std::unordered_map

在标头 <unordered_map> 定义

```
template<
    class Key,
    class T,
    class Hash = std::hash<Key>,                                    (1)   (C++11 起)
    class KeyEqual = std::equal_to<Key>,
    class Allocator = std::allocator< std::pair<const Key, T> >
> class unordered_map;
```

```
namespace pmr {
    template <class Key,
              class T,
              class Hash = std::hash<Key>,                          (2)   (C++17 起)
              class KeyEqual = std::equal_to<Key>>
              using unordered_map = std::unordered_map<Key, T, Hash, Pred,
                        std::pmr::polymorphic_allocator<std::pair<const Key,T>>>;
}
```

unordered_map 是关联容器，含有带唯一键的键-值 pair 。搜索、插入和元素移除拥有平均常数时间复杂度。

元素在内部不以任何特定顺序排序，而是组织进桶中。元素放进哪个桶完全依赖于其键的哈希。这允许对单独元素的快速访问，因为一旦计算哈希，则它准确指代元素所放进的桶。

`std::unordered_map` 满足*容器 (Container)* 、*知分配器容器 (AllocatorAwareContainer)* 、*无序关联容器 (UnorderedAssociativeContainer)* 的要求。

迭代器非法化

操作	非法化
所有只读操作、 swap 、 std::swap	决不
clear 、 rehash 、 reserve 、 operator=	始终
insert 、 emplace 、 emplace_hint 、 operator[]	仅若重哈希导致
erase	仅为指向被擦除元素者

注意

- swap 函数不非法化容器内的任何迭代器，但它们非法化标记交换区域结尾的迭代器。
- 指向存储于容器中的关键或元素的引用和指针仅因擦除该元素才被非法化，即使在非法化对应迭代器时。

成员类型

成员类型	定义
`key_type`	`Key`
`mapped_type`	`T`
`value_type`	std::pair<const `Key, T`>
`size_type`	无符号整数类型（通常是 std::size_t）
`difference_type`	有符号整数类型（通常是 std::ptrdiff_t）
`hasher`	`Hash`
`key_equal`	`KeyEqual`
`allocator_type`	`Allocator`
`reference`	`value_type&`
`const_reference`	const `value_type&`
`pointer`	std::allocator_traits<`Allocator`>::pointer
`const_pointer`	std::allocator_traits<`Allocator`>::const_pointer
`iterator`	指向 `value_type` 的常*老式向前迭代器 (LegacyForwardIterator)*

`const_iterator`	指向 const `value_type` 的*老式向前迭代器 (LegacyForwardIterator)*
`local_iterator`	类别、值、差、指针和引用类型都与 `iterator` 相同的迭代器类型。 能用此迭代器在单个桶迭代，但不能跨桶。
`const_local_iterator`	类别、值、差、指针和引用类型都与 `const_iterator` 相同的迭代器类型。 能用此迭代器在单个桶迭代，但不能跨桶。
`node_type`(C++17 起)	表示容器结点的结点把柄特化

`insert_return_type`(C++17 起)	描述插入 `node_type` 结果的类型，下列类型的特化
	`template<class Iter, class NodeType>`
	`struct /*未指定*/`
	`{`
	`  Iter position;`
	`  bool inserted;`
	`  NodeType node;`
	`};`
	以模板实参 `iterator` 和 `node_type` 实例化。

成员函数

(构造函数) (C++11)	构造 `unordered_map` (公开成员函数)
(析构函数) (C++11)	析构 `unordered_map` (公开成员函数)
operator= (C++11)	赋值给容器 (公开成员函数)
get_allocator (C++11)	返回相关的分配器 (公开成员函数)

迭代器

begin **cbegin** (C++11)	返回指向起始的迭代器 (公开成员函数)
end **cend** (C++11)	返回指向末尾的迭代器 (公开成员函数)

容量

empty (C++11)	检查容器是否为空 (公开成员函数)
size (C++11)	返回容纳的元素数 (公开成员函数)
max_size (C++11)	返回可容纳的最大元素数 (公开成员函数)

修改器

clear (C++11)	清除内容 (公开成员函数)
insert (C++11)	插入元素或结点 (C++17 起) (公开成员函数)
insert_or_assign (C++17)	插入元素，或若键已存在则赋值给当前元素 (公开成员函数)
emplace (C++11)	原位构造元素 (公开成员函数)
emplace_hint (C++11)	使用提示原位构造元素 (公开成员函数)
try_emplace (C++17)	若键不存在则原位插入，若键存在则不做任何事 (公开成员函数)
erase (C++11)	擦除元素 (公开成员函数)
swap (C++11)	交换内容 (公开成员函数)

extract (C++17)	从另一容器释出结点 (公开成员函数)
merge (C++17)	从另一容器接合结点 (公开成员函数)

查找

at (C++11)	访问指定的元素，同时进行越界检查 (公开成员函数)
operator[] (C++11)	访问或插入指定的元素 (公开成员函数)
count (C++11)	返回匹配特定键的元素数量 (公开成员函数)
find (C++11)	寻找带有特定键的元素 (公开成员函数)
contains (C++20)	检查容器是否含有带特定键的元素 (公开成员函数)
equal_range (C++11)	返回匹配特定键的元素范围 (公开成员函数)

桶接口

begin(size_type) **cbegin**(size_type) (C++11)	返回一个迭代器，指向指定的桶的开始 (公开成员函数)
end(size_type) **cend**(size_type) (C++11)	返回一个迭代器，指向指定的桶的末尾 (公开成员函数)
bucket_count (C++11)	返回桶数 (公开成员函数)
max_bucket_count (C++11)	返回桶的最大数量 (公开成员函数)
bucket_size (C++11)	返回在特定的桶中的元素数量 (公开成员函数)
bucket (C++11)	返回带有特定键的桶 (公开成员函数)

哈希策略

load_factor (C++11)	返回每个桶的平均元素数量 (公开成员函数)
max_load_factor (C++11)	管理每个桶的平均元素数量的最大值 (公开成员函数)
rehash (C++11)	为至少为指定数量的桶预留存储空间并重新生成散列表 (公开成员函数)
reserve (C++11)	为至少为指定数量的元素预留存储空间并重新生成哈希表。 (公开成员函数)

观察器

hash_function (C++11)	返回用于对键散列的函数 (公开成员函数)
key_eq (C++11)	返回用于比较键的相等性的函数 (公开成员函数)

非成员函数

operator== **operator!=** (C++20 中移除)	比较 unordered_map 中的值 (函数模板)
std::swap(std::unordered_map) (C++11)	特化 std::swap 算法 (函数模板)
erase_if(std::unordered_map) (C++20)	擦除所有满足特定判别标准的元素 (函数模板)

推导指引(C++17 起)

示例

运行此代码

```
#include <iostream>
#include <string>
#include <unordered_map>

int main()
{
    // 创建三个 string 的 unordered_map （映射到 string ）
    std::unordered_map<std::string, std::string> u = {
        {"RED","#FF0000"},
        {"GREEN","#00FF00"},
        {"BLUE","#0000FF"}
    };

    // 迭代并打印 unordered_map 的关键和值
    for( const auto& n : u ) {
        std::cout << "Key:[" << n.first << "] Value:[" << n.second << "]\n";
    }

    // 添加新入口到 unordered_map
    u["BLACK"] = "#000000";
    u["WHITE"] = "#FFFFFF";

    // 用关键输出值
    std::cout << "The HEX of color RED is:[" << u["RED"] << "]\n";
    std::cout << "The HEX of color BLACK is:[" << u["BLACK"] << "]\n";

    return 0;
}
```

输出：

```
Key:[RED] Value:[#FF0000]
Key:[BLUE] Value:[#0000FF]
Key:[GREEN] Value:[#00FF00]
The HEX of color RED is:[#FF0000]
The HEX of color BLACK is:[#000000]
```

来自“https://zh.cppreference.com/mwiki/index.php?title=cpp/container/unordered_map&oldid=58701”