9 顺序容器

1 顺序容器概述

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

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。它是固定大小的。

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。

2 容器库概览

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。

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());

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。

赋值函数assign

容器大小操作:size()，empty()和max\_size()

3 顺序容器操作

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。

顺序容器的删除操作：pop\_back(), pop\_front(), erase(), clear()

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。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

5 额外的string操作

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。