11.1答：map根据关键字来读取和存储元素。vector通过元素在容器中位置来存取元素。

11.3答：#include<iostream>

#include<map>

#include<string>

using namespace std;

int main()

{

map<string, size\_t> word\_count;

string word;

while (cin >> word)

word\_count[word]++;

for (auto e : word\_count)

cout << e.first << " occurs" << e.second << ((e.second > 1) ? " tiems" : " time") << endl;

return 0;

}

11.2答：list 类型适用于需要在容器的中间位置插入和删除元素的情况，如以无序的方式读入一系列学生的数据；

vector 类型适用于需要随机访问元素的情况。如：在学号为1…n的学生中，访问第 x 学号的学生的信息。

deque 类型适用于在容器的尾部或首部有插入和删除元素情况。如：对服务窗口先来先服务的情况。

map适用于需要 key – value 对的集合的情况。如：字典电和话簿的建立和使用。

set类型适用于使用键集合的情况。例如，黑名单的建立和使用。

11.4答：#include<iostream>

#include<map>

#include<string>

#include<algorithm>

using namespace std;

int main()

{

map<string, size\_t> word\_map;

string word;

while (cin >> word)

{

for (auto i = 0; i != word.size(); ++i)

word[i] = tolower(word[i]);

auto ptr = std::find\_if(word.begin(), word.end(), [](const char c) { return c == '.' || c == ','; });

if (ptr != word.end())

word.erase(ptr);

++word\_map[word];

}

for (auto e : word\_map)

cout << e.first << ' ' << e.second << endl;

return 0;

}

11.5答：map必须指明关键字类型和值的类型，而定义一个set只需指明关键字的类型

11.6答：set包含不重复的元素，list则不关心元素的是否重复

11.7答：#include <iostream>

#include <vector>

#include <map>

#include <string>

#include <algorithm>

#include <list>

#include <set>

#include <utility>

using namespace std;

int main()

{

map<string, vector<string>> famls;

string fam\_name, chil\_name;

while (cin >> fam\_name)

{

while (cin >> chil\_name)

famls[fam\_name].push\_back(chil\_name);

}

for (auto e : famls)

{

cout << e.first << " ";

for (auto v : e.second)

cout << v << " ";

cout << endl;

}

}

11.8答：#include<iostream>

#include<vector>

#include<map>

#include<string>

using namespace std;

int main()

{

vector<string> exclude = { "chen", "xun", "is" };

string word;

while (cin >> word)

{

for (const auto &e : exclude)

{

if (e == word)

cout << "excluded!" << endl;

}

}

return 0;

}

11.9答：map<std::string, std::list<std::size\_t>>

11.10答：list不支持<操作

11.11答：#include<iostream>

#include<vector>

#include<map>

#include<string>

#include"my\_Sales\_data.h"

#include <set>

using namespace std;

bool comapareIsbn(const Sales\_data &s1, const Sales\_data &s2)

{

return s1.isbn() < s2.isbn();

}

int main()

{

bool(\*fp) (const Sales\_data &s1, const Sales\_data &s2) = comapareIsbn;

multiset<const Sales\_data, bool(\*) (const Sales\_data&, const Sales\_data&)> m(fp);

return 0;

}

11.12答：std::vector<std::pair<std::string, int>> v;如下题所示

11.13答：#include <iostream>

#include <vector>

#include <map>

#include <string>

#include <algorithm>

#include <list>

#include <set>

#include <utility>

using namespace std;

int main()

{

int i;

string word;

pair<std::string, int> pair1;

vector< pair<string, int>> v;

while (cin>>word&&cin>>i)

{

//pair1 = pair<string, int>(word, i);

//pair1 = { word, i };

pair1 = make\_pair(word, i);

v.push\_back(pair1);

}

for (auto e : v)

cout << e.first << " " << e.second << " "<< "\n";

return 0;

}

11.14答：#include <iostream>

#include <vector>

#include <map>

#include <string>

#include <algorithm>

#include <list>

#include <set>

#include <utility>

using namespace std;

int main()

{

map<string, vector<string,string>> famls;

string fam\_name, chil\_name, birthday;

while (cin >> fam\_name)

{

while (cin >> chil\_name&&cin >> birthday)

famls[fam\_name].push\_back({ chil\_name, birthday });

}

for (auto e : famls)

{

cout << e.first << " ";

for (auto v : e.second)

cout << v << " ";

cout << endl;

}

}

11.15答：//map<int, vector<int>>

//mappped\_type : vector<int>;

//key\_type : int;

//value\_type : pair<int, vector<int>>;

11.16答：#include <iostream>

#include <vector>

#include <map>

#include <string>

#include <algorithm>

#include <list>

#include <set>

#include <utility>

using namespace std;

int main()

{

std::map<int, std::string> m;

m[100] = "chen";

map<int, string>::iterator it = m.begin();

it->second = "xun";

cout << it->second;

return 0;

}

11.17答：copy(c.begin(), c.end(), back\_inserter(v));

11.18答#include <iostream>

#include <vector>

#include <map>

#include <string>

#include <algorithm>

#include <list>

#include <set>

#include <utility>

using namespace std;

int main()

{

map<string, size\_t> word\_count;

map<string, size\_t>::const\_iterator map\_it = word\_count.cbegin();

while (map\_it != word\_count.cend())

{

cout << map\_it->first << " occurs "<< map\_it->second << " times" << endl;

++map\_it;

}

return 0;

}

11.19答：#include <iostream>

#include <vector>

#include <map>

#include <string>

#include <algorithm>

#include <list>

#include <set>

#include <utility>

#include "my\_Sales\_data.h"

using namespace std;

bool comapareIsbn(const Sales\_data &s1, const Sales\_data &s2)

{

return s1.isbn() < s2.isbn();

}

int main()

{

bool(\*fp) (const Sales\_data &s1, const Sales\_data &s2) = comapareIsbn;

multiset<const Sales\_data, bool(\*) (const Sales\_data&, const Sales\_data&)> m(fp);

multiset<const Sales\_data, bool(\*) (const const Sales\_data &, const Sales\_data &)>::iterator begin = m.begin();

return 0;

}

11.20答：#include <iostream>

#include <vector>

#include <map>

#include <string>

#include <algorithm>

#include <list>

#include <set>

#include <utility>

using namespace std;

int main()

{

map<string, size\_t> word\_count;

string word;

while (cin >> word)

{

//auto it = word\_count.insert({ word, 1 });

pair<map<string, size\_t>::iterator, bool>it = word\_count.insert({ word, 1 });

if (!it.second)

++it.first->second;

}

for (auto e : word\_count)

cout << e.first << " " << e.second << endl;

}

11.21#include <iostream>

#include <vector>

#include <map>

#include <string>

#include <algorithm>

#include <list>

#include <set>

#include <utility>

using namespace std;

int main()

{

map<string, size\_t> word\_count;

string word;

while (cin >> word)

{

//auto it = word\_count.insert({ word, 1 });

//pair<map<string, size\_t>::iterator, bool>it = word\_count.insert({ word, 1 });

++word\_count.insert({ word, 0 }).first->second;

//if (!it.second)

// ++it.first->second;

}

for (auto e : word\_count)

cout << e.first << " " << e.second << endl;

}

11.22答：#include <iostream>

#include <vector>

#include <map>

#include <string>

#include <algorithm>

#include <list>

#include <set>

#include <utility>

using namespace std;

int main()

{

std::map<std::string, std::vector<int>> m;

std::pair<std::string, std::vector<int>> pair1{ "chen", { 1, 2, 3, 4, 5, 6 } };

std::pair<std::map<std::string, std::vector<int>>::iterator, bool> it;

it = m.insert(pair1);

for (const auto e : m)

{

std::cout << e.first << endl;

for (const auto &l : e.second)

std::cout << l << endl;

std::cout << endl;

}

return 0;

}

11.23答：#include <iostream>

#include <vector>

#include <map>

#include <string>

#include <algorithm>

#include <list>

#include <set>

#include <utility>

using namespace std;

int main()

{

multimap<string, vector<string>> famls;

string faml\_name, child\_name;

vector<string> v\_child\_name;

while (cin >> faml\_name)

{

while (cin >> child\_name)

v\_child\_name.push\_back(child\_name);

}

famls.insert({ faml\_name, v\_child\_name });

for (auto e : famls)

{

cout << e.first << " ";

for (auto s : child\_name)

cout << s << " ";

cout << endl;

}

return 0;

}

11.24 25 26答：#include <iostream>

#include <vector>

#include <map>

#include <string>

#include <algorithm>

#include <list>

#include <set>

#include <utility>

using namespace std;

int main()

{

std::map<int, int> m;

m[0] = 1;

for (const auto e : m)

std::cout << e.first << " " << e.second << endl;

vector<int> v;

v[0] = 1;

for (const auto e : v)

std::cout << e << endl;

map<int, string> my\_map = { { 1, "chen" }, { 2, "xun" } };

map<int, string>::key\_type my\_key= 1;

map<int, string>::mapped\_type my\_value = my\_map[my\_key];

return 0;

}

11.27答：如果统计元素的个数使用count，如果查找元素使用find

11.28答：返回一个迭代器指向查找的关键字对应的vector。

11.29答：若果关键字不在容器中，upper\_bound指向和lower\_bound返回相等的迭代器指向给定关键字的插入点。equal\_rang返回

11.30答：匹配元素最后一个元素之后的位置。

11.31-32-答：#include <iostream>

#include <vector>

#include <map>

#include <string>

#include <algorithm>

#include <list>

#include <set>

#include <utility>

using namespace std;

int main()

{

multimap<string, string> m = { { "chen", "1" }, { "chen", "2" }, { "chen", "3" }, { "chen", "4" } };

for (auto e : m)

cout << e.first << ' ' << e.second << endl;

cout << "-------------------------------" << endl;

multimap<string, string>::iterator it = m.find("chen");//返回的第一个匹配元素的迭代器

if (it != m.end())

m.erase(it);

for (auto e : m)

cout << e.first << ' ' << e.second << endl;

}

11.33答：#include <iostream>

#include <vector>

#include <map>

#include <string>

#include <algorithm>

#include <list>

#include <set>

#include <utility>

#include <fstream>

#include <sstream>

using namespace std;

void my\_word\_transform(const string &map\_file, const string &input\_file)

{

ifstream fin(map\_file);

string line;

map<string, string> trans\_map;

while (getline(fin, line))

{

string key, value;

key = line.substr(0, line.find(' '));

value = line.substr(line.find(' ') + 1);

trans\_map[key] = value;

}

fin.close();

fin.open(input\_file);

string text;

while (getline(fin, text))

{

istringstream stream(text);

string word;

while (stream >> word)

{

map<string, string>::const\_iterator it = trans\_map.find(word);

cout << (it == trans\_map.end() ? word + " " : it->second + " ");

}

cout << endl;

}

}

int main()

{

my\_word\_transform("map\_file.txt", "input\_file.txt");

return 0;

}

11.34答：如果使用下标操作如何查找key对应的元素？

11.35答：是一样的效果#include <iostream>

#include <vector>

#include <map>

#include <string>

#include <algorithm>

#include <list>

#include <set>

#include <utility>

#include <fstream>

#include <sstream>

using namespace std;

void my\_word\_transform(const string &map\_file, const string &input\_file)

{

ifstream fin(map\_file);

string line;

map<string, string> trans\_map;

string key, value;

//while (getline(fin, line))

//{

// string key, value;

// key = line.substr(0, line.find(' '));

// value = line.substr(line.find(' ') + 1);

// trans\_map[key] = value;

//}

while (fin >> key&&getline(fin, value))

{

trans\_map.insert({ key, value.substr(1) });

}

fin.close();

fin.open(input\_file);

string text;

while (getline(fin, text))

{

istringstream stream(text);

string word;

while (stream >> word)

{

map<string, string>::const\_iterator it = trans\_map.find(word);

cout << (it == trans\_map.end() ? word + " " : it->second + " ");

}

cout << endl;

}

}

int main()

{

my\_word\_transform("map\_file.txt", "input\_file.txt");

return 0;

}

11.36答：该什么代替就什么代替，如果一个关键字一个空格，那么用空格代替value。

11.37答：void my\_count(unordered\_map<string, int> &m)

{

string word;

while (cin >> word )

++m[word];

for (const auto &e : m)

std::cout << e.first << " " << e.second << endl;

}

通常一个无序容器可以替换对应的有序容器，反之亦然。一个无序容器的输出会与使用的有序容器的版本不同。

11.38答：#include <iostream>

#include <vector>

#include <map>

#include <string>

#include <algorithm>

#include <list>

#include <set>

#include <utility>

#include <fstream>

#include <sstream>

#include <unordered\_map>

using namespace std;

void my\_count(unordered\_map<string, int> &m)

{

string word;

while (cin >> word )

++m[word];

for (const auto &e : m)

std::cout << e.first << " " << e.second << endl;

}

void my\_word\_transform(const string &map\_file, const string &input\_file)

{

ifstream fin(map\_file);

string line;

unordered\_map<string, string> trans\_map;

string key, value;

//while (getline(fin, line))

//{

// string key, value;

// key = line.substr(0, line.find(' '));

// value = line.substr(line.find(' ') + 1);

// trans\_map[key] = value;

//}

while (fin >> key&&getline(fin, value))

{

trans\_map.insert({ key, value.substr(1) });

}

fin.close();

fin.open(input\_file);

string text;

while (getline(fin, text))

{

istringstream stream(text);

string word;

while (stream >> word)

{

unordered\_map<string, string>::const\_iterator it = trans\_map.find(word);

cout << (it == trans\_map.end() ? word + " " : it->second + " ");

}

cout << endl;

}

}

int main()

{

my\_word\_transform("map\_file.txt", "input\_file.txt");

return 0;

}