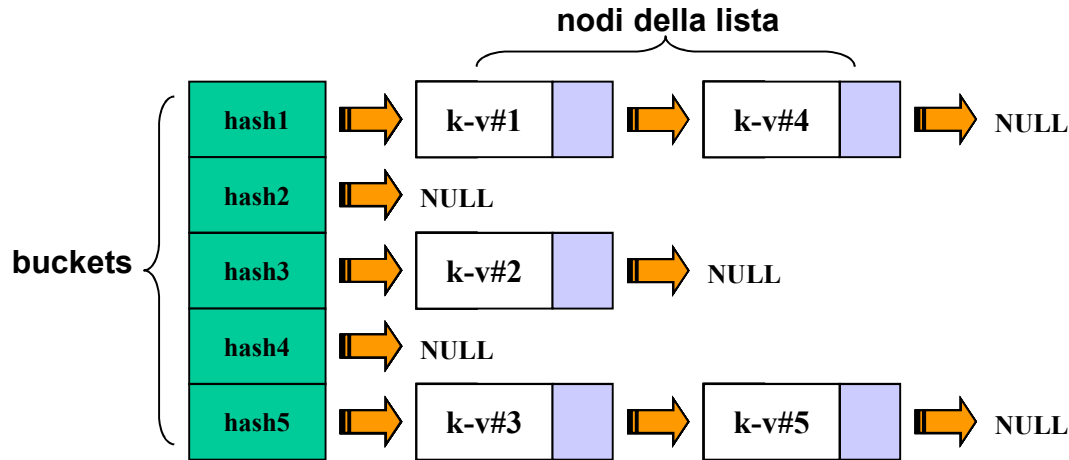


Tabelle hash

Le tabelle sono costituite da insiemi di elementi, ciascuno rappresentato da una **coppia chiave-valore**: la chiave serve ad identificare univocamente l'elemento, il valore è l'informazione utile. Rispetto alle liste, garantiscono maggiore **efficienza temporale** nelle operazioni di ricerca, inserimento, cancellazione (non di visita) al costo di **maggiori necessità in termini di memorizzazione**.

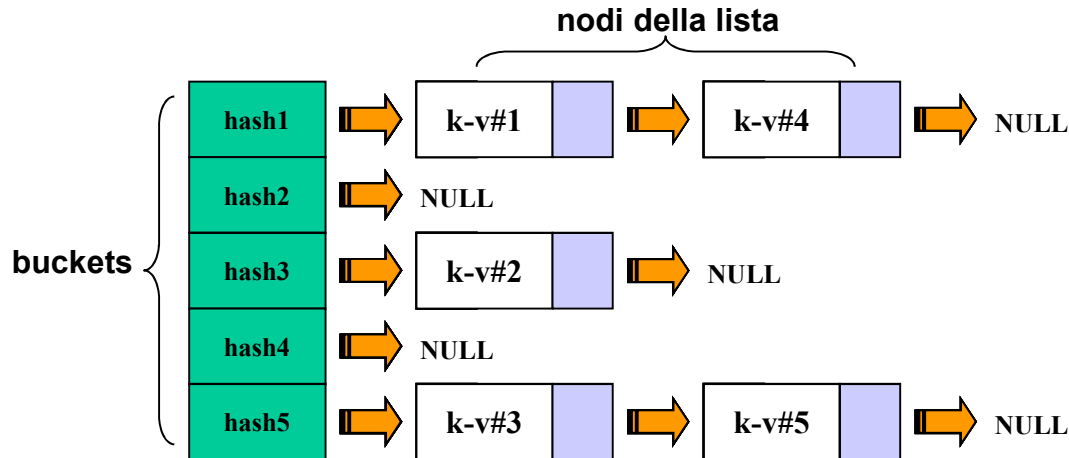
Impiegando una tecnica di accesso detta hashing limitiamo i requisiti di memoria richiesti. L'**hash** è una funzione che indicizza/riassume/codifica i valori di chiave: questa tecnica può generare delle **collisions** (analogo hash per valori chiave differenti). Gestendo le chiavi collidenti per mezzo di liste dinamiche esterne alla tabella (dette liste di collisione), ottengo una tabella hash ad **indirizzamento chiuso** (a differenza dell'**indirizzamento aperto**, dove le collisioni sono gestite occupando posizioni successive della tabella).

Tabelle hash



un esempio di tabella hash ad indirizzamento chiuso:
la tabella può essere rappresentata come un vettore di **buckets** (sono le celle in verde, ciascuna identificata da un diverso **codice hash** generato dall'algoritmo). Ogni bucket punta ad una **lista** (se allocata, altrimenti a null) la quale conterrà **uno o più nodi** (a seconda delle collisioni su quel codice hash). Ogni nodo conterrà una specifica **coppia chiave(k) valore(v)** (celle bianche) oltre al **puntatore** al nodo successivo (celle grigie) ... come già sapevamo a proposito delle liste dinamiche.

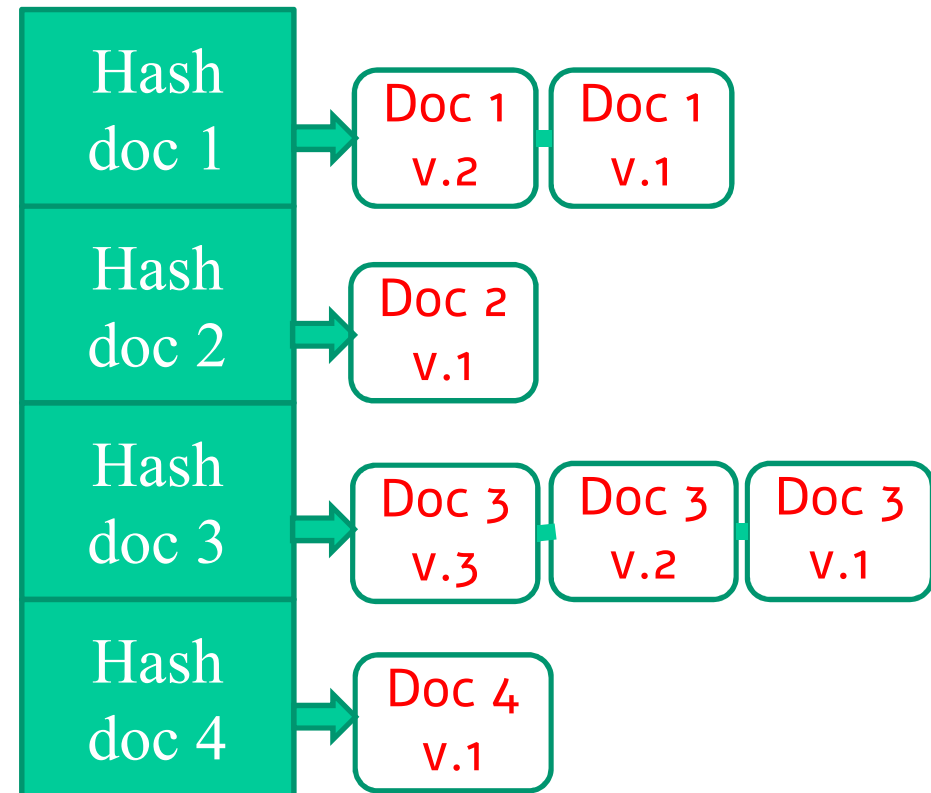
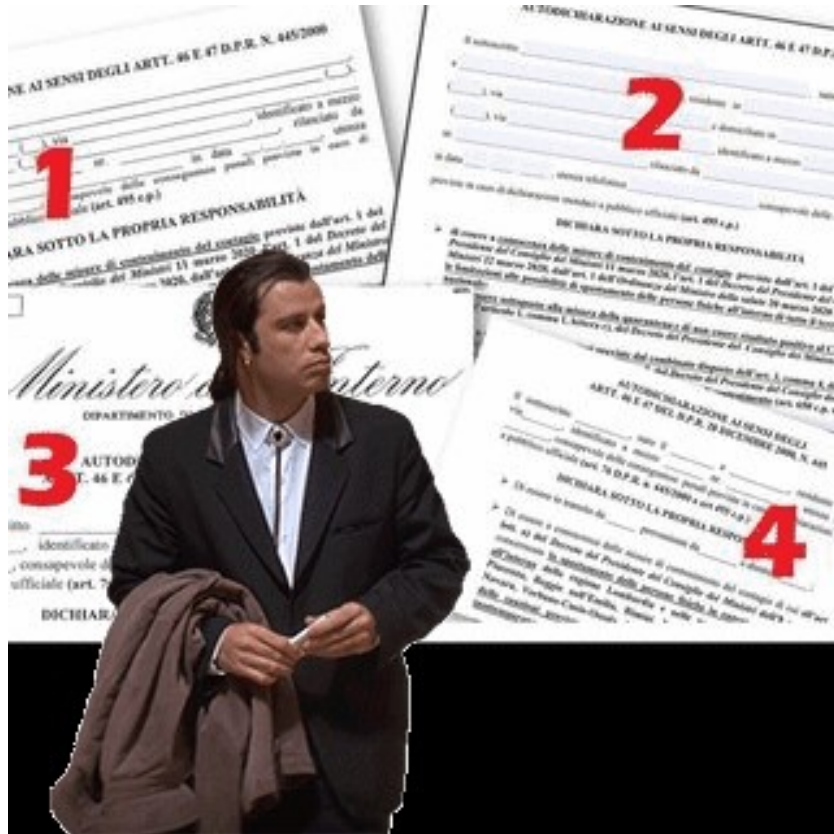
Tabelle hash



Sempre con riferimento a tabelle hash ad indirizzamento indiretto e chiuso – e alla proprietà di ordinamento:

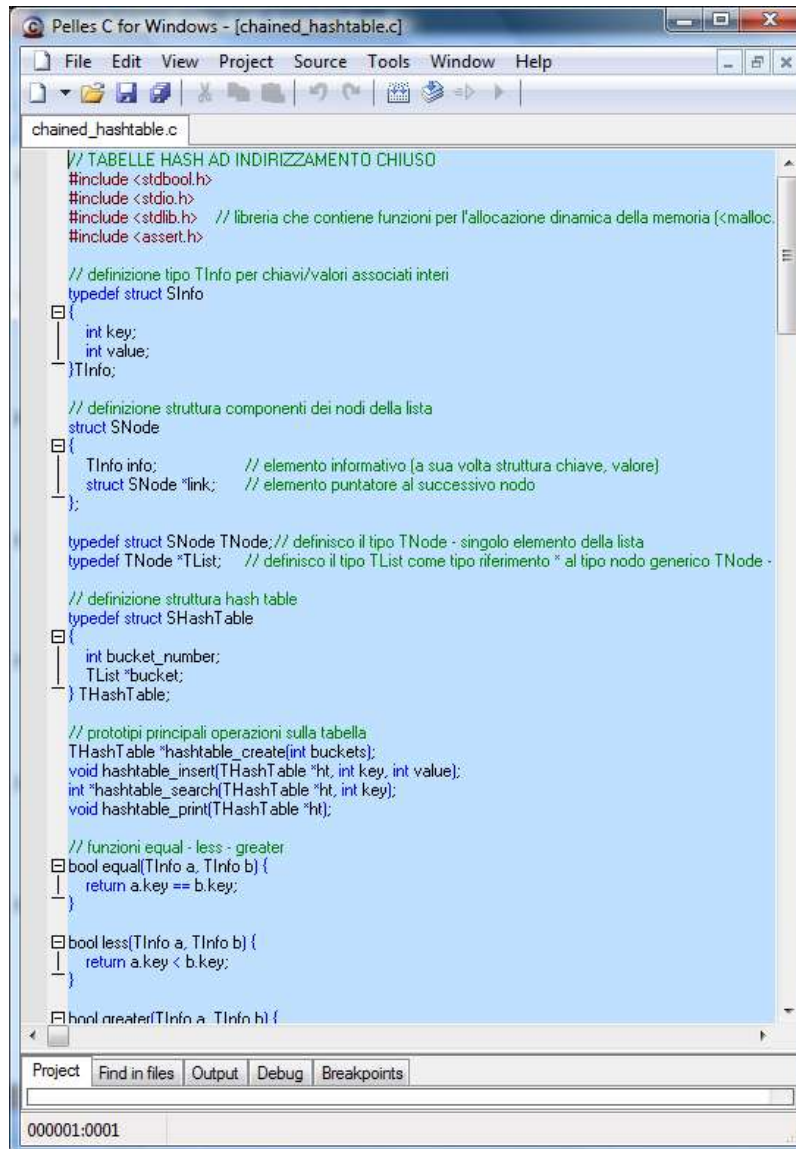
- qualora le liste collegate ai buckets non siano ordinate, l'operazione di **inserimento** avrebbe complessità $\Theta(1)$, considerando sia l'onere del calcolo dell'hash, che quello del successivo inserimento nel nodo in testa alla lista;
- qualora invece le liste collegate siano ordinate, l'operazione di inserimento si complicherebbe, in funzione del fattore di carico n/m (con m dimensione del vettore buckets ed n numero di elementi), ma verrebbe poi parzialmente semplificata l'operazione di **ricerca** (in quando non sarebbe più necessario navigare tutti i nodi della lista individuata a seguito del calcolo dell'hash).

Tabelle hash



Un esempio? Posso utilizzare una tabella hash ad indirizzamento chiuso per gestire il versioning della documentazione

Tabelle hash



```
// TABELLE HASH AD INDIRIZZAMENTO CHIUSO
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h> // libreria che contiene funzioni per l'allocazione dinamica della memoria (<malloc,
#include <assert.h>

// definizione tipo TInfo per chiavi/valori associati interi
typedef struct SInfo
{
    int key;
    int value;
} TInfo;

// definizione struttura componenti dei nodi della lista
struct SNode
{
    TInfo info; // elemento informativo (a sua volta struttura chiave, valore)
    struct SNode *link; // elemento puntatore al successivo nodo
};

typedef struct SNode TNode; // definisco il tipo TNode - singolo elemento della lista
typedef TNode *TList; // definisco il tipo TList come tipo riferimento "al tipo nodo generico TNode"

// definizione struttura hash table
typedef struct SHashTable
{
    int bucket_number;
    TList *bucket;
} THashTable;

// prototipi principali operazioni sulla tabella
THashTable *hashtable_create(int buckets);
void hashtable_insert(THashTable *ht, int key, int value);
int *hashtable_search(THashTable *ht, int key);
void hashtable_print(THashTable *ht);

// funzioni equal - less - greater
bool equal(TInfo a, TInfo b) {
    return a.key == b.key;
}

bool less(TInfo a, TInfo b) {
    return a.key < b.key;
}

bool greater(TInfo a, TInfo b) {
    return a.key > b.key;
}
```

CHAINED_HASHTABLE.c
creazione di una tabella
hash ad indirizzamento
chiuso (con liste dinamiche
concatenate e per coppie di
chiavi-valori interi) – ricerca
e stampa di un singolo
elemento / di tutti gli
elementi della tabella.
Gestione delle chiavi
duplicate.