|  |  |  |  |
| --- | --- | --- | --- |
| C++ Standard Template Library (STL) | | | |
| Sequence | Container Name | Properties | Functions |
| Vector  #include <vector>  [vector<int> vec] | \*fast insert/remove at the end: O(1)  \* slow insert/remove at the beginnig or middle: O(n)  \* slow search: O(n)  \*support random access : vec[2]  \*copy constructor : vec2(vec)  \*support global initializer : vec = {1, 2, 3};  \*half-open : [begin, end)  \*Array based containners invalidates pointers:  🡪Native pointers, iterators, references | .at()  .begin()  .clear()  .empty()  .end()  .push\_back()  .size()  .swap()  .insert() |
| Deque  #include <deque>  [deque<int> deq]  {Similar to Vector} | \*fast insert/remove at the beginnig and the end: O(1)  \*slow insert/remove at middle: O(n)  \*slow search: O(n)  \*support random access : deq[2]  \*support global initializer : deq = {1, 2, 3}; | .begin()  .end()  .push\_back()  .push\_front()  .size()  .swap() |
| List  #include <list>  [list<int> mylist] | \*fast insert/remove at any place: O(1)  \*slow search: O(n)  \*don’t support random access : mylist[2]  \*support global initializer : mylist = {1, 2, 3}; | .begin()  .clear()  .empty()  .end()  .erase()  .find()  .insert()  .push\_back()  .push\_front()  .size()  .splice()  .swap() |
| Array  #include <array>  [array<int, 3> arr;] | \* size can not be changed  \* array<int, 3> a, array<int, 4> b; a & b are different in type | .begin()  .clear()  .empty()  .end()  .size()  .swap() |

|  |  |  |  |
| --- | --- | --- | --- |
| Associative | Set & Multiset  #include <set>  set<int> myset; | \*no duplicates  \*search fast O(log(n))  \*traversing is slow  \*don’t support random access : myset[2]  \*read only data structure  \*for set it’s value can not be modified  \*multiset support duplicate values | .find()  .insert()  .erase() |
| Map & Multimap  #include <map>  map<char, int> m;  {Similar to set} | \*no duplicates  \*search fast O(log(n))  \*traversing is slow  \*don’t support random access : mymap[2]  \*read only data structure  \*for map it’s key can not be modified  \*for multiset it’s key can not be modified  \*multiset support duplicate keys | .find()  .insert()  .erase()  make\_pair() |
| Array :: Using Map and Unordered Map | \*Search time: unoredered\_map 🡪 O(1)::O(n)  map 🡪 O(log(n))  \*Can’t use multimap and unordered multimap because they don’t have [] operator and also they don’t have unique key | .at() |
| Unordered | Unordered Set & Unordered Multiset  #include <set>  unordered\_set<int> myset; | \*no duplicates  \* fastest insert at any place O(1)  🡪Associative containner takes O(log(n))  🡪vector, deque takes O(n)  \*search fastest O(1))::O(n)  \*traversing is slow  \*don’t support random access : myset[2]  \*read only data structure  \*for set it’s value can not be modified  \*multiset support duplicate values  \*hash collision => performance degrade | .find()  .insert()  .erase()  .load  .load\_factor()  .bucket()  .bucket\_count()  .at() |
| Adaptor | Stack | \*LIFO | push()  pop()  top() |
| Queue | \*FIFO | pop()  push()  front()  back() |
| Priority Queue | \*first item has the greatest priority | push()  pop()  top() |

|  |  |  |  |
| --- | --- | --- | --- |
| Iterators | Random Access Iterator : vector, deque, array | \*support ++it(fast)/it++, –it/i—  \*it = it + 5, it = it - 3  \*support compare it1 < it2 |  |
| Bidirectional Iterator : list, set/multiset, map/multimap | \*support ++it/it++, –it/i— |  |
| Forward Iterator : forward\_list | \*support ++it/it++ |  |
| Input Iterator : | \*read and process values while iterating forward  \*int x = \*it; |  |
| Output Iterator : | \*Output values while iterating forward  \* \*it = 100; |  |
|  |  | \*Every Container has a iterator and a const\_iterator  [  set<int>::iterator it;  set<int>::const\_iterator cit;  ] | for\_each()  advance()  distance()  .cbegin()  .cend() |
| Iterators Adaptor |  | \*a special more poweful iterator  \*Insert iterator[vector<int>::iterator it;]  🡪back\_insert\_iterator  [back\_insert\_iterator<vector<int>> bit;]  [ back\_inserter() ]  🡪front\_insert\_iterator  [front\_insert\_iterator<vector<int>> fit]  \*Stream iterator  🡪 istream\_iterator  [istream\_iterator<string>(cin);]  🡪 ostream\_iterator  [ostream\_iterator<string>(cout, " ");]  \*Reverse iterator  [reverse\_iterator<vector<int>::iterator> rit;]  \*Move iterator | .rbegin()  .rend()  back\_inserter() |
| Algorithms |  | \*mostly loops  \*algorithms process data in half-open way:  [.begin(), .end())  \*Algorithms works with native c++ array | min\_element()  sort()  reverse()  copy()  find\_if()  count\_if()  transform()  bind()  function<>() |
| Functors |  | less, greater, greater\_equal, not\_equal\_to, logical\_and, logical\_not, logical\_or, multiplies, minus, plus, divide, modulus, negate |  |