Le hachage :

# Recherche

**Rech** (c, var trouv : bool, var j : entier)

/\* recherche la donnée c dans la table T (globale) et retourne le booléen trouv à vrai si la donnée existe et faux sinon. Retourne aussi l’indice ou la donnée est sensée se trouver\*/

**Début**

J 🡨h(c) ; /\*l’adresse primaire de c\*/

Trouv🡨faux

TQ (>T[j].vide ET >Trouv)

1. **SI** (T[j].val=c)
   1. trouv🡨vrai
2. **SINON**
   1. j🡨j-1
   2. **SI** (j<0) j🡨N-1 FSI /\*gestion circulaire de la table\*/
3. **FSI**

**FTQ**

**FIN**

# Insertion

**Inser** (c :typeqlq)

**Début**

1. Rech (c,trouv,j)
2. **SI** (>trouv)
   1. /\*Dans ce cas j est l’indice d’une case vide\*/
   2. /\*Si ce n’est pas la dernière case vide qui reste, on peut l’utiliser pour y insérer c\*/
   3. **Si** (NbIns<N-1)
      1. NbIns++
      2. T[j].val🡨c
      3. T[j].vide🡨faux
   4. **Sinon**
      1. Écrire (‘débordement : l’insertion ne peut se faire !’)
   5. **FSI**
3. **FSI**

**Fin.**

# Suppression

**Supprimer** (c :typeqlq)

**Début**

1. Rech (c,trouv,j)
2. **Si** (trouv)
   1. /\*on va tester toutes les case i au-dessus de j jusqu’à trouver une case vide\*/
   2. i🡨j-1
   3. **SI** (i<0) i🡨N-1 FSI
   4. **TQ** (>T[i].vide)
      1. /\*Pour vérifier si la case i pause problème quand on videra la case j, il suffit de voir où se trouve son adresse primaire\*/
      2. r🡨h(T[i].val) /\*soit l’@ primaire de la donnée de la case \*/
      3. **Si** (r<i<j) OU (i<j<r) OU (j<=r<u) /\*si l’une de ces conditions est vraie, \*/
         1. T[j]🡨T[i] /\*alors on déplace la donnée de i vers j\*/
         2. j🡨i /\*et on positionne j vers la nouvelle case\*/
      4. **FSI**
      5. i🡨i-1 /\*Dans tous les cas, on passe à la prochaine case à tester\*/
      6. **Si** (i<0) i🡨N-1 **FSI**
   5. **FTQ**
   6. /\*A la fin de la boucle, on aura testé toutes les cases au-dessus de j qui pourraient être gênées par l’apparition d’un vide au niveau de la casse j. Les données qui posaient problème ont été déplacées. La case j peut maintenant être vider sans problème\*/
   7. T[j].vide🡨vrai
   8. NbIns—
3. **FSI**

**Fin.**

Struction= Enregistrement

Val : entier

Eff : booléen

Vide : booléen

Fin

T : tableau [1..Max] de structure

**Insert\_DH** (V : entier, var NB\_Ins : entier)

**Var** j,i : entier, Trouv ; booleen

**Début**

1. **Si** (NB\_Ins<=n-1)
   1. Rech\_DH (V,trouv,i,j)
2. **Si** (Non trouv) alors
   1. T[j].val🡨V
   2. T[j].eff🡨Faux
   3. T[j].Vide🡨Faux
   4. Nb\_Ins++
3. **FSI**

**Fin.**

**Proc Rech\_DH** (V:entier, var trouv: booléen, var i: entier)

**Var** pas : entier

**Début**

1. i=h(v)
2. Pas=h(v)
3. J🡨-1
4. Trouv🡨faux
5. **TQ** (Non trouv) ET (Non T[i].Vide) **alors**
   1. **SI** (T[i].val=v) ET (Non T[i].eff) **alors**
      1. trouv🡨vrai
   2. **Sinon**
      1. **Si** (T[i].eff) ET (J=-1) **alors** j🡨i
      2. i🡨i-pas
      3. **Si** (i<0) **alors** i🡨n-1
   3. **FSI**
6. **FTQ**
7. **Si** (non trouv) **ET** (j>0) **alors** i🡨j

**Fin.**

# Recherche hi

**Procédure** Rech\_hi (v : entier, var trouv : booléen, i, j : entier)

**Début**

1. i🡨h(v)+1
2. j🡨-1
3. trouv🡨faux
4. **Si** (T[i].vide) **alors**
   1. **TQ** (non trouv) ET (i<0) **faire**
      1. **Si** (T[i].val=v) **alors**
         1. trouv🡨vrai
      2. **Sinon**
         1. j🡨i
         2. i🡨T[i].lien
      3. **FSI**
   2. **FTQ**
5. **FSI**

**Fin.**

# Recherche chaîne interne :

**Procédure** recherche\_chain\_interne (k :typeqlq, T :tab, var trouv : booléen, var i : entier, var prec : booléen)

**Début**

1. Trouv🡨faux
2. i🡨h(v)+1
3. prec🡨0
4. **Si** (K=T[i].clé) **ET** (T[i].vide=faux) **alors** trouv🡨vrai
5. **Sinon**
   1. **TQ** (T[i].lien<>0) **ET** (T[i]<>K) **faire**
      1. prec🡨i
      2. i🡨T[i].lien
   2. **FTQ**
   3. **Si** (T[i].clé=k) **alors** trouv🡨vrai
6. **FSI**

**Fin.**

# Recherche CI :

**Procédure** recherche\_CI (k :typeqlq, T :tab, var trouv : booléen, var i : entier, var j : entier)

**Var** prec : entier

**Début**

1. Trouv🡨faux
2. i🡨h(k)
3. j🡨-1
4. **Si** (T[i].vide) **alors** trouv🡨faux
5. **Si** (k=T[i].clé) **ET** (>T[i].vide) **alors** trouv🡨vrai
6. **Sinon** // chercher dans la liste des synonymes
   1. **TQ** (T[i].lien<>0) **ET** (T[i].clé<>k) **faire**
      1. j🡨i
      2. i🡨T[i].lien
   2. **FTQ**
   3. **Si** (T(i].clé=k) **alors** trouv🡨vrai
7. **FSI**

**Fin.**

# Insertion CI :

**Procédure** Insertion\_CI (K :Typeqlq, T : tab)

**Début**

1. Recherche\_CI (K,T,Trouv,i,j)
2. **Si** (Trouv=vrai) alors
   1. Ecrire (‘Valeur déjà existante’)
3. **Sinon** //Chercher un position vide à partir de R-1
4. //Après avoir vérifié si i<>h(K)
   1. **Si** (h(K)=i) **alors**
      1. T[i].clé🡨k
      2. T[i].lien🡨-1
      3. T[i].vide🡨faux
   2. **Sinon**
      1. R🡨R-1
      2. **TQ** (T[R].vide=faux) **ET** (R<>0) **Faire**
         1. R🡨R-1
      3. **FTQ**
      4. **Si** (R=0) **alors** écrire (‘Impossible d’insérer débordement’)
      5. **Sinon** //Insérer à partir de R
         1. T[R].clé🡨K
         2. T[R].lien🡨-1
         3. T[i].lien🡨R
         4. T[R].vide🡨faux
         5. //R pointe la dernière case occupée
      6. **FSI**
   3. **FSI**
5. **FSI**

**Fin.**

# Suppression :

**Procédure** suppression (c : etnier, T : tab)

**Var** trouv : booléen, i, j : entier, k, k’ : entier

**Début**

1. Recherche (c,trouv,i,j)
2. **Si** (trouv) **alors**
   1. K🡨T[i].lien
   2. **TQ** (K>=0) **faire**
      1. **Si** (h(T[k].val)=i) **alors**
         1. T[i].val=T[k].val
         2. i🡨K
         3. J🡨K’
      2. **FSI**
      3. K’🡨K
      4. K🡨T[K].lien
   3. **FTQ**
   4. **Si** (J=-1) **alors** j🡨-K
   5. **TQ** (Abs(T[j].lien)=<>i) **faire**
      1. J🡨Abs(T[j].lien)
   6. **FTQ**
   7. **Si** (T[j].lien <0) **alors**
      1. T[j].lien🡨-T[j].lien
   8. **Sinon**
      1. T[j].lien🡨T[i].lien
   9. **FSI**
   10. T[i].vide🡨vrai
3. **FSI**

**Fin.**

# Recherche :

**Procédure** Recherche (T :tab, D: entier, var i : entier, var trouv :booléen, var j :ptr maillon)

**Début**

1. I🡨h(D)
2. trouv🡨faux
3. **Si** (T[i].cle=d)
   1. trouv🡨vrai
4. **Sinon**
   1. **Si** (T[i].lien<>Nil)
      1. j🡨T[i].lien
      2. **TQ** (j<>NIL) et ( !trouv)
         1. **Si** (valeur(j)=d)
            1. trouv🡨vrai
         2. **Sinon**
            1. j🡨suivant(j)
         3. **FSI**
      3. **FTQ**
   2. **FSI**
5. **FSI**

**Fin.**

# Insertion :

**Procédure** Insertion (V : entier, T :Tab)

**Var** i: entier, trouv :booléen, j :ptr maillion

**Début**

1. i🡨h(v)
2. recherche (T,V,I,trouv,j)
3. **Si** (trouv) **alors** écrire (‘La valeur existe déjà’)
4. **Sinon**
   1. **Si** (T[i].vide) **alors**
      1. T[i].cle🡨v
      2. T[i].vide🡨faux
   2. **Sinon**
      1. **Si** (T[i].lien=Nil) **alors**
         1. Allouer(p)
         2. aff\_val(p,v)
         3. aff-adr(p,Nil)
         4. T[i].lien🡨p
      2. **Sinon**
         1. Allouer(p)
         2. Aff\_val(p,v)
         3. aff\_adr(p,T[i].lien)
         4. T[i].lien🡨p
      3. **FSI**
   3. **FSI**
5. **FSI**

**Fin.**

# Suppression :

**Procédure** Suppression (T :tab, d: entier)

**Début**

1. i🡨h(d)
2. Recherche (T,v,i,trouv,j)
3. **Si** (>Trouv) **alors**
   1. Ecrire (‘Suppression impossible’)
4. **Sinon**
   1. **Si** (T[i].cle=v) et (T[i].lien=Nil) **alors**
      1. T(i].vide🡨vrai
   2. **FSI**
   3. **SI** (T[i].cle=v) et (T[i].lien<>Nil) **alors**
      1. p🡨T[i].lien
      2. T[i].cle🡨valeur(T[i].lien)
      3. T[i].lien🡨suivant(T[i].lien)
      4. Libérer (p)
   4. **FSI**
   5. **Si** (T[i].cle<>v) //la valeur existe surement dans la liste des synonymes
      1. Q🡨T[i].lien
      2. **TQ** (suivant (Q)<>j)
         1. Q🡨suivant(Q)
      3. **FTQ**
      4. Aff\_adr (Q,suivant(j))
      5. Liberer(j)
   6. **FSI**
5. **FSI**

**Fin.**

The end…