# Les boucles (les structures itératives)

**L'utilité :**

Est une structure de l'algorithmique qui permet de répéter un traitement plusieurs fois pour une même données.

**Par exemple :**

Ecrire un algorithme qui permet d'afficher le mot ''bonjour'' 50 fois.

**50** : nombre de répétition.

Le traitement à répéter le mot **''Bonjour''**.

***Remarque :*** on dit le nombre de répétition ou bien le nombre d'itérations (qui signifié le passage d'un pas à l'autre dans une boucle).

- *On distingue trois types de boucles :*La structure pour, la structure tant que et la structure répéter jusqu'à.

***a- La structure Pour :***

**Propriétés :**

- On utilise la structure pour quand le nombre d'itération est connu à l'avance.
- Le compteur est initialisé à 1.
- Le pas d'incrémentation =1

***Syntaxe :***

*Pour compteur allant de (valeur initiale) à (valeur finale) faire*
*∑ Instructions*
*Fin pour*

***Exemple :***

*Pour i allant de 1 à 3 faire*
*Ecrire (''bonjour'')*
*Fin pour*

***En pascal***

```
For i :=1 to 3 do
Begin
Writeln('bonjour');
End;
```

### *b- La structure Tant que :*

Contrairement à la boucle pour, la structure tant que permettent de faire des itérations tant que la condition est vérifiée.

***Propriétés :***

- Le nombre d'itérations ne connu pas à l'avance,
- Prmet de vérifier si la condition est vraie pour exécuter le bloc d'instructions, si la condition est fausse on sort de la boucle.

***Syntaxe :***

*Tant que (condition) faire*
*∑ Instructions*
*Fin tant que*

**Exemple :**

*S ←0*
*Tant que ( i<=5 )faire*
*S ←S+i*
*i ←i+1*
*Fin tant que*
*Ecrire(s)*

Cet algorithme s'arrête des que le compteur i>5

**En Pascal:**

```
S: =0;
While (i<=5) do
Begin
S: =S+1;
i:=i+1;
end;
Writeln('la somme=',S);
```

***Remarque :***

Si on connait le nombre de répétition a traiter on utilise la boucle pour mais si on connu la condition mais on ne connu pas le nombre de répétition on utilise la boucle tant que.

### *c- La structure répéter jusqu'à :*

La structure répéter jusqu'à est semblable à la structure de tant que mais la différence est la boucle tant que permet d'exécuter le bloc d'instructions tant que la condition est vraie contrairement à la boucle répéter jusqu'à qui permet d'exécuter le bloc d'instruction jusqu'à la condition devient vraie.

***Remarque :***

La boucle tant que vérifie la condition avant chaque itérations (au début) mais La boucle répéter jusqu'à vérifie la condition après chaque itération (à la fin).

***Syntaxe :***

*Répéter*
*∑ Instructions*
*Jusqu'à (condition sera vraie)*

***En pascal :***

*Repeat*
*∑Instructions*
*Until (condition sera vraie)*

***Exemple :***

```
S ←0
répéter
    S ←S+i
    i ←i+1
Jusqu'a (i>5)
Fin répéter
Ecrire(s)
```

**En Pascal:**

```
S: =0;
repeat
Begin
S: =S+1;
i:=i+1;
   until( i>5)
end;
Writeln('la somme=',S);
```

## ***Application :*** **Les Structures Répétitives**

### **Exercice N°1 :**

1- Écrire un algorithme qui affiche tous les entiers pairs de 1 à 24.

***Solution :***

*Algorithme pair*

*Variables i : entier*

*Début*

*Pour i allant de 1 à 24 faire*

*Si (i mod 2=0) alors écrire (''ce nombre'',i, ''est pair'')*

*Sinon écrire (''ce nombre '',i ''est impair'')*

*Fin si*

*Fin pour*

*Fin*

***En pascal :***

```
Program pair;
Var I: integer;
Begin
For I :=1 to 24 do
If(I mod 2=0) then writeln ('le nombre',I,'est pair')
Else  writeln ('le nombre',I,'est impair') ;
Readln ;
End.
```

### **Exercice N°2**

1- Ecrire un algorithme qui calcule la formule suivante :

$$S=1+\frac{1}{2}+\frac{1}{4}+\frac{1}{6}+\cdots$$

**Solution:**

*Algorithme somme*

*Variable s, i: entier*

*Debut*

*S⟵1*

*Pur I allant de 1 à 3 faire*

*S ⟵ S+1/2*i*

*Ecrire (S)*

*Fin pour*

*Fin*

***En Pascal***

*Program somme;*

*Var S,I :integer;*

*Begin*

*S:=1;*

*For i: =1 to 3 do*

*S: =S+ (1/ (2*i));*

*Writeln('la somme=',S);*

*Readln ;*

*End.*

**Exercice N°3 :**

1- Compléter le programme suivant :

```
Program calcul;
Var i:entire
begin

i: = 1 ;
while i <= 15000 do
writeln(i);
   i: = i + 2;
readln
end.
```

2- Que ce fait ce program ?

3- Remplacer la boucle while par la boucle For et la boucle repeat until.

***Solution :***

```
Program calcul;
Var i:integer;
begin

i: = 1 ;
while i <= 15000 do
writeln(i);
   i: = i + 2;
readln
end.
```

- Cet algorithme permet d'afficher des nombres impairs:1 3 5 7……..15000.

*Algorithme calcul*
*Variable i : entier*
*Début*
*Pour i allant de 1 à 15000 avec un pas de 2 faire*
*Ecrire(i)*
*Fin*

## En pascal

En pascal cette structure n'existe pas, mais dans on peut la replacer par la boucle while ou bien la structure repeat .

# Les tableaux à une dimension (vecteurs)

## ***Introduction***

Dans le cas ou on veut écrire un algorithme permettent de calculer la moyenne de 100 étudiants, donc on besoin de déclarer 100 variables.

***Remarque :***

Pour 100 étudiants on, déclare 100 variables.

.
.
.

Pour n étudiants on déclare n variables.

Ce n'est pas pratique et pose deux problèmes d'un part le temps d'exécution et d'autre part l'espace mémoire.

Donc pour résoudre ce problème, on utilise une structure appelé tableau, au lieu de déclarer n variable on déclare une seule variable de type tableau.

**Définition :**

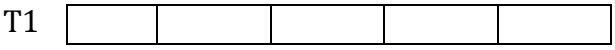
Un tableau (ou un vecteur) est une structure ou une variable qui permet de regrouper plusieurs éléments (d'un nombre entier fini) de même type.

Ces éléments (sont des suites de cases les unes à coté des autres) ou bien sont des composants d'un tableau ou on peut les traiter (accéder, supprimer ou modifier) élément par élément à partir de son indice (n° du rang) qui indique la position de cet élément.

*Un tableau T1 est identifié par :*

- Son Nom : identificateur d'un tableau
- Le type : il faut déclarer le type des éléments d'un tableau (entier, réel, Caractère,..)
- Sa dimension (longueur du tableau).

T1 : tableau [1..5] : réel



***Syntaxe :***

**Variable <nom du tableau > : tableau [borne inferieur.. Borne supérieur] :<type du tableau>**

**En pascal :**

Var **<nom du tableau > array [borne inferieur.. Borne supérieur] :<type du tableau>**

**Exemple 1 :**

*Variable T1 : tableau [1..5] : entier*

**En pascal :**

*Var T1: array [1..5] of integer ;*

**Exemple2 :**

*Const max=5*
*Type*
*Tab : tableau [1..max] d'entier*
*Var T1 : tab*

**En pascal :**

*Const max=5*
*Type*
*Tab: array [1..5] of integer;*
*Var T1:tab;*

***a- Accès aux éléments d'un tableau :***

Pour accéder aux éléments d'un tableau on utilise T1[i]

***Exemple :***

$T_1$ [1] accès au 1 er élément
$T_1$ [2] accès au 2 éme élément
T1 [i] accès au i éme élément

***b- Remplissage d'un tableau :***

*Procédure remplir (var T : tab, n : entier)*
*Var i : entier*
*Début*
*Pour i allant de 1 à n faire*
*Ecrire (T[',i,'])*
*Lire(T[i])*
*Fin pour*

***c- Affichage les éléments d'un tableau :***

*Procédure affiche(t : tab, n : entier)*
*Var i : entier*
*Début*
*Pur i allant de 1 à n faire*
*Ecrire(T[i])*
*Fin pour*
*Fin*

### d- Les différents traitements sur les éléments d'un tableau

Ecrire un algorithme qui permet de calculer la somme des éléments d'un tableau.

Algorithme tab

*Var tab : tableau [1..5] d'entier*
*I,som :entier*
*Debut*
*Som⟵ 0*
*Pour I allant de 1 à 5 faire*
*Ecrire ('tab [', i',]')*
*Lire(tab[i])*
*Som ⟵ Som+tab [i]*
*Fin pour*
*Ecrire ('la somme==',Som)*
*Fin*

### e- *Recherche dans un tableau*

Il existe deux types de recherches : recherche séquentielle et recherche dichotomie

**a) Recherche séquentielle** : dans le cas d'un tableau n'est pas trie, on utilise ce type de recherche.

**Exemple** :

*Algorithme recher-séquentielle*
*Const M=10*
*Var T : tableau [1..10] d'entier*
*I,n :entier*
*Trouve : booléenne*
*Début*
*Pour I allant de 1 à 10 faire*
*Lire (T[i])*
*Fin pour*
*Ecrire (''donner la valeur à rechercher'')*
*Lire(x)*
*Pour I allant de 1 à 10 faire*
*Si (T[i] ==x) alors trouve faux sinon trouve vrai*
*Fin si*
*Fin pour*
*Si trouve vrai écrire(x,''cet élément est existe'') sinon écrire (x,''cet élément n''existe pas')*
*fin si*
*Fin*

**b) <u>Recherche dichotomique</u>** : on utilise ce type de recherche dans le cas d'un tableau trie.

***<u>Principe :</u>***

Le principe de cet algorithme repose sur la décomposition du tableau en deux sous tableaux et voir la position de l'élément à rechercher si :

Si val = T[milieu] alors la valeur est trouvée et la recherche est terminée.

Si val < T[milieu] alors on va chercher la valeur dans la partie gauche du tableau T.

Si val > T[milieu] alors on va chercher la valeur dans la partie droite du tableau T.

Donc on utilise trois variables (gauche, milieu et droite)

L'élément à rechercher appelé x

La division utilisée est une division entière (Div).

```
Algorithme recherche_dichotomique
variables T :tableau [0.. N-1] :entier
 variable  x,lnf, Sup, N, Midd : entier
 Début
écrire ('donner la valeur à rechercher')
lire(x)
Inf ← 0
Sup ← N-1
Mi dd← (Inf + Sup)/2
 tant que ( val <> T[Midd] && Inf <= Sup) faire
si val < T[Midd] alors Sup = Midd - 1 sinon Inf = Midd + 1
 fsi
Mi dd← (Inf + Sup)/2
Fin tq
si T[Midd] = val
alors écrire ( " l'élément existe dans le : » ,Midd);
sinon écrire ( « Elément n''existe pas " );
 fin si
Fin
```

***<u>Algorithme de Tri d'un tableau</u>*** *:*

Il existe plusieurs algorithmes de tri parmi eux on cite : Tri à bulle et Tri par sélection.

- **<u>Tri à bulles</u>**

Le principe de cet algorithme repose sur le parcoure du tableau et tester si un élément de position i est supérieur à l'élément i+1 on les permute. le programme arrête si on a pas besoin de faire une permutation.

*Algorithme Tri _bulle*

*Variables tab : tableau [0..N-1] d'entier*

*N,i,j :entier*

*Début*

*Pour i allant de 1 à N-1 faire*

*Si T[i]>T[i+1]alors echange(T[i],T[i+1])*

*Fin si*

*finpour*

***Remarque*** **:**
Echange : est une procédure (voir l'exercice 2 de la fiche TP8)

# Les tableaux à deux dimensions (matrices)

***Définition :***

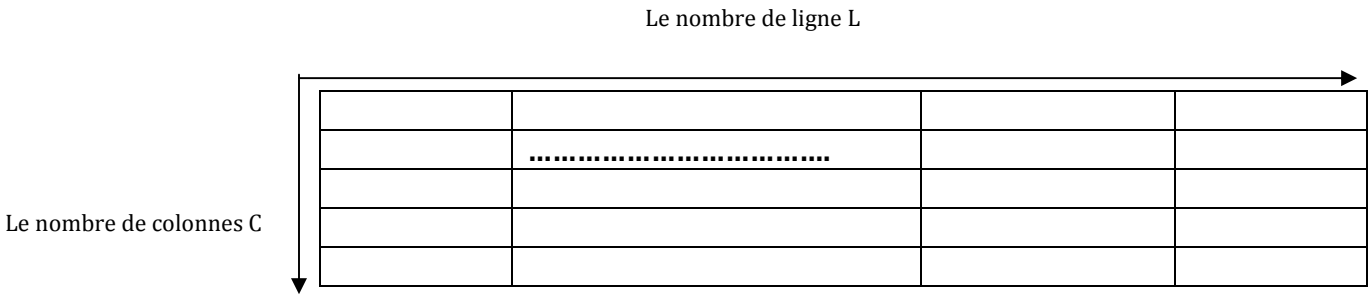
Une matrice ou un tableau à deux dimensions et une structure qui permet de regrouper plusieurs tableaux de même taille et le même type.

***Par exemple*** : On a 4 tableaux de 10 éléments, Nous donne une matrice de 4 lignes et 10 colonnes :

| | | | | |
|---|---|---|---|---|
| T1 | | | | |
| T2 | | | | |
| T3 | | | | |
| T4 | | | | |

Le type d'une matrice doit être le même type des éléments des tableaux regroupés.

**Exemple :** écrire un algorithme qui permet de calculer la moyenne de 3 sections chaque section contient 130 étudiants, Donc nous donne une matrice de 3X130 réel.
L le nombre de lignes et C est le nombre de colonnes
Dans notre exemple L=3 et C=130.



***Déclaration :***

**Syntaxe :**

*Nom_tab : Tableau [min_ligne. .max_ligne , min_colonne. .max_colonne] de type_ élément*

**Exemple :**

*M : tableau [1..3,1..130] de réel*

**En pascal :**

*M: array [1...3,1..130] of real ;*

**Exemple2:**

*CONST NbLigne = 30*

*NbColonne = 20*

*TYPE MAT : Tableau [1.. NLigne, 1.. NColonne] d'entier*

*VAR M : MAT*

a- **Accès aux éléments d'une matrice :** soit un tableau de N ligne et M colonne

A[I,J] désigne une matrice A de Iième et J ième colonnes.

**Exemple :** Soit la matrice A donnée par:

| | | | |
|---|---|---|---|
| **1** | **0** | **67** | **10** |
| **7 =A [2,1]** | **11** | **8** | **3** |
| **4** | **6** | **12** | **2** |
| **23** | **16=A [4,2]** | **25** | **18** |

**A [1,2] =0** ⇨ le contenu du 1ére ligne et 2éme colonne de A.

**A [3,4]=2** ⇨ le contenu du 1ére ligne et 2éme colonne de A.

**b- Remplir les éléments d'une matrice :**

Pour remplir une matrice on besoin de deux boucles l'une pour les lignes et l'autre pour les colonnes.

*Algorithme remplir*

*VAR M : Tableau [1.. 4, 1..4] d'entier*

*i ,j : entier*

*DÉBUT Pour i de 1 à 4Faire*

*Pour j de 1 à 4 Faire*

*Écrire ("A [" , i , ", " , j , "] :")*

*Lire (A [i, j])*

*Fin pour*

*Fin pour*

*FIN*

L'algorithme permet de remplir les éléments d'une matrice ligne par ligne.

***En pascal :***

**c- Affichage le contenu d'une matrice**

*Algorithme Afficher*

*VAR A : Tableau [1.. 4, 1..4] d'entier*

*i,j : entier*

*DÉBUT*

*Pour i de 1 à 4 Faire*

*Pour j de 1 à 4 Faire*

*Écrire (A [i, j])*

*Fin pour*

*Fin pour*

*FIN*

***En pascal***

**Quelques exemples de traitements sur une matrice :**

**Exemple 1 :**

Ecrire un algorithme qui permet de calculer la somme des éléments d'un tableau

*Procédure SOMME (M1 , M2 : MAT) ;*

*VAR M3 : MAT ;*

*n, m : entier*

*VAR i, j : entier*

*DÉBUT*

*Pour i de 1 à n Faire*

*Pour j de 1 à m Faire*

*M3 [i, j] ← M1 [i, j] +M2 [i, j]*

*Fin pour*

*Fin pour*

*FIN*

**Exemple 2**:Ecrire un algorithme permettent de donner la transposée d'une matrice A

*Algorithme transposée*

*Variable mat : tableau [1..n, 1..m] d'entier*

*Début*

*Ecrire (''Donner la taille de la matrice caree'')*

*Lire (n,m)*

*Pour i allant de 1 à n faire*

*Our j allant de 1 à m faire*

*Ecrire ('A [',i,', ',j,']=')*

*Lire (A [i,j])*

*Pour i allant de 1 à n faire*

*Pour i allant de 1 à n faire*

*B [i,j]←A[j,i]*

*Lire (B [i,j])*

*Fin*

***En pascal***

```
Program transposée;
Var mat: array [1..5 1..5] of integer;
Begin
Writeln('donner la taille de la matrice A') ;
Readln (n) ;
For i :=1 to n do
begin
For j :=1 to n do
B [i,j]:= A[j,i];
Writeln(B[i,j]);
End;
readln;
End.
```

## ***Application : Les Tableaux à une dimension(les vecteurs)***

**Exercice N°1 :**

Ecrire un programme pascal qui permet de remplir un tableau de 10 entiers puis déterminer les nombres d'éléments divisibles par deux.

**Exercice N°2 :**

Faire un programme déterminant le minimum d'un tableau de 7 réels.

**Solution :**

*Algorithme tableau*

*Variables T : tableau [1..5] de réel*

*I :entier*

*Début*

*Pour i allant de 1 à 7 faire*

*Ecrire ('T [', i,']=')*

*Lire(T[i])*

*Min ← T [1]*

*Pour i allant de 2 à 7 faire*

*Si T[i]<min alors min ← T [i]*

*Fin si*

*Fin pour*

*Ecrire('le minimum=',min)*

*Fin*

***En pascal***

*Program tableau ;*

*Var T : array [1..7] of real ;*

*Var i: integer;*

*Begin*

*For i:=1 to 7 do*

*begin*

*Writeln('T[',i,']);*

*Readln(T[i]);*

*End;*

*Min:=T[1];*

*For i:=2 to 7 do*

*If T[i]<min then min :=T[i];*

*Writeln('le minimum=',min);*

*Readln ;*

*End.*

**Exercice N°3 :**

Ecrire un programme permettant de calculer la somme et la moyenne des éléments d'un tableau de 100 entiers.

Algorithme SomMoy

Variable T1 : tableau [1..100] d'entier

S, I : entier

Moy :réel

*Début*

*Pour i allant de 1 à 7 faire*

*Ecrire ('T [', i,']=')*

*Lire(T[i])*

S ← 0

Pour I allant de 1 à 100 faire

S ← S+T1[I]

Fin pour

Ecrire(S)

Moy← S/100

Ecrire(Moy)

Fin

***<u>En pascal :</u>***

*Program tableau;*

*Var T1 : array1..7[] of real ;*

*Var I,s :integer;*

*Moy:real;*

*Begin*

*For i:=1 to 100 do*

*begin*

*Writeln('T1['I,']);*

*Readln(T1[I]);*

*End;*

*Pour I :=1 to 100 do*

*S:=S+T1[I];*

*Writeln('la somme=',S);*

*Moy:=S/100;*

*Writeln('la moyenne=',Moy);*

*Readln ;*

*End.*

## Exercice N°4 :

Ecrire un programme permettant d'ajouter un élément à la fin d'un tableau.

***Solution***

*Algorithme ajout*

*Variable T1 : tableau [1..7] de réel:*

*N,pos,i :entier*

*X: reel*

*Début*

*Si (N=0) alors écrire ('le tableau est vide')*

*Sinon ('donner la valeur à rechercher')*

*Lire(x)*

*Ecrire ('donner sa position')*

*Iire (pos)*

*Si(pos<1)ou (pos>N) alors écrire(la ^position est hors limite')*

*Sinon N ← N+1*

*Pour i allant de N à Pos+1 faire*

*T[i] ← T[i-1]*

*Fin pour*

*T[pos] ← x*

*Fin si*

*Fin si*

*Fin si*

*Fin*

***Remarque : dans cet exercice la position =N (l'insertion s'effectue à la fin du tableau)***

***En pascal :***

```
Program ajout
   Var  T1 :array[1..7] of real ;
   N,pos,i: integer;
  X:real;
Begin
      If N=0 then writeln('le tableau est vide')
                Else writeln('donner la valeur du l''elemnt à rechercher') ;
       Readln(x) ;
      Writeln('donner sa position') ;
      Readln(pos) ;
If (pos<1) or (pos>N) then writeln('la position est hors limite')
```

```
Else  N :=N+1
For i :=N to pos+1 do
begin
T[i]:=T [i-1];
End;
T[pos]:=x;
Readln;
End.
```

# Les procédures et les fonctions

## *Introduction*

Dans le cas de besoin d'écrire des programmes importants et complexes, il est difficile de prendre une idée générale sur son fonctionnement et aussi il reste difficile de trouver les erreurs et de les corriger, ou il faut éviter de répéter des suites d'instructions autant de fois en effectuant des calculs par des données différentes.

Comme solution, on décompose le problème en sous parties ou il faut chercher une solution à chaque sous partie.

En algorithmique ces sous parties sont des sous programmes qui prendre deux formes soit la forme d'une procédure soit la forme d'une fonction.

**Définition** :

- Un sous programme est un bloc d'instructions, il est déclaré avant le début du programme principal. Le sous programme peut utiliser dans sa partie déclarative les variables de l'algorithme principale ( on dit des variables globales) et peut utiliser ses propres variables (les variables locales) et qui ne peut pas être utilisés dans le corps principale de l'algorithme ou par d'autres sous programme.
- Une procédure : est un sous programme qui compose d'une séquence des instructions qui possède un nom spécifique.

## Le rôle d'utilisation des procédures et les fonctions :

- Pour un programme être plus lisible.
- La facilité de traquer les erreurs.
- Le même sous programme peut être utilisé dans plusieurs programmes afin d'économiser le temps d'exécution et minimiser l'espace mémoire.

## 1-Les procédures :

**Syntaxe** :

*Procédure _ nom de la procédure (liste des paramètres : type)*

*Partie déclaratives*

*Début*

*∑Instructions*

*Fin*

**En pascal :**

*Procédure _ nom de la procédure (liste des paramètres : type) ;*

*Partie déclaratives*

*begin*

*∑Instructions*

*End ;*

La structure de la procédure est identique à celle de l'algorithme qui possède une entête (nom), une partie déclarative (déclaration des variables) et un corps (séquences des instructions).Le traitement d'une procédure est basé sur 2 étapes : la déclaration et l'appel.

### Appel de la procédure :

Pour activer une procédure il suffit de faire un appel de cette procédure par l'instruction suivante :

*Nom_ procédure (listes des paramètres)*

En algorithmique, on distingue deux catégories d'une procédure, une procédure avec paramètres et procédure sans paramètres.

### *a) Procédure sans paramètre*

***Exemple :***

*Algorithme Affiche*

*Const language= ''pascal ''*

*Procédure proc*

```
Var  i:entier
Début
Pour  i allant de 1 à 5  faire
Ecrire (''***** '')
Fin pour
Fin
Début
Ecrire (langage)
Proc ()
Fin
```

***En pascal :***

```
Program affiche ;
Const=''language'';
Procedure proc ();
Var i: integer;
Begin
For i: =1 to 5 do
Writeln('****');
End;
Begin(programme principle)
Writeln(langague)
Proc();
Readln;
End.
```

### *b) Procédure paramétrée (avec paramètre)*

Est un sous programme qui utilise des paramètres ou des variables pour faire la transmission entre la procédure appelée et le programme appelant.

On distingue deux types de paramètres.

- **Les paramètres formels (fictifs)** : qui se trouvent dans l''entete de la partie déclarative de la procédure (variables locales).

  ***Exemple :***

  *Procédure_nom (moy1 :type1 ;moy2 :type2 ;… moy n :type n)*

- **Les paramètres effectifs** : qui se trouvent dans l'instruction de l'appel

***Exemple 1:***

*nom_procedure(x,y)*

***Exemple 2 :***

***Algorithme somme***
***Variable n : entier***
*Procédure Som( n :entier)*
*S⟵0*
*Pour I allant de 1 à n faire*
*S⟵ S+i*
*Fin pour*
*Ecrire(S)*
*Fin*
***Début (programme principale)***
***Lire(a)***
***Som(a)***
***Fin***

***En pascal :***

```
Program somme ;
Var n : integer ;
Procedure somme(n :integer) ;
S :=0 ;
For i :=1 to n do
                S: =S+I;
                Writeln('la some=',S);
                Begin
                Writeln('a=');
                Readln(a);
                Somme (a);
                Readln;end.
```

**1) <u>Passage des paramètres</u>** : il existe deux types :

- passage par valeur.
- passage par variable.

**Passage par valeur**

```
Algorithme  calcul
Var x : entier
Procédure traitement (j : entier)
Début

j ⟵  (3+j)*2

écrire (''la  valeur j'',j)
Fin
Début
Écrire (''donner la valeur de x'')
Lire(x)
Traitement(x)
Écrire (''après appel x='', x)
Fin
```

| | | |
|---|---|---|
| Avant l'appel | x=3 | j=3 |
| Exécution | | j=12 |
| Après | x=3 | j=12 |

***<u>Remarque :</u>***

Le paramètre effectif est recopié dans la valeur de j=3, après l'appel de la procédure, l'algorithme affiche la valeur du variable j=12.Dans ce type de passage ne modifiée pas le paramètre qui est passé par valeur.

**Passage par variable**

```
Algorithme  calcul
Var x : entier
Procédure traitement (var j : entier)
Début

j ⟵  (3+j)*2

écrire (''la  valeur j'',j)
Fin
Début
Écrire (''donner la valeur de x'')
Lire(x)
Traitement(x)
écrire (''après appel x='',x)
Fin
```

| | | |
|---|---|---|
| Avant l'appel | x=3 | j=3 |
| Exécution | | j=12 |
| Après | x=12 | j=12 |

***Remarque :***

Le paramètre effectif est recopié dans la valeur de j=3, après l'appel de la procédure, l'algorithme affiche la valeur du variable j=12.Dans ce type de passage modifie le paramètre qui est passé par variable.

## 2- Les fonctions :

Et un sous programme possède des paramètres et retourne qu'une seule variable (un seul résultat). Dans la partie déclarative d'une fonction, il faut préciser le type de la fonction et le type de résultat.

**Structure d'une fonction :**

**syntaxe**

*Fonction _ nom(paramètre formels :type) :type*

*Les déclarations*

*Début*

*Instructions*

*Nom_ fonction ⟵ résultat*

*Fin*

**Appel d'une fonction**

*Nom_ fonction (liste des paramètres effectifs)*

**Exemple**

Ecrire une fonction max qui donne le maximum entre deux variables x et y

*Algorithme maximum*

*Variable x,y :entier*

*Fonction max (var a,b :entier) :entier*

*Variable c : entier*

*Début*

*Si (a>b) alors c ⟵ a*

*Sinon c ⟵ b*

*Max ⟵ c*

*Fin si*

*Fin*

*Début*

*Lire(x)*

*Lire(y)*

*Z← max(x,y)*

*Ecrire (''le maximum='',z)*

*Fin*

***<u>En pascal</u>***

```
Program maximum;
Var x,y :integer ;
Function max (var x,y :integer) :integer ;
Var c: integer;
Begin
If (a>b) then c: =a
Else c: =b;
max:=c;
end;
begin
writeln('x=');
readln(x);
writeln('y=');
readln(y);
z:=max(x,y);
writeln('le maximum=',z) ;
readln ;
end.
```

# Application : Les procédures et les fonctions

## Exercice N°1

**Ecrire :**

1. Une Procédure **Min** de 2 entiers.
2. Une Procédure **Max** de 2 entiers.
3. Une Procédure **MinMax** de 2 entiers, qui appelle les procédures Min et Max.
4. Une Programme qui lit 2 entiers, appelle **Min, Max** et affiche les résultats.

**Solution :**

***Procédure Min (var a,b :entier)***
***Variable min : entier***
***Début***
***Si (a<b) alors min ← a***
***Sinon min ← b***
***Fin si***
**2)**
***Procédure Max (var a,b :entier)***
***Variable max : entier***
***Début***
***Si (a>b) alors max ← a***
***Sinon max ← b***
***Fin si***

***3)***

```
Algorithme MinMax
Variables x,y :entier
Procédure Minimum (var a,b :entier)
Variable min : entier
Début
Si (a<b) alors min ← a
Sinon min ← b
Fin si
Ecrire('le minimum=',min)
fin
Procédure Maximum (var a,b :entier)
Variable max : entier
Début
Si (a>b) alors max ← a
Sinon max ← b
    Fin si
    Ecrire ('le maximum', max)
    Fin
    Début (du programme principal)
    Lire(x)
    Lire(y)
    Minimum(x,y)
    Maximum(x,y)
    Fin
```

***En pascal :***

```
Program MinMax
Var x,y :integer ;
Procedure Minimum (var a,b :integer)
Var min :integer;
Begin
If(a<b) then min :=a
Else min:=b;
Writeln('le minimum=',min);
End;
Procedure Maximum (var  a,b :integer);
Var max :integer;
Begin
If(a>b) then max :=a
Else max:=b;
Writeln('le maximum=',max);
End;
Begin
Minimum(x,y);
Minimum(x,y);
Readln ;
End.
```

**Exercice N°2**

Ecrire une procédure Echange2 qui échange éventuellement les deux réels a et b.

```
Algorithme permutation
Variable x,y :entier
Procédure échange (var a,b :entier)
Variable c : entier
Début
c ← a
a ← b
b ← c
Ecrire(a)
Ecrire(b)
Fin
Début
Lire(x)
Lire(y)
```

*Echange(x,y)*
*Fin*

***En pascal***

```
Program permutation;
Var x,y :integer ;
Procedure echange (var a,b :integer) ;
Var c: integer;
Begin
C: =a;
A: =b;
B: =c;
Writeln('a=',a);
Writeln('b=',b);
End;
Begin
Writeln('x=');
readln(x);
Writeln('y=);
Readln(y);
Echange(x,y);
Readln;
End.
```

**Exercice N°4**

Faire une fonction facto(n) qui renvoie n! .

**Solution :**

```
Algorithme factorielle
Variable T,x :entier
Fonction fact(var n :entier) ;entier
Variable F,i:entier;
Debut
  F ← 1
Pour i allant de 1 à n faire
  F ← F*i
Ecrire(F)
Fin
Début
Lire(x)
T← fact(x)
fin
```

**en pascal :**

```
Program factorielle ;
var T,n :integer ;
Function fact (var n :integer) :integer ;
Var F,I:integer;
Begin
F:=1;
For I :=1 to n do
F:=F*I;
Writeln(le factorielle'=,F);
```

*End;*
*Begin*
*Writeln('x=');*
*Readln(x);*
*T:=fact(x);*
*Writeln(le factorielle'=,T);*
*Readln ;*
*End.*

**Exercice N°3**

Ecrire un programme qui saisie deux tableaux A et B par des nombres réels, calcule la fonction somme donnée par S=2*A-3*B puis affiche Z.

**Solution :**

*Algorithme vect*
*Type tab : tableau [1..7] d'entier*
*Variable T1, T2 : tab*
*Fonction Somme (var A, B : tab, n : entier) ; entier*
*Variable S : entier*
*Début*
*Pour i allant de 1 à n faire*
*Début*
*Pour j allant de 1 à n faire*
*S ← 2*A[i]-3*B[j]*
*Somme ← S*
*Finpour*
*Début*
*Pour i allant de 1 à 7 faire*
*Ecrire ('T1[',I,']=')*
*Lire(T1[i])*
*Pour i allant de 1 à 7 faire*
*Ecrire('T2[',I,']=')*
*Lire(T2[i])*
*Z ← Somme (T1 [i], T2 [j])*
*Ecrire(Z)*
*Fin*

**En pascal :**

*Program vect ;*
*Type tab: array [1..7] of integer;*
*Var T1, T2 : tab*
*Function Somme (var A, B : tab, n integer) :integer ;*
*Variable S : integer ;*
*begin*
*for i := 1 to n do*
*begin*
*for j := 1 to n do*
*S:= 2*A[i]-3*B[j];*
*Somme:= S;*
*begin*
*for i := 1 to7 do*
*writeln ('T1[',I,']=');*
*readln(T1[i]);*

```
for i := 1 to 7 do
 writeln('T2[',I,']=');
readln(T2[i]) ;
Z := Somme (T1 [i], T2 [j]) ;
Writeln('la somme=',Z) ;
Readln ;
End.
```

**Exercice N°5**

Soit une matrice M de 20 lignes et 20 colonnes à valeurs entières, donner une fonction qui calcule la somme des éléments diagonaux de cette matrice M.

```
Algorithme Matrice
Const n=10
        Type
        Mat : tableau [1..n,1..n]d'entier
Var M : Mat
Fonction Somme (A: Mat ; n : entier) : entier
Variable : i, Sdiag : entier
Début
Sdiag◄— 0

Pour i allant de 1 à 10 faire
Sdiag ◄— Sdiag+A[i,i]
Fin pour
Somme ◄— Sdiag
Fin
Début
Pour i allant de 1 à10faire
Pour j allant de 1à 10 faire
Ecrire('M[',i,',j,']=')
Lire (M[i,j])
S ◄— somme (M[i,j])
Ecrire ('la somme des éléments diagonaux=',S)
Fin
```

**En pascal :**

```
Program matrice ;
Const n=10 ;
Type Mat: array [1...10, 1..10]of integer ;
Var  M:Mat;
Function Somme (A: Mat; n: integer): integer;
Var  i, Sdiag : integer;
begin
Sdiag:= 0;
Pour i := 1 to 10 do
Sdiag:=Sdiag+A[i,i];
Somme :=Sdiag ;
End ;
```

```
begin
for i :=1 to10 do
for j :=1to 10 do
writeln('M[',i,',j,']=');
readln (M[i,j]);
S :=Somme (M[i,j]);
writeln ('la somme des éléments diagonaux=',S);
readln;
end.
```

# Les enregistrements et les fichiers

## A/Les enregistrements

### 1- Définition et l'utilité :

Un enregistrement est une structure permettent de ranger un ensemble des données de type différents.

**Exemple** : écrire un algorithme qui permet de remplir les données d'un patient dans une clinique médicale.

| | **N° patient** | **Nom** | **prénom** | **date de naissance** | **N°CNAS** |
|---|---|---|---|---|---|
| Type : | entier | chaine | chaine | entier | entier |

Mais on ne peut pas regrouper ces champs de type différents dans un tableau, la solution est une nouvelle structure qui est appelée un enregistrement.

**Syntaxe :**

*Type structure Nom_ type=enregistrement*
*Champ1 : type1*
*…….*

*Champ n : type n*
*Fin nom_ type*

**En pascal :**

*Nom_ type=record*
*Champ 1 :type1*
*…….*
*Champ n : type n*
*End ;*

**Déclaration d'un enregistrement à partir d'un type :**

*Var nom_var :nom_type*

**Exemple :**

*Type structure tpatient*

*Nom : chaine*

*Prénom : chaine*

*Âge : entier*

*Fin structure*

*Var patient1, patient2, patient3 : tpatient*

**Représentation :**

Patient1

| **Patient1.nom** | **Patient1.prénom** | **Patient1. âge** |
|---|---|---|

Patient 2

| **Patient2.nom** | **Patient2.prénom** | **Patient2 .âge** |
|---|---|---|

Patient3

| **Patient3.nom** | **Patient3.prénom** | **Patient3. âge** |
|---|---|---|

**Accès aux champs d'un enregistrement :**

L'Accès aux champs d'enregistrement grâce à l'operateur ''. ''. Par exemple : pour accéder au nom du patient 2 on utilise l'expression patient2 .Nom qui signifie le nom du patient 2(écriture à gauche).

**Exemple** : écrire un algorithme qui permet de rempli les données es patient 1et patient 2 et afficher la différence d'âge entre les deux patients.

*Algorithme diff*

*Type structure tpatient*
*N : chaine*
*Prénom : chaine*
*Âge : entier*
*Fin structure*
*Var patient1, patient2 : tpatient*

*Début*

*Ecrire (''écrire le nom du patient 1'')*

*Lire (patient1.nom, patient1.prenom, patient1.age)*

*Ecrire (''écrire le nom du patient 2'')*

*Lire (patient2.nom, patient2.prenom, patient2.age)*

*Ecrire (''la différence d''age entre patient1 et patient2,*

*Si patient1.age>patient2.age alors écrire patient1.âge-patient2.âge*

*Sinon patient2.âge-patient1.âge*

*Fin si*

*Fin*

***Exemple2 :***

*Type structure date*

*Jour : entier*

*Mois : entier*

*Année : entier*

*Fin structure*

*Structure personne*

*Nom : chaine*

*Date de naissance : date*

*Fin structure*

Dans cet exemple un type structuré peut être utilisé comme un champ d'un autre type, par exemple pour accéder à l'année de naissance d'une personne on utilise :

*Pers1.date de naissance.année*

***Les tables d'enregistrements :***

***Déclaration***

***Syntaxe :***

*Var groupe : tableau [1..N] de personne*

*Const N=7(nombre de personnes du groupe)*

*Type structure personne*

*Nom : chaine*

*Prénom : chaine*

*Age : entier*

*Fin structure*

| *indice* | nom | Age | prénom |
|---|---|---|---|
| ***1*** | | | |
| ***2*** | | | |
| ***3*** | | | |
| ***4*** | | | |

Groupe [3] :3éme personne du groupe.

Groupe [4] :4éme personne du groupe.

Groupe.nom [3] n'est pas valide, on utilise groupe [3].Nom

***Les enregistrements comme paramètre d'une fonction :***

***Exemple***

*Structure time*

*Heure : entier*

*Minute : entier*

*Seconde : entier*

*Fin structure*

*Fonction nbminute(temp :time) :entier*

*Var nbminute : entier*

*Début*

*Nbminute=temp.heure*60*

*Ecrire (nbminute)*

*fin*

## B /Les fichiers

Un fichier est un ensemble d'enregistrement de m type qui sont stockés sur disque dur ou un cd ;

On distingue deux types :

- fichier à accès séquentiel.
- fichier à accès direct.

➢ ***Les fichiers à accès séquentiel :***

**Déclaration :**
**Syntaxe :**

***En algorithmique:***

*type_fichier = fichier de type _composant*

***En pascal***

*Type_nom fichier=file of type_composant*
*Var nom_logique=nom-fichier*

**Exemple :**

*Type étudiant=enregistrement*
*Nom, groupe : chaine*
*Fin structure*
*Master1=file of étudiant*
*Var S1 :master1*

**Différents opérations effectuées sur le fichier**
**a- Association :**

Les utilisateurs mettre les données dans le fichier qui possède deux nom : nom externe ou physique qui existe sur les périphériques d'e/s et un nom interne ou logique qui existe sur programme, donc il faut associer le nom logique au nom physique.

**Syntaxe :**

*Associer (nom logique)(nom physique)*

**En pascal**

*Assign (nom logique)(nom physique)*

**Remarque :**

Dans le cas ou le fichier ne trouve pas le chemin d'accès vers le programme ou le nom logique, il affiche une erreur.

**b- Ouverture et fermeture**

**Syntaxe :**

*Ouvrir (nom logique)*

**En pascal**

*Reset (nom logique)*

c- **Fermeture :** Pour fermer les fichiers utilisés on utilise :

*Fermer (nom logique)*

**En pascal**

*Close (nom logique)*

Si le fichier n'existe pas, on va le crée par :

*Recrée (nom logique)*

**En pascal**

*Rewrite (nom logique)*

**d- Ecriture dans un fichier :**

*Ecrire (nom logique, variable)*

***En pascal***

*Write (nom logique, variable)*

**c- Lecture**

*Lire (nom logique, variable)*

***En pascal :***

*Read (nom logique, variable)*

## ➢ Les fichiers à accès direct :

Pour positionner le pointeur de Fichier sur l'enregistrement numéro N.

### Syntaxe :

*Pointer (nom logique, numéro)*

**En pascal :**

*Seek (nom_logique, numero)*

Pou connaitre le nombre total d'enregistrements du fichier on utilise la fonction :

*Taille_fichier (nom logique)*

***En pascal :***

*Filesize (nom_logique) ;*

### *Fonction position_fichier :*

Est une fonction qui retourne la position du pointeur du fichier spécifié par l'instruction :

*Position_fichier(nom logique)*

***En pascal :***

*Filepos (nom_logique) ;*

### *Procédure effacé :*

*Effacer (nom logique)*

***En pascal :***

*Erase (nom_logique) ;*

### *Procédure vide_buffer :*

*Renommer (ancien nom logique, nouveau nom logique)*

**En pascal**

*Rename (ancien_nom_logique, nouveau_nom_logique) ;*

***Procédure tronquer***

Pour tronquer un fichier on utilise l'instruction :

*Tronquer (nom logique)*

***En pascal :***

*Truncate (nom_logique) ;*

- ***Les fichiers texte :***

**En Algorithmique :**

*Type nom_logique :texte ;*

**En Pascal :**

*Var nom_logique : texte ;*

*On cite quelque fonction qui sont effectuées sur les fichiers texte :*

***Foction fin_ligne :***

*Fin_ligne (nom logique)*

**En pascal :**

*Eoln (nom_logique) ;*

***Fonction chercher_fin_ligne :***

*Chercher_fin_ligne (nom logique )*

**En pascal**

*Seekeoln (nom_logique)*

***Fonction chercher fin fichier :***

*Chercher_fin_fichier (nom logique)*

**En Pascal**

*Seekeof (nom_logique)*

## ***Procédure ajouté :***

*Ajouter (nom logique)*

**En Pascal :**

*Append (nom_logique) ;*

# ***Application :***

## **Exercice N°1 :**

1- Traduire cet algorithme en langage pascal.
2- Étudier et tester cet exemple sur la machine pour comprendre.

*Algorithme saisie*

*Type personne = Enregistrement*

*Nom, Prénom : chaine*

*Age : entier*

*Ville, emploi: Chaine*

***Fin***

***Variables***

*T :* ***Tableau*** *[1..1000] de personne*

*i,N : entier*

***Début***

*Lire (N) { N est le nombre de personne}*

***Pour*** *i allant de 1* ***à*** *N* ***Faire***

*Lire(T[i]. Nom)*

*Lire(T[i]. Prénom)*

*Lire(T[i]. Age)*

*Lire(T[i]. Ville)*

*Lire(T[i]. emploi)*

***Fin Pour***

***Pour*** *I = 1* ***à*** *N* ***Faire***

*Ecrire(T[I]. Nom)*

*Ecrire(T[I]. Prénom)*

*Ecrire(T[I]. Age)*

*Ecrire(T[I]. Ville)*

*Ecrire(T[I]. emploi)*

***Fin Pour***

***FIN***

***Solution :***

***Program*** *saisie;*

*Type personne = Record ;*

*Nom, Prenom : string ;*

*Age : integer ;*

*Ville, emploi: string ;*

***End ;***

***Var***

*T :* ***array*** *[1..1000] of personne;*

*i,N : integer;*

***begin***

*writeln('donner la valeur de N=') ;*

*readln(N) { N est le nombre de personne}*

***for*** *i := 1* ***to*** *N* ***do***

***writeln('donner le nom');***

*readln(T[i]. Nom);*

***writeln('donner le prénom');***

*Readln(T[i]. Prénom);*

***writeln('donner l''age');***

*readln(T[i]. Age) ;*

***writeln('donner la ville');***

*readln(T[i]. Ville) ;*

***writeln('donner l'emploi');***

*readln(T[i]. emploi) ;*

*readln ;*

***end.***

***Exercice N°2 :*** *A tester pour comprendre*

```
PROGRAM RESULTATS;
USES WINCRT;
Label 1 ;
TYPE INDIVIDU = RECORD;
Nom : string[20] ;
Prenom : string[20] ;
Tp1 : real ;
Tp2 : real ;
End ;
VAR etudiant : array[1..100] OF element ;
Moy : real ;
I,N :integer ;
Fin : Boolean ;
Fichier1 : FILE OF ELEMENT;
BEGIN
ASSIGN(Fichier1,'D:\etudiant.PAS') ;
Rewrite(Fichier1) ;
Writeln('remplir les donnees dans fichier1') ;
Fin := False ;
I :=1 ;
WHILE NOT Fin DO
BEGIN
Writeln('Appuyer sur ENTREE Pour Terminer la saisie') ;
Write('LE NOM : ') ;READLN(etudiant[I].Nom) ;
If (etudiant[I].Nom = '') then goto 1 ;
Write('LE PRENOM : ') ;READLN(etudiant[I].Prenom) ;
Write('NOTE DE TP1 : ') ;READLN(etudiant[I].Tp1) ;
Write('NOTE DE TP2 : ') ;READLN(etudiant[I].Tp2) ;
Write(Fichier1,etudiant[I]) ;
I :=I+1 ;
End ;
 Close(Fichier1) ;
Readln ;Readln ;
END.
```