本节主题：

哈希表的运算

哈希表的存储结构

```
#define MaxSize 100              //定义最大哈希表长度
#define NULLKEY -1   //定义空关键字值
#define DELKEY  -2   //定义被删关键字值
typedef int KeyType;          //关键字类型
typedef char * InfoType;     //其他数据类型
typedef struct
{
    KeyType key;              //关键字域
    InfoType data;            //其他数据域
    int count;                //探查次数域
} HashData;

typedef HashData HashTable[MaxSize]; //哈希表类型
```

	key	data	count
[0]	77		2
[1]	-1		0
[2]	54		1
[3]	16		1
[4]	43		1
[5]	31		1
[6]	29		3
[7]	46		1
[8]	60		1
[9]	74		1
[10]	10		1
[11]	-1		0
[12]	90		1

创建哈希表(除留余数法)

```
void CreateHT(HashTable ha,KeyType x[],int n,int m,int p)
{
   int i,n1=0;
   for (i=0; i<m; i++)
   {
      ha[i].key=NULLKEY;
      ha[i].count=0;
   }
   for (i=0; i<n; i++)
      InsertHT(ha, n1, x[i], p);
}
```

```
/*
  HashTable ha - 哈希表
  KeyType x[] - 建表的数据
  int n - 数据个数
  int m - 哈希表长度
  int p - 除留余数法中的p，
          h(k)=k mod p
*/
```

	key	data	count
[0]	77		1
[1]	-1		1
[2]	54		1
[3]	16		1
[4]	43		1
[5]	31		1
[6]	29		3
[7]	46		1
[8]	60		1
[9]	74		1
[10]	10		1
[11]	-1		1
[12]	90		1

插入及建表

```
void InsertHT(HashTable ha,int &n,KeyType k,int p)
{
   int i,adr;
   adr=k % p;
   if (ha[adr].key==NULLKEY || ha[adr].key==DELKEY)
   {
      //没有冲突时直接插入
   }
   else
   {
      //发生冲突时采用线性探查法解决冲突
   }
   n++;
}
```

```
/*
  HashTable ha - 哈希表
  int n - 哈希表中已有数据个数
  KeyType k - 要插入的数据
  int p - 除留余数法中的p，h(k)=k mod p
*/
```

```
ha[adr].key=k;
ha[adr].count=1;
```

```
i=1;
do
{
   adr=(adr+1) % p;
   i++;
}
while (ha[adr].key!=NULLKEY && ha[adr].key!=DELKEY);
ha[adr].key=k;
ha[adr].count=i;
```

删除

```
int DeleteHT(HashTable ha,int p,int k,int &n)
{
   int adr;
   adr=SearchHT(ha,p,k);
   if (adr!=-1)   //在哈希表中找到该关键字
   {
      ha[adr].key=DELKEY;
      n--;
      return 1;
   }
   else
      return 0;
}
```

```
/*
  HashTable ha - 哈希表
  int n - 哈希表中已有数据个数
  KeyType k - 要删除的数据
  int p - 除留余数法中的p，h(k)=k mod p
*/
```

查找

```
int SearchHT(HashTable ha,int p,KeyType k)
{
   int i=0,adr;
   adr=k % p;
   while (ha[adr].key!=NULLKEY && ha[adr].key!=k)
   {
      i++;
      adr=(adr+1) % p;
   }
   if (ha[adr].key==k)        //查找成功
      return adr;
   else                       //查找失败
      return -1;
}
```

	key	data	count
[0]	77		2
[1]	-1		0
[2]	54		1
[3]	16		1
[4]	43		1
[5]	-2		1
[6]	29		3
[7]	46		1
[8]	60		1
[9]	74		1
[10]	10		1
[11]	-1		0
[12]	90		1

m=13

n=11

p=13

查找16?

查找29?

查找42?

哈希表示例

- 前提
  - 将关键字序列{7,8,30,11,18,9,14}散列存储到散列表中
  - 散列表的存储空间是一个下标从0开始的一维数组
  - 散列函数为：H(key)=(key×3) mod 7
  - 处理冲突采用线性探测再散列法
  - 要求装填（载）因子为0.7。
- 问题

(1)请画出所构造的散列表。

(2)分别计算等概率情况下，查找成功和查找不成功的平均查找长度。

（1）n=7，α=0.7=n/m，则m=n/0.7=10。

计算各关键字存储地址的过程如下：

- H(7)=7×3 mod 7=0
- H(8)=8×3 mod 7=3
- H(30)=30×3 mod 7=6
- H(11)=11×3 mod 7=5
- H(18)=18×3 mod 7=5　冲突
- d1=(5+1) mod 10=6　仍冲突
- d2=(6+1) mod 10=7
- H(9)=9×3 mod 7=6　冲突
- d1=(6+1) mod 10=7　仍冲突
- d2=(7+1) mod 10=8
- H(14)=14×3 mod 7=0　冲突
- d1=(0+1) mod 10=1

下标	0	1	2	3	4	5	6	7	8	9
关键字	7	14	-1	8	-1	11	30	18	9	-1
探测次数	1	2	0	1	0	1	1	3	3	0

哈希表示例

- 前提
  - 将关键字序列{7,8,30,11,18,9,14}散列存储到散列表中
  - 散列表的存储空间是一个下标从0开始的一维数组
  - 散列函数为：H(key)=(key×3) mod 7
  - 处理冲突采用线性探测再散列法
  - 要求装填（载）因子为0.7。
- 问题

(1)请画出所构造的散列表。

(2)分别计算等概率情况下，查找成功和查找不成功的平均查找长度。

（2）在等概率情况下：

- $ASL_{成功}=(1+2+1+1+1+3+3)/7=12/7=1.71$
- $ASL_{不成功}=(3+2+1+2+1+5+4)/7=18/7=2.57$
- 由于任一关键字k，H(k)的值只能是0～6之间，不成功的情况，共有7种：H(k)为0需要比较3次，H(k)为1需要比较2次，H(k)为2需要比较1次，H(k)为3需要比较2次，H(k)为4需要比较1次，H(k)为5需要比较5次，H(k)为6需要比较4次。

下标	0	1	2	3	4	5	6	7	8	9
关键字	7	14	-1	8	-1	11	30	18	9	-1
探测次数	1	2	0	1	0	1	1	3	3	0