CS205 Assignment 4 Report

SID: 11910718 Name: 陆彦青

Part 1. Introduction & Features

1. In this assignment, a matrix class is declared which is designed for matrix with float elements.
2. The class matrix has various members and functions. Its members include row, column, data and num. Its member functions include constructors, the destructor and a series of operator overloading. In addition, there are functions that can get the **determinant, transpose matrix, inverse matrix** and generate identity matrix.
3. Data copy is not used in copy constructor and "=" overloading. Directly copying the pointer can promote the speed and reduce the waste of memory.
4. This program support MacOS and ARM-based Linux. The user do not need to change any code or compiler parameter. A smart CMake can automatically compile the program on different platform.
5. Almost all the operations of matrix are speeded up by **SIMD** and **multithreading**. The class is robust enough even when matrix with 200M elements is operated.

Part 2. Functions Presentation

The structure of this matrix class is as follows:

```
class matrix {
private:

    size_t num; //元素个数
    size_t row{};
    size_t column{};
    float *data{};
public:
    matrix();
    matrix(size_t, size_t);
    matrix(matrix &);
    ~matrix();

    matrix & operator=(const matrix &);
    matrix & operator+(const matrix &);
    matrix & operator-(const matrix &);
    matrix & operator*(matrix &);
    matrix & operator*(float);
    void operator*=(float);
    void operator*=(matrix &);
    void operator+=(matrix &);
    void operator-=(matrix &);
    float & operator[](size_t);

```

```
    void trans(); //转置矩阵
    double det(); //求行列式
    static matrix & inverse(matrix &); //矩阵求逆
    static matrix & identity(size_t); //生成n阶初等矩阵

    friend bool operator==(const matrix &, const matrix &);
    friend ostream & operator<<(ostream &, matrix &);
    friend istream & operator>>(istream &, matrix &);
    friend matrix & operator*(float, matrix &);

    size_t getRow() const {return row;};
    size_t getColumn() const {return column;};
    float *getData(size_t i) {return &data[i];};
    void setData(size_t i, float v) {data[i] = v;};
};
```

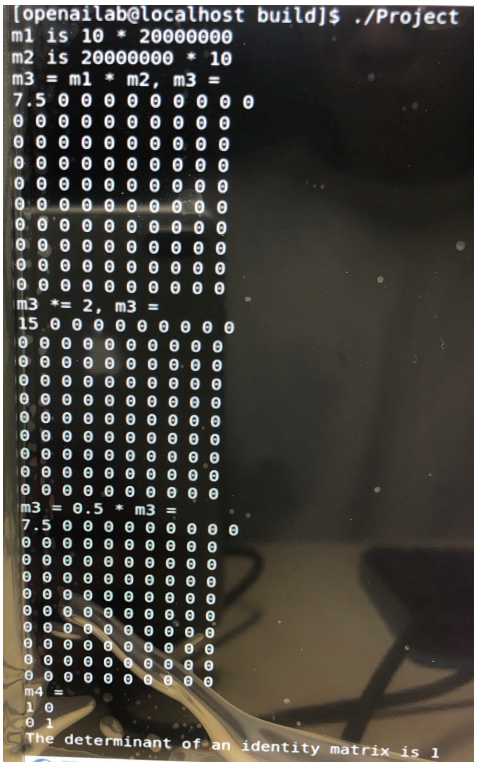
I design a `main()` to test these functions, the result is as follows:

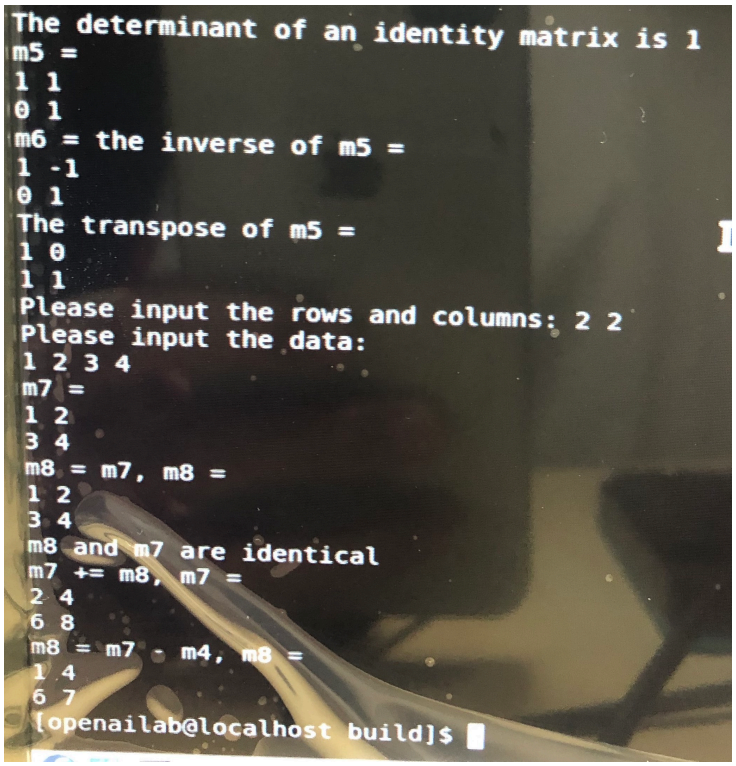
```
m1 is 10 * 20000000
m2 is 20000000 * 10
m3 = m1 * m2, m3 =
7.5 0 0 0 0 0 0 0 0 0
4.5 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
m3 *= 2, m3 =
15 0 0 0 0 0 0 0 0 0
9 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
m3 = 0.5 * m3 =
7.5 0 0 0 0 0 0 0 0 0
4.5 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

```
m4 =
1 0
0 1
The determinant of an identity matrix is 1
m5 =
1 1
0 1
m6 = the inverse of m5 =
1 -1
0 1
The transpose of m5 =
1 0
1 1
Please input the rows and columns: 2 2
Please input the data:
1 2 3 4
m7 =
1 2
3 4
m8 = m7, m8 =
1 2
3 4
m8 and m7 are identical
m7 += m8, m7 =
2 4
6 8
m8 = m7 - m4, m8 =
1 4
6 7
```

You can see the complete codes in the file `main.cpp` which has been uploaded to the GitHub repository.

The same program is also complied and runned in the ARM development board, the result is as follows:





You can see that the result is the same (I have not found any method to install screen-shot app on the development board so I take the photos).

Part 3. Difficulties & Improvement

1. Since only the pointer of data is copied in the copy constructor and “=” overloading, an issue is that the memory of data may be released repeatedly along with the matix. My solution is to add a digit at the end of the data array to mark the number of matrices that use this data. The initial value of this digit is set as 0. The number would add 1 when copy constructor or “=” is called and minus 1 when the destructor is called. If the number equals 0, the pointer would be deleted in the destructor.
2. In the early version, the elements of a inverse matrix that should be 0 may be presented as -0. It is caused by the accuracy loss when float/double are divided. I fix this small issue by adding a ternary operator(三元运算符) to replace every -0 with 0.

Part 4. Efficiency Analysis

In the mid-term project, detailed comparison about different methods has been made. Thus, in this assignment I compare the duration of matrix multiplication in the computer(Intel CPU, x86 architecture) with ARM development board.

1. Computer(CPU: Intel Core i5, 2.4GHz):

Items	1	2	3	4	5	Average	Standard Deviation/Average
Duration / ms	205	107	150	129	114	141	0.56

2. Development board(CPU: ARM 4-core A53, 1.3GHz):

Items	1	2	3	4	5	Average	Standard Deviation/Average
Duration / ms	4488	4446	4502	4517	4482	4487	0.01

The test matrix are 10*2000000 and 2000000*10 respectively. Since the RAM of the ARM development board is only 1 GB, I do not use bigger matrices.

From the result, we can see that computer is over 30 times faster than development board in the situation of computing matrix multiplication. This difference reflects the big performance gap between these 2 devices.

However, the data of development board is more steady than computer. It may because that the utilization of CPU in the computer changes more frequent than the development board because it usually has more background programs.

Part 5. Source Code

All the source codes are included in a GitHub repository: https://github.com/Beauchamp-West/2020Fall_CS205/tree/master/Assignment4