TD – Hachage

1) Essai linéaire

- Module de recherche
Soit T une table (globale) de N éléments indicés de 0 à N-1
Chaque élément est formée de 2 champs : val (de type qlq) et vide de type booleen

```
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 j où la donnée est sensée se trouver */
debut
  j := h(c);      /* l'adresse primaire de c */
  trouv := FAUX;

  TQ  (Non T[j].vide  ET Non Trouv)
        SI (T[j].val = c)
              Trouv := VRAI
        SINON
              j := j – 1;
              SI ( j < 0 )   j:=N-1 FSI;        /* gestion circulaire de la table */
        FSI
  FTQ
fin
```

- Module d'insertion

Soit NbIns un entier (global) initialisé à 0
à chaque insertion d'une donnée, NbIns est incrémenté
à chaque suppression, il est decrémenté

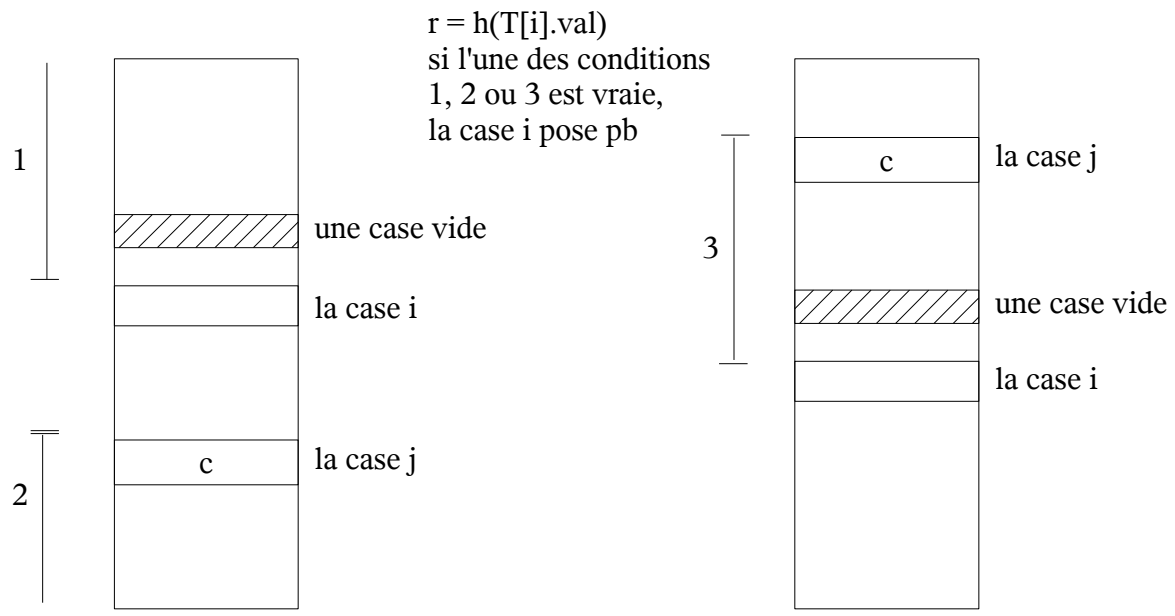
```
Inserer( c:typeqlq )
debut
  Rech(c,trouv,j);
  SI (Non trouv)
        /* dans ce cas j est l'indice d'une case vide */
        /* si ce n'est pas la dernière case vide qui reste, on peut l'utiliser pour y inserer c */
        SI (NbIns < N-1)
              NbIns++;
              T[j].val := c;
              T[j].vide := FAUX
        SINON
              ecrire(« Débordement: l'insertion ne peut se faire! » )
        FSI
  FSI
fin
```

- Module de suppression

pour supprimer une donnée c, on la recherche pour récupérer son adresse j (l'indice dans T).

avant de rendre vide T[j], il faut tester toutes les données qui se trouvent au dessus de j (jusqu'à trouver une case vide) pour vérifier si elles ne seront pas perdues si jamais on rend la case j vide.
s'il y a une telle donnée se trouvant l'adresse i, on la déplace dans la case j et on essaye de vider la case i (en procédant de la même manière que précédemment).
s'il n'y a aucune donnée pouvant être génée par l'effacement de la case j, cette dernière sera vidée

```
supprimer ( c:typeqlq )
debut
  Rech(c,trouv,j);
  SI (trouv)
        /* on va tester toutes les cases i au dessus de j jusqu'à trouver une case vide */
        i := j-1;
        SI (i<0)  i := N-1  FSI;
        TQ  ( Non T[i].vide )
              /* pour vérifier si la case i pose problème qd on videra la case j, il suffit de voir où se
trouve son adresse primaire */
              r := h (T[i].val);        /* soit r l'adr primaire de la donnée de la case i */
              SI (r<i<j) OU (i<j<=r) OU (j<=r<i)  /* si l'une de ces conditions est vrai, */
                    T[j] := T[i];                 /* alors on deplace la donnée de i vers j, */
                    j := i;                       /* et on positionne j vers la nouvelle case */
              FSI;
              i := i-1;                 /* dans tous les cas, on passe à la prochaine case à tester */
              SI (i<0)  i := N-1  FSI;
        FTQ;
        /* à la fin de la boucle, on aura tester toutes les cases au dessus de j qui pourraient être
genées par l'apparition d'un vide au niveau de la case j. Les données qui posées pb ont été
deplacées. La case j peut maintenant etre vider sans pb */
        T[j].vide := VRAI;
        NbIns--;
  FSI
fin
```



1: r < i < j
2: i < j <= r

3: j <= r < i