

顺序容器

1、vector（向量）：相当于数组，但其大小可以不预先指定，并且自动扩展。它可以像数组一样被操作，

由于它的特性我们完全可以将vector 看作动态数组。在创建一个vector 后，它会自动在内存中分配一块连续的

内存空间进行数据存储，初始的空间大小可以预先指定也可以由vector 默认指定，这个大小即capacity（）函数

的返回值。当存储的数据超过分配的空间时vector 会重新分配一块内存块，但这样的分配是很耗时的，效率非常低。

2、deque（队列）：它不像vector 把所有的对象保存在一块连续的内存块，而是采用多个连续的存储块，并且在一个

映射结构中保存对这些块及其顺序的跟踪。向deque 两端添加或删除元素的开销很小，它不需要重新分配空间。

3、list（列表）：是一个线性链表结构，它的数据由若干个节点构成，每一个节点都包括一个信息块（即实际存储的数据）、

一个前驱指针和一个后驱指针。它无需分配指定的内存大小且可以任意伸缩，这是因为它存储在非连续的内存空间中，

并且由指针将有序的元素链接起来。

4、set, multiset, map, multimap 是一种非线性的树结构，具体的说采用的是一种比较高效的特殊的平衡检索二叉树——红黑树结构。

各种容器优劣分析

1、Vector：

优点：

- A、支持随机访问，访问效率高和方便，它像数组一样被访问，即支持[] 操作符和vector.at()。
- B、节省空间，因为它是连续存储，在存储数据的区域都是没有被浪费的，但是要明确一点vector 大多情况下并不是满存的，在未存储的区域实际是浪费的。

缺点：

- A、在内部进行插入、删除操作效率非常低。
- B、只能在vector 的最后进行push 和pop，不能在vector 的头进行push 和pop。
- C、当动态添加的数据超过vector 默认分配的大小时要进行内存的重新分配、拷贝与释放，这个操作非常消耗能。

2、List：

优点：

不使用连续的内存空间这样可以随意地进行动态操作，插入、删除操作效率高；

缺点：

- A、不能进行内部的随机访问，即不支持[] 操作符和vector.at()，访问效率低。
- B、相对于vector 占用更多的内存。

3、Deque：

优点：

- A、支持随机访问，方便，即支持[] 操作符和vector.at()，但性能没有vector 好；
- B、可以在两端进行push、pop。

缺点：

在内部进行插入、删除操作效率低。

综合：

vector 的查询性能最好，并且在末端增加数据也很好，除非它重新申请内存段；适合高效地随机存储。

list 是一个链表，任何一个元素都可以是不连续的，但它都有两个指向上一元素和下一元素的指针。所以它对插入、删除元素性能是最好的，而查询性能非常差；适合大量地插入和删除操作而不关心随机存取的需求。

deque 是介于两者之间，它兼顾了数组和链表的优点，它是分块的链表和多个数组的联合。所以它有被list 好的查询性能，有被vector 好的插入、删除性能。如果你需要随即存取又关心两端数据的插入和删除，那么deque 是最佳之选。

关联容器

关联容器的特点是明显的，相对于顺序容器，有以下几个主要特点：

- A，其内部实现是采用非线性的二叉树结构，具体的说是红黑树的结构原理实现的；
- B，set 和map 保证了元素的唯一性，multiset 和multimap 扩展了这一属性，允许元素不唯一；
- C，元素是有序的集合，默认在插入的时候按升序排列。

基于以上特点，

A，关联容器对元素的插入和删除操作比vector 要快，因为vector 是顺序存储，而关联容器是链式存储；比list 要慢，是因为即使它们同是链式结构，但list 是线性的，而关联容器是二叉树结构，其改变一个元素涉及到其它元素的变动比list 要多，并且它是排序的，每次插入和删除都需要对元素重新排序；

B，关联容器对元素的检索操作比vector 慢，但是比list 要快很多。vector 是顺序的连续存储，当然是比不上的，但相对链式的list 要快很多是因为list 是逐个搜索，它搜索的时间是跟容器的大小成正比，而关联容器查找的复杂度基本是Log(N)，比如如果有1000 个记录，最多查找10 次，1,000,000 个记录，最多查找20 次。容器越大，关联容器相对list 的优越性就越能体现；

C，在使用上set 区别于vector,deque,list 的最大特点就是set 是内部排序的，这在查询上虽然逊色于vector，但是却大大的强于list。

D，在使用上map 的功能是不可取代的，它保存了“键- 值”关系的数据，而这种键值关系采用了类数组的方式。数组是用数字类型的下标来索引元素的位置，而map 是用字符型关键字来索引元素的位置。在使用上map 也提供了一种类数组操作的方式，即它可以通过下标来检索数据，这是其他容器做不到的，当然也包括set。（STL 中只有vector 和map 可以通过类数组的方式操作元素，即如同ele[1] 方式）。

map, multimap, unordered_map

map的特性是，所有元素都会根据元素的键值自动被排序。map的所有元素都是pair,同时拥有键值（key）和实值（value）。pair的第一元素被视为键值，第二元素被视为实值。map不允许两个元素拥有相同的键值

multimap的特性以及用法与map完全相同，唯一的差别在于它允许键值重复。unordered_map与map的区别就在于不会根据key的大小进行排序。

简单对比

map与unordered_map相比:

map底层实现为红黑数, unordered_map底层实现为哈希表, 两者均不能有重复的键, 均支持[]运算符

map与multimap相比:

两者底层实现均为红黑树, 但是multimap支持重复的键, 不支持[]运算符。

内部实现机理不同

map: map内部实现了一个红黑树(红黑树是非严格平衡二叉搜索树, 而AVL是严格平衡二叉搜索树), 红黑树具有自动排序的功能, 因此map内部的所有元素都是有序的, 红黑树的每一个节点都代表着map的一个元素。因此, 对于map进行的查找, 删除, 添加等一系列的操作都相当于是对红黑树进行的操作。map中的元素是按照二叉搜索树(又名二叉查找树、二叉排序树, 特点就是左子树上所有节点的键值都小于根节点的键值, 右子树所有节点的键值都大于根节点的键值)存储的, 使用中序遍历可将键值按照从小到大遍历出来。

unordered_map: unordered_map内部实现了一个哈希表(也叫散列表, 通过把关键码值映射到Hash表中一个位置来访问记录, 查找的时间复杂度可达到 $O(1)$, 其在海量数据处理中有着广泛应用)。因此, 其元素的排列顺序是无序的。哈希表详细介绍

优缺点以及适用处

map:

优点: 有序性, 这是map结构最大的优点, 其元素的有序性在很多应用中都会简化很多的操作。

内部实现一个红黑书使得map的很多操作在 $\lg(n)$ 的时间复杂度下就可以实现, 因此效率非常的高。

缺点: 空间占用率高, 因为map内部实现了红黑树, 虽然提高了运行效率, 但是因为每一个节点都需要额外保存父节点、孩子节点和红/黑性质, 使得每一个节点都占用大量的空间。

适用处: 对于那些有顺序要求的问题, 用map会更高效一些

unordered_map:

优点: 因为内部实现了哈希表, 因此其查找速度非常的快

缺点: 哈希表的建立比较耗费时间

适用处: 对于查找问题, unordered_map会更加高效一些, 因此遇到查找问题, 常会考虑一下用unordered_map

总结:

内存占有率的问题就转化成红黑树 VS hash表, 还是unordered_map占用的内存要高。

但是unordered_map执行效率要比map高很多

对于unordered_map或unordered_set容器, 其遍历顺序与创建该容器时输入的顺序不一定相同, 因为遍历是按照哈希表从前往后依次遍历的

map和unordered_map的使用

unordered_map的用法和map是一样的, 提供了insert, size, count等操作, 并且里面的元素也是以pair类型来存贮的。其底层实现是完全不同的, 上方已经解释了, 但是就外部使用来说却是一致的。