容器库

容器库是类模板与算法的汇集，允许程序员简单地访问常见数据结构，例如队列、链表和栈。有两 (C++11 前)三 (C++11 起)类容器：

- 顺序容器
- 关联容器
- 无序关联容器 (C++11 起)

其中每一个被设计为支持不同的一系列操作。

容器管理着为其元素分配的存储空间，并提供成员函数来直接访问或通过迭代器（具有类似于指针的属性的对象）访问它们。

许多容器有几个共同的成员函数，并且共享功能。决定使用哪种类型的容器来满足特定需求通常不仅仅取决于容器提供的功能，还取决于其某些成员的效率（复杂性）。对于序列容器来说尤其如此，它在插入/删除元素和访问它们之间的复杂性上提供了不同的权衡。

顺序容器

顺序容器实现能按顺序访问的数据结构。

array (C++11)	静态的连续数组 (类模板)
vector	动态的连续数组 (类模板)
deque	双端队列 (类模板)
forward_list (C++11 起)	单链表 (类模板)
list	双链表 (类模板)

关联容器

关联容器实现能快速查找（*O(log n)* 复杂度）的数据结构。

set	唯一键的集合，按照键排序 (类模板)
map	键值对的集合，按照键排序，键是唯一的 (类模板)
multiset	键的集合，按照键排序 (类模板)
multimap	键值对的集合，按照键排序 (类模板)

无序关联容器 (C++11 起)

无序关联容器提供能快速查找（均摊 *O(1)* ，最坏情况 *O(n)* 的复杂度）的无序（哈希）数据结构。

unordered_set (C++11 起)	唯一键的集合，按照键生成散列 (类模板)
unordered_map (C++11 起)	键值对的集合，按照键生成散列，键是唯一的 (类模板)
unordered_multiset (C++11 起)	键的集合，按照键生成散列 (类模板)
unordered_multimap (C++11 起)	键值对的集合，按照键生成散列 (类模板)

容器适配器

容器适配器为顺序容器提供了不同的接口。

stack	适配一个容器以提供栈（LIFO 数据结构） (类模板)
queue	适配一个容器以提供队列（FIFO 数据结构） (类模板)
priority_queue	适配一个容器以提供优先级队列 (类模板)
flat_set (C++23)	调整容器以提供按键排序的唯一键集合 (类模板)

flat_map (C++23)	调整容器以提供按唯一键排序的键值对集合 (类模板)
flat_multiset (C++23)	调整容器以提供按关键字排序的关键字集合 (类模板)
flat_multimap (C++23)	调整容器以提供按关键字排序的键值对集合 (类模板)

迭代器失效

只读方法决不会使迭代器或引用失效。修改容器内容的方法可能会使迭代器和/或引用失效，如下表所示。

类别	容器	插入后……		擦除后……		条件
		迭代器有效?	引用有效?	迭代器有效?	引用有效?	
顺序容器	array	N/A		N/A		
	vector	否		N/A		插入更改容量
		是		是		在被修改元素前
		否		否		在被修改元素处或元素后
	deque	否	是	是，除了被擦除元素		修改首元素或尾元素
			否	否		只修改中间元素
	list	是		是，除了被擦除元素		
	forward_list	是		是，除了被擦除元素		
关联容器	set multiset map multimap	是		是，除了被擦除元素		
无序关联容器	unordered_set unordered_multiset unordered_map unordered_multimap	否	是	N/A		插入导致重哈希
		是		是，除了被擦除元素		无重哈希

本节未完成
原因： iterators from C++23 adaptors

此处**插入**指代任何添加一或多个元素到容器的方法，而**擦除**指代任何从容器移除一或多个元素的方法。

- 插入方法的例子是 `std::set::insert`、`std::map::emplace`、`std::vector::push_back` 和 `std::deque::push_front`。
  - 注意 `std::unordered_map::operator[]` 也算，因为它可能插入元素到映射中。 (C++11 起)
- 擦除方法的例子是 `std::set::erase`、`std::vector::pop_back`、 `std::deque::pop_front` 和 `std::map::clear` 。
  - `clear` 会使所有迭代器和引用失效。因为它会擦除所有元素，这在技术上遵照上述规则。

除非另有规定（显式地或通过根据其他函数定义函数）， 否则将容器作为参数传递给库函数不会使迭代器失效或更改容器内对象的值。

尾后迭代器（past-the-end iterator）需要特别留意。通常像指向未被擦除元素的正常迭代器一般使此迭代器失效。所以 `std::set::end` 永远不会失效，`std::unordered_set::end` 只有在重哈希（rehash）时会失效 (C++11 起)， `std::vector::end` 总是会失效（因为它始终在被修改元素后出现），以此类推。

有一个例外：删除 `std::deque` 末元素的擦除操作会使尾后迭代器失效，尽管它不是容器的被擦除元素（或者说根本不是元素）。与 `std::deque` 迭代器的通用规则结合后，最终结果是修改操作中只有“删除首元素”（而不是“删除末元素”） *不会* 使`std::deque::end` 失效。

线程安全

1. 能同时在不同容器上由不同线程调用所有容器函数。更广泛而言， C++ 标准库函数不读取能通过其他线程访问的对象，除非这些对象能直接或间接地经由函数参数，包含 this 指针访问。
2. 能同时在同一容器上由不同线程调用 `const` 成员函数。而且，成员函数 `begin()`、`end()`、`rbegin()`、`rend()`、`front()`、`back()`、`data()`、`find()`、`lower_bound()`、`upper_bound()`、`equal_range()`、`at()` 和除了关联容器中的 `operator[]` 对于线程安全的目标表现如同 `const` （即它们也能同时在同一容器上由不同线程调用）。更广泛而言，C++ 标准库函数不会修改对象，除非这些对象能直接或间接地经由函数参数，包含 this 指针访问。 (C++11 起)
3. 同一容器中不同元素能由不同线程同时修改，除了 `std::vector<bool>` 的元素（例如， `std::future` 对象的 vector 能从多个线程接收值）。
4. 迭代器操作（例如自增迭代器）读但不修改底层容器，而且能与同一容器上的其他迭代器操作同时由 const 成员函数执行。会使任何迭代器失效的容器操作都会修改容器，且不能与任何在既存迭代器上的操作同时执行，即使这些迭代器尚未失效。
5. 同一容器上的元素可以同时由不指定为访问这些元素的函数修改。更广泛而言， C++ 标准库函数不间接读取能从它的参数访问的对象（包含容器的其他对象），除非其规定要求如此。
6. 任何情况下，容器操作（还有算法，或其他 C++ 标准库函数）可于内部并行化，只要不更改用户可见的结果（例如 `std::transform` 可并行化，但指定了按顺序观览序列的每个元素的 `std::for_each` 不行）

函数表格

注意：`std::basic_string` 不被标准视为容器，但由于其相似性，其行为与容器非常相似。为方便起见，此处将其列为“伪容器”（Pseudo container）。

- C++03 起存在的函数
- C++11 起存在的函数
- C++17 起存在的函数
- C++20 起存在的函数
- C++23 起存在的函数

成员函数表格

		伪容器	顺序容器					关联容器			
头文件		<string>	<array>	<vector>	<deque>	<forward_list>	<list>	<set>		<map>	
容器		basic_string	array	vector	deque	forward_list	list	set	multiset	map	mult
	(构造函数)	basic_string	(隐式)	vector	deque	forward_list	list	set	multiset	map	mult
	(析构函数)	~basic_string	(隐式)	~vector	~deque	~forward_list	~list	~set	~multiset	~map	~mult
	operator=	operator=	(隐式)	operator=	operator=	operator=	operator=	operator=	operator=	operator=	opera
	assign	assign		assign	assign	assign	assign				
	assign_range	assign_range		assign_range	assign_range	assign_range	assign_range				
迭代器	begin	begin	begin	begin	begin	begin	begin	begin	begin	begin	beg
	cbegin	cbegin	cbegin	cbegin	cbegin	cbegin	cbegin	cbegin	cbegin	cbegin	cbe
	end	end	end	end	end	end	end	end	end	end	en
	cend	cend	cend	cend	cend	cend	cend	cend	cend	cend	ce
	rbegin	rbegin	rbegin	rbegin	rbegin		rbegin	rbegin	rbegin	rbegin	rbe
	crbegin	crbegin	crbegin	crbegin	crbegin		crbegin	crbegin	crbegin	crbegin	crbe
	rend	rend	rend	rend	rend		rend	rend	rend	rend	re
	crend	crend	crend	crend	crend		crend	crend	crend	crend	cre
元素访问	at	at	at	at	at					at	
	operator[]	operator[]	operator[]	operator[]	operator[]					operator[]	
	data	data	data	data							
	front	front	front	front	front	front	front				
	back	back	back	back	back		back				
容量	empty	empty	empty	empty	empty	empty	empty	empty	empty	empty	emp
	size	size	size	size	size		size	size	size	size	si
	max_size	max_size	max_size	max_size	max_size	max_size	max_size	max_size	max_size	max_size	max_
	resize	resize		resize	resize	resize	resize				
	capacity	capacity		capacity							
	reserve	reserve		reserve							
	shrink_to_fit	shrink_to_fit		shrink_to_fit	shrink_to_fit						
修改器	clear	clear		clear	clear	clear	clear	clear	clear	clear	cle
	insert	insert		insert	insert	insert_after	insert	insert	insert	insert	ins
	insert_range	insert_range		insert_range	insert_range	insert_range_after	insert_range	insert_range	insert_range	insert_range	insert
	insert_or_assign									insert_or_assign	
	emplace			emplace	emplace	emplace_after	emplace	emplace	emplace	emplace	empl
	emplace_hint							emplace_hint	emplace_hint	emplace_hint	emplac
	try_emplace									try_emplace	
	erase	erase		erase	erase	erase_after	erase	erase	erase	erase	era
	push_front				push_front	push_front	push_front				
	prepend_range				prepend_range	prepend_range	prepend_range				
	emplace_front				emplace_front	emplace_front	emplace_front				
	pop_front				pop_front	pop_front	pop_front				
	push_back	push_back		push_back	push_back		push_back				
	append_range	append_range		append_range	append_range		append_range				
	emplace_back			emplace_back	emplace_back		emplace_back				
	pop_back	pop_back		pop_back	pop_back		pop_back				
	swap	swap	swap	swap	swap	swap	swap	swap	swap	swap	sw
	merge					merge	merge	merge	merge	merge	mer
	extract [1]							extract	extract	extract	extr
链表操作	splice					splice_after	splice				
	remove					remove	remove				
	remove_if					remove_if	remove_if				
	reverse					reverse	reverse				
	unique					unique	unique				
	sort					sort	sort				
桶和哈希	begin(size_type)										
	cbegin(size_type)										
	end(size_type)										
	cend(size_type)										
	bucket_count										
	max_bucket_count										
	bucket_size										
	bucket										
	load_factor										
	max_load_factor										
	rehash										
查找	count							count	count	count	cou
	find	find						find	find	find	fi
	contains	contains						contains	contains	contains	cont
	lower_bound							lower_bound	lower_bound	lower_bound	lower_
	upper_bound							upper_bound	upper_bound	upper_bound	upper_
	equal_range							equal_range	equal_range	equal_range	equal_
观察器	key_comp							key_comp	key_comp	key_comp	key_
	value_comp							value_comp	value_comp	value_comp	value
	hash_function										
	key_eq										
分配器	get_allocator	get_allocator		get_allocator	get_allocator	get_allocator	get_allocator	get_allocator	get_allocator	get_allocator	get_all
适配器	extract [2]										
	replace										
容器		basic_string	array	array	vector	deque	forward_list	list	set	multiset	ma
头文件		<string>	<array>	<vector>	<deque>	<forward_list>	<list>	<set>		<map>	
		顺序容器						关联容器			

- 注意：两个不同的 `extract` 行中的函数具有不同的含义和语法：

1. ↑ 例如， `node_type extract(const_iterator)` 或 `node_type extract(Key&)`
2. ↑ 例如， `container_type extract() &&`

非成员函数表

		伪容器	顺序容器					关联容器					
头文件		<string>	<array>	<vector>	<deque>	<forward_list>	<list>	<set>		<map>		<unorde	
容器		basic_string	array	vector	deque	forward_list	list	set	multiset	map	multimap	unordered_set	un
非成员函数	operator==	operator==	operator==	operator==	operator==	operator==	operator==	operator==	operator==	operator==	operator==	operator==	
	operator!= (C++20 中移除)	operator!=	operator!=	operator!=	operator!=	operator!=	operator!=	operator!=	operator!=	operator!=	operator!=	operator!=	
	operator< (C++20 中移除)	operator<	operator<	operator<	operator<	operator<	operator<	operator<	operator<	operator<	operator<		
	operator<= (C++20 中移除)	operator<=	operator<=	operator<=	operator<=	operator<=	operator<=	operator<=	operator<=	operator<=	operator<=		
	operator> (C++20 中移除)	operator>	operator>	operator>	operator>	operator>	operator>	operator>	operator>	operator>	operator>		
	operator>= (C++20 中移除)	operator>=	operator>=	operator>=	operator>=	operator>=	operator>=	operator>=	operator>=	operator>=	operator>=		
	operator<=>	operator<=>	operator<=>	operator<=>	operator<=>	operator<=>	operator<=>	operator<=>	operator<=>	operator<=>	operator<=>		
	swap	swap	swap	swap	swap	swap	swap	swap	swap	swap	swap	swap	
	erase	erase		erase	erase	erase	erase						
	erase_if	erase_if		erase_if	erase_if	erase_if	erase_if	erase_if	erase_if	erase_if	erase_if	erase_if	
容器		basic_string	array	array	vector	deque	forward_list	list	set	multiset	map	multimap	
头文件		<string>	<array>	<vector>	<deque>	<forward_list>	<list>	<set>		<map>		<unorde	
		顺序容器					关联容器					无	

<、 <=、 >、 >= 及 != 运算符分别从 operator<=> 与 operator== 合成。 (C++20 起)

缺陷报告

下列更改行为的缺陷报告追溯地应用于以前出版的 C++ 标准。

缺陷报告	应用于	出版时的行为	正确行为
LWG 51 (https://cplusplus.github.io/LWG/issue51)	C++98	容器迭代器可能会由于任意库操作而失效	只有在指定情况下会失效

参阅

C++ 已命名的要求（requirement）：

- *容器 (Container)*
- *序列容器 (SequenceContainer)*
- *连续容器 (ContiguousContainer)*
- *可逆容器 (ReversibleContainer)*
- *关联容器 (AssociativeContainer)*
- *知分配器容器 (AllocatorAwareContainer)*
- *无序关联容器 (UnorderedAssociativeContainer)*

valarray	数值数组，数组掩码和数组切分 (类模板)
basic_string	存储并操作字符序列 (类模板)
basic_string_view (C++17)	只读的字符串视图 (类模板)
span (C++20)	对象的连续序列上的无所有权视图 (类模板)
mdspan (C++23)	多维非拥有数组视图 (类模板)

来自“https://zh.cppreference.com/mwiki/index.php?title=cpp/container&oldid=76392”