Corrigé Examen Algo2 13/14

SOIT

```
Tab UN TABLEAU ( 10 ) DE ARB DE CHAINES ;
Pos UNE FONCTION ( ENTIER ) ;
Hash UNE FONCTION ( ENTIER ) ;
Inserer UNE FONCTION ( ARB DE CHAINES ) ;
Chaine_prefixee UNE FONCTION ( ARB DE CHAINES ) ;
Prefixe UNE FONCTION ( CHAINE ) ;
Rotg , Rotd DES FONCTIONS ( ARB DE CHAINES ) ;
Transformer UNE ACTION ;
In , Pr DES ACTIONS ;
Init UNE ACTION ;
Taille UN ENTIER ;
I , K DES ENTIERS ;
Une_chaine UNE CHAINE ;
A UN ARB DE CHAINES ;
```

DEBUT

```
Taille := 10;
APPEL Init ;

ECRIRE ( 'Construction de la structure...' ) ;
POUR I := 1 , 200
   Une_chaine := ALEACHAINE ( 20 ) ;
   K := MOD ( Hash ( Une_chaine ) , Taille ) + 1 ;
   AFF_ELEMENT ( Tab [ K ] , Inserer ( ELEMENT ( Tab [ K ] ) , Une_chaine ) ) ;
  { a, nil : pb }

FPOUR ;
ECRIRE ( 'Les arbres' ) ;
POUR I := 1 , Taille
   ECRIRE ( 'Arbre' , I ) ;
   APPEL In ( ELEMENT ( Tab [ I ] ) ) ;

FPOUR ;

ECRIRE ( 'Plus petites chaines commençant par "a"  ' ) ;
POUR I := 1 , Taille
   ECRIRE ( 'Arbre' , I ) ;
   SI Chaine_prefixee ( ELEMENT ( Tab [ I ] ) , 'a' ) <> NIL
      ECRIRE ( INFO ( Chaine_prefixee ( ELEMENT ( Tab [ I ] ) , 'a' ) ) )
   FSI
FPOUR ;

ECRIRE ( 'Transformation des arbres ' ) ;
POUR I := 1 , Taille
   SI Chaine_prefixee ( ELEMENT ( Tab [ I ] ) , 'a' ) <> NIL
      ECRIRE ( 'arbre' , I ) ;
      A := ELEMENT ( Tab [ I ] ) ;
      APPEL Transformer ( A , Chaine_prefixee ( ELEMENT ( Tab [ I ] ) , 'a' ) ) ;
      APPEL Pr ( A )
   FSI
FPOUR
```

```
FIN
{Initialisation du tableau à Nil}
ACTION Init ;
SOIT
   I UN ENTIER ;

DEBUT
   POUR I := 1 , Taille
      AFF_ELEMENT ( Tab [ I ] , NIL )
   FPOUR
FIN
{Rang du caractère C parmis les caractères alpha-numériques}
FONCTION Pos ( C ) : ENTIER ;
SOIT
   I UN ENTIER ;
   C UNE CHAINE ;
   Trouv UN BOOLEEN ;
   Alphabet UNE CHAINE ;

DEBUT
   Alphabet := 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ' ;
   I := 1 ;
   Trouv := FAUX ;
   TQ ( I <= 52 ) ET NON Trouv
      SI C = CARACT ( Alphabet , I )
         Trouv := VRAI
      SINON
         I := I + 1
      FSI ;

   FTQ ;
   Pos := I ;

FIN
{Transformer une chaine en un entier }
FONCTION Hash ( Ch ) : ENTIER ;
SOIT
   I , S DES ENTIERS ;
   Ch UNE CHAINE ;

DEBUT
   S := 0 ;
   POUR I := 1 , LONGCHAINE ( Ch )
      S := S + Pos ( CARACT ( Ch , I ) )
   FPOUR ;
   Hash := S ;
```

```
FIN
{ Inserer la chaine Ch dans l'arbre A}
FONCTION Inserer ( A , Ch ) : ARB DE CHAINES
SOIT
   Ch UNE CHAINE ;
   A , B DES ARB DE CHAINES ;

DEBUT
   SI A = NIL
      CREERNOEUD ( A ) ;
      AFF_INFO ( A , Ch ) ;
      Inserer := A
   SINON
      SI Ch < INFO ( A )
         B := Inserer ( FG ( A ) , Ch ) ;
         AFF_FG ( A , B ) ;

      SINON
         B := Inserer ( FD ( A ) , Ch ) ;
         AFF_FD ( A , B ) ;

      FSI ;
      AFF_PERE ( B , A ) ;
      Inserer := A ;
   FSI
FIN
{Parcours inordre de l'arbre }
ACTION In ( A )
SOIT
   A UN ARB DE CHAINES ;

DEBUT
   SI A <> NIL
      APPEL In ( FG ( A ) ) ;
      ECRIRE ( INFO ( A ) ) ;
      APPEL In ( FD ( A ) ) ;

   FSI
FIN
{Parcours préordre de l'arbre }
ACTION Pr ( A )
SOIT
   A UN ARB DE CHAINES ;

DEBUT
   SI A <> NIL
      ECRIRE ( INFO ( A ) ) ;
      APPEL Pr ( FG ( A ) ) ;
```

```
    APPEL Pr ( FD ( A ) ) ;

  FSI
FIN
{Récuperer les K premiers caractères d'une chaine de caractères}
FONCTION Prefixe ( Une_chaine , K ) : CHAINE ;
SOIT
  Une_chaine , Ch DES CHAINES ;
  K , I DES ENTIERS ;

DEBUT
  Ch := '' ;
  POUR I := 1 , MIN ( K , LONGCHAINE ( Une_chaine ) )
    Ch := Ch + CARACT ( Une_chaine , I )
  FPOUR ;
  Prefixe := Ch
FIN
```

```
{Plus petit mot commençant par Ch dans l'arbre a}
FONCTION Chaine_prefixee ( A , Ch ) : ARB DE CHAINES ;
SOIT
  A , P UN ARB DE CHAINES ;
  Arret UN BOOLEEN ;
  Ch UNE CHAINE ;
 { un car : pb }
DEBUT
  Chaine_prefixee := NIL ;
  P := A ;
  Arret := FAUX ;
  TQ ( P <> NIL ) ET NON Arret
    SI Prefixe ( INFO ( P ) , LONGCHAINE ( Ch ) ) = Ch
      Chaine_prefixee := P ;
      SI Prefixe ( INFO ( P ) , LONGCHAINE ( Ch ) ) > Ch
        Arret := VRAI
      SINON
        P := FG ( P )
      FSI
    SINON
      SI Ch < INFO ( P )
        P := FG ( P )
      SINON
        P := FD ( P )
      FSI
    FSI
  FTQ
FIN
```

```
{Rotation droite du noeud H }
FONCTION Rotd ( H ) : ARB DE CHAINES
```

```
SOIT
   X , H DES ARB DE CHAINES ;

DEBUT
   X := FG ( H ) ;
   AFF_FG ( H , FD ( X ) ) ;
   SI FD ( X ) <> NIL
      AFF_PERE ( FD ( X ) , H )
   FSI ;
   AFF_FD ( X , H ) ;
   AFF_PERE ( H , X ) ;
   Rotd := X ;
FIN
```

```
{Rotation gauche du noeud H }
FONCTION Rotg ( H ) : ARB DE CHAINES
SOIT
   X , H DES ARB DE CHAINES ;

DEBUT
   X := FD ( H ) ;
   AFF_FD ( H , FG ( X ) ) ;
   SI FG ( X ) <> NIL
      AFF_PERE ( FG ( X ) , H )
   FSI ;
   AFF_FG ( X , H ) ;
   AFF_PERE ( H , X ) ;
   Rotg := X ;

FIN
```

```
{ Rendre le noeud d'adresse P la nouvelle Transformer de l'arbre A}
ACTION Transformer ( A , P )
SOIT
   A , P , X , Q , Q2 DES ARB DE CHAINES ;
   Continue UN BOOLEEN ;

DEBUT
   Continue := P <> A ;
   TQ Continue
      Q := PERE ( P ) ;
      Q2 := PERE ( Q ) ;
      SI P = FG ( Q )
         X := Rotd ( Q )
      SINON
         X := Rotg ( Q )
      FSI ;
      SI Q2 = NIL
         Continue := FAUX
```

```
        SINON
          SI FG ( Q2 ) = Q
            AFF_FG ( Q2 , X )
          SINON
            AFF_FD ( Q2 , X )
          FSI ;
          AFF_PERE ( X , Q2 ) ;
          P := X ;
        FSI ;
      FTQ ;
      A := P ;
FIN
```

Inconvénient/Avantage de la structure :

Perte de Taille pointeurs en Ram et
Gagner Taille fois en temps de recherche.