**C++ 语 言 程 序 设 计**

实

验

报

告

实 验 二

姓名： 刘健恒

学号： SZ170320207

班级： 自动化2班

**一 实验项目**

* 1. 编写矩阵类，实现矩阵初始化、求逆、转置、访问等基本功能
  2. 基于运算符重载，实现矩阵的加减乘、输入输出的操作

**二 实验原理**

1. 说明矩阵求逆与行列式的原理与操作方法（给出算法的流程图与简要说明，可参考附件PPT内容）

**矩阵求行列式**：

通过高斯消元法，将矩阵化为上三角矩阵，矩阵对角线元素累乘即为所求的行列式。

**矩阵求逆**：

先判断矩阵是否可求逆，若可求逆则利用高斯-若当消元法

将原矩阵与单位矩阵组合为4x8矩阵，然后通过初等行变换，将左边四列化为对角线矩阵，最后将对角线上元素化一，由单位矩阵变化得的末矩阵即为所求的逆矩阵。

**求行列式流程图**   **求逆矩阵流程图**

开始i=0

开始

N

i < 4

可逆

Y

[i][i] != 0

初始化

4x8矩阵

N Y

交换行且变号

将i列进行下三角化0

下三角

化0

主对角线元素累乘得行列式

上三角

化0

结束

对角线元素化1

提取右侧4x4矩阵为所求逆矩阵

结束

1. 矩阵类程序实现与结果（给出算法的源代码，说明关键代码的操作含义，给出运行结果）

MATRIX\_4x4.cpp

#include "MATRIX\_4.h"

#include <iostream>

#include <cmath>

using namespace std;

Matrix\_4x4::Matrix\_4x4()

{

for (int i=0; i<4; i++)

{

for (int j=0; j<4; j++)

{

if (i==j)

matrix[i][j]=1;

else

matrix[i][j]=0;

}

}

}

Matrix\_4x4::Matrix\_4x4(const double (&mat)[4][4])

{

for (int i=0; i<4; i++)

{

for (int j=0; j<4; j++)

{

matrix[i][j]=mat[i][j];

}

}

}

Matrix\_4x4::Matrix\_4x4(const Matrix\_4x4 &mat)

{

for (int i=0; i<4; i++)

{

for (int j=0; j<4; j++)

{

matrix[i][j]=mat.matrix[i][j];

}

}

}

Matrix\_4x4 operator+ (const Matrix\_4x4 &m1, const Matrix\_4x4 &m2)

{

Matrix\_4x4 m0;

for (int i=0; i<4; i++)

{

for (int j=0; j<4; j++)

{

m0.matrix[i][j]=m1.matrix[i][j] + m2.matrix[i][j];

}

}

return m0;

}

Matrix\_4x4 operator- (const Matrix\_4x4 &m1, const Matrix\_4x4 &m2)

{

Matrix\_4x4 m0;

for (int i=0; i<4; i++)

{

for (int j=0; j<4; j++)

{

m0.matrix[i][j]=m1.matrix[i][j] - m2.matrix[i][j];

}

}

return m0;

}

Matrix\_4x4 operator\* (const Matrix\_4x4 &m1, const Matrix\_4x4 &m2)

{

double m[4][4]= {0};

Matrix\_4x4 m0(m);

for (int i=0; i<4; i++)

{

for (int j=0; j<4; j++)

{

for (int p=0; p<4; p++)

{

m0.matrix[i][j]+=m1.matrix[i][p] \* m2.matrix[p][j];

}

}

}

return m0;

}

double Matrix\_4x4::det()

{

double DET=1;

double temp;

Matrix\_4x4 t(matrix);

Matrix\_4x4 s(matrix);

for (int i=0; i<3; i++)

{

int cnt=1;

while((t[i][i]==0)&&(i+cnt<4))

{

for(int k=i; k<4; k++)

{

temp=t[i][k];

t[i][k]=t[i+cnt][k];

t[i+cnt][k]=(-1)\*temp;

}

cnt++;

}

s = t;

for (int j=i; j<4; j++)

{

for (int p=1; p<4; p++)

{

if((i+p<4)&&(t[i][i]!=0))

{

t[i+p][j]-=t[i][j]\*(s[i+p][i]/t[i][i]);

if(abs(t[i+p][j])<1e-14)

t[i+p][j]=0;

}

}

}

}

for (int i=0; i<4; i++)

DET \*= t.matrix[i][i];

return DET;

}

Matrix\_4x4 Matrix\_4x4::inverse()

{

if (this->det()!=0)

{

double temp;

double n[4][8];

double s[4][8];

//初始化数组

for (int i=0; i<4; i++)

{

for (int j=0; j<8; j++)

{

if(j<4)

n[i][j]=matrix[i][j];

else if (i+4==j)

n[i][j]=1;

else

n[i][j]=0;

}

}

//下三角化0

for (int i=0; i<3; i++)

{

int cnt=1;

//检测是否为0

while((n[i][i]==0)&&(i+cnt<4))

{

for(int k=i; k<8; k++)

{

temp=n[i][k];

n[i][k]=n[i+cnt][k];

n[i+cnt][k]=temp;

}

cnt++;

}

for (int i=0; i<4; i++)

{

for (int j=0; j<8; j++)

{

s[i][j]=n[i][j];

}

}

for (int j=i; j<8; j++)

{

for (int p=1; p<4; p++)

{

if((i+p<4)&&(n[i][i]!=0))

{

n[i+p][j]-=n[i][j]\*(s[i+p][i]/n[i][i]);

if(abs(n[i+p][j])<1e-14)

n[i+p][j]=0;

}

}

}

}

//上三角化0

for (int i=3; i>=0; i--)

{

//复制数组

for (int q=0; q<4; q++)

{

for (int w=0; w<8; w++)

{

s[q][w]=n[q][w];

}

}

for (int j=i; j<8; j++)

{

for (int p=1; p<4; p++)

{

if((i-p>=0)&&(n[i][i]!=0))

{

n[i-p][j]-=n[i][j]\*(s[i-p][i]/n[i][i]);

if(abs(n[i-p][j])<1e-14)

n[i-p][j]=0;

}

}

}

}

//化左为单位矩阵

for (int i=0; i<4; i++)

{

for (int j=0; j<8; j++)

n[i][j]=n[i][j]/s[i][i];

}

Matrix\_4x4 tempIn;

for (int i=0; i<4; i++)

{

for (int j=0; j<4; j++)

{

tempIn[i][j]=n[i][j+4];

}

}

return tempIn;

}

else

{

cout<<"不可逆！"<<endl;

return Matrix\_4x4(matrix);

}

}

Matrix\_4x4 Matrix\_4x4::transpose()

{

Matrix\_4x4 temp;

for (int i=0; i<4; i++)

{

for (int j=0; j<4; j++)

{

temp[j][i] = matrix[i][j];

}

}

return temp;

}

ostream& operator <<(ostream& out\_put, const Matrix\_4x4 &m)

{

out\_put<<endl;

for (int i=0; i<4; i++)

{

for (int j=0; j<4; j++)

{

out\_put<<m.matrix[i][j]<<" ";

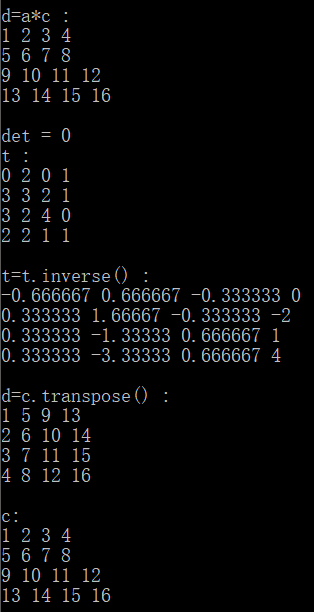
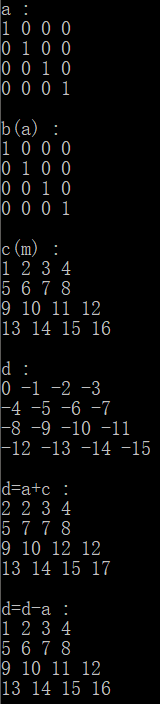
}

out\_put<<endl;

}

return out\_put;

}



**三 实验总结与建议**

（总结实验实施过程，说明实验过程中遇到的问题与解决方案；提出实验环节的建议）

本次矩阵运算的实验，使得我们更深入了解头文件的规范化使用，对数组的运用更深入，并且使用了函数重载，对运算符进行重载，使得日后使用更灵活变通，了解了const修饰防止数据被意外修改的安全措施。