|  |  |  |
| --- | --- | --- |
| STL标准容器 | 特点 | 典型的使用场景 |
| vector（矢量） | 1. 支持对元素的随机访问； 2. 提供自动内存管理功能，可动态地改变vector对象的长度； 3. vector的成员函数swap()效率比非成员函数swap()高； 4. 在尾部添加或删除元素的时间固定，但在头部和中间执行线性时间的插入和删除； 5. 是可反转容器。 | 1. 最简单的序列类型，除非其他类型的特殊优点能更好地满足需求，否则通常默认使用vector； 2. 需要一个高效随机访问和动态随机增长的容器。 |
| deque（双端队列） | 1. 支持对元素的随机访问； 2. 从对象的开始位置插入和删除元素的时间是固定的，但在序列中部执行线性时间的插入和删除操作； 3. 在序列中部的插入、删除元素和对元素的随机访问的速度比vector要慢。 | 多数操作发生在序列的起始和结尾处（如需要在容器的前后两端进行高效插入和删除）。 |
| list（双向链表） | 1. 不支持数组表示法和随机访问； 2. 在链表中任意位置进行插入和删除的时间都是固定的； 3. 从容器中插入或删除元素后，链表迭代器指向的元素不变（但链接的元素可能改变）； 4. 是可反转容器。 | 需要频繁地在容器中间插入或删除元素，而不需要进行随机访问的操作。 |
| queue（队列） | 1. 不允许随机访问队列元素和遍历队列； 2. 把使用限制在定义队列的基本操作上：可以将元素添加到队尾、从队首删除元素、查看队首和队尾的值、检查元素数目和测试队列是否为空； 3. 先进先出，支持在一端插入元素，在另一端删除元素。 | 用于实现队列数据结构，需要频繁地对队首和队尾元素进行操作。 |
| stack（栈） | 1. 不允许随机访问栈元素和遍历栈； 2. 把使用限制在定义栈的基本操作上：将压入堆到栈顶、从栈顶弹出元素、查看栈顶的值、检查元素数目和测试栈是否为空； 3. 后进先出，支持在一端插入和删除元素。 | 用于实现栈数据结构。 |
| set（关联集合） | 可反转，可排序，且键是唯一的（不能存储多个相同的值）。 | 需要存储一组有序且唯一的元素。 |
| multiset | 1. 存储有序； 2. 允许存储重复的元素； 3. 不支持常数时间内的随机访问操作。 | 需要按顺序存储并统计重复元素的情况 |
| map（映射） | 键值对的容器，根据键值快速查找、插入和删除值。 | 需要根据键来查找对应值的情况 |
| multimap | 1. 可反转、经过排序； 2. 键和值的类型不同，且一个键可能与多个值相关联； | 1. 处理具有相同键的数据集合； 2. 实现索引或关联数据结构； 3. 解决名称冲突问题。 |