjects/CppClass/Project5/src/Test1.cpp
        /mnt/d/VScodeProjects/CppClass/Project5/src/Matrix.cpp:264 : sub
Error: The index was out of range of Matrix when using 'cofactorMatrix()'.
        /mnt/d/VScodeProjects/CppClass/Project5/src/Test1.cpp
        /mnt/d/VScodeProjects/CppClass/Project5/src/Matrix.cpp:314 : cofactorMatrix
==================== 5.The Begin index is bigger than End index
Error: The Begin index is bigger than End index when using 'sub()'.
        /mnt/d/VScodeProjects/CppClass/Project5/src/Test1.cpp
        /mnt/d/VScodeProjects/CppClass/Project5/src/Matrix.cpp:269 : sub
==================== 6.The Matrices are in different size(or not square) when calculati
Error: Matrices of different size(or empty) cannot add together.
        /mnt/d/VScodeProjects/CppClass/Project5/src/Test1.cpp
        /mnt/d/VScodeProjects/CppClass/Project5/src/Matrix.cpp:409 : operator+
Error: Only square Matrix has determinant.
        /mnt/d/VScodeProjects/CppClass/Project5/src/Test1.cpp
        /mnt/d/VScodeProjects/CppClass/Project5/src/Matrix.cpp:525 : det
Error: Only square Matrix has inverse.
        /mnt/d/VScodeProjects/CppClass/Project5/src/Test1.cpp
        /mnt/d/VScodeProjects/CppClass/Project5/src/Matrix.cpp:576 : inv
==================== 7.The Matrix is not invertible
Error: The Matrix is not invertible.
        /mnt/d/VScodeProjects/CppClass/Project5/src/Test1.cpp
        /mnt/d/VScodeProjects/CppClass/Project5/src/Matrix.cpp:584 : inv
```

Test case #3: 其他特殊功能

直接编译并运行`Test2.cpp`：

```
g++ ./src/Test2.cpp -o Test2 -w & ./Test2
```

1. 通过字符串构造单通道矩阵，只能识别空格(','列分割，';'行分割，空格在无逗号时代替逗号)；
2. 自动转换矩阵的数据类型，计算时转换其后矩阵为与第一个出现的矩阵相同的类型；
3. 通过圆括号索引`(row,col,channel=0)`(可写)，`getRow()`，`getCol()`，`getChannel()`，`getChannelMat()`(可写)返回私有变量的访问；
4. 特殊函数：余子式矩阵、返回指定通道矩阵、矩阵整数幂运算(快速幂法)、行列式、逆矩阵、转置、逆时针旋转90度。

问题 5　调试控制台　终端

```
Mat a is: Matrix 3x3:
[
             1             2             3
             4             5             5
             7             8             8
]

Mat b is: Matrix 2x3:
[
             1             2             3
             4             5             6
]

Mat c is: Matrix 2x3:
[
       1.000000      2.200000     -3.100000
   30000.000000      0.050000     -0.000100
]

Mat b + c is: Matrix 2x3:
[
             2             4             0
         30004             5             6
]

Mat c + b is: Matrix 2x3:
[
       2.000000      4.200000     -0.100000
   30004.000000      5.050000      5.999900
]

Mat (Matrix<bool>)c - b is: Matrix 2x3:
[
             0             0             0
             0             0             0
]

size of a: 3 x 3 x 1
============================================================
Mat a.cofactorMatrix(0, 0) is: Matrix 2x2:
[
             5             5
             8             8
]

Mat a.getChannelMat(0) is: Matrix 3x3:
[
             1             2             3
             4             5             5
             7             8             8
]

Mat ((Matrix<double>)a ^ -3) is: Matrix 3x3:
[
     22.777778     -91.851852      49.851852
    -40.222222     159.148148     -86.148148
     20.666667     -80.444444      43.444444
]

Mat a.det() is: -3
Mat a.inv() is: Matrix 3x3:
[
     -0.000000      -2.666667       1.666667
     -1.000000       4.333333      -2.333333
      1.000000      -2.000000       1.000000
]

Mat a.inv() * a is: Matrix 3x3:
[
      1.000000      -0.000000      -0.000000
      0.000000       1.000000       0.000000
      0.000000       0.000000       1.000000
]

Mat a.transpose() is: Matrix 3x3:
```

```
[
                1               4               7
                2               5               8
                3               5               8
]

Mat a.rotate90() is: Matrix 3x3:
[
                3               5               8
                2               5               8
                1               4               7
]
```

Part 4 - Difficulties & Solutions

问题：在比较矩阵时遇到的问题

- 矩阵比较时应该返回什么?

在使用大于或小于比较矩阵时，由于不同位置可能出现不同结果，因此无法只返回一个数来表示，程序**返回bool类型相同大小的矩阵**，每个位置存储对应元素比较的结果。而如果对比的矩阵大小不一致，虽然没有比较的意义，但考虑到使用等号也应该需要判断不同大小的矩阵，因此比较符号中遇到不同大小的矩阵不报错，应返回0.所以两者会存在一定的矛盾之处，应该返回数值还是矩阵？程序中比较不同大小的矩阵，或比较含数据的矩阵与空矩阵时，返回1x1值为0的bool矩阵，其余情况返回对应大小对应元素比较值构成的矩阵。

- 两矩阵的数据类型不同应该怎么返回?

考虑到不同数据类型的矩阵之间的比较不应该只是返回false，应该比较其具体内容，因此程序将比较中后者隐式转换为与前者相同类的矩阵，再进行比较，此时，通过隐式的强制类型转换存在一定的精度损失，例如在比较整数矩阵与浮点数矩阵时，浮点数矩阵转换为整数矩阵损失了小数部分，使得返回值更容易出现1，而不同的浮点数类型转换后由于内存中最后一位有效数字可能不精确，容易出现两者其实相等，但仅差误差的一位判断为0的情况。因此推荐在使用计算或比较操作符运算时，将最前参与运算的矩阵显式转换为类型最大的矩阵以避免此类问题。

Part 5 - Summary

在本次项目中，矩阵类是否应该设计成模板类，影响着后续比较多的方面，如果不把矩阵类设计为模板类，要么矩阵类名将会变得多种多样，要么通过枚举确定数据类型，也将给判断矩阵类型与转换带入许多复杂的问题。在设计矩阵类模板时，我们更应该考虑到使用的方便性，设计字符串到指定矩阵类的隐式构造方法，设计不同矩阵类间的隐式转换，使使用者使用矩阵时像直接书写基本数据类型一样，做到类型转换无需显式表达或存储为临时变量，减小使用者的代码量。对错误提示而不中断程序，不影响后续无关的正常步骤的执行，同时通过提示的错误函数和所在位置可以反推错误代码的位置；提供了求行列式、逆矩阵、矩阵整数幂等特色方法，可以为使用者提供更多的方便之处。

以上是本次报告的所有内容，感谢您的阅读!