# 一 何为vector

# 二 vector常用函数

## 2.1 push\_back

a) 将元素放在矢量(vector)的末尾.

b) 负责矢量的内存管理, 为新的成员分配内存.

c) 只要能够取得足够的内存, 就会给新成员分配内存了.

## 2.2 erase

1. 删除给定区间的元素.
2. 接受2个迭代器参数. 这两个迭代器规定了要删除的区间.
3. 第一个迭代器规定了删除的第一个元素.
4. 第二个迭代器规定了删除的最后一个+1的元素(并不包含该迭代器指向的那个)

如下表示删除scores的第一个和第二个.

也就是 scores.begin()和 scores.begin()+1 的这两个元素.

|  |
| --- |
| scores.erase(scores.begin(), scores.begin()+2) |

# 三 map

## 3.1 unordered\_map和map

1. Unordered\_map和map都是key-value格式
2. Unordered\_map内存存储(元素存储)是无序的. Map的元素是有序的(按照二叉树搜索存储的), 用中序遍历可以获得(得到)有序遍历.
3. 使用时map的key需要定义operator<。而unordered\_map需要定义hash\_value函数并且重载operator==。但是很多系统内置的数据类型都自带这些.
4. 如果需要元素自动排序使用map, 如果不需要使用unordered\_map.

## 3.2 unordered\_map用法

a) 使用

|  |
| --- |
| #include<tr1/unordered\_map>//在unordered\_map之前加上tr1库名，  using namespace std::tr1;//与此同时需要加上命名空间 |

### 查找元素是否存在

    若有unordered\_map<int, int> mp;查找x是否在map中

    方法1:  若存在  mp.find(x)!=mp.end()

    方法2:  若存在  mp.count(x)!=0

### 插入数据

    map.insert(Map::value\_type(1,"Raoul"));

### 遍历map

|  |
| --- |
| unordered\_map<key,T>::iterator it;      (\*it).first;        //the key value      (\*it).second   //the mapped value      for(unordered\_map<key,T>::iterator iter=mp.begin();iter!=mp.end();iter++)            cout<<"key value is"<<iter->first<<" the mapped value is "<< iter->second;      也可以这样      for(auto& v : mp)          print v.first and v.second |

### lower\_bound

map中的lower\_bound和upper\_bound的意思其实很简单，就两句话：

**map::lower\_bound(key):返回map中第一个大于或等于key的迭代器指针 (>=, 不小于)**

**map::upper\_bound(key):返回map中第一个大于key的迭代器指针 (>, 大于)**

所以，理解这两个函数请不要按照字面意义思考太复杂，因为仅仅是不小于（lower\_bound）和大于（upper\_bound）这么简单。

# 四 multimap和map区别

## Multimap和map大部分相同.

其区别是:

1. Multimap的value可以有多个value, 对应相同的key
2. 这里的相同并不是说key值有一个, 是多个key值,但是着多个key值是相同的值..
3. Map不允许有key值相同的情况.
4. 另外multimap和map都是默认(系统)排序的, 默认从小到大(可重载比较函数)

## Second含义

对于key-value格式.

1. First代表key.
2. Second代表value