```
#define MAXTABLESIZE 100000 /* 允许开辟的最大散列表长度 */
typedef int ElementType;    /* 关键词类型用整型 */
typedef int Index;          /* 散列地址类型 */
typedef Index Position;     /* 数据所在位置与散列地址是同一类型 */
/* 散列单元状态类型，分别对应：有合法元素、空单元、有已删除元素 */
typedef enum { Legitimate, Empty, Deleted } EntryType;

typedef struct HashEntry Cell; /* 散列表单元类型 */
struct HashEntry{
    ElementType Data; /* 存放元素 */
    EntryType Info;   /* 单元状态 */
};

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 */
    Cell *Cells;   /* 存放散列单元数据的数组 */
};

int NextPrime( int N )
{ /* 返回大于N且不超过MAXTABLESIZE的最小素数 */
    int i, p = (N%2)? N+2 : N+1; /*从大于N的下一个奇数开始 */

    while( p <= MAXTABLESIZE ) {
        for( i=(int)sqrt(p); i>2; i-- )
            if ( !(p%i) ) break; /* p不是素数 */
        if ( i==2 ) break; /* for正常结束，说明p是素数 */
        else  p += 2; /* 否则试探下一个奇数 */
    }
    return p;
}

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 */
    H->TableSize = NextPrime(TableSize);
    /* 声明单元数组 */
    H->Cells = (Cell *)malloc(H->TableSize*sizeof(Cell));
    /* 初始化单元状态为“空单元” */
    for( i=0; i<H->TableSize; i++ )
        H->Cells[i].Info = Empty;

    return H;
}
```