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;定义名为a的allocator对象于分配内存或构造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 arry 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的

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。