# Le double hachage

## Module de recherche :

```
Procédure Rech(c :typeqq ; var trouv : booléen, i :entier) ;
Var j,p : entier;
Debut
   Trouv := faux;   j := -1;    // j contiendra en cas de non trouv la première
                                // case effacée si rencontrée
   i :=h(c) ; p :=h2(c);            // P : le pas de décrémentation
         Tant que (non t[i].vide et non trouv)   faire
            Si (t[i].val=c)  et  (non t[i].efface) alors trouv := vrai
               Sinon si (j<0)  et  (t(i).efface) alors j :=i ;
                        i:=i-p ;   si (i<0) alors i:=i+N ;
            Fsi;
         Ftq;
          Si (non trouv)  et  (j>=0) alors i :=j ;
Fin.
```

## Module de suppression :

```
Procédure Supp (c: typeqq) ;
Var i : entier ; trouv : booléen ;
Debut
   Rech(c,trouv,i);
   Si (trouv) alors
               t[i].efface :=vrai ; nbIns := nbIns – 1 ;
   Fsi ;
Fin.
```

## Module d'insertion :

```
Procédure Ins (c : typeqq) ;

Var i :entier ; trouv : booléen ;
Debut
   Rech(c,trouv,i) ;
   Si (non trouv) et (nbIns<n-1) alors
            t[i].val := c ; t[i].efface :=faux ;
            t[i].vide :=faux ;
            nbIns++
  Fsi
Fin.
```

## Module d'insertion amélioré :

```
Procédure Ins(c : typeqq) ;
Var i :entier ; trouv : booléen ;
Debut
   Rech(c,trouv,i) ;
   Si (non trouv) alors
         Si  (t[i].vide) et (nbIns<n-1)  alors
            t[i].val := c ;
            t[i].efface :=faux ;
            t[i].vide :=faux ;
            nbIns++
         sinon
            si (t[i].efface)   alors
                     t[i].val := c ;
                     t[i].efface :=faux ;
                     t[i].vide :=faux ;
         Fsi
   Fsi
Fin.
```