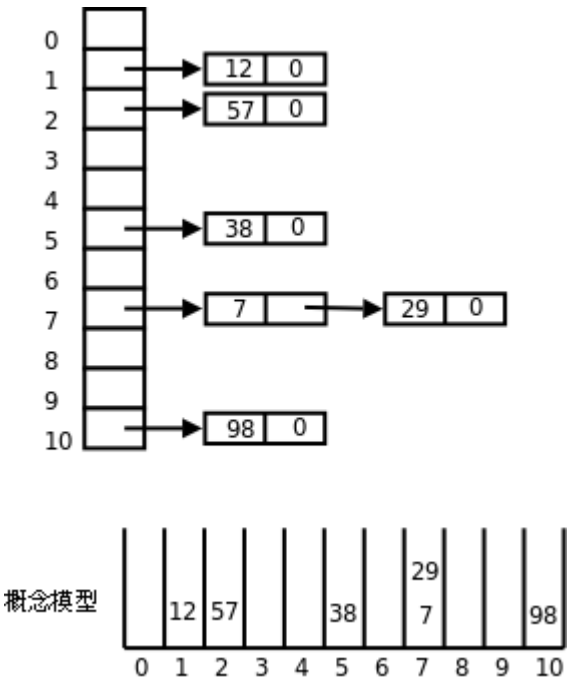
哈希表

下图示意了哈希表（Hash Table）这种数据结构。

图 . 哈希表



如上图所示，首先分配一个指针数组，数组的每个元素是一个链表的头指针，每个链表称为一个槽（Slot）。哪个数据应该放入哪个槽中由哈希函数决定，在这个例子中我们简单地选取哈希函数 h(x) = x % 11，这样任意数据 x 都可以映射成 0~10 之间的一个数，就是槽的编号，将数据放入某个槽的操作就是链表的插入操作。

如果每个槽里至多只有一个数据，可以想像这种情况下 search、insert 和 delete 操作的时间复杂度都是 O(1)，但有时会有多个数据被哈希函数映射到同一个槽中，这称为碰撞（Collision），设计一个好的哈希函

数可以把数据比较均匀地分布到各个槽中，尽量避免碰撞。如果能把 n 个数据比较均匀地分布到 m 个槽中，每个糟里约有 n/m 个数据，则 search、insert 和 delete 和操作的时间复杂度都是 O(n/m)，如果 n 和 m 的比是常数，则时间复杂度仍然是 O(1)。一般来说，要处理的数据越多，构造哈希表时分配的槽也应该越多，所以 n 和 m 成正比这个假设是成立的。

请读者自己编写程序构造这样一个哈希表，并实现 search、insert 和 delete 操作。

如果用我们学过的各种数据结构来表示 n 个数据的集合，下表是 search、insert 和 delete 操作在平均情况下的时间复杂度比较。

表 . 各种数据结构的 search、insert 和 delete 操作在平均情况下的时间复杂度比较

数据结构	search	insert	delete
数组	O(n)，有序数组折半查找是 O(lgn)	O(n)	O(n)
双向链表	O(n)	O(1)	O(1)
排序二叉树	O(lgn)	O(lgn)	O(lgn)
哈希表（n 与槽数 m 成正比）	O(1)	O(1)	O(1)