1 顺序容器概述

参考文档：顺序容器类型

六种顺序容器：vector,deque,list,forward\_list,array,string

这些容器在两方面性能折中：1，向容器添加或从容器删除元素的代价；2，非顺序访问容器中元素的代价

array是固定大小的

string和vector将元素保存在连续的内存空间中。因此，由下标来计算地址是非常快速的，但是在中间位置添加或删除元素就会非常耗时。

list和forward\_list设计目的是令容器任何位置的添加和删除操作都很快速，代价是不支持元素的随机访问。与其他容器相比额外内存开销大。

deque是一个更为复杂的数据结构，支持快速的随机访问。在deque的中间位置添加或删除元素代价很高。但是，在deque的两端添加或删除元素很快。

array是一种更安全，更容易使用的数组类型。forward\_list的设计目标是达到与最好的手写的单向链表数据结构相当的性能。不支持size操作。

通常，使用vector是最好的选择，除非你有很好的理由选择其他容器。

参考文档：选择容器的基本原则

2 容器库概览

参考文档：容器操作

forward\_list迭代器不支持递减运算符(--)。

begin和end有多个版本：带r的版本返回反向迭代器。begin,rbegin,cbegin和crbegin。

参考文档：容器定义和初始化

为了创建一个容器为另一个容器的拷贝，两个容器的类型及其元素类型必须匹配，不过，当传递迭代器参数来拷贝一个范围时，就不要求容器类型是相同的了。

vector<const char\*> articles;forward\_list<string> words(articles.begin(), articles.end());

与内置数组一样，标准库array的大小也是类型的一部分，array<int,42> arr;

交换c1和c2：swap(c1, c2)或者c1.swap(c2)，swap通常比从c2向c1拷贝元素快很多。

参考文档：容器赋值运算

赋值函数assign

容器大小操作:size()，empty()和max\_size()

3 顺序容器操作

参考文档：向顺序容器添加元素的操作

在迭代器p指向的元素之前插入一个或多个元素c.insert(p,b,e);

当我们使用各种插入操作时，必须记得不同容器使用不同的策略来分配空间，而这些策略直接影响性能。

当调用push或insert时，我们将元素对象拷贝到容器中，而当我们调用emplace时，则是将参数传递给元素类型的构造函数，直接构造元素。

程序有两种不同方式来获取c中的首元素和尾元素的引用，直接的方法是调用front和back。间接的方法是通过解引用begin和end返回的迭代器。

在容器中访问元素的成员函数front,back,下标和at返回的都是引用。

参考文档：顺序容器的删除操作

删除的方法：pop\_back(),pop\_front(),erase(),clear()

vector和string不支持push\_front和pop\_front

编程：删除list中的奇数。

添加删除vector，string或deque元素的循环程序必须考虑迭代器，引用和指针可能失效的问题。

编程：删除偶数元素，复制每个奇数元素

不要保存end返回的迭代器。每次使用end()求值。

4 vector对象是如何增长的

vector和string中的元素是连续存储的，向它们添加元素，如果没有空间容纳新元素，必须分配新的空间，将已有的元素移到新空间中，释放旧空间。

参考文档：容器大小管理操作

vector的实现采用的策略似乎是在每次需要分配新内存空间时将当前容量翻倍。

5 额外的string操作

substr操作返回一个string，它是原始string的一部分或全部的拷贝。

构造string的其他方法

改变string的其他方法

string搜索操作

compare函数

数值转换

6 容器适配器

除了顺序容器外，标准库还定义了三个顺序容器适配器：stack,queue和priority\_queue。

每个适配器都定义两个构造函数：默认构造函数创建一个空对象，接受一个容器的构造函数拷贝该容器来初始化适配器。

默认情况下，stack和queue都是基于deque实现的，priority\_queue实在vector之上实现的。

stack可以使用除了array和forward\_list之外的任何容器类型来构造。

queue可以构造于list和deque上，但不能基于vector构造。

priority\_queue可以构造于vector或deque之上，但不能基于list构造。

参考文档：栈的操作

参考文档：queue和priority\_queue的操作