

STL 容器

# allocator

allocator类将对象分配和对象构造分开。当allocator对象分配内存的时候，它会分配适当大小并排列成保存给定类型对象的空间。

操作：

`allocator<T> a;` 定义为a的allocator对象可分配内存或构造T类型对象。

`a.allocate(n);` 分配原始的构造内存以保存T类型的n个对象

`a.deallocate(p, n)` 释放内存，在名为p的T\*指针中包含的地址处保存T类型的n个对象。

a. `construct(p, t)` 在  $T^*$  指针  $p$  所指向的内存中构造一个新元素。运行  $T$  类型的复制构造函数用  $t$  初始化该对象。

a. `destory(p)` 运行  $T^*$  指针  $p$  所指向的对象的析构函数。

vector deque array 访问元素的时候 ~~直接~~ 随机访问就可以 list 之类的 range-base 循环……感觉 insert 和 emplace 在这里就不太好用了。

```
list2.splice(find(list2.begin(), list2.end(),  
                 3), list1);
```

第 3 个元素添加 list1;

## remove 和 erase 的区别:

vector 中的 remove 的作用是将等于 value 元素放到 vector 的尾部, 但并不减少 vector 的 size.

vector 中的 erase 的作用是删除掉某个位置 position 或一段区域 (begin, end) 中的元素, 减少其 size.

list 容器中的 remove 成员函数, 原型是 `void remove(const value_type& val);`

他的作用是删除 list 中值与 val 相同的节点。释放该节点的资源。

而 list 容器中的 erase 成员函数, 原型是 `iterator erase (iterator position);`

作用是删除 position 位置的节点。这也是与 remove 不同的地方。

考虑到 `list::erase` 是与位置有关, 故 erase

还存在 API: `iterator erase (iterator first, iterator last);`

对于 set 来说, 只有 `erase` API, 没有 `remove` API。

`erase` 的作用是把符合要求的元素都删掉。

- 1) `void erase (iterator position);`
- 2) `size_type erase (const value_type & val);`
- 3) `void erase (iterator first, iterator last);`

综上所述, `erase` 一般是要释放资源, 真正删除元素的, 而 `remove` 主要用在 `vector` 中, 用于将不符合要求的元素移到容器尾部, 而并不删除不符合要求的元素。

Bucket 是 Hash table 内元素的虚拟子数组。

Hash table 由多个 Bucket 组成, Bucket 以 Hash key 值为索引, 每个 Bucket 存放着所有 Hash key 相同的 (key, value)