Faculté des Sciences

Département Informatique

Module : Algorithmique II

Niveau Tronc Commun informatique appliquée

Enseigné par :

- Pr. Ayoub AMRANI
- Email: amrani.ayoub@uit.ac.ma

Déroulement du cours

- Rappel
- Fonction et Procédure
- Enregistrements et unions
- Les fichiers
- Recherche d'un élément dans un tableau
- Tri des éléments d'un tableau
- La complexité

Chapitre 2 : Fonction et Procédure

Procédure

2.1. Définition:

Une série d'instructions regroupées sous un nom, qui permet d'effectuer des actions par un simple appel de la procédure dans un algorithme ou dans un autre sous-algorithme.

Avantage

→ Améliorer la lisibilité des programmes
→ Optimiser le nombre d'instructions dans un programme
→ faciliter la mise au point et la correction des erreurs

Procédure

2.2. Appel de la procédure

```
Procédure Nom_de_la_procédure (Liste d'arguments : type)
        Les déclarations
DÉBUT
        Instructions de la procédure
FIN
```

- La première ligne s'appelle l'en-tête de la procédure.
- La liste d'arguments est une suite de données à échanger avec d'autres programmes.

```
Nom_de_la_procédure (Liste d'arguments)
```

Procédure

2.3. Procédure simple (sans paramètres)

Exemple:

```
ALGORITHME PROC
CONST
        titre = "Algorithmique"

Procédure TRAIT( )
VAR
        i : entier
DÉBUT
        Pour i de 1 à 13 faire
                Écrire ( " - " )
        FinPour
FIN

DÉBUT(Programme principal)
        Ecrire (titre)
        TRAIT( )        /* Appel de la fonction TRAIT*/
FIN
```

Procédure

2.3. Procédure simple (sans paramètres)

Remarque:

- La structure d'une procédure est analogue à celle d'un algorithme. Elle possède une entête, une partie déclarative et un corps.
- Pour appeler une procédure simple, il suffit d'écrire le nom de cette procédure.
- Une variable **locale** (privée) est déclarée dans une procédure et ne peut être utilisée qu'à l'intérieur de celle-ci.

Exercice 1 :

1. Ecrivez une procédure qui affiche « bonjour »
2. Appelez la procédure

Procédure

2.4. Procédure paramétrée (avec paramètres)

Une procédure paramétrée est un module qui utilise des paramètres pour faire passer des informations entre la procédure appelée et le programme appelant.

Procédure Nom_de_la_procédure (pf1 : type 1 ; pf2 : type 2 ; .....)

Procédure

2.4. Procédure paramétrée (avec paramètres) passage par valeur

Exemple:

```
ALGORITHMEPASSAGE_VALEUR
VAR
        x : entier

Procédure ESSAI ( i : entier )
DÉBUT
i ← 3 * i
Écrire ( "Le paramètre valeur i =" , i )
FIN

DÉBUT (P.P)
        Ecrire ( "Donner la valeur de X : " )
        Lire (x)
        Ecrire ( "Avant appel x = " , x )
        ESSAI ( x )                    /* Appel de la fonction ESSAI*/
        Ecrire ( "Après appel x = " , x )
FIN
```

Procédure

2.4. Procédure paramétrée (avec paramètres) passage par valeur

Interprétation:

- Le passage de paramètres par **valeur** permet au programme appelant de transmettre une valeur à la procédure appelée.
- Le paramètre formel « i » est une variable simple.
- Le paramètre effectif correspondant « x » est une expression de même type que le paramètre formel.

x = 10	P.P	Procédure
Appel de « ESSAI »	x = 10	i = 10
Exécution de « ESSAI »	x = 10	i = 30
Après « ESSAI »	x = 10	i = 30

Procédure

2.4. Procédure paramétrée (avec paramètres) passage par valeur

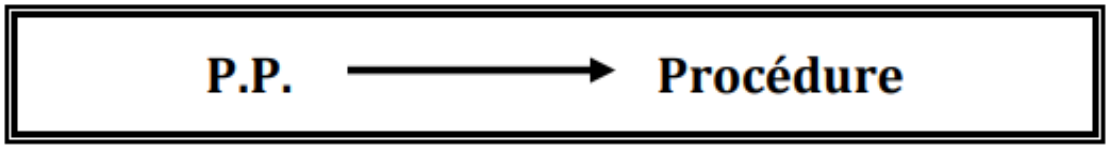
Remarque:

- Les paramètres effectifs et les paramètres formels doivent s'accorder du point de vue nombre et ordre.
- Leurs types doivent être identiques selon le mode de passage des paramètres

Conclusion:

Le transfert d'information est effectué dans un seul sens, du programme principal vers la procédure.

Sens du transfert :



Procédure

2.4. Procédure paramétrée (avec paramètres) passage par variable

Exemple:

```
ALGORITHME PASSAGE_VARIABLE
VAR
        x : entier

Procédure ESSAI ( VAR i : entier )
DÉBUT
 i ← 3 * i
Écrire ( "Le paramètre variable i =" , i )
FIN

DÉBUT(P.P)
        Ecrire ( "Donner la valeur de X : " )
        Lire (x)
        ESSAI ( x )                    /* Appel de la fonction ESSAI*/
        Ecrire ( "Après appel x = " , x )
FIN
```

Procédure

2.4. Procédure paramétrée (avec paramètres) passage par variable

Interprétation:

- Le passage de paramètres par **variable** permet au programme appelant de transmettre une valeur à la procédure appelée et vice-versa.
- Le paramètre formel « i » est une variable.
- Le paramètre effectif correspondant « x » est une expression de même type que le paramètre formel.

x = 10	P.P	Procédure
Appel de « ESSAI »	x = 10	i = 10
Exécution de « ESSAI »	x = 30	i = 30
Après « ESSAI »	x = 30	i = 30

Procédure

2.4. Procédure paramétrée (avec paramètres) passage par variable

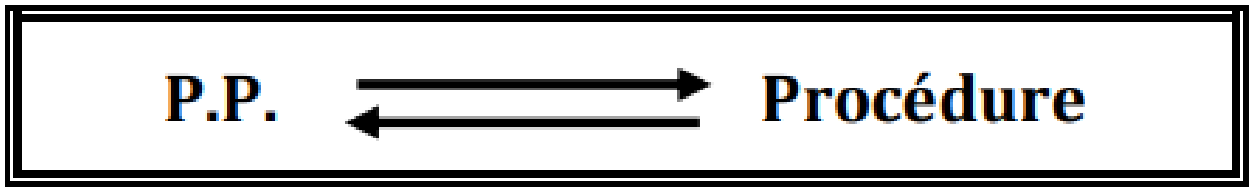
Remarque:

- Le transfert d'information est effectué dans les deux sens, du programme principal vers la procédure et vice-versa.

Conclusion:

Toute modification du paramètre formel entraîne automatiquement la modification de la valeur du paramètre effectif.

Sens du transfert :



Procédure

Exercice 2 :

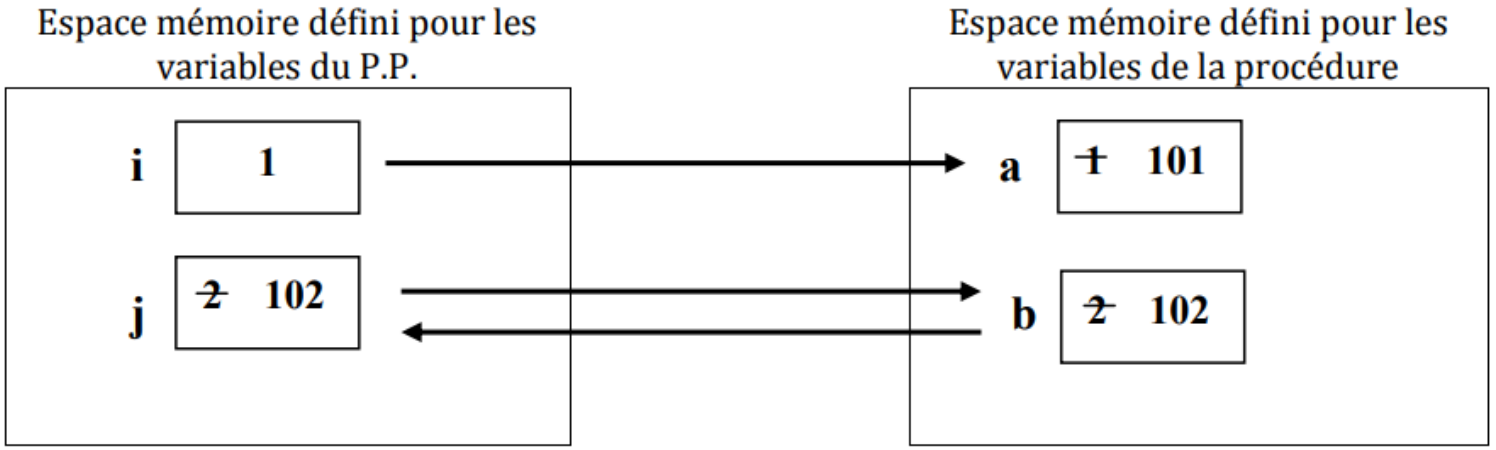
Faire la trace d'exécution de l'algorithme suivant :

```
ALGORITHME PARAMÈTRE
VAR
        i, j : entier
Procédure TRANSMISSION ( a : entier ; VAR b : entier )
DÉBUT
a ← a + 100
  b ← b + 100
Écrire ( " a =" , a , " b =" , b)
FIN

DÉBUT(P.P)
        i ← 1
        j ← 2
        TRANSMISSION ( i , j )
        Écrire ( " i =" , i , " j =" , j)
FIN
```

Procédure

Exercice 2 :



Procédure

Exercice 3 : PROCEDURE PERMUTATION DE DEUX ENTIER

```
ALGORITHME PERMUTATION
VAR
        x, y : entier

Procédure ÉCHANGE ( VAR a , b : entier )
VAR
vs : entier
DÉBUT
vs ← a
a  ← b
b  ← vs
Écrire ( " Dans ECHANGE a =" , a , " b =" , b)
FIN

DÉBUT(P.P)
        Écrire ( " Dnner x et y")
        Lire (x,y)
        Écrire ( " Avant appel de ECHANGE x =" , x , " y =" , y)
        ECHANGE ( x , y )
        Écrire ( " Aprés appel de ECHANGE x =" , x , " y =" , y)
FIN
```

Fonction

2.5. Définition:

une fonction est tout simplement une procédure qui ne retourne qu'un seul résultat, une seule valeur, au programme appelant

Remarque:

→ Il est obligatoire de préciser, dès le début le type de la fonction qui est en même temps le type du résultat à retourner.

Fonction

2.6. Structure d'une fonction:

```
Fonction Nom_de_la_fonction (Paramètres formels) : type
        Les déclarations
DÉBUT
        <Séquence d'instructions>
        Nom_de_la_fonction ⟵ Résultat
Fin
```

2.7. Structure d'une fonction:

```
Nom_de_la_fonction (<liste des paramètres effectifs>)
```

Fonction

Exemple:

Ecrire une fonction MIN qui retourne le minimum de 2 entiers a et b.

```
Fonction MIN (a , b : entier) : entier
VAR
    c : entier
DÉBUT
        Si (a<=b) alors
                c ← a
        si non
                c ← b
        Fin si
    MIN ← c
FIN
```

APPEL

```
1-    z ← MIN(3,5)
2-    Ecrire ( " Donner x et y")
      Lire (x,y)
      z ← MIN(x,y)
3-    t ← (7 + MIN(x,y)) / 2
4-    Ecrire ( " Le minimum = " , MIN(x,y))
5-    si (MIN(x,y) > 0 ) alors ……
```

Fonction

Exercice :

Comment utiliser la fonction MIN pour afficher le minimum de trois entiers x, y et z en une seule instruction et sans utiliser de variables supplémentaires.

Ecrire (" Le minimum = " , MIN (MIN(x,y) , z))

Remarque :

- **Le nom de la fonction joue un double rôle, c'est à la fois l'identifiant de la fonction et une variable locale.**
- **Dans une fonction, le passage de tous les paramètres se fait par valeur.**

Fonction

Conclusion :

Lors de l'utilisation d'une fonction, il faut :

- Spécifier le type de la fonction
- Déclaré, si nécessaire, une variable locale de même type que la fonction (pour faire les calculs intermédiaires)

Fonction

Exercice 1 :

Écrire une fonction qui permet de calculer le prix TTC , cette fonction va recevoir un paramètre de type Réel dont le nom est "prixHT" et un second paramètre de type Réel dont le nom est "tva".

```
fonction calculPrixTTC( prixHT, tva: réel): réel
Variables  prixTTC :  réel
Début
   prixTTC ← prixHT * (1 + tva / 100)
   retourner   prixTTC
Fin
```

Fonction

Exercice 2 :

Écrire une procédure qui permet d'afficher si un nombre entier passé en paramètre est pair ou impair.

```
procédure parite (nb:entier)
Début
  if(nb mod 2 = 0) alors
    Ecrire (nb,"est impair")
    Sinon
    Ecrire (nb," est pair")
  Fin
```

Chapitre 3 : Enregistrements

Enregistrements

3.1. Définition:

- Un enregistrement est un objet qui permet de regrouper des éléments qui n'ont pas forcément le même type dans une même structure.
- Un enregistrement est défini par un ensemble d'éléments appelés champs. Ces champs peuvent être de type simple (entier, réel, booléen) ou composé (tableau)

Exemple : Un enregistrement qui permet de représenter les informations d'un étudiant

Nom	Chaine de caractères
Prénom	Chaine de caractères
Matricule	Entier
Section	Car
Groupe	Entier

Enregistrements

3.2. Déclaration:

La déclaration d'une variable de type Enregistrement se fait dans la partie déclaration de l'algorithme.

```
Var
<Idf_enregistrement>= Enregistrement
                      <Idf_champs1> : <type_champs1> ;
                      <Idf_champs1> : <type_champs1> ;
                      <Idf_champs1> : <type_champs1> ;
                       .....
                      <Idf_champs n> : <type_champs n> ;
```

Enregistrements

3.3. Exemple:

Déclaration d'une variable Enregistrement appelée E qui représente les informations d'un étudiant

```
Var
E = Enregistrement
                Nom : Tableau de [1..50] de Car ;
                Prénom : Tableau de [1..50] de Car ;
                Matricule : Entier ;
                Section : Car ;
                Groupe : Entier ;
    Fin ;
```

Enregistrements

Note importante

Il est possible de déclarer un type d'Enregistrement dans la partie déclarations des types

Syntaxe

```
Type
<Idf_type_enregistrement>= Enregistrement
                          <Idf_champs1> : <type_champs1> ;
                          <Idf_champs1> : <type_champs1> ;
                          <Idf_champs1> : <type_champs1> ;
                           .....
                          <Idf_champs n> : <type_champs n> ;
                             Fin ;
Var
<idf_enregistrement> : <Idf_type_enregistrement> ;
```

Enregistrements

Exemple

```
Type
Etudiant = Enregistrement
                    Nom : Tableau de [1..50] de Car ;
                    Prénom : Tableau de [1..50] de Car ;
                    Matricule : Entier ;
                    Section : Car ;
                    Groupe : Entier ;
                    Notes : Tableau de[1..3] de Réels ;
    Fin ;
Var
E : Etudiant ;
```

Enregistrements

3.4. Les opérations sur les enregistrements

Référence d'un champ

Nous accédons aux champs d'un enregistrement en indiquant le nom de l'enregistrement suivi d'un point ensuite par le nom du champ

Syntaxe

```
<idf_enregistrement>.<idf_champs>
```

Exemple

E.Nom fait référence au champ Nom de l'enregistrement E

E.Prénom fait référence au champ Prénom de l'enregistrement E

E.Matricule fait référence au champ Matricule de l'enregistrement E

Enregistrements

3.4. Les opérations sur les enregistrements

Initialisation, lecture et affichage d'un champ

Le champ d'un enregistrement peut être de type simple ou composé.

- S'il est de type simple, alors nous pouvons appliquer les opérations de lecture, d'affectation et d'affichage comme dans le cas des variables simples.
- Si le champ est de type composé (tableau ou enregistrement), il faudra dans ce cas, descendre dans la hiérarchie jusqu'à trouver les éléments de type simple.

Enregistrements

3.4. Les opérations sur les enregistrements

Initialisation, lecture et affichage d'un champ

Exemple

Les champs de type simple :

```
E.Matricule ← 1717548963 ;  /  Lire (E.Matricule) ;  / Ecrire
(E.Matricule) ;
E.Section ← 'B' ;  /  Lire (E.Section) ;  / Ecrire
(E.Section) ;
E.Groupe← 3 ;   /  Lire (E.Groupe) ;  / Ecrire (E.Groupe) ;
```

Enregistrements

3.4. Les opérations sur les enregistrements

Initialisation, lecture et affichage d'un champ

Le champ **E.Notes** est un tableau de réels, nous ne pouvons pas faire une affectation ou une lecture directe. Il faut accéder aux éléments du tableau un par un.

```
E.Notes[1] ← 10.2 ;    /  Lire (E.Notes[1]) ;  /
Ecrire(E.Notes[1]) ;
E.Notes[2] ← 12.5 ;   /  Lire (E.Notes[2]) ;  /
Ecrire(E.Notes[2]) ;
E.Notes[3] ← 13.0 ;   /  Lire (E.Notes[3]) ;  /
Ecrire(E.Notes[3]) ;
```

Enregistrements

3.4. Les opérations sur les enregistrements

Initialisation, lecture et affichage d'un champ

Concernant les chaines de caractères, il est possible de les considérer comme des éléments de type simple et donc il est permis de faire l'affectation ou la lecture directement.

```
E.Nom← "FARIDA" ;   / Lire(E.Nom) ;    / Ecrire(E.Nom) ;
E.Prénom← "Bachtobdj" ;   / Lire(E.Prénom) ; / Ecrire(E.Nom) ;
```

Enregistrements

3.5. Emboîtement d'enregistrements

Il est possible qu'un champ dans un enregistrement soit lui-même un enregistrement.

Exemple

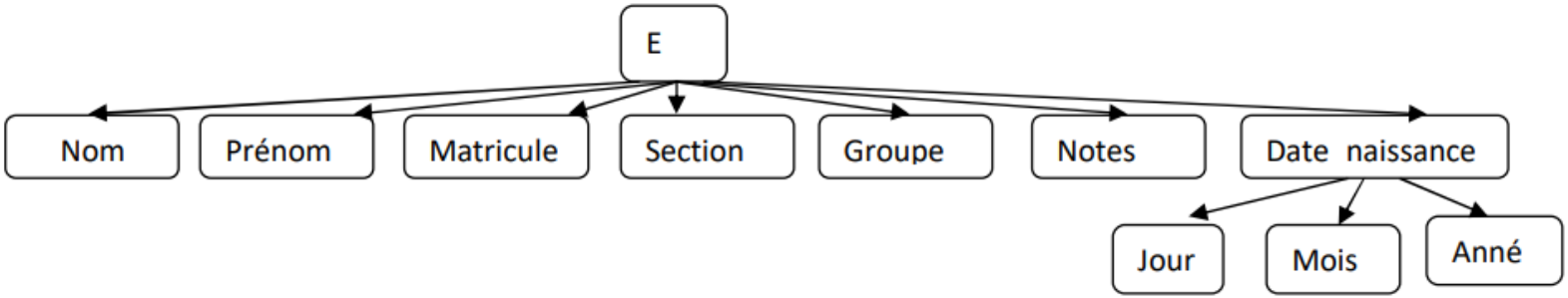
```
Type
Date=Enregistrement
      Jour :Entier ;
      Mois : Entier ;
      Année : Entier
      Fin ;

Etudiant = Enregistrement
                  Nom : Tableau de [1..50] de Car ;
                  Prénom : Tableau de [1..50] de Car ;
                  Matricule : Entier ;
                  Section : Car ;
                  Groupe : Entier ;
                  Notes : Tableau de[1..3] de Réels ;
                  Date_naissance : Date ;
    Fin ;
Var
E : Etudiant ;
```

Enregistrements

3.5. Emboîtement d'enregistrements

L'imbrication peut se voir dans le schéma suivant :



Enregistrements

3.6. Exercice

Un des services médicaux dans un hôpital étatique décide d'automatiser son archive. Pour cela il estime d'enregistrer les fiches de ses patient sur l'ordinateur en s'adaptant uns structure hétérogène. Cette dernière doit comporter les informations suivantes :

- Un identifiant du dossier (entier) *
- Un Nom (chaine de 25 caractères) et un prénom (25 caractères)
- Un numéro d'assurance (10 caractères maximum)
- Un numéro de tel (14 caractères maximum)
- Une adresse (Rue, désignation, commune, région=code sur 2 chiffres)
- Description de la maladie
- Date d'entrée et date de sortie(JJ.MM.AA)

Chapitre 3 : Les Fichiers

Les Fichiers

4.1. Definition

- **Un fichier (file)** est un ensemble structuré de données stocké en général sur un support externe (disquette, disque dur ou optique, ...). Ils servent à stocker des informations de manière permanente, entre deux exécutions d'un programme.
- **Un fichier structuré** contient une suite d'enregistrements homogènes, qui regroupent le plus souvent plusieurs composantes (champs).
- **Un fichiers séquentiel** contient des enregistrements qui sont mémorisés consécutivement dans l'ordre de leur entrée et peuvent seulement être lus dans cet ordre. Si on a besoin d'un enregistrement précis dans un fichier séquentiel, il faut lire tous les enregistrements qui le précèdent, en commençant par le premier.

Les Fichiers

4.2. Types de fichiers

- **Les fichiers textes :** sont les fichiers dont le contenu représente uniquement une suite de caractères imprimables, d'espaces et de retours à la ligne (.txt,...). Ils peuvent être lus directement par un éditeur de texte.
- **Les fichiers binaires :** sont les fichiers qui ne sont pas assimilables à des fichiers textes (.exe, .mp3, .png,...). Ils ne peuvent pas être lus directement par un éditeur de texte .

Les Fichiers

4.3. Structure de fichiers

- **Structure délimitée :** Elle utilise un caractère spécial, appelé caractère de délimitation, qui permet de repérer quand finit un champ et quand commence le suivant. Il va de soi que ce caractère de délimitation doit être strictement interdit à l'intérieur de chaque champ, faute de quoi la structure devient proprement illisible.

  "Fonfec";"Sophie";0142156487;"fonfec@yahoo.fr"

- **Structure à champs de largeur fixe :** les x premiers caractères de chaque ligne stockent le premier champs, les y suivants le second champs…..Il faut faire attention aux longueurs des champs

  Fonfec Sophie 0142156487 fonfec@yahoo.fr

Les Fichiers

4.4. Type d'accès

- **L'accès séquentiel :** n'accéder à une information qu'en ayant au préalable examiné celle qui la précède. Accès à un enregistrement par enregistrement
- **L'accès direct (ou aléatoire) :** accéder directement à l'enregistrement de son choix, en précisant le numéro de cet enregistrement. Mais cela veut souvent dire une gestion fastidieuse des déplacements dans le fichier.
- **L'accès indexé :** combiner la rapidité de l'accès direct et la simplicité de l'accès séquentiel. Il est particulièrement adapté au traitement des gros fichiers, comme les bases de données importantes.

Les Fichiers

4.5. Ouvrir un fichier (Traitement séquentiel)

Lorsqu'on désire accéder à un fichier, il est nécessaire avant tout accès, d'ouvrir le fichier.

Syntaxe :

Ouvrir Nom_du_Fichier en Num_canal en Mode

Nom_du_Fichier : c'est le nom physique du fichier.
Num_canal : c'est le nom logique du fichier. Pour ouvrir un fichier, il faut lui allouer un numéro du canal valide et disponible.
Mode : le mode d'ouverture du fichier conditionne le travail qui peut être effectué sur ses enregistrements.

Les Fichiers

4.5. Ouvrir un fichier (Traitement séquentiel)

Il existe trois modes d'ouverture du fichier texte :

- **Lecture :** permet d'ouvrir le fichier en lecture seul.
- **Ecriture :** indique son accès en écriture. Dans ce mode, un nouveau fichier est toujours créé. Si le fichier existe déjà, il est réinitialisé à vide et son contenu précédent est perdu.
- **Ajout :** permet d'ajouter des données à un fichier séquentiel existant en conservant le contenu précédent.

Exemple :

```
// Num_canal égal à 1 si fiche.txt est le premier fichier à ouvrir.
// Il est égal à 2 si un fichier est déjà ouvert et fiche.txt est
// le deuxième fichier à ouvrir
Ouvrir "fiche.txt" en 1 en Lecture
```

Cette instruction ouvre le fichier ***fiche.txt*** en lecture seule. Le fichier à comme nom physique ***fiche.txt*** et comme nom logique un (1).

Les Fichiers

4.6. Fermeture un fichier (Traitement séquentiel)

Une fois qu'on a terminé avec un fichier, il ne faut pas oublier de le fermer. On libère ainsi le canal qu'il occupait.

Syntaxe :

Fermer(Nom_du_Fichier)
Ou bien
Fermer (Num_canal)

Note :
Lorsqu'un fichier doit subir plusieurs interventions nécessitant plusieurs ouvertures, il sera nécessaire de fermer le fichier avant de le re-ouvrir.

Les Fichiers

4.7. Lire et écrire dans un fichier

Soit un fichier texte contenant des données organisées dans des enregistrements de longueur fixe.

Champs	Nom	Prénom	Ville	Tél
Enreg 1	Mahboub	Khadija	Agadir	028435867
Enreg 2	Sabir	Ahmed	Marrakech	024251674
Enreg 3	Latif	Karima	Agadir	028457197

La lecture et l'écriture dans ce fichier exigent la déclaration de la structure suivante :

Type

Client : Enregistrement

Nom : Tableau de [1… 25] de caractère

Prenom: Tableau de [1… 25] de caractère

Ville : Tableau de [1… 15] de caractère

Tel : Tableau de [1… 10] de caractère

Fin Enregistrement

Les Fichiers

4.7. Lire et écrire dans un fichier

Après avoir défini la structure client, on peut l'utiliser pour créer une ou plusieurs variables correspondant à cette structure :

```
// Clt : variable de type client
Variables Clt : client
```

On peut maintenant remplir les différentes informations contenues au sein de la variable Clt de la manière suivante :

```
Clt.Nom <- "Mahboub"
Clt.Prénom <- "Khadija "
Clt.Ville <- "Agadir "
Clt.Tel <- "028435867"
```

Les Fichiers

4.7. Lire et écrire dans un fichier

Après avoir rempli les différents champs de la variable Clt, on peut utiliser cette variable pour écrire directement dans le fichier.

Ecrire(Fichier 1, Clt)

Pour une opération de lecture, il suffit de recopier un enregistrement dans une variable de type client et d'écrire la syntaxe suivante :

Lire(Fichier 1, Clt)

Les Fichiers

4.8. Exercice d'application 1

Ecrire un algorithme permettant de créer un fichier **Clients.txt** puis saisir les informations du 1er client et de les écrire dans ce fichier.

Solution :

```
Algorithme Création_Fichier_Clients
Type
Client : Enregistrement
        Nom : Tableau de [1… 25] de caractère
        Prenom: Tableau de [1… 25] de caractère
        Ville : Tableau de [1… 15] de caractère
        Tel : Tableau de [1… 10] de caractère
Fin Enregistrement
// Clt : Variable de type client
Variable Clt : client
Début
// Création du fichier Clients.txt
Ouvrir "Clients.txt" en 1 en Ecriture
// Affectation de données aux champs de la structure
Clt.Nom <- "Mahboub"
Clt.Prénom <- "Khadija "
Clt.Ville <- "Agadir "
Clt.Tel <- "028435867"
// Ecriture dans le fichier
Ecrire (Fichier 1, Clt)
 // Fermeture du fichier
Fermer (1)
Fin
```

Les Fichiers

4.9. Exercice d'application 1

En utilisant une procédure **Affichage()**, écrire un algorithme permettant d'afficher à l'écran le contenu du fichier **Clients.txt**. Cet algorithme complète l'exercice précédent.

Remarque : La fonction booléenne **EOF** (acronyme pour End Of File) renvoie **vrai** si on a atteint la fin du fichier, sinon, elle renvoie **faux**. Cette fonction est nommée parfois **FinFichier**.