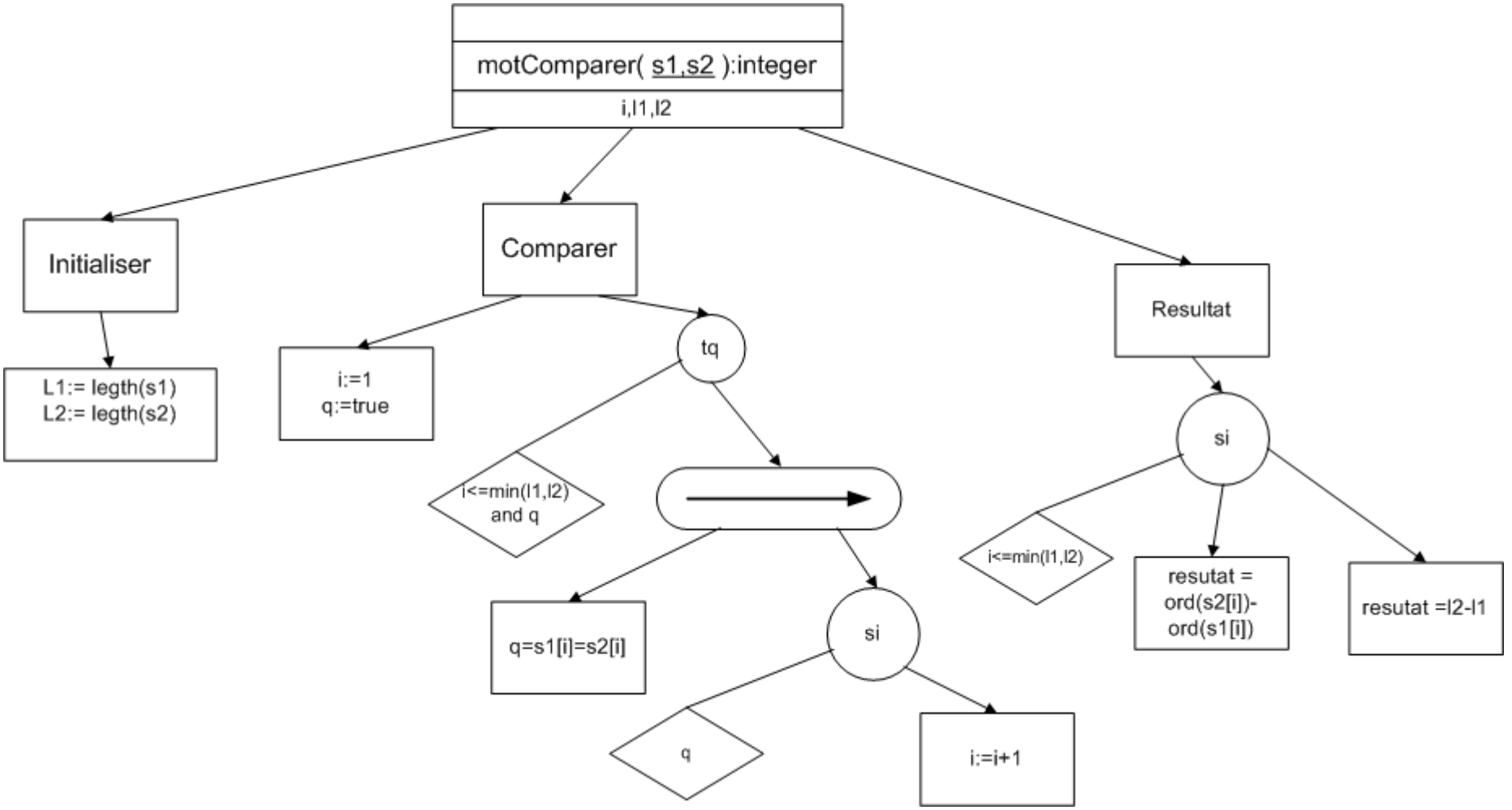
Les chaînes de caractères et les fichiers

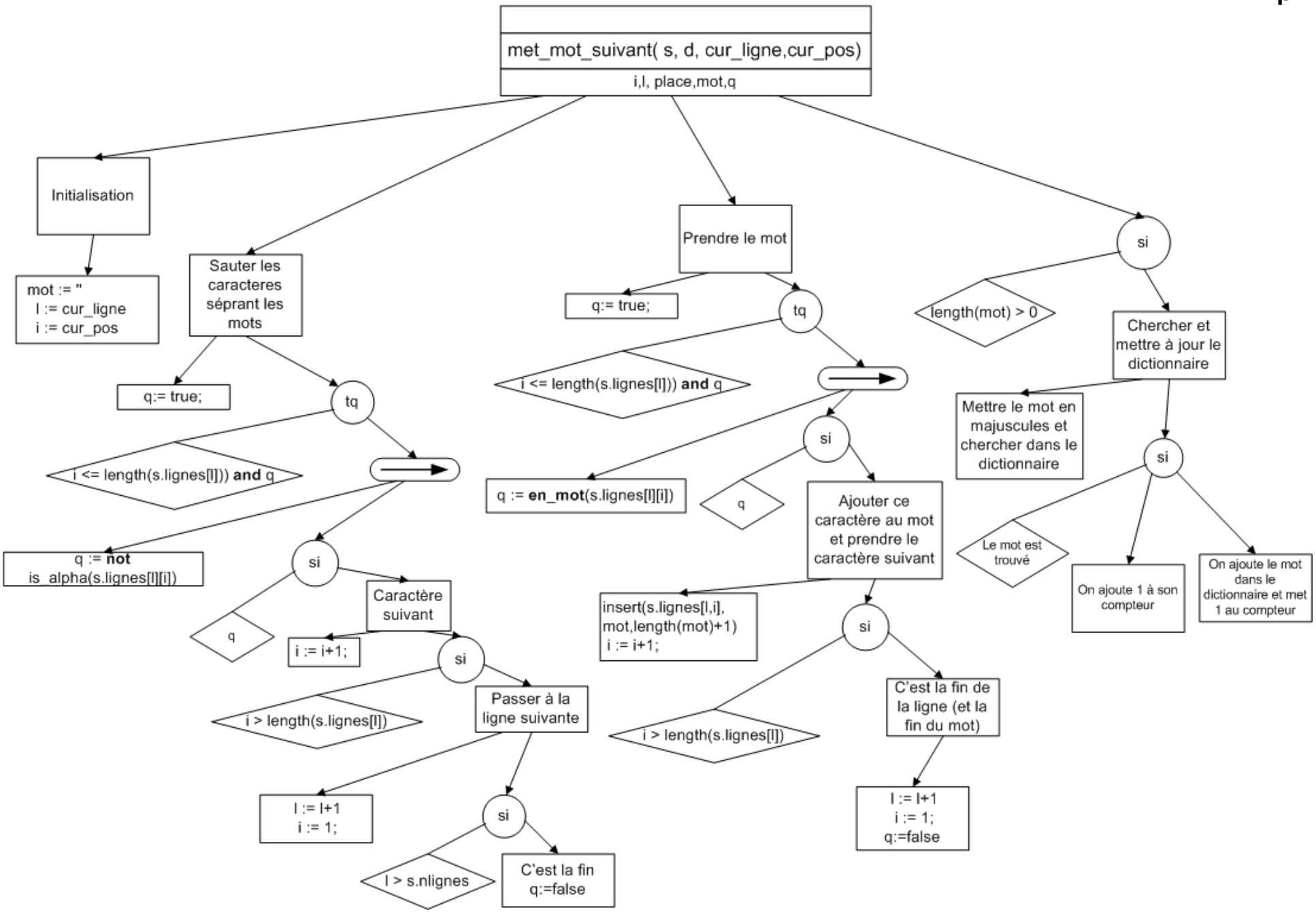
Ex. 1 Faire un dictionnaire des mots d'un texte et :

1. ***Sauvegarder le dans un fichier :***
2. ***Trier les mot en ordre alphabétique et en ordre décroissant de nombre d'occurrences.***

```
program ExDictionnaire;
uses
   Math;
const
   MAXLIGNES = 100;
   MAXMOTS = 200;
 type
   Dictelem = record
        mot : string[40];
        compte : integer
   end;
   Dictionnaire = record
                  nmots : integer;
                  elems : array [1..MAXMOTS] of Dictelem;
                end;
   texte =   record
             nlignes : integer;
           lignes : array [1..MAXLIGNES] of string[100]
           end;
   DictFile = file of Dictelem;
   compElem = function (e1,e2:Dictelem):integer;
 var
   Dict : Dictionnaire;
   source : texte;
   filesource : text;
   fDict : DictFile;
   procedure InitDict (var d : Dictionnaire);
   begin
     d.nmots := 0;
   end;
   procedure readSource(var f : text; var s : texte);
   var t: string[100];
   begin
     reset(f);
     with s do
     begin
       nlignes := 0;
       repeat
         readln(f,t);
         if length(t) > 0 then
           begin
             nlignes := nlignes+1;
             lignes[nlignes] := t;
           end;
        until (nlignes >= MAXLIGNES) or eof(f);
      end;
      close(f);
   end; {readSource}
   Procedure afficheSource(var s : texte);
   var i : integer;
   begin
     with s do
     for i := 1 to nlignes do writeln(lignes[i]);
   end; {writesource}
 function motComparer(var s1,s2 :string):integer;
 var i,l1,l2 : integer;
     q : boolean;
 begin
   l1 := length(s1);
   l2 := length(s2);
   i :=1; q := true;
   while  (i<=min(l1,l2)) and q do
   begin
      q := s1[i] = s2[i];
      if q then i := i+1;
   end;
   if i<=min(l1,l2) then
      motComparer := ord(s2[i]) - ord(s1[i])
   else motComparer := l2-l1;
 end; {motComparer}
 function cherche_dictionnaire(var m: string;
                               var d: dictionnaire):integer;
 var i : integer;
     q : boolean;
 begin
   i := 1;  q := true;
   cherche_dictionnaire := 0;
   while (i<= d.nmots) and q do
   begin
     q:= motComparer(m, d.elems[i].mot) <>0;
     if q then
       i := i+1
     else
       cherche_dictionnaire := i;
   end;
  end; { cherche_dictionnaire}
 function is_alpha(c:char):boolean;
 begin
   is_alpha := (c >='A') and (c <='Z')
                  or (c>='a') and (c <= 'z');
 end; { is_alpha }
 function en_mot(c:char):boolean;
 begin
   en_mot := is_alpha(c) or (c='-');
 end; { en_mot }
 procedure met_mot_suivant(var s:texte;
                 var d:dictionnaire;
                  var cur_ligne,cur_pos:integer);
 var i,l, place: integer;
     mot : string[40];
     q:boolean;
 begin
   mot := '';
   l := cur_ligne;
   i := cur_pos;
   q:= true;
 { passer les caractères entre les mots}
     while (i <= length(s.lignes[l])) and q do
      begin
        q := not is_alpha(s.lignes[l][i]);
        if q then
          begin
            i := i+1;
            if i > length(s.lignes[l]) then
```

```
          begin
            l := l+1; i := 1;
            if l > s.nlignes then q := false;
          end;
        end;
      end;{while}
      q := true;
{prendre le mot}
     while (i <= length(s.lignes[l])) and q do
     begin
      q := en_mot(s.lignes[l,i]);
      if q then
      begin
        insert(s.lignes[l,i],mot,length(mot)+1);
        i := i+1;
        if i > length(s.lignes[l]) then
        begin
          l := l+1;
          i := 1;
          q := false;
        end;
       end;
      end; {while}
      cur_pos := i;
      cur_ligne := l;
    if length(mot) > 0 then
    begin
      mot := UpperCase(mot);
      place := cherche_dictionnaire(mot,d);
     if   place> 0 then
       d.elems[place].compte :=
                  d.elems[place].compte+1
     else begin  {nouveau mot}
       place := d.nmots+1;
       if place <= MAXMOTS then
       begin
         d.elems[place].mot := mot;
         d.elems[place].compte := 1;
         d.nmots := place;
       end;
     end;
   end;
 end;{ met_mot_suivant}
 procedure afficher_dictionnaire (var d:dictionnaire);
 var i : integer;
 begin
  with d do
  for i := 1 to nmots do
    with elems[i] do writeln(mot : 50, compte:3);
  writeln('------------------------------');
 end; { afficher_dictionnaire}
procedure remplir_dictionnaire(var s:texte;
                               var d:dictionnaire);
 var l, i :integer;
 begin
  l:=1; i :=1;
  while l<=s.nlignes do met_mot_suivant(s,d,l,i);
 end; { remplir_dictionnaire}
 procedure write_dictionnaire(var f:DictFile;
                              var d:dictionnaire);
 var i:integer;
 begin
 rewrite(f);
 with d do
  for i:=1 to nmots do write(f,elems[i]);
 close(f);
 end; { write_dictionnare}
 procedure read_dictionnaire(var f:DictFile;
                             var d:dictionnaire);
 var i:integer;
 begin
 reset(f);
 i := 0;
 with d do
  while not eof(f) do
    begin
      i := i+1; read(f,elems[i]);
    end;
   close (f);
  d.nmots := i;
 end; { read_dictionnare}
 function comparer_alpha(e1,e2 : Dictelem):integer;
 begin
   comparer_alpha := motComparer(e1.mot,e2.mot);
 end; {  compare_alpha }
function comparer_freq(e1,e2 : Dictelem):integer;
 begin
   comparer_freq := e1.compte-e2.compte;
 end; {  compare_freq }
 Procedure triDict(var d:Dictionnaire;
                   compare:CompElem);
 var i:integer;
     t: dictElem;
     ex :boolean;
 begin
   repeat
    ex := false;
    with d do
    for i:=1 to nmots-1 do
      if compare(elems[i], elems[i+1])<0 then
      begin
        t := elems[i];  elems[i] := elems[i+1];
        elems[i+1] := t;  ex := true;
      end;
    until not ex;
 end;
begin {principal}
InitDict(Dict);
chdir('G:\Bogi\France');
assign(filesource, 'sourcetext.txt');
readSource(Filesource,source);
afficheSource(source);
remplir_dictionnaire(source,dict);
afficher_dictionnaire(Dict);
assign(fDict,'myDict.dat');
write_Dictionnaire(fDict,Dict);
triDict( Dict, comparer_alpha);
 afficher_dictionnaire(Dict);
 InitDict(Dict);
read_Dictionnaire(fDict,Dict);
triDict( Dict, comparer_freq);
 afficher_dictionnaire(Dict);
end.
```


motComparer(s1,s2):integer
i,l1,l2
Initialiser
L1:= legth(s1)
L2:= legth(s2)
Comparer
i:=1
q:=true
tq
i<=min(l1,l2)
and q
q=s1[i]=s2[i]
si
q
i:=i+1
Resultat
si
i<=min(l1,l2)
resutat =
ord(s2[i])-
ord(s1[i])
resutat =l2-l1


met_mot_suivant(s, d, cur_ligne,cur_pos)
i,l, place,mot,q
Initialisation
mot := ''
l := cur_ligne
i := cur_pos
Sauter les caracteres séprant les mots
q:= true;
tq
i <= length(s.lignes[l])) and q
q := not is_alpha(s.lignes[l][i])
si
q
Caractère suivant
i := i+1;
si
i > length(s.lignes[l])
Passer à la ligne suivante
l := l+1
i := 1;
si
l > s.nlignes
C'est la fin
q:=false
Prendre le mot
q:= true;
tq
i <= length(s.lignes[l])) and q
q := en_mot(s.lignes[l][i])
si
q
Ajouter ce caractère au mot et prendre le caractère suivant
insert(s.lignes[l,i], mot,length(mot)+1)
i := i+1;
si
i > length(s.lignes[l])
C'est la fin de la ligne (et la fin du mot)
l := l+1
i := 1;
q:=false
si
length(mot) > 0
Chercher et mettre à jour le dictionnaire
Mettre le mot en majuscules et chercher dans le dictionnaire
si
Le mot est trouvé
On ajoute 1 à son compteur
On ajoute le mot dans le dictionnaire et met 1 au compteur