Université A.Mira – Bejaia
Faculté de Technologie
Département de Technologie
1ère Année ST – Cycle Licence

Année Universitaire : 2022-2023

Semestre 2. Module : Informatique II

Chapitre III

Enregistrements & Fichiers (Résumé)

Sommaire

Chapitre III : Enregistrement et Fichiers

III.1. Déclaration d'Enregistrement

Un enregistrement est un type complexe de données qui permet de regrouper plusieurs données dans une seule structure.

C'est mieux de déclarer les enregistrements comme des types, et ceci en suivant la syntaxe ci-dessous :

En Algorithme

```
Type <id_Enreg> = Enregistrement
   <id_ch1> : <type1> ;
   <id_ch2> : <type2> ;
   ...
   <id_chN> : <typeN> ;
Fin
```

En Pascal

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

Tel-que : <id_ch1>, <id_ch2> …, <id_chN> sont des champs et <type1>, …, <typeN> sont leur types respectifs.

Par exemple :

On peut déclarer un type Produit 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;
```

III.2. Affectation, lecture et affichage des enregistrements

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 veut accéder. Soit :

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

Exemple :

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

En Algorithme

```
Type Complexe = Enregistrement
   X, Y : réel
Fin;
```

En Pascal

```
Type Complexe = RECORD
   X, Y : real;
End;
```

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

Algorithme	#	Programme PASCAL
Algorithme Exemple01;	01	Program Exemple01;
Type	02	Type
Complexe = Enregistrement	03	Complexe = Record
x, y : réel;	04	x , y : real;
Fin;	05	End;
Variables	06	Var
nb1, nb2, nb : Complexe;	07	nb1, nb2, nb : Complexe;
Début	08	Begin
écrire ('Donne le premier nombre nb1 : ') ;	09	Write ('Donne le premier nombre nb1 : ') ;
Lire(nb1.x, nb1.y) ;	10	Read(nb1.x, nb1.y) ;
	11	
écrire ('Donne le deuxième nombre nb2 : ') ;	12	Write ('Donne le deuxième nombre nb2 : ') ;
Lire(nb2.x, nb2.y) ;	13	Read(nb2.x, nb2.y) ;
	14	
nb.x ← nb1.x + nb2.x;	15	nb.x:= nb1.x + nb2.x;
nb.y ← nb1.y + nb2.y;	16	nb.y := nb1.y + nb2.y;
	17	
écrire ('Nb = ', nb.x, ' + ', nb.y, ' * i ') ;	18	write ('Nb = ', nb.x, ' + ', nb.y, ' * i ') ;
Fin.	19	End.

Ce qui correspond au nombre complexe $Nbr = 10 - 2i$

III.3. L'instruction WITH (Avec)

L'instruction PASCAL **WITH**, permet d'utiliser les champs de l'enregistrement sans répéter la variable d'enregistrement.

En Algorithme	*En Pascal*

Avec <var_enreg> **Faire**
<instructions_champs_enreg>;
Fin-Avec;
	With <var_enreg> **Do**
	Begin
	<instructions_champs_enreg>;
	End;

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

Algorithme	#	Programme PASCAL

Algorithme Exemple01;
Type
Complexe = Enregistrement
x, y : réel;
Fin;
Variables
nb1, nb2, nb : Complexe;
Début
écrire ('Donne le premier nombre nb1 : ') ;
Lire(nb1.x, nb1.y) ;
 	
écrire ('Donne le deuxième nombre nb2 : ') ;
Lire(nb2.x, nb2.y) ;
 	
Avec nb faire
x ← nb1.x + nb2.x;
y ← nb1.y + nb2.y;
 	
écrire ('Nb = ', x, ' + ', y, ' * i ') ;
Fin-Avec;
Fin.
	01
	02
	03
	04
	05
	06
	07
	08
	09
	10
	11
	12
	13
	14
	15
	16
	17
	18
	19
	20
	21
		Program Exemple01;
		Type
		Complexe = Record
		x , y : real;
		End;
		Var
		nb1, nb2, nb : Complexe;
		Begin
		Write ('Donne le premier nombre nb1 : ') ;
		Read(nb1.x, nb1.y) ;
		
		Write ('Donne le deuxième nombre nb2 : ') ;
		Read(nb2.x, nb2.y) ;
		
		with nb do begin
		x:= nb1.x + nb2.x;
		y := nb1.y + nb2.y;
		
		write ('Nb = ', nb.x, ' + ', nb.y, ' * i ') ;
		end;
		End.

Avec l'instruction **WITH**, on évite de répéter la variable d'enregistrement pour chaque champs. Un deuxième exemple, avec le type enregistrement Produit :

Algorithme	#	Programme PASCAL
Algorithme Exemple03; Type Produit = Enregistrement Designation : Chaîne[40]; Reference : Chaîne[20]; Quantite : entier; Prix_U : réel;	01 02 03 04 05 06 07	Program Exemple03; Type Produit = Record Designation : String[40]; Reference : String[20]; Quantite : integer; Prix_U : real;

```
                Fin;
  Variables
      p : Produit;
Début
    Avec p faire
      Designation ← 'Ordinateur';
      Reference ← 'HP-Z00365';
      Quantite ← 6;
      Prix_U ← 57000.00;

      écrire ('Produit : ') ;
      écrire('    Désignation : ', Designation);
      écrire('    Référence : ', Reference);
      écrire('    Quantité : ', Quantite);
      écrire('    Prix Unitaire : ', Prix_U);
    Fin-Avec;
Fin.
```

```
08              End;
09  Var
10      p: Produit ;
11  Begin
12      With p do begin
13          p.Designation := 'Ordinateur';
14          Reference :=  'HP-Z00365';
15          Quantite :=  6;
16          Prix_U :=  57000.00;
17
18          writeln ('Produit : ') ;
19          writeln('    Désignation : ', Designation);
20          writeln('    Référence : ', Reference);
21          writeln('    Quantité : ', Quantite);
22          writeln('    Prix Unitaire : ', Prix_U:0:2);
23      end;
24  End.
```

III.4. Les Fichiers

Un fichier est une collection de données sauvegardée dans un support de stockage secondaire : Disque dur, flash-disk, CD, … . En langage de programmation, un fichier est une structure de données permettant de manipuler les données sauvegardées en dehors de la mémoire centrale. En langage PASCAL, nous pouvons manipuler 3 types de fichiers.

- Les Fichiers à 'accès direct' (*structurés*) de type **FILE OF**.
- Les Fichiers à accès séquentiel (*Textuel*) de type **TEXT**
- Les Fichiers 'sans type' (*Binaire*) de type **FILE**.

Dans ce qui suit, nous allons voir uniquement les fichiers à accès direct : FILE OF … Ce type de fichier est constitué de blocs homogènes (de même type) dits : cellules, pouvant être de type simple : entier, réel, caractères, … ou de type enregistrements.

III.4.1. Éléments d'un fichier

Parmi les éléments qui sont liés au fichier, nous allons voir :

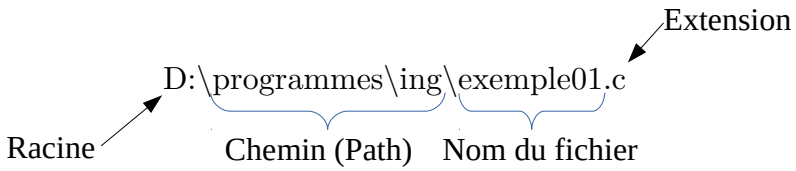
- *Nom externe (nom physique)*
- *Nom interne (nom logique)*

- *Tampon (Buffer)*
- *FDF : Fin De Fichier (EOF : End Of File)*
- *Indice (pointeur interne)*

a) Nom externe (nom physique)

Une chaîne de caractère contenant **l'**identifiant du support (Racine), le chemin vers le fichier (path), le nom du fichier proprement et son extension du fichier

Exemples :

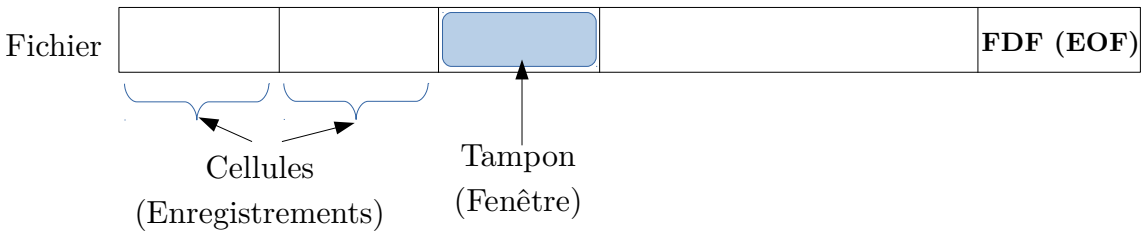


b) Nom interne (nom logique)

Nom interne est le nom avec lequel le fichier est identifié dans un programme (algorithme). C'est le nom logique qui doit être associé avec le nom externe (le nom physique). Plus précisément, un nom interne (logique) est un identificateur d'une variable pour accéder au données physiques du fichier.

c) Tampon (Buffer)

On appelle tampon (buffer) d'un fichier, une zone de la mémoire central (RAM) pouvant contenir une cellule (éventuellement un enregistrement) de ce fichier. C'est à travers ce tampon qu'on rends les cases du fichier visible : pour cela, on appelle le tampon ***fenêtre*** à travers laquelle on *voit* le fichier.



d) FDF (EOF)

Chaque fichier se termine par une cellule spéciale dite FDF (pour Fin de Fichier), en anglais c'est : EOF (**End Of File**). Ça permet de savoir la fin de fichier lors du parcours

des données du fichiers (voir les algorithmes et programmes) dans la suite du cours).

e) Indice (pointeur)

Pour chaque fichier, un indice (pointeur interne) caché, qui est géré automatiquement par le type fichier, est utilisé pour lire ou écrire la cellule pointée par cet indice.

III.4.2. Manipulation des fichiers de type FILE OF

Dans ce qui suit, nous allons voir comment :

- *Déclarer un fichier de type* ***File Of***
- *Créer un nouveau fichier*
- *Lire et Afficher le contenu d'un fichier*

Pour illustrer les opération ci-dessus, nous prenons un exemple de fichier d'étudiants, où chaque étudiant est caractérisé par : *Matricule, Nom, Prénom, date de naissance, filière, niveau et sa moyenne.*

a) Déclaration de fichier

L'exemple ci-dessous, explique comment déclarer un fichier de type **File Of** contenant des enregistrements d'étudiant :

Algorithme	#	Programme PASCAL

Algorithme Exemple06_Fichier;
Type
Etudiant = Enregistrement
Matricule : Chaîne[15] ;
Nom, Prenom : Chaîne[30];
Date_N : Chaîne[10];
Filiere : Chaîne[50];
Niveau : entier ;
Moyenne : réel;
Fin;
Variables
F : Fichier de Etudiant ;
Début
 	
Fin.
	01
	02
	03
	04
	05
	06
	07
	08
	09
	10
	11
	12
	13
	14
		Program Exemple06_Fichier;
		Type
		Etudiant = Record
		Matricule : String[15] ;
		Nom, Prenom : String[30];
		Date_N : String[10];
		Filiere : String[50];
		Niveau : integer ;
		Moyenne : real;
		End;
		Var
		F : File of Etudiant;
		Begin
		
		End.

Le lien du programme PASCAL ci-dessus : https://onlinegdb.com/Kgdczxk8A Lien de l'explication :		

Remarque :

- Un fichier peut être vu comme un tableau à une dimension (vecteur) illimité (pratiquement illimité) sauvegardé sur une mémoire secondaire

- Les cases (cellules) du fichier sont accessible par un pointeur invisible.

b) Création d'un nouveau fichier

Pour créer un nouveau fichier, on doit :

1- Déclarer la structure de fichier : on continue sur l'exemple 6 (fichier d'étudiants)

2- Assigner le nom logique du fichier avec le nom physique (chemin et nom complets du fichier)

3- Ouvrir le fichier en écriture : Réécrire / Rewrite

4- écrire les données dans un fichier

5- fermer le fichier

L'exemple illustre les étapes ci-dessus pour créer un nouveau fichier :

Algorithme	#	Programme PASCAL
Algorithme Exemple08_Fichier_Creer;	01	Program Exemple08_Fichier_Creer;
Type	02	Type
Etudiant = Enregistrement	03	Etudiant = Record
Matricule : Chaîne[15] ;	04	Matricule : String[15];
Nom, Prenom : Chaîne[30];	05	Nom, Prenom : String[30];
Date_N : Chaîne[10];	06	Date_N : String[10];
Filiere : Chaîne[50];	07	Filiere : String[50];
Niveau : entier ;	08	Niveau : integer ;
Moyenne : réel;	09	Moyenne : real;
Fin;	10	End;
Variables	11	Var
F : Fichier de Etudiant ;	12	F : File of Etudiant;
N, i : entier ;	13	N, i : integer ;
E : Etudiant ;	14	E : Etudiant ;
Début	15	Begin
Assigner (F, 'etudiants.dat ') ;	16	Assign(F, 'etudiants.dat ') ;
Réécrire(F) ;//Ouvrir le fichier en écriture	17	Rewrite(F);//Ouvrir le fichier en écriture
écrire ('Données le nombre d'étudiants : ') ;	18	write ('Données le nombre d'étudiants : ') ;
Lire(N);	19	Read(N);
écrire ('Données les informations : ') ;	20	write ('Données les informations : ') ;
Pour i←1 à N faire	21	for i:=1 to N do begin
Avec e faire	22	with e do begin
Lire(Matricule, Nom, Prenom) ;	23	Readln(Matricule, Nom, Prenom) ;

```
        Lire(Date_N);                          |        Readln(Date_N);
        Lire(Filiere) ;                        |        Readln(Filiere) ;
        Lire(Niveau) ;                         |        Readln(Niveau) ;
        Lire(Moyenne);                         |        Readln(Moyenne);
      Fin-Avec;                                |      end;
      écrire(F, e);//Ajouter e au fichier F    |      write(F, e);//Ajouter e au fichier F
    Fin-Pour;                                  |    end;
                                               |
    Fermer(F) ;                                |    Close(F) ;
Fin.                                           |  End.
```

Le lien du programme PASCAL ci-dessus : https://onlinegdb.com/Cegk4OtIM
Lien de l'explication :

Remarques

- L'instruction Réécrire(F); / Rewrite(F); permet de créer un nouveau fichier si le fichier n'existe pas, sinon (le fichier existe), cette instruction écrasera le fichier existant : ***IL FAUT FAIRE ATTENTION POUR NE PAR PERDRE DES DONNÉES.***

- L'instruction Écrire(F, e), en mode écriture, permet d'ajouter l'enregistrement e à la fin du fichier F.

- À la fin du programme, il ne faut pas oublier de fermer le fichier F : Fermer(F) ;/ Close(F);

c) Lire et afficher le contenu d'un fichier

Une fois un fichier est crée, on peut lire son contenu en l'ouvrant en mode lecture. L'exemple ci-dessous, montre comment lire le contenu du fichier ***etudiants.dat*** affiche son contenu sur l'écran :

Algorithme	#	Programme PASCAL
Algorithme Exemple09_Fichier_Afficher;	01	Program Exemple09_Fichier_Afficher;
Type	02	Type
Etudiant = Enregistrement	03	Etudiant = Record
Matricule : Chaîne[15] ;	04	Matricule : String[15] ;
Nom, Prenom : Chaîne[30];	05	Nom, Prenom : String[30];
Date_N : Chaîne[10];	06	Date_N : String[10];
Filiere : Chaîne[50];	07	Filiere : String[50];
Niveau : entier ;	08	Niveau : integer ;
Moyenne : réel;	09	Moyenne : real;

```
                Fin;
  Variables
       F : Fichier de Etudiant ;
       N, i : entier ;
       E : Etudiant ;
Début
    Assigner (F, 'etudiants.dat ') ;
     Reouvrir(F) ;//Ouvrir le fichier en Lecture/éc.

    écrire ('Le contenu du fichier etudiants : ') ;
    Tantque non(Eof(F)) faire
       Lire(F, e); //Lire e à partir du fichier F
       Avec e faire
          Écrire(Matricule, Nom, Prenom, Date_N);
          Écrire(Filiere, Niveau, Moyenne);
       Fin-Avec;
    Fin-Tantque;

    Fermer(F) ;
Fin.
```

```
10
11
12
13
14
```

```
                End;
  Var
       F : File of Etudiant;
       i : integer ;
       E : Etudiant ;
Begin
    Assign(F, 'etudiants.dat ') ;
    Reset(F);//Ouvrir le fichier en lecture/écriture

    write ('Le contenu du fichier etudiants : ') ;
    While not(Eof(f)) do begin
       Read(F, e); //Lire e à partir du fichier F
       With e do begin
          Writeln(Matricule, Nom, Prenom, Date_N);
          Writeln(Filiere, Niveau, Moyenne);
       end ;
    end ;

    Close(F) ;
End.
```

Le lien du programme PASCAL ci-dessus : https://onlinegdb.com/7cLdsW5pC
Lien du fichier *etudiants.dat* : https://www.dropbox.com/s/89nhm06qcdzdxg3/etudiants.dat?dl=0

Lien de l'explication :

Remarques

- L'instruction Reourvir(F); / Reset(F) permet d'ouvrir fichier existant pour lire son contenu ou le modifier. Si le fichier n'existe pas, il y aura une erreur pendant l'exécution : Runtime Error (Exception). Le pointeur de fichier est automatiquement au début du fichier (sur le premier enregistrement).

- L'instruction Lire(F, e) / Read(F, e) ;, en mode lecture, permet de lire l'enregistrement en cours et passe le pointeur à l'enregistrement suivant. À la fin du fichier, en aura le résultat de : *eof(F)* est *True*.

III.4.3. Quelques fonctions/procédures sur les fichiers

Dans cette section nous énumérons quelques fonctions et procédures prédéfinies pour qui sont utilisé pour manipuler les fichiers structurés (fichiers FILE OF). Quelques fonctions /

procédures sont utilisé dans les exemples précédent.

Le tableau suivant présente les principales fonctions / procédures sur les fichiers :

Fonction / Procédure	**Signification**
a) ASSIGN(Fichier, 'Nom_Fichier') ;	Procédure permettant d'associer le nom logique du fichier 'Fichier' au nom physique du fichier.
b) REWRITE (Fichier);	Procédure qui ouvre un fichier en mode écriture. Si le fichier n'existe pas, il sera créé, sinon, il sera écrasé (vider le fichier).
c) 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/écriture).
d) READ(Fichier, v1, v2, ..., vn) ;	Procédure de *lecture* des variables (tampons) v1, v2, ..., vn à partir de fichier (lecture par fichier et non par clavier).
e) WRITE(Fichier, v1, v2, ..., vn) ;	Procédure *d'écriture* des variables (tampons) v1, v2, ..., vn dans le fichier (écriture sur le fichier et non sur l'écran).
f) FILESIZE(Fichier) ;	Fonction qui retourne le nombre d'enregistrements dans le fichier.
g) FILEPOS(Fichier) :	Fonction qui retourne la position actuelle du pointeur de fichier (entre 0 et Filesize(File)).
h) SEEK(Fichier, indice) ;	Procédure qui permet de positionner le pointeur de fichier sur l'indice ($0 \leq$ indice $\leq$ FileSize(File)-1).
i) EOF(Fichier) ;	Fonction booléenne qui retourne TRUE si on est à la fin du fichier, ou bien FALSE dans le cas contraire.
j) CLOSE(Fichier) ;	Fermer le fichier après son utilisation.
k) RENAME(Fichier, 'nom_fichier2');	Procédure pour renommer le fichier logique *Fichier* au nom : '*nom_fichier2*' . Le fichier doit être fermé avant

	RENAME.
l) ERASE(Fichier);	Procédure pour supprimer le fichier logique *Fichier*. Le fichier doit être fermé avant ERASE.