Ch.8 STL的使用

2017 / 11

STL

- Standard Template Library
- 能过原始C++封装的工具
- 是C++标准库的一部分
- 强调通用性，要根据实际的情况使用

六大组件

- 容器
- 迭代器
- 算法
- 函数对象（仿函数）
- 分配器
- 适配器

容器—非常常用

- vector
- list
- map/set
- stack
- queue
- priority_queue

容器一会用到

- deque
- multmap
- set/multiset
- hashmap/hashtable(非标准)
- unordered_map(C++新标准)

STL要点

- 所有容器中存放的都是值而非引用，即容器进行安插操作时内部实施的是拷贝操作。因此容器的每个元素必须能够被拷贝
- 如果希望存放的不是副本，容器元素只能是指针

Vector

- 动态数组
- 随机访问
- 空间随着插入新的元素自动进行扩充
- 大小（size）和容量（capacity），size是当前已经存储的元素个数，capacity是整体的容量就多少

Vector扩充

- Vector插入新元素时，size() == capacity()即已经满了
- 申请一个新的更大的空间（连续的）
- 把原来的空间内容全部拷贝到新空间中
- 将原来的空间归还

迭代器

- 一个通用的指针
- 可遍历容器内全部或部分元素
- 指出容器中的一个特定位置

迭代器

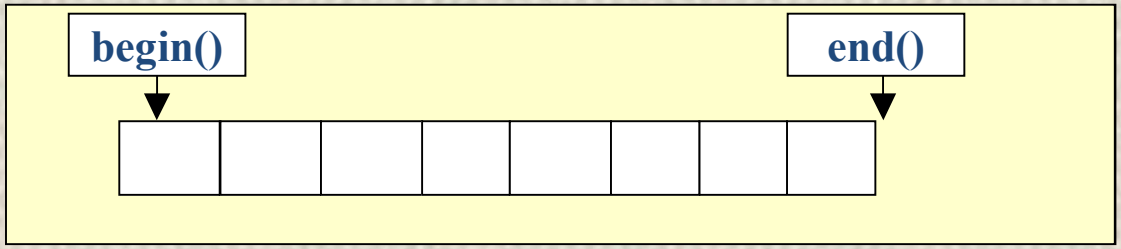
- 所有容器都提供获得迭代器的函数

操作	效果
begin()	返回一个迭代器，指向第一个元素
end()	返回一个迭代器，指向最后一个元素之后

操作	效果
*	返回当前位置上的元素值。如果该元素有成员，可以通过迭代器以**operator ->**取用
++	将迭代器前进至下一元素
==和!=	判断两个迭代器是否指向同一位置
=	为迭代器赋值（将所指元素的位置赋值过去）

Vector及其迭代器

- Std::vector<int>::iterator = ?
- [begin, end)



Vector操作

操作	效果
begin()	返回一个迭代器，指向第一个元素
end()	返回一个迭代器，指向最后一个元素之后
rbegin()	返回一个逆向迭代器，指向逆向遍历的第一个元素
rend()	返回一个逆向迭代器，指向逆向遍历的最后一个元素

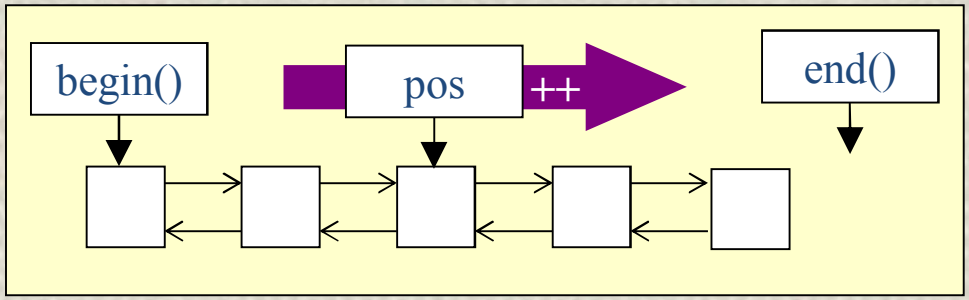
操作	效果
c.insert(pos,e)	在**pos**位置插入元素**e**的副本，并返回新元素位置
c.insert(pos,n,e)	在**pos**位置插入**n**个元素**e**的副本
c.insert(pos,beg,end)	在**pos**位置插入区间**[beg;end]**内所有元素的副本
c.push_back(e)	在尾部添加一个元素**e**的副本

Vector操作

操作	效果
c.pop_back()	移除最后一个元素但不返回最后一个元素
c.erase(pos)	删除**pos**位置的元素，返回下一个元素的位置
c.erase(beg,end)	删除区间**[beg;end]**内所有元素，返回下一个元素的位置
c.clear()	移除所有元素，清空容器
c.resize(num)	将元素数量改为**num**（增加的元素用**defalut**构造函数产生，多余的元素被删除）
c.resize(num,e)	将元素数量改为**num**（增加的元素是**e**的副本）

List和其迭代器

- Std::vector<int>::iterator = alist.begin();
- [begin, end)



List的一些操作

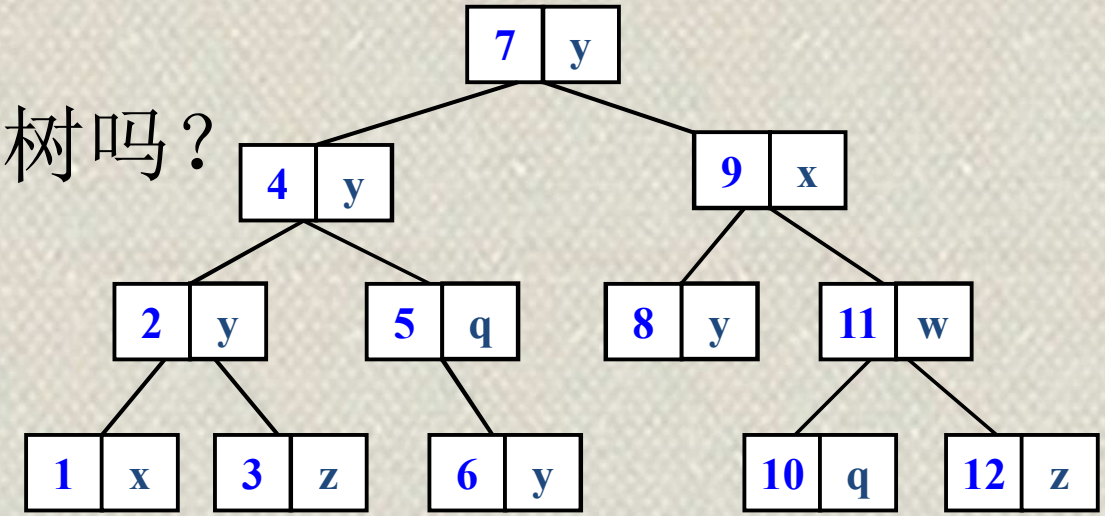
操作	效果
c.insert(pos,e)	在**pos**位置插入**e**的副本，并返回新元素位置
c.insert(pos,n,e)	在**pos**位置插入**n**个**e**的副本
c.insert(pos,beg,end)	在**pos**位置插入区间**[beg;end]**内所有元素的副本
c.push_back(e)	在尾部添加一个**e**的副本
c.push_front(e)	在头部添加一个**e**的副本

List操作

操作	效果
c.pop_back()	移除最后一个元素但不返回
c.pop_front()	移除第一个元素但不返回
c.erase(pos)	删除**pos**位置的元素，返回下一个元素的位置
c.remove(val)	移除所有值为**val**的元素
c.remove_if(op)	移除所有“**op(val)==true**”的元素
c.erase(from, to)	删除区间**[beg;end]**内所有元素，返回下一个元素的位置
c.clear()	移除所有元素，清空容器
c.resize(num)	将元素数量改为**num**（多出的元素用**defalut**构造函数产生）
c.resize(num,e)	将元素数量改为**num**（多出的元素是**e**的副本）

关联容器

- map
- multimap
- 还记得二叉排序树吗？
- 红黑树结构



树map

- 元素包含两部分(key,value)，key和value可以是任意类型
- 根据元素的key自动对元素排序
- 可以通过operator []直接存取元素值
- map中不允许key相同的元素，multimap允许key相同的元素
- 有序

Hash map

- Unordered_map
- 哈希结构

Stack

- 后进先出（LIFO）
- push(value)—将元素压栈
- top()—返回栈顶元素的引用，但不移除
- pop()—从栈中移除栈顶元素，但不返回

Queue

- 先进先出（FIFO）
- push(e)—将元素置入队列
- front()—返回队列头部元素的引用，但不移除
- back()—返回队列尾部元素的引用，但不移除
- pop()—从队列中移除元素，但不返回