```
Position Find( HashTable H, ElementType Key )
{
    Position CurrentPos, NewPos;
    int CNum = 0; /* 记录冲突次数 */

    NewPos = CurrentPos = Hash( Key, H->TableSize ); /* 初始散列位置 */
    /* 当该位置的单元非空，并且不是要找的元素时，发生冲突 */
    while( H->Cells[NewPos].Info!=Empty && H->Cells[NewPos].Data!=Key ) {
                                        /* 字符串类型的关键词需要 strcmp 函数!! */
        /* 统计1次冲突，并判断奇偶次 */
        if( ++CNum%2 ){ /* 奇数次冲突 */
            NewPos = CurrentPos + (CNum+1)*(CNum+1)/4; /* 增量为+[(CNum+1)/2]^2 */
            if ( NewPos >= H->TableSize )
                NewPos = NewPos % H->TableSize; /* 调整为合法地址 */
        }
        else { /* 偶数次冲突 */
            NewPos = CurrentPos - CNum*CNum/4; /* 增量为-(CNum/2)^2 */
            while( NewPos < 0 )
                NewPos += H->TableSize; /* 调整为合法地址 */
        }
    }
    return NewPos; /* 此时NewPos或者是Key的位置，或者是一个空单元的位置（表示找不到）*/
}

bool Insert( HashTable H, ElementType Key )
{
    Position Pos = Find( H, Key ); /* 先检查Key是否已经存在 */

    if( H->Cells[Pos].Info != Legitimate ) { /* 如果这个单元没有被占，说明Key可以插入在此 */
        H->Cells[Pos].Info = Legitimate;
        H->Cells[Pos].Data = Key;
        /*字符串类型的关键词需要 strcpy 函数!! */
        return true;
    }
    else {
        printf("键值已存在");
        return false;
    }
}
```