

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

1.1 LES TABLEAUX A UNE DIMENSION

Définition : C'est un objet composé de plusieurs éléments de **même type** et dont chaque élément est repéré par un **indice** (ou **index**). La structure TABLEAU est une structure **STATIQUE**



Elément

Ce tableau est constitué de 9 éléments

Indice

Valeur

Type entier

Nbr est l'identificateur ou le nom de ce tableau

L'accès à un élément du tableau s'effectue en précisant le nom du tableau suivi de la valeur de l'indice entre crochets **Nbr[4]** représente le 4^{ème} élément de **Nbr**

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

Exemple : Soit un tableau COEF contenant les coefficients de 7 matières

COEF	6	4	5	3	2	2	1
	1	2	3	4	5	6	7

On peut écrire les actions suivantes

$A \leftarrow \text{COEF}[3]$	$B \leftarrow \text{COEF}[k]$	$C \leftarrow \text{COEF}[(\text{test}+k) \text{ MOD } i]$
$\text{COEF}[7] \leftarrow E$	$\text{COEF}[m] \leftarrow F$	$\text{COEF}[a*\text{resDIV2}] \leftarrow G$

- On constate à travers ces exemples qu'un élément du tableau est considéré comme une variable et que la valeur de l'indice peut être une constante, une variable ou une expression .
- Le nombre d'éléments du tableau constitue sa **taille**
- Le nombre d'indices qui permet de désigner un élément particulier est appelée **dimension** du tableau
- Le type de l'indice est un type scalaire (ordinal) et souvent intervalle

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

Pour définir un tableau il faut :

- connaître le nombre de ses éléments c'est à dire sa taille
- le type de chaque élément
- le type de l'indice
- le nom du tableau

La déclaration d'un tableau se fait en précisant le mot **TABLEAU**, suivi du **type de l'indice** entre crochets et **du type des éléments**.

- **Type**

Type_tableau = TABLEAU [type_de_l'indice] type_des_éléments

- **Variable**

Nom_du_tableau : Type_tableau

Nom_du_tableau : TABLEAU [type_de_l'indice] type_des_éléments

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

Exemples :

CONSTANTE max=7

TYPE

T1 =1..7

Taille = (grand, petit, moyen)

Tab1 =TABLEAU [T1] d'entiers

Cas = TABLEAU [booléen] de réels

Lettre = 'a'..'f'

Tab2 = TABLEAU (lettre] de caractères

VARIABLES

U : Tab1

V : TABLEAU [Taille] de réels

W : TABLEAU [1..Max] de booléens

X : TABLEAU [Booléen] d'entiers

Y : TABLEAU [1..7] d'entiers

Z : Cas

G : Tab2

U	5	4	-5	3	0	25	6
	1	2	3	4	5	6	7

V	5.0	-5.25	25.0
	grand	petit	moyen

W	VRAI	FAUX	VRAI	VRAI	FAUX	VRAI	VRAI
	1	2	3	4	5	6	7

X	25	-5
	VRAI	FAUX

Y	5	4	-5	3	0	25	6
	1	2	3	4	5	6	7

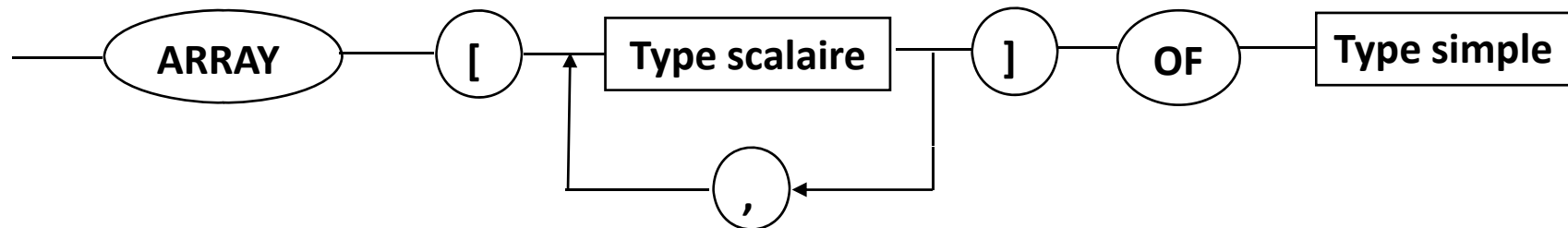
Z	2.5	- 0.5
	VRAI	FAUX

G	'A'	'4'	'!	'B'	'\$'	'h'
	'a'	'b'	'c'	'd'	'e'	'f'

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

Déclaration de tableaux en PASCAL :



Pour déclarer un tableau

- 1- on déclare le type de l'indice
- 2- on déclare le type de l'élément si ce dernier n'est pas un type standard
- 3- on déclare le type du tableau

Exemple

```
Const    Max =25
Type      Tind = 1..Max
          Ttab = TABLEAU [Tind] De ENTIER

VAR
          Ind : Tind
          TAB : Ttab
```

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

Exemple d'application :

Retrouver le nombre d'éléments égaux à une valeur donnée dans un tableau d'au maximum 100 éléments numériques entiers.

ANALYSE

- Soit un tableau T de 100 entiers
- Soit N une valeur
- Compter le nombre d'éléments égaux à N et l'écrire

ALGORITHME PRINCIPAL

CONST Max = 100

Type

Tind = 1..Max

Tab = TABLEAU [Tind] d'entiers

Variables

T : Tab

N: Entier

PROCEDURE LECT1D (VAR T : Tab)

FONCTION NBELEM (T: Tab ; N : ENTIER) : ENTIER

DEBUT

LECT1D (T)

Ecrire ('Donner la valeur recherchée : ')

Lire (N)

Ecrire ('Il y a ', NBELEM (T, N), ' éléments égaux à ', N)

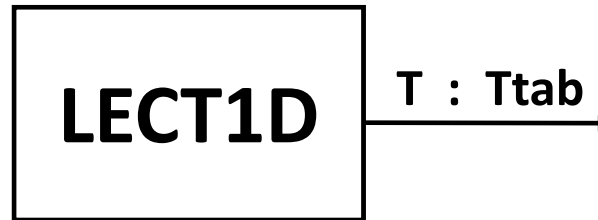
FIN

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

MODULE LECT1D

PROCEDURE



Rôle : Lit un tableau à une dimension dont les éléments sont des entiers

ANALYSE :

On lit les éléments un à un et on les range dans le tableau T.

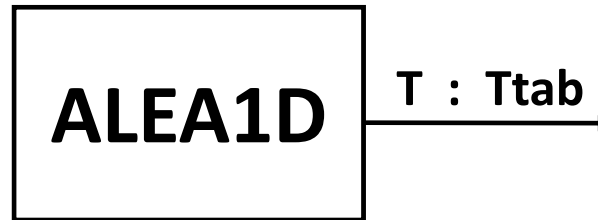
```
PROCEDURE LECT1DE (VAR T : Ttab )  
Variable i : Tind  
DEBUT  
    Pour i allant de 1 à Max faire  
        DPOUR  
            Ecrire ('T [, i,]=')  
            Lire (T[i])  
        FPOUR  
FIN
```

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

MODULE ALEA1D

PROCEDURE



Rôle : Remplit un tableau à une dimension aléatoirement

ANALYSE :

Génère des nombres aléatoires et les range dans le tableau T.

```
PROCEDURE LECT1DE (VAR T : Ttab )  
Variable i : Tind  
DEBUT  
    RANDOMIZE  
    Pour i allant de 1 à Max faire  
        DPOUR  
            T[i] ← RANDOM(100)  
        FPOUR  
FIN
```


CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

MODULE NBELEM

FONCTION



Rôle : Compte le nombre d'éléments de T égaux à N

ANALYSE :

On parcourt tous les éléments du tableau un par un et on les compare à N
Si égalité on compte

FONCTION NBELEM (T : Ttab; N : Entier) : ENTIER

Variable i : Tind

cpt : Entier

DEBUT

cpt ← 0

Pour i allant de 1 à Max faire

SI T[i]= N ALORS

cpt ← cpt +1

NBELEM ← cpt

FIN

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

1.2 LES ALGORITHMES DE TRI

Le but d'un tri est d'ordonner des éléments d'un tableau selon un certain **ordre** ou **critère** (de manière **Croissante** ou **décroissante**)

Principes généraux de quelques algorithmes de tri :

1- Tri par sélection :

Le plus petit élément du tableau est permuté avec le 1er élément du tableau, puis le plus petit élément du tableau restant est permuté avec le 2ième, etc...

-5	0	0	1	2	3	4	5	6	8
1	2	3	4	5	6	7	8	9	10

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

ANALYSE

Le plus petit élément du tableau est permuté avec le 1er élément du tableau, puis le plus petit élément du tableau restant est permuté avec le 2ème, etc...

PROC

```
PROCEDURE TRISEL ( T : Ttab)
VAR
  i : Tind
  Tmp , m : ENTIER
FONCTION Min ( T : Ttab ; I : Tind) : ENTIER
DEBUT
  POUR i ← 1 A Max-1 FAIRE
    DPOUR
      m ← Min (T,I)
      Si i <> m ALORS
        DSI
          tmp := t[i]
          t[i] := t[m]
          t[m] := tmp
        FSI
    FPOUR
  FIN
```

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

FONCTION



Rôle : Donne l'indice du plus petit élément du tableau commençant à t[i]

ANALYSE

On considère le 1^{er} élément du tableau comme étant le plus petit et on compare tous les autres éléments un à un à ce plus petit. Si infériorité l'élément en question devient le plus petit

ALGORITHME

```
FONCTION Min ( T : Ttab ; i : Tind ) : ENTIER
Var j : Tind
    m : entier
DEBUT
    m ← i
    POUR j ← i A Max FAIRE
        DPOUR
            Si T[j] < T[m] ALORS
                DSI
                    m ← j
            FSI
        FPOUR
    Min ← m
FIN
```

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

Tri par transposition :

Deux éléments qui se suivent sont comparés puis permutés si le 2ème élément est plus petit que le premier, puis un retour en arrière est effectué afin de vérifier si l'ordre n'a pas été modifié, auquel cas on le rétablit.

-5	0	3	4	5	6	0	8	2
-5	0	0	2	3	4	5	6	8

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

Tri "bulles "

Les éléments mal classés remontent dans le tableau comme des bulles à la surface d'un liquide. La "remontée" des éléments s'effectue avec des permutations si $\text{élément1} > \text{élément2}$. Il est évident que plusieurs passages sur l'ensemble des éléments sont nécessaires.

5	4	-5	3	0	8	6	0	2
---	---	----	---	---	---	---	---	---

-5	0	0	2	3	4	5	6	8
----	---	---	---	---	---	---	---	---

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

Tri par comptage avec 2 tables.

Il se fait en 2 étapes.

1^{ère} étape : Pour chaque élément on compte le nombre d'éléments qui lui sont inférieurs et on range le résultat trouvé dans un tableau de comptage (dans des cases de même indice bien entendu) de taille identique à celle du tableau à trier.

Correction de la table de comptage pour traiter les éléments identiques

2^{ème} étape : l'élément dont le compteur est égal à zéro est permuté avec le premier élément du tableau, l'élément dont le compteur est égal à 1 est permuté avec le 2^{ième} élément du tableau, etc. ... (Il ne faudra pas oublier de permuter aussi les éléments de la zone de comptage en même temps que ceux du tableau à trier).

Table de données	8	4	-5	3	0	9	5	0	2
Table de comptage	7	5	0	4	1	8	6	2	3

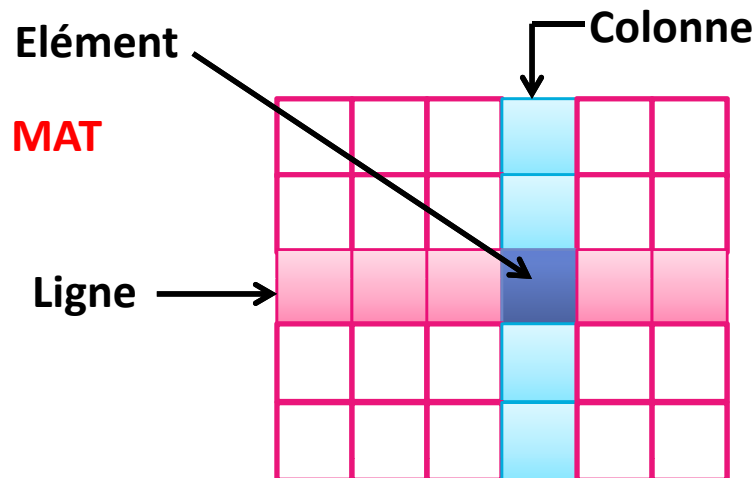
CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

1.1 LES TABLEAUX A 2 DIMENSIONS

Définition : C'est un tableau à une dimension dont les éléments sont des tableaux à une dimension

C'est un objet composé de plusieurs éléments de même type (lignes) ou (colonnes) et dont chaque élément est à son tour composé de plusieurs éléments de même type.



L'accès à un élément du tableau s'effectue en précisant le nom de l'objet suivi, entre crochets, de l'indice de ligne et de l'indice de colonne séparés par une virgule.

MAT [3,4] désigne l'élément pointé

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

Déclaration de tableaux à deux dimensions

CONST

MaxL = 5

MaxC = 6

TYPE

TindL : 1..MaxC

TindC : 1..MaxL

TLig = TABLEAU [TindL] DE ENTIER

TCol = TABLEAU [TindC] DE ENTIER

TMat = TABLEAU [TindL] DE Tcol

TMat2 = TABLEAU [TindC] DE TLig

TMat3 = TABLEAU [TindL , TindC] DE ENTIER

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

Déclaration de tableaux à deux dimensions

VAR

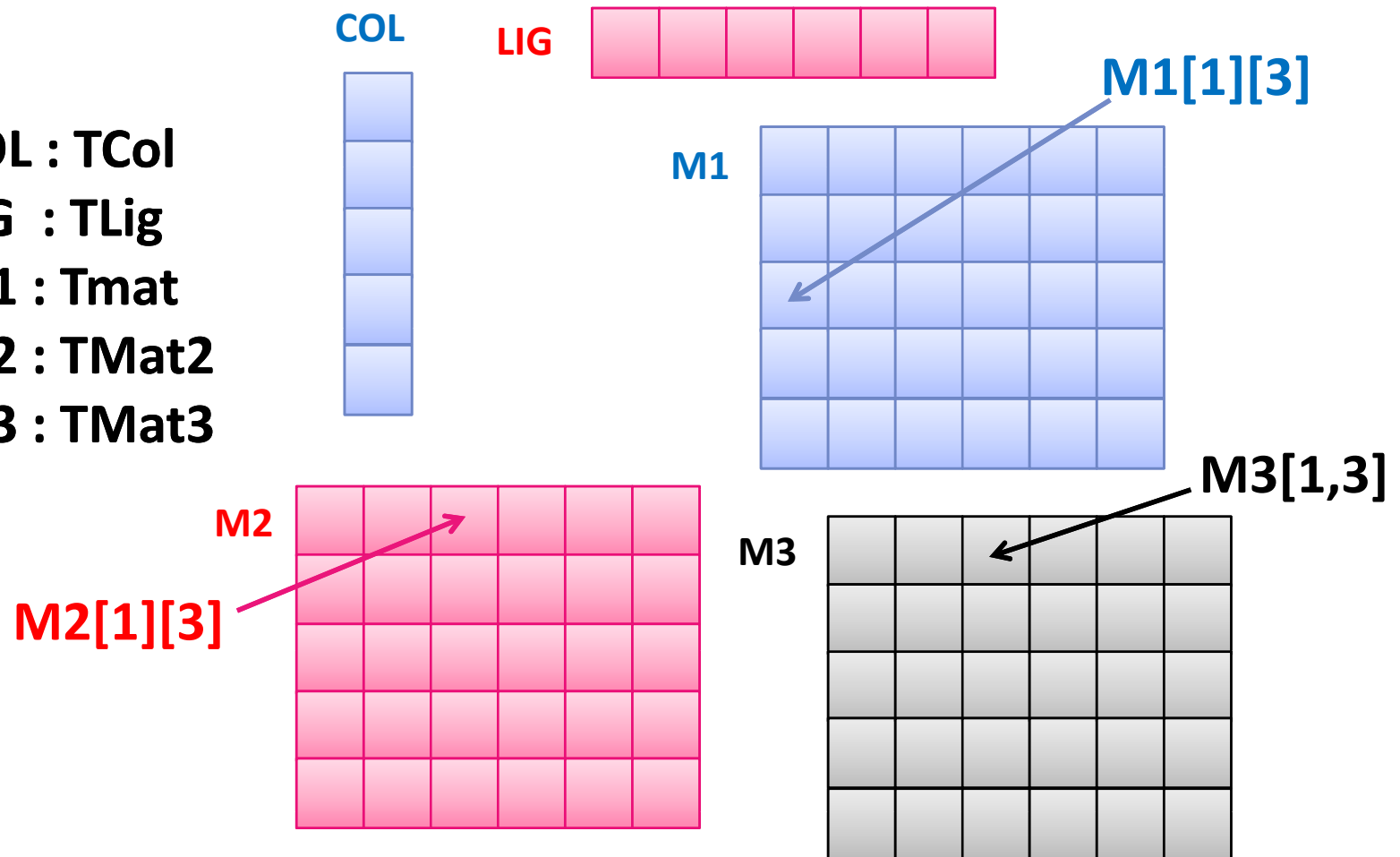
COL : TCol

LIG : TLig

M1 : Tmat

M2 : TMat2

M3 : TMat3



CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

Exemples :

TYPE

Matière = (Algo, Math, Strm, Ieco)

Etudiant = TABLEAU [Matière] de réel

Promo1 = TABLEAU [1..350] de Etudiant

Promo2 = TABLEAU [1..350, Matière] de réel

VARIABLES

P : Promo1

Q : Promo2

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

Exemple d'application :

Soit un tableau à deux dimensions. Ranger les cumuls des éléments d'une même colonne dans un tableau à une dimension et les éléments d'une même ligne dans un autre

SLIG

MAT

4	-5	3	6	0	1	9
3	2	5	-1	2	4	15
1	4	-3	0	3	2	7
2	4	1	1	9	0	17
1	3	0	4	5	7	20

SCOL

11	8	6	10	20	14
----	---	---	----	----	----

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

ANALYSE

- Soit un tableau à 2 dimensions
MAT
- Pour chaque ligne calculer la somme des éléments et la ranger dans **SLIG[I]**
- Pour chaque colonne calculer la somme des éléments et la ranger dans **SCOL[I]**
- Afficher **MAT** , **SLIG** et **SCOL**

ALGORITHME SomLigCol

CONST

MaxL= 6

MaxC = 5

TYPE

TindL : 1..MaxC

TindC : 1..MaxL

TMat = TABLEAU [TindL , TindC] DE ENTIER

TLig = TABLEAU [TindL] DE ENTIER

TCol = TABLEAU [TindC] DE ENTIER

VAR

MAT : TMat

SCOL : TLig

SLIG : TCol

IndL : TindL

IndC : TindC

PROCEDURE InitMat (VAR M : TMat)

PROCEDURE SomLig (M : TMat ; VAR S : Tcol)

PROCEDURE SomCol (M : TMat ; VAR S : TLig)

PROCEDURE Afficher (M : TMat ; L : Tcol ; C : TLig)

BEGIN

InitMat (MAT)

SomLig (MAT,SLIG)

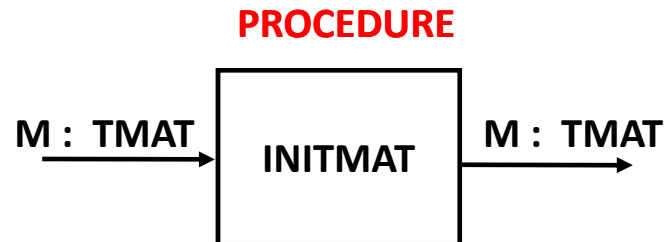
SomCol (MAT,SCOL)

Afficher (MAT,SLIG,SCOL)

END.

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX



Rôle : Initialise le tableau à 2 dimensions MAT

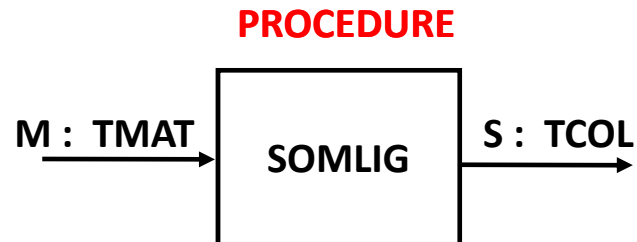
ANALYSE

Pour chaque élément c.-à-d. pour chaque ligne et pour chaque colonne de M on lit ou bien on génère un entier que l'on affecte à l'élément M[L,C]

```
PROCEDURE INITMAT ( VAR M : TMAT)
Var L : TindL
    C : TindC
DEBUT
    RANDOMIZE
    POUR L ← 1 A MaxL FAIRE
        POUR C ← 1 A MaxLC FAIRE
            DPOUR
                M[ L , C ] ← RANDOM(99)
            FPOUR
FIN
```

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX



Rôle : Calcule pour chaque ligne la somme des éléments et la range dans S[L]

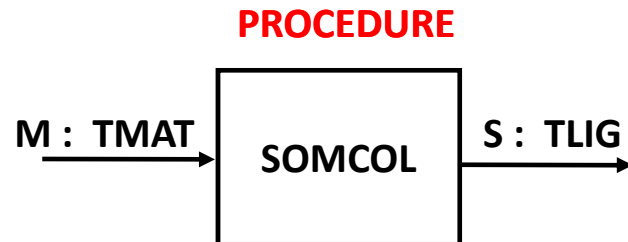
ANALYSE

Pour chaque ligne L de M on cumule les éléments M[L,C] dans S[L] initialement = 0
 $S[L] \leftarrow S[L] + M[L,C]$

```
PROCEDURE SomLig (M : TMAT ; VAR S : Tcol)
Var L : TindL
    C : TindC
DEBUT
    POUR L ← 1 A MaxL FAIRE
        DPOUR
            S[L] ← 0
            POUR C ← 1 A MaxC FAIRE
                S[L] ← S[L] + M[L,C]
            FPOUR
        FIN
    FIN
```

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX



Rôle : Calcule pour chaque colonne la somme des éléments et la range dans S[C]

ANALYSE

Pour chaque ligne C de M on cumule les éléments M[L,C] dans S[C] initialement = 0
 $S[C] \leftarrow S[C] + M[L,C]$

```
PROCEDURE SomCol (M : TMAT ; VAR S : TLig)
Var L : TindL
    C : TindC
DEBUT
    POUR C ← 1 A MaxC FAIRE
        DPOUR
            S[C] ← 0
            POUR L ← 1 A MaxL FAIRE
                S[C] ← S[C] + M[L,C]
            FPOUR
        FIN
    FIN
```


CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

```
PROCEDURE Somligcol ( MAT : Tmat ; Var SLIG : Tlig ; Var SCOL : Tcol)
Var
    L : Tindl
    C: Tindc
DEBUT
    POUR C  $\leftarrow$  1 to maxC FAIRE
        SCOL[C]  $\leftarrow$  0
        POUR L  $\leftarrow$  1 to maxL FAIRE
            DEBUT
                SLIG [L]  $\leftarrow$  0
                POUR C  $\leftarrow$  1 to maxC FAIRE
                    DEBUT
                        SLIG[L] := SLIG[L] + MAT [L,C]
                        SCOL[C] := SCOL[C] + MAT [L,C]
                    FIN
                FIN
            FIN
        FIN
    FIN
FIN
```

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

EXEMPLE

FUSION DE N (MAX 8) TABLEAUX TRIES DANS UN SEUL TABLEAU RES
SOIENT N TABLEAUX A UNE DIMENSION CONTENANT CHACUN UN CERTAIN
NOMBRE D'ENTIERES (AU MAXIMUM 10). CHACUN DES TABLEAUX EST TRIE PAR
ORDRE CROISSANT
ON VOUDRAIT LES FUSIONNER DANS UN SEUL TABLEAU RES A UNE DIMENSION
TRIE SANS UTILISER DE PROGRAMME DE TRI POUR RES

T1

-8	-5	-5	0	9
----	----	----	---	---

T2

-3	0	5	15	18	25
----	---	---	----	----	----

T3

8	14
---	----

T4

1	4	8	11
---	---	---	----

RES

-8	-5	-5	-3	0	0	1	5	5	8	8	9	11	15	14	18	25
----	----	----	----	---	---	---	---	---	---	---	---	----	----	----	----	----

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

ANALYSE ET STRUCTURE DE DONNEES

STRUCTURE DE DONNEES

ANALYSE

On parcourt tous les tableaux élément par élément à partir du premier et à chaque fois le plus petit élément de tous les tableaux est transféré dans RES. On n'avance que dans le tableau qui a donné le plus petit

	Données										Taille	EltCour	Indice
1	-8	-5	-5	0	9						5	9	5
2	-3	0	5	15	18	25					6	5	3
3	8	14									2	8	1
4	1	4	8	11							4	8	3
5											0		
6											0		
7											0		
8											0		

RES

								...									
--	--	--	--	--	--	--	--	-----	--	--	--	--	--	--	--	--	--

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

CONST

MaxElt = 10
MaxTab = 8
GV = 32767
MaxRes = 80

TYPE

Tres = 1..MaxRes
Tind = 1..MaxElt
Tjnd = 1..MaxTab
TabRes = TABLEAU [Tres] DE ENTIER
TTab = TABLEAU [Tind] DE ENTIER
TDon = TABLEAU [Tjnd] DE TTab
Ttai= TABLEAU [tjnd] DE ENTIER

VAR

Don : TDon
N : Tjnd
Taille : Ttai
Res : TabRes
i : Tind ;
TR : Tres

Declare.txt

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

ANALYSE

Soit N le nombre de tableaux à fusionner
Pour chaque tableau
- Lire le Nombre d'éléments (Taille[i])
- lire les éléments et les ranger dans Don
Fusionner les tableaux dans RES

ALGORITHME FUSIONTAB

```
{ $i declare.txt}
```

```
PROCEDURE LECT1D (N : Tjnd ; VAR T : Ttab)
```

```
PROCEDURE FUSION ( D : Tdon ; T : Ttai ; N : Tjnd;  
Tr : Tres ; VAR R : TabRes ;)
```

```
DEBUT
```

```
    ECRIRE ('Donnez le nbr de Tableaux à fusionner : ')
```

```
    LIRE (n) ;
```

```
    TR ← 0
```

```
    POUR I ← 1 A N FAIRE
```

```
        DPOUR
```

```
            ECRIRE ('Donnez le nbr d"éléments du tableau : ',I)
```

```
            LIRE (Taille[i]) ;
```

```
            TR ← TR + Taille[i]
```

```
            LECT1D (Taille[i],Don [i]);
```

```
        FPOUR
```

```
        FUSION (Don,Taille,N,TR,Res)
```

```
FIN
```

ALGORITHME

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

MODULE LECT1D

PROCEDURE



Rôle : Lit un tableau à une dimension dont les éléments sont des entiers

ANALYSE :

On lit les N éléments un à un et on les range dans le tableau T.

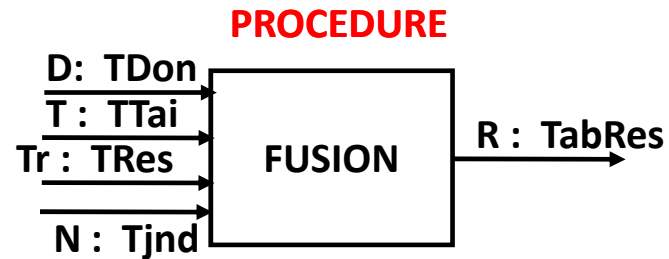
```
PROCEDURE LECT1DE (N: ENTIER ;VAR T : Ttab )
Variable i : Tind
DEBUT
    Pour i allant de 1 à N faire
        DPOUR
            Ecrire ('T [, i,']=')
            Lire (T[i])
        FPOUR
    FIN
```

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

Module FUSION

ALGORITHME



Rôle : Fusionne dans R les N tableaux à une dimension contenus dans D et de tailles contenues dans T

ANALYSE

-Initialisation de EltCour et Indice

- $i \leftarrow \text{IndMin}(\text{EltCour})$
- $\text{Res}[k] \leftarrow \text{EltCour}[i]$
- $\text{indice}[i] \leftarrow \text{indice}[i] + 1$
- Si $\text{Indice}[i] \leq T[i]$ alors
 - $\text{EltCour}[i] \leftarrow D[i, \text{Indice}[i]]$
- Sinon
 - $\text{EltCour}[i] \leftarrow \text{GV}$
- $k \leftarrow k + 1$

Ce processus est répété jusqu'à épuisement de tous les éléments

```

PROCEDURE FUSION ( D : Tdon ; T : Ttai ; N : Tjnd ;
                    VAR R : TabRes ; VAR Tr : Tres )

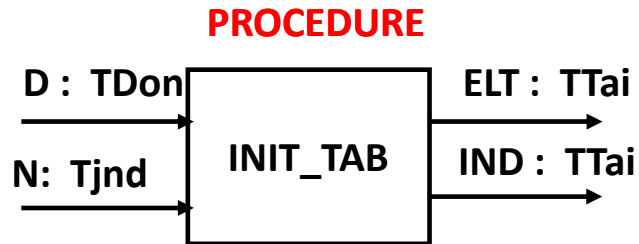
VAR
    k ,S: Tres
    Indice , EltCour : Ttai
    i : Tind ;

FONCTION IndMin( T : Ttai ; N : Tjnd ) : Tjnd
PROCEDURE Init_Tab ( D: Tdon ; N : Tjnd ; VAR E , IND : Ttai)
DEBUT
    Init_Tab ( D,N,EltCour, Indice)
    REPETER
         $i \leftarrow \text{IndMin}(\text{EltCour}, N)$ 
         $\text{Res}[k] \leftarrow \text{EltCour}[i]$ 
         $\text{indice}[i] \leftarrow \text{indice}[i] + 1$ 
        Si  $\text{Indice}[i] \leq T[i]$  alors
             $\text{EltCour}[i] \leftarrow D[i, \text{Indice}[i]]$ 
        Sinon
             $\text{EltCour}[i] \leftarrow \text{GV}$ 
         $k \leftarrow k + 1$ 
    JUSQU'À  $k > \text{Tr}$ 
FIN
  
```

CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

Module INIT_TAB



Rôle : Initialise la table des éléments courants E
et celle des indices correspondants I

ALGORITHME

ANALYSE

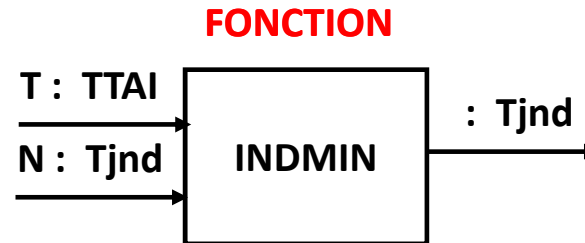
```
POUR I ← 1 à N FAIRE  
  ELT [i] ← D [i,1]  
  IND[i] ← 1
```

```
PROCEDURE Init_Tab ( D: Tdon ; N : Tjnd ; VAR E , IND : Ttai)  
VAR i : Tjnd  
DEBUT  
  POUR I ← 1 A N FAIRE  
    DPOUR  
      E [i] ← D [i,1]  
      IND[i] ← 1  
    FPOUR  
FIN
```


CHAPITRE 5 : LES OBJETS STRUCTURES

1- LES TABLEAUX

Module INDMIN



Rôle : Donne l'indice du plus petit élément contenu dans T

ANALYSE

On considère le 1^{er} élément comme étant le plus petit ensuite on compare tous les autres à ce plus petit. Si infériorité l'élément en question devient le plus petit

ALGORITHME

```
FONCTION INDMIN ( T : Ttai ; N : Tjnd) : Tjnd
VAR i : Tjnd
    M: entier
DEBUT
    M ← 1
    POUR i ← 2 A N FAIRE
        SI T [i] < T[M] ALORS
            M ← i
    INDMIN ← M
FIN
```

CHAPITRE 5 : LES OBJETS STRUCTURES

2- LES CHAINES DE CARACTERES

3.1 LES CONSTANTES CHAINES DE CARACTERES

C'est une suite de caractères entre apostrophe

Exemples :

- ' Donnez un nombre entier compris entre 1 et 6: '
- ' Ce programme donne le Nième terme de la suite de FIBO '
- ' L''equation n''admet pas de solutions '

3.2 LES VARIABLES CHAINES DE CARACTERES

C'est un TABLEAU à une dimensions de caractères

Exemple :

CITATION

I	L		V	A	U	T		M	I	E	U	X		A	V	O	I	R		P	E	U	R		Q	U	E		D	'	A	V	O	I	R		H	O	N	T	E	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43

L'accès à un élément se fait en citant le nom de la chaîne suivi de l'indice entre crochet **CITATION [21]**

CHAPITRE 5 : LES OBJETS STRUCTURES

2- LES CHAINES DE CARACTERES

3.2 LES VARIABLES CHAINES DE CARACTERES

- La longueur maximale d'une chaîne de caractères est de 255
- Une position supplémentaire contient la longueur de la chaîne
- La dimension d'une chaîne est sa longueur c.-à-d. le nombre de caractères qui la compose
- Par défaut la longueur dimension d'une chaîne est de 255

•Exemples

TYPE

Phrase = Chaine

Mot = Chaine [25]

VAR

P : Phrase

M : Mot

- Le type Chaine (STRING) est considéré en même temps comme un type simple et un type structuré.

Exemples

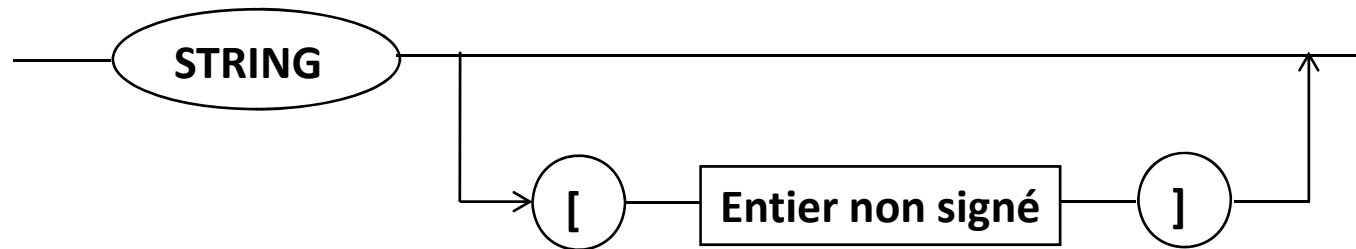
LIRE (P)

SI P[I] = 'A' ALORS Cum \leftarrow Cum + 1

CHAPITRE 5 : LES OBJETS STRUCTURES

2- LES CHAINES DE CARACTERES

Déclaration de Chaines en PASCAL :



NOTA : Il existe une multitude de fonctions standards qui permettent de manipuler les chaînes de caractères

CHAPITRE 5 : LES OBJETS STRUCTURES

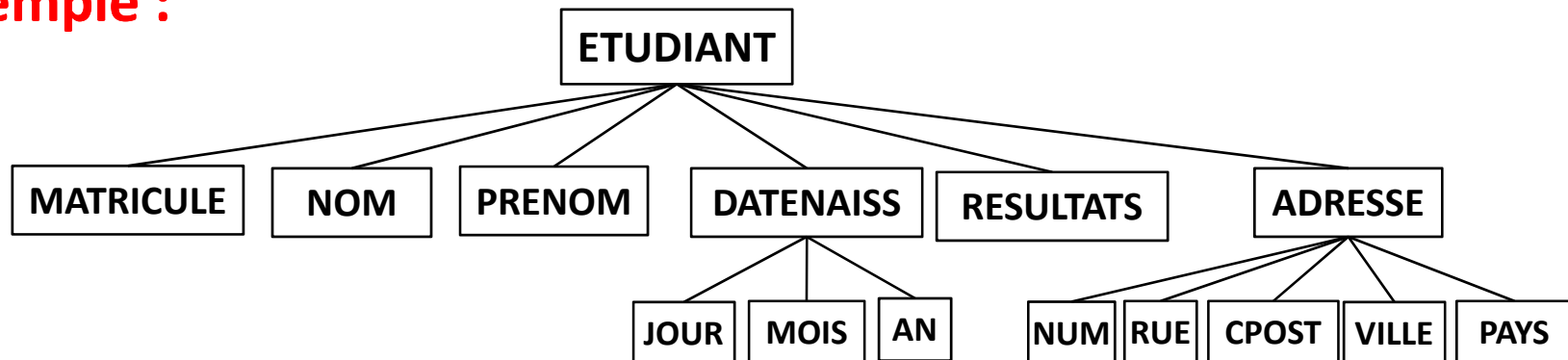
3- LES ENREGISTREMENTS

Définition : Un enregistrement est un ensemble d'éléments pouvant être de types différents.

Les éléments sont appelés **Champs** et peuvent être à leur tour des structures ou des éléments simples

Un enregistrement est utilisé pour regrouper dans une même structure un ensemble d'informations caractéristiques d'un objet déterminé.

Exemple :



CHAPITRE 5 : LES OBJETS STRUCTURES

3- LES ENREGISTREMENTS

La déclaration d'un enregistrement se fait en précisant le mot **ENREGISTREMENT** suivi de la description des champs et terminé par le mot **FIN**

EXEMPLE :

TYPE

TRESULT = TABLEAU [1..6] de REEL

TADR = ENREGISTREMENT

NUM : ENTIER

RUE ,VILLE , PAYS : CHAINE [30]

CPOST : ENTIER

FIN

ResElev = **ENREGISTREMENT**

MATRICULE : ENTIER

NOM,PRENOM : CHAINE [25]

RESULTATS : TRESULT

DATNAISS : **ENREGISTREMENT**

JOUR : 1..31

MOIS : 1..12

AN : ENTIER

FIN

ADRESSE : TADR

FIN

VAR

ETUDIANT : RESELEV

PROMO : TABLEAU [1..300] DE RESELEV

COEF : TABLEAU [1..6] DE ENTIER

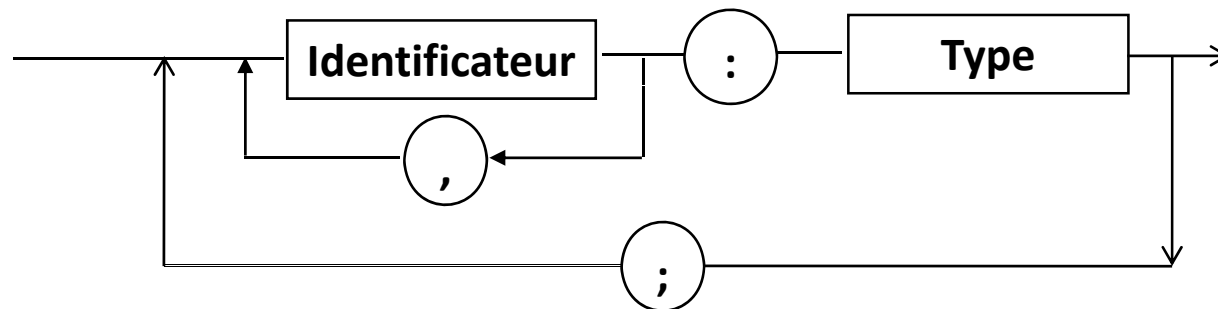
CHAPITRE 5 : LES OBJETS STRUCTURES

3- LES ENREGISTREMENTS

Déclaration d'enregistrements en PASCAL :



Liste de champs



CHAPITRE 5 : LES OBJETS STRUCTURES

3- LES ENREGISTREMENTS

L'accès à un champs se fait en précisant le nom de la variable de type enregistrement suivi d'un point suivi du nom du champs

Variable.Champs

Exemples :

ETUDIANT.NOM ← 'MEDJAOU'

ETUDIANT.DATNAISS.JOUR ← 25

SI ETUDIANT.RESULTATS[5] >= 16

L'INSTRUCTION AVEC.. FAIRE (WITH .. DO)

L'instruction **AVEC** permet d'abrégé (de mettre en facteur) la chaîne de références conduisant à désigner plusieurs champs

Exemple:

AVEC ETUDIANT FAIRE

NOM ← 'BEN ALI'

RESULTATS[1] ← 12

ECRIRE (DATNAISS.JOUR)

FIN

WITH ETUDIANT DO

NOM := 'BEN ALI' ;

RESULTATS[1] := 12 ;

Writeln (DATNAISS.JOUR) ;

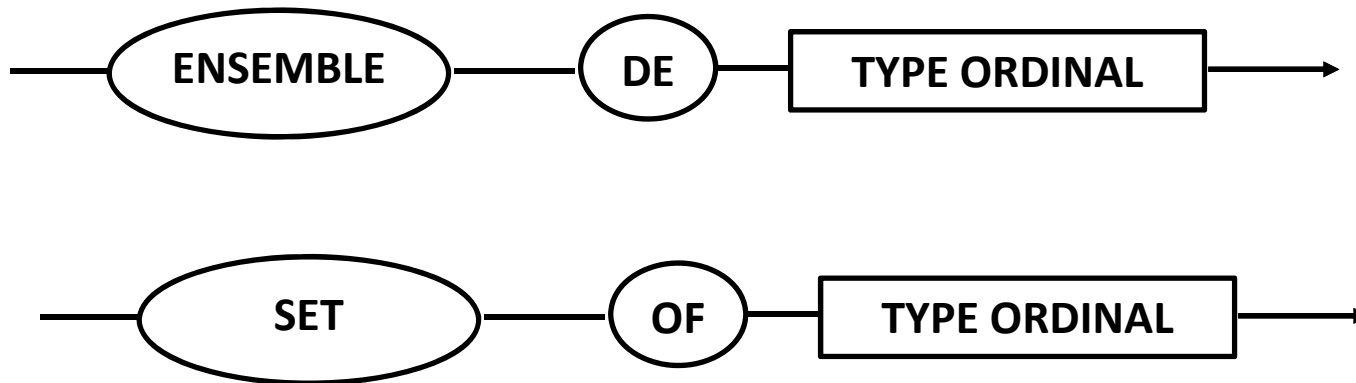
END;

Etudiant.Datnaiss ← DatJour

CHAPITRE 5 : LES OBJETS STRUCTURES

4- LES ENSEMBLES

Définition : Un ensemble est une variable qui contient un nombre **fini** d'éléments de même type. Ce type doit être de type **ordinal** c'est-à-dire qu'il ne doit être ni de type réel, ni de type chaîne de caractères, ni de type enregistrement. Ce type doit être **énuméré** et ne doit pas excéder **256** combinaisons possibles. La déclaration d'une telle variable se fait par l'utilisation de la syntaxe Ensemble de ou Set Of.



La borne supérieure – la borne inférieure du type de base ordinal doit être comprise entre 0 et 255.

CHAPITRE 5 : LES OBJETS STRUCTURES

4- LES ENSEMBLES

VARIABLES ENSEMBLE

Contrairement aux autres structures de données, nous ne pouvons pas accéder aux éléments d'une variable de type ensemble. Les ensembles se manipulent dans leur totalité.

Type

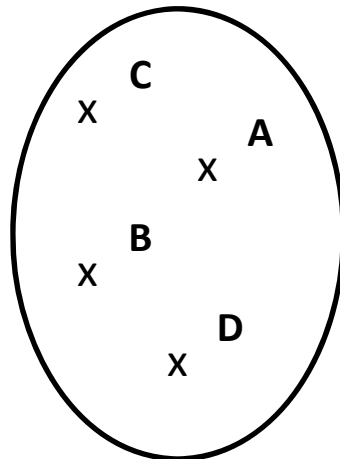
Vingt = Ensemble de 20..40

LetHexa = Ensemble de 'A' .. 'F'

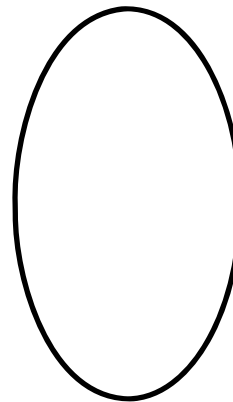
Var

E : Vingt

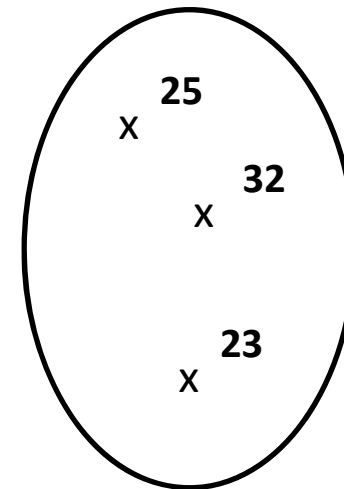
L : LetHexa



$L \leftarrow ['A'..'D']$



E vide
 $E \leftarrow []$



E contenant les valeurs
23, 25 et 32
 $E \leftarrow [23, 25, 32]$

CHAPITRE 5 : LES OBJETS STRUCTURES

4- LES ENSEMBLES

Exemples

Type

Jour = (Sam, Dim, Lun, Mar, Mer, Jeu, Ven)
CharSet = Ensemble de Char
Chiffre = Ensemble de 0..9
Jours = Ensemble de Jour

Var

J : Jours
C : CharSet
I, K : 0..9
N : Chiffre
Séparateur : CharSet

{ Constructeurs d'ensemble }

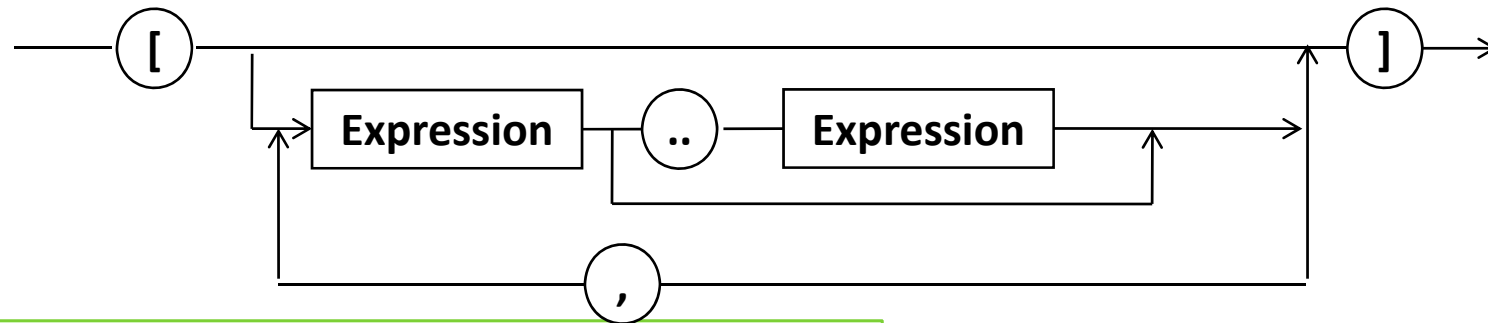
$C \leftarrow ['0'..'9', 'A'..'Z', 'a'..'z', '_']$
 $N \leftarrow [1, 5, I + 1 .. J - 1]$
 $J \leftarrow [\text{Mon}..\text{Fri}]$
 $\text{Séparateur} \leftarrow [';', ',', '.', ':', '?', '!', ';']$

CHAPITRE 5 : LES OBJETS STRUCTURES

4- LES ENSEMBLES

EXPRESSION D'ENSEMBLES

On ne peut pas définir de constantes de type ensemble, mais on peut définir des expressions d'ensembles et des ensembles constants



Examples

type

Chiffres = Ensemble de 0..9

Lettres = Ensemble de 'A'..'Z'

Const

ChPair : Chiffre = [0, 2, 4, 6, 8]

Voyelles : Lettres = ['A', 'E', 'I', 'O', 'U', 'Y']

ChHexa: Ensemble de '0'..'z' = ['0'..'9', 'A'..'F', 'a'...'f']

Sep : Ensemble de Char = [' ', ',', '.', ':', '?', '!', ';']

L'ensemble constant []

représente l'ensemble vide

Si a, b, c sont des constantes de

même type, $[a,b,c]$ représente l'ensemble constant comprenant les 3 valeurs a,b,c .

**[a..b] est l'ensemble constant
comprenant les valeurs a,succ(a),
..pred(b) et b**

CHAPITRE 5 : LES OBJETS STRUCTURES

4- LES ENSEMBLES

OPERATEURS SUR LES ENSEMBLES

Les opérations suivantes ont pour opérandes deux ensembles du même type T

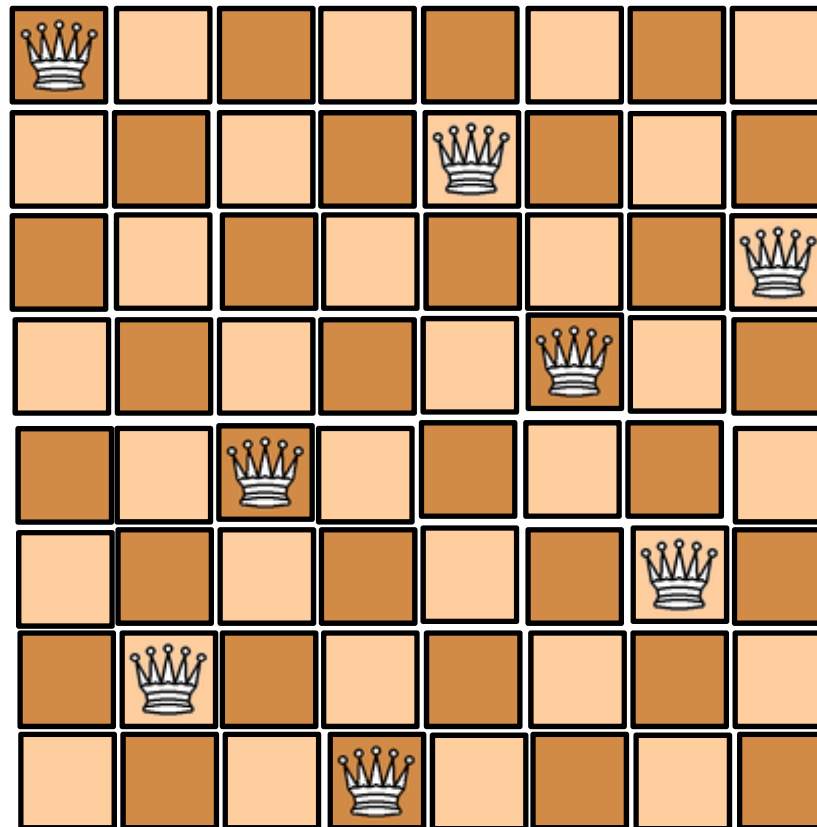
OPERATION	OPERATEUR	EXEMPLE	TYPE DU RESULTAT
Affectation	\leftarrow	$S \leftarrow [a,b,c]$	Ensemble de T
Union	+	$S + [e,f]$	«
Intersection	*	$S * P$	«
Différence	-	$S - P$	«
Egalité	=	$S = P$	Booléen
Inégalité	\neq	$S \neq P$	Booléen
Inclusion	\leq et \geq	$(S \leq P)$ ou $(P \geq S)$	Booléen

L'opérateur **DANS** (**IN**) a pour opérande gauche une valeur de type compatible avec T et comme opérande droite une expression de type Ensemble de T. Il rend une valeur **VRAI** si l'opérande gauche appartient à l'ensemble opérande droite et **FAUX** sinon

Si C est de type Char on peut écrire : **Si C DANS Sep ALORS....**

Problème des huit dames

Le but du "problème" des huit dames, est de placer huit dames d'un jeu d'échecs sur un échiquier de 8×8 cases sans que les dames ne puissent se menacer mutuellement, conformément aux règles du jeu d'échecs . Par conséquent, deux dames ne devraient jamais partager la même rangée, colonne, ou diagonale.



Problème des huit dames

Analyse :

Si nous arrivons à la huitième colonne, et que nous y trouvons une position sans conflit, nous aurons une solution répondant au problème. Mais il s'agit, bien sûr, de les trouver toutes. Pour cela, nous avons utilisé un algorithme dit de retour en arrière appelé backtracking

Cet algorithme consiste, lorsqu'une solution est trouvée (ou pas trouvée), pour une colonne donnée, à une étape quelconque de l'algorithme, à revenir à la colonne précédente pour essayer d'y trouver une autre solution sans conflit, et à repartir ensuite, en avançant jusqu'à la dernière colonne, où jusqu'à une impossibilité. Lorsque l'opération a été répétée jusqu'à la colonne 1, c'est qu'il n'y a plus de solution dans cette colonne. Il faut alors changer la place de la dame en colonne 1, et répéter le processus jusqu'à ce que la dame de la colonne 1 ait été placée sur les huit lignes.

Problème des huit dames

L'algorithme peut alors être résumé aux étapes suivantes :

Placer la première reine sur la colonne 1 ($C = 1$)

Si $C > 8$, imprimer la solution, et aller en 5.

Placer une reine sur la ligne 1 de la colonne C ($L = 1$).

Si une position est sans conflit, passer à la colonne suivante ($C = C + 1$), et aller en 2.

Si la reine de la colonne C est sur la ligne 8 ($L = 8$) passer à la colonne précédente $C = C - 1$

Si $C = 1$ et $L = 8$: terminé. Sinon faire $L = L + 1$.

Si $L > 8$ aller en 5. Sinon aller en 4.

Problème des huit dames

Nous utilisons un tableau COL pour représenter la position d'une dame dans une colonne. Ainsi, COL [i] représente la *ième* colonne, et sa valeur la ligne sur laquelle se trouve la dame, dans cette colonne. (Si COL[i] = 3 la reine de la *ième* colonne se trouve sur la troisième ligne). D'autre part, pour tester l'absence de conflit sur une diagonale, il suffit de vérifier que les dames ne sont pas sur des droites de pente **+ 1 ou - 1**.

Avec deux dames de position x1, y1 et x2, y2, il suffit de tester que :

$$|y1 - y2| - |x1 - x2| \neq 1 \quad \text{Soit} \quad |y1 - y2| / |x1 - x2| \neq 0.$$

Ceci est effectué par la fonction CONFLIT.

CHAPITRE 6 : LES FICHIERS

INTRODUCTION

Nous avons vu dans les chapitres précédents comment construire des programmes et des modules qui utilisent des objets de divers types. Nous avons aussi vu l'énorme intérêt de leur sauvegarde sur un support magnétique (disquettes, disques durs) même s'ils sont inachevés ce qui nous évite de les réécrire à chaque fois que nous les utilisons.

Toute information transite par la **mémoire principale** pour être traitée. Elle est perdue aussitôt l'ordinateur éteint. Pour ne pas perdre l'information il faut la stocker sur **mémoires auxiliaires ou secondaires**

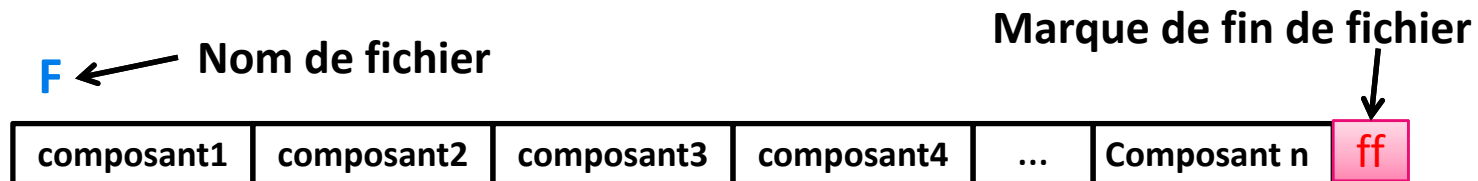
Les informations que l'on veut stocker peuvent prendre deux formes :

- données sous forme de composants (enregistrements ou articles)
- texte

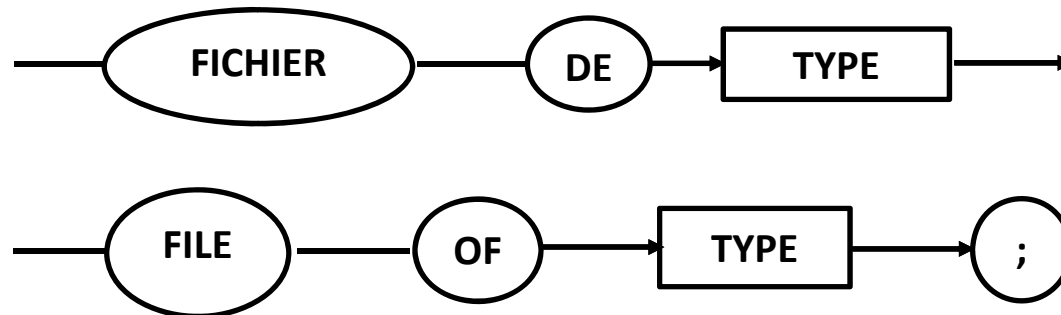
CHAPITRE 6 : LES FICHIERS

1- FICHIERS DE DONNEES

C'est un ensemble de composants ou d'enregistrements (d'articles) traitant du même sujet et regroupés sous un même nom.



La déclaration d'un fichier de données se fait en précisant le mot FICHIER suivi du type des composants. Ce dernier pouvant avoir n'importe quel type (élémentaire ou structuré) sauf fichier .



CHAPITRE 6 : LES FICHIERS

1- FICHIERS DE DONNEES

Exemple :

Type

T= TABLEAU [1..20] d'entiers

F = FICHIER de T

Article = ENREGISTREMENT

Nom : chaîne [25]

Notel : entier

FIN

Var

F1 : FICHIER d'entiers (* chaque composant est un entier *)

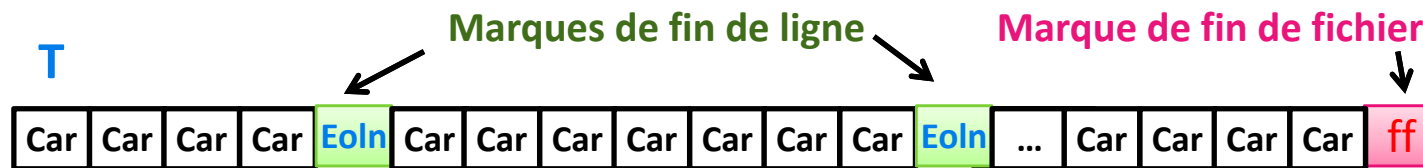
Enr : F (* chaque composant est un tableau *)

Répertoire : FICHIER de Article (* chaque composant est un enregistrement *)

CHAPITRE 6 : LES FICHIERS

2- FICHIERS TEXTE

C'est une suite de caractères du jeu ASCII découpée sous forme de lignes, pouvant être de longueurs différentes. (Chaque ligne se terminant par une marque de fin de ligne appelée **EOLN** (End Of Line) correspondant au caractère "retour chariot ou Return"



la déclaration d'un fichier texte se fait simplement en précisant le type TEXTE (prédéfini)

Exemple :

Variable Livre : TEXTE

En Pascal

Var Livre : TEXT ;

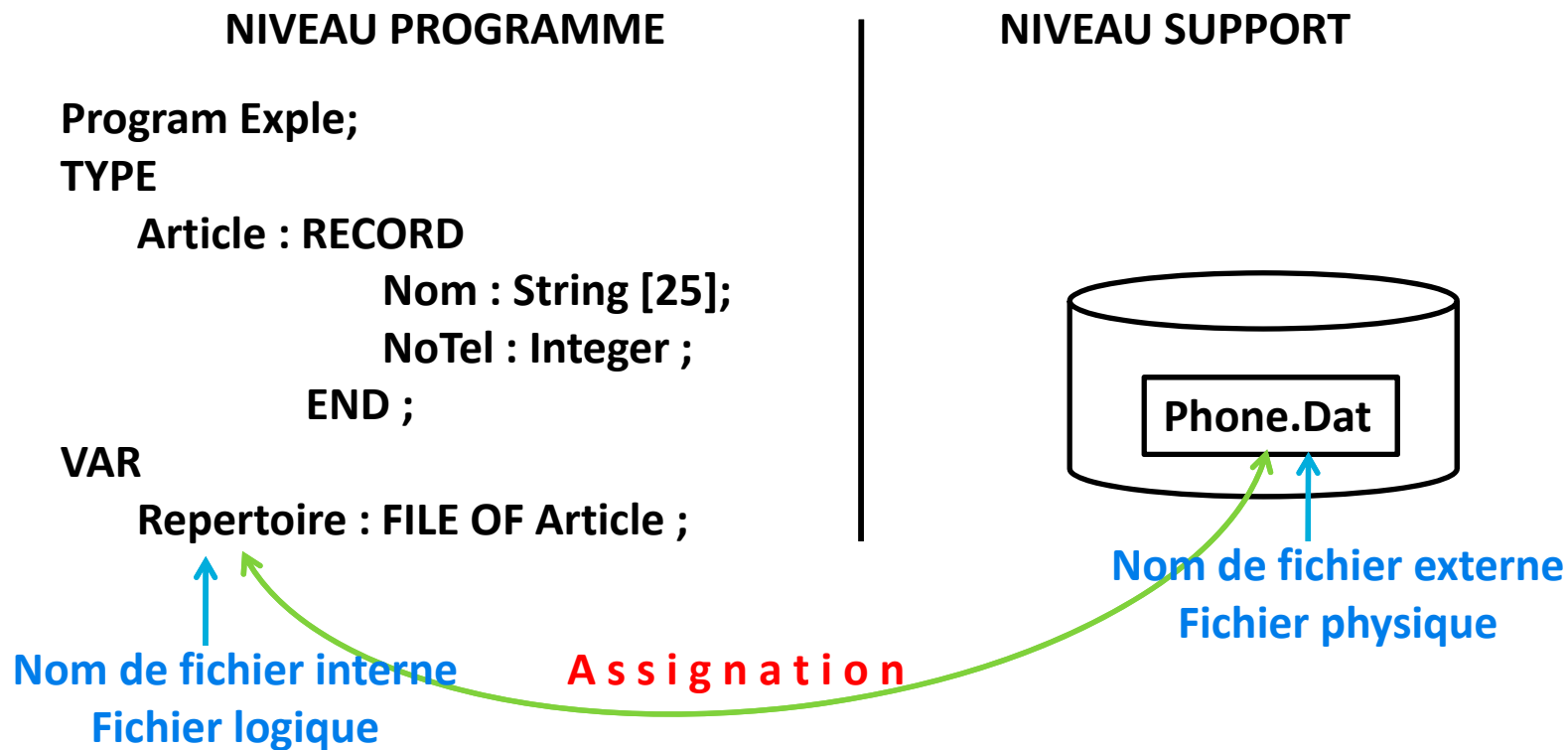
CHAPITRE 6 : LES FICHIERS

3- OPERATIONS FONDAMENTALES SUR LES FICHIERS

3-1 L'ASSIGNATION

Un fichier est désigné par deux noms : un nom logique dit nom interne et un nom physique dit nom externe.

La première opération consiste à faire le lien entre ces deux noms de fichier



CHAPITRE 6 : LES FICHIERS

3- OPERATIONS FONDAMENTALES SUR LES FICHIERS

3-1 L'ASSIGNATION

Une variable fichier interne doit être reliée à un fichier externe. Cette liaison se fera grâce à une procédure d'assignation.

```
ASSIGN ( Variable_fichier_interne, Nom_de_fichier_externe );
```

Variable_fichier_interne : Fichier

Nom_de_fichier_externe : chaîne de caractères

Exemple :

```
ASSIGN ( Repertoire , 'C:\Pascal\Phone.Dat' );
```

```
Fic_Ext := 'C:\Pascal\Phone.Dat' ;
```

```
ASSIGN ( Repertoire , Fic_Ext );
```

CHAPITRE 6 : LES FICHIERS

3- OPERATIONS FONDAMENTALES SUR LES FICHIERS

3-2 OUVERTURE DE FICHIERS

Après l'étape d'assignation, un fichier doit être ouvert. Deux cas sont possibles :

Le fichier n'existe pas, nous allons le créer. Il s'agit donc d'un nouveau fichier. Dans ce cas l'ouverture se fait à l'aide de la procédure :

REWRITE (F)

où F est le nom du fichier logique, par exemple : REWRITE (répertoire)

Rewrite crée un nouveau fichier de nom externe celui qui est assigné à F. Si un fichier externe de même nom existe, il est détruit et un nouveau fichier vide est créé à sa place. Si F est ouvert, il est fermé et recréé. La position courante est placée en début du fichier vide.

CHAPITRE 6 : LES FICHIERS

3- OPERATIONS FONDAMENTALES SUR LES FICHIERS

3-2 OUVERTURE DE FICHIERS

Après l'étape d'assignation, un fichier doit être ouvert. Deux cas sont possibles :

Le fichier existe déjà et nous voulons le consulter ou le mettre à jour (création de nouveaux composants, suppression de composants, modification de composants) alors l'ouverture se fera par la procédure:

RESET(F)

où F est le nom du fichier logique, par exemple : RESET (répertoire)

Reset ouvre le fichier externe existant qui est assigné au fichier logique F.

Une erreur se produit si le fichier donné n'existe pas. Si F est déjà ouvert, il est automatiquement fermé et réouvert. La position courante du fichier est au début du fichier.

CHAPITRE 6 : LES FICHIERS

3- OPERATIONS FONDAMENTALES SUR LES FICHIERS

3-3 FERMETURE DE FICHIERS

Lorsque le traitement d'un fichier est terminé il ne faut pas oublier de le fermer grâce à la procédure :

CLOSE (F)

où F est le nom du fichier logique, par exemple : CLOSE (Repertoire)

CHAPITRE 6 : LES FICHIERS

3- OPERATIONS FONDAMENTALES SUR LES FICHIERS

3-4 FONCTION IORESULT

On peut contrôler si une opération d'entrée sortie s'est déroulée normalement par l'utilisation de la fonction IORESULT.

Elle retourne une valeur indiquant si la dernière opération d'E/S s'est déroulée sans erreur. (0 indique une opération sans erreur)

Pour capturer si une erreur d'E/S se produit il faut positionner à Off le switch d'E/S {\$i-}

Exemple :

```
Begin
  Assign(F, FileName) ;
  {$I-}
  Reset(F);
  {$I+}
  if IOResult = 0 then
    writeln ('Le fichier ', FileName, ' existe')
  else
    Close(F);
end.
```

CHAPITRE 6 : LES FICHIERS

3- OPERATIONS FONDAMENTALES SUR LES FICHIERS

3-5 LECTURE DE FICHIERS / ECRITURE DANS LES FICHIERS

Nous pouvons lire ou écrire un composant d'un fichier à l'aide des procédures que nous connaissons déjà et qui sont WRITE et WRITELN, et READ et READLN à la seule différence que nous avons utilisé jusqu'à présent les noms de fichiers par défaut (écran pour l'écriture et le clavier pour la lecture). Donc pour lire ou écrire dans un fichier il suffit de préciser son nom.

WRITE (f , paramètre1 , paramètre2,...,paramètre n)

WRITELN (f , paramètre1 , paramètre2,..., paramètre n)

READ (f , paramètre1 , paramètre2,..., paramètre n)

READLN (f , paramètre1 , paramètre2 , ...,paramètre n)

Où f : fichier

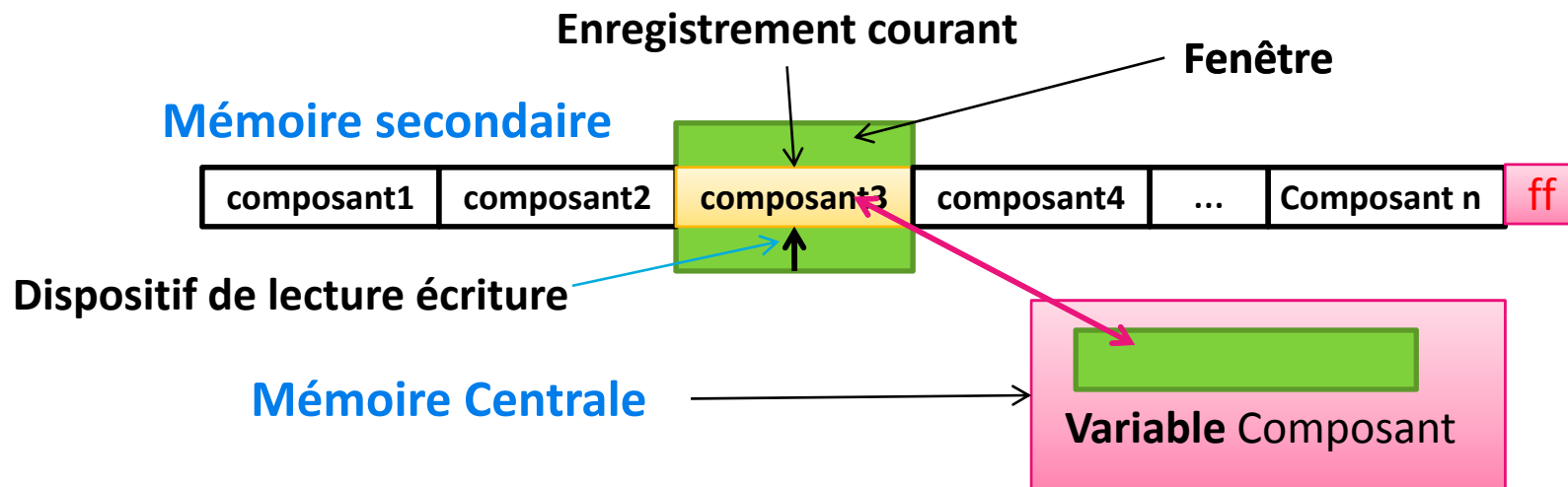
Exemple : READ (repertoire, article)
 WRITE (repertoire, article)

CHAPITRE 6 : LES FICHIERS

3- OPERATIONS FONDAMENTALES SUR LES FICHIERS

3-6 Variable fenêtre ou variable tampon

Il faut toujours déclarer une variable appelée variable tampon ou variable fenêtre qui est du même type que le composant du fichier. Et, lors d'une lecture, en écrivant LIRE (F , fenêtre) le contenu du composant se trouvant à la position fixée par le pointeur est mis dans la variable fenêtre. De la même manière, en écrivant ECRIRE (F, fenêtre) le contenu de la variable fenêtre est écrit dans le composant se trouvant à la position fixée par le pointeur



CHAPITRE 6 : LES FICHIERS

4-FICHIERS A ACCES SEQUENTIEL ET FICHIERS A ACCES DIRECT

ORGANISATION DES FICHIERS

L'organisation des fichiers traite les différentes manières possibles d'organiser les enregistrements d'un fichier.

IL EXISTE PLUSIEURS TYPES D'ORGANISATION DE FICHIERS

ORGANISATION SEQUENTIELLE

ORGANISATION DIRECTE ALEATOIRE OU RELATIVE

ORGANISATION INDEXEE

ordre physique = ordre logique : *organisation séquentielle* (naturelle à Pascal)

enregistrements de taille fixe, portant un numéro qui est relatif au début du fichier : *organisation relative*

chaque enregistrement est associé à une clé : *organisation indexée*

CHAPITRE 6 : LES FICHIERS

4-FICHIERS A ACCES SEQUENTIEL ET FICHIERS A ACCES DIRECT

Méthodes d'accès aux données

Les méthodes par lesquelles on lit ou on écrit un enregistrement d'un fichier sont appelées les *méthodes d'accès*.

La *méthode d'accès* que l'on veut utiliser doit être spécifiée *au moment de l'ouverture du fichier*. Un même fichier peut être accédé par des méthodes différentes selon son *organisation* qui ,elle a été définie *au moment de sa création*.

Les différentes méthodes d'accès sont:

accès séquentiel : enregistrements traités en séquence

accès direct : accès direct par le numéro d'enregistrement

accès indexé : accès par l'ordre des clés d'accès

Rq : *on ne peut utiliser que la méthode d'accès séquentielle avec une organisation de fichier séquentielle (Turbo Pascal permet également la méthode d'accès direct)*

CHAPITRE 6 : LES FICHIERS

4-FICHIERS A ACCES SEQUENTIEL ET FICHIERS A ACCES DIRECT

Dans les fichiers à accès séquentiel, les composants ou articles sont rangés les uns à la suite des autres de telle manière à ce que si l'on veut accéder à un composant nous sommes obligés de lire tous ceux qui se trouvent avant.

Lors de la lecture dans un fichier à accès séquentiel après les étapes d'assignation et d'ouverture, le pointeur se place automatiquement sur le premier composant (position 0) , et aussitôt que vous faites une lecture le pointeur se place automatiquement sur le composant suivant Mais, faites attention ! vérifiez si ce n'est pas la fin du fichier, car tenter de lire au delà de la fin du fichier provoque une erreur qu'il faudra éviter.

Cependant, dans les fichiers à accès direct, il est possible d'accéder directement à un composant. Dans ce cas la répartition des composants, lors de la création du fichier ou de leur recherche dans des opérations de consultation ou de mise à jour, se fait grâce à l'utilisation de formule de transformation de la forme $F(\text{indicatif}) = \text{Adresse physique ou adresse relative}$ ou grâce à des tables -d'index dans lesquelles on aura des couples (indicatif, position).

Dans les fichiers à accès direct, une fois que vous avez la position du composant (qui est fournie par votre formule ou votre table index) vous pouvez y accéder directement en utilisant la procédure `SEEK (f, POS)`.

Exemple : copier un fichier dans un autre.

ALGORITHME copie_de_fichiers

CONST long_max_nom = 30

TYPE t_auteur_livre = chaîne[long_max_nom]

t_livre = enregistrement

nom_auteur : t_auteur_livre

cote : entier

annee : entier

fin

t_fichier_biblio = fichier de t_livre_cote

VAR original : t_fichier_biblio

copie : t_fichier_biblio

livre : t_livre

DEBUT (* copie_de_fichiers *)

ASSIGN (ORIGINAL,'C:\pascal\Book.dat')

ASSIGN (ORIGINAL,'C:\pascal\BookCopie.dat')

RESET(original) (* ouvrir le fichier à copier *)

REWRITE(copie) (* initialiser la copie *)

TANT QUE non FF(original) faire

Dtq

LIRE (original,livre) (* parcourir les éléments du fichier à copier *)

ECRIRE (copie,livre) (* copier l'élément courant d'une fenêtre dans l'autre *)

(* passer à l'élément à copier suivant *)

Ftq

(* fermeture DES fichiers*)

FERMER (ORIGINAL)

FERMER (COPIE)

FIN

Exemple : Lecture d'un fichier de nombres réels et calcul de leur somme

...

TYPE FichierSequentiel = **FILE OF Real** ;

...

VAR F: FichierSequentiel;

...

FUNCTION SommeFichierReel (**VAR** Fichier: FichierSequentiel): **Real**;

VAR X, S: **Real** ;

BEGIN

 S:=0;

WHILE NOT Eof (Fichier) **DO**

BEGIN

Read (Fichier, X);

 S := S + X

END ; (* WHILE, Eof(Fichier) *)

 SommeFichierReel := S

END ; (* -- SommeFichierReel *) ..

BEGIN {main}

Assign (F, 'C:\REELS.DAT');

Reset (F); (* ouverture du fichier en mode lecture *)

WriteIn (SommeFichierReel(F));

Close (F)

END . (* -- main *)

CHAPITRE 6 : LES FICHIERS

4-FICHIERS A ACCES SEQUENTIEL ET FICHIERS A ACCES DIRECT

Cas particulier : les fichiers de texte

- Les fichiers de texte sont des ***cas particuliers de fichiers***. Un fichier de texte est formé d'éléments bien connus : les caractères. Chacun a déjà manipulé de tels fichiers : un programme (Pascal ou autre) est en fait un fichier de texte!
- Les caractères contenus dans un fichier de texte sont organisés en lignes, chacune terminée par une ***marque de fin de ligne***. Après la dernière ligne, le fichier se termine par une ***marque de fin de fichier***.
- Tout ce qui a été dit est valable pour les fichiers de texte. Précisons simplement qu'un fichier est un fichier de texte s'il est déclaré au moyen du type prédéfini **text**.

Rq : Un fichier de type text n'est généralement pas équivalent à un fichier de "type" : file of char qui, lui, ne possède pas une structure de lignes !

• Déclaration d'un fichier de texte :

f_text : text (* text est un fichier de type texte,
mot réservé en Pascal, file of n'est pas nécessaire avec les fichiers textes *)

CHAPITRE 6 : LES FICHIERS

4-FICHIERS A ACCES SEQUENTIEL ET FICHIERS A ACCES DIRECT

Lecture dans un fichier texte

Lecture d'un fichier texte, 2 options

Un fichier texte (étant une suite séquentielle de caractères) peut se lire caractère par caractère, ou ligne par ligne.

Le choix dépendra du programme à réaliser, généralement on lit ligne par ligne.

Partie déclarative des deux programmes

```
exemple : text;      (* fichier texte *)  
caractere : char;    (* caractère simple *)  
chaine_caracteres : string; (* chaîne de caractères *)
```

Version lecture caractère par caractère

```
while not eof ( exemple ) do  
Begin  
  while not eoln ( exemple ) do  
    Begin  
      read ( exemple, caractere );  
      ... (* traiter le caractère lu *)  
    End;  
  readln ( exemple );  
End;
```

Version lecture ligne par ligne

```
while not eof ( exemple ) do  
Begin  
  readln ( exemple, chaine_caractere );  
  ... (* traiter la chaine lue *)  
End;
```

CHAPITRE 6 : LES FICHIERS

Commandes supplémentaires pour les fichiers

filepos (< File >); (* position actuelle du repère *)

filesize (< File >); (* taille du fichier en enregistrements 0 si vide*)

Rq : ces commandes sont basées sur les enregistrements d'un fichier mais elles peuvent être utilisées avec un fichier texte (moins utile)

erase (< File >); (* efface le fichier File *)

rename (< File >, < nouveau_nom >); (* renomme le fichier *)

*Rq : ces commandes **nécessitent que le fichier soit fermé** avant de pouvoir être utilisées*

seek (< File >, < Num >);

(* positionne le repère sur l'enregistrement de numéro Num *)

Rq : la commande seek permet donc d'accéder directement à un enregistrement spécifique, cela peut être très utile dans un gros fichier binaire, elle permet d'utiliser un accès direct, et ce même dans un fichier séquentiel

CHAPITRE 6 : LES FICHIERS

Accès direct à un enregistrement dans un fichier par la commande seek

const

```
long_max_nom = 30; (* nombre maximal de caractères d'un nom *)  
long_max_titre = 80; (* nombre maximal de caractères d'un titre *)
```

type

```
t_auteur_livre = string [ long_max_nom ];  
t_titre_livre = string [ long_max_titre ];  
t_livre = record      (* représente une fiche bibliographique *)  
    titre : t_titre_livre;    (* titre du livre *)  
    auteur : t_auteur_livre; (* nom de l'auteur *)  
    Cote : integer;    (* cote en bibliothèque *)  
    annee : integer;   (* année de parution *)  
end;  
t_fichier_biblio = file of t_livre;
```

var

```
BU : t_fichier_biblio;  
Courant : t_livre;  
Num : integer;
```

CHAPITRE 6 : LES FICHIERS

Accès direct à un enregistrement dans un fichier par la commande seek

```
begin
  (* assignation *)
  ASSIGN( BU, 'C:\Biblio.dat')
  (* ouverture *)
  reset (BU); (* saisie du numéro de l'enregistrement recherché *)
  repeat
    writeln ('quel est le numéro de l'enregistrement ?');
    readln (Num);
  until Num in [0..filesize(BU)-1];
  (* lecture de l'enregistrement *)
  seek (BU, Num);
  read (BU, Courant); (* affichage *)
  with Courant do
    writeln (titre, '/', auteur, '/', annee, '/', cote);
  (* fermeture *)
  close (BU);
end.
```

CHAPITRE 6 : LES FICHIERS

Commandes supplémentaires pour les fichiers Texte

APPEND (< File >) (*Ouvre un fichier existant déjà pour ajouter du texte en fin de fichier *)

```
Program exemple ;
uses WinCrt;
var F: Text;
    lig : string;
begin
    Assign(F, 'TEST.TXT');
    Rewrite(F);          { Création d'un nouveau fichier}
    Writeln(F, 'Le cours d'Algo de la section B');
    Close(F);
    Append(F);           { Ajouter du texte à la fin }
    Writeln(F, 'se déroule en Amphi 6');
    Close(F);
    Reset (f) ;
    While not eof (f) do
        begin
            READLN(f,lig);
            Writeln (lig);
        End;
    Close (f)
End.
```


CHAPITRE 6 : LES FICHIERS

```
Program exemple ;
uses WinCrt;
type
  TenrLiv = record
    NumLiv : integer;
    Titre , Auteur : string[25] ;
    Theme , Editeur : string[15] ;
    Langue : Char ; Annee : integer ;
  end ;
var
  f: file of tenrliv;  x : tenrliv ; size : Longint;
begin
  Assign(f, 'c:\livre.dat');  Reset(f);
  size := FileSize(f); (* Donne le nombre d'enregistrements de f*)
  Writeln('Taille du fichier en nb d"enregistrements: ',size);
  Writeln('Se positionner sur l"enregistrement du milieu du fichier');
  Seek(f,size div 2);
  Writeln('la Position est ',FilePos(f));
  while not eof(f) do
    begin
      read(f,x);
      writeln(x.numliv,x.titre,' ',x.auteur,' ',x.theme,' ',x.editeur,' ',x.langue,' ',x.annee);
    end ;
  Close(f);
end.
```

CHAPITRE 6 : LES FICHIERS

Program exemple ;

(* Ce programme permet de créer un fichier de 100 enregistrements vides *)

uses WinCrt;

type

 TenrLiv = record

 NumLiv : integer;

 Titre , Auteur : string[25] ;

 Theme , Editeur : string[15] ;

 Langue : Char ; Annee : integer ;

 end ;

var

 f: file of tenrliv; livre : tenrliv ; i : Integer ;

begin

 Assign(f, 'c:\livre.dat');

 Rewrite(f);

 for i :=0 to 99 do

 write(f,livre);

 Close(f);

 writeln ('fichier créé avec succès') ;

end.

CHAPITRE 6 : LES FICHIERS

Program exemple ;

(* Ce programme permet de remplir le fichier créé précédemment en utilisant une méthode de transformation de la clé qui est le N° du livre *)

uses WinCrt;

type

TenrLiv =

var

f: file of tenrliv; livre : tenrliv ; i : Integer ;fini : Boolean ;

Function Transform (Nl : Integer) : integer ; (* donne un entier compris entre 0 et 100 *)

begin

Assign(f, 'c:\livre.dat');

Reset(f); fini := false ;

While not fini do

BEGIN

With Livre do

Begin

Readln (numlivre) ;

Readln (titre,auteur ,...,annee) ;

seek(f,Transform(numlivre)) ;

End ;

write(f,livre) ;

(* demander si d'autres livres si non fini := true*)

End ;

End.

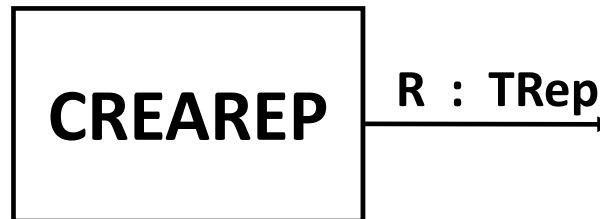
CHAPITRE 6 : LES FICHIERS

EXEMPLE

construire un répertoire téléphonique, puis le lister et enfin le consulter en séquentiel puis en direct.

1. création du répertoire

PROCEDURE



Rôle : Crée un fichier R de type TRep

TYPE

comp = ENREGISTREMENT

nom : chaîne [25]

notel : entier

FIN

TRep = FICHIER de comp

ANALYSE :

On va créer pour la première fois le fichier R et on enregistrera un à un les composants.

CHAPITRE 6 : LES FICHIERS

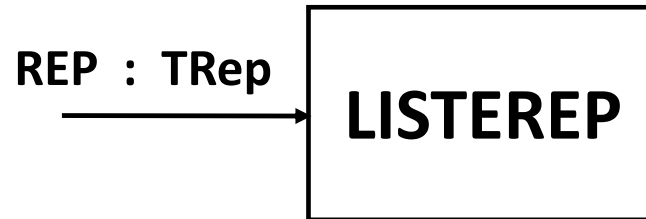
1. création du répertoire

```
PROCEDURE creafich (var rep:Trep);    (* PROCEDURE DE CREATION DU FICHIER REPERTOIRE *)
var  article:composant; (* déclaration de la variable tampon *)
    encore:char ;
BEGIN
    clrscr; writeln('Nous allons créer le fichier répertoire - tapez sur ENTREE '); readln;
    assign(rep,'C:\RepTel.dat');      (*Lien entre REP et RepTel.DAT*)
    {$i-}                             (* détection automatique d'erreur supprimée *)
    rewrite(rep);                      (* ouverture du fichier que l'on va créer *)
    {$i+}                             (* détection automatique d'erreur rétablie *)
    if ioresult = 0 then (* opération d'entree/sortie sans problème *)
    BEGIN
        repeat
            clrscr;   Gotoxy(20,5);
            write('NOM : '); readln(article.nom);
            Gotoxy(20,8);
            write('NUMERO DE TELEPHONE : '); readln(article.notel);
            Write (rep,article);
            write('Y at-il encore des éléments a rajouter ? (tapez O/N)');
            readln(encore);
        until (encore = 'n') OR (encore = 'N');
        Close (rep);
    END
    ELSE (* problème au niveau de l'opération d'E/S *)
        writeln('ERREUR A L'OUVERTURE DU FICHIER REP');
END;
```

CHAPITRE 6 : LES FICHIERS

2. Liste du répertoire

PROCEDURE



Rôle : Liste les composants du fichier Rep

ANALYSE : Tous les composants de Rep sont lus un à un et écrit sur l'imprimante

```
PROCEDURE listfile (var rep:r); (* PROCEDURE D'IMPRESSION DU FICHIER REPERTOIRE
var  article : composant;
BEGIN
    assign(rep,'C:RepTel.dat');      (* assignationentrerepetf1.dat*)
    reset(rep);(* ouverture du fichier répertoire *)
    {$i+}
    writeln(lst,'          NOM    NUMERO DE TELEPHONE ');
    While Not eof(Rep) do          (* détection de la fin du fichier *)
    begin
        read(rep,article);
        writeln(article.nom : 20,article.notel:10);
    end ;
    close (rep);                    (* fermeture du fichier rep    *)
END;
```

CHAPITRE 6 : LES FICHIERS

```
PROCEDURE CONSREPS ( var rep:Trep; ncor:chaîne; var tele:longint);  
(*  PROCEDURE DE CONSULTATION SEQUENTIELLE DU FICHIER REPERT      *)  
(*  on donne un nom et elle retourne un numéro de téléphone      *)  
(*  ou zéro dans le cas ou le nom n'existe pas                    *)  
VAR      trouve: Boolean;  
          article: composant;  
BEGIN  
    reset(rep);          (*  ouverture de rep      *)  
    trouve:=False;  
    repeat  
        read(rep,article);  
        if article.nom = ncor then  
            BEGIN  
                tele := article.notel;  
                trouve :=true;  
            END;  
    until eof(rep) OR trouve ;  
    if not trouve then tele :=0;  
    close (rep);          (*  fermeture de rep      *)  
END;
```

CHAPITRE 6 : LES FICHIERS

3. CONSULTATION EN ACCES DIRECT DU RÉPERTOIRE

```
PROCEDURE CONSREPD( var rep:Trep; position:integer; var tele:longint);  
(* PROCEDURE DE CONSULTATION EN ACCES DIRECT DU FICHIER REPERT *)  
(* on donne la position du correspondant et elle retourne *)  
(* un numéro de téléphone ou zéro dans le cas ou le nom n'existe pas *)  
VAR article: Composant;  
BEGIN  
    reset(rep);          (* ouverture de rep *)  
    seek(rep,position-1 );  
    read(rep,article);  
    tele := article.notel;  
    close (rep);          (* fermeture de rep *)  
END;
```


CHAPITRE 6 : LES FICHIERS

MISE A JOUR DES FICHIERS

La mise à jour d'un fichier consiste à :

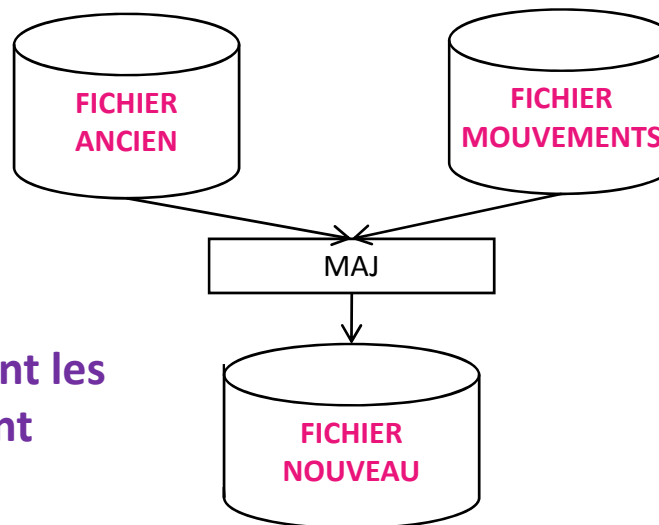
- ajouter ou insérer de nouveaux composants
- modifier des composants qui existent
- supprimer des composants qui existent

MISE A JOUR DES FICHIERS SEQUENTIELS

La mise à jour d'un fichier séquentiel nécessite la création d'un nouveau fichier mis à jour

Fichier permanent contenant les enregistrements à un instant T

Fichier permanent contenant les enregistrements à un instant T+1



Fichier contenant les mouvements c.-à-d. les modifications à apporter sur le fichier permanent
Création de nouveaux , Modification ou suppression d'anciens

CHAPITRE 6 : LES FICHIERS

MISE A JOUR DES FICHIERS

PRINCIPE DE LA MISE A JOUR DES FICHIERS SEQUENTIELS

- Lecture d'un enregistrement du F-ANC et d'un enregistrement du F-MVT
- Trois cas peuvent se produire :
 1. L'indicatif du F-ANC est $<$ à l'indicatif du F-MVT
Cela signifie que pour cet enregistrement il n'y a pas de mouvement et auquel cas il faut le réécrire dans F-NOUV tel quel. Ceci s'appelle une **RECONDUCTION**
 2. L'indicatif du F-ANC est $=$ à l'indicatif du F-MVT
Cela signifie que pour cet enregistrement il y a un mouvement et ce mouvement dépend du code de mise à jour (C , S , ou M)
Cas Code_Maj Parmi
 'C' : 'ERREUR'
 'M' : Modification de l'Ancien par le mouvement
 'S' : Suppression
Fincas
 3. L'indicatif du F-ANC est $>$ à l'indicatif du F-MVT
Cela signifie que pour ce mouvement il n'y a pas d'enregistrement ancien et donc c'est une création. Le code_Maj doit être obligatoirement = 'C'
Si Code_Maj ='C' ALORS Création
Sinon 'ERREUR'

CHAPITRE 6 : LES FICHIERS

MISE A JOUR DES FICHIERS

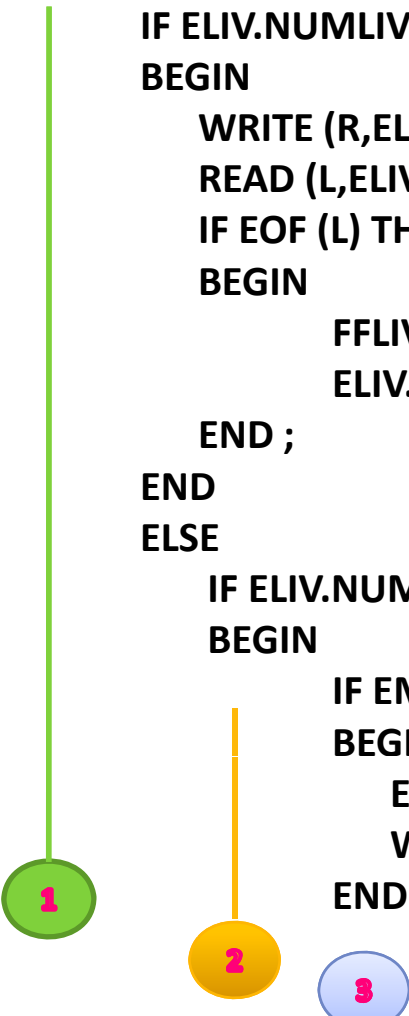
ALGORITHME

```
PROCEDURE MAJLIVRE ( VAR L : TFLIV ; VAR M : TFMOUV ; VAR R : TFLIV ) ;
VAR
  FFLIV , FFMOUV : BOOLEAN ;
  ELIV , ERES : TENRLIV ;
  EMOUV : TENRMOUV ;
BEGIN
  ASSIGN ( L , 'C:\LIVRE.DAT' ) ; ASSIGN ( M , 'C:\MOUVLIV.DAT' ) ; ASSIGN ( R , 'C:\NOUVLIV.DAT' ) ;
  RESET ( M ) ; RESET ( L ) ; REWRITE ( R ) ;
  READ ( L , ELIV ) ;
  IF EOF ( L ) THEN
    BEGIN
      FFLIV := TRUE ;
      ELIV.NUMLIV := HIGH_VALUE ;
    END ;
  READ ( M , EMOUV ) ;
  IF EOF ( M ) THEN
    BEGIN
      FMOUV := TRUE ;
      EMOUV.NUMLIV := HIGH_VALUE ;
    END ;
```

CHAPITRE 6 : LES FICHIERS

MISE A JOUR DES FICHIERS

```
WHILE NOT FFLIV AND NOT FFMOUV DO
BEGIN
    IF ELIV.NUMLIV < EMOUV.NUMLIV THEN
    BEGIN
        WRITE (R,ELIV) ;
        READ (L,ELIV) ;
        IF EOF (L) THEN
        BEGIN
            FFLIV := TRUE ;
            ELIV.NUMLIV := HIGH_VALUE ;
        END ;
    END
ELSE
    IF ELIV.NUMLIV = EMOUV.NUMLIV THEN
    BEGIN
        IF EMOUV.CMOUV = 'M' THEN
        BEGIN
            ERES.NUMLIV := EMOUV.NUMLIV ;
            WRITE (R,ERES)
        END
    END
```



2

3

```
ELSE
  IF EMOUV.CMOUV = 'C' THEN
    WRITELN ( 'ERREUR');
  READ (L,ELIV) ;
  READ (M,EMOUV) ;
  IF EOF (L) THEN
    BEGIN
      FFLIV := TRUE ;
      ELIV.NUMLIV := HIGH_VALUE ;
    END ;
  IF EOF (M) THEN
    BEGIN
      FFMOUV := TRUE ;
      EMOUV.NUMLIV := HIGH_VALUE ;
    END ;
```

END

1

```
ELSE
BEGIN
  IF EMOUV.CMOUV <> 'C' THEN
    WRITELN ( 'ERREUR : MOUVEMENT POUR LIVRE INEXISTANT ' )
  ELSE
    BEGIN
      ERES.NUMLIV := EMOUV.NUMLIV ;
      WRITE (R,ERES) ;
    END;
    READ (M,EMOUV) ;
    IF EOF (M) THEN
      BEGIN
        FFMOUV := TRUE ;
        EMOUV.NUMLIV := HIGH_VALUE ;
      END ;
    END ;
  END ;
END ;
END ;
```

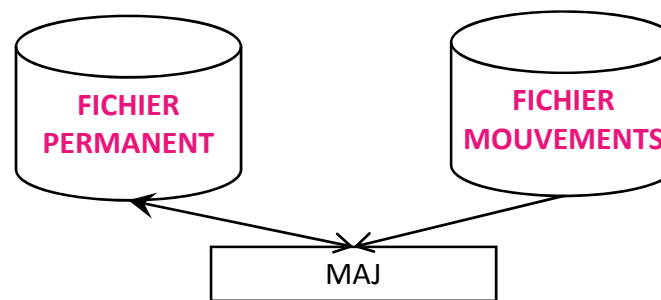
CHAPITRE 6 : LES FICHIERS

MISE A JOUR DES FICHIERS A ACCES DIRECT

La mise à jour d'un fichier à accès direct ne nécessite pas un nouveau fichier
La mise à jour se fait directement sur le fichier permanent

Fichier permanent contenant les enregistrements à TOUT instant T

Fichier contenant les mouvements c.-à-d. les modifications à apporter sur le fichier permanent : Création de nouveaux , Modification ou suppression d'anciens



CHAPITRE 6 : LES FICHIERS

MISE A JOUR DES FICHIERS

PRINCIPE DE LA MISE A JOUR DES FICHIERS A ACCES DIRECT

- TANT QUE NON FF(F-MVT) FAIRE
DEBUT

Lecture d'un enregistrement F-MVT

CAS Code-Maj PARMi

'C' : Si l'enregistrement correspondant à la Clé n'existe pas ALORS **Création** de
l'enregistrement en écrivant les données sur le fichier permanent
Sinon ERREUR

'M' : Si l'enregistrement correspondant à la Clé existe ALORS **Modification** de
l'enregistrement en **réécrivant** les données sur le fichier permanent
Sinon ERREUR

'S' : Si l'enregistrement correspondant à la Clé existe ALORS **Suppression Logique**
de l'enregistrement en mettant à jour le Caractère d'effacement
Sinon ERREUR

FIN CAS

LECTURE (F-MVT,E-MVT)

FIN

CHAPITRE 6 : LES FICHIERS

TRI DE FICHIERS

1- TRI INTERNE

Utilisé si le fichier à trier n'est pas volumineux

On charge tout le fichier en mémoire dans un tableau à une dimension

On trie le tableau en mémoire en utilisant une méthode de tri

On recopie le tableau ainsi trié dans le fichier élément par élément

ALGORITHME

DEBUT

RESET (F) ;

$i \leftarrow 1$

TANT QUE NO FF (F) FAIRE

DTQ

LIRE (F ,T[i])

$i \leftarrow i+1$

FTQ

TriSelection (T)

REWRITE (F) ;

pour $j \leftarrow 1$ à i FAIRE

ECRIRE (F, T[j])

FIN

CHAPITRE 6 : LES FICHIERS

2- TRI EXTERNE Tri balancé par monotonies naturelles

Définition : On appelle monotonie une suite triée d'entiers.

Soit F0 un fichier d'entiers le fichier à trier ; on utilise trois fichiers de travail : F1, F2 et F3. L'idée consiste à traiter les monotonies « comme elles se présentent », On peut facilement repérer la fin d'une monotonie : lorsqu'on rencontre un élément plus petit, ou lorsqu'on atteint la fin du fichier.

Etape 1 : Eclatement de F0 sur F2, F3 (monotonies naturelles)

F0 : 11 18 32 47 12 25 10 53 62 21
donne F2 : 11 18 32 47 10 53 62
 F3 : 12 25 21

Etape 2 : Interclassement ou fusion de F2, F3 sur F0, F1 (monotonies naturelles)

F0 : 11 12 18 25 32 47
F1 : 10 21 53 62

Etape 3 : Interclassement ou fusion de F0, F1 sur F2, F3 (monotonies naturelles)

F2 : 10 11 12 18 21 25 32 47 53 62

CHAPITRE 6 : LES FICHIERS

2- TRI EXTERNE Tri balancé par monotonies naturelles : ALGORITHME

```
PROCEDURE Tri_Externe ( VAR F0 : fichier d'entiers )
VAR      F1, F2, F3 : fichiers d'entiers
PROCEDURE ECLATEMENT ( VAR F0,F1,F2 : FICHER d'ENTIERES)
PROCEDURE INTERCLASSEMENT ( VAR F0,F1,F2,F3 : FICHER d'ENTIERES)
PROCEDURE RECOPIE (VAR F0,F1: FICHER d'ENTIERES)
DEBUT
    FINI ← FAUX
    Eclatement (F0, F2, F3)                { étape 1 }
    TANT Que NON FINI FAIRE                { autres étapes }
    DEBUT
        SI FILESIZE (F3) <> 0 ALORS
            InterClassement (F2, F3, F0, F1) ;
            SI FILESIZE (F1) <> 0 ALORS
                InterClassement (F0, F1, F2, F3) ;
            SINON
                FINI ← VRAI
        SINON
            DSIN
                FINI ← VRAI
                RecopieFichier(F2,F0)
            FSIN
    FIN
FIN
```