Longqing Chen ID: 22010057

I didn’t have time to finish the ARM part of the assignment, and will be working on it in the future. Here is what I have now (the comments will help clarify my codes)

Codes:

matrix.hpp

#ifndef MATRIX

#define MATRIX

#include <iostream>

using namespace std;

class Matrix

{

private:

    int row;

    int col;

    float \*entry;

    int \* refcount; //cv, counting the number of matrices

public:

    Matrix() {row = 1; col = 1; \*entry = 1.0;}

    Matrix(int r, int c);

    Matrix(int r, int c, float \*e, int \* rv);

    Matrix(const Matrix &mat) {row = mat.row; col = mat.col; entry = mat.entry;} //copying the matrix

    ~Matrix() {release();}

    static inline int CV\_XADD(int\* addr, int delta){int tmp = \* addr; \* addr += delta; return tmp;} //CV\_XADD

    static inline void matmul\_n(size\_t r1, size\_t c, size\_t c2, float \* m1, float \* m2, float \* result); //matrix multiplication

    Matrix &operator = (const Matrix & mat);

    Matrix &operator + (const Matrix & mat);

    Matrix &operator - (const Matrix & mat);

    Matrix &operator \* (const Matrix & mat);

    int GetRow() {return row;}

    int GetCol() {return col;}

    void release();

    friend std::ostream & operator <<(std::ostream &os, const Matrix & other); //cout

    friend Matrix operator \*(double real, const Matrix & mat);//scaling

    friend Matrix operator \*(const Matrix & mat,double real);//also scaling

};

#endif

matrix.cpp

#include <iostream>

#include "matrix.hpp"

using namespace std;

int main()

{

    Matrix mat1(3,5);

    cout << "Matrix 1 is: ";

    cout << mat1;

    cout << endl;

    Matrix mat3 = mat1;

    cout << "Matrix 3 is the same as Matrix 1: ";

    cout << mat3;

    cout << endl;

    Matrix mat7 = mat3 + mat1;

    cout << "Matrix 3 + Matrix 1: ";

    cout << mat7;

    cout << endl;

    Matrix mat2(5,2);

    cout << "Matrix 2 is: ";

    cout << mat2;

    cout << endl;

    Matrix mat4 = mat1\*mat2;

    cout << "Matrix 1 x Matrix 2: ";

    cout << mat4;

    cout << endl;

    Matrix mat5 = 2\*mat1;

    cout << "Scale Matrix 1 by 2: ";

    cout << mat5;

    cout << endl;

    Matrix mat6 = mat1\*3;

    cout << "Scale Matrix 1 by 3: ";

    cout << mat6;

    cout << endl;

    return 0;

}

Matrix & Matrix::operator = (const Matrix & mat)

{

    if(this != & mat)

    {

        if(mat.refcount)

            CV\_XADD(mat.refcount,1);

        release();

        row = mat.row;

        col = mat.col;

        entry = mat.entry;

        return \* this;

    }

    else

    {

        return \*this;

    }

}

Matrix & Matrix::operator \*(const Matrix & mat)

{

    int rownum = this->row;

    int colnum = mat.col;

    Matrix \* mdn = new Matrix(rownum,colnum);

    matmul\_n(rownum,mat.row,colnum,entry,mat.entry,mdn->entry);

    return \*mdn;

}

Matrix & Matrix::operator +(const Matrix & mat)

{

    if (mat.row != this->row)

        exit(1);

    int rownum = this->row;

    int colnum = this->col;

    Matrix \* mpn = new Matrix(rownum,colnum);

    for (int i=0;i<rownum\*colnum;i++)

        mpn -> entry[i] = this->entry[i] + mat.entry[i];

    return \* mpn;

}

Matrix & Matrix::operator -(const Matrix & mat)

{

    if (mat.row != this->row)

        exit(1);

    int rownum = this->row;

    int colnum = this->col;

    Matrix \* mpn = new Matrix(rownum,colnum);

    for (int i=0;i<rownum\*colnum;i++)

        mpn -> entry[i] = this->entry[i] - mat.entry[i];

    return \* mpn;

}

Matrix operator \* (double real, const Matrix & other)

{

    int rownum = other.row;

    int colnum = other.col;

    Matrix \* mxn = new Matrix (rownum,colnum);

    for(int i=0;i<rownum\*colnum;i++)

        mxn -> entry[i] = real\*other.entry[i];

    return \* mxn;

}

Matrix operator \* (const Matrix & other,double real)

{

    int rownum = other.row;

    int colnum = other.col; //I omit the steps that examine whether the multiplication would produce errors.

    Matrix \* mxn = new Matrix (rownum,colnum);

    for(int i=0;i<rownum\*colnum;i++)

        mxn -> entry[i] = real\*other.entry[i];

    return \* mxn;

}

std::ostream & operator << (std::ostream & os, const Matrix & mat)

{

    using namespace std;

    for(int r=0;r<mat.row;r++)

    {

        os << endl;

        for(int c=0;c<mat.col;c++)

            os<<mat.entry[r \* mat.row + c] << " ";

    }

    return os;

}

void Matrix::release()

{

    if (refcount && CV\_XADD(refcount,-1) ==1)

        delete[] entry;

        refcount = 0;

}

Matrix::Matrix(int r, int c)

{

    this -> row = r;

    this -> col = c;

    entry = new float[r\*c];

    refcount = new int;

    \*refcount = 1;

}

Matrix::Matrix(int r, int c, float \*e, int \* rv)

{

    this -> row = r;

    this -> col = c;

    entry = e;

    refcount = rv;

    \* rv = \* rv + 1;

}

inline void Matrix::matmul\_n(size\_t r1, size\_t c, size\_t c2, float \* m1, float \* m2, float \* result)

    {

        float s;

        for (int i=0;i<r1;++i)

        {

            for (int k=0;k<c;++k)

            {

                s=m1[i\*c + k];

                for (int j=0;j<c2;++j)

                {

                    result[i\*c2+j] += s\*m2[k\*c2+j];

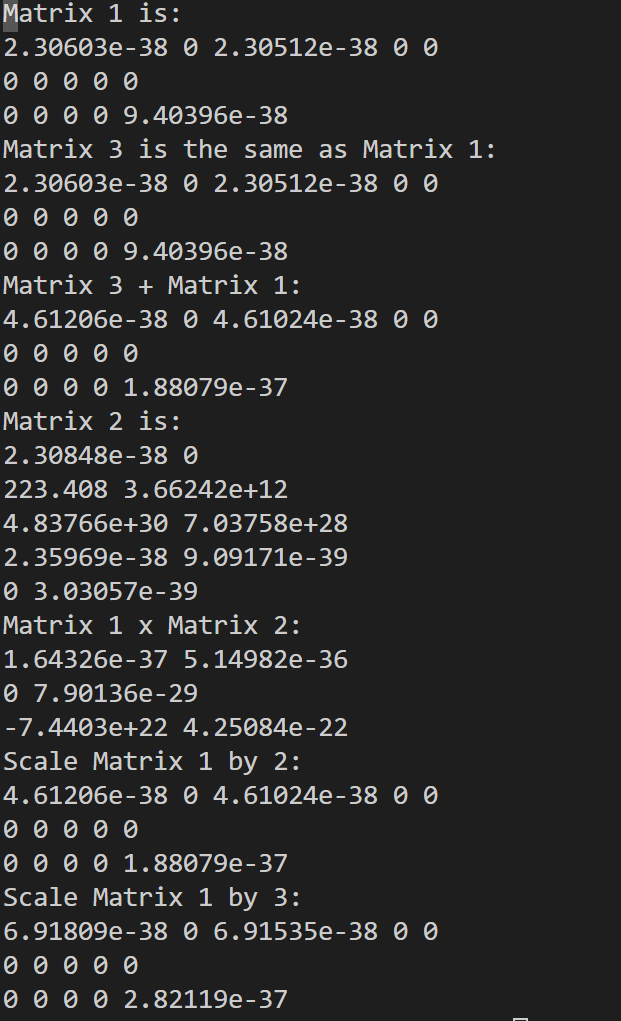
                }

            }

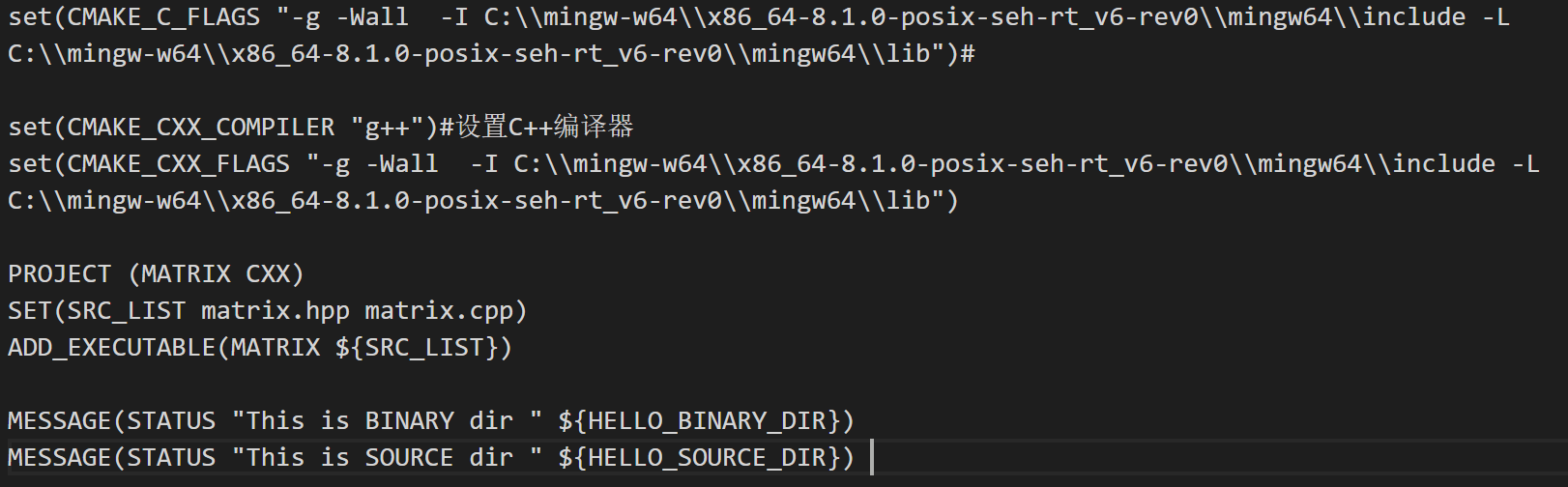
        }

    }

Here are the results:



This is my CMakeLists file:



I executed the following commands in CMD;

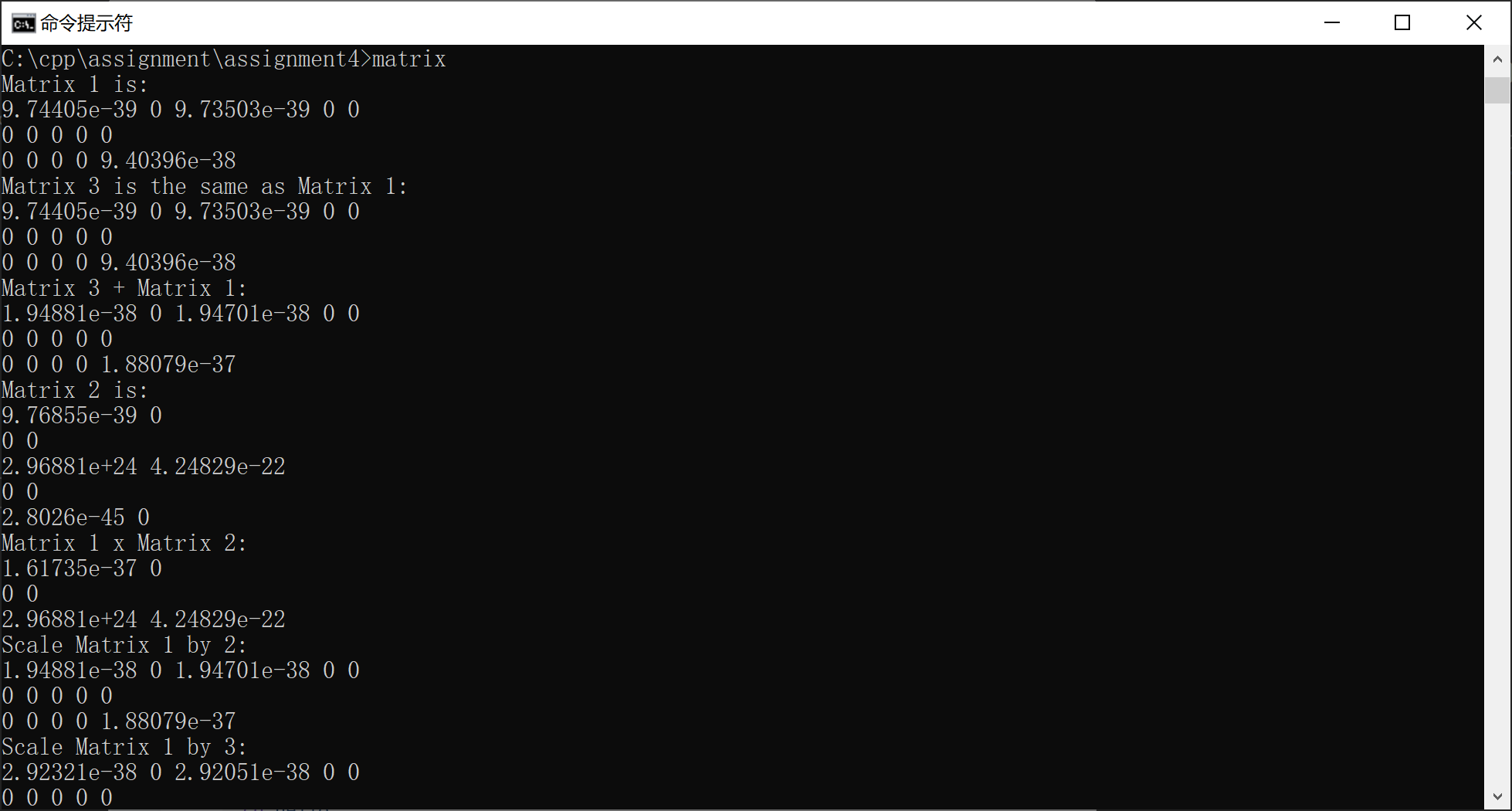
cmake -G “MinGW Makefiles” .

make

And run the program

Matrix

And got the same result:



This is the link to my GitHub where I host this assignment:

<https://github.com/lounachen/cpp/tree/master/assignment/assignment4>