Sommaire

Chapitre 2 : Enregistrement et Fichiers (Partie 1)

2.1. Notion d'Enregistrement

Un enregistrement est une structure (complexe) de données permettant de représenter un ensemble de données (dites champs) hétérogènes, c'est-à-dire de types différents.

On peut considérer un enregistrement comme étant un tableau à une dimension, où chaque case (composante) du tableau représente un champs dans l'enregistrement. La différence réside dans :

- Une case de tableau est accessible à travers un indice entier, par contre un champs est repéré par un nom (identificateur) ;
- Les cases du tableau sont toutes du même type (structure homogène), par contre, les champs de l'enregistrement ne sont pas obligatoirement du même type (structure hétérogène).

Exemple

Pour représenter les informations liées à un produit quelconque dans une même structure, on peut utiliser un enregistrement constitué de champs suivants :

- La désignation du produit
- La référence du produit
- Sa quantité en stock
- Son prix unitaire
- Etc…

Les informations précédentes (les champs de l'enregistrement) sont de types différents, elles ne peuvent être décrites par un tableau (dont les cases sont du même type). Les types de ces informations sont respectivement :

- Chaîne de caractères
- Chaîne de caractères
- Entier
- Réel

2.2. Déclaration d'un Enregistrement

La syntaxe générale pour déclarer un enregistrement est comme suit :

En Algorithme

```
Type <id_Enreg> = Enregistrement
    <id_ch1> : <type1>
    <id_ch1> : <type2>
    ...
    <id_chN> : <typeN>
Fin
```

En Pascal

```
Type <id_Enreg> = RECORD
    <id_ch1> : <type1>
    <id_ch1> : <type2>
    ...
    <id_chN> : <typeN>
End;
```

Où <id_ch1>, <id_ch2>, …, <id_chN> sont les identificateurs des champs et <type1>, <type2>, …, <typeN> leurs types respectifs.

Pour l'exemple du Produit précédent, la déclaration de l'enregistrement sera comme suit :

En Algorithme

```
Type Produit = Enregistrement
    Designation : Chaîne
    Reference : Chaîne
    Quantite : Entier
    Prix_U : Réel
Fin
```

En Pascal

```
Type Produit = RECORD
    Designation : String[30];
    Reference : String[13];
    Quantite : Integer;
    Prix_U : Real;
End;
```

2.3. Accès aux champs d'un enregistrement

Pour référencer ou accéder à un champ quelconque d'un enregistrement, on utilise la notation pointée. La notation pointée utilise la variable enregistrement (identificateur) suivie d'un point puis du nom du champ auquel on veuille accéder. Soit :

```
<Variable_Enregistrement>.<Nom du champ>
```

Exemple :

Un nombre complexe peut être décrit par un enregistrement . Soit :

```
Type  COMPLEX = Enregistrement
    X,Y : Réel
Fin
```

Un nombre complexe peut alors être décrit par un enregistrement de deux champs : X est la partie réelle et Y est la partie imaginaire. Par exemple :

```
NBR : Complex {dans la partie déclaration du programme}
…
{dans la partie corps du programme}
NBR.X = 10;
NBR.Y =-2;
```

Ce qui correspond au nombre complexe *NBR = 10 - 2i*

2.4. L'instruction WITH

L'instruction PASCAL **WITH**, permet d'éviter les répétitions de la variable enregistrement lors de l'accès à ses différents champs. Sa syntaxe générale est :

```
With  <var_enreg> Do
Begin
     <instructions_champs_enreg>;
End;
```

Dans le cas des deux affectations précédentes, on peut écrire :

```
With   NBR  Do
Begin
     X := 10 ;
     Y :=-2
End;
```

Avec l'instruction **WITH**, on évite de répéter la variable d'enregistrement pour chaque champs.

Pour l'exemple de Produit (premier exemple) :

```
var p:Produit;
...
With  P  Do
Begin
     designation := 'Ordinateur' ;
     reference := 'HP Z3098';
     Quantite := 10;
     Prix_U := 29000.00;
End;
```

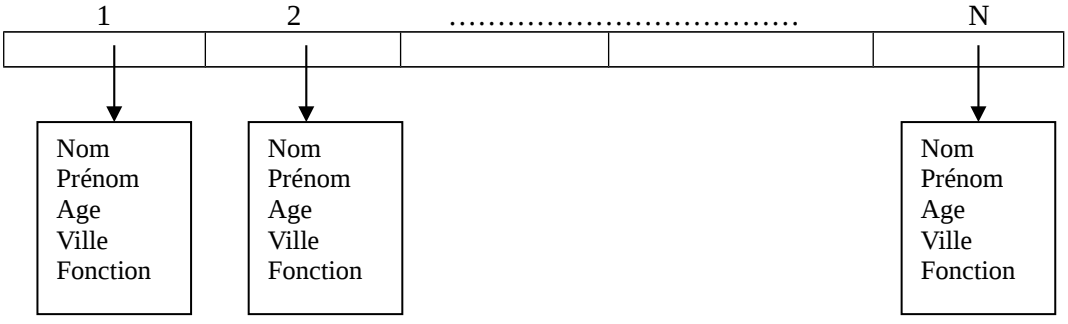
Exemple (Tableau d'enregistrement)

Considérons une base de données comportant des individus décrits par les informations suivantes :

- Nom
- Prénom
- Age
- Ville
- Fonction

On veut organiser ces informations de telle sorte qu'on puisse sélectionner les individus ayant un certain âge, ou habitant une certaine ville ou ayant un fonction donnée etc… De façon générale gérer la base de données. Proposer une structure de donnée.

La structure de donnée :



a)- Écrivons d'abord la séquence d' initialisation de la base de données :

```
Algorithme  LireEcrire
  Type Individu = Enregistrement
      Nom, Prénom : chaine
      Age : entier
      Ville, Fonction : Chaine
  Fin

  Variables
      T : Tableau [1..1000] de Individu
      i,N : entier
Début
      Lire (N) { N est le nombre d'individus }

      Pour  i = 1 à N Faire
            Lire(T[i]. Nom)
            Lire(T[i]. Prénom)
            Lire(T[i]. Age)
            Lire(T[i]. Ville)
            Lire(T[i]. Fonction)
      FinPour

      Pour  I = 1 à N Faire
          Ecrire(T[I]. Nom)
          Ecrire(T[I]. Prénom)
          Ecrire(T[I]. Age)
          Ecrire(T[I]. Ville)
          Ecrire(T[I]. Fonction)
      Fin Pour
FIN
```

```
PROGRAM  LireEcrire;
uses wincrt;
  Type Individu = RECORD
      Nom, Prénom : string[20] ;
      Age : integer ;
      Ville, Fonction : string[25] ;
  End;

Var
      T : array[1..1000] of Individu;
      I,N : integer ;
BEGIN
      Writeln('Introduire le nombre d'' individus : ') ;
      readln (N) { N est le nombre d'individus }
      Writeln('Introduire Nom,Prénom,Age,Ville, Fonction de chaque Individu ') ;
      For  I := 1 to N do
       begin
            Readln (T[I].Nom);
            Readln(T[I].Prenom);
            Readln(T[I].Age);
            Readln(T[I].Ville);
```

```
        Readln(T[I].Fonction);
    End;

   For  I := 1 to N do
    begin
        Write(T[I].Nom);
        Write(T[I].Prenom);
        Write(T[I].Age);
        Write(T[I].Ville);
        Write(T[I].Fonction);
        Writeln;
    end;
END.
```

Exercice :

Écrire la séquence de programme qui permet de lister seulement les individu ayant l'âge entre 20 et 30 et résidant à BEJAIA.

Il faut alors tester : IF (T[I].Age >=20) AND (T[I].Ville = 'BEJAIA') avant d'écrire l'enregistrement correspondant.

2.5. Les Fichiers

Un fichier est une collection de données. PASCAL permet de créer et de manipuler 3 types de fichiers.

- Les Fichiers à 'accès directe' de type **FILE OF**.
- Les Fichiers à accès séquentiel de type **TEXT**
- Les Fichiers 'sans type' de type **FILE**.

Les fichiers de type **FILE OF** sont constitués d'enregistrements de même type, alors que les fichiers TEXT admettent des enregistrements de type différents.

Déclaration de fichier en PASCAL :

```
<Nom Fichier> : FILE OF  <Type_Enregistrement> ;
<Nom Fichier> : TEXT ;
<Nom Fichier> : FILE ;
```

Dans la suite, on s'intéressera uniquement aux fichier de type **FILE OF**.

2.5.1. Les fichiers de type FILE OF

Exemple : Soit la séquence PASCAL suivante :

```
Type  ETUDIANT = RECORD
                   Nom : string[20] ;
                   Prenom : string[20] ;
                   Age : integer ;
      End ;
Var
      F1 : FILE OF  ETUDIANT ;  F2 : FILE OF Char ;
      F3: FILE OF Real;
```

On a ainsi défini trois fichiers, le premier fichier F1 est constitué d'enregistrement de type ETUDIANT, et le deuxième F2 est constitué de caractères, et le troisième F3 comporte des réels.

2.5.2. Les fonctions et procédures sur les fichiers de type FILE OF

i) RESET (Fichier)

Procédure qui ouvre un fichier déjà existant et le prépare pour une lecture ou une écriture. Le pointeur de fichier est positionné sur le premier enregistrement du fichier. (ouverture en lecture/écirture)

ii) READ, WRITE, READLN, WRITELN

Lecture ou écriture sur un fichier de type FILE OF.

Exemple

```
READ(Fichier, Var1,Var2, …,Varn);
WRITE(Fichier, Var1,Var2,…,Varn);
```

Idem pour READLN et WRITELN.

iii) SEEK (Fichier, N) :

Procédure permettant de positionner le pointeur de Fichier sur l'enregistrement Numéro N.

iv) FILEPOS(Fichier)

Fonction qui retourne la position du pointeur du fichier spécifié.

v) FILESIZE (Fichier)

Fonction qui retourne le nombre total d'enregistrements du fichier.

vi) REWRITE(Fichier) :

Procédure qui crée le fichier spécifié s'il n'existait pas. Dans le cas où il existait, il sera détruit. (ouverture du fichier en écriture)

vii) CLOSE(Fichier)

Procédure qui ferme le fichier.

viii) EOF(Fichier)

Fonction qui retourne TRUE si la fin de fichier est rencontrée sinon retourne FALSE.

ix) ASSIGN(Fichier, 'Nom_Physique') :

Procédure permettant d'associer le nom logique du fichier 'Fichier' au nom physique du fichier.

2.6. Exemples

2.6.1. Exemple 1 : Ouverture et fermeture des fichiers

```
ASSIGN(F1,'CLIENT.DAT') ;
REWRITE(F1) ; {création et ouverture du fichier F1 en écriture}
      … … …
{ initialisation du fichier avec READ/WRITE}
      … … …
CLOSE(F1) ; {Fermeture du fichier}


RESET(F1); {Ouverture du fichier en lecture/écriture}
      … … …
      … … …
{Manipulation du fichier READ,WRITE,SEEK etc…  }
      … … …
      … … …

CLOSE(F1) ; {fermeture du fichier}
```

2.6.2. Exemple 2 : Notes des TPs des étudiants

(A Tester sur machine pour comprendre)

```
PROGRAM  RESULTATS ;
       USES  WINCRT;
       Label 1 ;
TYPE    INDIVIDU = RECORD
             Nom : string[20] ;
             Prenom : string[20] ;
             Tp1 : real ;
             Tp2 : real ;
 End ;
VAR         ETUDIANT : array[1..100]  OF  INDIVIDU ;
            Moy : real ;
            I,N :integer ;
            Fin : Boolean ;
            Fichier1 : FILE OF  INDIVIDU ;

 BEGIN
         ASSIGN(Fichier1,'D:\eleves.PAS') ;
         Rewrite(Fichier1) ;
         Writeln('SAISIE DES DONNES DANS Fichier1') ;
      Fin := False ;
      I :=1 ;
      WHILE  NOT Fin  DO
           BEGIN
           Writeln('Appuyer sur ENTREE Pour  Terminer la saisie') ;
           Write('LE NOM : ') ;READLN(ETUDIANT[I].Nom) ;
           If (ETUDIANT[I].Nom  = '') then
                goto 1 ;
```

```
           Write('LE PRENOM : ') ;READLN(ETUDIANT[I].Prenom) ;
           Write('NOTE DE TP1 : ') ;READLN(ETUDIANT[I].Tp1) ;
           Write('NOTE DE TP2 : ') ;READLN(ETUDIANT[I].Tp2) ;
           Write(Fichier1,ETUDIANT[I]) ;
           I :=I+1 ;
         End ;
1 :  Close(Fichier1) ;

     writeln('LECTURE DU FICHIER ET CALCULE DE LA MOYENNE DES TP') ;
     {Elle n'est pas nécessaire mais on le fait quand même }
     {car les infos dont on a besoin se trouvent déjà dans le tableau ETUDIANT}
     RESET(Fichier1) ;
     I :=1 ;
     WHILE  NOT  EOF(Fichier1)  DO
       BEGIN
            READ(Fichier1,ETUDIANT[I]) ;
            I :=I+1 ;
       End ;
      Writeln('AFFICHAGE DES RESULTATS : APPUYER SUR UNE TOUCHE') ;
      Readln;Readln;
      N := I-1 ;
      FOR  I := 1  TO  N  DO
      BEGIN
       MOY :=  (ETUDIANT[I].Tp1  + ETUDIANT[I].Tp2)/2 ;
       write(ETUDIANT[I].Nom) ;
       write(ETUDIANT[I].Prenom) ;
       write(ETUDIANT[I].Tp1:6 :2); {avec 2 chiffres après la virgule}
       write(ETUDIANT[I].Tp2:6 :2) ;
       write(MOY:6 :2) ;
       writeln;
      End ;
     CLOSE(Fichier1) ;
     Readln ;Readln ;
END.
```

2.6.3. Exemple 3 : Nombres complexes

(A tester également sur machine)

Soit un tableau T contenant N nombres complexes, écrire un programme qui conserve le tableau dans un fichier 'complex.pas' et recherche l'élément du tableau le plus grand en module.

```
PROGRAM  maximum;
      USES  WINCRT;
TYPE   complex = RECORD
       X: real ;
       Y: real
 End ;
VAR             T: array [1..100]  OF  complex ;
                I,N,Pos :integer ;
                max :real;
```

```
                F1 : FILE OF  complex ;
 BEGIN
      ASSIGN(F1,'D:\complex.PAS') ;
      Rewrite(F1) ;
      Write('INTRODUIRE  N  ') ;  READ(N) ;
      Writeln('SAISIE DES DONNES DANS F1') ;
      For I :=1 to N do
       BEGIN
            Write('Partie reelle : ') ;read(T[I].X) ;
            Write('Partie imaginaire : ') ;read(T[I].Y) ;
            Write(F1,T[I]) ;
       End ;


      Writeln('LECTURE DU FICHIER ') ;  {non nécessaire, mais on}
      {le montre quand même}
      RESET(F1) ;
      I :=1 ;
      WHILE  NOT  EOF(F1)  DO
        BEGIN
             READ(F1,T[I]) ;
             I :=I+1 ;
        End ;
       Writeln('AFFICHAGE DES RESULTATS : APPUYER SUR UNE TOUCHE') ;
       Readln;Readln;
       N:=I-1;
       {Recherche du plus grand en module }
       Max := SQRT(SQR(T[1].X)+SQR(T[1].Y));
       FOR  I := 2  TO  N  DO
       BEGIN
           If  Max <  SQRT(SQR(T[I].X)+SQR(T[I].Y))Then
                 Begin
                      Max  :=  SQRT(SQR(T[I].X)+SQR(T[I].Y));
                      Pos := I ;
                 End ;
        End;
        Write(' Le PLUS GRAND EN MODULE EST : ');
        writeln(T[Pos].X,'  ', T[Pos].Y) ;
        write('  Sa POSITION EST : ', Pos) ;
        CLOSE(F1) ;
        Readln ;Readln ;
END.
```