阅读：14,917　　作者：解学武

哈希查找算法及C语言实现

上一节介绍了有关哈希表及其构造过程的相关知识，本节将介绍如何利用哈希表实现查找操作。

在哈希表中进行查找的操作同哈希表的构建过程类似，其具体实现思路为：对于给定的关键字 K，将其带入哈希函数中，求得与该关键字对应的数据的哈希地址，如果该地址中没有数据，则证明该查找表中没有存储该数据，查找失败：如果哈希地址中有数据，就需要做进一步的证明（排除冲突的影响），找到该数据对应的关键字同 K 进行比对，如果相等，则查找成功；反之，如果不相等，说明在构造哈希表时发生了冲突，需要根据构造表时设定的处理冲突的方法找到下一个地址，同地址中的数据进行比对，直至遇到地址中数据为 NULL（说明查找失败），或者比对成功。

> 回顾：哈希表在构造过程中，处理冲突的方法有：开放定址法、再哈希法、链地址法、建立公共溢出区法。

假设哈希表在构造过程采用的开放定址法处理的冲突，则哈希表的查找过程用代码实现为：

```
01.  #include "stdio.h"
02.  #include "stdlib.h"
03.  #define HASHSIZE 7 //定义散列表长为数组的长度
04.  #define NULLKEY -1
05.  typedef struct{
06.      int *elem;//数据元素存储地址，动态分配数组
07.      int count; //当前数据元素个数
08.  }HashTable;
09.  //对哈希表进行初始化
10.  void Init(HashTable *hashTable){
11.      int i;
12.      hashTable->elem= (int *)malloc(HASHSIZE*sizeof(int));
13.      hashTable->count=HASHSIZE;
14.      for (i=0;i<HASHSIZE;i++){
15.          hashTable->elem[i]=NULLKEY;
16.      }
17.  }
18.  //哈希函数(除留余数法)
19.  int Hash(int data){
20.      return data%HASHSIZE;
21.  }
22.  //哈希表的插入函数，可用于构造哈希表
23.  void Insert(HashTable *hashTable,int data){
24.      int hashAddress=Hash(data); //求哈希地址
25.      //发生冲突
```

```
    while(hashTable->elem[hashAddress]!=NULLKEY){
        //利用开放定址法解决冲突
        hashAddress=(++hashAddress)%HASHSIZE;
    }
    hashTable->elem[hashAddress]=data;
}

//哈希表的查找算法
int Search(HashTable *hashTable,int data){
    int hashAddress=Hash(data); //求哈希地址
    while(hashTable->elem[hashAddress]!=data){//发生冲突
        //利用开放定址法解决冲突
        hashAddress=(++hashAddress)%HASHSIZE;
        //如果查找到的地址中数据为NULL，或者经过一圈的遍历回到原位置，则查找失败
        if (hashTable->elem[hashAddress]==NULLKEY||hashAddress==Hash(data)){
            return -1;
        }
    }
    return hashAddress;
}
int main(){
    int i,result;
    HashTable hashTable;
    int arr[HASHSIZE]={13,29,27,28,26,30,38};
    //初始化哈希表
    Init(&hashTable);
    //利用插入函数构造哈希表
    for (i=0;i<HASHSIZE;i++){
        Insert(&hashTable,arr[i]);
    }
    //调用查找算法
    result= Search(&hashTable,29);
    if (result==-1) printf("查找失败");
    else printf("29在哈希表中的位置是:%d",result+1);
    return  0;
}
```

运行结果为：

```
29在哈希表中的位置是:2
```

查找算法的效率分析

在构造哈希表的过程中，由于冲突的产生，使得哈希表的查找算法仍然会涉及到比较的过程，因此对于哈希表的查找效率仍需以平均查找长度来衡量。

在哈希表的查找过程中需和给定值进行比较的关键字的个数取决于以下 3 个因素：

- 哈希函数：哈希函数的“好坏”取决于影响出现冲突的频繁程度。但是一般情况下，哈希函数相比于后两种的影响，可以忽略不计。
- 处理冲突的方式：对于同一组关键字，设定相同的哈希函数，使用不同的处理冲突的方式得到的哈希表是不同的，表的平均查找长度也不同。
- 哈希表的装填因子：在一般情况下，当处理冲突的方式相同的情况下，其平均查找长度取决于哈希表的装满程度：装的越满，插入数据时越有可能发生冲突；反之则越小。

> 装填因子=哈希表中数据的个数/哈希表的长度，用字符 α 表示（是数学符号，而不是字符 a）。装填因子越小，表示哈希表中空闲的位置就越多。

经过计算，在假设查找表中的所有数据的查找概率相等的情况下，对于表长为 m，数据个数为 n 的哈希表：

- 其查找成功的平均查找长度约为：-1/α * ln(1-α)
- 其查找不成功的平均查找长度约为：1/(1-α)

通过公式可以看到，哈希表的查找效率只同装填因子有关，而同哈希表中的数据的个数无关，所以在选用哈希表做查找操作时，选择一个合适的装填因子是非常有必要的。