Université Djilali BOUNAAMA Khemis Miliana	**Année Universitaire 2018/2019 (2ème Semestre)**
Faculté des Sciences et de la Technologie	**Matière : Informatique 2 (Cours)**
Département : Sciences de la Matière	**Chapitre n°2**
1ère Année ST-SM	**Notions d'Algorithme, Organigramme et Langage de Programmation**

Notion d'Algorithme, Organigramme et Langage de Programmation

I. Etapes de Résolution d'un Problème (Mathématique) en Utilisant l'Outil Informatique

La résolution d'un problème mathématique en algorithmique passe par six étapes fondamentales :

1. Analyse du problème.
2. Etablir l'algorithmique.
3. Programmation (utilisation d'un langage de programmation).
4. Vérification et correction du programme.
5. Exécution du programme.

Exemple : Addition de deux nombres réels.

II. Notions d'Algorithme

Un **algorithme** est une suite d'action qui, correctement exécutées donneront le résultat désiré (attendu).

Un **algorithme** est le résultat de la décomposition d'un problème complexe en opérations élémentaires à exécuter en plusieurs étapes successives.

Un **algorithme** est une séquence (suite) d'actions élémentaires qui, exécutées par un processeur bien défini réalisera un travail bien précis (demandé).

Un **algorithme** est une suite de règles, de raisonnements ou d'opérations, qui transforment des grandeurs données (données d'entrée) en d'autres grandeurs (données de sortie).

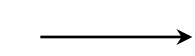
Structure Générale d'un Algorithme

Un algorithme est composé de :

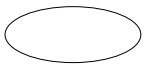
Parties principale	exemple
a. L'entête : qui contient le nom de l'algorithme permettant de l'identifier.	addition;
b. Les déclarations de constantes, variables, structure …….. : cette partie de l'algorithme permet de déclarer tout type de variable, constante ou structure.	**const** a, b, pi; **var** x, y, delta; **tableau** A, B, T1;
c. Les déclarations de fonctions et procédures ……. : dans cette partie toutes les fonctions et les sous programmes du problème sont déclarées.	**fonct** f1, f2, fon;
d. Corps de l'algorithmique ……. : cette partie de l'algorithme contient toutes les instructions permettant de traiter et de résoudre le problème posé.	**début de l'algorithme** instruction1; instruction2; instruction3; **fin de l'algorithme**

III. Notions d'Organigramme

Un organigramme est un schéma symbolique conventionnel qui illustre les étapes d'un algorithme et leurs relations. Son utilité s'impose, en particulier, lorsque le problème à résoudre est compliqué. L'organigramme permet de schématiser globalement le problème à traiter. On utilise généralement les symbole suivant :

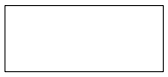


Flèche : indique le sens du traitement

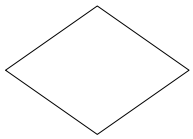


Début et Fin de l'organigramme

Entrée et Sortie de données



Initialisations, Calcul divers



Décision (Test)

IV. Notions de Langage de Programmation

Un langage de programmation est un langage artificiel comprenant un ensemble de caractères, de symboles et de mots régis par des règles qui permettent de les assembler, utilisé pour donner des instructions à une machine.

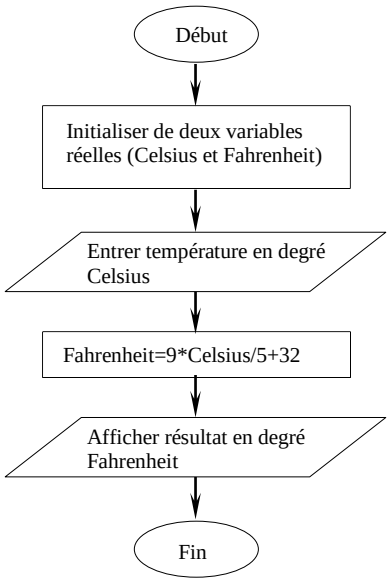
Les langages de programmation les plus utilisés à l'UKM sont : PASCAL, FORTRAN, MATLAB, C, C++…etc

Dans notre cours nous allons adopter le langage de programmation **FORTRAN**[1]

[1] De Formula Translation (**For**Tran), est le premier langage de programmation évolué. Il a rendu possible d'utiliser des noms symboliques pour la représentation des quantités mathématiques, et d'écrire des formules mathématiques dans des formes

V. Exemple pratique d'Algorithme-Organigramme-FORTRAN

Problème mathématique : Soit C : température en degré Celsius et soit F : température en degré Fahrenheit. Ecrire l'algorithme qui permet la conversion de la température du **degré Celsius** en **degré Fahrenheit**. Représenter l'organigramme correspondant. Ecrire le programme Fortran convenable. La formule de conversion est : $\boldsymbol{F = 9 \times C/5 + 32.}$

Algorithme	Organigramme	Programme-2-1 Fortran
Algorithme conversion; Tc, Tf : réels; **Début**; **Lire** (Tc); Tf ← 9*Tc/5+32; **Ecrire** (Tf); **Fin.** **Remarque** : le symbole ← se lit : reçoit la valeur de		**program** conversion **real** Tc, Tf **read**(*,*)Tc Tf = 9*Tc/5+32 **write**(*,*)Tf **end program** conversion

VI. Structure générale d'un programme Fortran

1. Le nom du programme est introduit par le mot clé **program**, le nom du programme n'est pas obligatoire en Fortran.
2. Dans la partie déclarative du programme les variables utilisées doivent être déclarées avec leurs types correspondant. Dans le programme précédent les deux variables Tf (température en degré Fahrenheit) et Tc (température en degré Celsius) sont des réelles.
3. Dans la partie corps du programme le problème posé est traité et résolus.
4. la fin du programme est marquée par le mot clé **end** (tout simplement) ou bien **end program** <nom du programme>

Exercices :

1. Ecrire l'algorithme, l'organigramme et le programme Fortran qui permettent de résoudre l'équation ax+b=0.
2. Ecrire l'algorithme, l'organigramme et le programme Fortran qui permettent de calculer la valeur de F telle que : $F = 5x^2 - 2x + 15$.

Remarque : Demander la version convenable du logiciel Fortran auprès de vos chargés de cours, TD ou TP.

raisonnables et compréhensibles, telle que X = B/(2*A). L'idée du Fortran a été proposée en 1953 par John Backus, à New York, et le premier programme en Fortran a été exécuté en Avril 1957.

Université Djilali BOUNAAMA Khemis Miliana	**Année Universitaire 2018/2019 (2ème Semestre)**
Faculté des Sciences et de la Technologie	**Matière : Informatique 2 (Cours)**
Département : Sciences de la Matière	**Chapitre n°3**
1ère Année ST-SM	**Eléments préliminaires d'Algorithme-Organigramme-Fortran**

Eléments préliminaires d'Algorithme-Organigramme-Fortran

I. Déclaration des variables et des constantes

Dans la partie déclarative toutes les variables et constantes du problème à traiter doivent être identifiées par un **identificateur** et déclarées avec leur types.

Déclaration des constantes :

Algorithme	Programme Fortran	Exemple Fortran
Const pi = 3.141593	Type de la constante, **PARAMETER** :: g = 9.81	Ex1 : **real**, **PARAMETER** :: g = 9.81 Ex2 : **integer**, **PARAMETER** :: theta = 45

Déclaration des variables :

Algorithme	Programme Fortran	Exemple Fortran
Var x, y, z : type; Ex1 : **Var** x, y, z : réel;	Type de la variable, identificateurs (séparés par des virgules)	Ex1 : **real** x, a, delta Ex2 : **integer**, theta, x1, T Ex3 : **character** lettre, section, groupe

II. Identificateurs

Toutes les variables et constantes du problème à traiter

Types des identificateurs

Type entier : il s'agit des nombres entiers ……-6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, ……, en Fortran on les appelle integer (prononcé intidjer).

Algorithme	Programme Fortran
Var x1, y, z : entier;	**integer**, x1, y, z

Type réel : il s'agit des nombres réels appartenant à ***R***, en Fortran on les appelle real.

Algorithme	Programme Fortran
Var z2, alpha, B, T : réel;	**real** x1, y, z z2, alpha, B, T

Type caractère : les caractères sont en général les lettres (de a à z) majusculs ou minusculs, les chiffres (de 0 à 9), les ponctuations (la virgules '**,**', le point-virgule '**;**' ', le point '**.**' …).

Algorithme	Programme Fortran
Var salle, groupe, section : caractère;	**character** salle, groupe, section

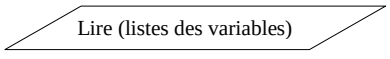
Type chaine de caractères : les chaînes de caractères sont un ensemble de caractères.

Algorithme	Programme Fortran
Var nom, prenom, adresse : chaîne de caractère;	**Character***n nom, prenom, adresse

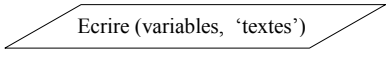
n dans **character***n est un nombre entier positif permet de définir la longueur de la chaine de caractère (i.e. nombre de caractères composant la chaine.

III. Actions élémentaires

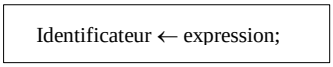
Action de lecture de données (Saisie) : Permet d'introduire les données utilisées dans le problème.

Algorithme	Organigramme	Programme en langage Fortran
Lire (Listes des variables);		**Read(*,*)** Id1, Id2, Id3

Action d'écriture (d'affichage) de résultats : Permet d'écrire ou d'afficher les résultats d'un traitement.

Algorithme	Organigramme	Programme en langage Fortran
Ecrire (variables, 'textes');		**Write(*,*)** x1, T, delta **Print***, x1, T, delta

Action d'affectation : Permet d'affecter une valeur à une variable.

Algorithme	Organigramme	Programme en langage Fortran
Identificateur ← expression;		Identificateu r= expression Ex : x1=a*b (et on lit x1 **reçoit** la valeur du produit a fois b.

Remarque : l'expression désigne une valeur, exprimée par composition d'opérateurs appliqués à des opérandes, qui sont : des valeurs, des constantes, des variables, des appels à des fonctions ou des sou-expression.

Ex1 : f=2*x, **Ex2** : z=3*cos(alpha), **Ex3** : B=3*exp(-x*x)

IV. Opérateurs (Opérateurs les plus utilisés)

Opérateurs arithmétiques

Opérateurs	Priorité	Signification	Exemple
((……))	1	Parenthèse	
**	2	Exponentiation	2 ** 4 (=24)
*	3	Multiplication	2 * A
/	4	Division	B / DELTA
+	4	Addition	A + 6.9
-	4	Soustraction	A - 6.9

Opérateurs relationnels

Opérateurs	Signification	Exemple Fortran
<	Inférieur à	a < b ou a .LT. b
<=	Inférieur ou égale à	a <= b ou a .LE. b
>	Supérieur à	a > b ou a .GT. b
>=	Supérieur ou égale à	a >= b ou a .GE. b

Opérateurs de comparaison

Opérateurs	Signification	Exemple Fortran
= =	Egale à	a = = b ou a .EQ. b
/=	Différent de	a /= b ou a .NE. b

Opérateurs logiques

Opérateurs	Signification
.NOT.	Négation logique
.AND.	Intersection logique
.OR.	Union logique
.EQV. et .NEQV.	Equivalence et non équivalence logiques

V. Fonctions Mathématiques

Les fonctions intrinsèques les plus utilisées en Fortran sont :

ABS(X)	Valeur absolue d'un nombre entier, réel ou complexe
ASIN(X)	Arc sinus de x
ACOS(X)	Arc cosinus de x
ATAN(X)	Arc tangente de x dans l'intervalle de $-\pi/2$ a $\pi/2$
ATAN2(Y, X)	ATAN2 arc tangent de y/x dans l'intervalle de $-\pi$ a π
COS(X)	Cosinus d'un nombre réel ou complexe, x
COSH(X)	Cosinus hyperbolique de x.
COT(X)	Cotangent de x.
EXP(X)	Valeur de la fonction exponentielle de x
INT(X)	Conversion d'un entier, réel ou complexe, x en entier : exemple : INT(3.9) donne 3, INT(- 3.9) donne - 3.
LOG(X)	Logarithme népérien d'un nombre réel ou complexe, x. Notez qu'un argument entier posera de problème avec Fortran.
LOG10(X)	Logarithme népérien à base de 10 de x.
MAX(X1, X2)	Maximum de deux nombres entiers ou réels
MIN(X1, X2)	Minimum de deux nombres entiers ou réels
MOD(K, L)	Reste de la division de K sur L. les arguments doivent être entiers tous les deux ou réels tous les deux.
NINT(X)	Plus proche de x, e.g. NINT(3.9) donne 4, tandis que NINT(-3.9) donne -4.
REAL(X)	La fonction REAL convertis un entier, réel ou complexe x en réel, e.g. REAL(2)/4 donne 0.5, tandis que REAL(2/4) donne 0.0.
SIN(X)	Sinus d'un nombre réel ou complexe, x
SINH(X)	Sinus hyperbolique de X.
SQRT(X)	Racine carrée d'un nombre réel ou complexe x.
TAN(X)	Tangente de x.
TANH(X)	Tangente hyperbolique de x.

Université Djilali BOUNAAMA Khemis Miliana	**Année Universitaire 2018/2019 (2ème Semestre)**
Faculté des Sciences et de la Technologie	**Matière : Informatique 2 (Cours)**
Département : Sciences de la Matière	**Chapitre n°4**
1ère Année ST-SM	**Structures de Contrôle**

Structures de Contrôle

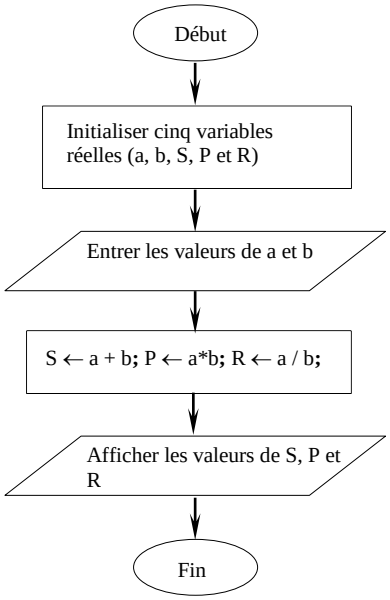
Une structure de contrôle sert à contrôler le déroulement d'un traitement qui peut s'exécuter de différentes manières, à savoir, séquentiellement, alternativement ou répétitivement.

I. Traitement séquentiel

C'est une suite d'instruction qui s'exécute l'une à la suite de l'autre.

Algorithme	Programme en langage Fortran
Début Instruction1; Instruction2; Instruction3; Instruction4; …… ; instructionN; **Fin.**	Instruction1 Instruction2 Instruction3 Instruction4 …… instructionN **end**

Exemple : Addition, produit et rapport de deux nombre réels a et b

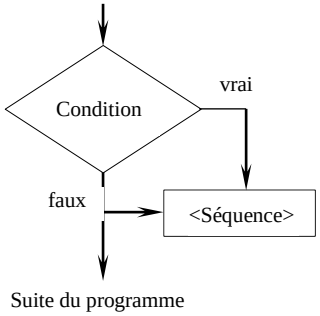
Algorithme	Organigramme	Programme-4-1 Fortran
Algorithme arithmétique; a, b, S, P, R : réels; **Début** **Lire** (a,b); S ← a + b; P ← a * b; R ← a / b; **Ecrire** (S, P, R); **Fin.**		**Program** arithmetic **real** a, b, S, P, R **read**(*,*) a, b S = a + b P = a * b R = a / b **write**(*,*) S, P, R **end** **Remarque :** les écritures suivantes sont équivalentes **:** **read**(*,*) a, b **et** **read***, a, b **write**(*,*) S, P, R **et** **print***, S, P, R

II. Traitement alternatif

On utilise le traitement alternatif lorsqu'on a le choix d'exécuter telle séquence d'instructions si une condition donnée est vraie, et une autre séquence, si la condition est fausse. Le bloc peut être composé d'une ou plusieurs instructions.

On distingue deux types d'alternatif, l'alternatif simple et l'alternatif composé.

II.1. Alternatif simple : dans ce cas si la une condition est vérifiée alors une séquence d'instructions est exécuter sinon ne rien faire.

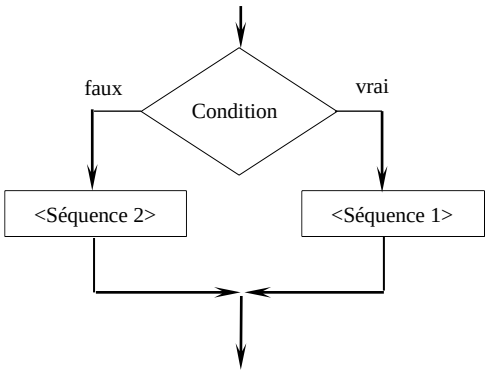
Syntaxe Algorithmique	Organigramme	Fortran
Si (condition) est vrai **alors** <séquence> ;		**IF** (condition) **THEN** <séquence>

Exemple

Algorithmie : **Si** (a>b) **alors** c=a ; **FORTRAN** : **if** (a>b) **then** c=a **ou if** (a.GT.) **then** c=a

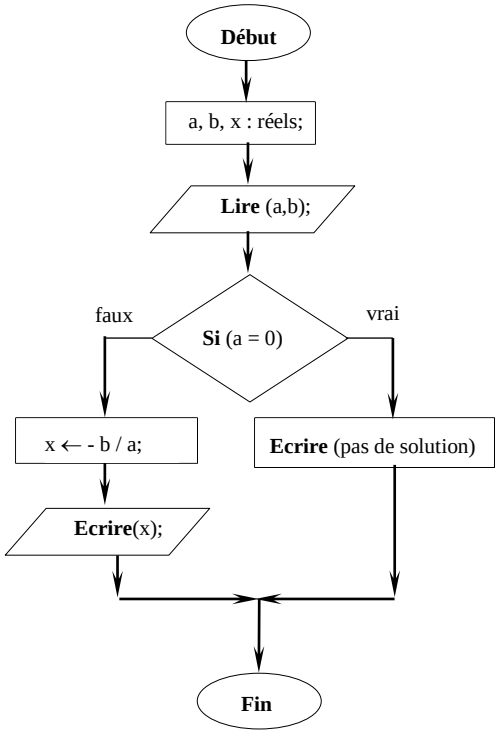
II.2. Alternatif composée : dans ce cas si la une condition est vérifiée alors une séquence d'instructions est exécuter sinon une autre séquence d'instructions qui est exécutée.

Syntaxe Algorithmique	Organigramme	Fortran

Si (condition) est vrai **alors**
<séquence 1>
Sinon
<séquence 2>
		**IF** (condition) **THEN**
		<séquence 1>
		ELSE
		<séquence 2>
		END IF

Exemple : Résolution de l'équation du premier ordre ax+b=0

Syntaxe Algorithmique	Organigramme	Programme-4-2 Fortran

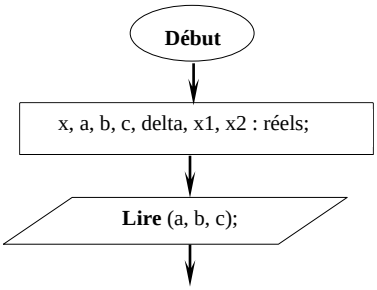
Algorithme equation1;
a, b, x : réels;
 	
Début
Lire (a,b);
Si (a=0) **alors**
Ecrire (l'équation n'a pas de solution);
Sinon
x ← -b/a;
Ecrire(x);
Fin Si;
Fin.
		**Program** equation1
		real a , b, x
		read*,a
		read*,b
		if (a= =0) **then**
		print*,"l'equation n'a pas de solutions"
		else
		x = -b / a
		write(*,*) x
		endif
		end

On peut imbriquer plusieurs alternatifs entre elles de plusieurs manières.

Syntaxe Algorithmique	**Syntaxe Fortran**

Si (condition 1) est vrai **alors**
<séquence 1>
Sinon Si (condition 2) est vrai **alors**
<séquence 2>
Sinon Si (condition 3) est vrai **alors**
<séquence 3>
……..
Sinon Si (condition N) est vrai **alors**
<séquence N>
Sinon
<séquence N>
	if (condition 1) **then**
	<séquence 1>
	else
	if(condition 2) **then**
	<sequence 2>
	else
	if(condition 3)**then**
	<sequence 3)
	…….
	else
	if(condition N)**then**
	<sequence N)
	else
	<sequence N)
	endif
	endif
	endif
	……
	……
	endif

Exemple : Résolution de l'équation du deuxième ordre $ax^2 + bx + c = 0$ (Voir TP).

Syntaxe Algorithmique	**Organigramme**	**Programme-4-3 Fortran**
Algorithme equation2; x, a, b, c, delta, x1, x2 : réels; **Début** **Lire** (a, b, c); **Si** (. . . . **!----------** Ce problème est proposé en travaux pratiques (TP algorithmique), il convient à l'étudiant d'écrire l'**algorithme** adéquat. **!----------**	 . . . . **!----------** Ce problème est proposé en travaux pratiques (TP algorithmique), il convient à l'étudiant de proposer **l'organigramme** adéquat. **!---------- **	**Program** equation2 **real** x, a, b, c, delta, x1, x2 **print***, "entrer la valeur de a" **read***, a **print***, "entrer la valeur de b" **read***,b **print***, "entrer la valeur de c" **read***,c . . . . **!----------** Ce problème est proposé en travaux pratiques (TP algorithmique), il convient à l'étudiant d'écrire le **programme Fortran** adéquat. **!---------- **

II.3. Alternatif multiple (Action à choix multiples) la situation est similaire à celle de **IF**. Elle permet la sélection entre un nombre de cas ou situations, basée sur un sélecteur. Dans ce cas c'est plus convenable que **IF**.

Syntaxe Algorithmique	Syntaxe Fortran
Suivant (expr) **faire** **expr1** : début action1 **fin ;** **expr1** : début action1 **fin ;** **expr1** : début action1 **fin ;** **……** **expr1** : début action1 **fin ;** **Sinon** (expr par défaut) **fin ;** **Fin suivant**	**SELECT CASE** (expr) **CASE** (selecteur1) <séquence 1> **CASE** (selecteur2) <séquence 2> **CASE DEFAULT** <sequence N> **END SELECT**

Où **expr** doit être entier, caractère ou logique.si expr évalue un sélecteur particulier , la séquence correspondant est exécutée, autrement le cas par défaut (**case default**) est sélectionné. **CASE DEFAULT** est optionnel, mais il ne peut être que seul. I ne peut être la dernière clause de l'action **CASE**.

must be integer, character or logical. If it evaluates to a particular selector, that block is executed, otherwise CASE DEFAULT is selected. CASE DEFAULT is optional, but there may be only one. It does not necessarily have to be the last clause of the CASE construct.

The general form of the selector is a list of non-overlapping values and ranges, of the same type as expr, enclosed in parentheses, e.g.

CASE('a':'h', 'i':'n', 'o':'z', '_')

Note that the colon may be used to specify a range of values. If the upper bound of a range is absent, the CASE is selected if expr evaluates to a value that is greater than or equal to the lower bound, and vice versa.

Parts of the CASE construct may be named in the same way as the IF construct.

Exemple : Introduire une lettre au clavier et dire s'il s'agit d'une voyelle ou d'une consonne.

Algorithmique	Organigramme	Programme-4-4 Fortran

Algorithme voyelle;
CH : caractère;
 	
Début
 	
Lire (CH);
		CHARACTER CH
		READ*, CH
		IF (CH >= 'A' .and. CH <= 'Z'.or. CH >= 'a' .and.
		% CH <= 'z') **THEN SELECT CASE** (CH)
		CASE ('A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o', 'u')
		PRINT*, 'Voyelle'
		CASE DEFAULT
		PRINT*, 'Consonne'
		END SELECT
		ELSE
		PRINT*, 'Autre chose'
		END IF
		END

III. Traitement répétitif (Programmation en boucle)

Ce type est utilisé lorsqu'on a à traiter un problème d'une manière répétitive plusieurs fois successivement. Si par exemple on a affaire à calculer la somme d'une suite arithmétique de raison r=1, i.e., U=1+2+3+4+5+6+…+i+i+1+ …+n-1+n, dans ce cas l'opération à répéter plusieurs fois est de rajouter l'unité (1) à la somme jusqu'à ce qu'une condition bien déterminée soit réalisée, e.g., jusqu'à attendre le terme d'ordre 100.

L'opération à répéter se fait suivant une boucle.

Les boucles, en **Fortran** se présentent en trois formes principales :
La boucle **"Pour"**, en Fortran **"do simple"**
La boucle " **do conditionnelle**"
La boucle "**do-while**".

III.1. La boucle "Pour", en Fortran "do-simple"

Cette boucle est introduite comme suit :

En Algorithmique :

pour i allant de valeur initiale **à** valeur finale **faire**;
<sequence>
Fin pour;

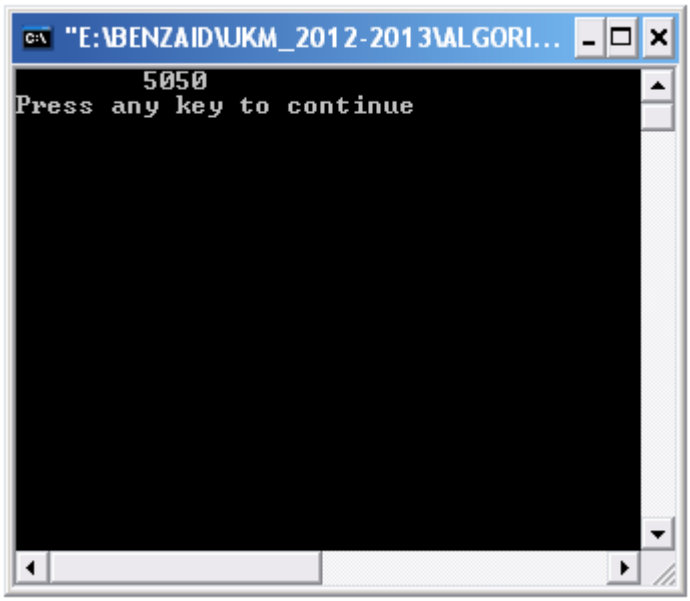
En Fortran :

do i=valeur initiale, valeur finale
<sequence>
end do

i, valeur initiale et valeur finale sont les paramètres de la boucle. **valeur initial** désigne la valeur de début de la boucle, et **valeur finale** désigne la valeur d'arrêt de la boucle. **i**, qui ne peut être de type réel, prend la valeur initiale et la <sequence> est exécutée jusqu'à la valeur **valeur finale**. La boucle "**do-simple**" est donc exécutée (**valeur finale - valeur initiale + 1**) fois. Si **valeur initial** est supérieure à **valeur finale**, la boucle n'est jamais exécutée.

Exemple : Somme de la suite jusqu'au terme n=100, i.e., U=1+2+3+4+5+6+…+98+99+100.

Syntaxe Algorithmique	Organigramme	Programme-4-5 Fortran
Algorithme suite; i, u : entiers; **Début** u ← 0 ; **pour** i allant de 1 **à** 100 faire; u ← u+i; **Fin pour**; **Ecrire**(valeur de u); **Fin.**		**Program** suite **integer** i, u u=0 **do** i=1,100 u=u+i **enddo** **print***,u **end** **Remarque :** la troisième ligne du programme (u=0) peut être omise. Dans le cas où u n'est pas initialisée elle est prise **nulle** par défaut.



La variation du compteur i de la boucle **"do-simple"** précédente est prise par défaut avec un **pas unité**, en ce sens que i est incrémenté automatiquement d'une unité.

Dans certain cas on a besoin d'une variation différente de l'unité. Cette situation est prise en charge par le Fortran comme suit :

En Algorithmique :

pour i allant de valeur initiale **à** valeur finale avec **pas** **faire**;
    <sequence>
Fin pour;

En Fortran :

do i=valeur initial, valeur finale, pas
    <sequence>
end do

Le paramètre **pas** est entier (il peut être positif ou négatif) avec une valeur par défaut qui est égale à l'unité. La valeur initiale de i étant valeur initiale, le pas (entier) est additionné jusqu'à ce que i atteigne sa valeur finale (c'est une condition de sortie de la boucle).

Exemple 1 : Somme de la suite jusqu'au terme n=100, i.e., U=0+2+4+6+8+10+…+96+98+100.

Syntaxe Algorithmique	Organigramme	Programme-4-6 Fortran
Algorithme suite; i, u : entiers; **Début** u ← 0 ; **pour** i allant de 0 **à** 100 avec pas **faire**; u ← u+i; **Fin pour**; **Ecrire**(valeur de u); **Fin.**		**Program** suite **integer** i, u u=0 **do** i=0,100,2 u=u+i **enddo** **print***,u **end** **Remarque :** la troisième ligne du programme (u=0) peut être omise. Dans le cas où u n'est pas initialisée elle est prise **nulle** par défaut.

Exemple 2 : Somme de la suite jusqu'au terme n=0, i.e., U= - 1000 - 98 - 96 …… -6 -4 -2 0.

III.2. La boucle "do-conditionnel exit"

Cette boucle est introduite comme suit :

do
  if (expression logique) exit
  <sequence>
end do

Exemple : Somme de la suite jusqu'au terme n=100, i.e., U=0+2+4+6+8+10+…+96+98+100.

Syntaxe Algorithmique	Organigramme	Programme-4-7 Fortran

III.3. La boucle "tant que … faire", en Fortran "do while"

Algorithmique :

Tant que (condition) **faire**;
<sequence>
Fin tant que;

Fortran :

do while (expression logique)
<sequence>
end do

Exemple : Somme de la suite jusqu'au terme n=100, i.e., U=1+2+3+4+5+6+…+98+99+100.

Syntaxe Algorithmique	Organigramme	Programme-4-8 Fortran
i, S : **entiers** i=0 S=0 Tant que (i<=100) faire S ← S+i; i ← i+1; **Fin tant que**; **Fin**.		**integer** i, S i=0 S=0 **do while** (i<=100) S=S+i i=i+1 **end do** **print** *, S **end**

Université Djilali BOUNAAMA Khemis Miliana	**Année Universitaire 2018/2019 (2ème Semestre)**
Faculté des Sciences et de la Technologie	**Matière : Informatique 2 (Cours)**
Département : Sciences de la Matière	**Chapitre n°5**
1ère Année ST-SM	**Opérations Mathématiques en Algorithmique**

Opérations Mathématiques en Algorithmique

Une structure de contrôle sert à contrôler le déroulement d'un traitement qui peut s'exécuter de différentes manières, à savoir, séquentiellement, alternativement ou répétitivement.

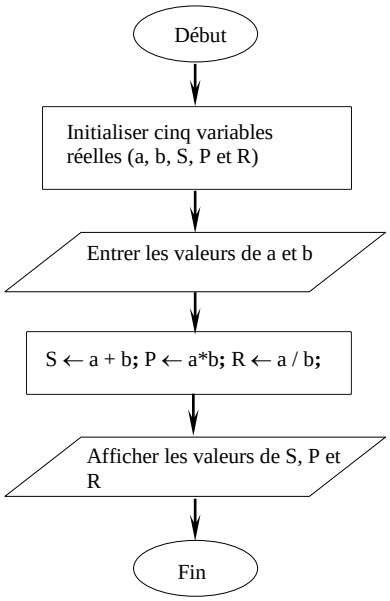
Opérateurs	Priorité	Signification	Exemple
**	1	Exponentiation	2 ** 4 (=24)
*	2	Multiplication	2 * A
/	3	Division	B / DELTA
+	3	Addition	A + 6.9
-		Soustraction	A - 6.9

I. Traitement séquentiel

C'est une suite d'instruction qui s'exécute l'une à la suite de l'autre.

Algorithme	Programme en langage Fortran
Début Instruction1; Instruction2; Instruction3; …… ; instructionN; **Fin.**	**program** Instruction1 Instruction2 Instruction3 …… instructionN **end**

Exemple : Addition, produit et rapport de deux nombre réels a et b

Algorithme	Organigramme	Programme-4-1 Fortran
Algorithme arithmétique; a, b, S, P, R : réels; **Début** **Lire** (a,b); S ← a + b; P ← a * b; R ← a / b; **Ecrire** (S, P, R); **Fin.**		**Program** arithmetic **real** a, b, S, P, R **read**(*,*) a, b S = a + b P = a * b R = a / b **write**(*,*) S, P, R **end** **Remarque :** les écritures suivantes sont équivalentes : **read**(*,*) a, b **et** **read***, a, b **write**(*,*) S, P, R **et** **print***, S, P, R

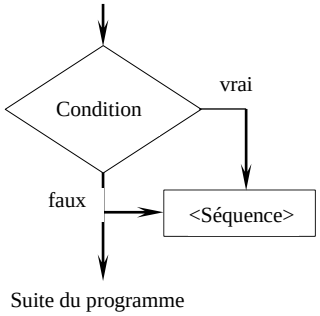
II. Traitement alternatif

On utilise le traitement alternatif lorsqu'on a le choix d'exécuter telle séquence d'instructions si une condition donnée est vraie, et une autre séquence, si la condition est fausse. Le bloc peut être composé d'une ou plusieurs instructions.

On distingue deux types d'alternatif, l'alternatif simple et l'alternatif composé.

II.1. Alternatif simple : si la condition est vérifiée alors une séquence d'instructions est exécutée sinon ne rien faire.

Syntaxe Algorithmique	Organigramme	Fortran

Si (condition) est vrai **alors** <séquence> ;		**IF** (condition) **THEN** <séquence>

Université Djilali BOUNAAMA Khemis Miliana	**Année Universitaire 2018/2019 (2ème Semestre)**
Faculté des Sciences et de la Technologie	**Matière : Informatique 2 (Cours)**
Département : Sciences de la Matière	**Chapitre n°6 (première partie)**
1ère Année ST-SM	**Tableaux à une Dimension (Vecteurs)**

Tableaux à une Dimension (Vecteurs)

Un tableau à une dimension, ou vecteur, est un ensemble de données toutes de même type. Les tableaux à une dimension sont utilisés pour le tri des vecteurs et des listes.
Exemples : Tableau d'entiers, tableau de réels, tableau de caractères, tableau de type énumérés.

Exemple 1 : Le tableau ci-dessous est un tableau d'entiers qui comprend cinq (5) éléments, ou nombre, de type entier.

1	2	3	4	5
-20	33	-5	2	0

i=1, 2, 3, 4, 5 s'appelle **indice** du tableau.
-20, 33, -5, 2, 0 : s'appellent **valeurs** des éléments du tableau.

Exemple 2 : Le tableau Tab1 est un tableau de réels comprend cinq (5) éléments, ou nombre, de type réel.

Indice :	→	***1***	***2***	***3***	***4***	***5***
Nom :	**Tab1**	1.5	-2.0	-5	2	0

Contenu du tableau

Exemple 3 : Le tableau Tab2 est un tableau de chaînes de caractères comprend cinq (5) éléments de type chaîne de caractères.

Indice :	→	***1***	***2***	***3***	***4***	***5***
Nom :	**Tab2**	ST	SM	MI	GC	GP

Contenu du tableau

I. Déclaration d'un tableau à une dimension

Tout tableau à une dimension doit être déclaré par son nom (identificateur) et par le nombre d'élément qu'il peut contenir et par le type de ses éléments.

Syntaxe Algorithmique	Fortran
1. Variable Tab1 : Tableau[10] d'entiers; **2. Variable Tab2** : Tableau[1,10] de réels; **3. Variable Tab3** : Tableau[0,10] de caractères;	**1. integer**, **dimension**(10) **::** Tab1 **2. real**, **dimension**(1:10) **::** Tab2 **3. character**, **dimension**(0:10) **::** Tab3

La première ligne permet de déclarer une variable de type tableau (ou liste) avec 10 éléments entiers qui peuvent être écrits Tab1(1), Tab1(2), …, Tab1(10). Le nombre d'éléments dans un tableau est appelé taille du tableau (dans le cas de **Tab1** la taille est égale à 10.
L'indice doit être une expression entière, sa valeur doit être comprise dans l'intervalle défini dans la déclaration du tableau. De ce fait, Tab(i+1) est une référence valable pour Tab1 tant que (i+1) est compris dans l'interval [1,10]. La valeur la plus petite de l'indice est par défaut égale à 1. Toutefois on peut utiliser n'importe quelle valeur. Par exemple la ligne trois ci-dessus, i.e., **character**, **dimension**(0:10) **::** Tab3, la valeur la plus petite de l'indice est zéro "0" et permet de déclarer le tableau Tab3 qui contient 11 éléments, de Tab(0) à Tab(10). La valeur supérieure de l'indice doit être spécifiée.

II. Accès aux éléments d'un tableau

Chaque élément d'un tableau à une dimension est identifié par la position à laquelle il se trouve dans le tableau.
L'indice de la position du 1er élément est égal à 1, l'indice de la position du 2ème élément est égal à 2, et ainsi de suite jusqu'au dernier élément dont l'indice de la position est égal à la valeur de la dimension. De ce fait il ne faut jamais confondre l'indice de la position au contenu qu'il peut y avoir à cette position.

Exemple :
Le tableau suivant **integer**, **dimension**(5) **::** Tab1;

1	2	3	4	5
-20	33	-5	2	0

Le contenu du tableau à la position 1 est -20, le contenu à la position 3 est -5, et ainsi de suite.

L'accès aux éléments d'un tableau se fait comme suit :

Syntaxe Algorithmique	Fortran
Tab[indice] **;**	Tab(indice)

Exemples :

Syntaxe Algorithmique	Fortran
1. Tab[2] ← 7 ; {affectation d'une valeur à la position 2 du tableau}. **2.** lire tab[5]) ; {Initialisation de la position 5 avec une valeur saisie au clavier} **3.** afficher (tab[3]); {Affichage du contenu de la position 3}	**1. Tab**(2) = 7 **2. read***,Tab(5) **3. print***,Tab(3)

III. Opération sur les tableaux à une dimension (exemples pratiques)

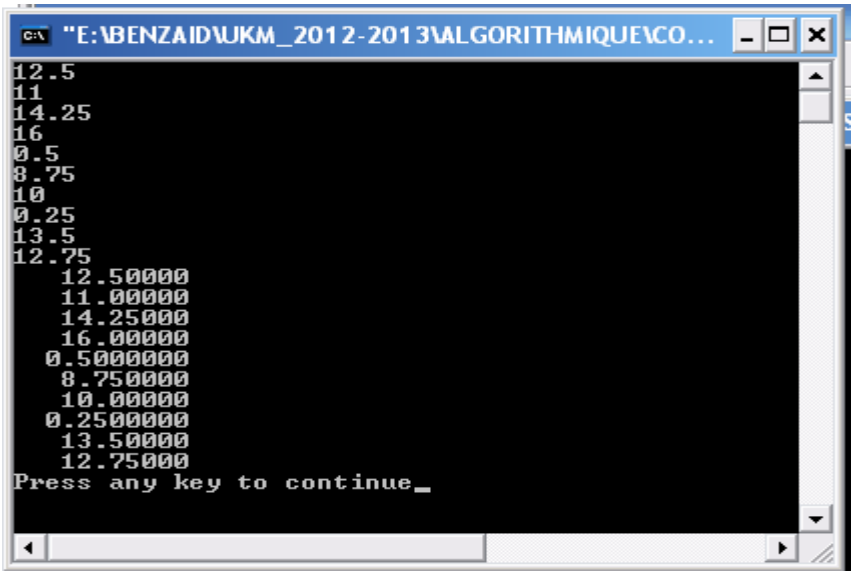
III.1. Saisie et affichage des éléments d'un tableau

Soit à saisir les notes de l'examen de la matière algorithmique d'un groupe composé de 10 étudiants.

Syntaxe Algorithmique	Organigramme	Programme-6-1 Fortran

algorithme saisi;
Note : Tableau[10] de réels;
 	
Début
 	
! Saisi des éléments du tableau
 	
pour i allant de 1 **à** 10 **faire**;
  Lire (Note[i]);
Fin pour;
 	
! Affichage des éléments du tableau
 	
pour i allant de 1 **à** 10 **faire**;
  Ecrire (Note[i]);
Fin pour;
 	
Fin.
		program saisi
		real, **dimension**(10) **::** Note
		
		! Saisi des éléments du tableau
		
		do i = 1 , 10
		read*, Note(i)
		enddo
		
		! Affichage des éléments du tableau
		
		do i = 1 , 10
		print*, Note(i)
		enddo
		
		end

Exemple d'exécution du programme FORTRAN :



Dans ce programme on utilise deux boucles : une pour saisir les éléments du tableau, et l'autre pour les afficher.
Dans les deux cas le parcours du tableau se fait du début à la fin. Ceci n'est pas obligatoire, mais plus facile.

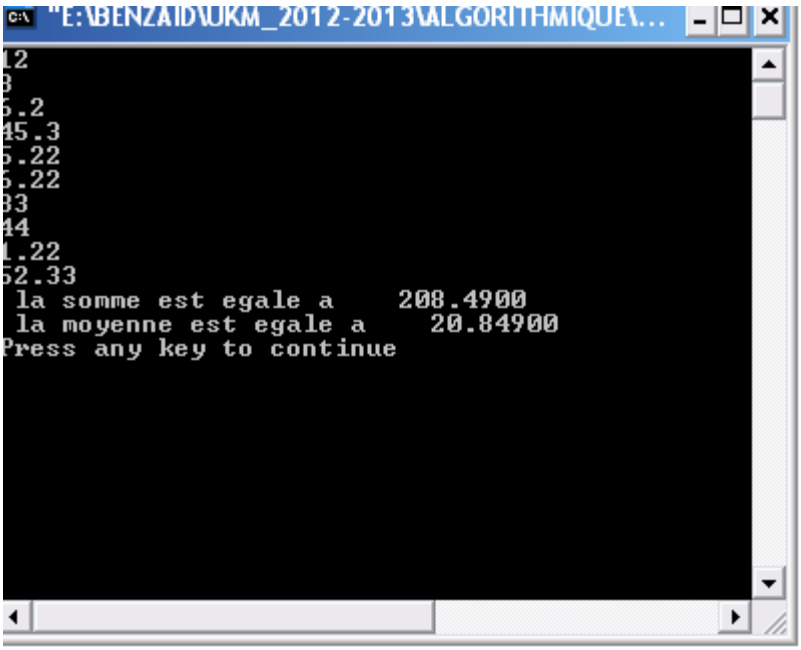
III.2. Calcul de la somme et de la moyenne

Soit à saisir les valeurs des éléments d'un tableau à une dimension et calculer la somme et la moyenne de ces éléments.

Syntaxe Algorithmique	Organigramme	Programme-6-2 Fortran

algorithme moyenne;
d=10 : **entier constant** ;
x : Tableau[d] de réels;
 	
Début
 	
! Saisi des éléments du tableau
 	
pour i allant de 1 **à** 10 **faire**;
  Lire (x[i]);
Fin pour;
 	
! Somme des éléments du tableau
 	
pour i allant de 1 **à** 10 **faire**;
  S ← S +x[i];
Fin pour;
 	
M ← S/d;
 	
! Affichage de la Somme et de la moyenne
Ecrire (" la somme est égale à", S);
Ecrire (" la moyenne est égale à ", M);
 	
Fin.
		program moyenne
		integer, **parameter ::** d=10
		real, **dimension**(d) **::** x
		real S, M
		
		! Saisi des éléments du tableau
		
		do i = 1 , d
		read*, x(i)
		enddo
		
		! Somme des éléments du tableau
		
		do i = 1 , d
		S = S + x(i)
		Enddo
		
		M=S/d
		
		! Affichage de la Somme et de la moyenne
		
		print*, "la somme est egale a ", S
		print*, "la moyenne est egale a ", M
		
		end

Exemple d'exécution du programme FORTRAN :



Problème à résoudre :

Soit un vecteur qui contient l'ensemble des notes des 158 étudiants, on veut calculer la note moyenne et déterminer le nombre d'étudiants qui ont une note supérieure à la note moyenne

1. Ecrire un algorithme
2. Faire une représentation par un organigramme
3. Ecrire un programme en langage FORTRAN

Remarque : Le travail est à remettre dans une semaine à compter d'aujourd'hui.

Université Djilali BOUNAAMA Khemis Miliana	**Année Universitaire 2018/2019 (2ème Semestre)**
Faculté des Sciences et de la Technologie	**Matière : Informatique 2 (Cours)**
Département : Sciences de la Matière	**Chapitre n°6 (deuxième partie)**
1ère Année ST-SM	**Tableaux à deux Dimensions (Matrices)**

Tableaux à deux Dimensions (Matrices)

Un tableau à une dimension, ou vecteur, est un ensemble de données toutes de même type. Les tableaux à une dimension sont utilisés pour le tri des vecteurs et des listes.
Exemples : Tableau d'entiers, tableau de réels, tableau de caractères, tableau de type énumérés.
Exemple 1 : Le tableau suivant est constitué de trois lignes et 5 colonnes, et est composé de 3×5=15 éléments, ou nombre, de type entier.

	1	2	3	4	5
1	-20	33	-5	2	0
2	2	-3	0	25	126
3	23	36	9	4	1

i=1, 2, 3 s'appelle **indice** de la ligne du tableau.
j=1, 2, 3, 4, 5 s'appelle **indice** de la colonne du tableau.
-20, 33, -5, 2, 0, … 9, 4, 1 : s'appellent **valeurs** des éléments du tableau.

Exemple 2 : Le tableau suivant est constitué de trois lignes et 5 colonnes, et est composé de 3×5=15 éléments, ou nombre, de type réel.

Indices colonnes(→) et lignes (↓)

↓ \ →	**1**	**2**	**3**	**4**	**5**
1	1.23	2.25	-5	5.33	0
2	2.33	-3.6	-5.0	2.66	2.6
3	23.5	36	9.2	4	1

Un élément du tableau est repéré par les deux indices. Par exemple la valeur de l'élément à la position **(3,2)** est **-5.0**.
Exemple 3 : Le tableau suivant est un tableau de chaînes de caractères comprenant quinze (15) éléments de type chaîne de caractères.

Indices colonnes(→) et lignes (↓)

↓ \ →	**1**	**2**	**3**	**4**	**5**
1	a	b	c	d	e
2	ab	abc	abcd	bc	bcd
3	un	une	la	le	les

I. Déclaration d'un tableau à deux dimensions

Tout tableau à deux dimensions doit être déclaré par son nom (identificateur) et par sa dimansion (dans ce cas deux dimension) et par le nombre d'élément qu'il peut contenir et par le type de ses éléments. En FORTRAN il existe plusieures manières de déclarer le tableau.

Syntaxe Algorithmique	**Fortran**
1. Variable Tab1 : Tableau[1:10,2:5] d'entiers; **2. Variable Tab2** : Tableau[1:2,1:3] de réels; **3. Variable Tab3** : Tableau[1:4,1:4] d'entiers;	**1. integer**, **dimension**(1:10,2:5) **::** Tab1 **2. real**, **dimension**(2,3) **::** Tab2 **3. integer** Tab3(4,4) **4. real, dimension**(4,6) **::** A, B(2,3), C(5,3) ! Seul A qui est un tableau (1:4,1:6) **5. integer** I(5,4), K(3,3), L(2,4) ! tous les tableaux sont différents

Les mêmes remarques que celles faites aux tableaux à une dimension peuvent être avancées.

II. Accès aux éléments d'un tableau à deux dimensions

Chaque élément d'un tableau à deux dimensions est identifié par la position à laquelle il se trouve dans le tableau : numéro de ligne et numéro de colonne.

L'indice de la 1ère ligne est égal à 1, l'indice de la 2ème ligne est égal à 2, et ainsi de suite jusqu'à la dernière ligne dont l'indice est égal à d1. Et l' L'indice de la 1ère colonne est égal à 1, l'indice de la 2ème colonne est égal à 2, et ainsi de suite jusqu'à la dernière colonne dont l'indice est égal à d2. d1 et d2 sont les dimensions du tableau.

Exemple :

Le tableau suivant **integer**, **dimension**(3**,**5) **::** Tab1;

	1	2	3	4	5
1	-20	33	-5	2	0
2	2	-3	0	25	126
3	23	36	9	4	1

Le contenu du tableau à la position (1**,**3) est **-5**, le contenu à la position (2**,**2) est **-3**, et ainsi de suite.

L'accès aux éléments d'un tableau se fait comme suit :

L'accès aux éléments d'un tableau se fait comme suit :

Syntaxe Algorithmique	Fortran
Tab[i**,**j] **;**	Tab(i**,**j) tels que **i** et **j** sont les indices de ligne et de colonne respectivement.

Exemples :

Syntaxe Algorithmique	Fortran
1. Tab[1,2] ← 21.5 ; {affectation d'une valeur à la position (1,2) du tableau}. **2.** lire tab[3,5]) ; {Initialisation de la position (3,5) avec une valeur saisie au clavier} **3.** afficher (tab[6,3]); {Affichage du contenu de la position (6,3)}	**1. Tab**(1,2) = 21.5 **2. read***, Tab(3,5) **3. print***, Tab(6,3)

III. Opération su les tableaux à deux dimensions (exemples pratiques)

III.1. Saisie et affichage des éléments d'un tableau à deux dimensions

Soit à saisir les notes de l'examen de la matière algorithmique de 5 groupes composé de 20 étudiants.

Syntaxe Algorithmique	Organigramme	Programme-6-3 Fortran
algorithme saisi; **Note** : Tableau[1**:**30,1**:**5] de réels; **Début** ! Saisi des éléments du tableau **pour** j allant de 1 **à** 5 **faire**; **pour** i allant de 1 **à** 30 **faire**; **Lire** (Note[i,j]); **fin pour**; **fin pour**; ! Affichage des éléments du tableau **pour** j allant de 1 **à** 5 **faire**; **pour** i allant de 1 **à** 30 **faire**; **Ecrire** (Note[i,j]); **fin pour**; **fin pour**; **Fin.**		**program** saisi **real**, **dimension**(1**:**30,1**:**5) **::** Note ! Saisi des éléments du tableau **do** j = 1 , 5 **do** i = 1 , 30 **read***, Note(i,j) **enddo** **enddo** ! Affichage des éléments du tableau **do** j = 1 , 5 **do** i = 1 , 30 **print***, Note(i,j) **enddo** **enddo** **end**

III.2. Calcul de la somme de deux matrices

Soit à calculer la matrice S(4×4), somme des deux matrices M1(4×4) et M2(4×4) à valeurs réelles.

Syntaxe Algorithmique	Organigramme	Programme-6-4 Fortran
algorithme somme; **M1, M2, S** : Tableau[1**:**4,1**:**4] de réels; **Début** ! Saisi des éléments des deux matrices **pour** i allant de 1 **à** 4 **faire**; **pour** j allant de 1 **à** 4 **faire**; **Lire** (M1[i,j]); **Lire** (M2[i,j]); **fin pour**; **fin pour**; ! Somme des deux matrices **pour** i allant de 1 **à** 4 **faire**; **pour** j allant de 1 **à** 4 **faire**; S[i,j] ← M1[i,j]+M2[i,j] **fin pour**; **fin pour**; ! Affichage de la Somme et de la moyenne **pour** i allant de 1 **à** 4 **faire**; **pour** j allant de 1 **à** 4 **faire**; **Ecrire** (S[i,j]); **fin pour**; **fin pour**; **Fin.**		**program** somme **real, dimension**(1**:**4,1**:**4) **::** M1, M2, S ! Saisi des éléments des deux matrices **do** i = 1 , 4 **do** j = 1 , 4 **read***, M1(i,j) **read***, M2(i,j) **enddo** **enddo** ! Somme des deux matrices **do** i = 1 , 4 **do** j = 1 , 4 S(i,j)=M1(i,j)+M2(i,j) **enddo** **enddo** ! Affichage de la Somme et de la moyenne **do** i = 1 , 4 **do** j = 1 , 4 **print***, S(i,j) **enddo** **enddo** **end**

Travail à effectuer:

Produit de deux matrices au sens mathématique

Ecrire un programme **FORTRAN** qui permet de calculer le produit de deux matrices.

Université Djilali BOUNAAMA Khemis Miliana	**Année Universitaire 2018/2019 (2ème Semestre)**
Faculté des Sciences et de la Technologie	**Matière : Informatique 2 (Cours)**
Département : Sciences de la Matière	**Chapitre n°7 (deuxième partie)**
1ère Année ST-SM	**Sous Programmes et Modules**

Sous Programmes et Modules

Introduction

Dans la pratique un programme **FORTRAN** est divisé en sous-programme (ou procédures), menant chacun une tâche bien déterminée. Ces sous-programmes sont souvent utilisés par différents programmes "principaux", et , en fait, par différents utilisateurs du même système.
Le programme principal et chaque sous-programme sont analysés séparément par le compilateur. Ils peuvent donc figurer soit daans un même fichier, soit dans des fichiers séparés. Il est préférable de faire un fichier par sous-programme. Au moment de la mise au point de petits programmes, il est plus facile de faire figurer le programme principale et les sous-programmes ensemble, puis de les séparer lorsque la compilation et les éssais d'exécutionont réussi.
Dans le cas où l'on fait figurer dans un même fichier programme tet sous-programme, le programme principal doit figurer entête, suivi des sous-programmes, chacun se terminant par END, pour bien délimiter les blocks.
On distingue quatre types de modules.
PROGRAM : programme principal, tout module dont la 1ère instruction n'est SUBROUTINE, FUNCTION ou BLICK DATA. Ce module peut comporter comme 1ère instruction PROGRAM, mais ce n'est pas obligatoire.
SUBROUTINE : ce module es un sous programme toujours appelé par un CALL à partir d'un autre module de type programme ou sous-programme.
FUNCTION : ce module est simplement appelé par son nom à partir d'un autre module.
BLOCK DATA ce module initialise des variables placées dans COMMON nommé.

Programme Principal
Un programme principal comporte deux parties
Les déclarations concernant les variables.
Les instructions exécutables.
L'instruction PROGRAM nom du programme est facultative.
Le programme doit terminer par END (c'est obligatoire).

Sous-Programmes
Il existe deux types de sous programme (interne et externe)
Un sous-programme interne est contenu dans une autre unité du programme et par conséquent il est comilé avec ce programme alors que le sous-programme externe ne l'est pas, il est en fait comilé séparément.
Sous-programme interne
Il existe deux types de sous-programmes les **FUNCTION** et **SUBROUTINE**
Les FUNCTION
Le **FORTRAN** fournit déjà des fonction dites intrinsèques, comme par exemple sin(), exp(),log() …. On peut construire son propre fonction.
Examinons l'exemple suivant :

```
PROGRAM Factoriel
Integer i
do I = 0 , 7
Print*, I , Fact(i)
enddo

FUNCTION fact(n)
integer fact, n, F
if (n = = 0 .or. n = = 1)then
Fact=1
Else
F=1
do I = 2 , n
F=i*F
enddo
Fact=F
endif
end function
end
```