TRANSFORMACIONES DE LLAVES (HASHING)

H (función de transformación) **:K**(llaves) ==> **A** (direcciones de memoria)

H, es una función que transforma una llave en un índice de memoria. Si se tiene un llave *k*, el primer paso en una operación de búsqueda es calcular su índice asociado (*h =H(k)*); es segundo paso será verificar si el elemento con la llave *k* es realmente identificado por *h* en el arreglo
(*[H(k).llave = k]*)

5.2 ELECCIÓN DE UNA FUNCIÓN DE TRANSFORMACIÓN DE LLAVES.

Un requisito básico de una buena función de transformación es que distribuya las llaves lo más uniformemente posible sobre la gama de valores índice.

5.3 MANEJO DE LAS COLISIONES

Se llama **colisión** al caso en que una llave que no sea la deseada se encuentra en la localización identificada. De otra forma: si resulta que un elemento de una tabla correspondiente a determinada llave no es el elemento deseado, entonces existe colisión, esto es, dos elementos tienen llaves que los mapean en el mismo índice.

Para manejar las colisiones, se realiza un segundo sondeo. Métodos:

- con un índice secundario:
  - **encadenamiento directo**, consiste en ligar todos los elementos con el índice primario idéntico H(k) en una lista ligada.
  - **área de desbordamiento**, consiste en almacenar los elementos colisionados en una lista secundaria; la desventaja es que reserva espacio para un apuntador a su lista de colisión
- dirección abierta (vacía):
  - **exploración lineal**: Ensayar la siguiente posición (tabla circular), hasta encontrar el elemento o posición vacía. Pero tiene la desventaja que la distribución no uniforme pues las entradas tienden a agruparse alrededor de las llaves primarias
  - **exploración cuadrática**: la siguiente posición es calculada por una función cuadrática. La distribución es más uniforme, el cálculo puede hacerse por recurencia para no tener que elevar al cuadrado. Desventaja es que los índices no recorren toda la tabla y podría ocurrir que quedando huecos no pudiésemos encontrarlos, pero si N es primo por lo menos visitaremos la mitad de la tabla.

Programa:

```
MODULE XRef;
   FROM InOut IMPORT Read,Done,EOL,Write,WriteCard,WriteString,WriteLn;
   FROM Storage IMPORT ALLOCATE ;
   CONST P= 7;(* primo, tama¤o tabla *)  BufLeng = 1000;  WordLeng =16;   free = 0;
TYPE WordInx = [0..P-1];  ItemPtr=POINTER TO Item;
      Word = RECORD   key: CARDINAL;  first,last:ItemPtr;  END;
      Item = RECORD     lno: CARDINAL;       next:ItemPtr;  END;
   VAR k0,k1,line:CARDINAL;  ch:CHAR; T:ARRAY [0..P-1] OF Word;(* tabla de transformaci¢n de llaves *)
      buffer:ARRAY [0..BufLeng-1] OF CHAR;
PROCEDURE PrintWord(k:CARDINAL);  VAR lim:CARDINAL;
   BEGIN    lim:= k + WordLeng;
   WHILE buffer[k] > 0C DO Write(buffer[k]);  k:= k+1; END;
   WHILE k < lim DO Write(" "); k:= k+1;  END;
```

```
                    END PrintWord;
ToniWeb(http://www.jazzcyber.com/jazz5/toniweb/); E-mail: toniweb@jazzcyber.com
                    PROCEDURE PrintTable;
                       VAR i,k,m:CARDINAL; item:ItemPtr;
                    BEGIN
                     FOR k:= 0 TO P-1 DO
                       WriteCard(k,2);WriteString("       ");
                       IF T[k].key # free THEN PrintWord(T[k].key);  item:=T[k].first;  m:= 0;
                          REPEAT
                             IF m=8 THEN WriteLn; m:= 0; FOR i:= 1 TO WordLeng DO Write(" "); END;
                             END;
                             m:= m+1; WriteCard(item^.lno,6); item:=item^.next;
                          UNTIL item = NIL;
                          WriteLn;
                       ELSE    PrintWord(T[k].key);Write(" ");WriteString("LIBRE");WriteLn;
                       END;
                    END;
                    END PrintTable;
                    PROCEDURE Diff(i,j:CARDINAL):INTEGER;
                    BEGIN
                    LOOP
                       IF buffer[i] # buffer[j] THEN  RETURN INTEGER(ORD(buffer[i])) - INTEGER(ORD(buffer[j]))
                       ELSIF buffer[i]= 0C THEN RETURN 0
                       END;
                       i:= i+1; j:= j+1;
                    END
                    END Diff;
                       PROCEDURE search;
                    VAR i,h,d:CARDINAL; found:BOOLEAN;  ch:CHAR;  x:ItemPtr;
                    BEGIN
                    i:= k0; h:= 0; ch:= buffer[i];
                    WHILE ch > 0C DO h:= (256*h+ ORD(ch)) MOD P; i:= i+1;  ch:= buffer[i];
                    END;
                    NEW(x);
                    x^.lno:= line;  x^.next:= NIL;  d:= 1;  found:= FALSE;
                    REPEAT
                       IF Diff(T[h].key,k0)= 0 THEN (* coincidencia *)
                          found:= TRUE; T[h].last^.next:= x;   T[h].last:= x;
                       ELSIF T[h].key = free THEN (* nueva entrada *)
                          WITH T[h] DO key:= k0; first:= x;  last:= x;  END;
                          found:= TRUE;  k0:=k1;
                       ELSE                    (* colision *)
                          h:= h+ d; d:= d+2;
                          IF h >= P THEN  h:= h-P; END;
                          IF d = P THEN WriteLn; WriteString("Desbordamiento de tabla"); WriteLn; PrintTable; HALT END;
                       END;
                    UNTIL found;
                    END search;
                 PROCEDURE GetWord;
                    BEGIN
                    k1:= k0;
                       REPEAT Write(ch);buffer[k1]:= ch; k1:= k1+1; Read(ch)
                       UNTIL (ch < "0") OR (ch >"9") & (CAP(ch)<"A") OR (CAP(ch)>"Z");
                       buffer[k1]:= 0C; k1:= k1 +1;  search;
                    END GetWord;
                 BEGIN
                 k0:= 1; line:= 0;
                 FOR k1:= 0 TO P-1 DO T[k1].key:= free END;
                 WriteCard(0,6);Write(" ");Read(ch);
                 WHILE ch # 33C DO
                    CASE ch OF
                       0C..35C:    Read(ch)|
                       36C..37C:   WriteLn;Read(ch);line:=line+1;
                                   WriteCard(line,6);Write(" ")|
                       " ".."@":   Write(ch);Read(ch)|
                       "A".."Z":   GetWord|
                       "["..""`":   Write(ch);Read(ch)|
                       "a".."z":   GetWord|
```

```
        "{".."~":   Write(ch);Read(ch)|
    END;
END;
WriteLn;WriteLn;WriteLn;
PrintTable;
END XRef.
```


5.4 ANALISIS DE LA TRANSFORMACIÓN DE LLAVES

En la inserción y recuperación por transformación de llaves tiene un rendimiento pésimo en el peor caso, porque todas los mapeos estén ocupados. El promedio de búsqueda es pequeño y depende exclusivamente del factor de carga del array de almacenamiento.

Caso concreto:

Una llave debe ser insertada en una tabla de tamaño n que ya contiene k elementos. Suponiendo que todas la llaves tienen las misma probabilidad y que la función de transformación H las distribuye uniformemente sobre el intervalo de los índices de la tabla. La probabilidad de encontrar una localización vacía la primera vez será de (n-k)/n. Además es la probabilidad de que se necesite una sola comparación. La probabilidad de que se requiera exactamente una segunda exploración es igual a la probabilidad de una colisión en el primer intento, multiplicada por la probabilidad de hallar una localización libre la siguiente vez. En general sería:

P1= (n - k)/n

P2= (k/n) × (n - k)/(n-1)

P3 =(k/n) × (n - k)/(n-1) × (n - k) × (n-2)

… … …

Pi = (k/n) * (n -k)/(n-1)×(k-2)/(n-2)× …× (n - k)/ (n-i+1)

Se necesitan en promedio menos de 3 exploraciones para encontrar una llave o una localización vacía en un array lleno al 90%, factor de carga 0.9, mediante exploración cuadrática. 11

Número esperado de exploraciones según el factor de carga:

Factor de carga	Exploración cuadrática	Exploración lineal
0.1	1.05	1.06
0.25	1.15	1.17
0.5	1.39	1.50
0.75	1.85	2.50
0.9	2.56	5.50
0.95	3.15	10.50
0.99	4.66	?????

Inconvenientes

- El tamaño del array es fijo y no puede ajustarse a la exigencia del momento.
- La eliminación de un elemento insertado es muy difícil.

Ventajas:

- Se conoce a priori el volumen de datos a tratar.
- El volumen de datos experimenta pocas variaciones.

ToniWeb(http://www.jazzcyber.com/jazz5/toniweb/); E-mail: toniweb@jazzcyber.com