```c
/* ClosedHash.h */
#ifndef _CLOSED_HASH_H
#define _CLOSED_HASH_H
#include "Vector.h"

//返回值：0--相同，>0--dataAddr<keyAddr，<0--dataAddr>keyAddr
typedef int HashCmp(const void *keyAddr, const void *dataAddr);
typedef void HashFree(void *);
//将关键码转换成相应的桶地址
typedef int HashFunc(const void *);
typedef int HashCollide(int hashKey, int count);

typedef struct
{
    VECTOR vEntry;
    VECTOR vLazy;
    int capacity;
    int size;
    int keySize;
    int valSize;
    HashFunc *hashFn;
    HashCollide *collideFn;
    HashCmp *cmpFn;
    HashFree *freeFn;
}HASH;

//散列表初始化
void HashNew(HASH *h, int capacity, int keySize, int valSize, HashFunc *hashFn,
HashCollide *collideFn, HashCmp *cmpFn, HashFree *freeFn);
//获取散列表数据数量
int HashSize(HASH *h);
//获取散列表容量
int HashCapacity(HASH *h);
//散列表销毁
void HashDispose(HASH *h);
//散列表读取
void *HashGet(HASH *h, const void *e);
//散列表插入
int HashPut(HASH *h, const void *e, const void *val);
//散列表删除
int HashRemove(HASH *h, void *e);
//重散列
static void HashRehash(HASH *hN, HASH *hO);
#endif
```