2022141450162 朱骥锋 assignment3

———

2. structured binding

```
template <typename Key, typename Value, typename F>
void update(std::map<Key, Value>& m, F foo) {
  for (auto&& [key value] : m) {
    value = foo(key);
  }
}
```

3. references

```
void swap(int& a,int& b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

4. streams

```
#include <iostream>
#include <string>
#include <fstream>

using namespace std;

struct Student{
    string name;
    int score;
};

int main() {
    int n;
    cout<<"please enter the number of students:";
    cin>> n;
    struct Student student[n];
    for (int i = 0; i < n; ++i) {
        cout<<"name"<<' '<<i+1<<":";
        cin>>student[i].name;
        cout<<"his or her score is:";
        cin>>student[i].score;
    }
    ofstream stu_file("stud.dat");
    if(stu_file.is_open())
    {
        for (int i = 0; i < n; ++i) {
            stu_file<<student[i].name<<' '<<student[i].score<<endl;

        }
    } else{
        cout<<"fail to open the file"<<endl;
        return 1;
    }
    ifstream content_file("stud.dat");
    if (content_file.is_open())
    {
        string name;
        int score;

        while (content_file>>name>>score)
        {
            cout<<"name:"<<name<<" score:"<<score<<endl;
        }
        content_file.close();
    } else{
        cout<<"fail to open the file"<<endl;
        return 1;
    }
    return 0;
}
```

输出样例

```
/tmp/4/cmake-build-debug/4
please enter the number of students:3
name 1:Biden
his or her score is:70
name 2:Trump
his or her score is:90
name 3:Obama
his or her score is:85
name:Biden score:70
name:Trump score:90
name:Obama score:85

进程已结束,退出代码0
```

5. STL

```
#include <iostream>
#include <vector>


int main() {
    std::vector<int> arr;
    std::cout<<"please write 5 integers:";
    for (int i = 0; i < 5; ++i) {
        int temp;
        std::cin>>temp;
```

```
            arr.push_back(temp);
        }
        std::cout<<"下面开始正向迭代器遍历："<<std::endl;
        for (auto i = arr.begin();i< arr.end()  ; ++i ) {
            std::cout<< *i;
        }
        std::cout<<std::endl<<"下面开始反向迭代器遍历："<<std::endl;
        for (auto j = arr.rbegin();j < arr.rend()  ; ++j ) {
            std::cout<< *j;
        }
        return 0;
    }
```

输出结果为：

```
/tmp/5/cmake-build-debug/5
please write 5 integers:0 1 2 3 4
下面开始正向迭代器遍历：
01234
下面开始反向迭代器遍历：
43210
进程已结束,退出代码0
```

6. Linear Algebra library

```
//以下是linearalgebra.cpp中代码
//
// Created by Edward on 2023/4/10.
//
#include "linearalgebra.h"

namespace algebra{
    using Matrix = std::vector<std::vector<double>>;
    Matrix zeros(size_t n, size_t m){
        Matrix zeros1(n,std::vector<double>(m));
        return zeros1;
    }

    Matrix ones(size_t n, size_t m){
        Matrix ones1(n, std::vector<double>(m, 1));
        return ones1;
    }

    Matrix random(size_t n, size_t m, double min, double max){
        std::random_device rd;
        std::mt19937 gen(rd());
        std::uniform_int_distribution<> dis(min, max);
        Matrix  temp(n, std::vector<double>(m));
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < m; ++j) {
                temp[i][j] = dis(gen);
            }
        }
        return temp;
    }

    void show(const Matrix& matrix){
        for (int i = 0; i < matrix.size(); ++i) {
            for (int j = 0; j < matrix[i].size(); ++j) {
                std::cout << std::fixed << std::setprecision(3) <<
std::setw(7) << matrix[i][j] << " ";
                std::cout<<std::endl;
            }
        }
    }

    Matrix multiply(const Matrix& matrix, double c){
        Matrix matrix1 = matrix;
        for (int i = 0; i < matrix.size(); ++i) {
            for (int j = 0; j < matrix[i].size(); ++j) {
                matrix1[i][j] *= c;
            }
        }
        return matrix1;
    }

    Matrix multiply(const Matrix& matrix1, const Matrix& matrix2){
        if(matrix1.empty() || matrix2.empty())
        {
            return Matrix();
        }
        if(matrix1[0].size() != matrix2.size())
        {
            return Matrix();
        }
        Matrix matrix3(matrix1.size(), std::vector<double>
(matrix2[0].size()));
        for (int i = 0; i < matrix1.size(); ++i) {
            for (int j = 0; j < matrix2[0].size(); ++j) {
                for (int k = 0; k < matrix1[0].size(); ++k) {
                    matrix3[i][j] += matrix1[i][k] * matrix2[k][j];
                }
            }
        }
        return matrix3;

    }

    Matrix sum(const Matrix& matrix, double c){
        Matrix sum1 = matrix;
        for (int i = 0; i < matrix.size(); ++i) {
            for (int j = 0; j < matrix[i].size(); ++j) {
                sum1[i][j] += c;
            }
        }
        return sum1;

    }

    Matrix sum(const Matrix& matrix1, const Matrix& matrix2){
        Matrix sum(matrix1.size(),std::vector<double>(matrix1[0].size()));
        if (matrix1.empty() && !matrix2.empty() || !matrix1.empty() && matrix2.empty()){
            throw std::logic_error("两个矩阵不能相加");
        } else if (matrix1.size()!=matrix2.size() || matrix2[0].size()!=matrix1[0].size()){
            throw std::logic_error("两个矩阵不能相加");
        }else{
            for (int i = 0; i < matrix1.size(); ++i) {
                for (int j = 0; j < matrix1[i].size(); ++j) {
                        sum[i][j] = matrix1[i][j] + matrix2[i][j];
                }
            }
        }
        return sum;

    }

    Matrix transpose(const Matrix& matrix){
        Matrix new_matrix = matrix;
        for (int i = 0; i < matrix.size(); ++i) {
            for (int j = 0; j < matrix[0].size(); ++j) {
                new_matrix[i][j] = matrix[j][i];
            }
        }
        return new_matrix;

    }

    Matrix minor(const Matrix& matrix,size_t n,size_t m){
        if(matrix.size()==0){
            return matrix;
        }
        Matrix minor_matrix=zeros(matrix.size()-1,matrix[0].size()-1);
        for(int i=0;i<matrix.size();i++)
            for(int j=0;j<matrix[0].size();j++){
                if(i==n||j==m){
                    continue;
                }
                minor_matrix[(i>n)?i-1:i][(j>m)?j-1:j]=matrix[i][j];
            }
        return minor_matrix;
    }

    double determinant(const Matrix& matrix) {
        if (matrix.size() == 1) {
            return matrix[0][0];
        } else if (matrix.size() == 2) {
            return matrix[0][0]*matrix[1][1] - matrix[0][1]*matrix[1][0];
        } else {
            double det = 0;
            for (int i = 0; i < matrix.size(); i++) {
                Matrix matrix_1 = minor(matrix, 0, i);
                double sign = (i % 2 == 0) ? 1 : -1;
                double matrix_1_det = sign * determinant(matrix_1);
                det += matrix[0][i] * matrix_1_det;
            }
            return det;
        }
    }

    Matrix inverse(const Matrix& matrix) {
        int n = matrix.size();
        Matrix aug(2 * n, std::vector<double>(2 * n));

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                aug[i][j] = matrix[i][j];
            }
            aug[i][i + n] = 1;
        }
        for (int i = 0; i < n; i++) {
            double pivot = aug[i][i];
            for (int j = 0; j < 2 * n; j++) {
                aug[i][j] /= pivot;
            }
            for (int j = 0; j < n; j++) {
                if (i != j) {
                    double factor = aug[j][i];
                    for (int k = 0; k < 2 * n; k++) {
                        aug[j][k] -= factor * aug[i][k];
                    }
                }
            }
        }
        Matrix inv(n, std::vector<double>(n));
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                inv[i][j] = aug[i][j + n];
            }
        }

        return inv;
    }

    Matrix concatenate(const Matrix& matrix1, const Matrix& matrix2, int axis) {
        int n1 = matrix1.size(), n2 = matrix2.size();
        int m1 = matrix1[0].size(), m2 = matrix2[0].size();
        Matrix new_matrix;
        if (axis == 0) {
            new_matrix = Matrix(n1 + n2, std::vector<double>(m1));
            for (int i = 0; i < n1; i++) {
                for (int j = 0; j < m1; j++) {
                    new_matrix[i][j] = matrix1[i][j];
                }
            }
            for (int i = 0; i < n2; i++) {
                for (int j = 0; j < m1; j++) {
                    new_matrix[i + n1][j] = matrix2[i][j];
                }
            }
        } else if (axis == 1) {
            new_matrix = Matrix(n1, std::vector<double>(m1 + m2));
            for (int i = 0; i < n1; i++) {
                for (int j = 0; j < m1; j++) {
                    new_matrix[i][j] = matrix1[i][j];
                }
                for (int j = 0; j < m2; j++) {
                    new_matrix[i][j + m1] = matrix2[i][j];
                }
            }
        }
        return new_matrix;
    }

    Matrix ero_swap(const Matrix& matrix, size_t r1, size_t r2){
        if (r1 < matrix.size() && r2 < matrix.size() && r1 >= 0 && r2 >= 0)
{
            algebra::Matrix swap_matrix = matrix;
            std::swap(swap_matrix[r1],swap_matrix[r2]);
            return swap_matrix;
        } else{
            throw std::logic_error("越界");
        }

    }

    Matrix ero_multiply(const Matrix& matrix, size_t r, double c){
        if (r < matrix.size()) {
            algebra::Matrix new_matrix = matrix;

            for (int j = 0; j < matrix[0].size(); ++j) {
                new_matrix[r][j] *= c;
            }

            return new_matrix;
        } else{
            throw std::logic_error("越界");
        }
    }

    Matrix ero_sum(const Matrix& matrix, size_t r1, double c, size_t r2){
        algebra::Matrix sum = matrix;
        if (r1 < matrix.size() && r2 < matrix.size() && r1 >= 0 && r2 >= 0)
{
            for (int i = 0; i < matrix[0].size(); i++) {
                sum[r2][i] += sum[r1][i] * c;
            }
            return sum;
        } else{
            throw std::logic_error("越界");}
    }

    Matrix upper_triangular(const Matrix& matrix){
        algebra::Matrix trgul = matrix;
        int n = matrix.size();
        for (int i = 0; i < n; i++) {
            int pivot_row = i;
            double pivot = trgul[i][i];
            for (int j = i + 1; j < n; j++) {
                if (abs(trgul[j][i]) > abs(pivot)) {
                    pivot_row = j;
                    pivot = trgul[j][i];
                }
            }
            if (pivot_row != i) {
                swap(trgul[i], trgul[pivot_row]);
            }

            for (int j = i + 1; j < n; j++) {
                double factor = trgul[j][i] / trgul[i][i];
                for (int k = i; k < n; k++) {
                    trgul[j][k] -= factor * trgul[i][k];
                }
            }
        }
        return trgul;
    }


}
```

```
//以下是linearalgebra.h代码
//
// Created by Edward on 2023/4/10.
//

#ifndef INC_6_LINEARALGEBRA_H
#define INC_6_LINEARALGEBRA_H
#include <iostream>
#include <vector>
#include <iomanip>
#include <random>
#include <algorithm>

using Matrix = std::vector<std::vector<double>>;
namespace algebra{
    Matrix zeros(size_t n, size_t m);
    Matrix ones(size_t n, size_t m);
    Matrix random(size_t n, size_t m, double min, double max);
    Matrix multiply(const Matrix& matrix, double c);
    Matrix multiply(const Matrix& matrix1, const Matrix& matrix2);
    Matrix sum(const Matrix& matrix, double c);
    Matrix sum(const Matrix& matrix1, const Matrix& matrix2);
    Matrix transpose(const Matrix& matrix);
    Matrix minor(const Matrix& matrix, size_t n, size_t m);
    double determinant(const Matrix& matrix);
    Matrix inverse(const Matrix& matrix);
    Matrix concatenate(const Matrix& matrix1, const Matrix& matrix2, int axis = 0 );
    Matrix ero_swap(const Matrix& matrix, size_t r1, size_t r2);
    Matrix ero_multiply(const Matrix& matrix, size_t r, double c);
    Matrix ero_sum(const Matrix& matrix, size_t r1, double c, size_t r2);
    void show(const Matrix& matrix);

}

#endif //INC_6_LINEARALGEBRA_H
```

```
Microsoft Windows [版本 10.0.22621.1413]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Edward>docker ps
CONTAINER ID   IMAGE                      COMMAND       CREATED       STATUS
   PORTS     NAMES
331f065bf5d5   scucpphw_test_img:latest   "/bin/bash"   2 weeks ago   Up 13
hours   22/tcp    frosty_wozniak

C:\Users\Edward> docker exec -it 331f065bf5d5 /bin/bash
root@331f065bf5d5:/# cd /ws
root@331f065bf5d5:/ws# cd LinearAlgebra
root@331f065bf5d5:/ws/LinearAlgebra# mkdir build
mkdir: cannot create directory 'build': File exists
root@331f065bf5d5:/ws/LinearAlgebra# cd build
root@331f065bf5d5:/ws/LinearAlgebra/build# cmake ..
-- Configuring done
-- Generating done
-- Build files have been written to: /ws/LinearAlgebra/build
root@331f065bf5d5:/ws/LinearAlgebra/build# make
Scanning dependencies of target main
[ 25%] Building CXX object CMakeFiles/main.dir/src/main.cpp.o
[ 50%] Building CXX object CMakeFiles/main.dir/src/linearalgebra.cpp.o
[ 75%] Building CXX object CMakeFiles/main.dir/src/unit_test.cpp.o
[100%] Linking CXX executable main
[100%] Built target main
root@331f065bf5d5:/ws/LinearAlgebra/build# ./main
RUNNING TESTS ...
[==========] Running 24 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 24 tests from LinearAlgebraTest
[ RUN ] LinearAlgebraTest.ZEROS
[ OK ] LinearAlgebraTest.ZEROS (0 ms)
[ RUN ] LinearAlgebraTest.ONES
[ OK ] LinearAlgebraTest.ONES (0 ms)
[ RUN ] LinearAlgebraTest.RANDOM1
random matrix [-5, 7)
[1.351 5.473 2.712 -3.406 ]
[4.551 0.999 -2.760 3.132 ]
[-2.704 5.518 1.233 2.779 ]
[5.497 -3.345 2.917 -3.365 ]

[ OK ] LinearAlgebraTest.RANDOM1 (0 ms)
[ RUN ] LinearAlgebraTest.RANDOM2
[ OK ] LinearAlgebraTest.RANDOM2 (0 ms)
[ RUN ] LinearAlgebraTest.MULTIPLY1
[ OK ] LinearAlgebraTest.MULTIPLY1 (0 ms)
[ RUN ] LinearAlgebraTest.MULTIPLY2
Matrix is empty
[ OK ] LinearAlgebraTest.MULTIPLY2 (0 ms)
[ RUN ] LinearAlgebraTest.MULTIPLY3
[ OK ] LinearAlgebraTest.MULTIPLY3 (0 ms)
[ RUN ] LinearAlgebraTest.MULTIPLY4
[ OK ] LinearAlgebraTest.MULTIPLY4 (0 ms)
[ RUN ] LinearAlgebraTest.SUM1
[ OK ] LinearAlgebraTest.SUM1 (0 ms)
[ RUN ] LinearAlgebraTest.SUM2
[ OK ] LinearAlgebraTest.SUM2 (0 ms)
[ RUN ] LinearAlgebraTest.TRANSPOSE
[ OK ] LinearAlgebraTest.TRANSPOSE (0 ms)
[ RUN ] LinearAlgebraTest.MINOR1
[ OK ] LinearAlgebraTest.MINOR1 (0 ms)
[ RUN ] LinearAlgebraTest.MINOR2
[ OK ] LinearAlgebraTest.MINOR2 (0 ms)
[ RUN ] LinearAlgebraTest.DETERMINANT1
[ OK ] LinearAlgebraTest.DETERMINANT1 (1 ms)
[ RUN ] LinearAlgebraTest.DETERMINANT2
[ OK ] LinearAlgebraTest.DETERMINANT2 (0 ms)
[ RUN ] LinearAlgebraTest.INVERSE1
Empty matrix
[ OK ] LinearAlgebraTest.INVERSE1 (0 ms)
[ RUN ] LinearAlgebraTest.INVERSE2
[ OK ] LinearAlgebraTest.INVERSE2 (0 ms)
[ RUN ] LinearAlgebraTest.CONCATENATE1
[ OK ] LinearAlgebraTest.CONCATENATE1 (0 ms)
[ RUN ] LinearAlgebraTest.CONCATENATE2
[ OK ] LinearAlgebraTest.CONCATENATE2 (0 ms)
[ RUN ] LinearAlgebraTest.ERO_SWAP
[ OK ] LinearAlgebraTest.ERO_SWAP (0 ms)
[ RUN ] LinearAlgebraTest.ERO_MULTIPLY
[ OK ] LinearAlgebraTest.ERO_MULTIPLY (0 ms)
[ RUN ] LinearAlgebraTest.ERO_SUM
[ OK ] LinearAlgebraTest.ERO_SUM (0 ms)
[ RUN ] LinearAlgebraTest.UPPER_TRIANGULAR1
[ OK ] LinearAlgebraTest.UPPER_TRIANGULAR1 (0 ms)
[ RUN ] LinearAlgebraTest.BONUS
[ OK ] LinearAlgebraTest.BONUS (0 ms)
[----------] 24 tests from LinearAlgebraTest (2 ms total)

[----------] Global test environment tear-down
[==========] 24 tests from 1 test suite ran. (2 ms total)
[ PASSED ] 24 tests.
<<<SUCCESS>>>
```