Algorithmes et structures de données : TD 10

Tables de hachage - Fonctions de hachage

Rappel :
`SetLength(tableau, n)` est de complexité O(n)
`SetLength(tableau, 1)` est de complexité O(1)
`New(element)` est de complexité O(1) quand `element` est d'un type de taille fixe

Exercice 10.1 *Fonction de hachage*

Considérer la fonction de hachage $h(c) = cmod13$ et une table de hachage avec $m = 13$ adresses.

1. Insérer les clés 26,37,24,30, et 11 dans la table de hachage ci-dessus en utilisant la résolution des collisions par adressage ouvert et sondage linéaire avec la fonction $h_i(c) = (h(c)+i)modm$.
2. Rajouter maintenant les clés 1,2,3,4,5,6,7,8,9,10. Quel problème rencontrez-vous ? Quelle solution proposez vous ?
3. On considére une nouvelle table de hachage, et cette fois-ci on résoud les collision par chaînage externe. Insérer les clés 26,37,24,30, 11 ainsi que 1,2,3,4,5,6,7,8,9,10 dans cette table de hachage, et ebauchez la mémoire d'une manière compacte.

Exercice 10.2 *Table de hachage*

```
program td_10;

{$APPTYPE CONSOLE}

uses
  SysUtils;

const m = 13;

type element = record
  cle       : string;
  date      : string;
end;

var newelement : element;
var cle        : string;
var HT : array[0..m-1] of element;

function hash(cle : string) : integer;
var j : integer;
var i : integer;
begin
  if length(cle) = 0 then
    result := 0
  else
  begin
    j := ord(cle[1]) mod m;
    for i := 2 to length(cle) do
      j := (j * 256 + ord(cle[i])) mod m;
    result := j;
  end;
end;

function findHT(cle : string) : integer;
var i : integer;
begin
  i := hash(cle);
  result := -1;
  while (result=-1) do
  begin
    if (length(HT[i].cle)=0) or (HT[i].cle = cle) then
      result := i;
    i := (i + 1) mod m;
```

```
  end;
end;

function exists(cle : string) : boolean;
var i : integer;
begin
     i := findHT(cle);
     if NOT(HT[i].cle='') then
       result:= true
     else                      // key is not in table
       result:= false;
end;

function lookup(cle : string) : element;
var i : integer;
begin
     i := findHT(cle);
     result:= HT[i];
end;

 procedure put(elem : element);
 var i : integer;
 begin
     i := findHT(elem.cle);
     if i>=0 then
       HT[i] := elem
     else
     begin

         { COMMENTAIRE
         if the table is almost full
             rebuild the table larger (note 1)
             i := findHT(key)
       HT[i] := elem;
       FIN DU COMMENTAIRE}

       WriteLn('Enlarge table ');
     end;
 end;

procedure showtable;
var i : integer;
begin

  { à faire }

end;
```

```
begin
  newelement.cle := 'SAM';
  newelement.date := '20.03.84';
  put(newelement);
  newelement.cle := 'EMMA';
  newelement.date := '13.02.86';
  put(newelement);
  newelement.cle := 'LEO';
  newelement.date := '21.02.85';
  put(newelement);
  newelement.cle := 'AXEL';
  newelement.date := '28.03.82';
  put(newelement);

  if (exists('AXEL')) then
  begin
        newelement := lookup('AXEL');
        WriteLn('Birthdate',newelement.date);
  end
  else
        WriteLn('Cette entrée n existe pas');
  showtable;

  ReadLn;
end.
```

1. Ecrire la fonction `showtable` qui affiche la table de hachage avec la clé et la date d'anniversaire.
2. Faites tourner le programme entier dans un tableau. Utiliser une nouvelle colonne pour chaque nouvelle variable locale.