Student Name: 杨卓
Student ID: 2022141450295

高级语言程序设计-II
Assignment 3

1 结构化绑定 Structured Binding

使用结构化绑定在一行内实现这个函数。

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

这个函数实现了传入一个 `std::map` 与 `F`，使用 `F` 类型的 `foo` 将 `std::map` 中的值和键值一一对应。根据主函数的内容，这里使用 `std::hash` 将值变为字符串的 Hash 值。

建议写清楚函数的目的，不然读懂有点困难。:)

2 引用 References

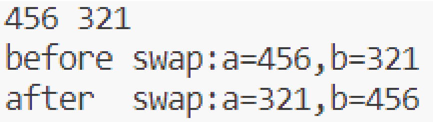
使用引用实现两个整数的交换。

```cpp
#include <iostream>
#include <cstdio>

template<typename T>
void swap(T& a, T& b) {
    T c = a;
    a = b;
    b = c;
}

int main() {
    int a, b;
    std::cin >> a >> b;
    std::cout << "before swap:";
    std::cout << "a=" << a << "," << "b=" << b << std::endl;
    swap(a, b);
    std::cout << "after  swap:";
    std::cout << "a=" << a << "," << "b=" << b << std::endl;
    return 0;
}
```

虽然要求是实现两个整数的交换，这里使用模板，可以实现任意两个相同类型的变量交换值。测试样例与结果如下：



```
456 321
before swap:a=456,b=321
after  swap:a=321,b=456
```

图 1: test_reference

如果只是实现整数之间的交换，除此之外还有另一种节省空间的写法：

```cpp
#include <iostream>
#include <cstdio>

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

int main() {
    int a, b;
    std::cin >> a >> b;
    std::cout << "before swap:";
    std::cout << "a=" << a << "," << "b=" << b << std::endl;
    swap(a, b);
    std::cout << "after  swap:";
    std::cout << "a=" << a << "," << "b=" << b << std::endl;
    return 0;
}
```

具体原理是利用异或的性质。

$$
\begin{aligned}
a &= a \text{ xor } b \\
b &= a \text{ xor } b \\
a &= a \text{ xor } b
\end{aligned}
$$

第二行 a xor b xor $b = a$，第三行 a xor b xor $a = b$。

3 流 Streams

主程序如下：

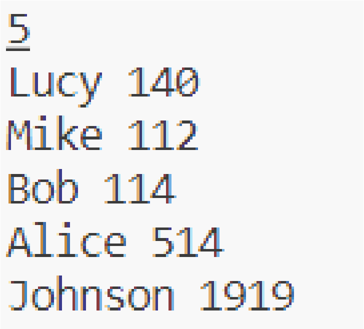
```cpp
#include <iostream>
#include <fstream>
#include <cstring>
```

```cpp
#include <vector>

// 用于存储学生信息
struct Student {
    std::string name;
    double score;
};

std::vector<Student> stu;

int main() {
    int n;
    std::cin >> n;
    for (int i = 1; i <= n; ++i) {
        std::string name;
        double scr;
        std::cin >> name >> scr;
        stu.push_back({name, scr});
    }
    std::ofstream fout("stu.dat");
    for (auto y : stu) {
        fout << y.name << " "<< y.score << std::endl;
    }
    return 0;
}
```



```
5
Lucy 140
Mike 112
Bob 114
Alice 514
Johnson 1919
```

图 2: 测试输入

输出文件的内容:

```
Lucy 140
Mike 112
Bob 114
Alice 514
Johnson 1919
```

4 STL 容器 Containers

代码如下：

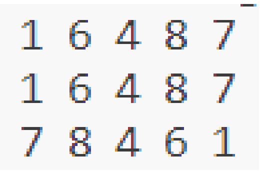
```
#include <iostream>
#include <cstdio>
#include <vector>

std::vector<int> vec;

int main() {
    for (int i = 0; i < 5; ++i) {
        int x;
        std::cin >> x;
        vec.push_back(x);
    }
    // 正向遍历
    for (auto it = vec.begin(); it != vec.end(); ++it) {
        std::cout << *it << " ";
    }
    std::cout << std::endl;
    // 反向遍历
    for (auto it = vec.rbegin(); it != vec.rend(); ++it) {
        std::cout << *it << " ";
    }
    std::cout << std::endl;
    return 0;
}
```

对于 std::vector 来说，begin() 和 end() 返回的是正向迭代器的开始和结束，而 rbegin() 和 rend() 表示的是反向迭代器的开始和结束，这两者不能混用。

自行构造数据进行测试：



```
1 6 4 8 7
1 6 4 8 7
7 8 4 6 1
```

图 3: 测试结果

5 线性代数库 Linear Algebra library

由于实现的部分过于长，不便于在此处全部列出，这里只列出头文件内容与部分内容。

```
// linearalgebra.h
```

```
#ifndef LINEARALGEBRA_H
#define LINEARALGEBRA_H

#include <vector>

using std::size_t;

namespace algebra {

using Matrix = std::vector<std::vector<double>>;
const double EPS = 1e-6;
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);
void show(const Matrix& matrix);// 3 decimal numbers
Matrix multiply(const Matrix& matrix1, const Matrix& matrix2);
Matrix multiply(const Matrix& matrix, double c);
Matrix sum(const Matrix& matrix1, 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);
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);
Matrix upper_triangular(const Matrix& matrix);

}

#endif //LINEARALGEBRA_H
```

随机部分使用 STL 中 random 部分。

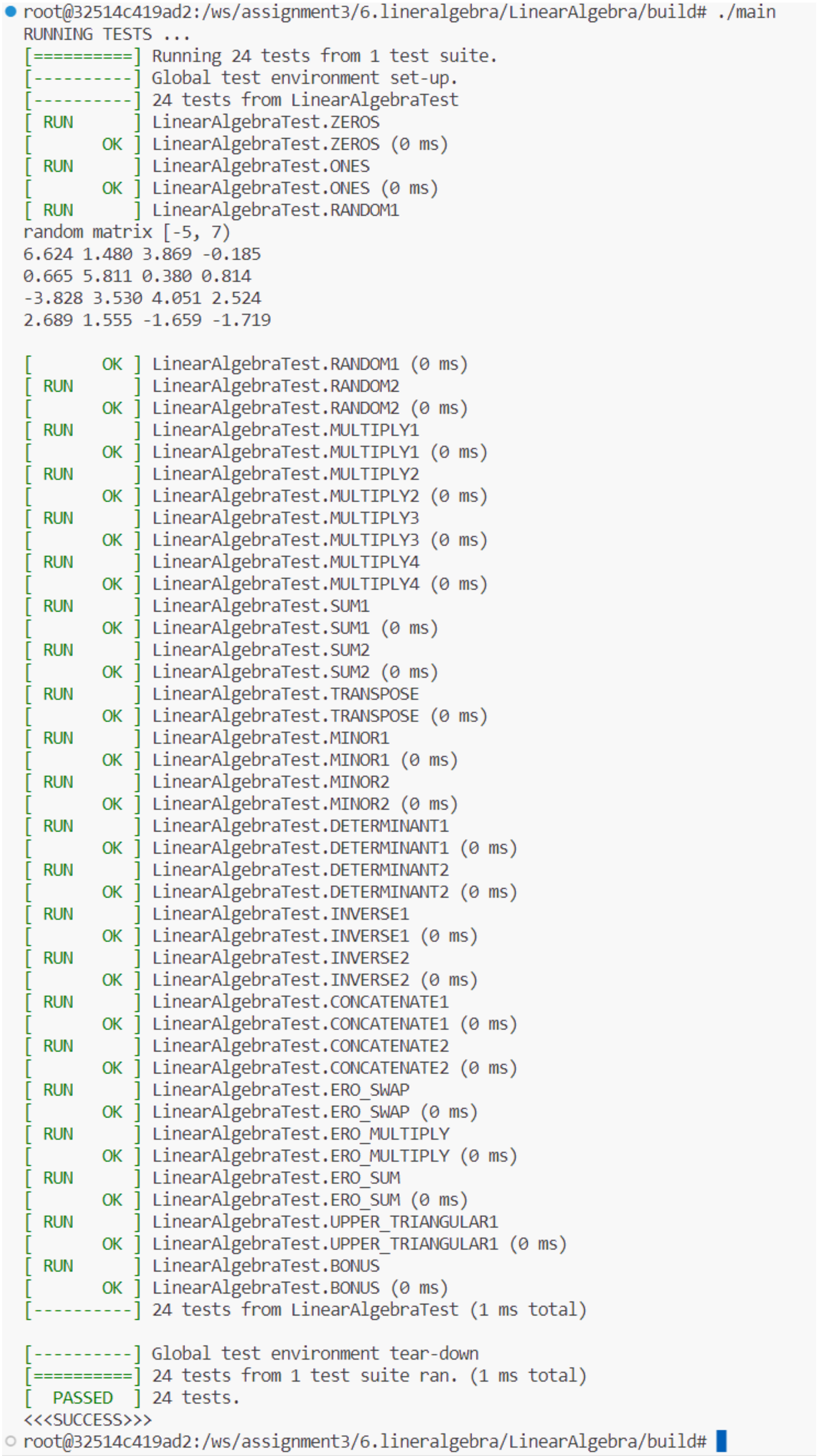
```
Matrix random(size_t n, size_t m, double min, double max) {
    if (max < min) {
        throw std::logic_error("max is less then min!");
    }
    // 抛出错误
    std::default_random_engine rg; // 定义随机引擎
    rg.seed(std::random_device()()); // 设置随机种子
    std::uniform_real_distribution<double> dist(min, max); // 定义均匀的实数分布
    Matrix mat(n);
    for (int i = 0; i < n; ++i) {
        mat[i] = std::vector<double>(m);
        for (int j = 0; j < m; ++j) {
            mat[i][j] = dist(rg); // 生成随机数
        }
    }
    return mat;
}
```

在消元成上三角的过程中，当我们选取的这一行的主元为 0 的时候，可以继续向下一行选择，直到当前元素不为 0，就将那一行与当前行交换，然后按照正常的步骤消元。如果不存在这样的行，就说明这个矩阵并不满秩，此时需要从下一列继续寻找。

具体实现则是通过两个变量，一个变量 i 表示当前的主元的行数，另一个变量 j 表示当前主元的列数。每次从第 i 行开始寻找第一个第 j 列非零的元素，假设在第 k 行，那么当 k!=i 时，交换这两行。之后用第 i 行第 j 列的元素将下面的元素消成 0。如果不存在 k，那么令 j++，即在下一列继续这样找。

```
Matrix upper_triangular(const Matrix& matrix) {
    if (matrix.size() == 0) return {};
    int n = matrix.size();
    int m = matrix[0].size();
    if (n != m) {
        throw std::logic_error("non-square!");
    }
    Matrix res = matrix;
    for (int i = 0, j = 0; i < n;) {
        int k = i;
        while (k < n && fabs(res[k][j]) < EPS) ++k;
        if (k == n) {
            k = 0;
            ++j;
            continue;
        }
        if (i != k) res = ero_swap(res, i, k);
        for (int p = i+1; p < n; ++p) {
            res = ero_sum(res, i, -res[p][j]/res[i][j], p);
        }
        ++i;
        ++j;
    }
    return res;
}
```

最后是通过运行的截图：



```
root@32514c419ad2:/ws/assignment3/6.lineralgebra/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)
6.624 1.480 3.869 -0.185
0.665 5.811 0.380 0.814
-3.828 3.530 4.051 2.524
2.689 1.555 -1.659 -1.719

[       OK ] LinearAlgebraTest.RANDOM1 (0 ms)
[ RUN      ] LinearAlgebraTest.RANDOM2
[       OK ] LinearAlgebraTest.RANDOM2 (0 ms)
[ RUN      ] LinearAlgebraTest.MULTIPLY1
[       OK ] LinearAlgebraTest.MULTIPLY1 (0 ms)
[ RUN      ] LinearAlgebraTest.MULTIPLY2
[       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 (0 ms)
[ RUN      ] LinearAlgebraTest.DETERMINANT2
[       OK ] LinearAlgebraTest.DETERMINANT2 (0 ms)
[ RUN      ] LinearAlgebraTest.INVERSE1
[       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 (1 ms total)

[----------] Global test environment tear-down
[==========] 24 tests from 1 test suite ran. (1 ms total)
[  PASSED  ] 24 tests.
<<<SUCCESS>>>
root@32514c419ad2:/ws/assignment3/6.lineralgebra/LinearAlgebra/build#
```

图 4: result