```
#define KEYLENGTH 15                    /* 关键词字符串的最大长度 */
typedef char ElementType[KEYLENGTH+1]; /* 关键词类型用字符串 */
typedef int Index;                      /* 散列地址类型 */
/******** 以下是单链表的定义 ********/
typedef struct LNode *PtrToLNode;
struct LNode {
    ElementType Data;
    PtrToLNode Next;
};
typedef PtrToLNode Position;
typedef PtrToLNode List;
/******** 以上是单链表的定义 ********/

typedef struct TblNode *HashTable; /* 散列表类型 */
struct TblNode {   /* 散列表结点定义 */
    int TableSize; /* 表的最大长度 */
    List Heads;    /* 指向链表头结点的数组 */
};

HashTable CreateTable( int TableSize )
{
    HashTable H;
    int i;

    H = (HashTable)malloc(sizeof(struct TblNode));
    /* 保证散列表最大长度是素数，具体见代码5.3 */
    H->TableSize = NextPrime(TableSize);

    /* 以下分配链表头结点数组 */
    H->Heads = (List)malloc(H->TableSize*sizeof(struct LNode));
    /* 初始化表头结点 */
    for( i=0; i<H->TableSize; i++ ) {
         H->Heads[i].Data[0] = '\0';
         H->Heads[i].Next = NULL;
    }

    return H;
}

Position Find( HashTable H, ElementType Key )
{
    Position P;
    Index Pos;

    Pos = Hash( Key, H->TableSize ); /* 初始散列位置 */
    P = H->Heads[Pos].Next; /* 从该链表的第1个结点开始 */
    /* 当未到表尾，并且Key未找到时 */
    while( P && strcmp(P->Data, Key) )
        P = P->Next;

    return P; /* 此时P或者指向找到的结点，或者为NULL */
}

bool Insert( HashTable H, ElementType Key )
{
    Position P, NewCell;
    Index Pos;

    P = Find( H, Key );
    if ( !P ) { /* 关键词未找到，可以插入 */
        NewCell = (Position)malloc(sizeof(struct LNode));
        strcpy(NewCell->Data, Key);
        Pos = Hash( Key, H->TableSize ); /* 初始散列位置 */
        /* 将NewCell插入为H->Heads[Pos]链表的第1个结点 */
        NewCell->Next = H->Heads[Pos].Next;
        H->Heads[Pos].Next = NewCell;
        return true;
    }
    else { /* 关键词已存在 */
        printf("键值已存在");
        return false;
    }
}

void DestroyTable( HashTable H )
{
    int i;
    Position P, Tmp;

    /* 释放每个链表的结点 */
    for( i=0; i<H->TableSize; i++ ) {
```

```
        P = H->Heads[i].Next;
        while( P ) {
            Tmp = P->Next;
            free( P );
            P = Tmp;
        }
    }
    free( H->Heads ); /* 释放头结点数组 */
    free( H );        /* 释放散列表结点 */
}
```