调用empty而不是检查size（）是否为零。

图形用户界面, 文本, 应用程序

描述已自动生成

当处理标准序列容器时，与范围成员函数相比，单元素成员函数的应用对内存分配器有更多的要求，更频繁地复制对象，和/或执行冗余操作。

* 在大多数情况下，范围插入函数（如 insert）比单元素插入函数更高效，因为它可以一次移动多个元素。
* 这种效率优势依赖于迭代器的功能，特别是前向迭代器的功能。
* 当使用输入迭代器时，范围插入函数的效率优势可能不存在，因为它只能一次一个位置地移动元素。

C++20引入了范围，对迭代器进一步抽象化，隐藏了细节。

std：：remove（）或std：：unique（）实际上并不从容器中删除元素（尽管它们的名称）。相反，它将应该保留的元素移动到容器的前面，然后返回一个sentinel，

C++20在头文件中添加了新版本的std：：erase（）和std：：erase\_if（）函数，<vector>可以立即从vector中擦除值，而无需先调用remove（）再调用erase（）。

**使用受限算法**

C++20 引入的 std::ranges 命名空间下的受限算法相较于 std 命名空间下基于迭代器的算法具有一些优势。受限算法具备以下特点：

* 支持投影，这简化了元素的自定义比较。
* 支持直接使用范围，而不是迭代器对。无需分别传递 begin() 和 end() 迭代器作为参数。
* 易于正确使用，并且由于受到 C++ 概念的约束，在编译时能提供描述性的错误信息。

我建议优先使用受限算法而非基于迭代器的算法。

你可能已经注意到，本书在很多地方使用了基于迭代器的算法。

标准库算法，安全高效。