TD 12
L'adresse dispersé

TAD
Semestre 2

Ce TD à été codé avec les pieds par nos chers professeurs, P comme Taille bien entendu, le rang k comme... jenesaispas
Les noms de variables ont donc été modifier en conséquences ;-)

1 Hachage direct

1.1

```
1  procedure chercherEmplacementClé(entree table <Table[T1,T2]>, entree
       emplacementSupposé <Entier>,
2                  entree clé <T1>, sortie trouvé <Booleen>,
3                  sortie emplacementClé <Entier>)
4  glossaire
5    i <Entier>; -- indice de parcours de table
6  debut
7    si table[emplacementSupposé].clé = clé alors
8      trouve <- VRAI;
9      emplacementClé <- emplacementSupposé;
10   sinon
11     -- recherche la clé dans la table table à partir du rang
           emplacementSupposé + 1
12     i <- (emplacementSupposé + 1) mod TAILLE_TABLEAU
13     trouve <- FAUX;
14     tantque i /= emplacementSupposé et non trouvé faire
15       si table[i].clé = clé alors
16         trouvé <- VRAI;
17         rang <- i;
18       sinon
19         i <- (i+1) mod TAILLE_TABLEAU;
20       fin si;
21     fin tantque;
22   fin si;
23 fin
```

1.2

```
procedure chercherPremierEmplacementLibre(entree table <Table[T1,T2]>,
                    entree debutRecherche <Entier>,
                    sortie videExiste <Booleen>,
                    sortie emplacementLibre<Entier>)
glossaire
  i <Entier>; -- indice de parcours de la table
debut
  si tab[debutRecherche].libre alors
    videExiste <- emplacementLibre;
    emplacementLibre <- debutRecherche;
  sinon
    -- rechercher le premier emplace libre à partir
    -- de debutRecherche + 1
    i <- (debutRecherche + 1) mod TAILLE_TABLEAU;
    trouve <- FAUX;

    tantque i /= debutRecherche et non videExiste faire
      si tab[i].libre alors
        videExiste <- VRAI;
        emplacementLibre <- i;
      sinon
        i <- (i + 1) mod TAILLE_TABLEAU;
      fin si;
    fin tantque;
  fin si;
fin
```

1.3

```
procedure chercher (entree tab <Table[T1, T2]>, entree clé <T1>, sortie
    trouvé <Booleen>,
          sortie element <T2>)
glossaire
  rang <Entier>; --resultat de la recherche
debut
  chercherEmplacementClé(tab, clé, hach(clé), trouvé, rang);
  si trouvé alors
    element <- tab[rang].element;
  fin si;
fin

procedure inserer(maj table <Table[T1, T2]>, entree clé <T1>, entree élément
    <T2>)
  declenche cléPrésente, tablePleine
glossaire
  h <Entier>;
  trouvé <Booleen>;
  rang <Entier>;
debut
  chercherEmplacementClé(tab, clé, hach(clé), trouvé, rang);
  si trouvé alors
    declenche (cléPrésente);
  fin si;
  chercherPremierEmplacementLibre(tab, clé, hach(clé), trouvé, rang);
  si non trouvé alors
```

```
25      declencher(taillePleine);
26    fin si;
27    -- insérer clé et élément dans tab à l'emplacement rang
28    tab[rang].clé <- clé;
29    tab[rang].élément <- élément;
30    tab[rang].libre <- FAUX;
31  fin
32
33  procedure supprimer(maj table <Table[T1, T2]>, entree clé <T1>)
34    declenche cléNonPrésente
35  glossaire
36    trouvé <Booleen>, rang <Entier>;
37  debut
38    chercherEmplacementClé(tab, clé, hash(clé), trouvé, rang);
39    si non trouvé alors
40      declencher (cléNonPrésente);
41    fin si;
42
43    tab[rang].libre <- VRAI;
44  fin
```