الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي و البحث العلمي

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de RELIZANE
Faculté des Sciences et de la Technologie
Département: Informatique

جامعة غليزان
RELIZANE UNIVERSITY

Polycopié de cours intitulé

Informatique 2
Cours et exercices

Préparé et présenté par :
Dr. BELHADEF Leila
Maître de Conférences classe 'B'

Le cours est destiné aux étudiants de License 1ére année Sciences et Technologie

Année universitaire : 2023/2024

Structure du polycopié

Plan pédagogique du cours

Matière : Informatique 2
Domain : Sciences et Technologie
Filière : 1ère année Licence Sciences et Technologie
Volume Horaire du cours : 1h30 par semaine
Volume Horaire du TP : 1h30 par semaine
Coefficient : 2
Crédits : 4
Evaluation : Contrôle continu : 40%, Examen : 60%.

Introduction

Ce cours représente une suite au cours informatique 1 : le cours informatique 1 traite principalement l'apprentissage des notions de base de l'algorithmique et de la programmation ensuite vient le cours informatique 2 pour consolider les notions de base et permettre à l'étudiant d'apprendre de nouveaux concepts ainsi que de nouvelles structures de données ainsi que leur utilisation.

Les TP ont pour objectif d'illustrer les notions enseignées durant le cours. Ces derniers doivent permettre à l'étudiant de maitriser les techniques de base en algorithmique et en programmation et d'apprendre de nouveaux concepts.

Les compétences à acquérir sont : La programmation avec une certaine autonomie ; La conception d'algorithmes du plus simple au relativement complexe.

Plan

Les principaux points traités dans ce cours sont :

- Représentation des données sous forme de tableaux
- Utilisation des fonctions et des procédures dans un programme
- Représentation des données sous forme d'enregistrement
- Utilisation de la notion de fichier

Les principaux points traités dans les TP sont :

- Résolution d'exercices et exécution des programmes correspondants sur ordinateur par les techniques de programmation vues en cours.

Chapitre 1: Les variables Indicées (Les Tableaux)

1 Définition

Un tableau est un ensemble de données du même type représenté par un nom.

Le but d'utiliser les tableaux c'est de pouvoir représenter en mémoire des données qui sont définies par plusieurs valeurs telles que les vecteurs et les matrices. Cette représentation nous permettra de manipuler ces données dans un programme.

2 Les tableaux unidimensionnels

Chaque donnée du tableau est repérée par un indice précisant sa position.

Exemple :

Soit T un tableau unidimensionnels de 5 données :

Valeurs de T:	-2	6	0	3	-4
Indice i :	1	2	3	4	5

Une donnée (élément) du tableau est représentée comme suit : **Nom du tableau** [indice], la donnée à la position 3 est : T[3]=0

2.1 Représentation en mémoire

Dans la mémoire centrale de l'ordinateur (RAM), un tableau est représenté sous forme d'une succession de cases mémoires l'une à la suite de l'autre.

Adresse	Mémoire
0003	-2
0004	6
0005	0
0006	3
0007	-4
	…

2.2 Déclaration

Algorithme :
Var
<nom du tableau> : Tableau [1..Nbr] <type de données> ;
Pascal :
var
<nom du tableau> :**Array** [1..Nbr] **of** <type de données> ;
Nbr : Nombre d'éléments du tableau (ce nombre doit être précisé).

Exemple: On peut déclarer les tableaux avec les déclarations suivantes :

Algorithme :

var

T :Tableau [1..5] d'entier ;

Pascal :

var

T : Array[1..5] **of** integer ;

Ou:

Algorithme :	**Pascal :**
const	**const**
Nbr=5;	Nbr=5;
var	**var**
T :Tableau [1..Nbr] d'entier ;	T : Array[1..Nbr] **of** integer ;

2.3 Operations sur les tableaux unidimensionnels

2.3.1 Lecture et affichage d'un tableau

La lecture et l'affichage d'un tableau doit se faire élément par élément.
Ecrire un programme qui permet de saisir un tableau de N entiers (N<=10), puis d'afficher le contenu de ce tableau.

```
Algorithme lecture_ecriture ;
Var
T : Tableau [1..10] d'entier ;
N,i :entier ;
Début
Ecrire('Donner la taille du tableau : ') ;
Lire (N) ;
Ecrire(' Donner les valeurs du tableau: ') ;
Pour i← 1 à N faire
Lire(T[i]) ;
Finpour ;
Ecrire(' Afficher le tableau: ') ;
Pour i← 1 à N faire
Ecrire(T[i]) ;
Finpour ;
Fin.
```

```
program lecture_ecriture ;
uses wincrt;
var
T:Array[1..10] of integer;
N, i:integer;
begin
write(' Donner la taille du tableau: ');
read (N);
writeln('Donner les valeurs du tableau: ');
for i:=1 to N do
Read (T[i]);
writeln(' Afficher le tableau: ');
for i:=1 to N do
Writeln (T[i]);
end.
```

Exemple d'exécution :
avec N=4 et T=[5,-2,3,0] :
Lorsque on déclare T:**Array**[1..10] **of** integer; cela signifie que le nombre d'élément du tableau est maximum 10.
Ensuite on précise le nombre d'élément que nous allons utiliser dans notre exemple :

```
write(' Donner la taille du tableau:');
read (N);
```

On saisit 4 pour que N=4
Les deux instructions suivantes ont le but d'insérer dans la mémoire les éléments du tableau :

```
for i:=1 to N do
Read (T[i]);
```

élément par élément : i :=1 on insère 5 donc T[1]=5
i :=2 on insère -2 donc T[2]=-2
i :=3 on insère 3 donc T[3]=3
i :=4 on insère 0 donc T[4]=0

De même pour les instructions :

```
for i:=1 to N do
Writeln (T[i]);
```

Ont le but d'afficher dans l'écran les éléments du tableau.

Ecran

```
Donner la taille du tableau: 4
Donner les valeurs du tableau:
5
-2
3
0
Afficher le tableau:
5
-2
3
0
```

2.3.2 La somme de deux Vecteurs

Ecrire le programme qui permet d'additionner deux vecteurs de réels.

```
Algorithme Somme;
Var
V1,V2, V:Tableau[1..100] de réel;
N, i:entier;
Début
Ecrire('Donner la taille des vecteurs') ;
Lire (N) ;
pour i← 1 à N faire
Lire (V1[i]) ;
FinPour ;

pour i← 1 à N faire
Lire (V2[i]) ;
FinPour ;

pour i← 1 à N faire
V[i] ← V1[i] + V2[i];
Écrire (V[i]) ;
FinPour ;
Fin.
```

```
Program Somme;
uses Wincrt;
var
V1, V2, V: Array[1..100] of real;
N, i : integer;
begin
write ('Donner la taille des vecteurs');
read (N);
for i:=1 to N do
read (V1[i]);

for i:=1 to N do
read (V2[i]);

for i:=1 to N do
begin
V[i]:=V1[i]+V2[i] ;
writeln(V[i]) ;
end;
end.
```

Après avoir précisé le nombre d'élément par vecteur N, on insère les éléments de V1:

```
for i:=1 to N do
read (V1[i]);
```

Ensuite on insère les éléments de V2 :

```
for i:=1 to N do
read (V2[i]);
```

Après vient l'addition de chaque éléments de V1 et V2 dans V et l'affichage des résultats:

```
for i:=1 to N do
begin
V[i]:=V1[i]+V2[i] ;
writeln(V[i]) ;
end;
```

3 Les tableaux bidimensionnels

Un tableau bidimensionnel se compose de lignes et de colonnes. Chaque élément du tableau se trouve à l'intersection d'une ligne et d'une colonne.

Exemple : soit A une matrice de deux lignes et 3 colonnes :

i	j	1	2	3
1		10	-7	3
2		2	6	0

Une donnée du tableau est représentée comme suit : **Nom du tableau** [ligne, colonne], la donnée à l'intersection de la ligne 1 avec la colonne 3 est A[1,3]=3.

3.1 Déclaration

Algorithme :
Var
<nom du tableau> : Tableau [1..N,1..M]<type de données> ;
Pascal :
Var
<nom du tableau> :**Array** [1..N,1..M] **of** <type de données> ;
N : nombre de lignes. **M** : nombre de colonnes.

Exemple :

var
A :Tableau [1..2,1..3] d'entier ;

var
A : **Array**[1..2,1..3] **of** integer ;

Ou bien on peut déclarer le nombre de lignes et de colonnes comme constantes (section 2.2).

3.2 Opérations sur les tableaux bidimensionnels

3.2.1 Lecture et affichage d'un tableau bidimensionnel

Ecrire le programme qui permet de lire et d'afficher les éléments d'une matrice de réels avec n lignes et m colonnes (n<=10 et m<=20)

Algorithme LectureAffichageMatrice ;
Var
A:**tableau** [1..10, 1..20] **de** réel ;
N,M, i,j:entier ;
Début
Ecrire ('Donner le nombre de lignes et de colonnes') ;
Lire (N, M);
Pour i← 1 **à** N **faire**
Pour j← 1 **à** M **faire**
Lire (A[i, j]) ;
FinPour ;
FinPour ;
Ecrire(' Afficher le tableau: ') ;
Pour i← 1 **à** N **faire**
Pour j← 1 **à** M **faire**
Écrire (A[i, j]) ;
FinPour ;
FinPour ;
Fin.

Pascal
Program LectureAffichageMatrice;
uses wincrt;
var
A:**array** [1..10, 1..20] **of** real;
N,M, i,j : integer;
begin
write ('Donner le nombre de lignes et de colonnes') ;
read(N, M);
for i:=1 **to** N **do**
for j:=1 **to** M **do**
Read(A[i, j]);
writeln (' Afficher le tableau: ') ;
for i:=1 **to** N **do**
for j:=1 **to** M **do**
writeln(A[i,j]);
end.

Exemple d'exécution :

avec N=2, M= 3 et A=$\begin{pmatrix} 0.5 & 3 & 12 \\ -1 & 5 & 0 \end{pmatrix}$

On précise le nombre de lignes et de colonnes que nous allons utiliser dans notre exemple :
write ('Donner le nombre de lignes et de colonnes') ;
read(N, M);
On saisit 2 pour que N=2 et 3 pour que M=3

Les instructions suivantes ont le but d'insérer dans la mémoire les éléments du tableau :
for i:=1 **to** N **do**
for j:=1 **to** M **do**
Read(A[i, j]);

Ligne par ligne : donc on insère les éléments de toute la ligne avant de passer à la suivante :
On commence par la première ligne : i :=1 et j=1 : on insère 0.5 donc A[1,1]=0.5
i :=1 et j=2 : on insère 3 donc A[1,2]=3
i :=1 et j=3 : on insère 12 donc A[1,3]=12
On continu avec la deuxième ligne : i :=2 et j=1 : on insère -1 donc A[2,1]=-1
i :=2 et j=2 : on insère 5 donc A[2,2]=5
i :=2 et j=3 : on insère 0 donc A[2,3]=0

De même pour les instructions :
for i:=1 **to** N **do**
for j:=1 **to** M **do**
writeln(A[i,j]);
Ont le but d'afficher dans l'écran les éléments du tableau A ligne par ligne.

Ecran

```
Donner le nombre de lignes et de colonnes : 2
3
0.5
3
12
1
5
0
Afficher le tableau:
0.5
3
12
1
5
0
```

3.2.2 Additionner deux matrices carrées

Ecrire le programme qui permet de calculer la somme de deux matrices carrées et d'afficher le résultat.

Solution :

```
Program LectureAffichageMatrice;
uses wincrt;
var
A,B,C: array [1..10, 1..10] of real;
N, i,j : integer;
begin
Write ('Donner le nombre de lignes') ;
read(N);
for i:=1 to N do
for j:=1 to N do
read(A[i, j]);

for i:=1 to N do
for j:=1 to N do
read(B[i, j]);

for i:=1 to N do
for j:=1 to N do
begin
C[i,j]:=A[i,j]+B[i,j];
Writeln(C[i,j]);
end;
end.
```

Après avoir précisé le nombre de lignes N qui est le même que le nombre de colonnes, on insère les éléments de A:

```
for i:=1 to N do
for j:=1 to N do
read(A[i, j]);
```

Ensuite on insère les éléments de B :

```
for i:=1 to N do
for j:=1 to N do
read(B[i, j]);
```

Après vient l'addition des éléments des matrices A et B dans C et l'affichage des résultats:

```
for i:=1 to N do
for j:=1 to N do
begin
C[i,j]:=A[i,j]+B[i,j];
Writeln(C[i,j]);
end;
```

Remarque:
Pour afficher les éléments d'un tableau il faut utiliser obligatoirement l'instruction **writeln**.

4 Exercices

Exercice 1
Ecrire un programme qui permet de trouver la valeur maximale dans un tableau de réels ainsi que sa position.

Exercice 2
Ecrire un programme qui permet de faire le produit scalaire de deux vecteurs V1 et V2.

Exercice 3
Soit T un tableau de N entiers. Ecrire un programme qui permet de calculer le nombre des occurrences d'un nombre X déterminé par l'utilisateur (c'est-à-dire combien de fois ce nombre X figure dans le tableau T).

Exercice 4
Soit T un tableau de réel. Ecrire un programme qui permet de trier les éléments du tableau T de manière croissante.

Exercice 5
Dérouler le programme suivant :

```
program exo ;
uses wincrt;
Var
 val,i,j :integer ;
mat :array [1..2,1..3] of integer;
begin
val:=1;
for i := 1 to 2 do
for j := 1 to 3 do
begin
mat[i,j]:=val;
val:=val+1;
end;
for i := 1 to 2 do
for j := 1 to 3 do
writeln (mat[i,j] );
end.
```

Série d'exercices avec solution

Exercice 6
Ecrire un programme qui permet de :

- Saisir la taille d'un tableau n (n<=4).
- Saisir un tableau T d'entier.
- Augmenter toutes les valeurs du tableau de 1.
- Afficher le nouveau tableau à l'écran.

Solution :

```
program exo1 ;
uses wincrt;
Var
 N, i :integer ;
T :array [1..4] of integer;
begin
write('Entrer le nombre de valeurs : ') ;
read(N) ;
for i := 1 to N do
  read (T[i] );
writeln ('Nouveau tableau : ') ;
for i := 1 to N do
begin
  T[i] := T[i] + 1 ;
  writeln (T[i] );
end;
end.
```

Modifier le programme comme suit pour avoir un affichage détaillé :

```
program exo 1 ;
uses wincrt;
var
T:array[1..4] of integer;
N, i:integer;
begin
write(' Donner la taille du tableau:');
read (N);
for i:=1 to N do
begin
  write ('Entrer T[',i,']=') ;
  read (T[i] );
end;
writeln ('Nouveau tableau : ') ;
for i:=1 to N do
  begin
  T[i] := T[i] + 1 ;
  writeln ('T[',i,']=',T[i] );
  end;
end.
```

Exercice 7
Tracer l'exécution du programme suivant :

```
program examen ;
uses wincrt;
var
  i, val :integer ;
T :array [1..4] of integer;
begin
  T[1] := 5;
  val:= 2;
     for i := 2 to 4 do
     begin
     if (i mod 2=0) then
        T[i] :=  T[i-1]
     else
        T[i] :=  T[i-1] + val;
     writeln ('T[',i,']=',T[i] );
     end;
end.
```

Solution :

i	**T**	**val**	**écran**
/	T[1]=5	2	/
2	T[2]=T[1] T[2]=5	2	T[2]=5
3	T[3]=T[2]+val T[3]=5+2=7	2	T[3]=7
4	T[4]=T[3] T[4]=7	2	T[4]=7

Exercice 8
Soit V un tableau d'entiers. Ecrire un programme permettant de construire deux tableaux VP et VI, avec VP contient les éléments pairs de V et VI contient les éléments impairs de V. Afficher les deux tableaux VP et VI.

Solution :

```
program pair_impair;
uses wincrt;
var
V, VP, VI : Array [1..100] of integer;
n ,i,j,k: integer;
begin
write('Donner la taille des tableaux');
read(n);
for i:= 1 to n do
read(V[i]);
j:=1;
k:=1;

for i:= 1 to n do
begin
    if(V[i] mod 2=0) then
    begin
      VP[j]:=V[i];
       j:=j+1;
    end
    else
    begin
       VI[k]:=V[i];
       k:=k+1;
    end;
end;

for  i:=1  to j-1 do
writeln ('VP[',i,']=',VP[i] );

for i:= 1 to k-1 do
writeln ('VI[',i,']=',VI[i] );
end.
```

Exemple d'exécution:

pour n=4 et V= [4,8,5,2]

Après lecture de la valeur de n et des valeurs de V, les indices des deux tableaux VP et VI sont initialisés j=1 et k=1 respectivement, vient par la suite, la boucle pour construire VP et VI :

n=4 donc la taille des tableaux sera au maximum 4 :

V

4	8	5	2
i=1	i=2	i=3	i=4

VP

j=1			

VI

k=1			

pour i=1:

V

4	8	5	2
i=1	i=2	i=3	i=4

VP

4			
j=1	j=2		

VI

k=1			

V[1] mod 2=0:
VP[1]=V[1]=4
j=j+1=2

pour i=2:

V

4	8	5	2
i=1	i=2	i=3	i=4

VP

4	8		
j=1	j=2	j=3	

VI

k=1			

V[2] mod 2=0:
VP[2]=V[2]=8
j=j+1=3

pour i=3:

V

4	8	5	2
i=1	i=2	i=3	i=4

VP

4	8		
j=1	j=2	j=3	

VI

5			
k=1	k=2		

V[3] mod 2≠0:
VI[1]=V[3]=5
k=k+1=2

pour i=4:

V

4	8	5	2
i=1	i=2	i=3	i=4

VP

4	8	2	
j=1	j=2	j=3	j=4

VI

5			
k=1	k=2		

V[4] mod 2=0:
VP[3]=V[4]=2
j=j+1=4

Affichage des tableaux VP et VI par les instructions :

```
for  i:=1  to j-1 do
writeln ('VP[',i,']=',VP[i] );
for i:= 1 to k-1 do
writeln ('VI[',i,']=',VI[i] );
```

Le nombre d'éléments dans VP=j-1=3
Le nombre d'éléments dans VI=k-1=1

Ecran

```
VP[1]=4
VP[2]=8
VP[3]=2
VI[1]=5
```

Exercice 9
Soit Mat matrice carrée de réels. Ecrire un programme qui détermine la somme des éléments de la diagonale de Mat.

Solution :

```
program diagonale;
Uses wincrt;
Var
Mat: Array [1..10,1..10] of real;
n,i,j: integer;
som:real;
begin
write('Donner le nombre de lignes :  ');
read(n) ;

for i := 1 to n do
for j := 1 to n do
read(Mat[i,j]);
Som:=0;

for i := 1 to n do
Som:= Som+Mat[i,i];

writeln ('Somme=', Som);
end.
```

Exemple d'exécution:
Pour n=3 et $Mat = \begin{pmatrix} 2 & 5 & 10 \\ 0.2 & 9 & -3 \\ 8 & 4 & 7 \end{pmatrix}$
Initialisation de Som=0,
pour i=1 : Som= Som+Mat[1,1]=0+2=2
pour i=2: Som= Som+Mat[2,2]=2+9=11
pour i=3: Som= Som+Mat[3,3]=11+7=18

Exercice 10
Soit Mat matrice de réels à deux dimensions n et m (dimensions maximales: 10 lignes et 20 colonnes). Ecrire un programme qui détermine les valeurs maximales de chaque lignes et les mettre dans un vecteur tel que :

$$\begin{pmatrix} 4 & -15.5 & 17.2 \\ 0 & 2 & 10 \\ 6 & 3.75 & 0.3 \end{pmatrix} \Longrightarrow \begin{pmatrix} 17.2 \\ 10 \\ 6 \end{pmatrix}$$

Solution :

```
program matrice;
Uses wincrt;
Var
V : Array [1..10] of real;
Mat: Array [1..10,1..20] of real;
n,m ,i,j: integer;
max:real;
begin
write('Donner le nombre de lignes et de colonnes ');
read(n,m) ;

for i := 1 to n do
for j := 1 to m do
read(Mat[i,j]);

for i := 1 to n do
begin
  Max:=Mat[i,1];
   for j := 2 to m do
  begin
    if(Mat[i,j]>Max)then
    Max:=Mat[i,j];
   end;
V[i]:=Max;
end;

for i := 1 to n do
writeln ('V[',i,']=',V[i] );
end.
```

Chapitre 2: Les fonctions et procédures (les sous-programmes)

1 Définition

Les sous-programmes représentent une partie de programme qui réalise un calcul ou un affichage leur utilisation permet d'intégrer une fonctionnalité au programme principal ou bien de réutiliser une partie de programme sans la réécrire, la déclaration d'un sous-programme se fait dans la partie déclaration du programme principale.

Rappel sur la structure d'un programme :

```
Program NomDuProgramme;
uses wincrt ;
   <Déclarations des constantes> ;
   <Déclarations des variables> ;
   <Déclarations des fonctions> ;
   <Déclarations des procédures> ;
 begin
    <Les instructions> ;
 end.
```

2 Les fonctions

Une fonction permet de recevoir les paramètres qui représentent les données en entrée.

Elle renvoie un résultat de type standard (réel, entier, caractère, chaîne de caractères ou booléen) qui représente la donnée en sortie.

2.1 Déclaration des fonctions

La structure générale d'une fonction :

```
Function Nomfonction (liste des paramètres) : type de données;
const
(* déclaration des constantes locales*)
Var
(* déclaration des variables locales*)
Begin
Instruction 1 ;
Instruction 2 ;
…..
Nomfonction := résultat ;
End ;
```

2.2 Appel de fonction

L'appel de fonction représente l'utilisation de celle-ci par le programme.

Exemple1 :

En utilisant une fonction écrire le programme qui calcule A*B, où A et B sont des entiers :

1- P=A*B ;
   - P est le résultat donc P peut être le nom de cette fonction,
2- Déclaration de la fonction :

```
Function P (A,B : integer) : integer ;
Begin
P:=A*B;
End;
```

3- Le programme

```
Program produit;
Uses wincrt;
Var
A1, B1, P1: integer;

Function P (A,B : integer) : integer ;
Begin
P:=A*B;
End;

Begin
{Début du programme donc on utilise les variables  A1, B1, P1}
Read(A1,B1) ;
P1 := P(A1,B1) ;
Write(P1);
End.
```

Déroulement:

1- Read(A1, B1) par exemple A1=3 et B1= 2
2- P1 := P(A1,B1) : Appel de fonction P avec A reçoit la valeur de A1 et B reçoit la valeur de B1, la fonction est exécutée est retourne le résultat dans P1.

Exemple2 :

Faire la somme des N premiers nombres positifs : S=1+2+3+…+N

1- Le corps du sous-programme

```
Som :=0 ;
for i :=1 to N Do
Som :=Som+i ;
```

- N la donnée en entrée fait partie des paramètres de la fonction.
- S est le résultat donc S peut être le nom de cette fonction
- La variable i prend des valeurs de 1 à N donc i est une variable locale du sous programme.

2- Déclaration de la fonction :

```
Function S (N : integer) :integer ;
Var
Som, i:integer;
begin
Som :=0;
For i :=1 to N Do
Som :=Som+i ;
S:=Som;
end;
```

3- Le programme:

```
Program somme;
Uses wincrt;
Var
N1,S1: integer;
    Function S (N : integer) :integer ;
    Var
    i,Som:integer;
    begin
    Som :=0;
    for i :=1 to N Do
    Som :=Som+i ;
    S:=Som;
    end;
begin
Read(N1);
S1:=S(N1);
write(S1);
end.
```

3 Les procédures

Une procédure peut recevoir en entrée les paramètres (ils ne sont pas obligatoires).
Elle renvoie aucun, un ou plusieurs résultats (données en sortie).
Une procédure qui effectue une action comme un affichage elle ne retourne aucun résultat.

3.1 Déclaration des procédures

La structure générale d'une procédure :

procedure NomProcedure (liste des paramètres) ;
const
(* déclaration des constantes locales*)
var
(* déclaration des variables locales*)
begin
Instruction 1 ;
Instruction 2 ;
…..
end ;

2.2 Appels de procédure

Comme l'appel de fonction l'appel de procédure représente l'utilisation de celle-ci par le programme.

Exemple1 :
En utilisant une procédure écrire le programme qui calcule A*B, ou A et B sont des entiers :

1- P=A*B ;
- A, B les données en entrées, P est le résultat.

2- Déclaration de la procédure :

Procedure Prod **(A,B :** integer **; var P:**integer **) ;**
begin
P:=A*B;
end;

3- Le programme

```
Program produit;
Uses wincrt;
Var
A1, B1, P1: integer;
Procedure  Prod (A,B : integer; var P:integer ) ;
Begin
P:=A*B;
End;
Begin
{Début du programme donc on utilise les variables  A1, B1, P1}
Read(A1,B1) ;
Prod (A1,B1,P1) ;
Write(P1);
End.
```

Déroulement:

1- Read(A1, B1) par exemple A1=3 et B1= 2

2- Prod (A1,B1,P1) ; : Appel de procédure Prod avec A reçoit la valeur de A1 et B reçoit la valeur de B1 on dit que A1 et B1 sont transmis par valeurs, et P prend l'adresse de la variable P1, on dit que P1 est transmit par adresse, donc le résultat qui se trouve dans P sera transmit à l'adresse P1.

Exemple2 :

Faire la somme des N premiers nombres positifs

1- Le corps du sous programme

```
S :=0 ;
For i :=1 to N Do
S :=S+i ;
```

- N la donnée en entrée fait partie des paramètres de la procédure.
- S est le résultat.
- La variable i est une variable locale du sous programme.

2- Déclaration de la procedure

```
Procedure Som (N : integer ; var S:integer) ;
Var
i:integer;
begin
  S :=0;
  For i :=1 to N Do
  S :=S+i ;
  End;
```

3- Le programme:

```
Program somme;
Uses wincrt;
Var
N1,S1: integer;
Procedure Som (N : integer ; var S :integer)  ;
Var
i:integer;
begin
  S :=0;
  For i :=1 to N Do
  S :=S+i ;
  End;
Begin
Read(N1);
Som(N1,S1);
Write(S1);
End.
```

3.3 Notions de variables globales et de variables locales

1- Les variables et constantes définis dans les déclarations locales de la procédure (fonction) ne sont utilisés que dans la procédure (fonction).

2- Les variables déclarées dans le **var** du programme principal sont appelées variables globales. Elles peuvent être utilisées partout dans le programme durant son exécution.

3- Si une variable locale à une procédure P a le même nom qu'une variable globale, cette variable globale sera masquée durant l'exécution de P.

Exemple :
Soit le programme suivant :

```
Program  Exo;
Uses wincrt ;
Var
A :integer ;
X, Y : integer ;
    Procedure P1 (B:integer; var S: integer);
    Var
    A:integer;
    Begin
    A:=4;
    B:=B*2;
    S:=A+B;
    End;
Begin
Read(x);
A:=6;
P1(X,Y);
Write(X, ' ',Y, ' ',A) ;
End .
```

1) Dérouler le programme pour X=3.
2) Réécrire le programme en utilisant une fonction à la place de la procédure.

Solution : 1) Déroulement du programme pour X=3.

Ce tableau illustre le principe de variable globale et de variable locale (la variable A).

X	Y	A (programme)	P1 (appel de procédure)	B	S	A (procédure)	Ecran
3	/	6	/	/	/	/	/
3	/	6	P1(3,Y)	3	/	/	/
3	10	6	/	6	10	4	3 10 6

2) Réécrire le programme en utilisant une fonction :

On peut modifier toutes les fonctions en procédures mais le contraire n'est pas toujours vrai. En effet la fonction ne retourne qu'un seul résultat de type standard, vu que la procédure P1 retourne un seul résultat (**var** S: integer) S entier donc on peut la modifier en une fonction.

```
Program  Exo ;
Uses wincrt ;
Var
A :integer ;
X, Y : integer ;
    function P1 (B:integer):integer;
    Var
    S, A:integer;
    Begin
    A:=4;
    B:=B*2;
    S:=A+B;
    P1:= S;
    End;
Begin
Read(x);
A:=6;
Y:=P1(X);
Write(X, ' ',Y, ' ',A) ;
end .
```

3.4 Procédure simple

C'est une procédure qui n'a pas de paramètres, elle consiste à donner un nom à un groupe d'instructions.

Exemple :

La procédure affiche est une procédure simple qui affiche à chaque fois qu'on l'appelle le message 'Etudiants ST'.

```
Program affichage;
Uses wincrt;
Procedure affiche ;
begin
write('Etudiants ST');
end;
begin
affiche;
end.
```

4 Utilisation des tableaux comme paramètre

Pour utiliser les tableaux comme paramètre dans une fonction ou procédure il faut utiliser la notion de **type** afin de déclarer un type tableau et l'utilisé dans la déclaration des fonctions ou procédure.

Rappel sur la déclaration de tableau :
Var
T : **array** [1..100]**of** integer;

Afin d'utiliser un tableau comme paramètre il faut le déclarer comme suit :
Type
Tab = **array** [1..100]**of** integer;
Var
T : tab ;

Exemple 1 :

1- Ecrire une fonction qui permet de faire le produit scalaire de deux vecteurs.
2- Ecrire le programme qui appel cette fonction et affiche le résultat.
3- Réécrire ce programme en utilisant une procédure (qui calcule le produit scalaire).

1- Ce que doit réaliser la fonction :

```
Prod:=0;
for i:=1 to N do
Prod:= Prod + R1[i]*R2[i] ;
```

- R1 et R2 les données en entrée c'est les paramètres de la fonction (deux tableaux), Prod est une variable locale.
- **P** est le nom de cette fonction

2- Le programme

```
Program ProduitScalaire;
Uses Wincrt;
Type
Tab = Array[1..100] of real;

Var
V1, V2 : Tab;
N, i : integer;
Ps : real;
```

```
function P (R1,R2: Tab):real;
var
i:integer;
Prod:real;
begin
Prod:=0;
for i:=1 to N do
Prod:= Prod + R1[i]*R2[i] ;
P:=Prod;
end;
begin  {début du programme}
Read (N);
for i:=1 to N do
readln (V1[i]);
for i:=1 to N do
readln (V2[i]);
Ps:=P(V1,V2);
Write('Le produit scalaire = ', Ps);
end.
```

3- Solution avec procedure:

```
Program ProduitScalaire;
Uses Wincrt;
Type
Tab = Array[1..100] of real;
Var
V1, V2 : Tab;
N, i : integer;
Ps : real;

procedure P (R1,R2: Tab; var Prod:real);
var
i:integer;
begin
Prod:=0;
for i:=1 to N do
Prod:= Prod + R1[i]*R2[i] ;
end;
```

```
begin {début du programme}
Read (N);
for i:=1 to N do
readln (V1[i]);
for i:=1 to N do
readln (V2[i]);
P(V1,V2,Ps);
Write('Le produit scalaire = ', Ps);
end.
```

Exemple2 :

La somme de deux Vecteurs :

1- Ecrire une procédure qui permet d'additionner deux vecteurs de réels.
2- Ecrire le programme qui appel cette procédure et affiche le résultat.
3- Réécrire ce programme en utilisant une fonction.

1- Déclaration de la procédure :

```
Procedure add (V1,V2: Tab;  var V: Tab);
Var
i:integer;
begin
for i:=1 to N do
V[i]:=V1[i]+V2[i] ;
end;
```

2- Le programme

```
Program Somme;
Uses Wincrt;
Type
Tab =Array[1..100] of real;
Var
N, i : integer;
V1, V2, V : Tab;
Procedure add (V1,V2: Tab;  var V: Tab);
Var
i:integer;
begin
for i:=1 to N do
V[i]:=V1[i]+V2[i] ;
End;
```

```
Begin
Write ('Donner la taille des vecteurs');
Read (N);
for i:=1 to N do
read (V1[i]);
for i:=1 to N do
read (V2[i]);
add(V1,V2,V);
for i:=1 to N do
Writeln(V[i]) ;
End.
```

Remarques:

- La variable N est une variable globale utilisée par la procédure et le programme.
- En donnant le même nom aux variables globales et aux paramètres de la procédure cela peut rendre le programme pas clair mais il reste correct car les paramètres (V1,V2) de la procédure ne prennent que les valeurs des tableaux du programme lors de l'appel de procédure et à la fin des calculs les valeurs du paramètre V de la procédure vont être transmis à l'adresse V du programme.

3- On ne peut pas transformer cette procédure en fonction car elle retourne un type non standard (un tableau).

5 La récursivité

Une fonction (ou procédure) peut s'appeler elle-même: on dit que c'est une fonction (ou procédure) récursive.

Exemple

En utilisant un sous-programme (fonction ou procédure) écrire le programme qui permet de calculer la factorielle d'un entier positif.

La définition récurrente de la factorielle de n :

$$n! = \begin{cases} 1 & si\ n \leq 1 \\ n \times (n-1)! & si\ n > 1 \end{cases}$$

Solution avec une fonction:

```
Program factorielle ;
Uses wincrt ;
Var
N1,F1 :integer ;

Function Factor(N:integer):integer;

begin
if (N<=1) then
Factor:= 1
else
Factor:=N* Factor(N-1);
end;

begin
write('Donner un entier : ') ;
read(N1);
F1:= Factor(N1);
writeln('Factorielle=', F1);
end.
```

Solution avec une procédure:

```
Program factorielle ;
Uses wincrt ;
Var
N1,F1 :integer ;
Procedure Factor(N:integer; Var F:integer);
begin
if (N<=1) then
F:= 1
else
begin
 Factor(N-1,F);
 F:=F*N;
  end;
end;

begin
write('Donner un entier : ') ;
read(N1);
Factor(N1,F1);
writeln('Factorielle=', F1);
end.
```

Remarques:

Toute fonction (ou procédure) récursive doit posséder un cas limite qui arrête la récursivité.

Le processus récursif remplace la boucle à l'envers: on part du nombre n, et on remonte jusqu'à 1, pour pouvoir calculer la factorielle par exemple.

6 Exercices

Exercice 1

```
Program test;
Uses wincrt;
Var
N1,N2,N3, S: integer;
    Function F (N : integer) :integer ;
    Var
    i,som:integer;
    begin
    Som :=0;
    for i :=1 to N Do
    Som :=Som+i ;
    F:=Som;
    end;
begin
write(' Donner trois entiers') ;
Read(N1,N2,N3);
S:=F(N1) +F(N2)+F(N3);
write('S=', S);
end.
```

1) Quel est le résultat du programme pour N1=5, N2=3 et N3=1.
2) Quel est l'avantage d'utiliser la fonction ?
3) Quelles sont les variables locale de **F** ?
4) Quelles sont les paramètres de **F** ?
5) Quelles sont les variables globales du programme ?

Exercice 2

En utilisant une fonction (ou procédure) écrire un programme qui permet de calculer la somme suivante:

$$s = 1 + \frac{x^2}{2} + \frac{x^4}{2^2} + \frac{x^6}{2^3} + \cdots + \frac{x^{2n}}{2^n}$$

Exercice 3

1. Ecrire une fonction qui permet de calculer la somme des éléments pairs d'un tableau de n entiers ($n \leq 20$).
2. Ecrire le programme qui appel cette fonction.

Exercice 4

1. Ecrire une fonction qui permet de calculer le nombre des occurrences d'un nombre X déterminé par l'utilisateur dans un tableau d'entiers.
2. Ecrire le programme qui appel cette fonction.

Série d'exercices avec solution

Exercice 5

En utilisant un sous programme (fonction ou procédure) écrire le programme qui permet de trouver le maximum entre trois entiers.

Solution avec une fonction:

```
Program maximum ;
Uses wincrt ;
var
A, B,C , Max :integer ;

Function M(x,y,z :integer) :integer;
Begin
if (x>y)and (x>z) then M:=x
        else if (y>z) then M:=y
        else M:=z;
end;

Begin
write('Donner trois entiers : ') ;
Read(A,B,C);
Max:=M(A,B,C);
Write('Max=',Max);
End.
```

Solution avec une procédure:

```
Program maximum;
Uses wincrt ;
var
A, B,C , Max :integer ;

Procedure Maxi(x,y,z :integer; Var M:integer) ;
Begin
if (x>y)and (x>z) then M:=x
        else if (y>z) then M:=y
        else M:=z;
end;

Begin
write('Donner trois entiers : ') ;
Read(A,B,C);
Maxi(A,B,C,Max);
Write('Max=',Max);
End.
```

Exercice 6

Donner la déclaration d'une fonction récursive qui permet de calculer X^Y (X et Y entiers strictement positifs).

```
Function puissance (X,Y:integer): integer;
begin
if (Y<=1) then
puissance:= X
else
puissance:=X* puissance(X,Y-1);
end;
```

Exercice 6

1- Ecrire une fonction **Som** qui permet de calculer la somme des valeurs d'une matrice.
2- Ecrire une procédure **Max** qui permet de trouver la valeur max dans une matrice et sa position.
3- En utilisant les sous programmes **Som** et **Max** écrire le programme qui permet de :
   a- Lire une matrice **Mat**
   b- Afficher la somme des valeurs de **Mat**
   c- Afficher la valeur maximum de **Mat** et sa position.

Solution :

```
program som_max;
uses wincrt;
type
matrice=array[1..10,1..10]of real;
var
Mat1: matrice;
n,m,i,j,posi1,posj1:integer;
som1,max1:real;
(*la fonction Som*)
function Som (Mat: matrice):real;
var  S:real;
begin
….
Som:=S;
end;
(* la procedure Max *)
procedure Max (Mat: matrice; var posi, posj:integer; var maxi: real);
begin
…..
end;
begin  (*le programme*)
        writeln('Donner la valeur le nombre de lignes et de colonnes');
        readln(n,m);
        for i:=1 to n do
        for j:=1 to m do
        readln(mat1[i,j]);
        …….. (*appel de fonction Som*)
        ……..(*appel de procédure Max*)
        writeln('somme=',som1);
        writeln('max= m[',posi1,',',posj1,']=',max1);
end.
```

Chapitre 3: Les enregistrements et fichiers

1 Introduction

Les variables standards ne sont pas suffisantes pour représenter toutes les structures de données ainsi que les tableaux qui représentent un ensemble de données du même type, d'où la nécessité de pouvoir utiliser de nouveaux types.

2 Structure de données non standards

2.1 Types énumérés

Les types énumérés définissent un ensemble ordonné de valeurs.

Syntaxe :

type

Nomtype = (élément1, . . . , élémentN) ;

Exemple :

type

mois = (jan, fev, mar, avr, mai ,jun, jui, aou, sep, oct, nov, dec **);**

2.1.1 Déclaration :

var m1, m2 : mois;

2.1.2 Manipulation des types énumérés :

1- Affectation:

m1 := mai ;

m2 := m1;

2- Fonctions et structures: Vu que les types énumérés sont ordonnés cela permet de les utiliser par des fonctions standards, dans des boucles ou dans une sélection à choix multiples :

Fonctions: Pred (précédent), Succ (suivant), Ord (numéro d'ordre dans la déclaration à partir de 0) :

pred(m1) = avr

ord(m1) = 4

Boucles:

for m1 := sep **to** jui **do** ...

Sélection à choix multiples:

case m1 **of**

jan : write('janvier');

fev : ...

end ;

2.2 Types intervalles

Un type intervalle est une partie de l'intervalle des valeurs d'un type scalaire.

Syntaxe :
type
intervalle = $val_1..val_n$;

Exemple :
type
mois = 1..12;
num_wilaya=1..58 ;
lettre_min='a'.. 'z' ;

2.2.1 Déclaration :

var
m1 :mois ; (*m1 est de type mois*)
w1,w2 : num_wilaya ; (*w1 et w2 sont de type num_wilaya *)

Ou bien on peut les déclarés directement dans var sans utiliser la déclaration de type :
var
m1 : 1..12;
w1,w2 : 1..58 ;

3 Structure d'un enregistrement (notion de champs)

Le type enregistrement (record en pascal) est composé d'un ensemble d'objet ou champs qui peuvent être de différents types.

Syntaxe :
type enregistrement = **record**
champ1 : type1 ;
champ2 : type 2;
….
end ;

Exemple :
Déclarez en pascal les enregistrements avec les descriptions suivantes :
1- une voiture est caractérisée par sa marque, sa couleur, sa puissance.

```
type voiture = record
marque : string[30] ;
couleur: string[15];
puissance :real ;
end ;
 var V1: voiture;
```

2- une matière est caractérisée par son nom, son coefficient et le nombre d'heure par semaine.

```
type matiere = record
nom : string[30] ;
coef: integer;
nbr_heure :real ;
end ;
var
M1 :matiere ;
```

3.1 Manipulation des structures d'enregistrements

1- Accès au champ de l'enregistrement :

Un champ d'une variable de type enregistrement est désigné par le nom de la variable, suivi d'un point et du nom du champ concerné.

Exemple :

M1.nom

2- Affectation

M1.nom := 'informatique2' ;

3- Lecture

Write ('Donner le nom de la matière') ;

Readln(M1.nom) ;

La lecture est réalisée obligatoirement en utilisant le readln pour que tous les champs soient lus correctement.

4- Afficher

Writeln(M1.nom)

Exemple :

Déclarer en pascal un enregistrement **Etudiant** qui représente les informations d'un étudiant : nom, prénom, date de naissance, lieu de naissance, numéro de groupe, section.

Solution :

La date de naissance représente trois données donc on doit créer un enregistrement qui contient trois champs : jours, mois et année. Ce dernier sera par la suite utilisé afin de représenter le champ date de naissance dans l'enregistrement **Etudiant**.

Déclaration des types enregistrements **date** et **Etudiant** :

```
type
date = record
annee : 1970..2015 ;
mois : 1..12 ;
jour : 1..31 ;
end ;
Etudiant = record
Nom, Prenom: string[30] ;
datenaiss : date ;
lieunaiss : string[20] ;
num_groupe : integer ;
Section : char ;
end ;
```

1) Créer par affectation, l'enregistrement **etudiant1** avec :
   Nom: Omar
   Prénom: Youssef
   Date de naissance: 06/04/2004
   Lieu de naissance: Relizane
   Numéro de groupe: 5
   Section: A
2) Afficher les champs : nom, prénom et date de naissance

```
Program enregistrement ;
Uses wincrt ;

type
date = record
jour : 1..31 ;
mois : 1..12 ;
Annee : 1900..2014 ;
end ;
```

```
Etudiant = record
Nom, Prenom: string[30] ;
datenaiss : date ;
lieunaiss : string[20] ;
num_groupe : integer ;
Section : char ;
end ;
var
etudiant1 : Etudiant ;
begin
etudiant1.nom := 'Omar' ;
etudiant1.Prenom := 'Youssef' ;
etudiant1.Datenaiss. jour:=06 ;
etudiant1.Datenaiss. mois:=04 ;
etudiant1.Datenaiss.annee:=2004 ;
etudiant1.lieunaiss := 'Relizane' ;
etudiant1.Num_groupe : =5 ;
etudiant1.Section := 'A' ;
Writeln('L''etudiant : ', etudiant1.nom , '  ' ,etudiant1.Prenom ) ;
Writeln('né le : ', etudiant1.Datenaiss. Jour, '/', etudiant1.Datenaiss. Mois, '/',
etudiant1.Datenaiss.annee) ;
end .
```

2) Lire et afficher les champs de l'enregistrement etudiant2 :

```
Program enregistrement ;
Uses wincrt ;
type
date = record
jour : 1..31 ;
mois : 1..12 ;
Annee : 1900..2014 ;
end ;
Etudiant = record
Nom, Prenom: string[30] ;
datenaiss : date ;
lieunaiss : string[20] ;
num_groupe : integer ;
Section : char ;
end ;
```

```
var
etudiant2 : Etudiant ;
begin
     write (' Donner un nom :') ;
    readln(etudiant2.Nom) ;
    write(' Donner un prénom :') ;
    readln(etudiant2.Prenom) ;
   write(' Donner le num de groupe :') ;
    readln(etudiant2.num_groupe) ;
    write(' Donner la section:') ;
     readln(etudiant2.section) ;
    writeln('L"etudiant : ', etudiant2.nom , '  ',etudiant2.Prenom ) ;
    writeln ('Section et groupe : ', etudiant2.section, '  ',etudiant2.Num_groupe) ;
end.
```

3.2 Instruction with

L'instruction **with** … **do** permet de simplifier l'écriture du chemin d'accès aux différents champs de l'enregistrement.

Exemple :

Déclarer un enregistrement qui représente les informations d'un employé : nom, prénom, adresse, grade (A,B ,C et D), ensuite lire les champs de l'enregistrement **Emp1** (de type **Employe**) et afficher le nom de l'employé.

Solution :

```
Program emploi ;
Uses wincrt ;

type
Employe = record
Nom, Prenom: string[30] ;
Adresse : string[20] ;
Grade: char ;
end ;

var
Emp1: Employe ;
```

```
Begin
With emp1 Do
begin
    Write (' Donner un nom :') ;
    readln(Nom) ;
    Write(' Donner un prénom :') ;
    readln(Prenom) ;
    Write(' Donner le grade:') ;
   readln(grade) ;
    Writeln('l"employé : ', nom , '  ',Prenom ) ;
   End ;
End .
```

3.3 Tableau d'enregistrement

On peut créer un tableau où chaque case représente un enregistrement.

Exemple :

1) Créer un tableau **EMP** qui contiendra les informations sur 20 employés.
2) Afficher les noms des employés avec le grade A.

Solutions:

```
Program emploi;
uses wincrt;

type
Employe = record
Nom, Prenom: string[30] ;
Adresse : string[20] ;
Grade: char ;
end ;

var
emp :array [1..20] of Employe ;
 i:integer;
```

```
begin
for i:=1 to 20 do
with emp[i] do
begin
     Write (' Donner un nom :') ;
     readln(Nom) ;
     Write(' Donner un prénom :') ;
     readln(Prenom) ;
    Write(' Donner le grade :') ;
    readln(num_groupe) ;
end ;

for i:=1 to 20 do
if(emp[i].grade='A')  then
    begin
     Writeln('l''employé:', emp[i].nom ,'  ',emp[i].Prenom ) ;
    end;
end .
```

4 Notion de fichier

4.1 Définition

Un fichier est une structure de données qui permet de stocker des informations de manière permanente, sur un support externe un flash disque ou un disque dur…

Un fichier est représenté par :

- un identifiant (nom)
- un emplacement (dans par exemple le disque dur)
- un contenu (données).

4.2 Types de fichiers

Il existe deux types de fichiers :

1- Les fichiers textes (Text)

Les fichiers textes **(Text)** sont écrits au format texte (chaînes de caractères, nombres).

Déclaration :

Var f : Text;

2- Les fichiers typés (File of ...)

Les fichiers typés (**File of**) sont des fichiers dans lesquels les données sont du même type. Le type peut être standard ou non standard.

Déclaration :

Var f : **File of** {type};

Exemple :

Var f : **File of** Integer;

4.3 Principales commandes sur les fichiers

Les fichiers sont manipulés par des procédures particulières, dans la suite nous allons présenter les procédures principales:

4.3. 1 Assignation physique :

assign (Nom Fichier logique, Nom du fichier physique);

La commande ***Assign*** permet d'associer un fichier physique (sur le disque dur) à sa représentation virtuelle (une variable F de type fichier).

4.3.2 Création et ouverture d'un fichier :

Après l'assignation vient l'étape d'ouverture du fichier F afin de pouvoir effectuer des opérations de lecture ou d'écriture :

rewrite (F) : permet la création d'un nouveau fichier F, soit le rewrite le crée si il n'existe pas, ou bien si il existe déjà, le rewrite l'écrase pour l'écrire de nouveau.

reset (F) : permet l'ouverture du fichier F déjà existant pour lire son contenu.

4.3.3 Fermeture d'un fichier :

close (F): fermeture du fichier F.

4.3.4 Procédures de lecture et d'écriture

read (F, donnée) : lit à partir du fichier F la valeur de **donnée**.

write (F, donnée) : écrit la valeur de **donnée** à la fin du fichier F.

Append(F) : réouvre en écriture un fichier de type text pour ajouter du texte à la fin du fichier.

4.3.5 Indicateur de fin de fichier :

Eof (F) ; (Valeur booléenne, True / False)

4.4 Lecture et écriture dans un fichier

Après l'assignation on ouvre un fichier soit en lecture, soit en écriture.

En lecture : on fait reset(f); puis des read(f, ...);

En écriture : on fait rewrite(f); puis des write(f, ...);

Fin du fichier : En lecture, avant de faire un read, il faut tester s'il y a encore quelque chose à lire, donc on doit vérifier à chaque read si c'est la fin du fichier. Pour cela on utilise l'indicateur de fin de fichier.

4.5 Exemple

Ecrire un programme qui permet de créer un fichier qui contient un nom puis de lire ce fichier et d'afficher son contenu à l'écran.

Ce premier programme consiste à créer un nouveau fichier et de sauvegarder le nom de l'utilisateur :

```
Program fichier_ecriture;
uses  wincrt;
var    f : Text;
       S : String;
begin
  Assign(f,'c:\fichier.txt');
  Rewrite(f);
  Write ('Entrer le nom d''utilisateur : ');
  Readln(S);  (* Lecture à partir du clavier *)
  Write(f,S) ;  (* Ecriture du nom dans le fichier f *)
  Close(f);
 end.
```

Ce deuxième programme consiste à ouvrir un fichier déjà existant et d'afficher son contenu à l'écran :

```
Program fichier_lecture;
uses   wincrt;
var
  f : Text;
  S : String ;
begin
  Assign(f,'c:\fichier.txt');
  Reset(f);
  while (not Eof(f)) do    (* Tant que le fichier n'a pas atteint la fin on peut lire son contenu *)
    begin
      Read(f,S);   (*Lecture à partir du fichier f*)
      Write(S);      (* Affichage à l'écran du nom*)
    end;
  Close(f);
end.
```

5 Exercices

Exercice 1

1- Déclarez un enregistrement qui représente les informations d'un étudiant : nom, prénom, date de naissance, lieu de naissance, numéro de groupe, section.

2- Lire et afficher les champs nom prénom et date de naissance de l'enregistrement **etudiant1**.

3- Créer un tableau **Etud** et le remplir avec les informations (nom, prénom, date de naissance, lieu de naissance, numéro de groupe, section) de 5 étudiants.
4- Afficher les noms des étudiants du groupe 3 section B.

Série d'exercices avec solution

Exercice 2

Créer des variables intervalles heure (h), minute (m) et seconde (s), dans un type enregistrement temps_t.
Soit t1 et t2 deux temps_t.
Faire un programme qui lit t1 et t2 et qui dit s'ils sont égaux.

Solution :

```
Program temps ;
uses wincrt ;
type
temps_t = Record
h : 0..23;
m : 0..59;
s : 0..59;
end;
var
t1, t2 : temps_t;
begin
write ('Temps1 (h m s) : ');
readln (t1.h, t1.m, t1.s);
write ('Temps2 (h m s) : ');
readln (t2.h, t2.m, t2.s);
{ Teste si les temps sont égaux }
if (t1.h = t2.h) and (t1.m = t2.m) and (t1.s = t2.s))then
writeln ('Egalité ')
else
writeln ('Pas d''egalité');
end.
```

Exercice 3

Ecrire un programme qui permet de créer un fichier qui contient n nombre réels puis de lire ce fichier et calculer leur somme.

Solution :

```
Program fichier_ecriture;
Uses  wincrt;

Var
f : file of real;
N,i : integer;
 x,s:real;

  begin
  Assign(f,'c:\Nombres.dat');
  Rewrite(f);
  Write('Donner la valeur de n ');
  readln (n);
  Write('Donner les nombres réels ');
  for i:=1 to n do
  begin
  Readln(x);
   Write(f,x) ;
   end;  (* ajoute les nombres dans le fichier rien n'est afficher*)
   Close(f);
(* Ouvrir le fichier Nombres.dat *)
  Reset(f);
  Writeln('Lecture du contenu du fichier et calcul de s : ');
  s:=0;
  while not Eof(f) do
    begin
      Read(f,x);
      s:=s+x;
    end;
      Write('s = ',s);
close (f);
end.
```

Exercice 4

Créer un enregistrement qui représente les informations d'un étudiant : nom, prénom, moyenne. Créer un fichier qui contient les informations des étudiants puis lire ce fichier et afficher les étudiants admis.

Solution :

```
Program admis;
Uses wincrt ;
type
etudiant = record
Nom, Prenom: string[30] ;
moy:real;
end ;

var
f : file of etudiant;
etud:etudiant;
n,i:integer;

begin
  write('Donner le nombre des étudiants ');
  readln (n);
  assign(f,'c:\etud.dat');
  rewrite(f);
  write('Donner les données des étudiants ');
  for i:=1 to n do
  begin

with etud do
begin
     Write (' Donner un nom :') ;
     readln(Nom) ;
     Write (' Donner un Prénom :') ;
     readln(Prenom) ;
     Write(' Donner la moyenne') ;
     readln(moy) ;
    write(f,etud);
end;
end;
close(f);
```

```
reset(f);
Writeln('les étudiant admis ');

  while not Eof(f) do
    begin
    read(f,etud);
    if(etud.moy>=10)then
    writeln(etud.Nom, '  ', etud.Prenom);
    end;
    close(f);
 end .
```

Bibliographie

- Toufik Djouder, Exercices corrigés en Algorithmique, Dar El Hadith Lil-Kitab, 2002.

- Cours informatique2, 1éré année Sciences et Technologie, 2014-2015, Université frères Mentouri Constantine 1, consulté en 2015.

- Cours informatique, 1éré année Sciences et Technologie, 2014-2015, Université de Béjaïa, consulté en 2015.

- Edouard Thiel, Algorithmes et programmation en Pascal, 2004, Université Aix-Marseille.