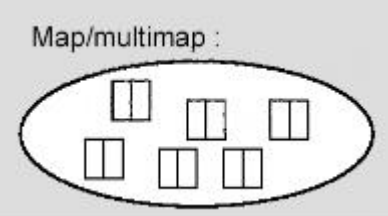
https://blog.csdn.net/bat603/article/details/1456141

map：关联容器，按照{键，值}方式组成集合，按照键组织成一棵红黑树，查找的时间复杂度O(logN)，其中键不允许重复。由于map的每个数据对应红黑树上的一个节点，这个节点在不保存你的数据时，是占用16个字节的，一个父节点指针，左右孩子指针，还有一个枚举值（标示红黑的，相当于平衡二叉树中的平衡因子），内存不连续，每个节点的内存都是单独申请的；



特点：

1、键和值分开（模版有两个参数，前面是键后面是值）；

2、键唯一，map中存在键-值对时，插入相同键会失败；

3、元素默认按键的升序排列，系统默认 '<' 号无法处理键值的大小比较时，map的insert编译会失败，此时需要重载 '<' 比较符号；

迭代器有效性：

插入，迭代器不会失效。删除，指向被删除节点迭代器失效，其他迭代器均有效；

头文件：

```
#include<map>
```

基本接口：

```
map.size();          //返回容器中，元素的数量；
map.max_size();      //容器能够存储的元素最大数量，受限于内存；
map.empty();         //容器是否空，空返回true，非空返回false；

//容器存在键值k，返回键值k对应value的引用；不存在时，插入键-值对，键为k，value使用默认值
map[const key_type& k];
//返回容器中，键值k对应的value的引用，若容器中键值k不存在，异常奔溃；
map.at(const key_type& k);

//可以通过迭代器修改键值对中的值value，it->second = xxx;
map.begin();         //返回指向第一个元素的迭代器
map.end();           //返回指向最后一个元素下一个位置的迭代器

//查找键值为k的键值对， 找到时，返回指向该键值对的迭代器
```

map.find(const key_type& k);
//判断map容器中是否存在键值为k的键值对，存在时返回1，不存在时返回0；
map.count(const key_type& k);

//map中键值唯一性，插入时会检查是否存在相同的键值。存在相同键值时，插入失败，返回map中该键对应的键值对；
map.insert(const value_type& val);
map.insert(iterator position, const value_type& val);
map.insert(InputIterator first, InputIterator last);

//删除容器中元素，仅指向被删除元素的迭代器失效，其他迭代器全部有效；所以可以使用erase(it++)进行安全删除操作；
map.erase(iterator position);
map.erase(iterator first, iterator last);
map.erase(const key_type& k);

```
1 //use iterator to delete all element safely
2 for (map<int, string>::iterator iter =map.begin(); iter != list.end();){
3  map.erase(iter++);
4 }
```

//删除容器中得所有元素，释放容器中对应元素占用的内存
map.clear();