# 概述

计算机科学领域使用符号表（symbol table）来表示“字典”。

符号表是将值（value）与键（key）关联起来的数据结构，它支持下列操作：

1. 搜索特定的键名，以判断符号表中是否存在这个键；
2. 获取与某个键名相关联的属性值；
3. 修改与某个键名相关联的属性值；
4. 将新的键名及与该键相关联的属性值插入符号表；
5. 删除某个键名及其属性值。

# 实现

可以实现符号表的方式：

1. 无序的数组实现

这种实现方式只需要维护一个普通的数组就可以。

其查找、插入及删除在最差的情况下时间复杂度均为O(n)。

1. 有序的数组实现

这种实现方式需要将键与值存放在有序的数组中。

由于元素是按照顺序存放在数组中的，因此可以采用简单的二分搜索法查找，这需要花费O(logn)级别的时间。而插入与删除操作在最差情况下的复杂度为O(n)。

1. 无序的链表实现

这种链表的每个结点都有两个字段，分别用来存放键与值。

查找、插入与删除操作在最差的情况下复杂度为O(n)。

1. 有序的链表实现

如果用这种方法实现，那么向链表中插入新键的时候，就必须按照顺序，把该键调整到它应该在的位置上。

其查找、插入及删除操作在最差的情况下复杂度为O(n)。

1. 二叉搜索树实现

这种办法的好处在于，开发者所要编写的代码比较少，而且查找操作的速度比较快（一般来说，只需要耗费O(logn)级别的时间）。

1. 平衡的二叉搜索树实现

这是在二叉搜索树基础上的扩充。其查找、插入及删除操作在最差的情况下都需要花费O(logn)级别的时间。

1. 三元搜索树实现
2. 哈希技术实现

各种实现方式的比较：

|  |  |  |  |
| --- | --- | --- | --- |
| **实现方式** | **查找** | **插入** | **删除** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 应用