[例] 有n = 11个数据对象的集合{18，23，11，20，2，7，27，30，42，15，34}。

符号表的大小用TableSize = 17，选取散列函数h如下：

h(key) = key mod TableSize　(求余)

| 地址 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 关键词 | 34 | 18 | 2 | 20 | | | 23 | 7 | 42 | | 27 | 11 | | 30 | | 15 | |

- 存放：

  h(18)=1，h(23)=6，h(11)=11，h(20)=3，h(2)=2，.....

  如果新插入35，h(35)=1，该位置已有对象！冲突！！

- 查找：
  - key = 22，h(22)= 5，该地址空，不在表中
  - key = 30，h(30)= 13，该地址存放是30，找到！

装填因子（Loading Factor）：设散列表空间大小为m，填入表中元素个数是n，则称α＝ n / m为散列表的装填因子

- α＝11 / 17 ≈ 0.65。

[例] 将acos、define、float、exp、char、atan、ceil、floor、clock、ctime，顺次存入一张散列表中。

散列表设计为一个二维数组Table[26][2]，2列分别代表 2个槽。

如何设计散列函数*h (key )* = ？

*h(key) = key[0] – 'a'*

acos　define　float　exp　char

atan　ceil　floor　clock　ctime

| | 槽 0 | 槽 1 |
|---|---|---|
| 0 | acos | atan |
| 1 | | |
| 2 | char | ceil |
| 3 | define | |
| 4 | exp | |
| 5 | float | floor |
| 6 | | |
| ..... | | |
| 25 | | |

如果没有溢出，

$T_{查询} = T_{插入} = T_{删除} = O(1)$

- “散列（Hashing）”的基本思想是：

（1）以关键字key为自变量，通过一个确定的函数 h（散列函数），计算出对应的函数值h(key)，作为数据对象的存储地址。

（2）可能不同的关键字会映射到同一个散列地址上，

即$h(key_i) = h(key_j)$（当$key_i \neq key_j$），称为“冲突(Collision)”。

----需要某种冲突解决策略

❖ 一个“好”的散列函数一般应考虑下列两个因素：

1. 计算简单，以便提高转换速度；
2. 关键词对应的地址空间分布均匀，以尽量减少冲突。

❖ 数字关键词的散列函数构造

1. 直接定址法

取关键词的某个线性函数值为散列地址，即

h(key) = a × key + b　　(a、b为常数)

| 地址h(key) | 出生年份(key) | 人数(attribute) |
|---|---|---|
| 0 | 1990 | 1285万 |
| 1 | 1991 | 1281万 |
| 2 | 1992 | 1280万 |
| … | …… | …… |
| 10 | 2000 | 1250万 |
| … | …… | …… |
| 21 | 2011 | 1180万 |

2. 除留余数法

散列函数为：h(key) = key mod p

例：h(key) = key % 17

| 地址 h(key) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 关键词 key | 34 | 18 | 2 | 20 | | | 23 | 7 | 42 | | 27 | 11 | | 30 | | 15 | |

- 这里：p = Tablesize = 17
- 一般，p 取素数

3. 数字分析法

分析数字关键字在各位上的变化情况，取比较随机的位作为散列地址

- 比如：取11位手机号码key的后4位作为地址：
  散列函数为：h(key) = atoi(key+7)　(char *key)

int atoi(char *s):
将类似"5678"的字符串转换为整数5678

如果关键词 key 是18位的身份证号码：

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 3 | 0 | 1 | 0 | 6 | 1 | 9 | 9 | 0 | 1 | 0 | 0 | 8 | 0 | 4 | 1 | 9 |
| 省 | | 市 | | 区（县）下属辖区编号 | | （出生）年份 | | | | 月份 | | 日期 | | 该辖区中的序号 | | | 校验 |

$h_1(key) = (key[6]-'0')\times10^4 + (key[10]-'0')\times10^3 + (key[14]-'0')\times10^2 + (key[16]-'0')\times10 + (key[17]-'0')$

$h(key) = h_1(key)\times10 + 10$ （当 key[18] = 'x'时）

或 $= h_1(key)\times10 + key[18]-'0'$ （当 key[18] 为'0'~'9'时）

4. 折叠法

把关键词分割成位数相同的几个部分，然后叠加

如： 56793542

```
  542
  793
+ 056
------
 1391
```

h(56793542) = 391

5. 平方取中法

如： 56793542

```
          56793542
        x 56793542
--------------------
3225506412905764
```

字符关键字

## ❖字符关键词的散列函数构造

1. 一个简单的散列函数——ASCII码加和法

对字符型关键词key定义散列函数如下：

$h(key) = (\Sigma key[i]) \bmod TableSize$

2. 简单的改进——前3个字符移位法

$h(key)=(key[0]\times 27^2 + key[1]\times 27 + key[2]) \bmod TableSize$

3. 好的散列函数——移位法

涉及关键词所有n个字符，并且分布得很好：

$$h(key)=\left(\sum_{i=0}^{n-1} key[n-i-1]\times 32^i\right) \bmod TableSize$$

## ❖ 如何快速计算：

h("abcde")='a'*$32^4$+'b'*$32^3$+'c'*$32^2$+'d'*32+'e'

((a×32+b)×32+c)×32+d ...

X×32 : X<<5

```
Index Hash ( const char *Key, int TableSize )
{
    unsigned int h = 0;      /* 散列函数值，初始化为0 */
    while ( *Key != '\0' )   /* 位移映射 */
        h = ( h << 5 ) + *Key++;
    return  h % TableSize;
}
```