Série 2 corrigé - Algorithme (les fichiers)

Exercice 1 :

Soit le type suivant :

```
Type Produit = Enregistrement
Code : Entier ;
Désignation : Chaîne [ 80 ] ;
Prix : Réel ;
 Fin ;
```

Soit F un fichier de produits. Ecrire une Fonction qui vérifie si les éléments de F sont triés par ordre croissant de leur Code.

Correction :

```
Fonction Ftrie(F :fichier de Produit) :booleen ;
Var Eprod :Produit ;
 Code :entier
Debut
Assigner(F,'Produit.dat') ; Relire(F) ;
Ftrie ←Vrai ;
Si Non FDF(F)
Alors Lire(F,Eprod) ;
Tantque non FDF(F) et Ftrie
Faire code ←Eprod.code ;
Lire(F,Eprod) ;
Si code> Eprod.code Alors Ftrie ← Faux Fsi ;
Fait
Fsi ;
Fin ;
```

EXERCICE 2 :

Soit Fmot un fichier de caractères alphabétiques contenant des mots séparés par un ou plusieurs caractères blanc.

1- Écrire une fonction Palindrome qui vérifie si un mot donné est un mot palindrome.

2- Écrire un algorithme qui affiche le nombre de mots palindromes et le plus court mot palindrome.

Correction : 1-

```
Fonction Palindrome(mot :chaine) :booléen ;
Var I,T :entier ;
Debut
T←Taille(mot) ; Plaindrome ←vrai ; I ←1 ;
Tantque Palindrome et I< T DIV 2
Faire Si mot[I]<>mot[T-I+1]
Alors Palindrome ←Faux Fsi ;
```



```
I ←I+1 ;
Fait ;
Fin ;
2-
Algorithme CourtPal ;
Type ch :chaine[50];
Var Fmot :fichier de caractère ;
 tmin,t,NBpal :entier ;
 X :caractère ;
 Mot,cpal :ch ;
 Fonction Palindrome(mot :chaine) :booléen ;
------
Debut
Assigner(Fmot,'fmot') ; relire(Fmot) ;
tmin ←51 ;cpal ←'' ; NBpla← 0;
Tantque non FDF(Fmot)
Faire Lire(Fmot,X) ; /*récupérer un mot
Mot←'' ; /*initialiser à vide
Tantque non FDF(Fmot) et (X<>' '
Faire
Mot←Mot+X ;
Lire(Fmot,X) ;
Fait ;
/*traiter le dernier caractère
Si X<>' ' Alors Mot←Mot+X Fsi

Si Mot<>''
Alors
Si Palindrome(mot)
Alors t ←Taille(mot) ; NBpal←NBpal+1 ;
Si t<tmin
Alors tmin ←t ;
Cpal ←mot ;
Fsi ;
Fsi ;
Fsi ;
Fait ;
Si NBcpal<>0
Alors Ecrire('Le nombre de mots palindrome est :',NBpal
 Ecrire('le plus court mot palindrome est :',cpal)
Sinon Ecrire('Pas de mots palindrome ou fichier est vide')
Fsi ;
Fermer(Fmot) ;
Fin.
```


EXERCICE 3 :

Soit F un fichier d'entiers (supposé existant) composé de séquences de nombres, chaque séquence est une répétition du même nombre (non nul). Toutes les séquences sont séparées par un zéro. Et aucune séquence du même nombre ne se répète dans le fichier.

F 14 14 14 0 5 5 0 29 29 29 29 0 6 6 6
Position 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

1- Ecrire une action paramétrée Compresser qui crée un fichier G d'enregistrement contenant pour chaque séquence le nombre représenté ainsi que la longueur de la séquence.

G

Nombre	14	5	29	6
Longueur	3	2	4	3

2- En utilisant un seul parcours du fichier G et sans reparcourir le fichier F, trouver la position dans F de la plus longue séquence. (ex : Position = 8)

3- Ecrire une action paramétrée Decompresser qui permet de reconstruire un fichier H (de même type que F) à partir d'un fichier de même ne type que G.

Correction :

1-

```
Type
Elcp=Enregistrement
Val,Long :entier ;
 Fin ;
Fent : Fichier de Entier ;
Felcp : Fichier de Elcp ;
L'assignation se fait à l'extérieur comme suit :
Assigner(F,'Fname') ; Assigner(G,'Gname') ;
Procedure Compresser(E/S/ F : Fent ; E/S/ G : Felcp) ;
Var X,I:Entier; EC:Elcp;
Debut
Relire (F); Réecrire (G);
Tantque Non FDF(F)
Faire Lire (F,X) ;
EC.Val ← X ; I ← 0 ;
Tantque Non FDF(F) et X<>0
FaireI ← I+1 ;
Lire (F,X) ;
Fait ;
Si FDF(F) Alors I ← I+1 Fsi;
EC.Long ← I ; Ecrire (G,EC) ;
Fait ;
Fermer(F) ; Fermer(G) ;
Fin ;
```

2-

```
Fonction PosMax(G :Felcp) : Entier ;
```



```
Var K,Max : Entier ; EC :Elcp ;
Debut
Relire(G) ; Max ← 0 ; K ← 0 ;
Tantque Non FDF(G)
Faire Lire(G,EC) ;
Si EC.Long > Max
Alors Max ← EC.Long ;
 PosMax ← K+1
Fsi ;
K ← K+EC.Long+1;
Fait ;
Fermer(G) ;
Fin ;
```

3-

```
Procedure Decompresser(E/S/ G :Felcp ; E/S/ F : Fent) ;
Var I:Entier; EC:Elcp;
Debut
Relire(G) ; Réecrire(F) ;
Tantque Non FDF(G)
Faire Lire(G,EC) ;
 Pour I ← 1 à EC.Long
 Faire Ecrire(F, EC.Val) ; Fait ;
 Si Non FDF(G) Alors Ecrire(F,0) Fsi ;
Fait ;
Fermer(G) ; Fermer(F) ;
Fin ;
```

Exercice 4 :

Considérons le type enregistrement suivant :

```
Type Etudiant = Enregistrement
Matricule : entier ;
Nom, Prenom : chaine [20] ;
Moyenne : réel ;
 Fin;
```

Soit T un tableau d'au plus 100 étudiants.

Ecrire un algorithme permettant de recopier tous les étudiants admis appartenant à T dans un fichier ADMIS de
type étudiant. Un étudiant est admis si sa moyenne est supérieure ou égale 10.

Correction :

```
Algorithme Etude ;
Type Etudiant = Enregistrement
Matricule : entier ;
Nom, Prenom : chaine [20] ;
Moyenne : réel ;
 Fin;
Var T :Tableau[1..100] de Etudiant ;
```

```
F :Fichier de Etudiant ;
X :Etudiant ;
I,N :entier ;

Debut
Ecrire('Donner le nombre d''etudiants') ;
/*lecture des éléments du tableau
Repeter Lire(N) Jusqu'à N>0 et N≤100 ;
Pour I←1 à N
Faire
Avec X
Faire Lire(Matricule) ;
Lire(Nom,Prenom) ;
Lire(Moyenne) ;
Fait ;
T[I]←X ;
/*On peut utiliser directement T[I] et on évite l'affectation ( en bleu )
Avec T[I]
Faire Lire(Matricule) ;
Lire(Nom,Prenom) ;
Lire(Moyenne) ;
Fait ;
Fait ;
/*création du fichier des admis
Assigner(F,'ADMIS') ; Reecrire(F) ;
Pour I←1 à N
Faire
Si T[I].Moyenne)≥10 Alors Ecrire(F,T[I]) Fsi;
/*même remarque, on peut utiliser un enregistrement X
X←T[I] ;
Si X.Moyenne)≥10 Alors Ecrire(F,X) Fsi ;
Fait ;
Fermer (F) ;
Fin.
```