1. //vector-----------------
2. vector<**int**>v;
3. v.erase(v.begin() + 2);//删除第2个元素，从0开始计数
4. v.erase(v.begin() + 1, v.begin() + 5);//删除第1到第5个元素
5. sort(v.begin(), v.end());//元素排序
6. //sort(v.begin(), v.end(), cmp);//自定义排序规则，cmp函数自定义
7. reverse(v.begin(), v.end());
9. //string-----------------
10. string s;
11. s = "123456";
12. string::iterator it;
13. it = s.begin();
14. s.insert(it + 1, 'p');// s 变为 1p23456
15. s.erase(it + 3);//删除第3个字符，从0开始计数
16. s = "abc123456";
17. s.replace(3, 3, "good");//从第3个开始，将连续的3个字符替换为"good",即将123替换为good
18. reverse(s.begin(), s.end());
20. //set--------------------
21. set<**int**>st;
22. //反向遍历，从大到小
23. set<**int**>::reverse\_iterator rit;//定义反向迭代器
24. **for**(rit = st.rbegin(); rit != st.rend(); rit ++)
25. {
26. cout << \*rit << endl;
27. }
29. //multiset--------------里面值可以重复
30. multiset<**int**> ms;
31. **int** n = ms.erase(3); //删除里面所有的3，返回删除元素总个数
32. multiset<**int**>::iterator it;
33. it = ms.find(3);
34. **if**(it != ms.end())//找到了3这个元素
35. {
36. cout << \*it <<endl;
37. }
38. **else**
39. {
40. cout<< "not find it"<<endl;
41. }
43. //map-----------键值-->映照数据
44. map<**int**> mp;
45. mp.erase(2);//删除键值为2的元素
46. map<**int**>::iterator it;
47. it = mp.find(2);//查找键值
48. **if**(it != mp.end())//找到了
49. {
50. cout << (\*it).first << (\*it).second <<endl;
51. }
53. **struct** Info
54. {
55. string name;
56. **float** score;
57. //重载'<'操作符，自定义排序规则，在map中
58. **bool** operator < (**const** Info &a) **const**
59. {
60. //按score从小到大排序
61. **return** a.score < score;
62. }
63. };
65. map<Info, **int**> mmp;
67. //multimap------------允许键值相同的存在
68. multimap<string, **double**>m;
69. m.insert(pair<string, **double**> ("jack", 20.4));
70. m.insert(pair<string, **double**> ("jack", 34.1));
71. m.erase("jack");//删除键值等于"jack"的元素
72. //find()函数只返回重复键值中的第一个元素的迭代器位置

75. //deque-------------双端队列
76. push\_back()方法在尾部插入元素，不断扩张队列
77. push\_front()和insert()在首部和中间位置插入元素，只是将原位置上的元素值覆盖，不会增加新元素
78. deque<**int**> d;
79. d.push\_back(1);
80. d.push\_back(2);
81. d.push\_back(3);
82. d.insert(d.begin() + 1, 88);
83. cout << d[0] << d[1] << d[2] <<endl; //输出 1 88 3
84. d.pop\_front();//头部删除元素
85. d.pop\_back();//尾部删除元素
86. d.erase(d.begin() + 1);

89. //list--------------链表
90. list<**int**> l;
91. l.push\_back(2);
92. l.push\_back(1);
93. l.push\_back(5);//链表尾部插入元素，链表自动扩张
94. l.push\_front(8);//链表头部插入元素，链表自动扩张
95. list<**int**>::iterator it;
96. it = l.begin();
97. it ++;//注意链表的迭代器只能进行 ++ 或者 -- ，不能 + n
98. l.insert(it, 20);
99. **for**(it = l.begin(); it != l.end(); it ++)
100. {
101. cout << \*it <<endl;  //输出 8 20 2 1 5
102. }
103. l.remove(2);//值为2的节点都删除
104. l.pop\_front();//删除首部元素
105. l.pop\_back();//删除尾部元素
106. l.sort();//排序
107. l.unique();//剔除重复元素， 3 8 1 1 1 3 1 ---->  3 8 1 3 1

110. //bitset 位集合容器
111. bitset<10>b; //能容纳10个元素，也就是10位，默认都为0
112. //赋值
113. b[1] = 1;
114. b[6] = 1;
115. b[9] = 1;
116. //第0位是最低位，第9位是最高位
117. **for**(**int** i = b.size() - 1; i >= 0; i --)
118. {
119. cout << b[i] << " ";   //输出 1001000010
120. }
121. b.set();//全部重置为1
122. b.set(1, 1);//将第1位置为1
123. b.set(6, 1);
124. b.set(9, 1);
125. b.reset(9);//将第9位置为0
126. cout << b <<endl;// 和上面效果相同，也是输出1001000010