Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Dr. Tahar Moulay de Saida

Faculté de Technologie

Département d'informatique

Algorithmique et structures de données 1

Présenté par :

Dr. BENYAHIA Kadda

Maître de conférences « B » en Informatique

Février 2020

Avant-propos

Le contenu de ce polycopie est issu d'un cours qui s'adresse au étudiant de première année scol-commun math et informatique. Il tient en compte des orientations du nouveau programme pédagogique défini par le conseil national des programmes et s'adresse aussi à tout lecteur désireux de s'initier à la construction d'algorithmes qui seront ensuite codés dans un langage de programmation.

Au niveau du cours, et pour des raisons de simplification, nous utilisons des notations claires (la syntaxe du pseudo-code des algorithmes) reprend celui couramment utilisé dans les écoles d'informatique et les autres ouvrages.

Nous avons traité l'algorithme comme un ensemble d'actions et nous avons détaillé chaque action a part afin de permettre aux lecteurs de comprendre son objectif, sa syntaxe et ses différentes manières d'utilisation.

Mots-clés du Support

Algorithme, programme, Structure de données, Actions de base, Variables, les actions de test, les boucles, les Fonctions, les Procédures, les fonctions, les Enregistrements,

Table des matières

Avant-propos	i
Table des matières	ii
1. L'ordinateur	
1.1 Introduction	1
1.2 L'architecture de Von Neuman	1
1.3 Représentation des informations	2
1.4 Les étapes de résolution d'un problème par ordinateur	2
1.4.1 La définition du problème	2
1.4.2 Analyse du problème	2
1.4.3 Elaboration d'un algorithme	3
1.4.4 Traduction des algorithmes en langage de programmation	3
1.4.5 Compilation et exécution	3
2. L'algorithme	
2.1 Qu'est ce qu'un algorithme ?	4
2.2 Structure générale d'un algorithme	5
2.2.1 Le nom de l'algorithme	5
2.2.2 La partie déclaration	6
2.2.3 Corps de l'algorithme	6
2.3.La représentation graphique (Organigramme)	7
2.4 Les Types De Données	8
2.4.1 Type entier	8
2.4.2 Type Réel	9
2.4.3 Type Booléen	10
2.4.4 Type caractère	10
2.5 Variables	12
2.5.1 Notion de variable	12
2.5.2 Déclaration des variables	12

2.6 Constantes 13
2.6.1 Notion de constante 13
2.6.2 Déclaration des constantes 13
2.7 Expression 14
2.7.1 Notion d'expression 14
2.7.2 Règles d'évaluation d'une expression 14
3. les actions simples
3.1 L'action d'affectation 15
Série d'exercices n° :1 18
3.2 L'action de lecture 20
3.3 L'action d'écriture 20
Série d'exercices n° :2 22
4. l'action de test
4.1 Définition 24
4.2.1 L'action de test simple (Si – Alors-sinon) 24
4.2.2 L'action de test à choix multiples 27
Série d'exercices n° :3 29
5 l'action de répétition
5.1 Définition 31
5.2 Les types des boucles 31
5.2.1 La boucle pour 31
5.2.2 La boucle répéter – jusqu'à 33
5.2.3 La boucle Tantque 34
5.3 Imbrication des boucles 35
Série d'exercices n° :4 36
6 les actions paramétrées
6.1 Introduction 39
6.2 Notion de paramètres 39
6.3 Variable locale et variable globale 39
6.4 Les procédures 41
6.4.1 Définition 41
6.4.2 Procédure sans paramètres 41
6.4.3 Déclaration de la procédure 41
6.4.5 Appel de la procédure 42
6.4.6 Le passage des paramètres 43
6.5 Les fonctions 45
6.5.1 Définition 45
6.5.2 Déclaration de la fonction 46
6.5.3 Appel de fonction 46
6.5.4 Les fonctions prédéfinies 47
Série d'exercices n° :5 48

6 les tableaux

7.1 Introduction 51
7.2 Définition 51
7.3 Les tableaux à une dimension 51
7.3.1 Déclaration 51
7.3.2 La lecture d'un tableau 52
7.3.3 L'affichage d'un tableau 53
Série d'exercices n° :6 54
7.4 Les tableaux à deux dimensions 56
7.4.1 Définition 56
7.4.2 La lecture d'une matrice 57
7.4.3 Affichage d'une matrice 57
Série d'exercices n° :7 58

8 les enregistrements

8.1 Introduction 60
8.2 Définition 60
8.3 Manipulation des enregistrements 61
Série d'exercices n° :8 62

9 Solutions d'une partie des exercices

L'ordinateur 1

1.1 Introduction

Un ordinateur est une machine électroniques permettant de manipuler des informations qu'on appelle des données et capable de faire "tourner" des programmes, c'est-à-dire une suite ou séquence d'instructions programmées à l'avance et qu'il va dérouler du début à la fin dans le but d'obtenir des résultats.

Pour apprendre a écrire des programmes, i faut d'abord comprendre comment un ordinateur peut dérouler un programme. Dans cette partie, on va présenter en bref l'architecture d'un ordinateur selon **Von Neumann** qui a défini en 1944 l'architecture des ordinateurs modernes encore largement utilisée aujourd'hui.

1.2 L'architecture de Von Neuman

Il décompose l'ordinateur en quatre parties distinctes :

-L'**Unité Arithmétique et Logique** UAL (ALU en anglais) est l'organe de l'ordinateur qui exécute les calculs. Certaines documentations lui rajoutent quelques registres (petites cases mémoires intégrées à l'UAL) et lui donnent le nom de **processeur** (CPU).

-L'Unité de Contrôle UC (CU en anglais), contrôle le séquençage des opérations, autrement dit le déroulement du programme. Elle prend ses instructions dans la mémoire et donne ses ordres à l'UAL selon que le programme lui ordonne d'effectuer.

- La mémoire peut être décrite comme une suite de petites cases numérotées, chaque case pouvant contenir une petite information.

C'est l'UC qui a comme rôle central de contrôler l'accès à la mémoire pour le programme et les données. Chaque numéro de case est appelé une adresse. Pour accéder à la mémoire, il suffit de connaître son adresse. Les instructions du programme pour l'UC et les données pour l'UAL sont placées dans des zones différentes de la même mémoire physique.

-Les Entrées/Sorties E/S (I/O en anglais) permettent de communiquer avec le monde extérieur et donc vous : ce peut être un clavier pour entrer les données, et un écran pour afficher les résultats. Il permet à l'ordinateur d'être interactif.

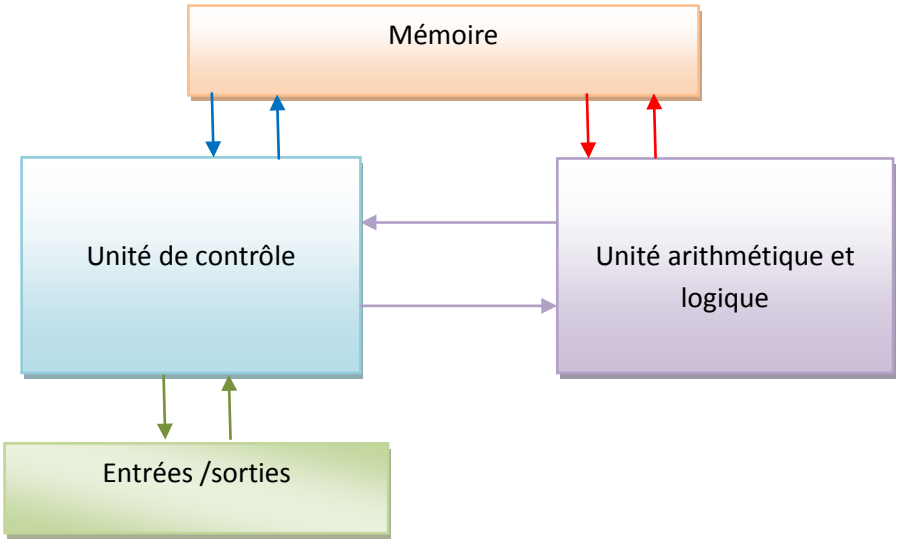


Fig 1 l'architecture d'un ordinateur de Von Newmann

1.3 Représentation des informations

Nous avons défini l'ordinateur comme une machine électronique (un ensemble de circuit) et dans un circuit , il n'y a que deux possibilités : soit le courant passe et dans ce cas cela équivaut à une valeur de un (1), soit le courant ne passe pas, et dans ce cas c'est la valeur zéro (0) qui est retenue. L'ordinateur donc ne manipule que deux valeurs 0 et 1 , on parle de binaire.une unité binaire (0/1) est appelé **bit.**

1.4 Les étapes de résolution d'un problème par ordinateur

1.4.1 La définition du problème

La phase de la définition du problème consiste à comprendre l'énoncé du problème pour déterminer toutes les informations disponibles et la forme des résultats désirés.Il est inutile de passer à la phase suivante sans bien identifier le problème.

1.4.2 Analyse du problème

Elle consiste à trouvez le moyen de passer dès données aux résultats, après la

décomposition du problème en sous problèmes plus simple à résoudre, on doit écrire une suite d'instructions indiquant de façon unique l'ordre dans lequel doit être effectuer ces instruction et en associant à chaque sous problème :

- Les données nécessaires
- Les données résultantes

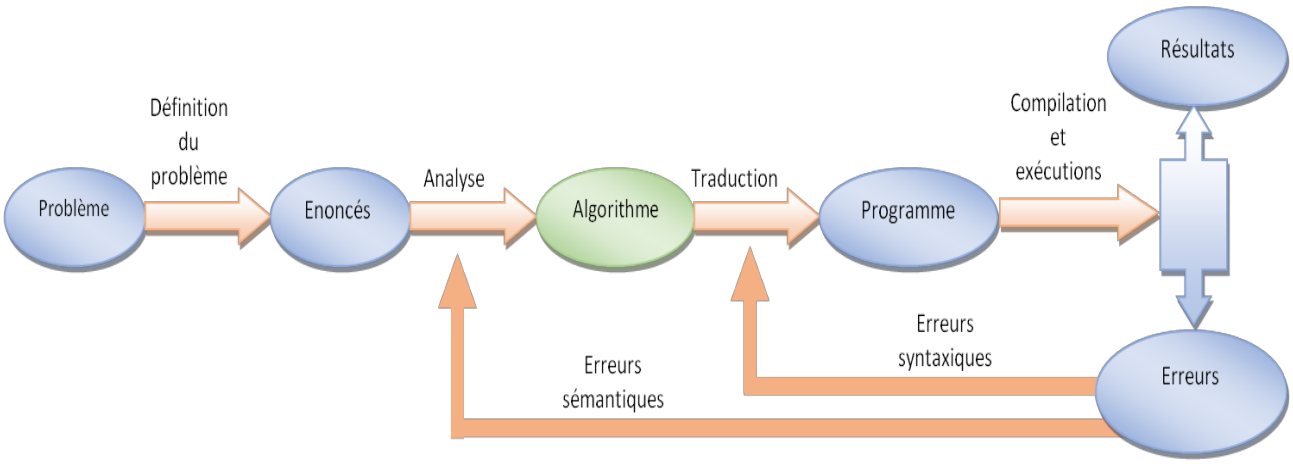


Fig 2 Les étapes de résolution d'un problème

1.4.3 Elaboration d'un algorithme

Une fois les étapes trouvées, il faut les écries d'une façon claire et non ambiguë, c'est l'élaboration d'un algorithme.

1.4.4 Traduction des algorithmes en langage de programmation

Les étapes 1,2 et 3 se font sans utiliser l'ordinateur. Pour exécuter l'algorithme sur machine, il faut le traduire dans un langage de programmation adapté à la machine utilisée.

1.4.5 Compilation et exécution

La compilation consiste à traduire le programme à un langage compréhensible par la machine, au cours de cette traduction, le compilateur détecte des éventuelles erreurs (syntaxiques / sémantiques). Une fois les erreurs sont corrigées, le programme sera exécuté et les résultats désirés seront obtenus.

Cette phase sert à vérifier l'exactitude du comportement du programme, si l'algorithme ne répond pas parfaitement à toutes les requêtes exprimées dans l'énoncé du problème, on doit revenir à l'étape 2.

L'algorithme 2

2.1 Qu'est ce qu'un algorithme ?

Un algorithme est une suite d'actions finie qui une fois s'exécute dans un ordre logique, produiront le résultat désiré. La suite d'opérations sera composée d'actions élémentaires appelées instructions.

Dans la vie quotidienne, un algorithme peut prendre la forme:

- d'un guide d'utilisation d'un appareil ;
- d'une recette de cuisine ;
- d'un guide touristique.

L'algorithme décrit un traitement sur un ensemble de données à travers des actions qui doivent être définie de façon non ambigüe et réalisable par une machine.

Un bon algorithme doit être :

-Déterministe :

-Bien structuré :

-Non ambiguë

-Efficace

2.2 Structure générale d'un algorithme

2.2.1 Le nom de l'algorithme

Chaque algorithme doit avoir un nom, c'est est un identificateur qui permet d'identifier un algorithme, il est généralement précédé par le nom algorithme. Il est préférable de donner un nom indiquant le contenu de l'algorithme pour permettre au lecteur d'avoir une idée de ce que fera l'algorithme.

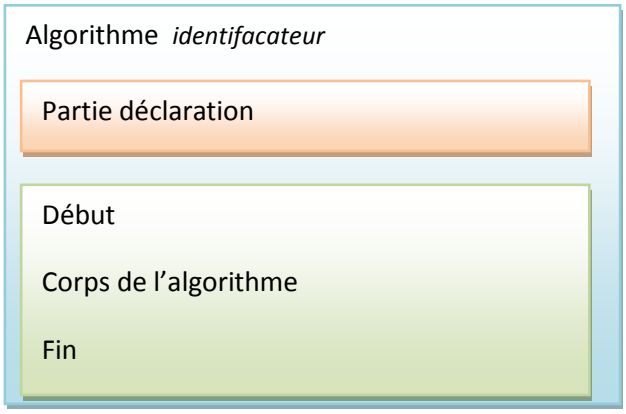


Fig2 Structure générale d'un algorithme

Identificateur :

C'est une suite de lettres et chiffres, il ne doit pas commencer par un chiffre et il ne doit pas contenir des caractères spéciaux (ponctuation, accentué, ..).

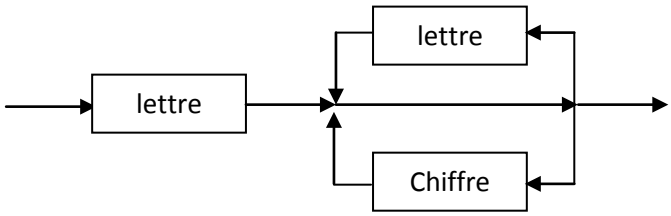


Fig 3 Syntaxe générale d'un identificateur

Exemples

somme	identificateur valide
somme_notes	identificateur valide
somme notes	identificateur non valide (contient espace)
som_note_élève	identificateur non valide (é è : accentués)

2.2.2 La partie déclaration

C'est une liste exhaustive des objets utilisésdans le corps de l'algorithme, qui généralement sont :

• Déclarations des constantes
• Déclarations des types
• Déclarations des variables
• Déclarations des sous-programmes (procédures et fonctions)

2.2.3 Corps de l'algorithme

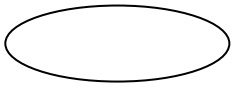
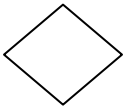
C'est dans cette partie où les actions (instructions) à exécuter par notre algorithme , sont placées . Généralement c'est une combinaison de :

- Affectation
- Lecture
- Ecriture
- Tests
- Boucles (itérations)

2.3. La représentation graphique (Organigramme)

Les algorithmes peuvent être représentés sous forme structurée ou sous forme graphique. Un organigramme est une représentation graphique d'un algorithme, c'est un schéma explicatif avec des symboles.Les actions dans un organigramme sont représentées par les symboles dont les formes sont normalisées. Ces symboles sont reliés entre eux par des lignes fléchées qui indiquent le chemin.

Liste des symboles :

Symbole	signification
	Désigne une action qui n'a pas d'autre symbole
	Utilisé pour début et fin
	Ligne de liaison entre symboles
	Désigne une action d'entrée -sortie
	Désigne une condition , le choix d'une voie parmi plusieurs

2.4 Les Types De Données

Un **type** en algorithmique est une information permettant de traduire les valeurs depuis une représentationbinaire (celle de l'ordinateur) vers une autre représentation plus adaptée à leur programmationdans un langage évolué. Cette notion est tellement importante que toute valeur a forcément un type.

Le rôle du type est d'assurer cette traduction en indiquant quelle place en mémoire occupe la valeuret quelle est la technique de codage utilisée. Il décrit ainsi l'ensemble d'opérateurs sur ces valeurs.

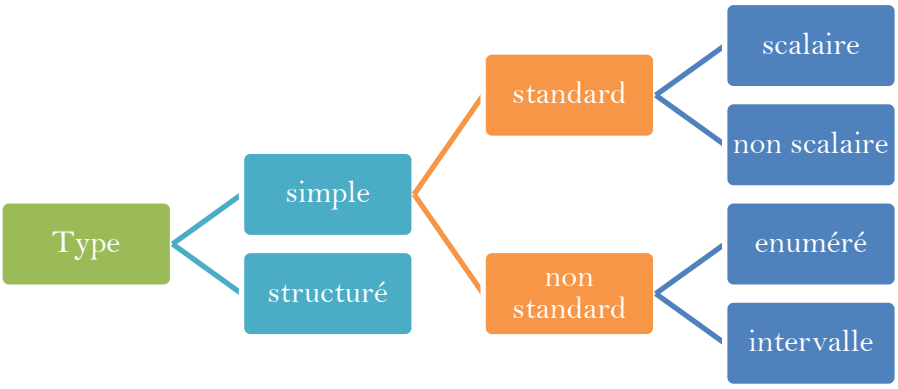


Fig 4 Les différents types en algorithmique

Nous distinguons quatre types standards en algorithmique :entier,réel,booleen et caractère.

2.4.1 Type entier

1 Définition

Le type entier est utilisé pour stocker des valeurs entières, positives ou négatives. C'est un sous ensemble des entiers qui prends ces valeur entre deux valeurs (- max, +max).

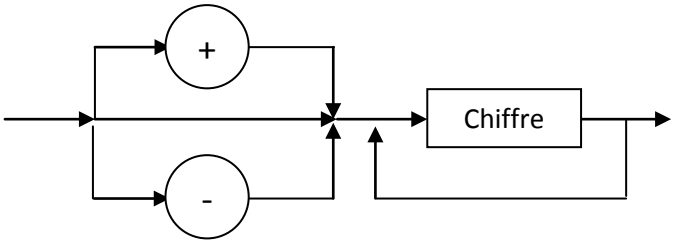


Fig 5 Structure du type entier

2 Opérateurs

Les opérateurs sur les entiers sont :

Opérateur	description	exemple
(+)	Identité	+5
(-)	Signe opposé	-5
+	Addition	3+8 = 11
-	Soustraction	12-6=6
*	Multiplication	-3*5=-15
Div	Division entière	14 div 3 =4
mod	Modulo , le reste de la division entière de A par B	14 mod 3 =2

Les opérateurs sur le type entiers

2.4.2 Type Réel

1 Définition

Le type réelest utilisé pour stocker les nombresà virgule, positives ou négatives. C'est un sous ensemble des nombres réels qui dépend de la machine utilisée.

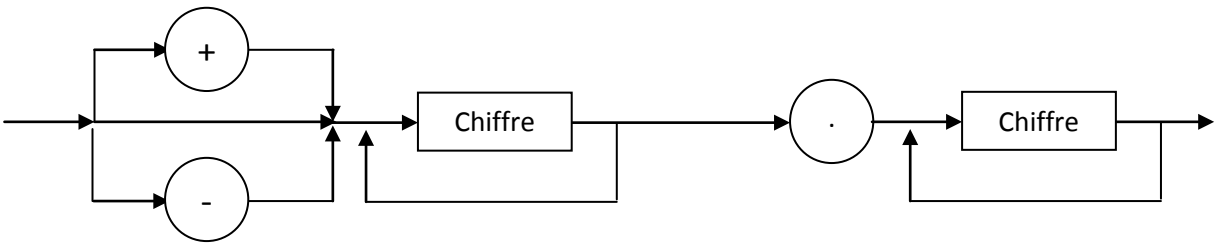


Fig 6 Structure du type réel

2 Opérateurs

Les opérateurs sur les réels sont :

Opérateur	description	exemple
(+)	Identité	+5.3
(-)	Signe opposé	-5.11
+	Addition	3.12+8.15 = 11.27
-	Soustraction	12-6.5=5.5
*	Multiplication	-3.6*5.1=-18.36
/	Division	14.2 div 3 =4.73

Les opérateurs sur le type réel

2.4.3 Type Booléen

1 Définition

Le type booléenest utilisé pour stocker deux valeurs (vrai et faux)

2 Opérateurs

Opérateur	Description	exemple
non	Négation	Non(a=b)
et	Conjonction	(A >b) **et**(b>c)
ou	Disjonction	(A >b) **ou**(b>c)

Table de vérité

a	b	Non(a)	a et b	a ou b
Faux	Faux	Vrai	Faux	Faux
Faux	Vrai	Vrai	Faux	Vrai
Vrai	Faux	Faux	Faux	Vrai
Vrai	Vrai	Faux	Vrai	Vrai

Opérateurs de relation

Opérateur	description
=	égale
<>	différent
<	Strictement inférieur
<=	Inférieur ou égale
>	Strictement supérieur
>=	Supérieur ou égale

2.4.4 Type caractère

1 Définition

L'ensemble des valeurs de type caractère est un jeu fini et totalement ordonné de caractère , Il est utilisé pour stocker des caractères (lettres , signes de ponctuation , chiffres , …).

La valeur de type caractère est représentée par le caractère lui-même placé entre deux apostrophes (')

Exemple :

‘a’ , ‘A’ , ‘+’, ‘-‘, ‘*’ , ‘ ‘ , ‘6’

2 Opérateurs

Opérateurs de relation

Opérateur	description
=	Egale
<>	Différent
<	Strictement inférieur
<=	Inférieur ou égale
>	Strictement supérieur
>=	Supérieur ou égale

Leur utilisation dépend de l’ordinateur utilisé. Dans certain langage ‘A’<’B’<’C’….<’Z’ , les lettres sont ordonné par ordre alphabétique.

Remarque

L’ordinateur utilise une table de correspondance qui associe une valeur entière (uncode) à un caractère qu’il s’agit de manipuler, c’est-à-dire, la plupart du temps, pour l’afficher àl’écran. Cette table de correspondance se nomme la table de symboles (table ASCII, Unicode).

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Table ASCII

2.5 Variables

2.5.1 Notion de variable

Une **variable** est une donnée qu'un algorithme peut manipuler. C'est un emplacement mémoire capable de contenir des valeurs de type défini au préalable et qui sert à stocker provisoirement des valeurs.

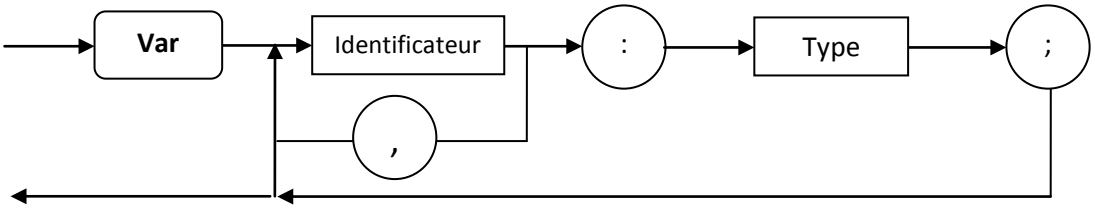
On peut simuler une variable à une boite, repérée par une étiquette. Pour avoir accès au contenu de la boite , il suffit de la désigner par son etiquette.

Toute variable possède :

- Un **type** (entier, réel, caractère ou booléen).
- Un **nom** ou identificateur que l'utilisateur choisit ; il permet au programme de reconnaître quelledonnée il doit manipuler.
- Une **valeur** qui peut évoluer au cours de l'algorithme, mais qui doit respecter le type.

2.5.2 Déclaration des variables

Pour utiliser une variable dans un algorithme, il faut d'abord la créer et lui donner un nom (identificateur), c'est la déclaration.

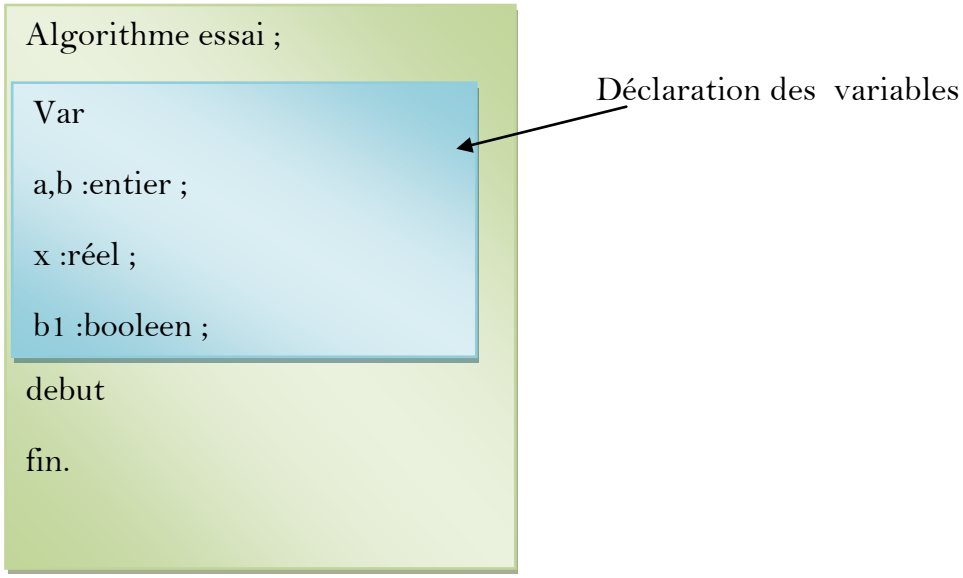


Déclaration des variables

Remarque

La valeur initiale de la variable (au début de l'algorithme) est indéterminée, donc il est indispensable de lui affecter une première valeur (initialiser la variable).

Exemple

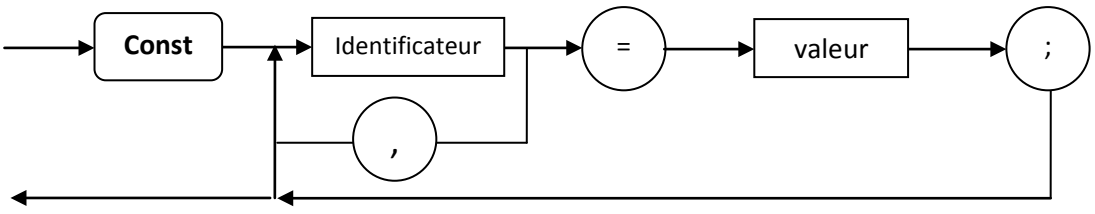


2.6 Constantes

2.6.1 Notion de constante

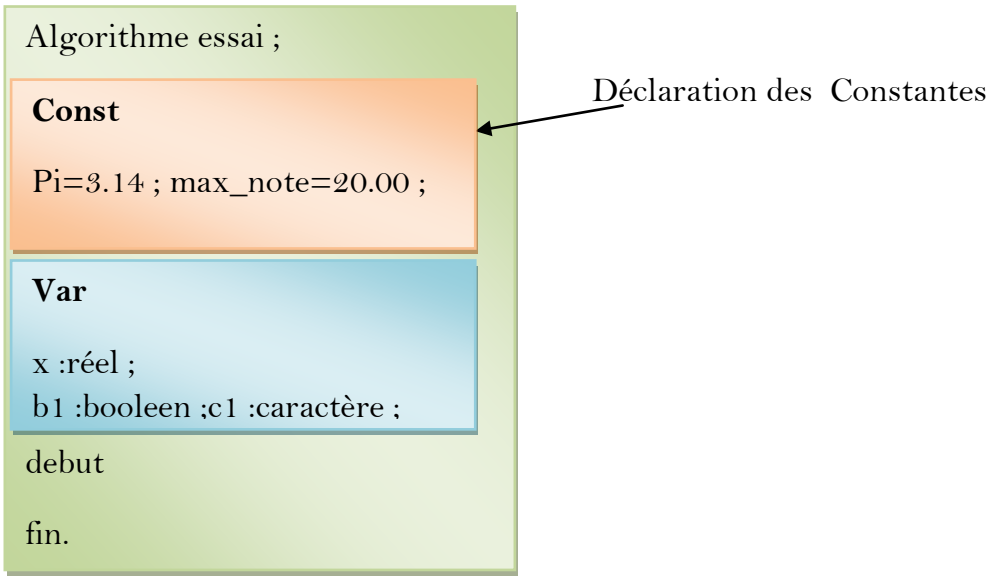
Une constante est une variable dont la valeur ne change jamais dans l'algorithme.Ellereprésente des nombres, des caractères, … dont la valeur **NE PEUT PAS** êtremodifiée au cours de l'exécutiondes actions de l'algorithme.

2.6.2 Déclaration des constantes



Déclaration des constantes

Exemple



2.7 Expression

2.7.1 Notion d'expression

Une expression est une composition d'opérateurs appliqués à des opérandes qui sont des constantes, des variables ou des sous expressions. L'évaluation d'une expression donne une valeur.

Exemple :

(m-1)*(k+j-d div 2) : c'est une expression qui renvoie une valeur entière.

(a=2.5) ou (b >10) : c'est une expression booléenne.

2.7.2 Règles d'évaluation d'une expression

L'évaluation d'une expression retourne une valeur calculée en termes de valeurs des variables de l'expression. Cette évaluation doit se faire dans l'ordre des priorités décroissantes des opérateurs fixées comme suit :

entiers	réels	booléen
Parenthèses	Parenthèses	Parenthèses
* , div , mod	* , /	Et
+ , -	+ ,-	Ou
		non

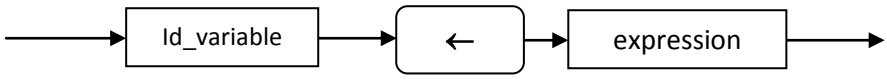
3 Les actions Simples

3.1 L'action d'affectation

L'affectation est la seule action qui nous permettre de manipuler une variable et de changer son contenu, c'est l'action qui nous permettre de donner une valeur à une variable.

Syntaxe

Id_varaible← expression



L'exécution de cette action consiste à remplacer la valeur de variable par celle obtenue en évaluant l'expression.

Notation :

X ← 5 se lit : X**reçoit** 5

Et signifie que la variable X prendre comme valeur 5.

Très important :

✍L'affectation écrase le contenu de la variable réceptrice.

A	Après l'exécution de l'action :	A
20	A←15 ;	15

✍L'affectation n'affecte pas le contenu de la variable émettrice.

A	B
20	10

Après l'exécution de l'action : A←B ;

A	B
10	10

✍La partie émettrice ne peut pas être une expression.

L'action A+1←B ; est une action erronée

✍Il faut que le type de l'expression soit compatible avec celui de la variable.

var A :entier ; X :réel

L'action : A ← X /2 ; est une action erronée

A est de type entier et X/2 est de type réel ; on peut pas affecter une valeur réel à une variable de type entier.

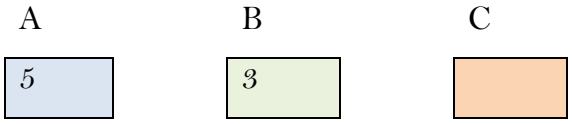
Exemple

Algorithme qui effectue la permutation entre 2 entier.

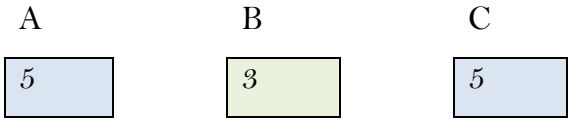
```
Algorithme permute ;
Var
A,B,C : entier ;
DEBUT
Ecrire('donner deux valeur ') ;
Lire(A,B) ;
C ←A ;
A ←B ;
B ←C ;
Ecrire('voici les deux valeurs après permutation :') ;
Ecrire(A ,B) ;
FIN.
```

L'illustration de déroulement de l'algorithme :

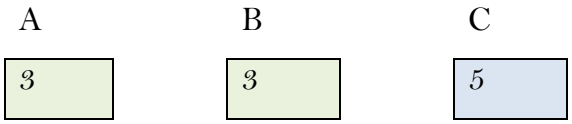
Lire(A,B) ;



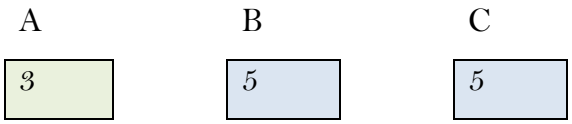
C ←A ;



A ←B ;



B ←C ;



Série d'exercices n° :1

Exercice 01

a, b, c, d, e, f, g, h étant des variables réelles, exprimer en algorithme l'expression suivante.

$$\frac{a+\frac{12-b}{c}}{15\frac{d}{e}}+\frac{f}{\frac{g}{h}}$$

Exercice 02

Soient les déclarations suivantes :

Var

j,i,m : entier ;

b1,b2 : booléen ;

r,s,t :réel ;

c1,c2 :char ;

Parmi les expressions suivantes, détecter celles qui sont invalides. Donner les types des résultats des expressions valides.

1. (j*5)+1
2. b1-(r*s)
3. (m+5.0)*s
4. (b1 ou b2) et (r=6.2)
5. (b/8) et (m=i)
6. b2 et (j<k)
7. (i*8) - (c1/4.2)
8. (c1*10.5)-r
9. (c2='b1')et(i=j*3)
10. b1 et non(j>5)

Exercice 03

Soient le déclaration de variables :

VAR

a, b, c, d, w, x, y, z : entier ;

m,n,k,t,v1,v2,v3,v4,v5 : booleen ;

Quel est le contenu de ces variables après exécution des actions suivantes :

I	**II**	**III**
a← 1 ; b ←2 ; c ← 3 ;	a← 1 ; b←2 ; c← 3 ; d← 4 ;	m←vrai; n ←faux; k ←vrai;
w←a + b * c ;	w← a + b*c-d ;	t←faux;
x←a * b + c ;	x ← (a + b)*c-d ;	v2← m ou n et k et t;
y←a + c div b ;	y← a + (b*c)-d ;	v3← (m et n) et k ou t;
z← c div a + b ;	z←a + b*(c-d) ;	v4← m ou (n et k) ou t;
		v5← m ou n et (k ou t);

Exercice 04

Soit la séquence suivante :

1) Lire m et n (m>0 , n>0)
2) Prendre pour valeur de p la valeur de m
3) Si n=1 alors ecrire p et fin , sinon aller à 4)
4) Prendre pour nouvelle valeur de p la valeur de p+m
5) Retrancher 1 de la valeur de n
6) Aller à 3)

Le travail demandé :

a) Simuler à la main cette séquence pour : m=6 et n=4 , m=7 et n=5
b) Que fait cette séquence ?
c) Simuler cette séquence pour les nombres 3 et 5 d'une part et 5 et 3 d'autre part. Que remarquez-vous ? proposez une amélioration.

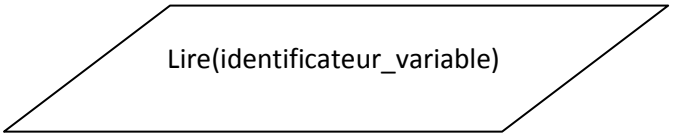
3.2 L'action de lecture

La lecture peut être définie comme une affectation à partir du clavier, c'est l'action par laquelle on invite l'utilisateur à rentrer une valeur qui sera affectée directement à la variable concerné par cette lecture.

Syntaxe :

Lire (identificteur_variable) ;

Représentation par organigramme :



Exemples :

Lire(X) ; où X est une variable de type **entier.**

Lire (Rep) ; où Rep est une variable de type **caractère**.

Si vous devez saisir plusieurs valeurs à placer chacune dans une variable, vous pouvez utiliser plusieurs « **Lire** », maisplus simplement placez les diverses variables à la suite d'un unique « **Lire** », séparées par des virgules.

Exemples :

Lire (X) ; Lire (Y) ; Lire (Z) ; c'est identique à **Lire(X,Y,Z)** ;

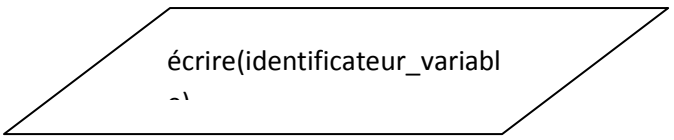
3.3 L'action d'écriture

Pour simuler l'affichage d'un texte ou d'une valeur sur l'écran, il faut utiliser laL'action "ecrire" qui prend àsa suite un texte ou une variable. Si vous mélangez du texte et des variables, séparez ceux-ci par desvirgules.

La syntaxe :

Ecrire (idntifiacteur_variable) ; : pour afficher le contenu d'une variable

Ecrire(" Texte ") ; : Pour afficher le texte tel qu'il est écris entre " "



Exemple :

A←5 ;

Ecrire (A) ; affiche la valeur ***5***

Ecrire('A') ; Affiche le caractère **A**

Exemple d'algorithme :

L'algorithme suivant lit deux valeurs entières à partir du clavier et affiche leur somme.

```
Algorithme somme ;
Var
A,B,S : entier ;
Debut
Ecrire('Donner une valeur entière ') ;
Lire(A) ;
Ecrire('Donner une autre valeur entière ') ;
Lire(B) ;
S←A+B ;
Ecrire('La somme est :') ;
Ecrire(S) ;
Fin.
```

Série d'exercices n° :2

Exercice 01

Soit les deux algorithmes suivants :

```
Algorithme   remise1 ;
Var
Prix,taux ,remise : réel ;
Debut
 prix ← 15 ;
 taux ← 10.5 ;
 remise ← (prix * taux) ;
 remise ← remise / 100 ;
 prix ← prix - remise ;
écrire(prix) ;
Fin.
```

```
Algorithme   remise2 ;
Var
Prix,taux ,remise : réel ;
Debut
Ecrire('donner le prix svp') ;
Lire(prix) ;
Ecrire('donner le taux de remise svp') ;
Lire(taux) ;
 remise ← (prix * taux) ;
 remise ← remise / 100 ;
 prix ← prix - remise ;
écrire(prix) ;
Fin.
```

Dérouler les deux algorithmes. Que remarquez-vous ?

Exercice 02

Ecrire les déclarations et simuler les algorithmes suivants :

```
Algorithme   algo1 ;
Début
a←10  ;
b ← 70 ;
q1 ← (a + b) / 5 ;
 q2 ← (a + b) div 5 ;
 r2 ← (a + b) mod 5 ;
écrire(q1, q2, r2) ;
 q1 ← (a + b) / 3 ;
q2 ← (a + b) div 3 ;
 r2 ← (a + b) mod 3 ;
écrire(q1, q2, r2) ;
Fin.
```

```
Algorithme   algo2;
Début
 lire(c) ;
 lire(m) ;
 b1 ← c <> 'r' ;
 b2 ← (m = 7) OU b1 ;
écrire("la valeur de b1 est : ", b1) ;
écrire("la valeur de b2 est : ", b2) ;
Fin.
```

Exercice 03

Ecrire un algorithme qui effectue l'addition, la soustraction, la multiplication et la division de deux entiers et affiche le résultat.

Exercice 04

Ecrire un algorithme qui a partir des valeurs de résistance R et de tension V , calcule la valeur du courant I.

Exercice 05

Ecrire un algorithme qui demande la valeur d'une température exprimée en Celsius puis calcule et affiche son équivalent en Fahrenheit. Utiliser la formule suivante :

[°C] = ([°F] - 32) x 5/9

Exercice 06

Ecrire un algorithme qui demande une valeur entière pour a, une valeur entière pour b, échange les valeurs de deux variables entiers a et b et affiche les nouvelles valeurs.

1- En utilisant une variable intermédiaire
   Sans utiliser une variable intermédiaire.

Exercice 07

Ecrire un algorithme qui lit sur l'entrée standard une valeur représentant une somme d'argent et qui calcule et affiche le nombre de billets de 1000DA, 500DA, et de pièces de 200 DA , 100 DA , 50 DA et 1DA qu'elle représente.

Exercice 08

Ecrire un algorithme qui affiche la somme des n termes de la suite :

$U_0=0$

$U_{n+1}=U_n+1$

Exemple : n=7 , la somme = 28

4 L'action de test

4.1 Définition

L'action de test ou conditionnelle est une action dans laquelle une expression booléenne sera évaluée et selon la valeur obtenue (vrai ou faux), l'algorithme va effectuer une action ou une autre. Deux actions sont présentées ; simple et à choix multiples.

La condition généralement est une comparaison qui comporte :

- Une valeur, un opérateur de comparaison et une autre valeur.

4.2.1 L'action de test simple (Si – Alors-sinon)

Une décision sera prises parmi deux , selon le résultat de l'évaluation de l'expression logique (la condition). Il existe deux formes simple et composé

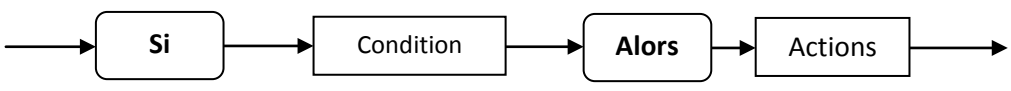
Forme simple :

La condition qui est une expression booléenne est évaluée, si le résultat a pour valeur Vrai les actions seront effectuées.

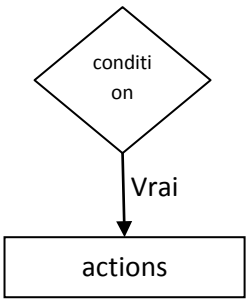
Syntaxe :

Si(condition) **alors**

Actions



Forme organigramme



Exemple :

Si (rep='o') alors écrire('terminer')

Forme composé

Après l'évaluation de la condition qui est une expression booléenne, si le résultat a pour valeur Vrai ***les actions1*** seront effectuées, si le résultat a pour valeur Faux **les actions 2** seront effectuées.

Syntaxe :

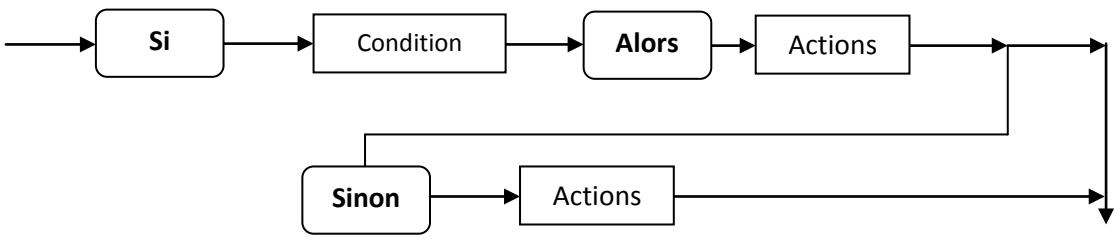
Si(condition) **alors**

Actions 1

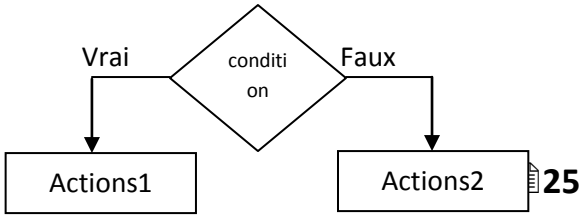
Sinon

Actions 2

Finsi



Forme organigramme



Exemple

Si (moy>=10) alors ecrire('admis')

Sinon écrire('Ajourné') ;

L'action de test peut présenter plusieurs choix (deux ou plus) , cette situation peut être représenter par une imbrication de condition si- alors-sinon.

Exemple :

Si (age>=9) et (age<11) alors écrire ('Poussin')

Sinon Si (age>=11) et (age<13) alors écrire ('Benjamin')

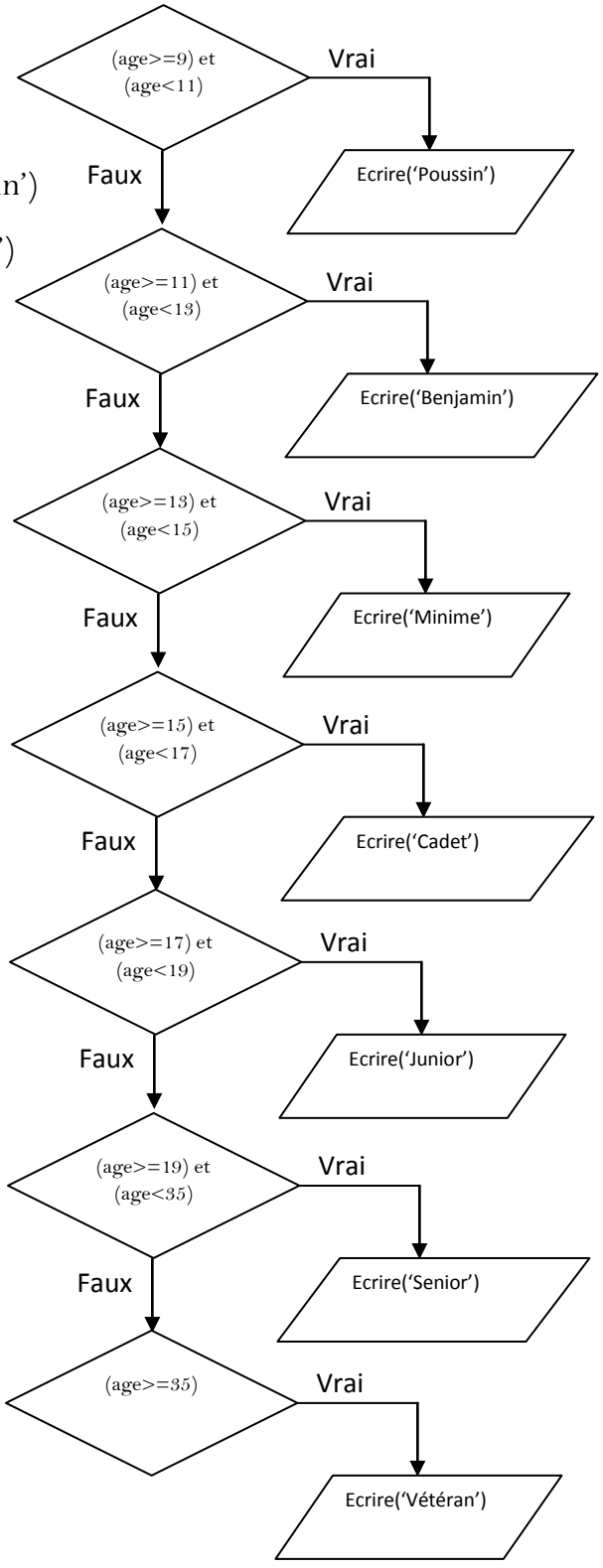
Sinon Si (age>=13) et (age<15) alors écrire ('Minime')

Sinon Si (age>=15) et (age<17) alors écrire ('Cadet')

Sinon Si (age>=17) et (age<19) alors écrire ('Junior')

Sinon Si (age>=19) et (age<35) alors écrire ('Senior')

Sinon Si (age>=35) alors écrire ('Vétéran') ;



4.2.2 L'action de test à choix multiples

Il arrive des fois que l'utilisation des tests imbriqués complique la situation, et surtout si le nombre de test est important. L'action de test à choix multiples offre la possibilité de remplacer plusieurs blocs de si alors sinon par une action dans laquelle on indique quoi faire lorsque une telle valeur est rencontrée.

Syntaxe :

Selon expression de sélection

Condition 1 : actions 1

Condition 2 : actions 2

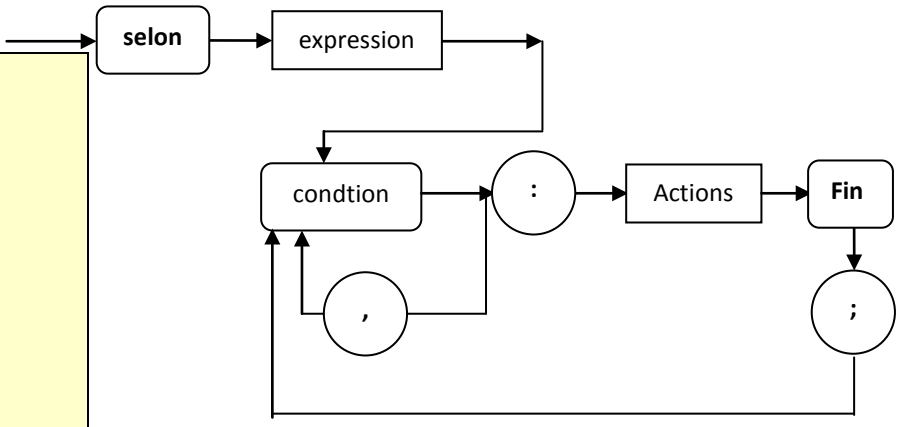
Condition 3 : actions 3

…

…

Sinon : actions n

Fin selon



Exemple :

Selon mois :

1 :écrire('janvier') ;

2 :écrire('Février') ;

3 :écrire('mars') ;

4 :écrire('Avril') ;

5 :écrire('Mai') ;

6 :écrire('Juin') ;

7 :écrire('Juillet') ;

8 :écrire('Aout') ;

9 :écrire('Septembre') ;

10 :écrire('Octobre') ;

11 :écrire('Novembre') ;

12 :écrire('Décembre') ;

Sinon ecrire (‘erreur’) ;

Fin ;

Chaque valeur correspond à une valeur possible de la variable du selon. Si plusieurs actions sont présentés, il est préférable de les placer entre début ..fin.

Série d'exercices n° :3

Exercice 01

Soit les deux algorithmes suivants :

```
Algorithme test_1 ;
Var
 nb : entier
Début
 écrire ("Donner un nombre entier") ;
 lire (nb) ;
 si nb <= 0 alors nb ← nb + 5
            sinon nb ← nb – 5 ;
écrire("maintenant le nombre vaut : " , nb) ;
Fin
```

```
Algorithme test_2 ;

Var
nb : entier ;
 Début
écrire ("Donner un nombre entier") ;
 lire (nb) ;
si nb <= 0 alors nb ← nb + 5 ;
 si nb > 0 alors nb ← nb – 5 ;
 écrire ("maintenant le nombre vaut : " , nb) ;
 Fin.
```

- Tracez les organigrammes correspondants
- Simuler chaque algorithme avec nb=5 , nb=0, nb=-5
- Les deux algorithmes sont ils équivalents ?

Exercice 02

Ecrire un algorithme qui demande un nombre entier à l'utilisateur et affiche pair si le nombre est pair, impair sinon.

Exercice 03

Un magasin est ouvert de 10 heures à 14 heures et de 16 heures à 20 heures, sauf le samedi après-midi et le vendredi toute la journée. On suppose que l'heure h est un entier entre 0 et 23. Le jour j code 0 pour lundi, 1 pour mardi, etc.
Ecrire un algorithme qui lit le jour et l'heure et affiche un message indiquant si le magasin est ouvert ou non.

Exercice 04

On désire calculer le montant d'une facture d'électricité sachant que l'abonné paye :
• des frais fixes d'abonnement de 250 da
• sa consommation selon un tarif à tranches :
1,20 da par kWh pour les 100 premiers kWh
2,00 da par kWh pour les 150 suivants
5,00 da par kWh pour ceux qui excèdent 250 kWh
On connaît pour l'abonné le relevé du compteur : AI(l'ancien index) et NI (le nouvel index).

Exercice 05

Ecrire un algorithme qui demande deux valeurs entières et affiche le menu ci-dessous

1 : L'addition
2 : La soustraction
3 : La multiplication
4 : La division
5 : Le reste de la division
Tapez votre choix (1..5)

et affiche le résultat de l'opération sur les deux valeurs selon le choix lu

Exercice 06

Ecrire un algorithme demandant la date sous forme de trois nombres et vérifiant que les trois nombres correspondent a une date valide. Ensuite, améliorer l'algorithme pour qu'il affiche le jour d'après

Note : Une année est bissextile si elle vérifie intégralement deux règles :

-les années divisibles par 4, et,

- les années divisibles par 400 mais pas par 100.

Exercice 07

Dans un lycée, la comite scientifique utiliser une politique de passage pour les élèves de la première année tronc commun dans le dernier conseil de l'année.

La première condition est que l'élève ne doit pas dépasser 40 heurs d'absence durant l'année.

Si le nombre d'heurs d'absence est < 40 heurs, il faut que le nombre d'heurs d'absence non justifier ne dépasse pas 20 heurs.

Dans le cas favorable, si la moyenne générale est ≥ 10, l'élève passe en deuxième année et 3 options sont proposées, les cas suivants sont possibles :

1 – Si l'élève a une moyenne générale ≥ 13, il aura l'option choisie dans la fiche de veux.
2 – Sinon, voir la moyenne des matières principales pour l'option choisie,
Si cette moyenne ≥ 10, lui donner l'option choisie.
Sinon, choisir l'option qui convient à ses notes sans voir la fiche de veux.

Si la moyenne générale est < 10, Dans ce cas la moyenne permise pour le passage est 09.30 a condition que le nombre d'heurs d'absence soit <10 , Dans ce cas, l'élève passe en deuxième année et la troisième option lui sera accordée quel que soit son choix.

- **Ecrire un algorithme qui affiche les informations d'un élève vis-à-vis son passage en deuxième année et l'option accordée.**

L'action de répétition 5

5.1 Définition

L'action de répétition ou la boucle c'est l'une des actions de base en algorithmique, autrement dit structure itérative qui désigne une suite d'actions destinée à être exécutée plusieurs fois. L'objectif d'utiliser une boucle est de répéter un bloc d'actions plusieurs fois.

5.2 Les types des boucles

Il existe plusieurs types de boucles : certaines ont un nombre fixe d'itérations, d'autres dépendent deconditions de sortie que vous aurez à définir.

5.2.1 La boucle pour

A boucle « Pour » est une boucle à l'usage quasiexclusifdes compteurs. À chaque passage dans la boucle, un compteur est incrémenté ou décrémenté, selon le cas.Elle est utilisé généralement lorsque on sait combien de fois on doit itérer un bloc d'actions.

Syntaxe :

Pour ***Compteur*** ← **V**$_{initiale}$ à V$_{finale}$ [PAS pas] Faire
Actions (les actions à répéter)
Fin Pour

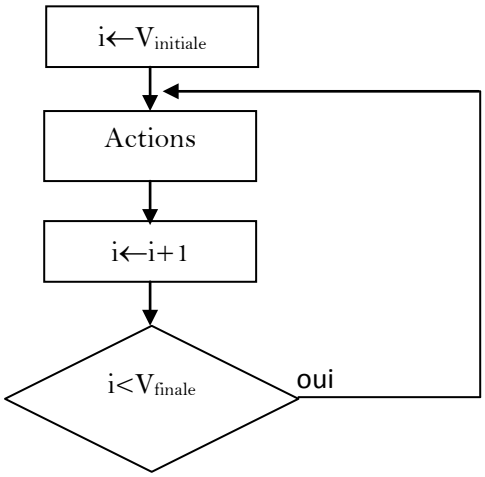
Compteur : variable qui prend ces valeurs entre **V**$_{initiale}$ et V$_{finale}$

Vinitiale : la valeur à partir de laquelle le compteur commence

Vfinale : la valeur à partir de laquelle le compteur termine

Le pas est optionnel , il est par défaut 1.

Représentation par organigramme :



Le cas d'un pas 1

Exemple :

s←0 ;

pour i ←1 à 5 faire

debut

s←s+i ;

fin ;

le déroulement de la séquence donne :

I	**s**
	0
1	0+1
2	0+1+2
3	0+1+2+3
4	0+1+2+3+4
5	0+1+2+3+4+5

Remarque :

Ne modifiez jamais un compteur de boucle "Pour" au sein de celle-ci.

Pour i ←1 à 5 faire

début

s←s+i ;

i←i+2 ; à éviter

fin ;

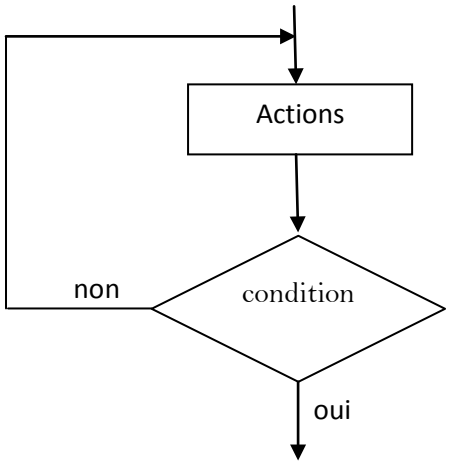
5.2.2 La boucle répéter – jusqu'à

L'action « répéter » permet de répéter l'exécution d'un bloc d'actions une ou plusieurs fois jusqu'a qu'une condition devienne vrai. Il y'aura toujours au moins un passage dans la boucle c'est que le bloc d'actions s'exécute au moins une fois.

Syntaxe :

Répéter
Actions
Jusqu'à (condition)

Forme organigramme



Exemple :

Répéter

Ecrire('donner une note valide :') ;

Lire(note) ;

Jusqu'à (note>=0) et (note<=20) ;

Le déroulement cette séquence donne :

donner une note valide : -1

donner une note valide : 25

donner une note valide : 15

5.2.3 La boucle Tantque

La boucle de type "Tant Que" permet la répétition d'un bloc d'actions tant que la condition testée est vérifiée. Lors de traitement de la boucle "Tant que», Il évalue l'expression booléenne (condition), si l'expression retourne VRAI, alors le bloc d'actions sera effectué, il remonte et teste de nouveau l'expression et exécute le bloc. Une fois la condition devienne fausse, il saute le bloc.

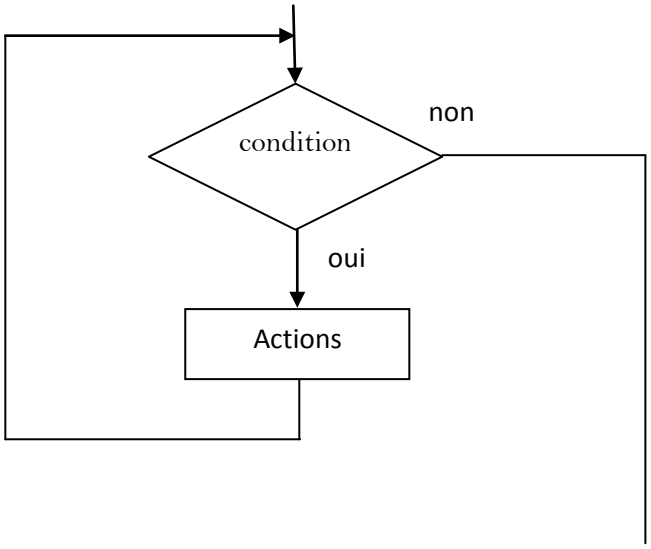
Le "Tant que" ressemble fortement au "Répéter" avec cependant deux importantes différences :

- Le bloc d'instructions peut ne sera pas exécuté.
- L'expression booléenne est inversée. « jusqu'à (X >0) » devient « tant que (X<=0) »

Syntaxe

```
Tant que ( condition) faire
Actions
Fin tantque
```

Forme organigramme



Exemple

écrire('Les nombre impairs inferieurs à 10 sont :') ;

i←1 ;

Tant que i<=10 Faire

Ecrire(i) ;i ←i+2 ;

FinTantQue

Le déroulement de cette séquence donne :

i	affichage
	Les nombres impairs inferieurs à 10 sont :
1	1 3 5 7 9
3	
5	
7	
9	
11	

5.3 Imbrication des boucles

Il est possible d'imbriquer les boucles, c'est-à-dire de mettre une boucle dans uneautre boucle, sur autant de niveaux que vous le souhaitez.

Exemple :

Pour i←1 à 5 faire

Début

j←i ;

Répéter

Ecrire(j) ;

j←j+1 ;

Jusqu'à (j>=10)

Fin ;

Série d'exercices n° :4

Exercice 01

Simuler à la main les algorithmes suivants

```
Algorithme Tanque1 ;
Var
nb : entier
Début
nb ←10 ;
tantque nb < 40 faire
debut
écrire(nb) ;
nb ←nb + 10 ;
fin ;
écrire ('le nombre
vaut ', nb) ;
Fin.
```

```
Algorithme Tanque2 ;
Var
nb : entier ;
Début
nb ←10 ;
tantque nb > 40 faire
debut
écrire(nb) ;
nb ←nb + 10 ;
fin ;
écrire ('le nombre
vaut ', nb)
Fin.
```

```
Algorithme Tanque3 ;
Var ;
x : entier ;
Début
lire(nb) ;
tantque nb <= 3 faire
debut
écrire('nb') ;
nb ←nb + 1 ;
fin ;
écrire (nb) ;
Fin.
```

Exercice 02

```
Algorithme inconnu ;
Var i   x,y : entier ;
Debut
ecrire("entrez vos deux nombres");
lire(x,y) ;
tantque (x <> y) faire
debut
si( y > x) alors  y = y - x
            sinon x = x - y ;
fin ;
ecrire('……est :',x );
Fin.
```

Le travail demandé :

- Simuler l'algorithme avec x=16 et y=4 ; x=18 et y=12 ; x=5 et y=7
- Que fait cet algorithme

Exercice 03

Ecrire un algorithme qui détermine si un nombre P tapé au clavier est parfait, c'est-à dire égal à la somme de ses diviseurs sauf lui. Par exemple, 1 est parfait car 1=1, 6 est parfait car 1+2+3=6, 28 est parfait car 1+2+4+7+14=28.

Exercice 04

Ecrire l'algorithme qui permet d'effectuer la division entière de deux entiers en utilisant la soustraction successive

Exercice 05

Dans un référendum, il peut y avoir des bulletins OUI, des bulletins NON , des bulletins blancs et des bulletins nuls.
Ecrire un algorithme qui pilote le dépouillement d'un référendum. L'assesseur saisit un 'O' pour chaque bulletin OUI, un 'N' pour chaque bulletin NON , un 'B' pour chaque bulletin Blanc et un 'X' pour chaque bulletin Nul. Il interrompe la saisie par le caractère 'F'.
L'algorithme affiche le nombre de votants, de OUI, de NON, de bulletin Nul et de bulletin Blanc, ainsi que les pourcentages correspondants.

Exercice 06

Ecrire un algorithme qui à partir d'un nombre entier N affiche deux autres nombresN1 et N2. Le premier (N1) sera constitué par les chiffres pairs de N et le second (N2) par les chiffres impairs.
Exemples :
N = 25461327 N1 = 2462 , N2 = 5137
N = 42613786 N1= 42686 , N2 = 137
N = 240682 N1 = 240682 , N2 = 0
N = 103 N1 = 0 , N2 = 13

Exercice 07

```
Algorithme calcule ;
Var
 a, b, c : entier ;
DEBUT
ecrire('donner deux entiers positifs') ;
lire ( a, b ) ;
c ← 1 ;
tantque( b ≠ 0 )   faire
                        debut
                        | si ( ( b mod 2 ) = 1 )  alors c← c*a ;
                        | a←a*a ;
                        | b ←b div 2 ;
                        fin ;
écrire( 'le résultat est :',c ) ;
FIN.
```

1- Dérouler cet algorithme avec a=5,b=3 et a=4 et b=4 ;
2- Que fait cet algorithme ?
3- Réécrire le avec la boucle Répéter.

6 Les actions Paramétrées

6.1 Introduction

Ecrire un algorithme qui résout un problème revient généralement à écrire des sous-algorithmes qui traitent des sous parties du problème initiale, ces sous algorithmes sont appelée les actions paramétrées. Elle permet aussi de regrouper des actions ou traitements qui doivent être faits de manière répétitive au sein d'un algorithme. Il existe deux types des actions paramétrées : les procédures et les fonctions.

Une action paramétrée doit obligatoirement avoir un nom (identifiant), et peut avoir une liste de paramètres. Une fois déclaré, un sous-algorithme peut être appelé au niveau de l'algorithme comme une action avec des paramètres.

Les sous-algorithmes rendent l'algorithme plus lisible, ils mettent en évidence sa structure logique.

6.2 Notion de paramètres

Les paramètres d'un sous-algorithme sont des variables qui admettent un type et qui sont associé à une variable dans l'algorithme principal.

La procédure somme(a ,b :entier) a deux paramètres a et b .

6.3 Variable locale et variable globale

1-Variable locale

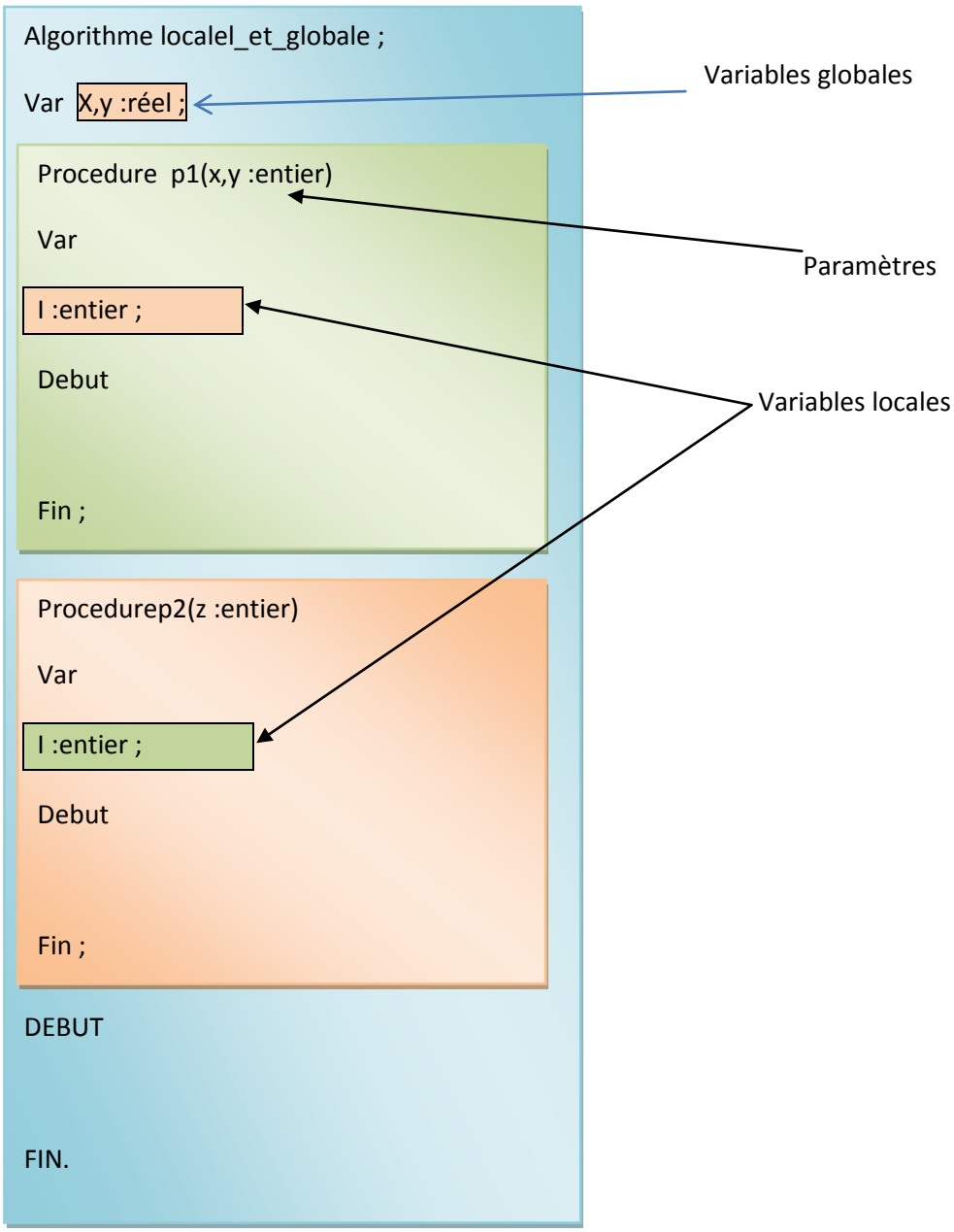
Une variable déclarée dans un sous-algorithme sous les motsclés Procédure ou Fonction ne pourra dansce cas qu'être lisible et modifiable uniquement dans ce sous-algorithme.

Les variables locales peuvent donc parfaitement porter un même nom et n'ont aucun rapport entre elles. Elles sont totalement indépendantes les unes des autres.

La variable i dans un sous-algorithme1 n'est pas de tout la même que la variable i du sous-algorithme 2.

2-Variable globale

Une variable globale est déclarée ou bien a une occurrence de définition à l'extérieur du sous-algorithme.



6.4 Les procédures

6.4.1 Définition

Les procédures sont des sous-algorithmes constitués d'une suite d'instructions indépendantes. Une procédure neretourne pas de résultat ou de valeur àl'algorithme qui l'a appelé.

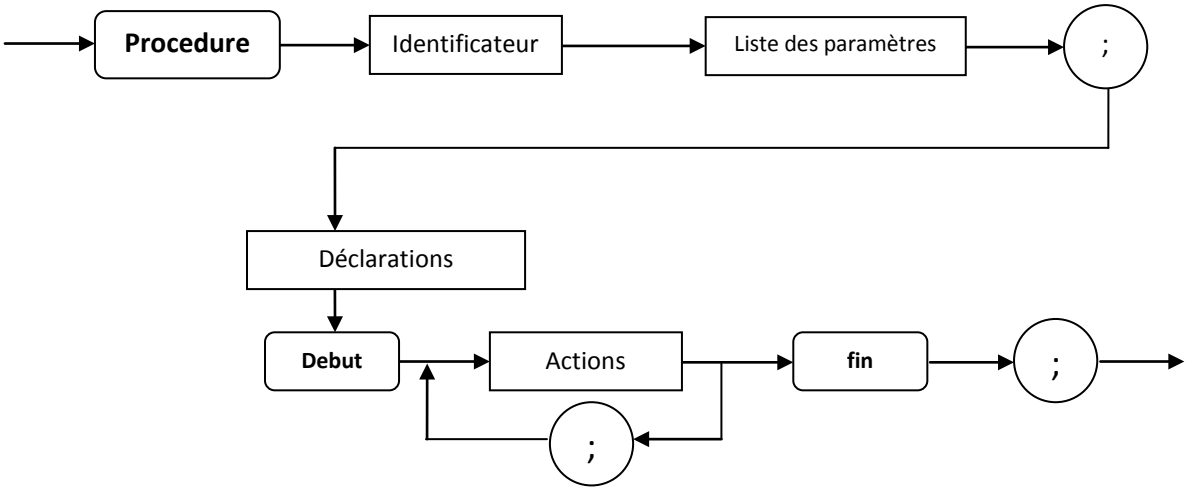
6.4.2 Procédure sans paramètres

Une procédure peut être sans paramètres, ce type de procédure est utilisé pour éviter d'avoir à réécrire plusieurs fois une même suite d'actions figurant plusieurs fois dans l'algorithme.

6.4.3 Déclaration de la procédure

Syntaxe

```
Procedureidentificateur (liste des paramètres )
Déclarationdes variable locales
Debut
Actions
….
Fin ;
```



Exemple

Procedure calculer(a ,b,c :entier)

Var

S :entier ;

Début

S←a+b ; c←S ;

Fin ;

6.4.5 Appel de la procédure

Lors de l'appel de la procédure, une association entre les variables de l'algorithme appelant et les paramètres de la procédure se faite tout en respectant :

- Le nombre de paramètres ;
- Le type des paramètres ;
- L'ordre des paramètres.

Exemple :

Algorithme exemple ;

Var x ,y,z :entier ;

Procedure calculer(a ,b,c :entier)

Var

S :entier ;

Début

S←a+b ; c←S ;

Fin ;(fin de déclaration de la procédure)

DEBUT

Ecrire('donner deux valeurs') ;

Lire(x,y) ;

Calculer(x ,y,z) ; ←——— l'appel de la procédure

Ecrire ('voici le résultat ', z) ;

FIN.

6.4.6 Le passage des paramètres

L'association des paramètres avec l'algorithme principal se faite en deux modes (par variable ou par valeur)

1. **Le passage de paramètre par valeur**

C'est la valeur de la variable associé dans l'algorithme principale qui sera copié dans le paramètre et donc les actions du sous-algorithme ne peuvent pas modifier le contenu de la variable associée. Dans ce type de passage, on peut utiliser une constante comme paramètre lors de l'appel.

Exemple

```
Algorithme passage_par_valeur;

 Var   x :entier ;

 Procedure double(a :entier)
 Début
4  Ecrire(a) ;
5  a←a*2 ;
6  Ecrire(a) ;
7  Fin ;
DEBUT
1   x ←4 ;
2   écrire(x) ;
3  double(x ) ;
8  Ecrire(x) ;
FIN.
```

Le déroulement de cette séquence donne le résultat suivant :

Point de l'algorithme	Variables et paramètres	écran
1	x 4	
2	x 4	4
3	x 4 a 4	4
4	x 4 a 4	4 4
5	x 4 a 8	4 4
6	x 4 a 8	4 4 8
7	x 4 a 8	4 4 8
8	x 4	2 4 8 4

2. Le passage de paramètre par variable

Lors de passage de paramètres, le variable associé dans l'algorithme principale devient des alias du paramètre. Les actions du sous-algorithme affectent obligatoirement le contenu de la variable associée. Dans ce type de passage, on ne peut pas utiliser une constante comme paramètre lors de l'appel.

Pour identifier le paramètre transmis par variable, il sera précédé par le mot **var**

Exemple

```
Algorithme passage_par_variable;

    Var   x :entier ;

    Procedure double(vara :entier)
    Début
4   Ecrire(a) ;
5   a←a*2 ;
6   Ecrire(a) ;
7   Fin ;
```

DEBUT

1 x ←4 ;

2 écrire(x) ;

3 double(x) ;

8 Ecrire(x) ;

FIN.

Le déroulement de cette séquence donne le résultat suivant :

Point de l'algorithme	Variables et paramètres	écran
1	x 4	
2	x 4	4
3	x 4 a	4
4	x 4 a	4 4
5	x 8 a	4 4
6	x 8 a	4 4 8
7	x 8 a	4 4 8
8	x 8 a	3 4 8 8

6.5 Les fonctions

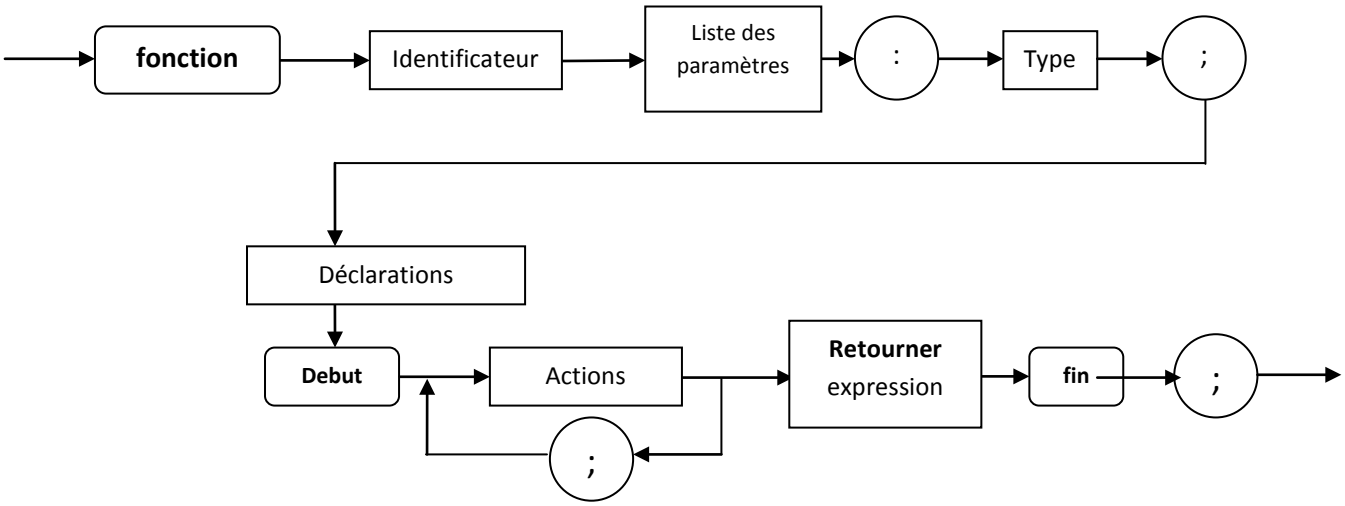
6.5.1 Définition

Les fonctions sont des sous-algorithmes constitués d'une suite d'instructions indépendantes dont l'objet est de calculer une valeur dépendant en générale , de paramètres citées dans sa déclaration. Une fonction retourne obligatoirement de résultat ou de valeur à l'algorithme qui l'a appelé.

6.5.2 Déclaration de la fonction

Syntaxe

Fonctionidentificateur (liste des paramètres) :type
Déclarationdes variable locales
Debut
Actions
….
Retourner expression
Fin ;



L'action « **Retourner** expression «, permet de transmettre la valeur de l'expression et met fin à l'exécution de la fonction.

Toute instruction placée après l'instruction **Retourner** est purement et simplement ignorée

6.5.3 Appel de fonction

L'appel d'une fonction correspond à une demande de son utilisation, ceci est fait dans l'algorithme principal ou dans un autre sous-algorithme. Afin d'appeler une fonction, on doit préciser :**son nom,** ainsi que les valeurs que l'on fournit pour les entrées.

On ne précise pas la valeur de lasortie, car c'est la fonction appelée qui est en charge de la fournir !

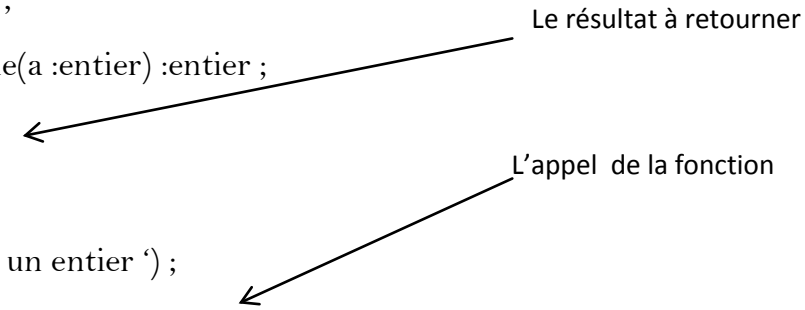
Exemple

L'exemple suivant contient la déclaration d'une fonction qui retourne le double d'un entier

Algorithme exemple_fonction;

```
    Var   x :entier ;

    fonctiondouble(a :entier) :entier ;
    Début
Retourner 2*a ;
Fin ;
DEBUT
    écrire ('donner un entier ') ;
    Lire(x) ;
   écrire ('son double est :' ; double(x)) ;
FIN.
```



6.5.4 Les fonctions prédéfinies

Tous les langages de programmation ont un certain nombre de fonctions qui permettent de connaitre directement des résultats afin de soulager le programmeur de lui éviter d'écrire des longs algorithmes.

Exemple

Calcule de sinus peut être appelé par : sin(x)

Série d'exercices n° :5

Exercice 01

Soit l'algorithme suivant :

```
Algorithme calcule ;
Var      X, Y, Z : entier ;
Procédure SomCar (A,B,C :entier)
Début
Ecrire(A,B,C) ; ……❷
A ← A * A ;
B ← B * B;
C ← A + B;
Ecrire(A,B,C) ;……❸
Fin ;
DEBUT   (**********Programme principal***********)
X ← 3; Y ← 4;  Z ← 0;
Ecrire(X,Y,Z) ;……❶
SomCar(X, Y, Z) ;
Ecrire (X,Y,Z) ;……❹
Fin.
```

Le travail demandé :

1- Dérouler l'algorithme et indiquer l'affichage ❶,❷,❸et ❹

2- Que remarquer vous ? proposez une correction

3- Remplacer la procédure par une fonction assurant le même calcule

Exercice 02

Soit la déclaration suivante :

```
Var X,Y,Z :entier ;
Procédure Traiter(A,B:entier ; var C: entier)
Début
Ecrire(A ,B,C) ;
A ← A + 1 ;
B ← 4 ;
C ← C * 2 ;
Ecrire(A ,B,C) ;
Fin ;
```

Debut X←1 ;Y←3 ;Z←5 ; Traiter(1,3,5) ; Ecrire(X,Y,Z) ; Fin .	Debut X←1 ;Y←3 ;Z←5 ; Traiter(X,Y,Z) ; Ecrire(X,Y,Z) ; Fin .	Debut X←1 ;Y←3 ;Z←5 ; Traiter(Z,Y,X) ; Ecrire(X,Y,Z) ; Fin .
Debut X←1 ;Y←3 ;Z←5 ; Traiter(2*X+1,Y,Z) ; Ecrire(X,Y,Z) ; Fin .	Debut X←1 ;Y←3 ;Z←5 ; Traiter(X,X,Z) ; Ecrire(X,Y,Z) ; Fin .	Debut X←1 ;Y←3 ;Z←5 ; Traiter(X,Y,Y) ; Ecrire(X,Y,Z) ; Fin .

Le travail demandé

Simuler les algorithmes suivant et indiquer ceux qui ont corrects, ceux qui sont incorrects à propos des appels ? justifier vos réponses

Exercice 03

Dérouler les algorithmes suivants :

```
Algorithme  v1 ;
Var  A, B : entier ;
Procédure Pr2(var A : entier)
Début
A ← A + 1 ; Ecrire(A) ;
Fin
Procédure pr1(var B : entier)
Début
A ← A + 1 ;
Pr2(A) ;
B ← B +1 ;
Pr2(B)
Ecrire( A, B) ;
Fin
Début (**programme principal**)
A ← 10 ;
Pr2(A) ;
Pr1(A) ;
B ← 10 ;
Pr2(B) ;
Pr1(B) ;
Ecrire (A, B) ;
Fin.
```

```
Algorithme  v2 ;
Var  A, B : entier ;
Procédure Pr2(var A : entier)
Début
A ← A + 1 ; Ecrire(A) ;
Fin
Procédure pr1( B : entier)
Début
A ← A + 1 ;
Pr2(A) ;
B ← B +1 ;
Pr2(B)
Ecrire( A, B) ;
Fin
Début (**programme principal**)
A ← 10 ;
Pr2(A) ;
Pr1(A) ;
B ← 10 ;
Pr2(B) ;
Pr1(B) ;
Ecrire (A, B) ;
Fin.
```

```
Algorithme  v3 ;
Var  A, B : entier ;
Procédure Pr2(  A : entier)
Début
A ← A + 1 ; Ecrire(A) ;
Fin
Procédure pr1(  B : entier)
Début
A ← A + 1 ;
Pr2(A) ;
B ← B +1 ;
Pr2(B)
Ecrire( A, B) ;
Fin
Début (**programme principal**)
A ← 10 ;
Pr2(A) ;
Pr1(A) ;
B ← 10 ;
Pr2(B) ;
Pr1(B) ;
Ecrire (A, B) ;
Fin.
```

```
Algorithme  v4 ;
Var  A, B : entier ;
Procédure Pr2( A : entier)
Début
A ← A + 1 ; Ecrire(A) ;
Fin
Procédure pr1(var B : entier)
Début
A ← A + 1 ;
Pr2(A) ;
B ← B +1 ;
Pr2(B)
Ecrire( A, B) ;
Fin
Début (**programme principal**)
A ← 10 ;
Pr2(A) ;
Pr1(A) ;
B ← 10 ;
Pr2(B) ;
Pr1(B) ;
Ecrire (A, B) ;
Fin.
```

Exercice 04

Ecrire une fonction *enminute* , qui calcule le nombre des minutes dans un nombre d'heures et un nombre de minutes.

-Ecrire une fonction qui renvoie le nombre des jours dans un mois.

Exercice 05

Ecrire une fonction nb_chiff_ent(n :entier) :entier

Qui retourne comme résultat le nombre de chiffre composant un entier N.

Les tableaux 7

7.1 Introduction

Un algorithme peut être amené à manipuler de nombreuses variables représentant des valeursdistinctes mais de même nature. Par exemple, un relevé de notesde plusieurs matières d'un étudiant nécessitera autant de valeurs réels que de matières à stocker.

Il est difficilement envisageable de définir « manuellement » autant de variables que de valeursà stocker. Les tableaux, en algorithmique, permettent de résoudre ce problème en proposant lacréation de plusieurs variables de même type, d'une manière très compacte.

7.2 Définition

Le type tableau représente un ensemble de valeurs portant le même type. C'est une structure qui stock dans une même variable un nombre fini d'éléments de même type.

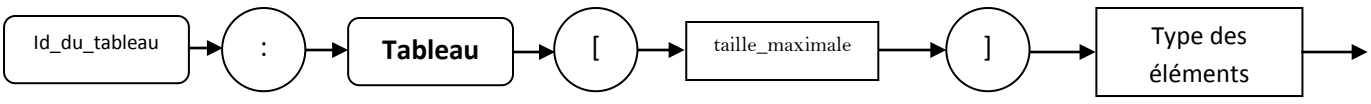
On distingue deux type de tableaux ; tableau à une seule dimension (vecteur) et tableaux à deux dimensions (matrice)

7.3 Les tableaux à une dimension

7.3.1 Déclaration

Syntaxe :

id_du_tableau : **tableau**[taille_maximale]type_des_éléments



Exemple

tab : tableau[100] entier

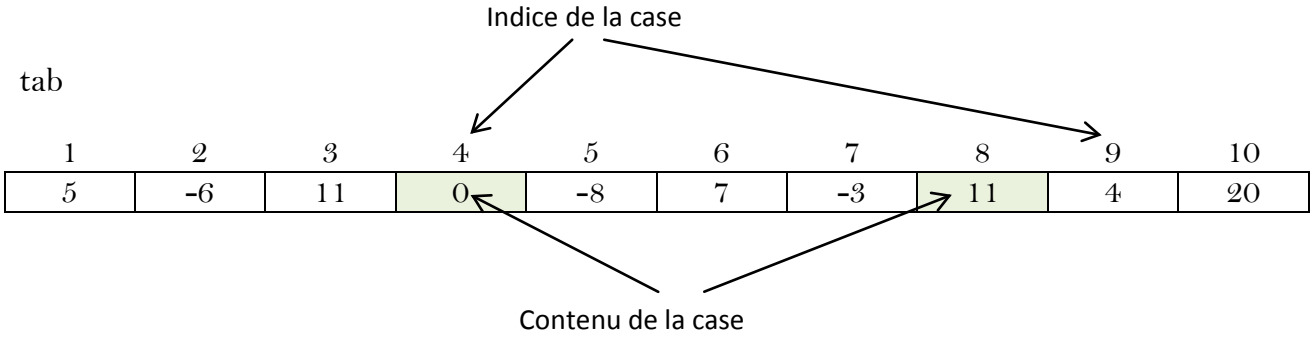
est la définition d'un tableau nommé tab qui peut stocker 100 valeursde type entier au maximum.

La taille maximale d'un tableau doit être une constante numérique et doit être défini lors de la déclaration du tableau.

Notations

L'accès aux éléments d'un tableau est direct, il suffit d'écrire l'identificateur du tableau suivi du numéro de la case.

Pour designer le contenu de la case 5 du tableau tab, j'écris : **tab[5]**



tab[4] : désigne le contenu de la case qui a comme indice 4 , et qui est 0

tab[11] : désigne le contenu de la case qui a comme indice 11 ; et qui n'existe pas

-Nous considérons 1 comme étant l'indice du premier élément ; certain langage commence par 0.

-Si aucune initialisation n'a été effectuée à un tableau, ses cases possèdent des valeurs aléatoires

-Chaque case d'un tableau peut être considérée comme une variable.

Exemple

7.3.2 La lecture d'un tableau

Var i ,n :entier

T : tableau[10] entiers

DEBUT

Ecrire('donner la taille de votre tableau :') ;

Lire(n) ;

Pour i ←1 à n faire

Début

Ecrire(‘ donner le ’, i, ’élément’) ;

Lire(T[i]) ;

Fin ;

FIN.

Le déroulement de cet algorithme donne :

n	i	T	affichage
			donner la taille de votre tableau :
			5
5			
	1	11	donner le , 1, élément 11
	2	11 10	donner le ,2, élément 10
	3	11 10 -6	donner le ,3, élément -6
	4	11 10 -6 0 3	donner le ,4, élément 0
	5	11 10 -6 0 3	donner le ,5, élément 3

7.3.3 L’affichage d’un tableau

Pour afficher un tableau T nous utilisons une boucle

Pour i ←1 à n faire écrire (T[i])

Série d'exercices n° :6

Exercice 01

Soit l'algorithme suivant :

```
Algorithme examen ;
Var t :tableau[1..100] des entiers
i,x :entier ;
Fonction Teste ( V :tableau) :entier
Début
j←1 ;
répéter
     Si ( V[ j ]< V[ j+1])  alors retourner 0
       sinon j←j+1 ;
jusqu'à (j=n)
retourner 1 ;
Fin ;
DEBUT
Lire(n) ;
pour i←1 a n faire lire(t[i]) ;
si Teste(t)=0 alors
debut
 i=1 ;
 répéter
  si t[i]<t[i+1] alors début
                          x←t[i] ;
                          t[i] ←t[i+1 ] ;
                          t[i+1] ←x ;
                          i←1 ;
                          fin
                 sinon  i←i+1 ;
  jusqu'à (i>=n)
fin ;
pour i←1 a n faire ecrire(t[i]) ;
FIN.
```

Le travail demandé :

1. Dérouler l'algorithme avec T :

6	9	4	8	12

8	6	4	2	1

2. Que fait cet algorithme ?

Exercice 02

Ecrire un algorithme qui affiche l'indice du plus petit élément d'un tableau d'entiers.

Exercice 03

Ecrire un algorithme permettant le calcul du nombre d'occurrences d'un élément donné dans un tableau.

Exercice 04

Ecrire l'algorithme permettant de calculer le produit scalaire de deux vecteurs réels u et v de dimension n :

$$u.v = \sum_{i=1}^{i=n} u[i].v[i]$$

Exercice 05

Ecrire un algorithme permettant de trier un tableau en ordre croissant

7.4 Les tableaux à deux dimensions

7.4.1 Définition

Tableau à deux dimensions ou matrice, sont des tableaux où chaque valeur est repérée par deux indices n lignes et colonnes.

3	0	1	9	-8	0
5	-4	-9	0	1	3
10	1	2	3	12	2
27	-15	13	20	15	6
29	17	11	17	-13	4

On accède à la i ème , j i ème valeur de la matrice M par la syntaxe

M[i,j]

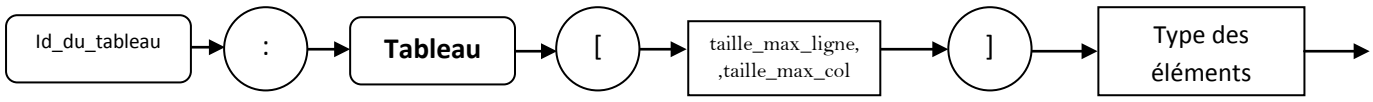
Exemple

M[3,1]←55 ; met la valeur 55 dans la case numéro 1 de la 3ème ligne.

Déclaration

Syntaxe :

id_du_tableau : **tableau**[taille_max_ligne,taille_max_col]type_des_éléments



id_max_ligne : nombre maximum de lignes .

id_max_col : nombre maximum de colonnes.

Exemple :

Mat : Tableau[10 ,15] entier

est la définition d'une matricede 10 lignes et 15 colonnes nommé Mat qui peut stocker des valeurs de type entier.

La manipulation des matrices nécessite l'utilisation de deux boucles imbriquée.

7.4.2 La lecture d'une matrice

```
Procedurelecture_matrice(M :tableau[10,10] entiers ; l,c : entiers)
Debut
Ecrire('donner le nombre de lignes ') ;
Lire(l) ;
Ecrire('donner le nombre de colonnes ') ;
Lire(c) ;
Pour i ←1 à l faire
Pour j ←1 à c faire
Début
Ecrire(' donner le ', i,j 'élément') ;
Lire(M[i,j]) ;
Fin ;
Fin ;
```

7.4.3 Affichage d'une matrice

```
Procedure affichage_matrice(M :tableau[10,10] entiers ; l,c : entiers)
Debut
Pour i ←1 à l faire
Pour j ←1 à c faire
écrire(M[i,j]) ;
Fin ;
```

Série d'exercices n° :7

Exercice 01

Quel résultat produira cet algorithme ?

```
Algorithme matrice ;
var
M :tableau[1, 2] Entier ;
i, j, val :Entier
Début
 Val ←1
Pour i ←1 à 2 faire
Pour j ←1 à 3 faire
debut
M[i, j] ←Val ;
Val ←Val + 1
Fin ;
Pour i ←0 à 1 faire
Pour j ←0 à 2 faire (Ecrire M[i, j]) ;
Fin.
```

Exercice 02

Ecrire l'algorithme qui saisit deux matrices A et B (m,n) par des nombres entiers, calcule la somme suivante C = 2*A-3*B puis affiche C.

Exercice 03

Ecrire un algorithme qui permet de :
- Saisir une matrice T(m,n) d'entiers.
- Calculer P le nombre des éléments pairs.
- Calculer R le nombre des éléments impairs.
- Afficher P et R.

Exercice 04

Soit une matrice carrée. Ecrire l'algorithme qui permet de faire la somme de la diagonale principale de cette matrice.

Exercice 05

Ecrire un algorithme qui remplace une matrice M par sa transposé.

Exercice 06

Dans une matrice donnée A, les éléments qui sont à la fois un maximum sur leur ligne et un minimum sur leur colonne sont appelés les points-cols. Ecrire un algorithme qui recherche et affiche les positions et les valeurs de tous les points-cols trouvés dans cette matrice.

8 Les enregistrements

8.1 Introduction

Nous nous retrouvons dans des situations où on est besoin de représenter dans une même variable des données de type hétérogène, c'est l'enregistrement ou structure qui fait l'objet de cette partie.

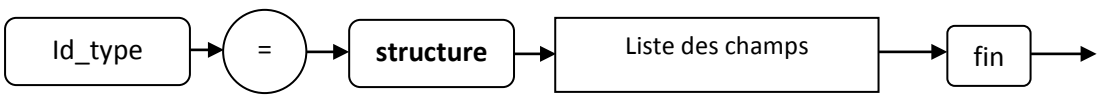
8.2 Définition

Un enregistrement est une structure de données composée d'un nombre fixe de composants appelées champs de type hétérogènes.

Déclaration

Syntaxe

```
Type id_type = structure
Champ 1: TypeElm1
Champ 2: TypeElm2
……….
Champ n: TypeElmn
Fin
```



Exemple

Type

etudiant =**structure**

Code :entier ;

Nom :chaine ;

Prenom :chaine ;

Age :entier ;

Fin ;

Var et1 :etudiant ;

Et1

code	**nom**	**prenom**	**Age**
013	ouhab	abdelkrim	19

8.3 Manipulation des enregistrements

Les champs d'une variable de type enregistrement sont traités comme des variable , chaque champ doivent être précédé par le no, de la varaible.

Exemple

Lire (et1.nom) : pour lire le champ nom de la variable et1

et1.age ←20 : pour affecter la valeur 20 au champ 20 de la variable et1

Les tableaux d'enregistrements

On peut déclarer un tableau d'enregistrements dans lequel chaque case contient un enregistrement.

Exemple

classe : tableau[1..20] de etudiant ;

Pour accéder à un champ

classe[i].nom : représente le champ nom de la case i du tableau classe

Série d'exercices n° :8

Exercice 01

Un compte en banque concerne une personne spécifiée par son **nom**, un numéro de compte et un **montant**.
Déclarez un enregistrement pour cette structure.

Exercice 02

Créer un tableau **Tab_Emp** qui contiendra les informations sur les **20** employés d'une entreprise (**Matricule, Nom, Salaire, Etat_civil**),

- Ecrire une procédure pour le remplir
- Ecrire une fonction qui affiche le nombre d'employés dont leur salaire est compris entre 1**5000** et 1**8000D**.

Exercice 03

Ecrire un algorithme qui lit deux nombres complexes **C1** et **C2** et qui affiche en suite leur somme et leur produit.
On utilisera les formules de calcul suivantes :

$(a + bi) + (c + di) = (a + c) + (b + d)i$
$(a + bi) * (c + di) = (ac - bd) + (ad + bc)i$

Exercice 04

Ecrivez un algorithme permettant de représenter les informations d'une référence bibliographique : le titre du livre, le nom de l'auteur, le nom de l'éditeur, l'année de publication et le nombre de pages.

Cet algorithme permet :

- La saisie des références (au minimum 2 et au maximum 150) dans un tableau,

-La saisie d'une année

- La recherche et l'affichage de tous les livres qui ont été publiés cette année.

Exemple de livre:

Algorithmes et langage pascal par kadda 2015 et qui compte 240 pages.

Solutions d'une partie des exercices

9

Série d'exercices n° :1

Exercice 01

L'expression est exprimée comme suit :

((a+(12-b)/c)/(15*(d/e)))+ (f/(g/h))

Exercice 02

1.(j*5)+1

e e e valide de type entier

2. b1-(r*s)

b r r Invalide (opération impossible entre booléenne et réel)

3. (m+5.0)*s

e r r valide de type réel (E⊂ R)

4. (b1 ou b2) et (r=6.2)

b b valide de type booléen

5. (b/8) et (m=i)

Invalide (1 : b non déclarée 2 : et est un opérateur booléen)

6. b2 et (j<k)

b Invalide (k non déclarée)

7. (i*8) - (c1/4.2)

Invalide (c1/4.2 , / opération impossible entre caractère et réel)

8. (c2='b1')et(i=j*3)

Invalide (c2 est n caractère et 'b1' est une chaine de caractère

9. b1 et non(j>5)

b b valide (b et b)

Exercice 03

I

L'action	a	b	c	w	x	y	z
a← 1	**1**	/	/	/	/	/	/
b ←2	**1**	**2**	/	/	/	/	/
c ← 3	**1**	**2**	**3**	/	/	/	/
w←a + b * c ;	**1**	**2**	**3**	**7**	/	/	/
x←a * b + c ;	**1**	**2**	**3**	**7**	**5**	/	/
y←a + c div b ;	**1**	**2**	**3**	**7**	**5**	**2**	/
z← c div a + b	**1**	**2**	**3**	**7**	**5**	**2**	**5**

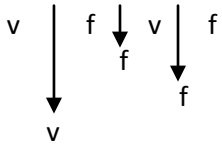
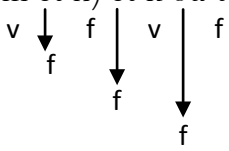
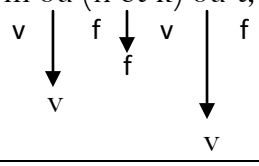
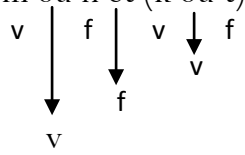
w←a + b * c **1 + 6** **7**	x←a * b + c **2 + 3** **5**	y←a + c div b **1 + 1** **2**	z← c div a + b **3 + 2** **5**

II

L'action	a	b	c	d	w	x	y	z
a← 1	**1**	/	/	/	/	/	/	/
b ←2	**1**	**2**	/	/	/	/	/	/
c ← 3	**1**	**2**	**3**	/	/	/	/	/
d ← 4	**1**	**2**	**3**	**4**	/	/	/	/
w← a + b*c-d ;	**1**	**2**	**3**	**4**	**3**	/	/	/
x ← (a + b)*c-d ;	**1**	**2**	**3**	**4**	**3**	**5**	/	/
y← a + (b*c)-d ;	**1**	**2**	**3**	**4**	**3**	**5**	**3**	/
z←a + b*(c-d) ;	**1**	**2**	**3**	**4**	**3**	**5**	**3**	**-1**

w←a + b * c - d **1 + 6** **7 - 4** **3**	x←(a + b) * c - d **3 * 3** **9 - 4** **5**	y←a + (b*c) - d **1 + 6** **7 - 4** **3**	z← a + b * (c-d) **2 * -1** **1 + -2** **-1**

III

v2← m ou n et k et t ; 	v3← (m et n) et k ou t 
v4← m ou (n et k) ou t; 	v5← m ou n et (k ou t); 

Série d'exercices n° :2

Exercice 01

Le déroulement de l'algorithme remise1 ;

prix	taux	remise	Affichage
15			
15	10.5		
15	10.5	157.5	
15	10.5	1.575	
13.425	10.5	1.575	
13.425	10.5	1.575	**13.425**

remise 1 donne toujours les mêmes résultats

Le déroulement de l'algorithme remise2 ;

prix	taux	remise	Affichage
			donner le prix svp
200			
200	25		donner le taux de remise svp'
200	25	5000	
200	25	50	
150	25	50	
150	25	50	**150**

remise 2 ,calcule le nouveau prix après la remise , selon la valeur de l'ancien prix et le taux de remise saisi par l'utilisateur (lire(prix) et lire(taux))

Exercice 04

Algorithme courant ;

Var

V,R,I : réel ;

Debut

Ecrire('donner la valeur de la résistance ') ; lire(R) ;

Ecrire('donner la valeur de la tension ') ; lire(V) ;

I←V/R ;

Ecrire('la valeur du courant', I) ;

Fin.

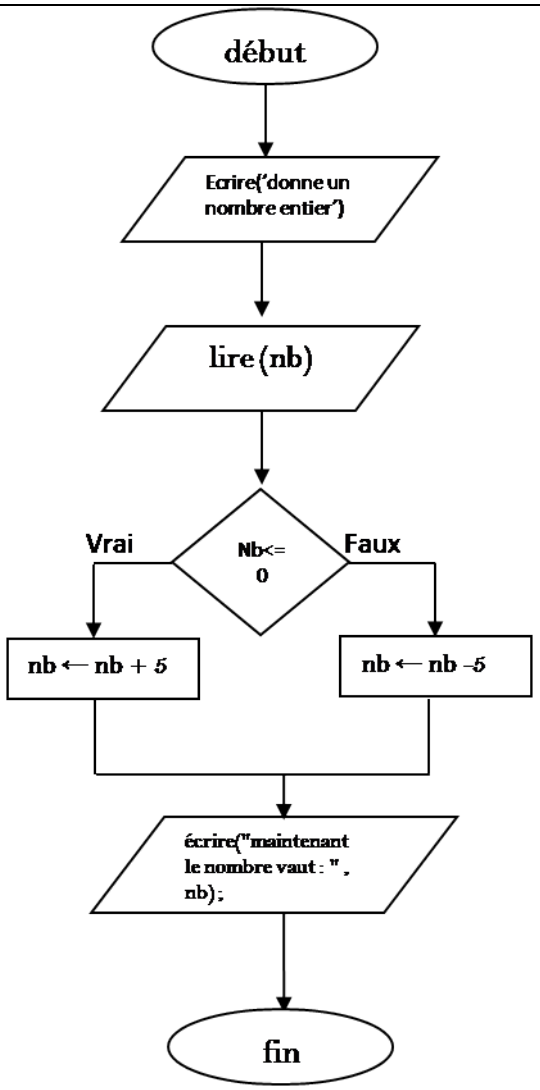
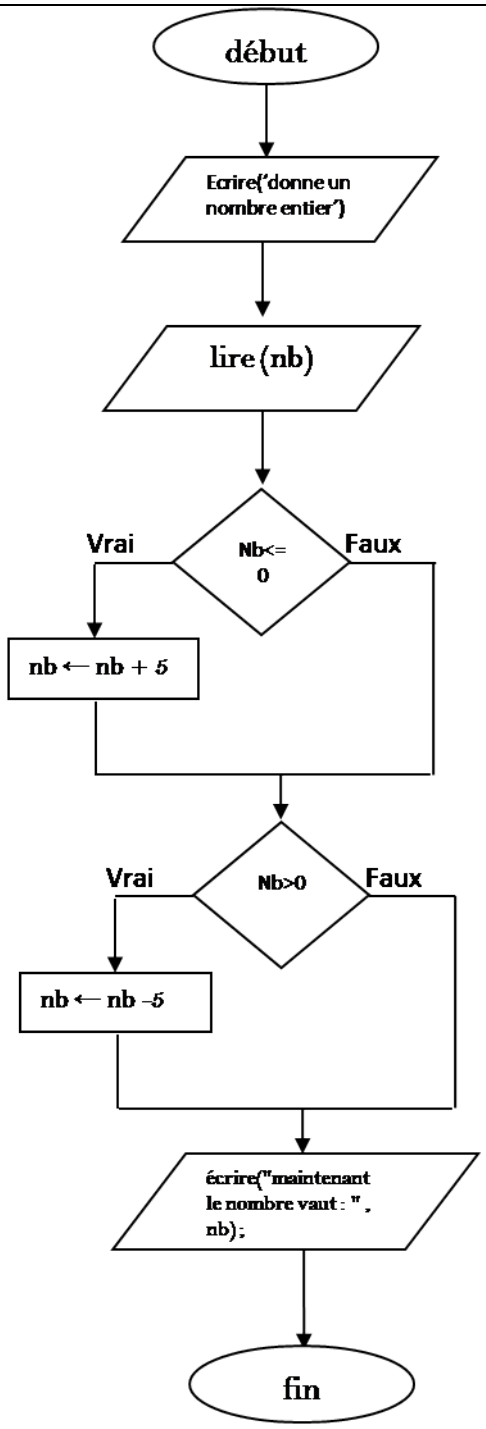
Exercice 07

```
Algorithme somme_argent ;
Var
Somme , B1000,b500,p200,p100,p50,p1 : entier ;
Debut
Ecrire('donner  une somme d'agent') ;
Lire(somme) ;
B1000 ←somme div 1000 ;
Somme ← somme mod 1000 ;
B500 ← somme div 500 ;
Somme ← somme mod 500 ;
P200 ← somme div 200 ;
Somme ← somme mod 200 ;
P100 ← somme div 100 ;
Somme ← somme mod 100 ;
P50 ← somme div 50 ;
P1 ← somme mod 50 ;
Ecrire ( ' cette somme contient :' )
Ecrire( b1000 , ' billet de 1000') ;
Ecrire( b500 , ' billet de 500') ;
Ecrire( p200 , ' pièce de 200') ;
Ecrire( p100 , ' pièce de 100') ;
Ecrire( p50 , ' pièce de 50') ;
Ecrire( p1 , ' pièce de 1 ') ;
Fin.
```

Série d'exercices n° :3

Exercice 01

1/ les organigrammes

Algorithme test_1	Algorithme test_2
	

-

1/ le déroulement des agorithmes:

Algorithme test_1

Nb=5

nb	Nb<=0	affichage
		écrire ("Donner un nombre entier") ;
5		
	faux	
0		
		maintenant le nombre vaut : 0

Nb=0

nb	Nb<=0	affichage
		écrire ("Donner un nombre entier") ;
0		
	Vrai	
5		
		maintenant le nombre vaut : 5

Nb=-5

nb	Nb<=0	affichage
		écrire ("Donner un nombre entier") ;
-5		
	Vrai	
0		
		maintenant le nombre vaut : 0

Algorithme test_2

Nb=5

nb	Nb<=0	Nb>0	affichage
			écrire ("Donner un nombre entier") ;
5			
	faux		
		vrai	
0			
			maintenant le nombre vaut : 0

Nb=0

nb	Nb<=0	Nb>0	affichage
			écrire ("Donner un nombre entier") ;
0			
	vrai		
5			
		vrai	
0			
			maintenant le nombre vaut : 0

Nb=-5

nb	Nb<=0	Nb>0	affichage
			écrire ("Donner un nombre entier") ;
-5			
	vrai		
0			
		faux	
			maintenant le nombre vaut : 0

3/ Les deux algorithmes ne sont pas équivalents

Exercice 03

```
Algorithme magasin ;
Var
h :0..23 ;
j :0..
Debut

Ecrire('donner le jour 0 :dimanche , 1 lundi , 2 mardi, 3 mercredi, …..') ;
Lire(j) ;
Donner('l''heure entre 0..23') ;
Lire(h) ;
Ouvert ← ((h >= 10) et (h <= 14) et (j<> 5)) ou ((h >= 16) et (h <= 20) et (j<5));
Si ouvert=vrai alors ecrire('ouvert')
Sinon ecrire('fermé') ;
Fin.
```

Exercice 07

```
Algorithme lycee ;
Var
NbAbs , NbAbsNJ : entier
MoyGen, MoyMP : réel
DEBUT
Ecrire("Donner nombre d'heurs d'absence : ") ;
Lire('NbAbs) ;

Si (NbAbs > 40) alors Ecrire(" élève ajournée pas de passage en deuxième année")
Sinon
Debut
                Ecrire(" le nombre d'heurs d'absence non justifier : ") ;
                Lire('NbAbsNJ) ;
                Si (NbAbsNJ > 20) alors Ecrire(" élève ajournée pas de passage en deuxième
année")
                 Sinon Ecrire(" le nombre la moyenne générale  : ") ;
                        Lire(MoyGen) ;
Si (MoyGen >=10) alors
 Si (MoyGen >=13 )alors ecrire(" élève passe en 2 année  il aura l'option choisie dans la fiche de
veux ")
 Sinon Debut
          Ecrire(" la moyenne des matières principales pour l'option choisie  ") ;
           Lire(MoyMP) ;
           SI (MoyMP >=10 )alors ecrire(" donner l'option choisie  ")
           Sinon   ecrire(" choisir l'option qui convient à ses notes sans voir la fiche de veux  ") ;
          FIN ;
Sinon
Si (MoyGen <9.30)  alors Ecrire(" élève ajournée pas de passage en deuxième année")
Sinon SI (NbAbs < 10)  alors Ecrire(" élève passe en 2 année et la troisième option lui sera
accordée ") .
Fin ;

FIN.
```

Série d'exercices n° :4

Exercice 01

Tanque1		Tanque2		Tanque1	
nb	écran	nb	écran	x	écran
10		10		1	
	10		Le nombre vaut 10		1
20				2	
	20				2
30				3	
	30				3
40				4	
	Le nombre vaut 40				4

Exercice 03

```
Algorithme parfait ;
Var
I,n,S : entier ;

Debut
Ecrire(' donner un nombre ') ;
Lire(N) ;
S ←0 ;
Pour i←1 à N-1 faire
Si (N mod i =0) alors s ←s+i ;

Si ( N=S) alors ecrire (N, ' est un nombre parfait ') ;
Fin.
```

Exercice 05

```
Algorithme Referendum ;
Var
vote : caractere /* le vote (O/N) */
nbVotants : entier /* nombre de votants */
nbOui : entier /* nombre de oui */
nbNon : entier /* nombre de non */
nbBlanc : entier /* nombre de blancs */
nbNul : entier /* nombre de nuls */
tauxOui : réel /* pourcentage de oui */
```

```
tauxNon : réel /* pourcentage de non */
tauxBlanc : réel /* pourcentage de blancs */
tauxNul : réel /* pourcentage de nuls */
Début
nbVotants←0 ;
nbOui←0 ;
nbNon←0 ;
répéter
écrire(quel vote (O/N/B/X) ?)
lire(vote)
selon vote dans
'O' : nbOui←nbOui + 1 ;
'N' : nbNon←nbNon + 1 ;
'B' : nbBlanc←nbBlanc + 1 ;
'X' : nbNul←nbNul + 1 ;
Fin ;
nbVotants←nbVotants + 1 ;
jusqu'à vote = 'F' ;
nbVotants←nbVotants– 1 ;
si nbVotants<> 0 alors
debut
tauxOui←nbOui / nbVotants ;
tauxNon←nbNon / nbVotants
tauxBlanc←nbBlanc / nbVotants
tauxNul←100 - tauxOui - tauxNon–tauxBlanc ;
écrire(nombre de votants ", nbVotants) ;
écrire("oui : ", nbOui, " pourcentage ", tauxOui) ;
écrire("non : ", nbNon, " pourcentage ", tauxNon) ;
écrire("blanc : ", nbBlanc, " pourcentage ", tauxBlanc) ;
écrire("non : ", nbNul, " pourcentage ", tauxNul) ;
fin ;
sinonécrire("pas de votant")
Fin.
```

Série d'exercices n° :5

Exercice 01

1/

Algorithme principal		
x	y	z
3	4	0

somcar		
a	b	c
3	4	0
9	16	25

Affichage				
❶	3	4	0	dans l'algorithme principal
❷	3	4	0	dans somcar
❸	9	16	25	dans somcar
❹	3	4	0	dans l'algorithme principal

2/La procédure ait modifié les valeurs des paramètres x , y et z , mais en la quittant, les paramètres récupèrent leurs valeurs initiales (passage par valeur)

La solution est de modifier le mode de transmission du paramètre c dans la procédure et la rendre par variable . **SomCar (A,B :entier, var C : entier)**

3/

```
Fonction SomCar (A,B :entier) :entier
Début
A ← A * A ;
B ← B * B;
 retourner A + B ;
Fin ;
```

Exercice 02

X←1 ;Y←3 ;Z←5 ; Traiter(1,3,5) ; Appel incorrect la transmission se faite par des variables, pas par des constantes	Affichage : 1 3 5 2 4 10 1 3 10	Affichage : 5 3 1 6 4 2 2 3 5
Affichage : 11 3 5 12 4 10 1 3 10	Affichage : 1 1 5 2 4 10 1 3 10	Affichage : 1 3 3 2 4 6 1 6 5

Exercice 04

```
fonction nb_chiff_ent(n :entier) :entier
var nb :entier ;
debut
nb←0 ;
répéter
n ←n div 10 ;
nb ←nb+1 ;
jusqu'à ( n=0)
retourner nb ;
fin ;
```

Série d'exercices n° :6

Exercice 01

1/ T : n=5

6	9	4	8	12

Teste(t)

j	V[j]	V[j+1]
1	6	9

6 <9 alors retouner **0**

Teste(t) va retourner 0
Fin de teste

Teste(t)=0 alors on va exécuter la boucle

i	T				
1	6	9	4	8	12
1	9	6	4	8	12
2	9	6	4	8	12
3	9	6	4	8	12
1	9	6	8	4	12
2	9	6	8	4	12
1	9	8	6	4	12
2	9	8	6	4	12
3	9	8	6	4	12
4	9	8	6	4	12
1	9	8	6	12	4
2	9	8	6	12	4
3	9	8	6	12	4
1	9	8	12	6	4
2	9	8	12	6	4
1	9	12	8	6	4
1	12	9	8	6	4
2	12	9	8	6	4
3	12	9	8	6	4
4	12	9	8	6	4
5					

L'affichage : 12 9 8 6 4

2/

T : n=5

8	6	4	2	1

Teste(t)

j	V[j]	V[j+1]
1	8	6
2	6	4
3	4	2
4	2	1
5		

J=5 (n) alors retourner 1
Teste(t) va retourner 1
Fin de teste

Test(t)=1 donc on n'exécute pas la boucle

L'affichage : 8 6 4 2 1

II/ que fait cet algorithme ? **L'algorithme trie un tableau en ordre décroissant.**

Exercice 02

```
Algorithme exo2 ;
Var
I,n,pos,min : entier ;
T : tableau[1..50] des entiers ;
Debut
Ecrire(' donner la taille de votre tableau ') ;
Lire(n) ;
Pour i←1 à n faire
Debut
Ecrire('T[',i,']=') ;
Lire(T[i]) ;
Fin ;
Min ← t[1] ;
pos←1 ;
pour i □2 à n faire debut
si t[i]<min alors min←t[i] ;
pos←i ;
fin ;
ecrire ( Min , 'est la plus petite valeur du tableau dans la case ',pos) ;
fin.
```

Exercice 03

```
Algorithme exo3 ;
Var
I,n,val,occ : entier ;
T : tableau[1..50] des entiers ;
Debut
Ecrire(' donner la taille de votre tableau ') ;
Lire(n) ;
Pour i←1 à n faire
Debut
Ecrire('T[',i,']=') ;
Lire(T[i]) ;
Fin ;
Ecrire('donner la valeur à rechercher') ;
Lire(val) ;
occ←0 ;
pour i←1 à n faire si (t[i]=val )alors occ←occ+1 ;
ecrire(val , ' se trouve' , occ , 'fois dans le tabelau' ) ;
FIN.
```

Série d'exercices n° :7

Exercice 01

Algorithme Somme_Matrices ;
Type Mat=tableau ⌈1..20,1..20⌉ de entier ;
Var

A, B, C : M,N ; i, j : entier ;

A,B,C : mat ;

Procédure lecture (var X :Mat) ;
Début
Ecrire('donner les dimensions de votre matrice :') ;
ecrire('Ligne :') ;lire(n) ;
ecrire('colonne :') ;lire(m) ;
Pour i de 1 à n Faire
Pour j de 1 à m Faire
debut
ecrire('X⌈',i,',',j,'⌉=');
Lire (X⌈i,j⌉) ;
Fin ;
Fin;

Procédure affichage (X :Mat) ;
Début
Pour i de 1 à n Faire
debut
Pour j de 1 à m Faire ecrire (X⌈i,j⌉,' ') ;
ecrire( );// sauter la ligne
Fin ;
Fin;

DEBUT
Lecture(A) ;
Lecture(B) ;
Pour i de 1 à n Faire
Pour j de1 à m Faire
C⌈i, j⌉← 2*A⌈i,j⌉-3*B⌈i,j⌉ ;
Affichage(c) ;

Fin.

Exercice 03

```
Algorithme Somme_Matrices ;
Type
 mat=tableau [1..50,1..50] des entiers ;
Var
 i, j,n ,m, D  entier ; A :mat ;
Procédure lecture (var X :Mat) ;
Début
Ecrire('donner les dimensions de votre matrice carré :') ;
lire(n) ;
Pour i de 1 à n Faire
Pour j de 1 à n Faire
debut
ecrire('X[',i,',',j,']=');
Lire (X[i,j]) ;
Fin ;
Fin;

Debut
Lecture(A) ;
D←0 ;
Pour i de 1 à n Faire
D←D+A[i,i] ;

Ecrire(' la somme de la diagonale est ' :D) ;

FIN.
```

Exercice 06

```
Algorithme cols ;
  M , max,min : tableau [50..50] des entiers ;
  N, M,I,J,X,C : entier ;
DEBUT
ecrire('Nombre de lignes: ');
lire(N );
ecrire('Nombre de colonnes ');
lire(M );
pour I←1 à N faire
pour J←1 à M faire    debut
                      ecrire('M[',I',',J,']=');
                       lire(M[I ,J]);
                       fin ;
```

```
/* Recherche du maximum sur la ligne I */
pour I←1 à N faire
Debut
        X←M[I ,1];
pour J←2 à M faire  debut
     si (M[I,J]>X )alors  X←M[I ,J];
                        fin ;
pour J←1 à M faire debut
                        si (M[I ,J]=X) alors MAX[I ,J] ←1;
                                          sinon MAX[I,J] ←0;
                        fin,
 fin ;
 /* Recherche du minimum sur la colonne J */
pour J←1 à M faire
Debut
      X←M[1 ,J];
 pour I←2 à N faire  debut
                        si (M[I,J]<X) alors X←M[I,J];
                        fin ;
 pour I←1 à N faire   debut
                        si (M[I,J]=X) alors  MIN[I,J] ←1;
                                         sinon   MIN[I,J] ←0;
                         fin ;
Fin ;


 Ecrire('les Points - cols sont :');
C←0 ;
pour I←1 à N faire
pour J←1 à M faire  debut
                        si (MAX[I,J]=1 ) et (MIN[I,J]=1) alors
             debut
             C←C+1;
              ecrire(M[I,J],' est maxiumum sur', I, ' et  minimum sur' ,J);
            fin ;
                         fin ;
 si (C=0)
    ecrire('La matrice ne contient pas de points-cols');
FIN.
```

Série d'exercices n° :8

Exercice 01

```
Type
Compte = Structure nom :chaine ;
Num :chaine ;
Montatnt :reel ;
Fin ;
```

Exercice 02

```
Type emp=structure
Matricule,nom :chaine ;
Salaire :reel ;
Etat_civil :caractere ;
Fin ;
Tab_emp=tableau[1..20] des emp ;
Procedure remplir(t :tab_emp)
Var i :entier ;
Debut
Pour i←1 à 20 faire
Debut
Ecrire('Matricule :') ;lire(t[i].matricule) ;
Ecrire('Nom :') ;lire(t[i].nom) ;
Ecrire('salaire :') ;lire(t[i].salaire) ;
Ecrire('Etat civil : m,d,v,c') ; lire(t[i].etat_civil) ;
Fin ;
Fin ;
Procedure nbsalaire(t :tab_emp) :entier
Var i,s : entier ;
Debut
S←0 ;
Pour i←1 à 20 faire
Debut
Si (t[i].salaire >=15000) et (t[i].salaire <=15000) alors S←S+1 ;
Fin ;
Retourner(S) ;
Fin ;
```

Exercice 03

```
Algorithme complexe ;
Type
Complexe =structure
a :reel ; b :réel ;
fin ;
var
c1,c2 ,som,pro: complexe ;
debut
ecrire('donner la partie réel du premier nombre') ;
lire(C1.a) ;
ecrire('donner la partie imaginaire du premier nombre') ;
lire(C1.b) ;
ecrire('donner la partie réel du deuxième nombre') ;
lire(C2.a) ;
ecrire('donner la partie imaginaire du deuxième nombre') ;
lire(C2.b) ;
som.a←c1.a+c2.a ;
som.b← c1.b+c2.b ;
pro.a ←(c1.a*c2.a-c1.b*c2.b) ;
pro.b ←(c1.a*c2.b+c1.b*c2.a) ;
ecrire(' la somme est ', som.a+'i'+som.b) ;
ecrire(' le produit est ', pro.a+'i'+pro.b) ;
fin ;
```

Références bibliographiques

- Algorithme et programmation en PASCAL, PATRICK COUSOT, BERTI Editions , 1993
- Le leader de l'algorithmique , cours et exercices avec corrections , CHAHID KHICHANE , EL MAARIFA Edition , 2004
- Exercices et problèmes d'algorithmique , Nicolas Flasque , Helen Kassel , Franck Lepoivre, Boris Velikson , Dunod Edition , 2010
- Cours d'algorithmique 1ere année ingénieur , Dr Slama , université de sidi belabes , 1994
- Algorithmiques & structures de données , support de cours , Nooman MAHJOUB, Moez CHEBI , MENSI Ali , Institut Supérieur des Etudes Technologiques de Zaghouan, 2015