哈希（散列）查找

散列的基本思想：在记录的存储地址和它的关键码之间建立一个确定的对应关系。这样，不经过比较，依次读取就能得到所查元素的查找方法。

散列表：采用散列技术将存储在一块的连续的存储空间中，这块连续的存储空间称为散列表。

散列函数：将关键码映射为散列表中适当存储位置的函数。

散列计数一般不适用于允许多个记录有同样关键码的情况。散列方法也不适用于范围查找，换言之，在散列表中，我们不可能找到最大或最小关键码的记录，也不可能找到在某一范围内的记录。散列技术最适合回答的问题是：如果有的化，哪个记录的关键码等于待查找值。

散列技术的关键问题

1、散列函数的设计。如何设计一个简单，均匀、存储利用率高的散列函数。

2、冲突的处理。如何采取合适的处理冲突的方法来解决

散列函数

1、直接地址法。

散列函数是关键码的线性函数。

$$H(key) = a * key + b$$

适用情况是事先知道关键码，关键码集合不是很大且连续性较好。

2、除留余数法

是一种最简单也是最常用的构造散列函数的方法，并且不要求实现知道关键码的分布。散列函数为：

$$H(key) = keymodp$$

一般情况下，选p为小于或等于表长的最小素数或不包含小于20的质因子的合数。

3、数字分析法

根据关键码在各个位上的分布情况，选取分布比较均匀的若干位组成散列地址。

使适用情况是能预先估计出全部关键码的每一位上各种数字出现的频度，不同的关键码集合需要重新分析。

4、平方取中法

对关键码平方后，按散列表大小，取中间的若干位作为散列地址。适合实现不知到关键码的分布且关键码的位数不是很大。

冲突的解决

1、线性探测法

当发生冲突时，从冲突位置的下一个位置起，依次寻找空的散列地址。开放地址法处理冲突得到的散列表叫闭散列表。

$$H_i=(H(key)+d_i)\ \%\ m \quad （d_i=1，2，\ldots，m-1）$$

堆积：在处理冲突的过程中出现的非同义词之间对同一散列地址争夺的现象。

2、二次探测法

当发生冲突时，寻找下一个散列地址的公式为：

$$H_i=(H(key)+d_i)\%\ m$$

$$(d_i=1^2,\ -1^2,\ 2^2,\ -2^2,\ \ldots,\ q^2,\ -q^2 且 q\leqslant m/2)$$

3、随机探测法

当发生冲突时，下一个散列地址的位移量是一个随机数列，即寻找下一个散列地址的公式为：

$$H_i=(H(\mathrm{key})+d_i)\%\ m$$

$$(d_i 是一个随机数列，\ i=1，\ 2，\ \ldots\ldots，\ m-1)$$

4、拉链法：

将所有散列地址相同的记录即所有同义词记录存储在一个单链表中。在散列表中存储的是所有同义词字表的头指针。

用拉链法处理冲突构造的散列表叫做开散列表。开散列表不会出现堆积的现象。

例：关键码集合 {47, 7, 29, 11, 16, 92, 22, 8, 3}，散列函数为 $H(key)=key \bmod 11$，用拉链法处理冲突，构造的开散列表为：