Container

12.19.2024 Len Fu

string

#include

string 生成

string str("123"); //"123" string str(3,"1"); //"111" string str = "123"; //"123"

string 头尾

str.front(); //第一个元素 str.back(); //最后一个元素

string 迭代器

string::iterator iter //一个string的迭代器 str.begin() //指向str第一个元素位置的迭代器 str.end() //指向str最后一个元素后一个位置的迭代器

string 插入

str.push_back('a'); //在尾部插入一个字符 str.insert(s1.begin(), 'a'); //在指定位置前面插入一个字符

string 删除

str.pop_back(); //删除最后一个元素 str.erase(str.begin()); //删除迭代器指向元素 str.erase(str.begin()+1, str.end()-2); //删除指定区间的元素 str.erase(1, 6); //删除从索引（1）开始的n（6）个字符

string 替换

str.replace(str.begin(), str.begin + 5, "boy"); //替换迭代器指定的区间 str.replace(6, 5, "girl"); //替换索引指定的区间，从下标6开始的五个元素

string 拼接

str1.append(str2); //str1str2 str1 = str1 + str2; //str1str2

string 容量

str.size() str.length()

string 遍历

for(int i = 0; i < str.size(); i ++) //索引遍历 for(string::iterator iter = str.begin(); iter != str.end(); iter ++) //迭代器遍历 for(auto &x : str) //迭代器另一种便捷方法

string 排序

sort(str.begin(), str.end());

string 比较

str1 < str2 //字典序比较，<、<=、==、>、>=都可以用 str1.compare(str2) //相等为0，str1>str2为1，反之-1

sting 查找

str.find("123") //从前往后找，若找到，返回首字母下标；反之，返回-1 str.rfind("123") //从后往前找 str.find_first_of("123") //查找第一个属于该字符串的字符返回下标 str.find_first_not_of("123") str.find_last_of("123") str.find_last_not_of("123")

string 某元素个数

str.count('a'); //返回str里a字符的个数

string 分割

str.substr(2, 5); //返回从索引2开始的五个元素组成的字符串

string 判空

str.empty() //返回布尔值

string 清空

str.clear();

vector

#include

vector 生成

vector vct(3); //0 0 0 vector vct(3, 5); //5 5 5 vector vct{1, 2, 3}; //1 2 3

vector 头尾

vct.front(); //第一个元素 vct.back(); //最后一个元素

vector 迭代器

vector::iterator iter //一个vector的迭代器 vct.begin() //指向vct第一个元素位置的迭代器 vct.end() //指向vct最后一个元素后一个位置的迭代器

vector 插入

vct.push_back(2); //在尾部插入一个2 vct.insert(vct.begin(), 2); //在指定位置前面插入一个元素

vector 删除

vct.pop_back(); vct.erase(vct.begin()); vct.erase(vct.begin()+1, vct.end()-2); vct.erase(1, 6);

vector 容量

vct.size()

vector 遍历

for(int i = 0; i < vct.size(); i ++) //索引遍历 for(vector::iterator iter = vct.begin(); iter != vct.end(); iter ++) //迭代器遍历 for(auto &x : vct) //迭代器另一种便捷方法

vector 排序

sort(vct.begin(), vct.end());

vctor 查找

vct.find(2) //从前往后找，若找到，返回指向该处的迭代器；反之，返回迭代器vct.end()

vctor 某元素个数

vct.count(2); //返回容器里2的个数

vector 判空

vct.empty() //返回布尔值

vector 清空

vct.clear();

set

头文件set主要包括set和multiset（两者方法相同，所以以下只用set举例）两个容器，分别是“有序集合”和“有序多重集合”，即前者的元素不能重复，而后者可以包含若干个相等的元

素，两者都会将其容器内的元素从小到大自动排序。

set和multiset的内部实现是一棵红黑树，它们支持的函数基本相同。

#include

set 生成

set st;

set 迭代器

set::iterator iter st.begin() st.end()

set 插入

st.insert(2); //插入一个元素

set 删除

st.erase(st.begin()); //删除迭代器指向元素 st.erase(2); //删除所有为2的元素

set 容量

st.size()

set 查找

st.find(2) //从前往后找，若找到，返回指向该处的迭代器；反之，返回迭代器st.end() st.lower_bound(x) //二分查找大于等于x的元素中最小的一个，并返回指向该元素的迭代器。 st.upper_bound(x) //二分查找大于x的元素中最小的一个，并返回指向该元素的迭代器。

set 某元素个数

st.count(2); //返回容器里2的个数

set 判空

st.empty() //返回布尔值

set 清空

st.clear();

queue

头文件queue主要包括循环队列queue和优先队列priority_queue两个容器。其中queue容器相当于队列，满足先进先出的原则，即队尾出、队首进。

#include

queue 生成

queue q;

queue 头尾

q.front(); q.back();

queue 插入

q.push(2); //在队尾插入一个元素

queue 删除

q.pop(); //在队首删除一个元素

queue 容量

q.size();

queue 判空

q.empty()

其中priority_queue容器相当于大根堆（或者小根堆），大根堆每次堆顶是最大元素，小根堆每次堆顶是最小元素。

#include

priority_queue 生成

priority_queue q; //大根堆 priority_queue<int, vector, greater> q; //小根堆

priority_queue 插入

q.push(2); //把一个元素插入堆

priority_queue 删除

q.pop(); //删除堆顶的元素

priority_queue 堆顶

q.top(); //返回堆顶元素

priority_queue 容量

q.size();

priority_queue 判空

q.empty()

stack

stack容器相当于栈，满足先进后出的原则，即插入和删除都只能再栈顶操作。

#include

stack 生成

stack sk;

stack 插入

sk.push(2); //把一个元素放入栈

stack 删除

sk.pop(); //删除栈顶的元素

stack 栈顶

sk.top(); //返回栈顶元素

stack 容量

sk.size();

stack 判空

sk.empty()

deque

双端队列deque是一个支持在两端高效插入或删除元素的连续线性存储空间。它就像是vector和queue的结合。与vector相比，deque在头部增删元素仅需要 O(1)的时间；与queue相比，deque像数组一样支持随机访问。

#include

dequeue 生成

dequeue dq;

dequeue 头尾

dq.front(); dq.back();

dequeue 迭代器

dq.begin() dq.end()

dequeue 插入

dq.push_front(2); //头插入 dq.push_back(2); //尾插入

dequeue 删除

dq.pop_front(); //头删除 dq.pop_back(); //尾删除

dequeue 容量

dq.size();

dequeue 判空

dq.empty()

dequeue 清空

dq.clear();

map

map容器是一个键值对key-value的映射，其内部实现是一棵以key为关键码的红黑树。map的key和value可以是任意类型.

#include

map 生成

map<key_type, value_type> name; map<int, int> mp;

map 迭代器

map<int, int>::iterator iter mp.begin() mp.end()

map 键值

iter->first //key iter->second //value

map 插入

mp[2] = 5; //直接添加 mp.insert(pair<int, int>(2, 5)); //insert一个pair

删除

mp.erase(iter); //删除迭代器所指的键值对

map 容量

mp.size()

map 查找

mp.find(2) //从前往后找，若找到，返回指向该处的迭代器；反之，返回迭代器mp.end()

map 某元素个数

st.count(2); //返回key为2的个数（map中只可能是0或者1）

map 判空

mp.empty() //返回布尔值

map 清空

mp.clear();

pair

pair相当于将两份数据打包成一对，两份数据的数据类型任意，多与其他容器混合使用，单独使用的情况比较少。

#include

pair 生成

pair<int, int> pr = make_pair(0,1); pair<int, int> pr(0, 1);

pair 两个值

pr.first pr.second

pair 多与其他容器结合使用

set<pair<int, int>> st; vector<pair<int, int>> vct(mp.begin(), mp.end());