**顺序容器**（容器不是排序的）

可变长动态数组 vector、双端队列 deque、双向链表 list

**关联容器**（关联容器内的元素是排序的）

set、multiset、map、multimap

**容器适配器**

栈 stack、队列 queue、优先级队列 priority\_queue

性质：

所有容器

1. 均有以下两个成员函数

|  |  |
| --- | --- |
| int size() | bool empty() |

2、只要容器类型相同，就可以用 <、<=、>、>=、==、!= 进行比较运算。

**顺序容器和关联容器共有**

|  |  |  |  |
| --- | --- | --- | --- |
| begin() | end() | rbegin() | rend() |
| erase(...) | clear() |  |  |

**迭代器**

正向迭代器 容器类名::iterator 迭代器名;

常量正向迭代器 容器类名::const\_iterator 迭代器名;

反向迭代器 容器类名::reverse\_iterator 迭代器名;

常量反向迭代器 容器类名::const\_reverse\_iterator 迭代器名;

vector<int> v; //v是存放int类型变量的可变长数组，开始时没有元素

for (int n = 0; n<5; ++n)

v.push\_back(n); //push\_back在vector容器尾部添加一个元素

vector<int>::iterator i; //定义正向迭代器

for (i = v.begin(); i != v.end(); ++i) { //用迭代器遍历容器

cout << \*i << " "; //\*i 就是迭代器i指向的元素

\*i \*= 2; //每个元素变为原来的2倍

}

cout << endl;

//用反向迭代器遍历容器

for (vector<int>::reverse\_iterator j = v.rbegin(); j != v.rend(); ++j)

cout << \*j << " ";

正向迭代器：支持以下操作：++p，p++，\*p，还可以用==和!=（注意不支持<）

双向迭代器：支持正向迭代器所有功能，还支持--p和p--

随机访问迭代器：p+=i，p-=i，p+i，p-i，p[i] （返回 p 后面第 i 个元素的引用）

|  |  |
| --- | --- |
| 容器 | 迭代器功能 |
| vector | 随机访问 |
| deque | 随机访问 |
| list | 双向 |
| set / multiset | 双向 |
| map / multimap | 双向 |
| stack | 不支持迭代器 |
| queue | 不支持迭代器 |
| priority\_queue | 不支持迭代器 |

#include <algorithm>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| advance(p, n) | distance(p, q) | iter\_swap(p, q) | copy | remove |
| random\_shuffle | fill | count\_if | InIt find(InIt first, InIt last, const T& val) | void sort(\_RandIt first, \_RandIt last) |

在 STL 中，默认情况下，比较大小是通过<运算符进行的，和>运算符无关

x和y相等是与x<y和y<x同时为假等价的，与==运算符无关

**顺序容器共有**

|  |  |  |
| --- | --- | --- |
| front() | back() | push\_back() |
| pop\_back() | insert(...) |  |

**vector**

|  |  |  |  |
| --- | --- | --- | --- |
| bool empty() | int size() | void pop\_back() | void clear() |
| bool empty() | void push\_back( const T & val) | iterator insert(iterator i, const T & val)插入 | iterator erase(iterator i)删除 |
| T & front() | T & back() | iterator begin() | iterator end() |

vector<vector<int> > v(3)

相当于生成了三个vector向量以列方向排列

**List**

|  |  |  |  |
| --- | --- | --- | --- |
| void push\_front(const T & val) | void pop\_front() | void sort() 默认升序 | void remove (const T & val) |
| remove\_if | void unique() | void merge(list <T> & x) |  |

**deque**

所有适用于 vector 的操作都适用于 deque

|  |  |
| --- | --- |
| void push\_front (const T & val) | void pop\_front() |

**关联容器：**

**set**：排好序的集合，不允许有相同元素。

**multiset**：排好序的集合，允许有相同元素。

**map**：每个元素都分为关键字和值两部分，容器中的元素是按关键字排序的。不允许有多个元素的关键字相同。

|  |  |  |
| --- | --- | --- |
| iterator find( const Key & val); | iterator insert (pair <Key, T> const &p); | void insert(iterator first, iterator last); |
| int count( const Key & val); | iterator lower\_bound( const Key & val); | iterator upper\_bound(const Key & val); |
| pair < iterator, iterator > equal\_range (const Key & val); | iterator erase(iterator it); | iterator erase(iterator first, iterator last); |

**multimap**：和 map 类似，差别在于元素的关键字可以相同。

map<int, string> data;

pair<int, string> one = make\_pair(1, "one");//**只能插入pair类数据**

data.insert(one);

map<int, string>::iterator p;//**需要定义一个map类型的迭代器**

p = data.find(2);

**容器适配器：**

|  |  |  |
| --- | --- | --- |
| push | top | pop |

容器适配器是没有迭代器的，因此 STL 中的各种排序、查找、变序等算法都不适用于容器适配器。

**stack：**

包含头文件#include <stack>

**queue：**

queue 可以用 list 和 deque 实现，默认情况下用 deque 实现。

**priority\_queue：**

priority\_queue 可以用 vector 和 deque 实现，默认情况下用 vector 实现。

优先队列的队头元素总是最大的，其内部并非完全有序

**C++ string类（C++字符串）**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| length | size | append | substr | swap |
| find | rfind | find\_first\_of | find\_last\_of | find\_first\_not\_of |
| find\_last\_not\_of | erase | insert | replace | compare |