MODULE I : ALGORITHMIQUE

1. **Eléments du langage de description algorithmiq**ue (LDA)
   1.1. Les éléments de base
      1.1.1. Notion de variable

Une variable est une donnée que l'on est appelé à manipuler dans l'algorithme. Elle peut être de plusieurs types :

Entier :
En algo :
En pascal
Les opérateurs sont : +, - , X , Div, MOD.
Réel :
Caractère
est alphanumérique
0 …….9
A……..Z
a…… Z
***, ,……..**
Opération
Concaténation
'a' + 'b' = 'ab'
(1 caractère) (1 caractere) (Chaîne de caractère)
'ab' est une chaîne de caractère.

Booléen

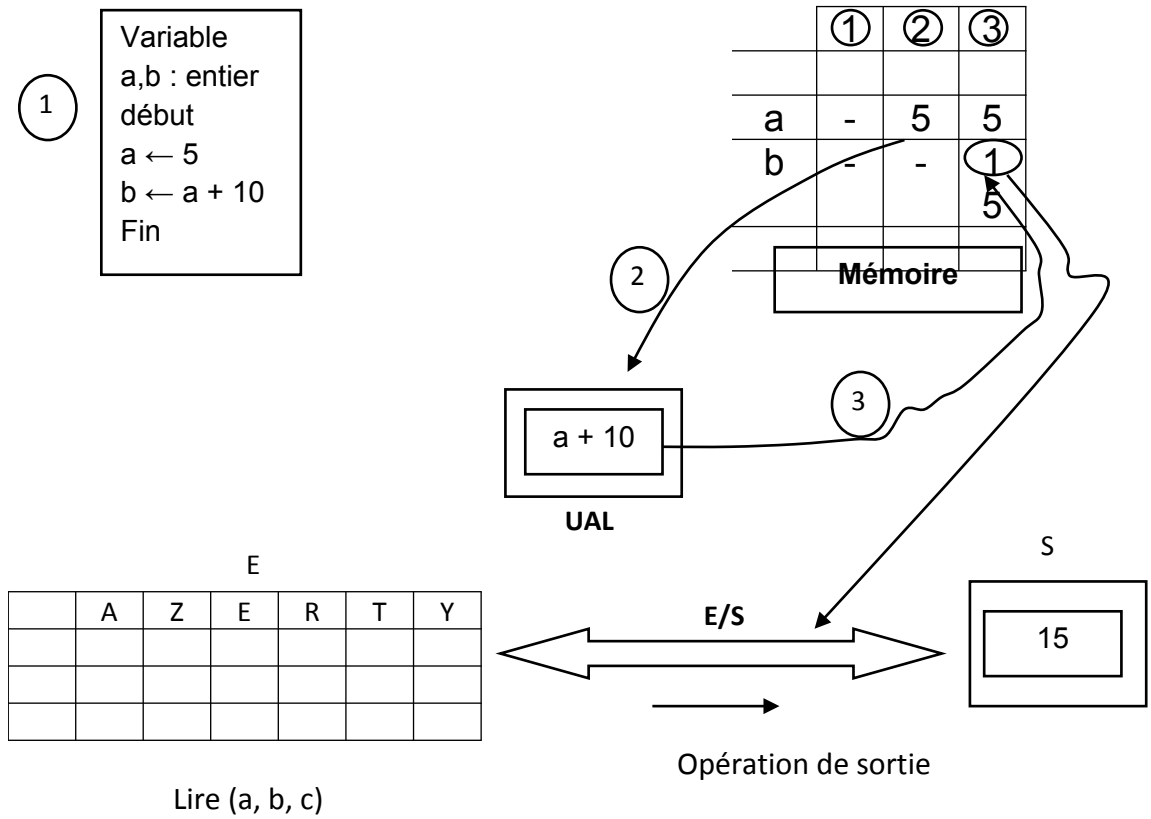
Les opérateurs. NON, OU, ET, ou *

Les comparaisons
>
<
> =
< =

Une variable doit avoir un identificateur. C'est une suite de caractère alphanumérique qui permet d'identifier de façon unique la variable.
Un identificateur ne doit pas commencer par un chiffre, il ne doit pas contenir d'espace et de caractère spéciaux.

Ex : taux : réel *ok*
1er mois : entier *Faux*
Le salaire *Faux*

      1.1.2. Instruction de base (E/S, affectation)



Entrée : Lire (variable 1, …., variable i)
Sortie : Ecrire (variable)

`Ecrire (a)` _____ affiche le contenu de a
Ecrire ('a') _____ affiche a

L'affectation

`Variable ← expression ou valeur`

Exemple :

```
a ← 5
delta ← b*b – 4* a* c
i ← i + 1 Incrémentation
I ← i – 1 Décrémentation
```

1.1.3. Structure d'un algorihtme

```
Algorithme    identification
Constante
  [                        ]
Variable
  [ Declaration des        ]
  [ Déclaration des S/ algorithmes ]
Debut
  [ Instructions           ]
Fin
```

Exemple : un algorithme qui calcule la somme de deux entiers saisis au clavier

```
Algorithme  somme_d_entiers
Variable
     a,b,r : entier
Debut
     Lire (a)
     Lire (b)
     r ← a + b
     Ecrire (r)
Fin
```

1.2. Les structures de contrôles

1.2.1. La conditionnelle

Syntaxe

```
Si (condition) Alors Instruction1
Sinon Instruction2
Finsi
```

Si (condition) est vrai Instruction1 s'exécute
sinon instruction2 s'exécute.

Remarque : `Condition` est une expression logique.
L'instruction 2 peut ne pas exister, dans ce cas la syntaxe est :

```
Si (condition) alors Instruction1
Finsi
```

Exemple :
Ecrire un algorithme pour résoudre une équation du second degré

```
Algorithme  Equation2
Variable
a,b,c, Delta,
Debut
    Lire (a,b,c)
    Delta ← b*b – 4 * a* c.
    Si (Delta <0) alors Ecrire ('Pas de solution dans ')
        Sinon
            Si (Delta >0) alors Ecrire ('2 solution dans ')
                        ← (-b – Sqrt (Delta)) / 2*a
                        ← (-b + Sqrt (Delta)) / 2*a
                       Écrire
            Sinon Ecrire ('solution doublé)
                  ← - b / 2 * a
                      Ecrire
            Finsi
    Fin si
Fin
```

1.2.2 Les structures itératives (répétitives)

1.2.1.1. La boucle

```
Pour indice ← valeur_début  valeur_fin  Faire
     Instruction
Fin Pour indice
```

```
Algorithme  les_dix
Variable
i : entier
Debut
    Pour  i ← 1 à 10  Faire
         Ecrire (i)
    Fin pour 'i'
Fin Algorithme
```

1.2.1.2. Tant que

Le nombre d'étapes est inconnu, mais ou veut commencer les itérations quand une condition est respectée.

```
Tant que (condition) Faire
     Instruction
Fin tant que
```

Un algorithme qui permet de saisir une phrase caractère par caractère principe : la phrase se termine par un point.

```
Algorithme Phrase
Constante
carfin = '•'
Variable
Carlu : caractère.

Début
     Carlu ← ' '
     Tant que (carlu < > carfin) Faire
          Lire (carlu)
     Fin tant que
Fin algorithme
```

Tracer pour : il fait chaud ce soir.

Ligne		Car fin	Carlu		Test condition
3		•	-		-
5		•	-		-
7		•	I I		-
8		•	I I		V
8-1		•	Ilı		V
82 821 822 823		• • • •	Ilı r •		V espace V F

Fin boucle

1.2.1.3. Répéter ……. Jusqu'à

Syntaxe

```
Répéter
Instructions
Jusqu'à (conditions)
```

2. Programmation procédurale

2.1. Notion de S/programme

Algorithme
Constante
Variable

Ressources actions
Ressources données

Début

Instructions

Fin

Un sous algorithme est un algorithme défini au niveau des ressources d'un algorithme principal et qui permet de scinder (diviser) en programme simple, un programme complexe.

S/programme.
Constante
Variable

Début

Fin

Le sous-algorithme a lui-même la structure d'un algorithme. Il existe deux types de S/algorithme (S/programme) :

- Les procédures et les fonctions.

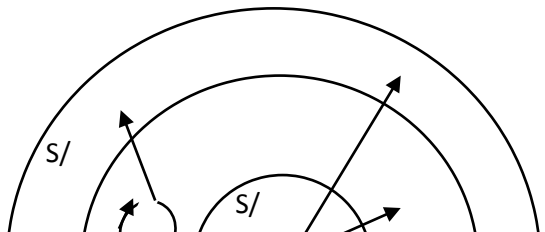
Une procédure est un sous-programme qui ne privilège pas de résultats .

Exemples :

- Le sous-algorithme qui calcule : une fonction.
- Le sous-algorithme qui affiche les éléments d'une matrice (4 X 4) : procédure
- Le sous-algorithme échange deux nombres : (une procédure)
- Le sous-algorithme le max de 5 nombres : (fonction)

La fonction privilégie un résultat du type simple : entier, réel, caractère, booleen

2.2. Visibilité des variables



→ voire
V_1 : locale SP1
V_2 : locale SP2
V_3 : locale SP1

Ex: saisir 2 entiers et les afficher par ordre croissant.

```
Algorithme  range (sans fonction procédure)
Variable
a,b,temp : entier

Début
     Lire (a,b)
     Si (a<b) alors Ecrire (a,b)
          Sinon
               temp ← b
               b ← a
               a ← temp
               Ecrire (a,b)
     Fin si
Fin
```

```
Algorithme  range (avec fonction procédure)
Variable
Compare : booléen
a,b : entier

procedure Echange (variable X,Y = entier)
     Variable
          temp: entier
     Déb
          temp ← X
          X ← Y
          Y ← temp
     Fin 'Procédure'


Fonction : Test_sup (m,n : entier) : booléen
Res: booléen
Variable
Début
```

```
     res ← m < n
     retourner (res)
Fin 'fonction'

Début
     Lire (a,b)
     Compare ← Test_sup (a,b)
     Si (compare = vrai) alors Ecrire (a,b)
          Sinon Echange (a,b)
            Ecrire (a,b)
     Finsi
Fin
```

2.3. Passage de paramètres

Posons le problème : soit la procédure incrémenter sans paramètre.

```
Procedure incrémenter( )
Debut
     k ← k + 1
      ← l + 1
Fin
```

Cette procédure ne sait incrémenter que les variables définies au niveau global, elle est incapable d'incrémenter d'autre variables par ex : m et p avec un pas différent de 1. Pour l'écrire de manière beaucoup plus général (n'importe quel variable entière et n'importe quel pas, il faut la réécrire avec des paramètres.

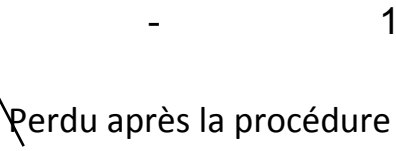
a. Passage de paramètre par valeurs

```
Algorithme     valeur 1
     Variable
     Nbre 1, compte, p : entier
     Procédure incrémenter_p (, pas : entier)
     Début
          Ecrire (
     Fin procédure
     Début
     incrémenter_p (nbre1, 1)
     Ecrire (nbre1)
     P ← -1
     Incrémenter p (compte,p)
     Ecrire (compte)
Fin
```

Trace

Nbre1	compte	p	affichage
0	-	-	-
0	4	-	-

Incrémenter p(0,1)	1	-	1	1
	0	-	1	0



Le passage de paramètre par valeur, consiste en une recopie des valeurs des paramètres effectifs au moment de l'exécution de la procédure incrémenter p (nbre1, 1).
La zone de mémoire créée à l'exécution de la procédure disparait avec ses données modifiées quand celles-ci n'est plus active.

Conclusion partielle
Un passage de paramètre par valeur, ne garde pas les modifications sur les paramètres.

b. Passage de paramètre par adresse

Pour garder les modifications effectuées sur les paramètres, l'entête de la procédure est :
Procédure incrémentation_ p (variable : entier ; pas : entier)
]

Récursivité

2.4. Les fonctions

Syntaxe :

```
Fonction  (paramètres) :
Début
     Retourner (expression)
Fin

Fonction est-pair(a:entier):booleen
variable
res: booleen
Début
|    Si (a mod 2 = 0) alors retourner (vrai)
|    |    Sinon retourner (faux)
|    Finsi
Fin fonction
```

NB: L'activation d'une fonction se fait dans une expression.

3 STRUCTURE DE DONNEES

3.1 Les tableaux

Les tableaux sont des structures de données simples permettant de stocker des données de même type en mémoire de manière continue. Une représentation simplifiée des tableaux serait les matrices.

3.1.1 Tableau à une dimension

Variable

T : tableau de Entier

Identificateur du tableau — indice_max

T

15
-20
19
61
100
64
66
93
97
20

T = 15

T = 64

T

Indice de la cellule

- Initialisation

```
Procédure init_tableau (Variable t : tableau [1..10] de entier)

Variable
i : entier
Début
    Pour i ← 1 à 10 Faire
        Écrire ('saisir T [', I, ']')
        Lire(T[i])
    Fin pour
Fin procédure
```

Les autres actions sur les tableaux :

- Recherche du minimum, du maximum..
- Les méthodes de tri ..

3.1.2 Tableau à 2 dimensions

Syntaxe

```
Mot2 : tableau [1..ligne,1..colonne] de entier
```

```
Procédure init_mat2(variable M : Tableau [1..3,1..4] de entier)
Variable
i, j : entier
Début
    Pour i ← 1 à 3 Faire
        Pour j ← 1 à 4 faire
            Lire (M[i,j])
        Fin pour j
    Fin pour i
Fin procédure
```

3.2. Les enregistrements

3.2.1 Définition

Syntaxe

```
Fin enregistrement
```

Exemple (1) :

```
Variable
Etudiant : enregistrement
Num_carte : entier
Prénom : Chaîne [25]
Nom : Chaîne [25]
Datenaiss : Chaîne [8]
Taille : Réel
Fin enregistrement

Algorithme
Type
     Etudiant : enregistrement
     Num_carte : entier
     Nom : Chaîne [25]
     Prénom : Chaîne [25]
     Datenaiss : Chaîne [8]
     Taille : Réel
Fin enregistrement
```

Variable

Yao, Konan : Etudiant ;

Les enregistrements contiennent des données qui ne sont pas de même types (contrairement au tableau).

3.2.2. **Tableau d'enregistrement**

On peut aussi créer des tableaux d'enregistrement

N°	Nom	Prix	Quantité
100	OMO 50g	100 F	116
101	BF 4	515 F	87
102	•	•	•

Mag 1[1]. Nom = OMO 50 g
Mag 2 [2]. Prix = 515 F
On pourra appliquer toutes les opérations sur les tableaux.