In generale una tabella altro non è che una serie ordinata di coppie di valori di qualche tipo, se il tipo di uno dei due valori è enumerabile allora in C tale tabella si può rappresentare con un vettore (scusa se ti dico cose banali ma mi servono per introdurre i concetti più spinosi), dove il tipo enumerabile è rappresentato dall'indice mentre l'altro valore (di qualunque tipo) è rappresentato dall'elemento corrispondente a quell'indice.

Nel caso della tabella hash la coppia è rappresentata da un indice (di tipo intero) e da una chiave (il cui tipo è irrilevante e poi capirai il perchè).

In C costruire una tabella e popolarla significa, ad esempio:

- definire un vettore di elementi di tipo opportuno
- realizzare una funzione che riceve in input l'elemento da inserire e la tabella stessa, che, attraverso un'istruzione di assegnamento, lo inserisce in una cella libera del vettore.
- Successivamente, nel main (o meglio nella procedura chiamante), ci si preoccuperà di realizzare opportunamente la chiamata di tale funzione passandole via via tutti gli elementi da inserire.

Ma la tabella hash ha una particolarità in più e cioè quella di poter calcolare l'indice da assegnare alla chiave in relazione al valore della chiave stessa, inoltre per sua natura (non biunivoca) la funzione generatrice potrebbe a diverse chiavi associare uno stesso indice (le famose collisioni). Questo comporta che la suddetta funzione si complica un pochino, infatti devo prevedere

1. un meccanismo che data una chiave mi tira fuori un indice (per fare questo puoi creare una funzione a parte)
2. devo poter "fare qualcosa" se all'indice calcolato c'è già un'altro elemento (posto occupato).

A questo punto faccio una piccola digressione: come faccio a capire che il posto è occupato? Bisogna prevedere una chiave fittizia tale che non potrà mai coincidere con una possibile chiave gestita dalla tabella e poi, in fase di inizializzazione del vettore, assegnare la chiave fittizia a tutti gli elementi. In tal modo, per dire che una cella del vettore è libera è sufficiente un matching rispetto alla chiave fittizia nota. Potresti obiettare il fatto che se la chiave è una stringa non è necessario utilizzare una chiave fittizia e che si potrebbe fare riferimento al valore NULL del puntatore, questo è verissimo ma la completa gestione della tabella hash prevede anche altri controlli su altrettante chiavi fittizie, non è il caso tuo (a meno di voler fare gli sboroni) ma tanto vale abituarsi all'idea e soprattutto alla tecnica.

Uno dei meccanismi più semplici per calcolare l'indice da una chiave è una funzione del tipo

```
int hash(char *chiave)
{
    int indice=0, base=128;
    for(;*chiave!='\0';chiave++)
      indice=(base*indice+*chiave)%DIM_TABELLA;
    return indice;
}
```

l'idea di questa tecnica di conversione sta nel considerare una stringa come un “numero” n-ario dove n=128, ed il calcolo è una sorta di algorimo di Horner modificato. Tale modifica introduce il

calcolo del modulo rispetto alle dimensioni della tabella per evitare di dover incappare numeri troppo grandi per essere rappresentati dalla macchina. Ad ogni modo puoi trovare metodi alternativi di conversione, questo è solo un esempio.

Per quanto riguarda invece la strategia da adottare in caso di collisioni, di solito, viene implementata la cosiddetta “scansione lineare”, che effettua il controllo delle collisioni e restituisce l'indice in cui può essere memorizzata la chiave:

una funzione di scansione lineare può essere questa:

```
int scan(tipochiave k)
{
     int  indice, //punta alla chiave corrente
          primo,  //se indice torna ad assumere il suo valore
                  //il ciclo di scansione si considera terminato

     int terminato, //nel while varrà 1 quando
                    //primo==indice

//inizializzazione
     primo=indice=hash(k);
     terminato=prenotato=0;

//ciclo di scansione
     while(!terminato && diz[indice].key!=LIBERO)
     //ci sono due vie per uscire dal ciclo:
     //1) quando si ritorna alla chiave di partenza (terminato)
     //2) quando si trova la prima chiave libera
          {
               //calcolo del prox indice
               indice=(indice+1)%DIM_TABELLA;

               //se l'indice corrente coincide col primo
               //allora considero il ciclo terminato
               if(primo==indice)
                    terminato=1;
          } //fine del ciclo while

     //se non si trova un posto libero per la chiave
     //allora si restituisce un valore di indice non ammissibile
    //che sarà interpretato all'esterno della funzione
     if(terminato)
          return DIM_TABELLA;
    //altrimenti è implicito che sia stata trovata una locazione
    //libera e quindi se ne restituisce l'indice
     else
          return indice;
}
```

a questo punto hai tutto ciò che ti serve per creare la famosa funzione di popolamento della tabella hash.

Alcuni suggerimenti:

1. conviene dichiarare la tabella hash come vettore di strutture (array di record per intenderci), dove la struttura è proprio la struct che rappresenta il token;

2. la funzione di popolamento della tabella conviene invocarla nell'if che controlla se un token è una parola chiave (in caso contrario invochi la funzione ponendo così l'identificatore nella tabella dei simboli);

3. ti conviene anche (perchè è utile e fa bella impressione) salvare la tabella in un file a parte.

...e ora dopo tanta teoria ecco un programmino di esempio:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define clrscr() system ("cls")
#define pause() system ("pause")

//costante di grandezza della tabella
#define DIM_TABELLA 20


int hash(char *chiave)
    {
       int indice=0, base=128;
       for(;*chiave!='\0';chiave++)
       indice=(base*indice+*chiave)%DIM_TABELLA;
       return indice;
    }


int scan(char* k, char* tabella[], char* sentinella)
    {
       int indice, //punta alla chiave corrente
           primo,  //se indice torna ad assumere il suo valore
                   //il ciclo di scansione si considera terminato
           terminato; //nel while varrà true quando primo==indice

       //inizializzazione
       primo=indice=hash(k);
       terminato=0;

       //ciclo di scansione
       while(!terminato && strcmp(tabella[indice],sentinella)!=0)
       //ci sono due vie per uscire dal ciclo:
       //1) quando si ritorna alla chiave di partenza (terminato)
       //2) quando si trova la prima chiave libera
            {
              //calcolo del prox indice
              indice=(indice+1)%DIM_TABELLA;

              //se l'indice corrente coincide col primo
              //allora considero il ciclo terminato
              if(primo==indice)
                  terminato=1;
            } //fine del ciclo while

       //esito della scansione

        if(terminato)
       //se non si trova un posto libero per la chiave
       //allora si restituisce un valore di indice non ammissibile
       //che sarà interpretato all'esterno della funzione
          return DIM_TABELLA;
       else
       //altrimenti è implicito che sia stata trovata una locazione
```

```
        //libera e quindi se ne restituisce l'indice
           return indice;
   }
                                                                    //continua alla pag successiva
void stampa_tab_hash(char* tabella[])
   {
      for(int i=0;i<DIM_TABELLA;i++)//inizializzazione della tabella hash
      printf("\n tab_hash[%d]= %s ",i,tabella[i]);
      printf("\n");
   }

int main()
   {
      char* libero="0:FREE:0"; //la scelta della stringa di inizializzazione
                               //è arbitraria, l'importante è che non coincida
                               //con una delle possibili chiavi.

      char* tab_hash[DIM_TABELLA];//dichiarazione della tabella hash

      for(int i=0;i<DIM_TABELLA;i++)//inizializzazione della tabella hash
         tab_hash[i]=libero;

      //in questo punto occorre definire il modo con cui
      //dare in pasto al programma una serie di chiavi,
      //l'input può essere un problema, perchè la traccia
      //può chiederti di leggere da un file (è molto probabile)
      //oppure può chiederti un input da tastiera
      //tuttavia per testare il programma userò un vettore di stringhe
      //che è più semplice (ma purtroppo meno utile per le applicazioni hash)
      //anche per non appesantirti la comprensione del codice
      //introducendo una struct

      char* elenco_id[]=//tabella dei token di 8 elementi
         {
            "pippo",
            "pluto",
            "paperino",
            "topolino",
            "provolino",
            "geppo",
            "pbolo",
            "ciccio"
         };

      int cella; //contiene l'indice in cui sarà posizionata la chiave corrente

      for(int i=0;i<8;i++)
      //sottopongo a scan gli 8 elementi di elenco_id
      //poichè la dimensione della tab hash è 7 l'ultimo elemento (ciccio)
      //non sarà inserito e verrà fuori il messaggio di tab satura
         {
            cella=scan(elenco_id[i],tab_hash,libero);
            if(cella<DIM_TABELLA)
                tab_hash[cella]=elenco_id[i];
            else
                printf("\n La tabella e' satura!");
         }

      stampa_tab_hash(tab_hash);
      system("PAUSE");
   }
```