**P475 3**

template <class K, class E>

bool LinearProbing<K, E>::Delete(const K& k) {

int i = h(k); // home bucket

int j;

for (j = i; ht[j] && ht[j]->first != k; j = (j + 1) % b) {

// 跳过删除的和不匹配的条目

}

if (!ht[j]) return false; // 元素未找到

delete ht[j]; // 删除键值对

ht[j] = DELETED; // 标记为已删除

return true;

}

template <class K, class E>

pair<K, E>\* LinearProbing<K, E>::Get(const K& k) {

int i = h(k); // home bucket

int j;

for (j = i; ht[j] && (ht[j] == DELETED || ht[j]->first != k); j = (j + 1) % b) {

// 跳过已删除的条目，但搜索时不停留在这些条目上

}

if (!ht[j] || ht[j] == DELETED) return 0; // 元素未找到或槽位已删除

if (ht[j]->first == k) return ht[j]; // 元素找到

return 0; // 元素未找到

}

在delete函数中，我们不能将被删除的节点简单设置为空，因为先前别的节点可能因为冲突被存放到后续的空间中，如果我们简单删除了当前节点，可能会造成部分节点被误判为不存在的情况。因此需要设置被删除节点为DELETED（一个特殊标记），来表示这个位置已经被删除，当在执行查找函数的时候，我们需要跳过这个位置继续查找后续空间，在执行插入函数的时候，我们可以把符合hash算法的元素直接插入到DELETED的位置。