Stl容器

queue容器
- 常用接口
  - queue<T>q;
  - q.push()
  - q.pop()
  - q.front()
  - q.back()
  - q.empty()
  - q.size()

stack容器
- 常用接口
  - stack<T> s
  - s.push()
  - s.pop()
  - s.top()
  - s.empty()
  - s.size()

string容器
- 构造函数
  - const char *str = "hello world";
  - string s(str)
  - string s(10,'a');
- 赋值操作
  - s2 = s1;
  - s.assign("hello world")
- 字符串拼接
  - s1 += "a"
  - s1 += s2
  - s1.append(s2)
  - s1.append(s2, a)//s2中前a个字符
  - s1.append(s2,a,b)//s2中a下标开始b个字符
- 查找和替换
  - int pos = s1.find("?")
  - int pos.= s1.rfind("?")
  - s1.replace(1,3,"?")
- 字符串比较
  - s1.compare(s2)
- 存取
  - s[i]=?
  - s.at(i)=?
- 插入删除
  - s1.insert(1,"?")
  - s1.erase(a,b)//a开始b个字符
- 字串
  - int pos = str.find("?")
  - string s2 = s1.substr(0,pos);

set容器
- 构造和赋值
  - s.insert()
- 大小和交换
  - s.size()
  - s.empty()
  - s.swap(s1)
- 插入和删除
  - s.insert()
  - s.erase(s.begin())
  - s.erase(30)
  - s.clear()
- 查找和统计
  - auto it = s.find(30)
- 排序
  - set<T,Mycompare> s

常用算法
- 遍历算法
  - for_each(v.begin(), v.end(),print01)
- transform
  - Int Transform(int v){return v+100;}
transform(v.begin(),v.end(),vTarget.begin(), Myprint);
- 查找算法
  - auto it = find(v.begin(), v.end(), ?)
- 有条件的查找
  - auto it = find(v.begin(), v.end(), greater< int>());
- 查找相邻重复元素
  - auto it = adjacent_find(v.begin(), v.end())
- 二分查找
  - sort(v.begin(), v.end(), greater<int>());必须降序
  - bool ret = binary_search(v.begin(), v,end(), ?);
- count
  - int num = count(v.begin(), v.end(), ?)
- 有条件的count
  - int num = count_if(v.begin(), v,end(), Greater20());
- 排序算法
  - sort(v.begin(),v.end(),greater<int>()/less< int>());
- 打乱
  - random_shuffle(v.begin(),v.end());
- 融合
  - merge(v1.begin(), v1.end(), v2.begin(), v2. end(), vTarget.begin());
- 反转
  - reverse(v.begin(), v.end());
- 拷贝
  - copy(v1.begin(),v1.end(), v2.begin());
- 替换
  - replace(v1.begin(), v2.begin(), 5, 100);
- 有条件的替换
  - replace_if(v1.begin(), v1.end(), greater20(), 3000);
- 调换
  - swap(v1, v2);
- 累加
  - int total = accumulae(v.begin(), v.end(), 1000);
- 填充
  - fill(v.begin(), v.end(), 100);
- 交集
  - auto itEnd = set_intersection(v1.begin(), v1. end(), v2.begin(), v2.end(), vTarget.begin());
  - for_each(vTarget.begin(), itEnd, Myprint());
- 并集
  - auto itEnd = set_union(v1.begin(), v1.end(), v2. begin(), v2.end(), vTarget.begin());
- 不同集
  - auto itEnd = set_difference(v1.begin(), v1. end(), v2.begin(), v2.end(), vTarget.begin());
- 子主题 22

vector容器
- 构造函数
  - V.push_back()
  - V(v1.begin(), v1.end())
  - V(v1)
- 赋值操作
  - v=v1
  - V.assign(v1.begin(), v1.end())
- 容量大小
  - V.empty()
  - V.capacity()
  - V.size()
  - V.resize()
- 插入删除
  - V.insert(v.begin(), int num);
  - V.erase(v.begin())
  - V.clear()
- 互换
  - V.swap(v1)
  - Vector<int>(v1).swap(v1)
- 预留空间
  - V.reserve(int size)

deque容器
- 构造函数
  - d1.push_back()
  - d1.push_front()
  - d1.assign()
- 大小操作
  - d1.empty()
  - d1.resize()
- 插入删除
  - d1.insert(d1.begin(), 1000)
  - d1.erase(d1.begin())
  - d1.clear()
- 数据存取
  - d1.front()
  - d1.back()
- 排序
  - sort（d1.begin(), l'd.end()）

list容器
- 构造函数
  - l1push_back()
  - l1(l2)
- 赋值和交换
  - l1 = l2
  - l1.assign(l2)
  - l1.swap(l2)
- 大小操作
  - l1.resize()
  - l1.empty()
- 插入删除
  - l1.push_back()
  - li.push_front()
  - l1.pop_back()
  - li.pop_front()
- 数据存取
  - 不支持l1[1]. l1.at(1)
  - l1.front()
  - l1.back()
- 反转和排序
  - l1.sort()
  - l1.sort(mycompare)

map容器
- 构造函数
  - map<int,int>m
  - m.insert(pair<int,int>(a,b))
  - map<int,int>m2(m)
- 大小和交换
  - m.empty()
  - m.swap(m2)
- 插入和删除
  - m.insert(pair<int,int>(a,b))
  - m.insert(make_pair(a,b))
  - m.erase(m.begin())
  - m.erase(3)//按照key删除
- 查找和统计
  - auto it = m.find(a)
  - (*it)->first,(*it)->second
- 排序
  - map<int,int,Mycompare>m

Presented with XMind