



## c++ Assignment 3

---

### 1 Structured Binding

---

```
1  #include <iostream>
2  #include <map>
3  #include <string>
4  #include <functional>
5
6  template<typename Key, typename Value, typename F>
7  void update(std::map<Key, Value> &m, F foo) {
8  // TODO
9      for (auto &&[key, value]: m) value = foo(key);
10 }
11
12 int main() {
13     std::map<std::string, long long int> m{
14         {"a", 1},
15         {"b", 2},
16         {"c", 3}
17     };
18     update(m, [](std::string key) {
19         return std::hash<std::string>{}(key);
20     });
21     for (auto &&[key, value]: m)
22         std::cout << key << ":" << value << std::endl;
23 }
```

该函数将遍历 m 里面的每一个键值对，并把 key 改为希哈值  
添加的一行代码，同时实现了结构绑定和范围循环，并更改 key 值

### 2 References

---

```
1  #include <iostream>
2  void change(int &a, int &b) {
3      int temp = a;
4      a = b;
5      b = temp;
```

```

6  }
7  int main() {
8      int a;
9      int b;
10     std::cin >> a >> b;
11     change(a, b);
12     std::cout << a << " " << b << std::endl;
13     return 0;
14 }

```

---

```

1  2 3
2  3 2
3
4  进程已结束,退出代码0

```

---

## 3 Stream

### 库函数的实现

```

1  #include <iostream>
2  #include <fstream>
3  #include <string>
4
5      using namespace std;
6
7      struct Student {
8          string name;
9          int score{};
10     };
11     int main() {
12         // 读入学生信息并写入文件
13         ofstream fout("stud.dat", ios::binary);
14         if (!fout) {
15             cerr << "Failed to open file" << endl;
16             return 1;
17         }
18         int n;
19         cout << "Enter number of students: ";
20         cin >> n;
21         Student stu;
22         for (int i = 0; i < n; ++i) {
23             cout << "Enter name and score for student " << i + 1 << ": ";
24             cin >> stu.name >> stu.score;
25             fout.write((char*)&stu, sizeof(stu));
26         }

```

```

27     fout.close();
28     // 从文件中读取学生信息并显示
29     ifstream fin("stud.dat", ios::binary);
30     if (!fin) {
31         cerr << "Failed to open file" << endl;
32         return 1;
33     }
34     while (fin.read((char*)&stu, sizeof(stu))) {
35         cout << "Name: " << stu.name << ", Score: " << stu.score << endl;
36     }
37
38     fin.close();
39
40     return 0;
41 }

```

---

```

1  /tmp/Streams/cmake-build-debug/Streams
2  Enter number of students: 3
3  Enter name and score for student 1: xiaoming 10
4  Enter name and score for student 2: xiaohong 9
5  Enter name and score for student 3: xiaohuang 8
6  Name: xiaoming, Score: 10
7  Name: xiaohong, Score: 9
8  Name: xiaohuang, Score: 8
9
10 进程已结束,退出代码0

```

---

## 4 STL(Containers)

---

```

1  #include <iostream>
2  #include <vector>
3  int main() {
4      std::vector<int> x;
5      std::cout << "输入五个整型\n";
6      for(int i = 0; i < 5; i++){
7          int temp;
8          std::cin >> temp;
9          x.push_back(temp);
10     }
11     std::cout << "正向迭代器遍历\n";
12     for(auto it = x.begin(); it != x.end(); it++ )
13     {
14         std::cout << *it;
15     }

```

```

16     std::cout << std::endl;
17     std::cout << "反向迭代器遍历\n";
18     for(auto it = x.rbegin(); it != x.rend(); it++ ) {
19         std::cout << *it;
20     }
21     return 0;
22 }

```

---

```

1  /tmp/STL(Containers)/cmake-build-debug/STL_Containers_
2  输入五个整型
3  0 1 2 3 4
4  正向迭代器遍历
5  01234
6  反向迭代器遍历
7  43210
8  进程已结束,退出代码0

```

---

## 5 Linear Algebra library

---

```

1  #ifndef LINEARALGEBRA_H
2  #define LINEARALGEBRA_H
3  #include <iostream>
4  #include <vector>
5  #include <iomanip>
6  #include <algorithm>
7  #include <random>
8  using Matrix = std::vector<std::vector<double>>>;
9  namespace algebra{
10     Matrix zeros(size_t n, size_t m);
11     Matrix ones(size_t n, size_t m);
12     Matrix random(size_t n, size_t m, double min, double max);
13     void show(const Matrix& matrix);
14     Matrix multiply(const Matrix& matrix, double c);
15     Matrix multiply(const Matrix& matrix1, const Matrix& matrix2);
16     Matrix sum(const Matrix& matrix, double c);
17     Matrix sum(const Matrix& matrix1, const Matrix& matrix2);
18     Matrix transpose(const Matrix& matrix);
19     Matrix minor(const Matrix& matrix, size_t n, size_t m);
20     double determinant(const Matrix& matrix);
21     Matrix inverse(const Matrix& matrix);
22     Matrix concatenate(const Matrix& matrix1, const Matrix& matrix2, int axis = 0 );
23     Matrix ero_swap(const Matrix& matrix, size_t r1, size_t r2);
24     Matrix ero_multiply(const Matrix& matrix, size_t r, double c);
25     Matrix ero_sum(const Matrix& matrix, size_t r1, double c, size_t r2);

```

```

26     Matrix upper_triangular(const Matrix& matrix);
27 }
28 #endif //LINEARALGEBRA_H

```

---

```

1  #include "linearalgebra.h"
2  using Matrix = std::vector<std::vector<double>>>;
3
4  namespace algebra {
5      using Matrix = std::vector<std::vector<double>>>;
6      Matrix zeros(size_t n, size_t m) {
7          Matrix zeros(n, std::vector<double>(m));
8          return zeros;
9      }
10
11     Matrix ones(size_t n, size_t m) {
12         Matrix ones(n, std::vector<double>(m, 1));
13         return ones;
14     }
15
16     Matrix Unit(size_t n) {
17         Matrix unit(n, std::vector<double>(n));
18         for (int i = 0; i < unit.size(); i++) {
19             unit[i][i] = 1;
20         }
21         return unit;
22     }
23
24     Matrix random(size_t n, size_t m, double min, double max) {
25         if (max > min) {
26             std::random_device rd;//创建随机数引擎
27             std::mt19937 gen(rd());
28             std::uniform_real_distribution<> dis(min, max);
29             Matrix random(n, std::vector<double>(m));
30             for (int i = 0; i < n; i++)
31                 for (int j = 0; j < m; j++)
32                     random[i][j] = dis(gen);
33             return random;
34         } else {
35             throw std::logic_error("min >= max");
36         }
37     }
38 }
39
40 void show(const Matrix &matrix) {
41     for (int i = 0; i < matrix.size(); i++) {
42         std::cout << "[";

```

```

43         for (int j = 0; j < matrix[0].size(); j++) {
44             std::cout << std::fixed << std::left << std::setw(10) << std::setprecision(3) << matrix[i
               ][j];
45         }
46         std::cout << "]" << std::endl;
47     }
48 }
49
50 Matrix multiply(const Matrix &matrix, double c) {
51     Matrix matrix_answer = matrix;
52     for (int i = 0; i < matrix.size(); i++)
53
54         for (int j = 0; j < matrix[i].size(); j++)
55             matrix_answer[i][j] *= c;
56     return matrix_answer;
57 }
58
59 Matrix multiply(const Matrix &matrix1, const Matrix &matrix2) {
60     //矩阵可成的前提是matrix1的列 == matrix2的行数
61
62     if (matrix1.empty() || matrix2.empty()) {
63         std::cout << "Matrix is empty" << std::endl;
64         return zeros(0, 0);
65     }
66     int m1 = matrix1.size(), n1 = matrix1[0].size();
67     int m2 = matrix2.size(), n2 = matrix2[0].size();
68     if (n1 == m2) {
69         int m = m1, n = n2;
70         Matrix matrix_answer(m, std::vector<double>(n));
71         for (int i = 0; i < m; i++) {
72             for (int j = 0; j < n; j++) {
73                 for (int k = 0; k < m2; k++) {
74                     matrix_answer[i][j] += matrix1[i][k] * matrix2[k][j];
75                 }
76             }
77         }
78         return matrix_answer;
79     } else {
80         throw std::logic_error("Matrix type does not match, not multiplyable");
81
82     }
83
84 }
85
86 Matrix sum(const Matrix &matrix, double c) {
87     Matrix matrix_answer = matrix;

```

```

88     for (int i = 0; i < matrix.size(); i++)
89
90         for (int j = 0; j < matrix[i].size(); j++)
91             matrix_answer[i][j] += c;
92     return matrix_answer;
93 }
94
95 Matrix sum(const Matrix &matrix1, const Matrix &matrix2) {
96     if (matrix1.empty() && matrix2.empty()) {
97         return zeros(0, 0);
98     } else if (matrix1.empty() && !matrix2.empty() || !matrix1.empty() && matrix2.empty()) {
99         throw std::logic_error("Matrix type does not match");
100     }
101     int m1 = matrix1.size(), n1 = matrix1[0].size();
102     int m2 = matrix2.size(), n2 = matrix2[0].size();
103     if (m1 == m2 && n1 == n2) {
104         Matrix matrix_answer(m1, std::vector<double>(n1));
105         for (int i = 0; i < m1; i++) {
106             for (int j = 0; j < n1; j++) {
107                 matrix_answer[i][j] = matrix1[i][j] + matrix2[i][j];
108             }
109         }
110         return matrix_answer;
111     } else {
112         throw std::logic_error("Matrix type does not match");
113     }
114 }
115
116 Matrix transpose(const Matrix &matrix) {
117     if (matrix.empty())
118         return zeros(0, 0);
119     else {
120         Matrix transpose(matrix[0].size(), std::vector<double>(matrix.size()));
121         for (int i = 0; i < transpose.size(); i++)
122             for (int j = 0; j < transpose[i].size(); j++) {
123                 transpose[i][j] = matrix[j][i];
124             }
125         return transpose;
126     }
127 }
128
129 Matrix minor(const Matrix &matrix, size_t n, size_t m) {
130     if (n < matrix.size() && m < matrix[0].size()) {
131         Matrix minor = matrix;
132         minor.erase(minor.begin() + n);
133         for (auto &i: minor)

```

```

134         i.erase(i.begin() + m);
135         return minor;
136     } else {
137         std::cout << "Out of range" << std::endl;
138         return zeros(0, 0);
139     }
140 }
141
142 double determinant(const Matrix &matrix) {
143     //判断是否为空矩阵
144     if (matrix.size() == 0 || matrix[0].size() == 0)
145         return 1;
146     //判断是否是方阵
147     else if (matrix.size() != matrix[0].size()) {
148         throw std::logic_error("Matrix is not square");
149     } else {
150         //一阶矩阵返回
151         if (matrix.size() == 1) {
152             return matrix[0][0];
153         } else {
154             double answer = 0;
155             for (int i = 0; i < matrix.size(); i++)
156                 answer += pow(-1, i + 1 + 1) * matrix[i][0] * determinant(minor(matrix, i, 0));
157             return answer;
158         }
159     }
160 }
161
162 //交换两行
163 Matrix ero_swap(const Matrix &matrix, size_t r1, size_t r2) {
164     if (r1 < matrix.size() && r2 < matrix.size() && r1 >= 0 && r2 >= 0) {
165         Matrix answer = matrix;
166         std::swap(answer[r1], answer[r2]);
167         return answer;
168     } else {
169
170         throw std::logic_error("Out of range");
171     }
172 }
173
174 //倍增
175 Matrix ero_multiply(const Matrix &matrix, size_t r, double c) {
176     if (r < matrix.size()) {
177         Matrix answer = matrix;
178         for (int i = 0; i < matrix[i].size(); i++) {
179             answer[r][i] *= c;

```



```

180         }
181         return answer;
182     } else {
183         std::cout << "Out of range" << std::endl;
184         return zeros(0, 0);
185     }
186 }
187
188 //倍加
189 Matrix ero_sum(const Matrix &matrix, size_t r1, double c, size_t r2) {
190     Matrix answer = matrix;
191     if (r1 < matrix.size() && r2 < matrix.size() && r1 >= 0 && r2 >= 0) {
192         for (int i = 0; i < matrix[0].size(); i++) {
193             answer[r2][i] += answer[r1][i] * c;
194         }
195     }
196     return answer;
197 }
198
199 //上三角
200 Matrix upper_triangular(const Matrix &matrix) {
201     if (0 == matrix.size()) {
202         return zeros(0, 0);
203     } else if (matrix.size() == matrix[0].size()) {
204         Matrix answer = matrix;
205         for (int i = 0; i < matrix[0].size() - 1; i++) {
206             if(answer[i][i] == 0)
207             {
208                 for (int j = i + 1; j < matrix.size(); j++) {
209                     if (answer[j][i] != 0) {
210                         answer = ero_swap(answer, i, j);
211                         break;
212                     }
213                 }
214             }
215             for (int j = i + 1; j < matrix.size(); j++) {
216                 answer = ero_sum(answer, i, -answer[j][i] / answer[i][i], j);
217             }
218         }
219         return answer;
220     } else {
221         throw std::logic_error("Matrix is not square");
222     }
223 }
224
225 //矩阵求逆

```

```

226 Matrix inverse(const Matrix &matrix) {
227     //行列式存在且不等于0,并且不是空矩阵
228     if (matrix.size() == 0 || matrix[0].size() == 0) {
229         std::cout << "Empty matrix" << std::endl;
230         return zeros(0, 0);
231     } else if(determinant(matrix) == 0) {
232         throw std::logic_error("non_singular_matrix");
233     } else {
234         //求出上三角矩阵
235         Matrix unit_to_inverse = Unit(matrix.size());
236         Matrix matrix_to_unit = matrix;
237         for (int i = 0; i < matrix[0].size() - 1; i++) {
238             if(matrix_to_unit[i][i] == 0)
239             {
240                 for (int j = i + 1; j < matrix.size(); j++) {
241                     if (matrix_to_unit[j][i] != 0) {
242                         matrix_to_unit = ero_swap(matrix_to_unit, i, j);
243                         unit_to_inverse = ero_swap(unit_to_inverse, i, j);
244                         break;
245                     }
246                 }
247             }
248             for (int j = i + 1; j < matrix.size(); j++) {
249                 unit_to_inverse = ero_sum(unit_to_inverse, i, -matrix_to_unit[j][i] / matrix_to_unit
250                     [i][i], j);
251                 matrix_to_unit = ero_sum(matrix_to_unit, i, -matrix_to_unit[j][i] / matrix_to_unit[i
252                     ][i], j);
253             }
254         }
255         for (int i = 1; i < matrix_to_unit.size(); i++) {
256             if(matrix_to_unit[i][i] == 0)
257             {
258                 for (int j = i + 1; j < matrix.size(); j++) {
259                     if (matrix_to_unit[j][i] != 0) {
260                         matrix_to_unit = ero_swap(matrix_to_unit, i, j);
261                         unit_to_inverse = ero_swap(unit_to_inverse, i, j);
262                         break;
263                     }
264                 }
265             }
266             for (int j = 0; j < i; j++) {
267                 unit_to_inverse = ero_sum(unit_to_inverse, i, -matrix_to_unit[j][i] / matrix_to_unit
268                     [i][i], j);
269                 matrix_to_unit = ero_sum(matrix_to_unit, i, -matrix_to_unit[j][i] / matrix_to_unit[i
270                     ][i], j);
271             }
272         }
273     }
274 }

```

```

268     }
269     for (int i = 0; i < matrix_to_unit.size(); i++) {
270         for(int j = 0; j < matrix_to_unit[0].size(); j++){
271             unit_to_inverse[i][j] /= matrix_to_unit[i][i];
272             matrix_to_unit[i][j] /= matrix_to_unit[i][i];
273         }
274     }
275     return unit_to_inverse;
276 }
277 }
278 //矩阵结合
279 Matrix concatenate(const Matrix &matrix1, const Matrix &matrix2, int axis) {
280     if (!axis && matrix1[0].size() == matrix2[0].size()) {
281         Matrix concatenate(matrix1.size() + matrix2.size(), std::vector<double>(matrix2[0].size()));
282         for (int i = 0; i < matrix1.size() + matrix2.size(); i++) {
283             concatenate[i] = i < matrix1.size() ? matrix1[i] : matrix2[i - matrix1.size()];
284         }
285         return concatenate;
286     } else if (!axis && matrix1[0].size() != matrix2[0].size()) {
287         throw std::logic_error("Matrix size not match");
288     } else if (axis && matrix1.size() == matrix2.size()) {
289         Matrix concatenate(matrix1.size(), std::vector<double>(matrix1[0].size() + matrix2[0].size()
290             ));
291         for (int j = 0; j < matrix1[0].size() + matrix2[0].size(); j++)
292             for (int i = 0; i < matrix1.size(); i++) {
293                 concatenate[i][j] = j < matrix1[0].size() ? matrix1[i][j] : matrix2[i][j - matrix1
294                     ][0].size());
295             }
296         return concatenate;
297     } else {
298         throw std::logic_error("Matrix size not match");
299     }
300 }

```

---

## 编译运行

---

```

1 root@fa571cbced78:/ws# cd /ws/LinearAlgebra
2 root@fa571cbced78:/ws/LinearAlgebra# mkdir build
3 root@fa571cbced78:/ws/LinearAlgebra# cd build
4 root@fa571cbced78:/ws/LinearAlgebra/build# cmake..
5 bash: cmake...: command not found
6 root@fa571cbced78:/ws/LinearAlgebra/build# cmake ..
7 -- The C compiler identification is GNU 11.2.0
8 -- The CXX compiler identification is GNU 11.2.0
9 -- Detecting C compiler ABI info
10 -- Detecting C compiler ABI info - done

```

```

11 -- Check for working C compiler: /usr/bin/cc - skipped
12 -- Detecting C compile features
13 -- Detecting C compile features - done
14 -- Detecting CXX compiler ABI info
15 -- Detecting CXX compiler ABI info - done
16 -- Check for working CXX compiler: /usr/local/bin/c++ - skipped
17 -- Detecting CXX compile features
18 -- Detecting CXX compile features - done
19 -- Found GTest: /usr/local/lib/libgtest.a
20 -- Configuring done
21 -- Generating done
22 -- Build files have been written to: /ws/LinearAlgebra/build
23 root@fa571cbced78:/ws/LinearAlgebra/build# make
24 Scanning dependencies of target main
25 [ 25%] Building CXX object CMakeFiles/main.dir/src/main.cpp.o
26 [ 50%] Building CXX object CMakeFiles/main.dir/src/linearalgebra.cpp.o
27 [ 75%] Building CXX object CMakeFiles/main.dir/src/unit_test.cpp.o
28 [100%] Linking CXX executable main
29 [100%] Built target main
30 root@fa571cbced78:/ws/LinearAlgebra/build# ls
31 CMakeCache.txt CMakeFiles Makefile cmake_install.cmake main
32 root@fa571cbced78:/ws/LinearAlgebra/build# ./main
33 RUNNING TESTS ...
34 [=====] Running 24 tests from 1 test suite.
35 [-----] Global test environment set-up.
36 [-----] 24 tests from LinearAlgebraTest
37 [ RUN      ] LinearAlgebraTest.ZEROS
38 [      OK  ] LinearAlgebraTest.ZEROS (0 ms)
39 [ RUN      ] LinearAlgebraTest.ONES
40 [      OK  ] LinearAlgebraTest.ONES (0 ms)
41 [ RUN      ] LinearAlgebraTest.RANDOM1
42 random matrix [-5, 7)
43 [1.351      5.473      2.712      -3.406      ]
44 [4.551      0.999      -2.760      3.132      ]
45 [-2.704      5.518      1.233      2.779      ]
46 [5.497      -3.345      2.917      -3.365      ]
47
48 [      OK  ] LinearAlgebraTest.RANDOM1 (0 ms)
49 [ RUN      ] LinearAlgebraTest.RANDOM2
50 [      OK  ] LinearAlgebraTest.RANDOM2 (0 ms)
51 [ RUN      ] LinearAlgebraTest.MULTIPLY1
52 [      OK  ] LinearAlgebraTest.MULTIPLY1 (0 ms)
53 [ RUN      ] LinearAlgebraTest.MULTIPLY2
54 Matrix is empty
55 [      OK  ] LinearAlgebraTest.MULTIPLY2 (0 ms)
56 [ RUN      ] LinearAlgebraTest.MULTIPLY3

```

```

57 [      OK ] LinearAlgebraTest.MULTIPLY3 (0 ms)
58 [ RUN      ] LinearAlgebraTest.MULTIPLY4
59 [      OK ] LinearAlgebraTest.MULTIPLY4 (0 ms)
60 [ RUN      ] LinearAlgebraTest.SUM1
61 [      OK ] LinearAlgebraTest.SUM1 (0 ms)
62 [ RUN      ] LinearAlgebraTest.SUM2
63 [      OK ] LinearAlgebraTest.SUM2 (0 ms)
64 [ RUN      ] LinearAlgebraTest.TRANSPOSE
65 [      OK ] LinearAlgebraTest.TRANSPOSE (0 ms)
66 [ RUN      ] LinearAlgebraTest.MINOR1
67 [      OK ] LinearAlgebraTest.MINOR1 (0 ms)
68 [ RUN      ] LinearAlgebraTest.MINOR2
69 [      OK ] LinearAlgebraTest.MINOR2 (0 ms)
70 [ RUN      ] LinearAlgebraTest.DETERMINANT1
71 [      OK ] LinearAlgebraTest.DETERMINANT1 (1 ms)
72 [ RUN      ] LinearAlgebraTest.DETERMINANT2
73 [      OK ] LinearAlgebraTest.DETERMINANT2 (0 ms)
74 [ RUN      ] LinearAlgebraTest.INVERSE1
75 Empty matrix
76 [      OK ] LinearAlgebraTest.INVERSE1 (0 ms)
77 [ RUN      ] LinearAlgebraTest.INVERSE2
78 [      OK ] LinearAlgebraTest.INVERSE2 (0 ms)
79 [ RUN      ] LinearAlgebraTest.CONCATENATE1
80 [      OK ] LinearAlgebraTest.CONCATENATE1 (0 ms)
81 [ RUN      ] LinearAlgebraTest.CONCATENATE2
82 [      OK ] LinearAlgebraTest.CONCATENATE2 (0 ms)
83 [ RUN      ] LinearAlgebraTest.ERO_SWAP
84 [      OK ] LinearAlgebraTest.ERO_SWAP (0 ms)
85 [ RUN      ] LinearAlgebraTest.ERO_MULTIPLY
86 [      OK ] LinearAlgebraTest.ERO_MULTIPLY (0 ms)
87 [ RUN      ] LinearAlgebraTest.ERO_SUM
88 [      OK ] LinearAlgebraTest.ERO_SUM (0 ms)
89 [ RUN      ] LinearAlgebraTest.UPPER_TRIANGULAR1
90 [      OK ] LinearAlgebraTest.UPPER_TRIANGULAR1 (0 ms)
91 [ RUN      ] LinearAlgebraTest.BONUS
92 [      OK ] LinearAlgebraTest.BONUS (0 ms)
93 [-----] 24 tests from LinearAlgebraTest (2 ms total)
94
95 [-----] Global test environment tear-down
96 [=====] 24 tests from 1 test suite ran. (2 ms total)
97 [ PASSED ] 24 tests.
98 <<<SUCCESS>>>

```

---