|  |  |
| --- | --- |
| 1. 无序容器 2. 使用无序容器 3. 管理桶 | 无序关联容器  哈希函数和关键字类型==运算符  Hash模板 |
| 1. 无序容器 2. 4个无序关联容器，不是使用比较运算符来组织元素，而是一个哈希函数和关键字类型==运算符。 3. 有时候很有用，因为维护元素的序代价高   Note：如果关键字类型固有就是无序的，或者性能测试发现问题可以用哈希计数解决，就可以使用无序容器   1. 使用无序容器 2. 定义与有序容器相关操作，但是因为序不同，输出的可能不同 3. 管理桶 4. 桶的概念是保存0或多个元素，使用哈希函数将元素映射到桶。 5. 为了访问一个元素，容器首先计算元素的哈希值，指出应该搜索哪个桶。 6. 具有一个特定哈希值的所有元素都保存在相同的桶中，无序容器的性能依赖于哈希函数的质量和桶的数量和大小 7. 一个桶可以保存一个元素，也可以一个桶保存多个元素。但多个元素中需要顺序搜索，可能大量的比较操作。 8. 提供了对桶的管理操作 9. 无序容器对关键字类型的要求 10. 默认，关键字使用==运算符比较元素，还使用一个hash<key\_type>类型的对象生成每个元素的哈希值。 11. 为内置类型，string，智能指针类型提供了hash模板hash<string>()(sd.isbn());，即只要传入值就可以自动得到哈希值。但是要指定为内置类型？ 12. 与容器不同，不能直接使用哈希模板，必须提供我们自己的hash模板版本，不使用默认的hash，为了让类当做关键字，提供函数来代替==运算符和哈希值计算函数   因为类不能生成哈希值，只有内置类型hash的定义了hash，所以要使用类的某个成员当做hash值来区分让类可以当关键字，所以要自定义。   1. 使用这些函数定义一个无序集合。 2. 若类作为关键字并且有默认的==操作，则只需要希值的函数还。   代码：  size\_t hasher(const Sales\_data &sd)  {  return hash<string>()(sd.isbn());  }  bool eqOp(const Sales\_data &lhs, const Sales\_data &rhs)  {  return lhs.isbn() == rhs.isbn();  }  using SD\_multiset = unordered\_multiset<Sales\_data, decltype(hasher)\*, decltype(eqOp)\*>;  SD\_multiset bookstore(42, hasher, eqOp); | |

