

```
1  #include <iostream>
2  #include <list>
3  #include <vector>
4  using namespace std;
5
6
7
8  template <class T> ostream& operator<<(ostream& str, const vector<T>& V);
9  template <class T> ostream& operator<<(ostream& str, const vector<T*>& V);
10 template <class T> ostream& operator<<(ostream& str, const list<T>& L);
11 template <class T> ostream& operator<<(ostream& str, const list<T*>& L);
12 void DB1_to_DB2(vector<list<int>>& DB1, vector<list<int*>>& DB2);
13 void DB2_to_DB3(vector<list<int*>>& DB2, list<vector<int*>>& DB3);
14
15
16
17
18
19
20 int main() {
21
22     vector<list<int>> DB1{ {1,2,3}, {4,5}, {6,7,8,9} };
23
24     vector<list<int*>> DB2;
25     list<vector<int*>> DB3;
26
27     DB1_to_DB2(DB1, DB2);
28     cout << "DB1: " << DB1 << endl;
29     cout << "DB2: " << DB2 << endl;
30
31     DB2_to_DB3(DB2, DB3);
32     cout << "DB2: " << DB2 << endl;
33     cout << "DB3: " << DB3 << endl;
34
35
36
37     system("pause");
38     return 0;
39 }
40
41
42 void DB2_to_DB3(vector<list<int*>>& DB2, list<vector<int*>>& DB3) {
43     // 1. delete the current DB3
44     for (auto it : DB3) {
45         it->clear();
46         delete it;
47     }
48     DB3.clear();
49     // 2. loop build new DB3 with same value as DB2 (loop in DB2)
50     for (auto it : DB2) {
51         vector<int*> pv{ new vector<int> };
52         for (auto itv : *it) {
```

```
53         pv->push_back(*itv);
54     }
55     DB3.push_back(pv);
56 }
57 }
58
59
60
61
62 void DB1_to_DB2(vector<list<int>>& DB1, vector<list<int*>*>& DB2) { // call by ↗
    reference '&' !!!!
63     // 1. delete the current DB2
64     for (auto it : DB2) {
65         for (auto iti : *it) {
66             delete iti;
67         }
68         it->clear();
69         delete it;
70     }
71     DB2.clear();
72
73     // 2.loop build new DB2 with same value as DB1 (loop in DB1)
74     for (auto it : DB1) {
75         list<int*>* p1{ new list<int*> }; // synchronously building DB2's layer ↗
76         element
77         for (auto it1 : it) {
78             int* pi{ new int(it1) };
79             p1->push_back(pi);
80         }
81         DB2.push_back(p1);
82     }
83
84
85
86
87
88
89 template <class T> ostream& operator<<(ostream& str, const vector<T>& V) {
90     str << "[vector: ";
91     for (auto& i : V) { str << i << " "; }
92     str << "];";
93     return str;
94 }
95
96 template <class T> ostream& operator<<(ostream& str, const vector<T*>& V) {
97
98     str << "[vector: ";
99     for (auto& i : V) { str << *i << " "; }
100     str << "];";
101     return str;
102 }
```

```
103
104 template <class T> ostream& operator<<(ostream& str, const list<T>& L) {
105
106     str << "<list: ";
107     for (auto& i : L) { str << i << " "; }
108     str << ">";
109     return str;
110
111 }
112
113 template <class T> ostream& operator<<(ostream& str, const list<T*>& L) {
114
115     str << "<list: ";
116     for (auto& i : L) { str << *i << " "; }
117     str << ">";
118     return str;
119
120 }
```