unordered_set

性质

- unordered_set 容器提供了和 unordered_map 相似的能力，但 unordered_set 可以用保存的元素作为它们自己的键。T 类型的对象在容器中的位置由它们的哈希值决定，因而需要定义一个 Hash< T > 函数。基本类型可以省去Hash< T >方法。
- 不能存放重复元素。
- 可指定buckets个数，可进行初始化，也可后期插入元素
- 1、无序集是一种容器，它以不特定的顺序存储惟一的元素，并允许根据元素的值快速检索单个元素。
- 2、在unordered_set中，元素的值同时是唯一标识它的键。键是不可变的，只可增删，不可修改
- 3、在内部，unordered_set中的元素没有按照任何特定的顺序排序，而是根据它们的散列值组织成桶，从而允许通过它们的值直接快速访问单个元素(平均时间复杂度为常数)。
- 4、unordered_set容器比set容器更快地通过它们的键访问单个元素，尽管它们在元素子集的范围迭代中通常效率较低。
- 5、容器中的迭代器至少是前向迭代器。

```
std::unordered_set<string> example;
std::unordered_set<string> things {16}; // 16 buckets
std::unordered_set<string> words {"one", "two", "three", "four"};// Initializer list
std::unordered_set<string> copy_wrds {words}; // Copy constructor
```

成员函数

构造

函数声明	功能介绍
explicit unordered_set (size_type n = /* see below */,const hasher& hf = hasher(),const key_equal& eql = key_equal(),const allocator_type& alloc = allocator_type());	构造一个空的unordered_set
explicit unordered_set (const allocator_type& alloc);	构造一个空的unordered_set
template unordered_set (InputIterator first, InputIterator last,size_type n = /* see below */,const hasher& hf = hasher(),const key_equal& eql = key_equal(),const allocator_type& alloc = allocator_type());	用[first,last)区间中的元素构造unordered_set
unordered_set (const unordered_set& ust);	unordered_set的拷贝构造
unordered_set (const unordered_set& ust, const allocator_type& alloc);	unordered_set的拷贝构造
unordered_set (unordered_set&& ust);	将ust移动到另外一个unordered_set中
unordered_set (unordered_set&& ust, const allocator_type& alloc);	将ust移动到另外一个unordered_set中
unordered_set (initializer_list<value_type> il,size_type n = /* see below */,const hasher& hf = hasher(),const key_equal& eql = key_equal(),const allocator_type& alloc = allocator_type());	将initializer_list移动到unordered_set中

迭代器

函数声明	功能介绍
begin	返回unordered_set第一个元素的迭代器
end	返回unordered_set最后一个元素下一个位置的迭代器
cbegin	返回unordered_set第一个元素的const迭代器
cend	返回unordered_set最后一个元素下一个位置的const迭代器

其余操作

函数声明	功能介绍
bool empty() const	检测unordered_set是否为空
size_t size() const	获取unordered_set的有效元素个数
iterator find(const key_type& k)	返回k在哈希桶中的位置
size_t count(const key_type& k)	返回哈希桶中关键码为k的键值对的个数
insert	向容器中插入键值对
erase	删除容器中的键值对
void clear()	清空容器中有效元素个数
void swap(unordered_set&)	交换两个容器中的元素
size_t bucket_count()const	返回哈希桶中桶的总个数
size_t bucket_size(size_t n)const	返回n号桶中有效元素的总个数
size_t bucket(const key_type& k)	返回元素key所在的桶号

查找

1. 调用 unordered_set 的 find() 会返回一个迭代器。这个迭代器指向和参数哈希值匹配的元素，如果没有匹配的元素，会返回这个容器的**结束迭代器(set.end())**。

```cpp
#include <iostream>
#include <unordered_set>

int main(){
    std::unordered_set<int> example = {1, 2, 3, 4};

    auto search = example.find(2);
    if (search != example.end()) {
        std::cout << "Found " << (*search) << '\n';
    } else {
        std::cout << "Not found\n";
    }
}/*output:
2
*/
```


注意：如果没有匹配的元素，返回结束迭代器(set.end())

擦除

1. 调用unordered_set容器的成员函数clear()可以删除它的全部元素。

2. 成员函数erase()可以删除容器中和传入参数的哈希值相同的元素。

3. 另一个版本的erase()函数可以删除迭代器参数指向的元素。这个版本的 erase() 会返回一个 size_t 类型的数作为被删除元素的个数。对于unordered_set来说，这个值只能是 0 或 1，但对于 unordered_multiset 容器来说，这个值可能会大于 1。显然，如果返回值是 0，那么容器中肯定没有这个元素。

```
std::pair<string, string> person { "John", "Smith"};
auto iter = names.find(person);
if(iter != std::end(names))
    names.erase(iter);
```


案例代码

```
void Sub_1()
{
    cout << "unordered_set测试" << endl;
    vector<int> nums1 = { 1,2,3,4,5,6 };
    unordered_set<int> set1(nums1.begin(), nums1.end());    //使用vector初始化set
    for (int num : set1)    //C++新特性 遍历容器中的每一个成员
    {
        cout << num << endl;
    }

    cout << "查找元素6" << endl;
    cout << *set1.find(6) << endl;  //.find返回的是一个地址 如果没找到返回结束迭代器
    if (set1.find(7) == set1.end())
    {
        cout << "没有查找到7" << endl;
        cout << *set1.find(7) << endl;
    }
}
```