Chapitre I

Introduction

1. Définitions

- **Un algorithme** est une suite d'instructions, qui une fois exécutée correctement, conduit à un résultat donné. La suite d'opérations sera composée d'actions élémentaires appelées ***instructions***. Un algorithme doit donc contenir uniquement des instructions compréhensibles par celui qui devra l'exécuter.
- ***L'algorithmique*** est la logique d'écrire des algorithmes. Pour pouvoir écrire des algorithmes, il faut connaître la résolution manuelle du problème, connaître les capacités de l'ordinateur en terme d'actions élémentaires qu'il peut assurer et la logique d'exécution des instructions.

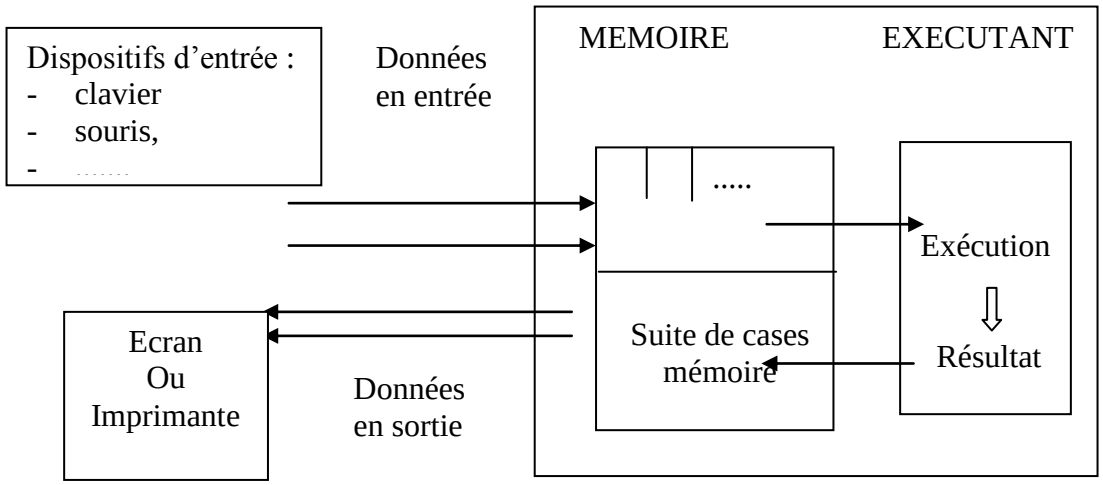
Les étapes de résolution d'un problème :

1. Comprendre l'énoncé du problème
2. Décomposer le problème en sous-problèmes plus simple à résoudre
3. Associer à chaque sous problème, une spécification :

- Les données nécessaires
- Les données résultantes
- La démarche à suivre pour arriver au résultat en partant d'un ensemble de données.

4. Elaboration d'un algorithme.

Illustration du fonctionnement d'un ordinateur



On peut dire que la partie EXECUTANT est le problème de l'algorithmique, et la partie MEMOIRE (stockage de donnée) concerne la matière " Structures de données ".

2. Structure d'un algorithme

ALGORITHME nom_de_l'algorithme

CONST {Définition des constantes}

TYPE {Définition de types}

VAR {Déclaration de variables}

DEBUT

{Suite d'instructions}

FIN

Exemple :

ALGORITHME AFFICHER

DEBUT

Ecrire ("La valeur de 3*5 est ", 3*5)

FIN

Cet algorithme permet d'afficher sur l'écran la phrase suivante : La valeur de 3*5 est 15

Exemple 2

On veut écrire l'algorithme qui permet de saisir 3 notes d'un étudiant dans trois matières, étant donnés les coefficients respectifs 2, 3 et 1.

Résolution

A partir de l'énoncé du problème, nous recherchons la solution par une démarche en 2 phases.

- On doit comprendre comment le résoudre manuellement,
- Définir ce qu'on a besoin comme ***données***, quelles est la démarche à suivre (***formules de calcul)*** pour arriver aux ***résultats.***

Pour notre problème, nous connaissons les coefficients et la formule de calcul ($\sum N_i * C_i / \sum C_i$), nous avons besoins des notes de chaque matière ***dans l'ordre,*** et enfin nous pouvons communiquer le résultat à l'utilisateur.

ALGORITHME MOYENNE
CONST C1=2
C2=3
C3=1
VAR
N1, N2, N3 : REEL
MOY: REEL
DEBUT
ECRIRE (" Donner trois valeurs réelles ")
LIRE (N1, N2, N3)
MOY← (N1*C1+N2*C2+N3*C3) / (C1+C2+C3)
ECRIRE (" La moyenne est = ", MOY)
FIN

Chapitre II

Les actions algorithmiques simples

0. Concepts de base

Dans tout ce qui suit, pour présenter les syntaxes, on suit les règles suivantes :

Ce qui est entre les crochets est optionnel.

La suite des points de suspensions "…" veut dire que ce qui précède peut se répéter plusieurs fois.

Le symbole " | " veut dire : " ou bien ".

Les mots en majuscule sont des mots réservés.

Ce qui est entre accolades est un commentaire, pour la lisibilité des algorithmes.

1. L'affichage : ECRIRE

Cette action permet de communiquer un résultat ou un message sur écran ou sur imprimante pour l'utilisateur.

Syntaxe ECRIRE (paramètre1 [[,paramètre2]…])

Paramètre = variable | expression | constante

Constante = nombre | message

Exemples

ECRIRE (" La valeur de 3*2 est égale à ", 3*2)

↑ ↑

message expression

ECRIRE (" La moyenne est = ", MOY)

↑

Variable

2. La saisie des données : LIRE

La saisie permet à l'ordinateur d'acquérir des données à partir de l'utilisateur, dans des cases mémoires bien définies (qui sont les variables déclarées).
Les variables sont des cases mémoire, supposées contenir un type de données, nommées par le nom de variable.

Syntaxe LIRE (variable1 [[, variable2] …])

Remarque :

1. La saisie se fait uniquement dans des variables. Ce sont les cases (cellules) qui pourront accueillir les données correspondantes.
2. La donnée à introduire doit être de même type que la variable réceptrice.

3. Les expressions arithmétiques

Parmi les opérateurs, on distingue les fonctions et les opérateurs.

Les fonctions

- La fonction **DIV** permet de donner le résultat de la division entière d'un nombre par un autre. 7 DIV 2 → 3
- La fonction **MOD** (se lit Modulo), permet de donner le reste de la division entière d'un entier par un autre. 7 MOD 2 → 1
- La fonction ****** ou ^ permet d'élever un nombre à la puissance d'un autre. 2**3 →8

Les opérateurs

- Sont le "+", "-", "/", "*" et le "-" un aire.

Ordre de priorité

Les opérateurs suivants sont ordonnés du plus prioritaire au moins prioritaire dans l'évaluation d'une expression arithmétique.

1- Les parenthèses
2- "- " un aire
3- Les fonctions
4- Les opérateurs de multiplication " * " et de division " / "
5- Les opérateurs d'addition " + " et de soustraction " - "

Remarque

Si l'ordre entre les opérateurs dans une expression est le même, on évalue l'expression de gauche à droite.

Exemples

3**2+4 = 9+4=13

3**(2+4)=3**6 car les parenthèses sont plus prioritaires

17 MOD 10 DIV 3=(17MOD10)DIV3=7DIV3=2

4. L'affectation

C'est l'action de charger une valeur dans une variable. Cette valeur peut elle-même être une variable, le résultat d'une expression arithmétique ou logique ou une constante.

Syntaxe

Variable1 ← variable2 | expression | constante

A ← B **se lit** " A reçoit B "

Le résultat de cette action est de mettre le contenu de la variable B dans la variable A. Si B était une expression, elle aurait été évaluée, ensuite sa valeur est transférée dans la variable réceptrice (à notre gauche).

Remarque

L'affectation ne vide pas la variable émettrice (à notre droite) de sa valeur. Par contre, le contenu de la variable réceptrice est écrasé et remplacé par la nouvelle valeur.

Exemple

Ecrire l'algorithme qui permet de calculer le discriminant Δ (delta) d'une équation du second degré.

TD Algorithmique I

Les actions algorithmiques simples

Exercice 1

Soit l'algorithme suivant :

```
ALGORITHME EQUATION2D
VAR a,b,c : REEL
        delta : REEL
DEBUT
  Ecrire("Donnez la valeur du premier paramètre")
  Lire(a)
  Ecrire("Donnez la valeur du second paramètre")
  Lire(b)
  Ecrire("Donnez la valeur du troisième paramètre")
  Lire(c)
  delta ←b^2 b * b - 4a * c
  Ecrire(" le discriminant est = Δ ")
Fin
```

1 - Décrire cet algorithme en détail (ligne par ligne), en donnant les éventuelles erreurs.

2 - Quelles sont les valeurs de delta dans les cas suivants :

a=2 b=-3 c=1

a=1 b=2 c=2

Exercice 2

Ecrire l'algorithme permettant de saisir l'abscisse d'un point A et de calculer son ordonné $f(x)= 2\,x^3 - 3x^2 + 4$

Evaluer le résultat en expliquant les ordres de priorité pour x=-2.

Exercice 3

Ecrire l'algorithme qui permet de permuter les valeurs de A et B sans utiliser de variable auxiliaire.

Exercice 4

Faire l'algorithme qui lit les coordonnées de deux vecteurs u et v, et de calculer leur norme et leur produit scalaire.

Exercice 5

Ecrire l'algorithme qui permet de saisir les paramètres d'une équation du second degré et de calculer son discriminant Δ.

Exercice 6

Ecrire l'algorithme permettant de calculer et d'afficher le salaire net d'un employé. Sachant que :

- Le salaire net = Salaire brut – Valeur de l'impôt – Valeur de CNSS
- Salaire brut = (Salaire de base + Prime de technicité + Prime de transport + Prime des enfants) * Taux de travail
- Taux de travail = Nombre de jours travaillés / 26
- Prime des enfants = Prime d'un enfant * Nombre d'enfants
- Valeur de l'Impôt = Taux de l'Impôt * Salaire Brut
- Valeur de CNSS = Taux de CNSS * Salaire Brut
- Taux CNSS = 26,5%
- Taux Impôt = 2%

Indication :

Décrire l'environnement de travail : toutes les variables en entrée, en sortie et de calcul.

Chapitre III

Les structures conditionnelles

1. Introduction

Nous présentons dans ce chapitre les notions logiques utilisées pour résoudre des problèmes dont la solution ne peut être décrite par une simple séquence d'actions mais implique un ou plusieurs choix entre différentes possibilités.

Ce sont les ***structures conditionnelles*** qui le permettent, en se basant sur ce qu'on appelle ***prédicat*** ou ***condition***.

2. Notion de PREDICAT

Un prédicat est une proposition qui peut être vrai ou fausse selon ce qu'on est entrain de parler.

En mathématiques, c'est une expression contenant une ou plusieurs variables et qui est susceptible de devenir une proposition vraie ou fausse selon les valeurs attribuées à ces variables.

Exemple :

(10 < 15) est un prédicat vrai

(10 < 3) est un prédicat faux

2.1. Evaluation d'une expression logique

Une condition est une expression de type logique. Ils lui correspondent deux valeurs possibles VRAI et FAUX qu'on note par V ou F.

Considérons deux variables logiques A et B. Voyons quels sont les opérateurs logiques et leurs ordres de priorités.

Notons que

- ✓ (A = faux) ⇔ non A
- ✓ (A = vrai) ⇔ A

Les opérateurs logiques sont :

- ✓ La négation : "non"
- ✓ L'intersection : "et"
- ✓ L'union : "ou"

2.2. Notation et Ordre de priorité des opérateurs logiques

1. non : ¬
2. et : ∧
3. ou : ∨

Tableaux d'évaluations

La négation d'une condition

A	**Non A**
Vrai	Faux
Faux	Vrai

L'intersection de deux conditions

A et B	**Vrai**	**Faux**
Vrai	Vrai	Faux
Faux	Faux	Faux

L'union de deux conditions

A ou B	**Vrai**	**Faux**
Vrai	Vrai	Vrai
Faux	Vrai	Faux

Théorème de DE MORGAN

- ✓ $\neg(A \wedge B) \Leftrightarrow \neg A \vee \neg B$
- ✓ $\neg(A \vee B) \Leftrightarrow \neg A \wedge \neg B$

3. La structure conditionnelle SI

Syntaxe

SI <Condition> ALORS

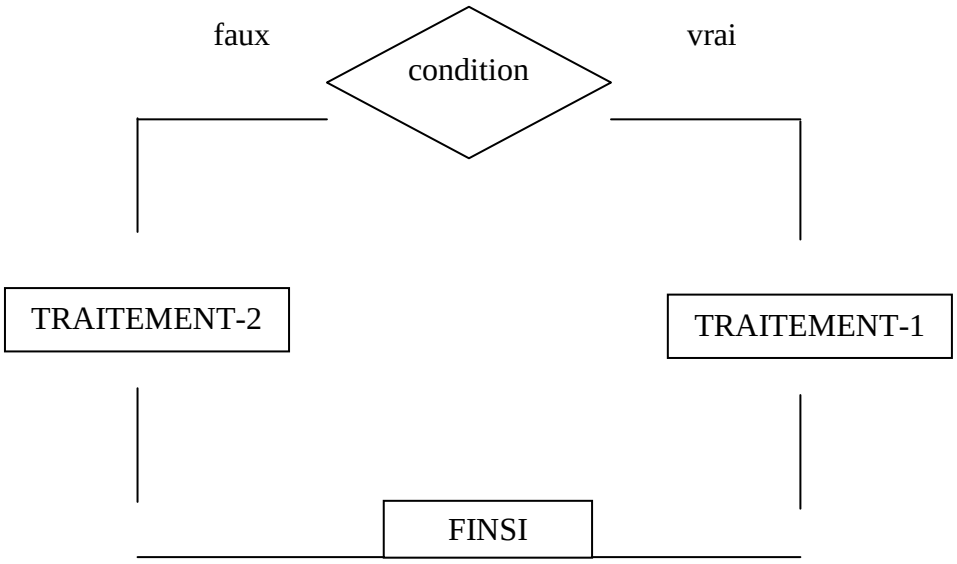
<suite d'action(s)-1>

[SINON

<suite d'actions(s)-2>]

FINSI

Format Organigramme



- La <condition> est un prédicat, qui peut être vrai ou faux, selon les valeurs des paramètres la constituant.
- Si la condition est vérifiée (sa valeur est vrai), c'est la <suite d'actions-1> qui sera exécutée. Ensuite, le système passe à l'exécution juste après le FINSI.
- Dans le cas contraire, lorsque la condition n'est pas vérifiée (valeur de la condition est faux), c'est la <suite d'actions-2> qui s'exécute, en cas où celle ci existe (facultative). Si elle n'existe pas, le système passe directement à l'instruction qui suit le FINSI.
- Les suites d'actions 1 et 2, peuvent être des actions simples ou même des structures conditionnelles.

Exemple 1

Lire un nombre réel, et dire s'il est positif ou strictement négatif.

```
ALGORITHME POS-NEG
VAR A : réel
DEBUT
        ECRIRE ("Donner un nombre ")
        LIRE(A)
        SI (A < 0) ALORS
                ECRIRE (A, " est négatif ")
        SINON
                ECRIRE (A, " est positif ")
        FINSI
FIN
```

Autrement :

```
ALGORITHME POS-NEG-1
VAR A : réel
        B : logique
DEBUT
        ECRIRE ("Donner un nombre ")
        LIRE(A)
        B ← (A < 0)
        SI (B) ALORS
                ECRIRE (A, " est négatif ")
        SINON
                ECRIRE (A, " est positif ")
        FINSI
FIN
```

Dans cet exemple, on a déterminé uniquement les cas de positivité ou de négativité, et on n'a pas déterminé le cas où A est nulle.

```
ALGORITHME POS-NEG-NUL
VAR A : réel
DEBUT
        ECRIRE ("Donner un nombre ")
        LIRE(A)
        SI (A < 0) ALORS
                ECRIRE(A, " est négatif ")
        SINON {A >= 0}
          SI (A > 0)ALORS
                ECRIRE(A, " est positif ")
          SINON {A = 0}
                ECRIRE (A, "est nulle")
      FINSI
        FINSI
FIN
```

Exemples

1) Ecrire l'algorithme qui permet de déterminer si un entier lu est pair ou impair.
2) Ecrire l'algorithme qui permet de saisir deux nombres A et B et de déterminer si la valeur de A est supérieure, inférieure ou égale à B.

4. La structure conditionnelle SELON

Cette structure conditionnelle est appelée aussi ***à choix multiple*** ou ***sélective*** car elle sélectionne entre plusieurs choix à la fois, et non entre deux choix alternatifs (le cas de la structure SI).

Syntaxe

```
SELON (sélecteur) FAIRE
Cas <liste de valeurs-1> : <suite d'action (s)-1>
Cas <liste de valeur-2> : <suite d'action (s)-2>
…….                                    ]
[SINON : <suite d'action (s)-n>        ]
FINSELON
```

Le ***sélecteur*** peut être une variable de type scalaire ou une expression arithmétique ou logique.

La structure SELON évalue le "sélecteur", passe à comparer celui ci respectivement avec les valeurs dans les listes. En cas d'égalité avec une valeur, les actions correspondantes, qui sont devant cette valeur seront exécutées.

Devant "Cas", il peut y avoir une seule valeur, une suite de valeurs séparées par des virgules et/ou un intervalle de valeurs.

Après avoir traité la suite d'actions correspondante, l'exécution se poursuit après le FINSELON.

Remarque

1. Le sélecteur doit avoir le même type que les valeurs devant les cas.
2. Le type de ces valeurs ne doit être, ni réel ni chaîne de caractères.

Exemple

Ecrire l'algorithme qui permet de saisir un numéro de couleur de l'arc-en-ciel et d'afficher la couleur correspondante : 1: rouge, 2 : orangé, 3 : jaune, 4 : vert, 5 : bleu, 6 : indigo et 7 : violet.

TD Algorithmique II

Les structures conditionnelles

Exercice 1

Evaluer les expressions logiques suivantes, avec (a, b, c, d) = (2, 3,5, 10) et (X, Y) = (V, F).

1) $(a < b) \wedge (a < c)$	2) $\neg ((a < b) \wedge (a < c))$	3) $\neg (a < b) \wedge (a < c)$
4) $(a < c) \wedge (c = d/2)$	5) $(d / a = c) = Y$	6) $(d / c = b) = Y$
7) $(d / c = b) = X$	8) $(a < b) \wedge (d < c)$	9) $(a < b) \wedge (d < c) = X$

Exercice 2

Ecrire l'algorithme permettant de lire la valeur de la température de l'eau et d'afficher son état :

GLACE Si la température inférieure à 0,

EAU Si la température est strictement supérieure à 0 et inférieure à 100,

VAPEUR Si la température supérieure à 100.

Exercice 3

Ecrire l'algorithme qui permet de saisir deux nombres A et B et de déterminer si la valeur de A est supérieure, inférieure ou égale à B.

Exercice 4

Ecrire l'algorithme qui permet de saisir deux nombres, et un opérateur et d'évaluer l'expression arithmétique correspondante.

Exercice 5

Ecrire l'algorithme permettant de lire la valeur de la variable DEVINETTE et d'afficher parmi les messages suivants celui qui correspond à la valeur trouvée :

ROUGE si la couleur vaut R ou r

VERT si la couleur vaut V ou v

BLEU si la couleur vaut B ou b

NOIR pour tout autre caractère.

Exercice 6

Ecrire l'algorithme qui permet de saisir le jour, le mois et l'année d'une date (Mois : numéro du mois), et de déterminer si elle est correcte ou non, et où est l'erreur.

Exercice 7

Ecrire l'algorithme qui lit un entier positif inférieur à 999 (composé de trois chiffres au maximum) et d'afficher le nombre de centaines, de dizaines et d'unités.

Chapitre IV

Les structures répétitives

1. Introduction

La structure répétitive (ou itérative) permet la répétition d'une suite d'actions un nombre fini de fois. Ce nombre peut être déterminé soit par le programmeur, soit par l'utilisateur, soit par un calcul effectué par le programme. Cette structure est aussi appelée "**boucle** ".

2. La boucle POUR … FIN POUR

C'est la structure la plus utilisée. Elle a pour rôle d'exécuter un traitement un certain nombre de fois, on va se servir d'un compteur (i) qui sera initialisé automatiquement à la valeur initiale et sera incrémenté de la valeur du pas jusqu'à la valeur finale.

Syntaxe :

POUR i de <expression de début> à <expression de fin> [Pas de <expression pas>]
<Instructions>
FIN POUR

Exemple :

Algorithme Moyenne1
VAR Note, Som, Moyenne : réel
i : ENTIER
DEBUT
Som := 0
i:= 0
POUR i de 1 à 5
Ecrire ("Entrez une note :")
Lire (Note)
Som <- Som + Note
FIN POUR
Moyenne := Som / i

Ecrire («La moyenne des notes est : ", Moyenne)

FIN

3. La boucle TANT QUE … FIN TANT QUE

Cette boucle permet de répéter un traitement particulier un certain nombre de fois ***tant que*** une condition est vraie.

Le nombre d'itération n'est pas connu à priori. Il dépend de la condition.

Syntaxe :

TANT QUE …Condition… FAIRE

…

<Instructions>

…

FIN TANT QUE

Tant que la condition est vérifiée (VRAIE), on exécute le corps de la boucle. L'arrêt se produit lorsque la condition n'est plus vérifiée.

Exemple :

Algorithme Moyenne2

VAR Note, Som, Moyenne : réel

Nb : ENTIER

DEBUT

Som:= 0

Nb:= 0

Ecrire ("Entrez une note :")

Lire (Note)

TANT QUE Note >= 0 FAIRE

Nb <- Nb + 1

Som <- Som + Note

Ecrire ("Entrez une note :")

Lire (Note)

FIN TANT QUE

Moyenne := Som / Nb

Ecrire («La moyenne des notes est : ", Moyenne)

Fin

4. La boucle REPETER … JUSQU'A CONDITION

Cette structure permet de rentrer dans la boucle quelque soit l'état de la condition (VRAIE ou FAUSSE) et les instructions contenues dans le corps de la boucle seront exécutées.

Syntaxe :

REPETER

<Instructions>

JUSQU'A (condition d'arrêt)

Cet ordre d'itération permet de répéter les <Instructions> une ou plusieurs fois et de s'arrêter sur une condition. En effet, lorsque la condition est vérifiée, la boucle s'arrête, si non elle ré-exécute les <Instructions>.

Exemple :

Algorithme Moyenne3

VAR Note, Som, Moyenne : réel

Nb : ENTIER

DEBUT

Som :=0

Nb := 0

REPETER

Ecrire ("Entrez une note :")

Lire (Note)

Nb :=Nb + 1

Som:= Som + Note

JUSQU'À Note < 0

Moyenne := Som / Nb

Ecrire («La moyenne des notes est : ", Moyenne)

FIN

Remarques

- Dans cette boucle, le traitement est exécuté au moins une fois avant l'évaluation de la condition d'arrêt.
- Il doit y avoir une action dans le bloc des instructions qui modifie la valeur de la condition.

TD Algorithmique III

Les structures répétitives

Exercice 1

Ecrire l'algorithme qui permet d'afficher les N premiers entiers impairs dans l'ordre décroissant.

Exercice 2

Ecrire l'algorithme qui permet d'afficher les diviseurs d'un entiers N.

Exercice 3

Ecrire l'algorithme qui permet de saisir un entier N et d'afficher s'il est premier ou non. Un nombre est dit premier s'il est divisible uniquement par 1 et par lui-même.

Exercice 4

Ecrire un algorithme qui permet de calculer la factorielle d'un entier N donné.

Exercice 5

Ecrire un algorithme qui permet de saisir des entiers alternatifs (si l'un est positif sont suivant doit être négatif et vice versa).

Exercice 6

Ecrire un algorithme qui demande un nombre de départ, et qui ensuite affiche les dix nombres suivants.

Par exemple, si l'utilisateur entre le nombre 17, le programme affichera les nombres de 18 à 27.

Chapitre V

Les tableaux

I. Les tableaux à une dimension

1. Problématique

1) Saisir la moyenne de 30 étudiants

2) Effectuer leur classement

- **Réponse**

pour i de 1 à 30 faire

Ecrire (" Donner la moyenne de l'étudiant N°",i)

Lire (moyenne)

Fin pour

- ***Conclusion*** : On ne peut pas effectuer le classement parce qu'on ne garde pas les moyennes précédentes et la variable moyenne contient uniquement la dernière valeur.

Il est plus convenable, alors, de définir un espace mémoire qu'on appelle MOY qui sera divisé en 30 parties équitables, indicées de 1 à 30.

MOY

Contenu →	15	12	5	10	4	50	….						
Indice →	1	2	3	4	5	6	7	8	9	10	11	12	13

On définit un tableau de 30 cases à une seule dimension qu'on appelle **Tableau.**

2. Définition

Un tableau est une structure de données linéaires qui permet de stocker des données de même type. Chacune des valeurs est repérée par un indice indiquant la position de la donnée dans le tableau.

- **Déclaration :**

VAR nom_tableau : tableau [indice min ……. indice max] de type de données

Exemple : CONST Bi=1

Bs=30

VAR MOY : Tableau [bi.. bs] de réel

i: entier

MOY[i]

↑ ↖ indice d'un élément du tableau

variable qui indique le nom du tableau

MOY[i] : représente l'élément du tableau MOY occupant l'indice " i ".

- **Création d'un tableau :**

Elle consiste en un remplissage des différentes cases qui le constituent.

On doit utiliser une boucle qui permet de saisir à chaque entrée dans la boucle la $i^{ième}$ case.

Le nombre d'éléments à saisir ne doit pas dépasser la taille du tableau pour ne pas déborder sa capacité.

On appelle dimension d'un vecteur le nombre d'éléments qui constituent ce vecteur.

Exemple : écrire un algorithme permettant de saisir les moyennes de 30 étudiants et d'afficher la moyenne de la classe.

```
ALGORITHME          Moyenne
CONST        N = 30
VAR
        MOY : Tableau[1..N] de réels
         i : entier
         SMOY,MOYG : réel
Début
 Pour i de 1 à N  Faire
        Ecrire (" donner la moyenne de l'étudiant N° " , i )
        Lire ( MOY [i])
Fin Pour
```

```
SMOY ← 0
Pour i de 1 à N Faire
        SMOY←SMOY+MOY[i]
Fin Pour

MOYG ← SMOY / 30
Ecrire (" La moyenne de la classe est ", MOYG)
Fin
```

- **Affichage d'un tableau :**

Il s'agit d'afficher les valeurs des cases du tableau.

Exemple : afficher le contenu du tableau MOY précédant.

```
Pour i de 1 à N faire
    Ecrire (MOY[i])
Fin Pour
```

3. Recherche dans un tableau

- **Recherche séquentielle**

On peut chercher le nombre d'apparition d'un élément dans un tableau, sa ou bien ses positions. Pour cela, on doit parcourir tout le tableau élément par élément et le comparer avec la valeur de l'élément à chercher.

Exemple : Chercher la position de la première occurrence d'un élément e dans un vecteur V contenant N éléments. (On suppose que le vecteur est défini)

```
i ← 1
Trouv ← vrai
Tant que ((i <= N) et (Trouv = vrai))  Faire
            Si  (V[i] = e ) Alors
                    Trouv ← Faux
            Sinon
                    i ← i +1
            Fin Si
```

```
        Fin Faire
Si (Trouv = vrai) Alors
        Ecrire(e, "se trouve à la position" , i)
Sinon
        Ecrire(e, "ne se trouve pas dans V")
           Fin Si
```

- **Recherche dichotomique**

Ce type de recherche s'effectue dans un tableau ordonné.

Principe

1. On divise le tableau en deux parties sensiblement égales,
2. On compare la valeur à chercher avec l'élément du milieu,
3. Si elles ne sont pas égales, on s'intéresse uniquement à la partie contenant les éléments voulus et on délaisse l'autre partie.
4. On recommence ces 3 étapes jusqu'à avoir un seul élément à comparer.

Exemple : On suppose qu'on dispose d'un tableau V de N éléments. On veut chercher la valeur Val.

```
ALGORITHME DICHOTHOMIE
...
Inf ← 1
Sup ← N
Trouv  ← vrai
Tant que ((Inf <= Sup) et (Trouv = vrai))
Faire
        Mil← (Inf+Sup)DIV 2
        Si (V[Mil] = Val) Alors
                Trouv← faux
        Sinon
                Si (V[Mil] < Val) Alors
                        Inf ← Mil + 1
                Sinon
                        Sup ← Mil -1
```

```
            Fin Si
        Fin Si
Fin Faire

Si (Trouv = faux) Alors
        Ecrire(Val, "existe à la position" , Mil)
Sinon
        Ecrire(Val, "n'existe pas dans V" )
Fin Si
```

II. Les tableaux à deux dimensions

Chaque élément est repéré par deux indices indiquant la ligne et l'autre indiquant la colonne.

- **Déclaration :**

VAR nom_tableau : tableau [indminL... indmaxL] [indminC... indmaxC] de type de données

Exemple : Soit un tableau de 3 lignes 4 colonnes :

	1	**2**	**3**	**4**
1	k	l	m	n
2	C	v	h	f
3	q	z	t	o

Ecrire un algorithme permettant de saisir des valeurs dans ce tableau.

Algorithme saisie

```
Var    M : tableau [1..3, 1..4] de caractère
       i,j : entier
Début

        pour i de 1 à 3 faire
            pour j de 1 à 4 faire
                Écrire («  donner une valeur »)
```

lire (M [i,j])

Fin pour

Fin pour

Fin

. LE TYPE CHAINE DE CARACTERE

II.1 Définition

Une variable de type chaîne est une succession de caractères. Une chaîne ne contenant aucun de caractère est appelée *chaîne vide*.

II.2 Déclaration d'une chaîne

VARIABLES
Chn : CHAINE
St : CHAINE[10]

- Chn est une chaîne qui peut contenir jusqu'à 255 caractères
- St est une chaîne qui peut contenir au maximum 10 caractères.

les chaînes de caractères

❶ La concaténation

C'est l'assemblage de deux chaînes de caractères.

```
ALGORITHME concat
VARIABLES
    chn1, chn2, chn3  : CHAINE
DEBUT
    chn1 ← 'Turbo '
    chn2 ← 'PASCAL'
    chn3 ← chn1 + chn2
    ECRIRE(chn) ⇨ Affiche 'Turbo PASCAL'
FIN.
```

❷ Longueur d'une chaîne

La fonction **LONG(chn)** renvoi le nombre de caractères qui forment la chaîne chn. La longueur d'une chaîne vide est zéro.

```
ALGORITHME Longueur
VARIABLES
    chn  : CHAINE
    n    : ENTIER
DEBUT
    chn ← 'Turbo PASCAL'
    n ← LONG(chn)
    ECRIRE(n) ⇨ Affiche 12
FIN.
```

❸ Accès à un caractère dans une chaîne

Les caractères d'une chaîne sont indicés de 1 à n avec n la longueur de la chaîne. L'accès à un caractère dans la chaîne est réalisé par une référence à la chaîe (identificateur) suivie d'un entier entre crochets indiquant la position du caractère au sein de cette chaîne.

```
ALGORITHME car7
VARIABLES
    chn : CHAINE
    c   : CARACTERE
DEBUT
    chn ← 'Turbo PASCAL '
    c ← chn[7]
    ECRIRE(c) ⇨ Affiche 'P'
```

FIN.