**序列式容器**

1、vector：

连续存储的容器，动态数组，在堆上分配空间。在最后插入（空间够）：很快；在最后插入（空间不够）：需要内存申请和释放，以及对之前数据进行拷贝；在中间插入（空间够）：内存拷贝；在中间插入（空间不够）：需要内存申请和释放，以及对之前数据进行拷贝。在最后删除：很快；在中间删除：内存拷贝；

适用场景：经常随机访问，且不经常对非尾节点进行插入删除。

2、list

动态链表，在堆上分配空间，每插入一个元素都会分配空间，每删除一个元素都会释放空间。

在任何位置插入和删除效率都比较好，因为插入和删除时候相对于数组来说，不需要移动元素

缺点：不支持随机访问。非连续的内存空间，所以说为了保存结点和结点之间的前驱和后继的关系，需要提供额外的空间开销

适用场景：经常插入删除大量元素。

**关联式容器**

1、set

底层使用平衡的搜索树——红黑树实现。set的特性是，所有元素都会根据元素的值自动被排序（默认升序），set元素的键值就是实值，实值就是键值，set不允许有两个相同的键值，不允许迭代器修改元素的值，其迭代器是一种constance iterators。插入删除操作时仅仅需要指针操作节点即可完成，不涉及到内存移动和拷贝，所以效率比较高。

适用场景：存储要求从高到低分的顺序排列。

2、map

map的所有元素都会根据元素的键值被自动排序，map的所有元素都是pair，同时拥有实值（value）和键值（key），pair的第一元素被视为键值，第二元素被视为实值，通常通过仿函数select提取出节点的键值进行比较，我们能修改节点的实值但不能修改其键值，对其他元素操作时，其之前和之后的迭代器都不会失效。map使用rb-tree作为底层容器，rb-tree提供了所有map需要的操作，map使用RB\_tree的 insert\_unique 来插入元素。map的所有操作都是通过查找匹配元素的键 key 来完成的，和其对应映射值 value 无关。因为 map 不允许数据冗余，所以每个元素的 key 值是唯一的。

特点：在两端插入和删除效率都比较高，在指定位置插入，也会引起数据元素的后移，支持随机访问。

适用场景：大量数据中快速查询，效率较高

3、queue

规则：只有一个入口和一个出口，分别位于最底端和最顶端，除了出口元素外，没有其他方法可以获取到内部的其他元素。不能遍历，不提供迭代器，也不支持随机访问。

适用场景：向序列两端频繁的添加或删除元素

4、deque

deque是一种双向开口的连续线性空间，运行在常数时间内对头端进行元素操作。deque没有容量概念，它是动态地以分段连续空间组合而成，可以随时增加一段新的空间并链接起来。deque虽然也提供随机访问的迭代器，但是其迭代器并不是普通的指针，其复杂程度比vector高很多，因此除非必要，否则一般使用vector而非deque。

适用场景：头端的快速移除，尾端的快速添加