关联容器

关联容器概述

1. 关联容器与顺序容器的根本不同是：关联容器中的元素是按关键字来保存和访问的。与之相对，顺序容器中的元素是按它们在容器中的位置来顺序保存和访问的。
2. 标准库提供了8个关联容器，容器之间的不同体现在：
   - 每个容器或者是一个set或者是一个map，set支持高效的关键字查询，而map主要用于类似字典查询的操作
   - 每个容器或者要求不重复的关键字或者允许重复关键字；（允许重复关键字的容器的名字中都包含单词multi）
   - 每个容器或者按顺序保存元素或者无序保存。
3. 关联容器的迭代器都是双向的。
4. 初始化map容器时可以用花括号提供关键字和值的键值对：

```
map<string,int> record={{"yang":5}}
```

5. 自定义类型作为关联容器的关键字类型时，需要定义严格弱序操作。

```
multiset<类型,decltype(关于类型的比较函数的函数名)*> MS(比较函数的函数名);
```

   用类型的比较函数初始化对象，表示向对象添加元素时通过调用该函数来为这些元素排序。
6. map的迭代器是一个pair，其第一个元素是const，即关键字的值是const，表示无法改变。set也一样，可以通过迭代器读，但不可以写。
7. 对于不包含重复关键字的容器，添加单一元素的insert和emplace返回一个pair，其first成员是一个迭代器，指向具有给定关键字的元素；second成员是一个bool值，指出元素是否插入成功或者已在容器中。若关键字已在容器中则insert什么事也不做，且返回值的bool为false。
8. 删除元素方面，关联容器有三个版本的erase，其中两个的参数是迭代器，用于删除迭代器指向的元素或迭代器包括的范围内的元素，返回值是void。另一个版本的erase的参数是一个关键字，返回值是0或者1，0表示要删除的元素不在容器中。对于允许重复元素的容器则可能返回值大于1。
9. 关联容器额外的类型别名：
   - key_type：此容器类型的关键字类型；
   - mapped_type：每个关键字关联的类型（即键值对中，“值”的类型），只适用于map
   - value_type：对于set来说与key_type相同，对于map来说，为pair<const key_type,mapped_type>

   因此map的下标运算符返回的类型与解引用map迭代器得到的类型不同。
10. 有序关联容器中的二分查找：

- lower_bound(K)：返回一个 指向第一个关键字**不小于**K的元素 的迭代器；
- upper_bound(K)：返回一个 执行第一个关键字**大于**K的元素 的迭代器；

若元素不在容器中，则同时调用lower_bound和upper_bound会返回相等的迭代器。

也可以使用这两个函数类得到具有该关键字的元素的范围。

11. 允许重复关键字的容器中，使用find函数查找关键字的迭代器得到的是该关键字的第一个记录：

```
//下面代码会依次输出Yang关键字对于的多个值
multimap<string,int> record;
record.insert("Yang",1),record.insert("Yang",2),record.insert("Yang",3);
//使用count和find结合版本
auto size=record.count("Yang");
auto it=record.find("Yang");
while(size--){
    cout<<it->second<<' ';
    ++it;
}

//使用lower_bound和upper_bound的版本
auto begin=record.lower_bound("Yang"),end=record.upper_bound("Yang");
for(;begin!=end;++begin){
    cout<<begin->second<<' ';
}

//使用equal_range函数
for(auto pos=record.equal_range("Yang");pos.first!=pos.second;++pos.first){
    cout<<pos.first->second<<' ';
}
```

其中equal_range函数的功能是返回包含指定关键字的范围，用一个pair表示。pair.first是该范围的起始点的迭代器，pair.second是该范围的结尾的迭代器。

无序容器

1. 无序容器在存储上组织为一组桶，每个桶保存零个或多个元素。无序容器使用一个哈希函数将元素映射到桶。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。
2. 无序容器提供了一组管理桶的函数
3. 无序容器使用关键字类型的运算符来比较元素，还使用一个hash<key_type>类型的对象来生成每个元素的哈希值。因此若无序容器的关键字类型为自定义类型，则不仅要定义自定义类型的运算符，还要定义其hash模板。在定义容器时要传入自定义的哈希模板以及==运算符比较函数作为参数。

```
size_t myhash(const 自定义类型 arg){
    return hash<内置类型>() (arg.数据成员); //数据成员类型要与<>内的内置类型相同
}
bool equal(const 自定义类型 arg1,const 自定义类型 arg2){
    return arg1.数据成员==arg2.数据成员;
}
unordered_set<自定义类型,decltype(myhash)*,decltype(equal)*> MySet;
```