CHAINAGE INTERNE

La suppression :

Cela consiste à rendre la case concernée vide et à rétablir le chainage de la liste, mais les valeurs suivantes dans la liste peuvent avoir comme adresse primaire l'indice de la case supprimée, dans ce cas elle seront perdues. Une petite gymnastique permet de les sauvegarder, il suffit de les retrouver et de les mettre à l'abri. Si vous voulez savoir comment lisez la suite.

La méthode :

Tout d'abord on cherche l'élément à supprimer dans la table.
Si il est trouvé alors sa position et celle de son précédent nous sont retournées par la procédure de recherche.
Au cas ou l'élément recherché a été trouvé et n'a pas de précédent alors une valeur -1 est retournée.

Cas particulier :

L'élément à supprimer est l'unique élément de la liste, dans ce cas quoi de plus simple, il devient une case vide.

Cas général : l'élément à supprimer fait partie d'une liste.

Consulter tous les suivants de la case à supprimer dans la liste. Si l'un d'eux a le numéro de la case à supprimer comme adresse primaire alors il est déplacé vers cette case et on réitère l'opération à partir de la case déplacée.

On s'arrête quand la fin de liste est atteinte.

Un autre Cas particulier :

Il peut arriver que l'élément recherché se trouve dans son adresse primaire, c'est-à-dire qu'on le trouve du premier coup dans la recherche, donc la recherche ne retourne pas de précédent. Si on fait des déplacements, un précédent verra le jour mais au cas ou on fait zéro deplacement le précédent reste inconnu, dans ce cas il faut le chercher.

La fin du cas général: rendre la case vide et mettre à jour son précédent. Il doit pointer le suivant de la case à supprimer.

Procedure Supprimer(c : entier)

```
Var trouv : bool ; i,j : ent ; K,K' : ent;
Debut
  Rech(c,trouv,i,j) ;
  Si trouv alors
    Si  T(i).lien#-i   alors            // si T(i).lien = - i c'est le cas d'une liste à un élément
        K=T(i).lien ; K'=i
        TQ  (K>=0) faire
           Si (h(T(K).val)=i) alors
             T(i).val=T(K).val ;
             i=K ;
             j = K' ;
           Fsi ;
           K'=K;
           K=T(K).lien
        FTQ ;

        Si (j=-1) alors
           J=Abs(T(i).lien) ;
           TQ (Abs(T(j).lien)<>i) faire  j=Abs(T(j).lien
        FTQ

        Si  T(j).lien<0 alors T(j).lien= - T(i).lien
           sinon T(j).lien=T(i).lien
        Fsi ;

    Fsi ;

    T(i).vide= vrai

  Fsi
FIN.
```

Faire les décalages **possibles**.
Verifions s'il y a au moins une donnée dont l'adr primaire=i.
Si c'est le cas on déplace la case i vers la case j et on change i et j vers la nouvelle position.

C se trouve dans son adresse primaire i et on a fait zéro deplacements alors on lui cherche un précédent

On met à jour le précédent et on supp la case i.

Si le précédent est la fin de liste

Enfin !

Chainage Interne

Une version légèrement améliorée de la suppresion

Procedure Supprimer(c : entier) ;

```
Var trouv : bool ; i,j : ent ;   K,K' : ent ;
Debut
  Rech(c,trouv,i,j) ;
  Si trouv alors
      K=T(i).lien ;
      TQ  (K>=0) faire
         Si (h(T(K).val)=i) alors
            T(i).val=T(K).val ;
            i =K ;
            j= K' ;
         Fsi ;
         K'=K;    K=T(K).lien
      FTQ ;
```

Faire les décalages **possibles**.
Verifions s'il y a au moins une donnée dont l'adr primaire=i.
Si c'est le cas on déplace la case i vers la case j et on change i et j vers la nouvelle position.

```
      Si (j=-1) alors
         J= - K ;
         TQ (Abs(T(j).lien)<>i) faire  j=Abs(T(j).lien
      FTQ
```

C se trouve dans son adresse primaire i et on a fait zéro deplacements alors on lui cherche un précédent

```
      Si  T(j).lien<0 alors T(j).lien= - T(i).lien
         sinon T(j).lien=T(i).lien
      Fsi ;
```

On met à jour le précédent et on supp la case i.

T(i).vide= vrai

Si le précédent est la fin de liste

```
  Fsi
FIN.
```

Enfin !

Chainage Interne