

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie d'Oran
Mohamed Boudiaf (USTOMB)
Faculté : Physique
Département : Génie Physique

SUPPORT DE COURS EN INFORMATIQUE O2

**Algorithmique et programmation
en Pascal et Fortran :
Cours et exercices corrigés**

Présenté par : Dr. SMAHI Zakaria

2021

INTRODUCTION

Ce polycoié de cours regroupe un certain nombre de notions de base sur l'algorithmique et la programmation en langage Pascal et Fortran. Ce cours d'algorithmique, destiné particulièrement aux étudiants de première année Sciences technologiques (ST).

Dans ce présent polycoie est illustré l'essentiel du cours d'algorithmique et de la programmation couvrant toutes les instructions de base, les structures conditionnelles et itératives et enfin les tableaux à une et à deux dimensions. Ce polycoie est aussi le fruit d'une longue expérience dans le domaine de l'algorithmique et de la programmation constituant ainsi un support de cours pour des étudiants n'ayant aucune connaissance en programmation.

Ce cours est rédigé dans un style simple et riche en termes d'exemples qui sont compréhensibles englobant une série d'exercices avec corrigé couvrant ainsi toutes les sections présentées en cours. En plus, nous avons fait appel à deux principal langages de programmation à savoir le Pascal et le Fortran. Ces deux derniers sont deux langages faciles à manipuler par l'étudiant débutant et possédant chacun une structure très proche de celle du langage naturel et de syntaxe algorithmique.

Pour terminer, la polycoie contient environ des séries d'exercices souvent conçus comme une application du cours. La solution proposée à chaque exercice n'est pas unique et dans plusieurs cas elle n'est optimale car on a toujours privilégié l'apport pédagogique et la simplicité.

La première série d'exercices présente un ensemble d'exercices de base permettant à l'étudiant de se familiariser, tout d'abord avec les différentes instructions de base en algorithmique et en programmation Pascal et Fortran et de traiter aussi quelques problèmes en utilisant les structures conditionnelles. On y trouve ensuite, une deuxième série d'exercices nécessitant l'exploitation des structures itératives (boucles: Pour, Tant que et Répéter) et enfin, les tableaux, à savoir les vecteurs et les matrices. Enfin, quel que soit le cours de programmation utilisé, le seul moyen d'apprendre à l'étudiant à programmer : c'est de le pousser à programmer lui-même.

Table des Matières

1. Qu'est-ce qu'un algorithme ?	1
1.1. Son origine	1
1.2. Son Rôle	1
2. Les étapes de résolution d'un problème	2
3. Langage de programmation	3
4. Représentation d'un algorithme	4
4.1. Le langage Naturel (Exp : Français)	4
4. 2. L'Organigramme: Représentation graphique	4
4. 3. Le pseudo-code ou Langage de Description d'Algorithme (LDA)	5
5. Structure générale d'un Algorithme / Programme	6
6. Notion de Constante	
7. Notion de Variable	7
8. Notion de Type	7
8.1. Le Type Entier	8
8.2. Le Type Réel	9
8.3. Le Type Caractère et Chaîne	10
8.4. Le Type Booléen ou Logique	11
Exercices sur les affectations	13
9. Les Actions / instructions Algorithmiques	14
9.1. L'instruction d'affectation	14
9.2. L'entrée/ la sortie d'information	15
9.2.1. Action de Lecture	15
9.2.2. Action d'Écriture	17
Exercices sur les entrées et sorties	20
9.3. Structures de contrôles	21
9.3.1. Structures de contrôles conditionnelles	21
a) Conditionnelle à un Choix (Test alternative simple)	21
b) Conditionnelle à deux Choix (Test alternative double)	22
c) Conditionnelle à Choix Multiple	23

9.3.2. Structures de contrôle répétitives	25
9.3.2.1. La Structure itérative POUR (For)	25
9.3.2.2. La Structure itérative TANT QUE(While)	26
9.3.2.3. La Structure itérative Répéter (Repeat)	28
9.3.3. Instruction de branchement (SAUT)	29
Exercices sur les structures conditionnelles	31
Exercices sur les BOUCLES	32
9.4. Tableaux et Matrices	33
9.4.1 Tableaux à une dimension (vecteurs)	33
a) Déclaration d'un vecteur	34
b) Initialisation d'un élément d'un vecteur (affectation)	34
c) Lecture et affichage des éléments d'un vecteur	35
d) Quelques opérations sur les vecteurs	36
i. Somme des éléments d'un tableau	36
ii. Recherche du maximum des éléments d'un vecteur	37
9.4.2. Matrices (Tableaux à deux dimensions)	39
a) Déclaration d'une matrice	39
b) Initialisation d'un élément d'une matrice	41
c) Lecture et affichage des éléments d'une matrice	41
d) Quelques opérations sur les matrices	42
i. Somme des éléments d'une matrice	42
ii. Recherche du maximum des éléments d'une matrice	44
Exercices sur les Tableaux	46
Exercices sur les Matrices	47
Liste de références	48
Solution des exercices	49

1. Qu'est-ce qu'un algorithme ?

Définition 1 : Un **algorithme** est une succession d'**instructions** (aussi appelées **commandes**) qui permettent la résolution d'un problème donné.

Définition 2 : Un algorithme est une suite finie de règles à appliquer dans un ordre déterminé à un nombre fini de données pour arriver, en un nombre fini d'étapes, à un certain résultat, et cela indépendamment des données.

1.1. Son origine:

Le terme algorithme vient du nom d'un célèbre mathématicien musulman de la première moitié du IXe siècle : Muhammad ibn Musa al Khuwarizmi né à Khiva (*Xiva* en **ouzbek** est une ville d'**Ouzbekistan**), son ancien nom, **Khwarezm** ou **Khorezm**).

Ses travaux sur les algorithmes, terme dérivé de son nom, permirent d'introduire la méthode de calcul utilisant les chiffres arabes et la notation décimale (*Encarta*).

1.2. Son Rôle:

- Le rôle de l'algorithme est fondamental. En effet, sans algorithme, il n'y aurait pas de programme (qui n'est jamais que sa traduction dans un langage compréhensible par l'ordinateur).
- De plus, les algorithmes sont fondamentaux en un autre sens: ils sont indépendants à la fois de l'ordinateur et des langages de programmation dans lesquels ils sont énoncés et traduits.

« **Ecrire un algorithme** », c'est :

- Analyser et comprendre le problème : étudier les données fournies et les résultats attendus

- Résoudre le problème, c'est trouver les structures de données adaptées ainsi que l'enchaînement des actions à réaliser pour passer des données aux résultats.

Comment exécuter un algorithme sur un ordinateur ?

Il faut traduire cet algorithme à l'aide d'un langage de programmation connu par l'ordinateur.

En Résumé

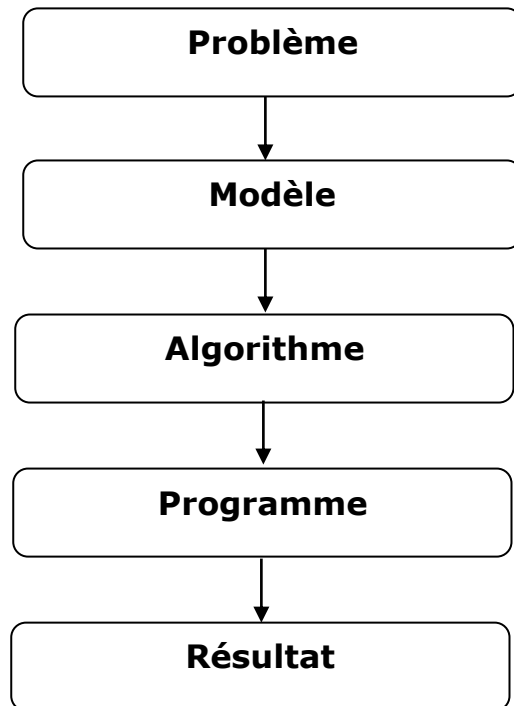
- Un Algorithme consiste à retranscrire un processus logique à l'aide d'un langage naturel.
- Un Algorithme est la description d'un traitement qui consiste à transformer des données, appelées « **entrées** », afin de produire d'autres données appelées « **sorties** ».
- Les entrées et les sorties représentent les variables manipulées par l'algorithme.
- Processus de Principe : Entrées -> Traitement -> Sorties

2. Les étapes de résolution d'un problème

Dans un algorithme, pour résoudre un problème donné, un processus d'analyse et de résolution sera appliqué. Ce processus est constitué des étapes suivantes :

1. Comprendre l'énoncé du problème : identifier et analyser le problème à étudier;
2. Décomposer le problème en plusieurs sous problèmes simples ;
3. Associer à chaque sous problème, une spécification :
 - Les données d'entrées nécessaires
 - Les données résultantes
 - Organiser les étapes (actions) à suivre pour arriver au résultat en partant d'un ensemble de données.
4. Elaboration d'un algorithme.
5. Traduire cet algorithme en langage de programmation

6. Compiler le programme pour qu'il puisse être exécutable pour arriver aux résultats escomptés.



3. Langage de programmation

Par définition, un **langage de programmation** est un ensemble d'instructions et de règles syntaxiques compréhensible par l'ordinateur et permettant de créer des algorithmes. Un **programme** est la traduction d'un algorithme dans le langage de programmation utilisé.

En résumé, un **langage de programmation** permet à un humain d'écrire un code pouvant être analysé par une machine puis transformé en un programme informatique.

Un **programme informatique** est une suite d'opérations prédéterminées pouvant être exécutées par une machine.

Il existe de nombreux langages de programmation : C, C++, Java, Basic, Pascal, Fortran, etc.

Les langages Pascal et Fortran sont utilisés dans ce cours en raison de leurs caractères pédagogiques.

4. Représentation d'un algorithme

4.1. Le langage Naturel (Exp : Français).

C'est une représentation en langage naturel.

Exemple : Comment formuler l'algorithme qui calcule la surface d'un disque ?

Réponse : Pour calculer la surface **S** d'un disque, on prend d'abord connaissance de son rayon **R**. Ensuite, la surface **S** du disque est égale au produit de la constante π par le carré du rayon **R** ($S = \pi * R^2$).


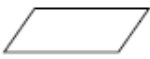






4. 2. L'Organigramme: Représentation graphique

Cette représentation utilise des symboles (carrés, losanges, etc.

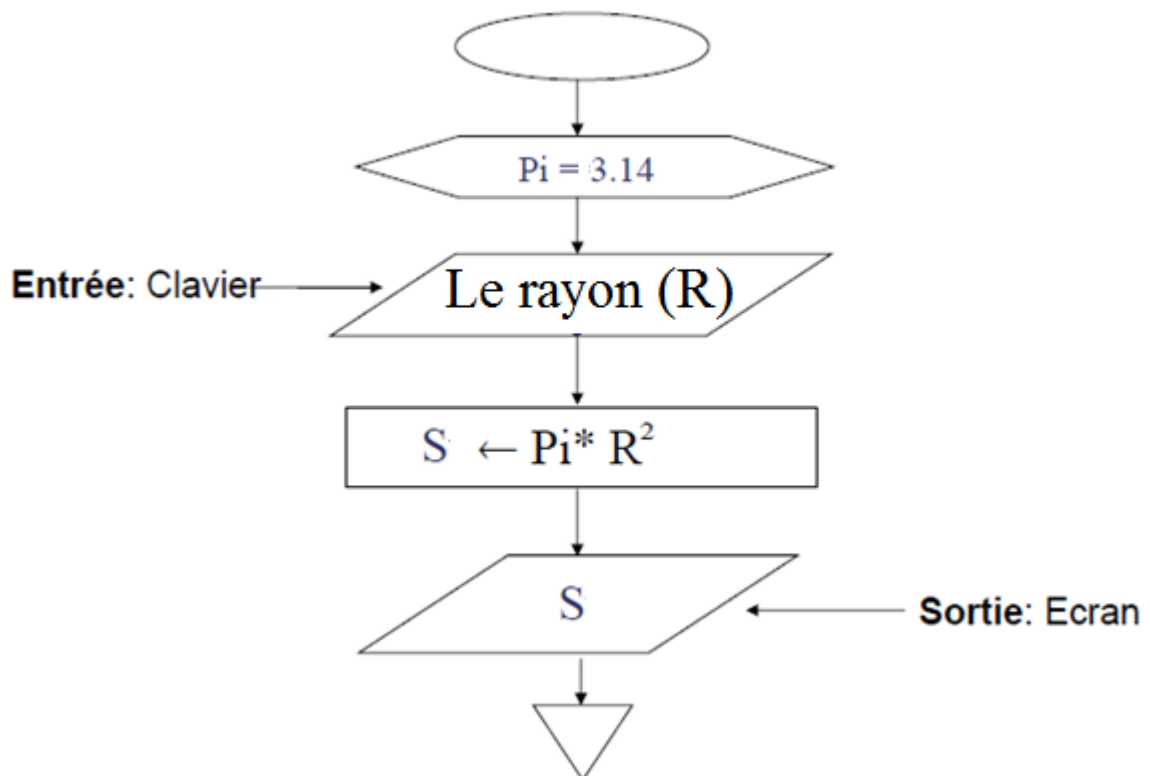
Figure1)

- **Avantage** : offre une vue d'ensemble de l'algorithme
- **Inconvénient** : représentation quasiment abandonnée aujourd'hui

Figure 1. Représentation graphique d'un Organigramme

Symboles	Signification
	L'ellipse : indique le début de l'algorithme
	Le parallélogramme : représente une opération de Lecture ou Ecriture
	Le rectangle : représente le contenu de chaque étape de traitement, il définit l'opération à exécuter.
	La flèche : sert à indiquer l'ordre d'exécution des différentes tâches.
	L'Hexagone : sert à indiquer l'initialisation de certains paramètres.
	Le losange : il représente l'existence d'un choix logique : la proposition logique est indiquée à l'intérieur du losange.
	Le cercle : indique la poursuite d'une étape précédente.
	Le triangle renversé : symbolise la fin de l'algorithme.

Exemple : Calcul de surface d'un disque



4.3. Le pseudo-code ou Langage de Description d'Algorithme (LDA)

C'est une représentation textuelle avec une série de conventions ressemblant à un langage de programmation (sans les problèmes de syntaxe).

- plus pratique pour écrire un algorithme
- représentation largement utilisée.

Exemple précédent devient:

```
Algorithme surf_Disq ;  
  Const Pi = 3.14 : Réel ;  
  Var S, R : Réel ;  
Début  
  Lire(Rayon) ;  
  S ← Pi * R**2 ;  
  Ecrire(S) ;  
Fin.
```

5. Structure générale d'un Algorithme / Programme

En générale, l'algorithme comprend deux parties essentielles:

- Une partie déclarative commençant par un entête comprenant le nom de l'algorithme (nom significatif) et contenant tous les objets impliqués par les différentes actions de l'algorithme (variables, constantes, structure, etc.).
- Une partie réservée aux actions (ou instructions) qui est délimitée par les deux mots-clés **Début** et **Fin**. Cette partie représente le corps de l'algorithme.

<p>Algorithme <i>Nom Algo ;</i> <i>Variable/Var..... ;</i> <i>Constante/Const..... ;</i> <i>Structure/Struct.... ;</i> <i>Type..... ;</i></p>	<p>Partie déclarative</p>
<p>Début <i>Lecture de données/Lire()..... ;</i> <i><Action1> ;</i> <i><Action2> ;</i> <i>..... ;</i> <i><Action N> ;</i> <i>Ecriture des résultats/Ecrire() ;</i> Fin.</p>	<p>Corps de l'algorithme</p>

Dans la partie déclarative, le nom d'un objet qui est définit par un identificateur doit respecter lors de sa définition les règles suivantes :

- Un identificateur doit être unique dans un algorithme
- Un identificateur doit commencer obligatoirement par une lettre.
- Un identificateur ne doit pas dépasser 8 caractères (en termes de taille).
- Un identificateur ne doit pas comporter le caractère « espace » ou de caractères spéciaux autres que le tiret de soulignement (_).

La syntaxe d'un programme PASCAL ou FORTRAN débute par le mot-clé program suivi du nom du programme. Ce nom ne doit plus être réutilisé dans la suite du programme. Ainsi, le programme est structuré de la manière suivante :

Pascal	Fortran
Program NomProg ; - Déclarations des variables Begin {pour débiter le programme} - Lecture des données - Traitement des données - Sortie des résultats End. {pour signaler la fin du programme}	Program NomProg - Déclarations des variables - Lecture des données - Traitement des données - Sortie des résultats End {fin du programme}

6. Notion de Constante

Une constante est une variable qui ne change pas de valeurs lors de l'exécution d'un programme.

Exemple :

La constante $PI = 3.14$

La constante $e=2.7$

7. Notion de Variable

Par définition, on appelle une variable tout emplacement de la mémoire de l'ordinateur dans lequel on stocke une information qui peut être changée. Une variable est donc un espace mémoire qui va contenir des données. Elle est constituée :

- d'un nom ou identificateur qui permet de reconnaître où elle se situe dans la mémoire;
- d'une valeur : le nombre ou plus généralement l'information stockée.
- d'un type : c'est à dire un ensemble contenant toutes les valeurs possibles qu'elle peut prendre.

8. Notion de Type

Toute variable utilisée dans un algorithme possède un domaine de définition. Ce domaine est appelé le Type de la variable.

On distingue deux familles de types :

- les types simples : ce sont des types qui sont supportés et reconnus par la majorité des langages de programmation (Entier, Réel, Caractère, Booléen ou logique).
- Les types composés ou complexes : ce sont des types qui sont construits à partir des types simples qu'il faut les déclarer dans la partie réservée aux types : tableaux, chaînes, enregistrements, etc.

8.1. Le Type Entier

C'est un sous ensemble fini de nombres entiers, dont la taille varie en fonction des performances techniques de la machine et celles du langage de programmation utilisé. A titre d'exemple, si un entier est codé sur deux octets alors son intervalle varie entre -32768 à +32767.

a) Représentation algorithmique :

Var Nomvar : **Entier** ; (Nomvar étant la variable qui sera utilisée dans l'algorithme)

Si on utilise plusieurs variables en même temps, on peut les regrouper en les séparant par une virgule.

Var Nomvar1, Nomvar2,, NomvarN : **Entier** ;

b) Représentation en langage Pascal :

Var Nomvar : **Integer** ;

Var Nomvar1, Nomvar2,, NomvarN : **Integer** ;

c) Représentation en langage Fortran :

Integer Nomvar

Integer Nomvar1, Nomvar2,, NomvarN

Remarque: En langage Fortran, toute variable dont le premier caractère comprend l'une des six lettres (I,J,K,L,M,N) est considérée comme variable de type Entier. Le reste des variables est par défaut de type Réel.

Les opérations arithmétiques de base sur le type Entier :

Opération	Symbole	Exemple
Addition	+	A=5, B=3 ; A+B donne 8
Soustraction	-	A=5, B=3 ; A - B donne 2
Multiplication	*	A=5, B=3 ; A*B donne 15
Division (entière)	Div	A=5, B=3 ; A Div B donne 1
Reste de la division (entière)	Mod	A=5, B=3 ; A Mod B donne 2

8.2. Le Type Réel

Le type Réel comprend aussi un sous ensemble fini de nombres réels dont la taille varie en fonction des performances techniques de la machine et celles du langage de programmation utilisé. A titre d'exemple, si un Entier est codé sur deux octets (2^{16} valeurs) alors le type Réel est codé sur quatre octets (2^{32} valeurs).

a) Représentation algorithmique :

Var Nomvar : **Réel** ;

Si on utilise plusieurs variables en même temps alors sa syntaxe est :

Var Nomvar1, Nomvar2,, NomvarN : **Réel** ;

b) Représentation en langage Pascal :

Var Nomvar : **Real** ;

Var Nomvar1, Nomvar2,, NomvarN : **Real** ;

c) Représentation en langage Fortran :

Real Nomvar

Real Nomvar1, Nomvar2,, NomvarN

Les opérations arithmétiques de base sur le type Réel :

Opération	Symbole	Exemple
Addition	+	A=5, B=3.5 ; A+B donne 8.5
Soustraction	-	A=5, B=3.5 ; A - B donne 1.5
Multiplication	*	A=5, B=3.5 ; A*B donne 17.5
Division (normale)	/	A=5, B=3.5 ; A / B donne 1.428

8.3. Le Type Caractère et Chaîne

Le type Caractère ou Chaîne est ensemble de caractères comportant :

- les 26 lettres alphabétiques en majuscules ('A' jusqu'à 'Z')
- les 26 lettres alphabétiques en minuscules ('a' jusqu'à 'z')
- les 10 chiffres arabes ('0' jusqu'à '9').
- quelques caractères spéciaux.

Remarque : Chaque valeur de caractère est délimitée par deux apostrophes ` ` . Généralement, le type caractère est codé sur un octet (2⁸ valeurs). Le type Chaîne est composé de plusieurs caractères.

a) Représentation algorithmique :

Var Nomvar : **Caractère** ;

Si on utilise plusieurs variables en même temps alors sa syntaxe est :

Var Nomvar1, Nomvar2,, NomvarN : **Caractère** ;

Var Nomvar : **Chaîne** ;

Si on utilise plusieurs variables en même temps alors sa syntaxe est :

Var Nomvar1, Nomvar2,, NomvarN : **Chaîne** ;

b) Représentation en langage Pascal :

Var Nomvar : **Char** ;

Var Nomvar1, Nomvar2,, NomvarN : **Char** ;

Var Nomvar : **String** ;

Var Nomvar1, Nomvar2,, NomvarN : **String** ;

c) Représentation en langage Fortran :

Character Nomvar (Nomvar contient par défaut un seul caractère)

Character Nomvar1, Nomvar2,, NomvarN

Character*5 Nomvar (Nomvar contient une chaîne de 5 caractères)

Character*5 Nomvar1, Nomvar2,, NomvarN

Les opérations de base sur le type Caractère :

Opération	Symbole	Exemple
Addition (Concaténation)	+ (Concate)	Si A='Bon' et B='jour' alors A+B (ou A Concate B) donne 'Bonjour'
Successeur ou Suivant	Succ ou Suiv	Si A='D' alors Succ(A) donne 'E'
Prédécesseur ou Précédent	Pred ou Prec	Si A='D' alors Pred(A) donne 'C'

8.4. Le Type Booléen ou Logique

Un type Booléen dit aussi logique est composé uniquement de deux valeurs contradictoires (Vraie (1) ou faux (0)). Il est codé sur un Octet (2⁸ valeurs).

a) Représentation algorithmique :

Var Nomvar : **Logique** ;

Si on utilise plusieurs variables en même temps alors sa syntaxe est :

Var Nomvar1, Nomvar2,, NomvarN : **Logique** ;

b) Représentation en langage Pascal :

Var Nomvar : **Boolean** ;

Var Nomvar1, Nomvar2,, NomvarN : **Boolean** ;

c) Représentation en langage Fortran :

Logical Nomvar (Nomvar contient une seule valeur (True ou False))

Logical Var Nomvar1, Nomvar2,, NomvarN

Les opérateurs logiques

Dans le type Booléen, on utilise trois types d'opérateurs logiques pour constituer une expression logique. Ces opérateurs sont : ET, OU, NON.

Opérateurs logiques	Fonction
Et	réalise une conjonction entre deux valeurs booléennes
Ou	réalise une disjonction entre deux valeurs booléennes
Non	réalise une négation d'une valeur booléenne

La table de vérité de ces opérateurs pour deux variables booléennes A et B, donne le tableau suivant :

A	B	NON A	NON B	A ET B	A OU B
0	0	1	1	0	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	1	1

Exercices sur les affectations

Exercice 1

a) À l'issue de l'algorithme suivant, quelles seront les nombres qui sont stockés dans les variables A, B?

```
Algorithme Exemple1 ;  
Var A,B,C : Entiers ;  
Début  
A ← 3 ;  
B ← 4 ;  
C ← A ;  
A ← B ;  
B ← C ;  
Ecrire (A,B) ;  
Fin
```

b) Que fait cet algorithme?

Exercice 2

À l'issue de l'algorithme suivant, quel nombre est stocké dans les variables A, B, C ?

```
Algorithme Exemple2 ;  
Var A,B,C : Entiers ;  
Début  
A ← 5 ;  
B ← 8 ;  
C ← A + B ;  
B ← A × B ;  
A ← C ;  
Ecrire (A,B,C) ;  
Fin
```

Exercice 3

Quelle est la valeur finale de la variable A dans la séquence d'instructions suivantes?

```
A ← 9 ;  
A ← A + 1 ;  
A ← 4 × A;
```

Exercice 4

Quelle est la valeur stockée dans la variable B à l'issue du déroulement de l'algorithme suivant?

```
Algorithme Exemple3;  
Var A,B : Logiques ;  
Début  
A ← Vrai ;  
B ← Non (A) ;  
Fin.
```

9. Les Actions / instructions Algorithmiques

Dans cette partie, les actions simples concernent les instructions d'Affectation, de Lecture et d'Ecriture.

9.1. L'instruction d'affectation

Affecter une variable consiste à lui donner une valeur. Cette valeur peut être soit une constante, soit une valeur d'une autre variable, soit le résultat d'un calcul.

a) Représentation algorithmique

L'affectation est représentée par une flèche orientée à gauche \leftarrow .

b) Représentation en langage Pascal

L'affectation est représentée par le symbole (deux points égale) $:=$.

c) Représentation en langage Fortran

L'affectation est représentée par le symbole (égale) $=$.

Exemple :

Si A, B sont deux variables de type Entier alors on peut écrire :

Algorithme	Pascal	Fortran
$B \leftarrow 15 ;$	$B := 15 ;$	$B = 15$
$A \leftarrow B + 4 ;$	$A := B + 4 ;$	$A = B + 4$
$A \leftarrow A + 1 ;$	$A := A + 1 ;$	$A = A + 1$

1/ La valeur 15 est affecté au terme de gauche (variable B)

2/ La variable A reçoit la valeur du terme de droite ((valeur de la variable B) + 4)

3/ La variable A reçoit la valeur de A (avant instruction) + la valeur 1).

Remarque :

- Si une variable est de type numérique alors on écrit $A \leftarrow 0$.
- Si une variable A est de type chaîne de caractères alors sa valeur de droite s'écrit entre guillemets. Exemple : $A \leftarrow "0"$, ou $A \leftarrow "Lettres"$

Quelques remarques :

Beaucoup de langages de programmation (C, Fortran, Pascal, ...) utilisent le signe égal = pour l'affectation ←. Attention aux confusions:

- l'affectation n'est pas commutative : $A=B$ est différente de $B=A$
- l'affectation est différente d'une équation mathématique :
- $A=A+1$ a un sens en langages de programmation
- $A+1=2$ n'est pas possible en langages de programmation et n'est pas équivalente à $A=1$

9.2. L'entrée/ la sortie d'information

L'entrée ou la sortie d'information est représentée par les primitives de lecture et d'écriture d'information. Ces instructions de lecture et d'écriture permettent à la machine de communiquer avec l'utilisateur

- **La primitive d'entrée** ou saisir (entrée clavier) et lire (lecture des données d'entrée). Le but de ces primitives est de permettre à l'ordinateur d'affecter une variable extérieure à une autre variable. Le nom de cette variable symbolise une adresse en mémoire centrale. A cette adresse se trouve la valeur à un moment donné de la variable.
- **La primitive de sortie** : écrire, afficher, imprimer. Le but est de permettre à l'ordinateur de sortir la valeur d'une variable vers les périphériques extérieurs (écran, imprimante, etc....).

Ces instructions de lecture et d'écriture permettent à la machine de communiquer avec l'utilisateur.

9.2.1. Instruction de Lecture

L'instruction de **lecture** permet d'entrer des données à partir des périphériques d'entrée (Clavier, Souris, Stylo optique, etc.) vers la mémoire centrale.

Syntaxe 1:

Algorithme	Pascal	Fortran
Lire (NomVar1) ;	Read (NomVar1); ou ReadLn (NomVar1);	Read*,NomVar1
Lire (NomVar2) ;	Read (NomVar2); ou ReadLn (NomVar2);	Read*,NomVar2
....
Lire (NomVarN) ;	Read (NomVarN); ou ReadLn (NomVarN);	Read*,NomVarN

Pour une lecture de plusieurs variables, les variables peuvent être représentées sur plusieurs lignes ou regroupées dans une même ligne séparées par des virgules (Syntaxe2).

Syntaxe 2:

	1ère forme	2ème forme
Algorithme	Lire (NomVar1) ; Lire (NomVar2) ; Lire (NomVarN) ;	Lire (NomVar1, NomVar2,...., NomVarN) ;
Pascal	ReadLn (NomVar1) ; ReadLn (NomVar2) ; ReadLn (NomVarN) ;	ReadLn (NomVar1, NomVar2,...., NomVarN);
Fortran	Read*, NomVar1 Read*, NomVar2 Read*, NomVarN	Read*, NomVar1, NomVar2,...., NomVarN

Remarque :

L'instruction de Lecture ne peut pas lire :

- une constante

Si PI=3.14 déclarée comme constante, Lire (PI) est impossible.

- une expression arithmétique ou logique.

Lire (a+b) est impossible.

- un message.

Lire ('Bonjour') est impossible.

9.2.2. Instruction d'Écriture

L'instruction d'écriture permet d'afficher des résultats sur les périphériques de sortie (écran, imprimante, fichier, etc.).

Une action d'écriture peut se faire sur:

- une ou plusieurs variables,
- des constantes,
- des expressions arithmétiques et logiques,
- des messages.

a) Écriture sur une ou plusieurs variables

L'écriture pour une seule variable est comme suit :

Syntaxe 1:

Algorithme	Écrire(NomVar) ;
Pascal	Write (NomVar); ou WriteLn(NomVar);
Fortran	Write(*,*) NomVar ou Print*, NomVar

Pour plusieurs variables, deux formats d'écriture peuvent être appliqués :

- soit une écriture de toutes les variables ensemble (Syntaxe 2) :

Syntaxe 2 :

Algorithme	Ecrire (NomVar1, NomVar2,....., NomVarN) ;
Pascal	WriteLn (NomVar1, NomVar2,....., NomVarN);
Fortran	Write(*,*) NomVar1, NomVar2,....., NomVarN

- soit une écriture pour chaque variable et dans ce cas on a le schéma suivant (Syntaxe3):

Syntaxe 3 :

	1^{ère} forme	2^{ème} forme
Algorithme	Ecrire (NomVar1) ; Ecrire (NomVar2) ; Ecrire (NomVarN)	
Pascal	Write (NomVar1) ; Write (NomVar2) ; Write (NomVarN) ;	Writeln (NomVar1) ; Writeln (NomVar2) ; Writeln (NomVarN) ;
Fortran	Write(*,*) NomVar1 Write(*,*) NomVar2 Write(*,*) NomVarN	Print*, NomVar1 Print *, NomVar2 Print *, NomVarN

b) Écriture d'une constante

C'est une opération qui consiste à afficher la valeur d'une constante

Écrire(ValConst) ou

Écrire(NomConst) ; NomConst contient la valeur de ValConst.

Exemples (Ecriture de la constante de pesanteur g)

Écrire (9.81) ;

Écrire(g) ; où g a été déclarée comme constante dans la partie déclaration

c) Écriture d'une expression arithmétique ou logique

Dans certaines situations d'optimisation du nombre de variables et d'instructions dans l'algorithme (ou programme), on réalise directement une primitive d'écriture sur une expression.

Écrire (expression) ;

Exemples :

Écrire (A*b+C /D) avec A, B, C ,D des entiers (ou Réels)

Écrire (A et B ou C) avec A,B,C des Booléens

d) Écriture d'un message

Dans un programme, il est fortement conseillé d'écrire des messages à l'écran afin de prévenir l'utilisateur de ce qu'il doit frapper (**Lire**) ou afficher à l'écran (**Write**).

Un message est une suite de caractère ayant un sens et délimité par deux apostrophes.

Exemple :

- Au moment de la lecture (saisie)
Ecrire (' Entrer la valeur de la température') ;
Lire (T) ;
- Pour afficher le résultat d'un calcul
Ecrire (' Voici le résultat de la valeur de la température') ;
Ecrire (T) ;

e) Écriture Mixte

Dans certains algorithmes (ou programmes), l'affichage est mixte où il est composé de messages et de valeurs.

Exemple :

Pour afficher le message suivant : " La valeur de la température est = 23 degrés Celsius " il faut procéder comme suit :

Écrire ('La valeur de la température est = ', TEMP, '°C') ; où TEMP est la variable qui contient cette valeur.

Exercices sur les entrées et sorties

Exercice 5 : Que fait l'algorithme suivant ?

```
Algorithme exo5 ;  
Var A,B,C : Entiers ;  
Début  
Lire (A) ;  
Lire (B) ;  
C ← A*B ;  
D ← 2*(A+B) ;  
Ecrire (C) ;  
Ecrire (D) ;  
Fin
```

Exercice 6 : Que fait l'algorithme suivant ?

```
Algorithme exo6 ;  
Var A, R, S : Réels ;  
Début  
Lire (A) ;  
R ← A/2 ;  
S ← 3, 14*R^2 ;  
Ecrire (S) ;  
Fin
```

Exercice 7 : Écrire un algorithme qui lit deux nombres entiers A et B et calcule le reste la division euclidienne de A et B . Traduire cet algorithme en Pascal puis en Fortran.

Exercice 8 : Écrire un algorithme (puis traduire en un programme Pascal et ensuite en Fortran) qui demande d'entrer un nombre puis affiche son image par la fonction f définie par : $f(x) = 2x^2 + x + 7$.

Exercice 9

a) Écrire un algorithme qui convertit des secondes en heures, minutes et secondes. Traduire cet algorithme en Pascal puis en Fortran.

b) Écrire un algorithme qui convertit des heures en jours et heures. Traduire cet algorithme en Pascal puis en Fortran.

Exercice 10 : Écrire un algorithme (puis traduire en un programme Pascal et ensuite en Fortran) qui demande d'entrer trois nombres A , B et C , et calcule et affiche leur moyenne non pondérée.

9.3. Structures de contrôles

Les instructions d'un programme sont exécutées d'une manière séquentielle : la première instruction, ensuite la deuxième, après la troisième et ainsi de suite. Cependant, dans plusieurs cas, on est amené soit à choisir entre deux ou plusieurs chemins d'exécution (un choix entre deux ou plusieurs options), ou bien à répéter l'exécution d'un ensemble d'instructions, pour cela nous avons besoins de structures de contrôle pour contrôler et choisir les chemins d'exécutions ou refaire un traitement plusieurs fois. Les structures de contrôle sont de deux types : Structures de contrôles conditionnelles et structures de contrôle répétitives (itératives). Mais dans certains programmes, une instruction peut être exécutée directement à un endroit précis en sautant des instructions. Cette opération est appelée : instruction de Branchement.

9.3.1. Structures de contrôles conditionnelles

Ces structures sont utilisées pour décider de l'exécution d'un bloc d'instructions : est ce que ce bloc est exécuté ou non. Ou bien pour choisir entre l'exécution de deux blocs différents. On distingue plusieurs formes de structures conditionnelles.

a) Conditionnelle à un Choix (Test alternative simple)

Un test simple contient un seul bloc d'instructions. Selon une condition (expression logique), on décide est ce que le bloc d'instructions est exécuté ou non. Si la condition est vraie, on exécute le bloc, sinon on ne l'exécute pas. La syntaxe d'un test alternatif simple est comme suit :

Syntaxe :

Algorithme	Pascal	Fortran
Si <Condition> Alors <instruction(s)> ; Finsi ;	If <condition> Then begin <instruction(s)>; End ;	If <condition> Then <instruction(s)> Endif ;

Exemple :

Algorithme	Pascal	Fortran
Lire(x) ; si (x > 2) alors x ← x + 3 ; finsi Ecrire (x) ;	Read(x); if (x > 2) then begin x:= x + 3; end ; write(x);	Read*, x if (x > 2) then x = x + 3 endif ; write (*,*) x;

Remarque :

Dans le langage PASCAL, un bloc est délimité par les deux mots clés **begin** et **end**. Si le bloc contient une seule instruction alors les mots (**begin** et **end**) seront facultatifs (on peut les enlever).

b) Conditionnelle à deux Choix (Test alternative double)

Un test double contient deux blocs d'instructions : on est amené à décider entre le premier bloc ou le second. Cette décision est réalisée selon une condition (expression logique ou booléenne) qui peut être vraie ou fausse. Si la condition est vraie on exécute le premier bloc, sinon on exécute le second.

Syntaxe :

Algorithme	Pascal	Fortran
Si <Condition> Alors <instruction(s) 1> ; Sinon <instruction(s) 2> ; Finsi ;	If <condition> Then begin <instruction(s)1>; End Else begin <instruction(s) 2> ; End ;	If <condition> Then <instruction(s)1> Else <instruction(s) 2> Endif ;

Exemple :

Ecrire un Algorithme(ou programme) qui permet de résoudre une équation de 1er degré de la forme $ax+b=0$.

Algorithmme	Pascal	Fortran
Algorithmme Equation1 ; Var a,b : Entier ; Var x : Réel ; Début Lire (a,b) ; Si (a = 0) alors Si (b = 0) alors Ecrire ('Infinité de solutions'); Sinon Ecrire ('Pas de solution') ; Else $x \leftarrow -b/a$; Ecrire (x) ; Finsi Finsi Fin.	Program Equation1 ; Var a,b : Integer ; Var x: Real ; Read (a,b); If (a = 0) Then If (b = 0) Then Write ('Infinité de solution') Else Write ('Pas de solution') Else begin $x := -b/a$; Ecrire (x) end End .	Program Equation1 Integer a, b Real x Read*, a, b If (a = 0) Then If (b = 0) Then Print* , "Infinité de solution" Else Print* , "Pas de solution" Else $x = -b/a$ Write(*,*) x Endif Endif End

Remarque :

- Dans le langage PASCAL, il ne faut jamais mettre de point-virgule avant **else**.
- Dans l'exemple précédent, on peut enlever **begin end** du **if** et ceux de **else** puisqu'il y a une seule instruction dans les deux blocs.

c) Conditionnelle à Choix Multiple

Par définition, une structure à choix multiple est une structure qui à partir d'un choix va se positionner sur le bon traitement sans passer par les autres ; on la note par **SELON**.

Syntaxe :

Algorithmme	Pascal	Fortran
Selon (NomVar) faire Val1 : <Instruction1> ; Val2 : <Instruction2> ; ValN : <Instruction N> ; Sinon <Instruction autre> ; FinSelon	case NomVar of Val1: instruction1; Val2: instruction2; ... ValN: instructionN else instructionAutre ; end	Select case (NomVar) Case (Val1) instruction1 Case (Val2) Instruction2 Case (ValN) InstructionN Case Default InstructionDefault End Select

Exemple en Algorithme:

Selon (Mois) Faire

1,3,5,7,8,10,12: **Ecrire**(" Nombre de jour est ", 31) ;

4,6,9,11: **Ecrire**(" Nombre de jour est ", 30) ;

2: **Ecrire**(" Nombre de jour est ", 28)

Sinon: Ecrire(" Nombre de jour est ", 0) ;

Fin Selon

Exemple en Pascal

Case Mois Of

1,3,5,7,8,10,12: **Write**(" Nombre de jour est ", 31) ;

4,6,9,11: **Write** (" Nombre de jour est ", 30) ;

2: **Write** (" Nombre de jour est ", 28)

Else Write (" Nombre de jour est ", 0) ;

End

Exemple en Fortran

Select Case Mois

Case (1,3,5,7,8,10,12)

Write(*,*) " Nombre de jour est ", 31

Case(4,6,9,11)

Write(*,*) " Nombre de jour est ", 30

Case(2)

Write(*,*) " Nombre de jour est ", 28

Case default

Write(*,*) " Nombre de jour est ", 0

End Select

9.3.2. Structures de contrôle répétitives

Par définition, les boucles sont utilisées pour qu'une séquence d'instructions soit répétée un nombre donné de fois ou tant qu'une condition n'est pas remplie.

Différentes structures permettent de réaliser cette forme de traitement:

- ❖ Pour.. Faire
- ❖ Tant que.. Faire
- ❖ Répéter, Jusqu'à

L'utilisation d'une telle ou telle structure dépend essentiellement de la nature du problème à résoudre.

9.3.2.1. La Structure itérative POUR (For)

La boucle Pour est une structure répétitive qui itère le même traitement pour une plage de valeurs entières comprises entre une borne inférieure et une borne supérieure. La mise à jour étant automatique, l'arrêt du traitement de la boucle Pour se réalise lorsqu'on dépasse l'une des bornes

La syntaxe en algorithmique de la boucle **Pour** s'écrit comme suit :

Pour i allant de Vi jusqu'à Vf, [Pas = Val_pas] **Faire**
<Traitement> ;
Fin Pour

Ou bien :

Pour i allant de Vi à Vf, [Pas = Val_pas] **Faire**
<Traitement> ;
Fin Pour

Ou bien :

Pour i de Vi à Vf, [Pas = Val_pas] **Faire**
<Traitement> ;
Fin Pour

Avec

i : variable de type entier servant de compteur.

Vi : valeur initiale que va prendre i.

Vf : valeur finale que prendra i.

Pas : contient une valeur entière indiquent la valeur de l'incrémentatation de i (mise à jour de i).

La syntaxe en Langage Pascal et Fortran de la boucle Pour est comme suit :

Pascal	Fortran
For I := Vi to Vf Do Begin Instruction; End;	Do I = Vi, Vf, Pas Instruction EndDo

Exemple : Écrire un programme affichant tous les nombres de 1 à 40.

Algorithmme	Pascal	Fortran
Pour i Allant de 1 à 40 Faire Ecrire(i) ; Fpour	For i:= 1 to 40 Do Begin Write(i); End;	Do i = 1, 40 Write(*,*) i EndDo

Deuxième syntaxe générale en Pascal :

FOR I := Vf DOWNTO Vi DO

Begin

Séquence d'instructions ;

End ;

Dans cette variante avec " DOWNTO ", le principe d'itération est le même, mais la variable de boucle prend tour à tour des valeurs décroissantes.

Exemple : Écrire un programme affichant tous les nombres de 10 à 1.

For i: = 10 Downto 1 Do

Begin

Write(i);

End;

9.3.2.2. La Structure itérative TANT QUE(While)

La boucle **Tant que** permet d'exécuter le corps de la boucle lorsque la condition d'exécution est vérifiée ; on s'arrêtera dès que la condition n'est plus vérifiée. La condition est testée en début de boucle.

Syntaxe en algorithmique :

Tant que <condition> **Faire**

<instruction(s)> ;

FinTant que;

La syntaxe en Pascal et Fortran est comme suit :

Pascal	Fortran
While <condition> Do Begin <instruction(s)> ; End;	Do While <condition> <instruction(s)> End do

<Condition> : expression logique qui peut être vraie ou fausse.

On exécute le bloc d'instructions tant que la condition est vraie. Une fois la condition est fausse, on arrête la boucle, et on continue l'exécution de l'instruction qui vient après fin Tant que (après end).

Remarque :

Comme dans la boucle **for**, il faut jamais mettre de point-virgule après **do**.

Toute boucle **Pour** peut être remplacée par une boucle **Tant que**, cependant l'inverse n'est pas toujours possible.

Exemple :

Écrire un algorithme (Programme) qui décompte de 9 à 0.

Algorithme Exemple ;

Var A : entier ;

A ← 10 ;

Tant que (A > 0) **faire**

A ← A - 1 ;

Ecrire (A) ;

Fin tant que

Fin

Cet exemple en pascal (Fortran) devient :

Pascal	Fortran
Program Exemple; Var A : Integer ; A := 10; While (A > 0) Do Begin A := A - 1; Write (A); End ; End.	Program Exemple Integer A A = 10 Do While (A > 0) A = A - 1 Write (*,*) A End do End

9.3.2.3. La Structure itérative Répéter (Repeat)

La structure de contrôle répétitive **Répéter** (**Repeat** en langage **PASCAL**) utilise une expression logique ou booléenne comme condition de sortie de la boucle : si la condition est vérifiée (elle donne un résultat vrai: TRUE) on sort de la boucle, sinon on y accède (on répète l'exécution du bloc).

La syntaxe de la boucle répéter est comme suit :

Algorithme	Pascal
Repéter <instruction(s)> ; Jusqu'à <condition>;	Repeat <instruction(s)>; Until <condition>;

<Condition> : expression logique qui peut être vraie ou fausse.

On exécute le bloc d'instructions jusqu'à avoir la condition correcte. Une fois la condition est vérifiée, on arrête la boucle, et on continue l'exécution de l'instruction qui vient après jusqu'à (après until).

Remarque :

Dans la boucle **repeat**, on n'utilise pas **begin** et **end** pour délimiter le bloc d'instructions (le bloc est déjà délimité par **repeat** et **until**).

La différence entre la boucle **répéter** et la boucle **tant que** est :

– La condition de répéter est toujours l'inverse de la condition **tant que** : pour répéter c'est la condition de sortie de la boucle, et pour **tant que** c'est la condition d'entrer.

– Le teste de la condition est à la fin de la boucle (la fin de l'itération) pour répéter. Par contre, il est au début de l'itération pour la boucle ***tant que***. C'est à dire, dans ***tant que*** on teste la condition avant d'entrer à l'itération, et dans répéter on fait l'itération après on teste la condition.

En langage Fortran90, il n'existe pas d'instruction Repeter.

Exemple :

Écrire un algorithme (Programme) qui calcul la somme des nombre de 0 à 10.

Algorithme	Pascal
Algorithme Som; Var i,S: Entiers; Début S ← 0; i ← 0; REPETER S ← S +i; i ← i + 1; JUSQU'A (i > 10) ; Ecrire (S); Fin.	Program Som; Var i,S : Integer; Begin S := 0; i := 0; Repeat S := S +i; i := i + 1; Until (i > 10) ; Write (S); End.

9.3.3. Instruction de branchement (SAUT)

Une instruction de branchement nous permet de sauter a un endroit du programme et continuer l'exécution a partir de cet endroit. Pour réaliser un branchement, il faut tout d'abord indiquer la cible du branchement via une étiquette.

Syntaxe générale :

Algorithme	Pascal	Fortran
ALLER A <étiquette> ;	GOTO <étiquette> ;	GOTO <étiquette>

Remarque :

- Une étiquette représente un numéro (nombre entier), exemple : 1, 2, 3, etc.
- Dans un programme PASCAL, il faut déclarer les étiquettes dans la partie déclaration avec le mot clé label. Par contre en Fortran, on n'a pas besoin de déclarer l'étiquette.

- Une étiquette désigne un seul endroit dans le programme, on ne peut jamais indiquer deux endroits avec une même étiquette.
- Par contre, on peut réaliser plusieurs branchements vers une même étiquette.
- Un saut ou un branchement peut être vers une instruction antérieure ou postérieure (avant ou après le saut).

Exemple:

Algorithme	Pascal	Fortran
Algorithme Exemple; Var x : Entier; Etiquette 5,6 ; Début Lire (x); Si (x <0) alors Aller à 5; Finsi x ← x + 1; Aller à 6; 5: x ← x +2; 6 : Ecrire (x); Fin.	Program Exemple; Var x : Integer ; Label 5,6 ; Begin Read (x); If (x<0) then Goto 5; x := x + 1; Goto 6; 5: x := x +2; 6: Write (x); End.	Program Exemple Integer x Read* ,x If (x <0) then Goto 5 Endif x = x + 1 Goto 6 5: x = x +2 6: Print* , x End

Exercices sur les structures conditionnelles

Exercice 11

Écrire un algorithme qui demande l'âge de l'utilisateur et affiche le message "vous êtes mineur" ou "vous êtes majeur" suivant le cas. Traduire cet algorithme en programme Pascal et en Fortran.

Exercice 12

Écrire un algorithme qui lit la température extérieure en degrés Celsius et affiche le message "il gèle" si le nombre est négatif et "alerte à la canicule" si le nombre est supérieur à 30. Traduire cet algorithme en programme Pascal et en Fortran.

Exercice 13

Écrire un algorithme qui, à partir d'un nombre entré par l'utilisateur, affiche ce même nombre s'il est positif et son opposé s'il est négatif (le nombre obtenu est appelé la valeur absolue du nombre entré). Traduire cet algorithme en programme Pascal et en Fortran.

Exercice 14

Concevoir un algorithme qui demande à l'utilisateur d'entrer un nombre (représenté par la variable a). Ensuite, si le nombre entré est différent de 1, l'algorithme doit stocker dans une variable b la valeur de $5/(x-1)$ et afficher la valeur de b. Traduire cet algorithme en programme Pascal et en Fortran.

Exercice 15

Écrire un algorithme qui permet d'indiquer si un nombre entier est pair ou non. Traduire cet algorithme en PASCAL puis en FORTRAN

Exercice 16

Écrire un algorithme qui, à partir de la donnée de la longueur de chacun des trois côtés d'un triangle, teste si le triangle est rectangle. Traduire cet algorithme en programme Pascal et en Fortran.

Exercices sur les BOUCLES

Exercice 17

Écrire un algorithme (Traduire en un programme PASCAL et FORTRAN) qui calcule la somme des nombres entiers de 0 à 50.

Exercice 18

Écrire un algorithme (Traduire en un programme PASCAL et FORTRAN) qui calcule le produit des nombres entiers de 1 à 10.

Exercice 19

Écrire un algorithme (Traduire en un programme PASCAL et FORTRAN) qui calcule la somme des 20 premiers nombres impairs.

Exercice 20

Écrire un algorithme (Traduire en un programme PASCAL et FORTRAN) qui calcule la somme des 20 premiers nombres pairs.

Exercice 21

Écrire un algorithme (Traduire en un programme PASCAL et FORTRAN) permettant de calculer le PGCD (algorithme d'Euclide) de deux nombres entiers A et B positifs.

Exercice 22

Écrire un algorithme (Traduire en un programme PASCAL et FORTRAN) qui calcule et affiche le factoriel d'un nombre entier N .

$$N! = 1 * 2 * \dots * (N - 2) * (N - 1) * N$$

Exercice 23

Écrire un algorithme (Traduire en un programme PASCAL et FORTRAN) qui calcule et affiche la N ème puissance d'un nombre réel X qui doit être lu et affiché en entrée.

$$X^N = X * X * \dots * X.$$

9.4. Tableaux et Matrices

Les variables utilisées, jusqu'à présent, étaient toutes de type simple prédéfini (entier, réel, caractère ou logique). Dans cette partie, nous allons voir une autre manière de représenter les données, c'est le type structuré Tableau. Une variable de type tableau n'est plus une variable simple, mais une structure de données, regroupant des informations de même type.

Les tableaux permettent de manipuler plusieurs informations de même type, de leur mettre un indice : la 1 ère info, la 2ème info, . . ., la ième info, . . . Ils sont stockés en mémoire centrale comme les autres variables, contrairement aux fichiers qui sont stockés sur le disque. Une propriété importante des tableaux est de permettre un accès direct aux données, grâce à l'indice. Nous distinguons deux types de tableaux: les tableaux à une dimension (Vecteurs) et les tableaux à plusieurs dimensions(Matrices).

Une propriété importante des tableaux est de permettre un accès direct aux données, grâce à l'indice.

9.4.1 Tableaux à une dimension (vecteurs)

Définition

On appelle un vecteur V d'ordre n , noté $V(n)$, un ensemble de n éléments de même type disposés dans un tableau à n cases. Un vecteur possède un identificateur (nom du vecteur) et un indice qui indique le numéro de la case.

Exemple: Soit P un vecteur de 5 cases (valeurs) :

5	10	-4	3	12
---	----	----	---	----

$P(i)$ est le contenu (l'élément) de la case $N^{\circ}i$.

$P(3)$ est égal à -4 , c'est à dire l'élément de la case $N^{\circ} 3$ est -4 .

a) Déclaration d'un vecteur:

La syntaxe algorithmique de la déclaration d'une variable de type tableau à une dimension (vecteur) se fait par :

Var <identificateur_tab >: Tableau (N) de <Type >.

Où:

- <identificateur_tab > est le nom du vecteur.
- <N > est le nombre d'éléments du vecteur.
- <Type > est le type des éléments du vecteur.

Exemple :

Algorithme	Var T: Tableau (100) d'Entiers ; Var P: Tableau (26) de Car;
Pascal	Var T: Array [1..100] of integer ; Var P: Array ['A'..'Z'] of Character ;
Fortran	Integer, Dimension T(100) Character, Dimension P(26)

Remarque : L'intervalle du tableau peut être de tout type intervalle, par exemple 1..10, 'a'..'z', false..true, ou encore un intervalle d'énumérés Dimanche..Samedi.

Remarque : Si on a plusieurs tableaux (vecteurs) de même type, on peut les déclarer tous ensembles en les séparant par des virgules.

Exemple: Soient A, B et C trois vecteurs de même type (entiers), leur déclaration sera faite comme suit:

Algorithme	Var A,B,C : Tableau(10) d'entiers ;
Pascal	Var A,B,C : Array[1..10] of Integer;
Fortran	Integer, Dimension A(10), B(10),C(10)

b) Initialisation d'un élément d'un vecteur (affectation) :

L'initialisation d'un élément du vecteur se fait comme suit :

< identificateur > (Indice) ← < expression > ;

Exemple:

Soit A un vecteur de 3 éléments entiers

$$A(1) \leftarrow -5 ;$$

$$A(2) \leftarrow 7 ;$$

$$A(3) \leftarrow A(1) - A(2) ;$$

Le déroulement de cet exemple nous donne le vecteur suivant:

A	-5	7	-12
---	----	---	-----

c) Lecture et affichage des éléments d'un vecteur

Les cases d'un tableau étant des variables simples, on leur attribue une à une des valeurs à partir du clavier par les instructions :

Lire (V(1)) , Lire (V(2)) ,. . . Lire (V(N)) ;

Remarquons que l'instruction *Lire* est répétée *N* fois. Afin d'éviter cette répétition, nous utilisons une des structures itératives que nous avons vu précédemment. Le nombre d'itérations étant connu (*N*), pour ce la nous utilisons la boucle **Pour**.

Syntaxe :

Pour i allant de 1 à N faire

Lire (V(i)) ;

Fpour ;

La Syntaxe en langage de programmation devient :

Pascal	Fortran
For i :=1 to N do begin Read (V[i]) ; End ;	Do i = 1, N Read (*,*) V(i) Endo

De même que pour le remplissage d'un vecteur, l'affichage de son contenu se fait par l'instruction :

Pour i allant de 1 à N faire

Ecrire (V(i)) ;

Fpour ;

De même, cette syntaxe en langage de programmation devient :

Pascal	Fortran
For i :=1 to N do begin Write (V[i]) ; End ;	Do i = 1, N Write(*,*) V(i) Endo

d) Quelques opérations sur les vecteurs

i. Somme des éléments d'un tableau

Pour calculer la somme des nombres contenus dans un vecteur, il faut ajouter un à un le contenu des cases, depuis la première jusqu'à la dernière. Le résultat final nous donne un nombre et non un vecteur.

Exemple:

Ecrire l'algorithme et le programme (Pascal/Fortran) qui calcule et affiche la somme des éléments d'un vecteur V de 10 éléments.

```

Algorithme Somme ;
Var V : Tableau (10) d'entiers ;
    S, i : entiers ;
Début
Ecrire (" Introduire les éléments du tableau V") ;
    S ← 0 ;
    Pour i allant de 1 à 10 Faire
        Lire (V(i)) ;
        S ← S + V(i) ;
    Fpour
    Ecrire (" La somme des éléments du tableau V est S = ", S) ;
Fin.

```

La Syntaxe en langage de programmation Pascal est :

```

Program Somme ;
Var V : Array [1..10] of Integer ;
    S, i : Integer ;
Begin
Write (' Introduire les éléments du tableau V ') ;
    S := 0 ;
    For i := 1 to 10 do
Begin

```



```

Lire (V[i]) ;
S := S + V[i] ;
End ;
Write (' La somme des éléments du tableau V est S = ', S) ;
End.

```

La Syntaxe en langage de programmation Fortran est :

```

Program Somme
Integer, Dimension V(10), S, i : entiers
Write(*,*) ' Introduire les éléments du tableau V '
  S = 0
  Do i = 1,10
    Read(*,*) V(i)
    S = S + V(i)
  Endo
  Write(*,*) ' La somme des éléments du tableau V est S = ', S
End

```

ii. Recherche du maximum des éléments d'un vecteur

La recherche d'un maximum des éléments d'un vecteur se fait comme suit:

- on fait entrer les éléments du vecteur;
- on suppose que la première case contient le maximum;
- on compare le contenu de la deuxième case avec le maximum précédent. Si celui-ci (l'élément de la deuxième case) est supérieur, il devient le maximum;
- on continue la même opération avec les cases suivantes.

Exemple:

Ecrire l'algorithme et le programme (Pascal/Fortran) qui recherche le maximum des éléments d'un tableau V de 5 éléments.

```

Algorithme MaxVect ;
Var T : Tableau (20) d'entiers ;
Var Max, i:entiers ;
Début
Ecrire(" Introduire les éléments du tableau ") ;
Pour i allant de 1 à 5 faire
  Lire(V(i)) ;

```

```

Fpour
Max ← V(1) ;
Pour i allant de 2 à 5 faire
Si (V(i) > Max ) alors
Max ← V(i) ;
Fsi
Fpour
Ecrire (" Le Maximum des éléments du tableau V est Max= ", Max) ;
Fin.

```

La Syntaxe en langage de programmation Pascal est :

```

Program MaxVect ;
Var T : Array [1..20] of Integer ;
Var Max, i: Integer ;
Begin
Write (' Introduire les éléments du tableau V') ;
For i := 1 to 5 do
Read(V[i]) ;
Max := V[1] ;
For i := 2 to 5 do
Begin
If (V[i] > Max ) Then
Max := V[i] ;
End
Write (' Le Maximum des éléments du tableau V est Max= ', Max) ;
End.

```

La Syntaxe en langage de programmation Fortran est :

```

Program MaxVect
Integer, Dimension T(20), Max, i
Write(*,*) ' Introduire les éléments du tableau V'
Do i = 1, 5
  Read(*,*) V(i)
Endo
Max = V(1)
Do i = 2, 5
If (V(i) > Max ) Then
Max = V(i)
Endif
Endo
Write(*,*) ' Le Maximum des éléments du tableau V est Max= ', Max
End

```

Remarque: Nous procédons de la même manière pour la recherche du plus petit élément (Minimum) du vecteur (Sauf dans le test Si, il faut mettre Inférieur (<)).

9.4.2. Matrices (Tableaux à deux dimensions)

Les tableaux manipulés jusqu'à présent sont à une dimension (vecteurs). Par ailleurs, lorsqu'un traitement utilise plusieurs tableaux à une dimension ayant le même nombre d'éléments et subissant le même traitement, on utilise un seul tableau à deux dimensions appelé : matrice. Ainsi par définition, on appelle une matrice A d'ordre $m \times n$, notée $A(M,N)$, un ensemble de $M \times N$ éléments rangés dans des cases disposées en M lignes et N colonnes.

Notons aussi, que l'élément d'une matrice noté $A(i, j)$ est repéré par deux indices; le premier indique la ième ligne et le second indique la jème colonne.

Exemple: Soit T une matrice d'ordre 3 X 2

11	5
0	-3
7	-6

$T(1, 1) = 11$, c'est-à-dire l'élément de la première ligne et la première colonne.

$T(2, 2) = -3$ et $T(3, 1) = 7$.

a) Déclaration d'une matrice :

En algorithmique, la déclaration d'une variable de type tableau à deux dimensions (matrice) se fait par la syntaxe suivante:

Var <iden_mat >: **Tableau** (NL ,NC) de <Type>.

Où :

- <iden_mat > est le nom de la matrice ;
- <NL> est le nombre de lignes ;
- <NC > est le nombre de colonnes;
- <Type > est le type des éléments de la matrice ;

Exemple:

Algorithme	Var P : Tableau (10 ,5) de réels ; Var i,j : entiers ;
Pascal	Var P : Array [1..10 ,1..5] of real ; Var i,j : integer ;
Fortran	Real, Dimension P(10,5) Integer, i,j

P est une matrice de 10 lignes et de 5 colonnes, ses cases sont destinées à contenir des éléments de type réel. $P(i, j)$ est le contenu de la case qui se trouve à la ligne i et la colonne j .

Remarque : Si on a plusieurs matrices de même ordre et de même type, on peut les déclarer à la fois en les séparant par des virgules.

Exemple: Soient A_1 , A_2 et A_3 trois matrices d'ordre 3×4 de même type (réel), leur déclaration peut être faite comme suit:

Algorithme	Var A_1, A_2, A_3 : Tableau (3,4) de réels ;
Pascal	Var A_1, A_2, A_3 : Array [1..3 ,1..4] of real ;
Fortran	Real, Dimension $A_1(3,4), A_2(3,4), A_3(3,4)$

b) Initialisation d'un élément d'une matrice :

Comme dans le cas d'un vecteur, l'initialisation d'un élément d'une matrice se fait par l'affectation selon la syntaxe suivante :

Algorithme	$\langle \text{identificateur} \rangle (\text{Indice-ligne}, \text{indice-colonne}) \leftarrow \langle \text{expression} \rangle ;$
Pascal	$\langle \text{identificateur} \rangle [\text{Indice-ligne}, \text{indice-colonne}] := \langle \text{expression} \rangle ;$
Fortran	$\langle \text{identificateur} \rangle (\text{Indice-ligne}, \text{indice-colonne}) = \langle \text{expression} \rangle$

Exemple:

Soit M une matrice d'ordre 2×3

$$M(1, 1) \leftarrow -3 ;$$

$$M(1, 2) \leftarrow 4 ;$$

$$M(1, 3) \leftarrow 7 ;$$

$$M(2, 1) \leftarrow M(1, 2) + 4 ;$$

$$M(2, 2) \leftarrow M(1, 1) + M(2, 1) ;$$

$$M(2, 3) \leftarrow M(1, 1) * M(2, 2) ;$$

Le déroulement de cet exemple nous donne la matrice suivante:

-3	4	7
8	5	-15

c) Lecture et affichage des éléments d'une matrice :

Une matrice de m lignes et de n colonnes est identique à m vecteurs superposés de n cases chacun. Par conséquent, le remplissage d'une matrice se fait, ligne par ligne ou colonne par colonne, par l'utilisation de l'instruction de Boucle comme suit:

Algorithme	Pascal	Fortran
Pour i allant de 1 à m faire Pour j allant de 1 à n faire Lire ($A(i,j)$) ; Fpour Fpour	For $i := 1$ to m do For $j := 1$ to n do Read($A[i,j]$);	For $i = 1, m$ For $j = 1, n$ Read(*,*) $A(i,j)$ Enddo Enddo

Dans cet exemple, on utilise deux boucles imbriquées où la première boucle englobe la seconde. L'indice i est utilisé pour le parcours des lignes et j pour les colonnes.

Déroulement du remplissage

– Initialement l'indice i de la première boucle prend la valeur 1 (première ligne).

- L'indice j de la boucle englobée progresse de 1 à n et l'instruction de lecture *Lire(Read)* est exécutée n fois, réalisant ainsi le remplissage (lecture) de la première ligne.
- L'indice i est incrémenté en prenant la valeur 2 (deuxième ligne).
- A nouveau, l'indice j progresse de 1 à n et l'instruction de lecture *Lire(Read)* est exécutée encore n fois, en effectuant ainsi le remplissage de la seconde ligne.
- De la même manière, l'opération du remplissage se poursuivra jusqu'à la dernière ligne ($i = m$).
- De même que pour le remplissage d'une matrice, l'affichage de son contenu se fait par l'instruction:

```

Pour i allant de 1 à m faire
    Pour j allant de 1 à n faire
        Écrire (A(i,j)) ;
    Fpour
Fpour
  
```

d) Quelques opérations sur les matrices :

Tout comme dans le cas des vecteurs, nous illustrons dans ce qui suit le calcul de la somme des éléments de la matrice et la recherche du maximum de la matrice.

i. Somme des éléments d'une matrice:

Le calcul de la somme des éléments d'une matrice de m lignes et de n colonnes est le même que celui des vecteurs. En effet, l'addition des éléments de la matrice se fait ligne par ligne, et le résultat final nous donne un nombre et non une matrice.

Exemple: Un algorithme (programme Pascal/Fortran) qui calcule et affiche la somme(S) des éléments d'une matrice P d'ordre 2×3 .

La syntaxe algorithmique est :

```

Algorithme Som ;
Var M : Tableau (10,10) d'entiers ;
Var S, i, j : entiers ;
  
```

```

Début
Ecrire (" Introduire les éléments de la matrice P ") ;
Pour i allant de 1 à 2 faire
Pour j allant de 1 à 3 faire
Lire (M(i, j)) ;
Fpour
Fpour
S ← 0 ;
Pour i allant de 1 à 2 faire
Pour j allant de 1 à 3 faire
S ← S + M(i, j) ;
Fpour
Fpour
Ecrire (" La somme des éléments de la matrice P est S = ", S) ;
Fin.

```

La syntaxe en langage de programmation est :

Pascal	Fortran
<pre> Program Som ; Var P : Array [1..10,1..10] of Integer ; Var S, i, j : Integer ; Begin Write (' Introduire les éléments de la matrice P ') ; For i := 1 to 2 do For j := 1 to 3 do Read (P[i, j]) ; S := 0 ; For i := 1 to 2 do For j := 1 to 3 do S := S + P[i, j] ; Write (' La somme des éléments de la matrice P est S = ', S) ; End. </pre>	<pre> Program Som Integer, Dimension P (10,10) Integer, S, i, j Write(*,*) 'Introduire les éléments de la matrice P ' For i = 1, 2 For j = 1, 3 Read(*,*) P(i, j) Enddo Enddo S = 0 For i = 1, 2 For j = 1, 3 S = S + P(i, j) Enddo Enddo Write(*,*) ' La somme des éléments de la matrice P est S = ', S End </pre>

Déroulement de l'algorithme:

- Entrée des éléments de la matrice.
- Initialisation de la variable S.
- Addition des éléments de la matrice ligne par ligne.

Application: Soit P une matrice d'ordre 2×3

4	5	7
3	-6	9

Le déroulement de la somme des éléments de cette matrice s'effectue comme suit :

i	j	P(i,j)	Somme (S)
1	1	4	4
1	2	5	9
1	3	7	16
2	1	3	19
2	2	-6	13
2	3	9	22

Ainsi, La somme finale des éléments de la matrice P est **22**.

ii. Recherche du maximum des éléments d'une matrice

Le principe est le même que pour la recherche du maximum des éléments d'un vecteur.

Exemple: Algorithme qui recherche le maximum des éléments d'une matrice P d'ordre 2 x 3.

La syntaxe algorithmique est :

```

Algorithme Max ;
Var P : Tableau (10,10) d'entiers ;
Var Max, i, j: entiers ;
Début
Ecrire (" Introduire les éléments du tableau P" ) ;
Pour i allant de 1 à 2 faire
Pour j allant de 1 à 3 faire
Lire(P(i, j)) ;
Fpour
Fpour
Max ← P(1, 1) ;
Pour i allant de 1 à 2 faire
Pour j allant de 1 à 3 faire
Si P(i, j) > Max alors
Max ← P(i, j);
Fsi
Fpour
Fpour
Ecrire (" Le Maximum des éléments de la matrice est Max= ", Max) ;
Fin.

```


La syntaxe en langage de programmation est :

Pascal	Fortran
<pre> Program Max ; Var P : Array [1..10,1..10] of Integer ; Var Max, i, j : Integer ; Begin Write (' Introduire les éléments de la matrice P ') ; For i := 1 to 2 do For j := 1 to 3 do Read (P[i, j]) ; Max := P[1, 1] ; For i := 1 to 2 do Begin For j := 1 to 3 do Begin If P(i, j) > Max Then Max := P[i, j]; End End Write (' Le Maximum des éléments de la matrice est Max= ', Max) ; End. </pre>	<pre> Program Max Integer, Dimension P (10,10) Integer, Max, i, j Write(*,*) ' Introduire les éléments de la matrice P ' For i = 1, 2 For j = 1, 3 Read(*,*) P(i, j) Enddo Enddo Max = P(1, 1) For i = 1, 2 For j = 1, 3 If P(i, j) > Max Then Max = P(i, j) Endif Enddo Enddo Write(*,*) ' Le Maximum des éléments de la matrice est Max= ', Max End </pre>

Application: Soit la matrice (P) suivante :

4	8	7
9	-6	5

Le déroulement de la recherche du maximum des éléments de la matrice P est le suivant:

i	j	P(i,j)	Max
1	1	4	4
1	2	8	8
1	3	7	8
2	1	9	9
2	2	-6	9
2	3	5	9

Ainsi, le Maximum des éléments de cette matrice est $Max = 9$.

Remarque : Nous procédons de la même manière pour la recherche du Minimum d'une matrice.

Exercices sur les Tableaux

Exercice 24

Écrire un algorithme qui recherche dans un vecteur V de dimension N , l'indice de la valeur X qui se trouve dans le vecteur. Traduire cet algorithme en programme Pascal et en Fortran.

Exercice 25

Écrire un algorithme qui inverse l'ordre d'un tableau de 10 entiers triés. En d'autres termes, si le tableau est trié du plus petit au plus grand, alors l'algorithme retourne le tableau trié du plus grand au plus petit ; réciproquement, si le tableau est trié du plus grand au plus petit, alors l'algorithme retourne le tableau trié du plus petit au plus grand. Traduire cet algorithme en programme Pascal et en Fortran.

Exercice 26

Soient deux vecteurs entiers $V1$ et $V2$ d'ordre 5. Écrire un algorithme qui calcule et affiche la somme des éléments de ces deux vecteurs. Traduire cet algorithme en programme Pascal et en Fortran.

Exercice 27

Écrire un algorithme permettant de trier les éléments d'un vecteur V par ordre croissant. Traduire cet algorithme en programme Pascal et en Fortran.

Exercices sur les Matrices

Exercice 28

Considérons deux matrices $M1$ et $M2$ de même ordre ($N \times P$) dont les éléments sont de type entier. Écrire un algorithme qui calcule et affiche la matrice $M = M1 + M2$. Traduire cet algorithme en programme Pascal et en Fortran.

Exercice 29

Écrire un algorithme qui effectue la lecture d'une matrice carrée $A(n,n)$ et affiche sa matrice transposée A^t . Traduire cet algorithme en programme Pascal et en Fortran.

Exercice 30

Écrire un algorithme qui effectue la lecture d'une matrice carrée $A(n,n)$ et affiche sa trace. La trace(A) = $\sum_i^n A(i, i)$, c'est à dire la somme des éléments de la diagonale de la matrice A). Traduire cet algorithme en programme Pascal et en Fortran.

Liste de Références

AMOUR R., FERMOUS R., AIT OUARABI M., BENZEKKA M., YOUNSI S., 2018. Algorithmique : Cours et Exercices en Programmation Pascal. USTHB, 70 p.

AMIROUCHE L. & KERIREM S., 2009." Introduction à l'Algorithmique".

GRAINE S., 2001 "Algorithmique et structures de donnée", les éditions l'Abeille.

LOUISNARD O., LETOURNEAU J.J., GABORIT P., 2000. Initiation au Fortran. Ecole des Mines d'Albi, 81p.

MORVANT E., 2009. Algorithme et programmation en Pascal. Cours, Saint-Louis Prépa ECE 1, 19p.

SALOTTI J. M., 1998. Cours et exercices corrigés en Pascal. Universités UFR SM / Université Bordeaux 2, 71p.

Thiel E., 2004. Algorithmes et programmation en Pascal. Faculté des Sciences de Luminy, 62p.

Solution des exercices

Exercice 1

- a) $A = 4$ et $B = 3$
 b) Il permute des valeurs des variables A et B.

Exercice 2

$A = 13$; $B = 40$; $C = 13$

Exercice 3

$A = 40$.

Exercice 4

$B = \text{Faux}$.

Exercice 5

A	5			
B		2		
C			10	
D				14

Cet algorithme calcul la surface **C** et le périmètre **D** d'un rectangle ayant une largeur **A** et une longueur **B**.

Exercice 6

A	6		
R		3	
S			28.6

Cet algorithme calcul la surface **S** d'un disque ayant comme diamètre **A**.

Exercice 7

Algorithme	Pascal	Fortran
Algorithme RESTDIV ; Var A,B,R: Entiers ; Début Lire (A,B) ; $R \leftarrow A \text{ Mod } B$; Ecrire(R) ; Fin.	Program RESTDIV ; Var A,B,R: Integer ; Begin Read (A,B) ; $R := A \text{ Mod } B$; Write(R) ; End.	Program RESTDIV Integer A,B,R Read*,A,B $R = \text{Mod} (A,B)$ Write(*,*) R End

Exercice 8

Algorithme	Pascal	Fortran
Algorithme Imagef ; Var X,F,: Réels ; Début Lire (x) ; $F \leftarrow 2 * X * X + X + 7$; Ecrire(F) ; Fin.	Program Imagef ; Var X,F: Real ; Begin Read (X) ; $F := 2 * X * X + X + 7$; Write(F) ; End.	Program Imagef Real X,F Read*,X $F = 2 * X * X + X + 7$ Write(*,*) F End

Exercice 9

a)

Algorithme	Pascal	Fortran
Algorithme ConverHMS ; Var S, H, M: Entiers; Début Lire(S); $H \leftarrow S \text{ div } 3600$; $S \leftarrow S \text{ mod } 3600$; $M \leftarrow S \text{ div } 60$; $S \leftarrow S \text{ mod } 60$; Ecrire(H,M,S); End.	Program ConverHMS ; Var S, H, M: Integer; Begin Read(S); $H := S \text{ div } 3600$; $S := S \text{ mod } 3600$; $M := S \text{ div } 60$; $S := S \text{ mod } 60$; Write(H,M,S); End.	Algorithme ConverHMS Integer S, H, M Read*,S $H = \text{Div}(S,3600)$ $S = S \text{ mod}(S,3600)$ $M = S \text{ div}(S,60)$ $S = S \text{ mod}(S,60)$ Write(*,*) H,M,S End

b)

Algorithme	Pascal	Fortran
Algorithme ConverJH ; Var J, H: entiers; Début Lire(H); $J \leftarrow H \text{ div } 24$; Ecrire (J); End.	Program ConverJH ; Var J, H: Integer; Begin Read(H); $H := H \text{ div } 24$; Write(J); End.	Algorithme ConverJH Integer J, H Read*,H $J = \text{Div}(H,24)$ Write(*,*) J End

Exercice 10

Algorithme	Pascal	Fortran
Algorithme Moypond ; Var A, B,C : Entiers; Var Moy : Réel ; Début Lire (A,B,C); Moy \leftarrow (A+B+C)/3; Ecrire (Moy); End.	Program Moypond ; Var A, B,C: Integer; Var Moy: Reel; Begin Read(H); Moy := (A+B+C)/3; Write(Moy); End.	Algorithme Moypond Integer A, B,C Real Moy Read*,A,B,C Moy = (A+B+C)/3 Write(*,*) Moy End

Exercices sur les structures conditionnelles

Exercice 11

Algorithme	Pascal	Fortran
Algorithme TypeAge ; Var Age : Entiers; Début Ecrire(" Quel âge avez-vous ?"); Lire(Age); Si (Age >= 18) Alors Ecrire (" Vous êtes majeur") ; Sinon Ecrire (" Vous êtes mineur") ; Fsi End.	Program TypeAge ; Var Age: Integer; Begin Write("Quel âge avez-vous ?"); read(Age); If (Age >= 18) Then Write (" Vous êtes majeur") Else Write (" Vous êtes mineur") ; End.	Program TypeAge Integer Age Write(*,*) 'Quel âge avez-vous ?' Read*, Age If (Age >= 18) Then Write(*,*) ' Vous êtes majeur' Else Write(*,*) ` Vous êtes mineur ' Endif End

Exercice 12

Solution avec Si

Algorithme	Pascal	Fortran
Algorithme TypeClimat ; Var Temp : Real; Début Ecrire (" Combien est la température Aujourd'hui ?"); Lire(Temp) ; Si (Temp > 30) Alors Ecrire (" Alerte à la canicule") ; Sinon Si (Temp < 0) Alors Ecrire (" Il gèle") ; Fsi Fsi End.	Program TypeClimat ; Var Temp : Real ; Begin Write(" Combien est la température Aujourd'hui?"); Read(Temp) ; If (Temp > 30) Then Write (" Alerte à la canicule ") Else Begin If (Temp < 0) Then Write (" Il gèle ") ; End End.	Program TypeClimat Real Temp Write(*,*) ' Combien est la température Aujourd'hui?' Read*, Temp If (Temp > 30) Then Write(*,*) 'Alerte à la canicule ' Else If (Temp < 0) Then Write(*,*) 'Il gèle' Endif Endif End

Solution avec Select

Algorithme	Pascal	Fortran
Algorithme TypeClimat ; Var Temp : Real; Début Ecrire (" Combien est la température Aujourd'hui ?"); Lire(Temp); Selon (Temp) Faire 30..60 : Ecrire (" Alerte à la canicule"); -50..0 : Ecrire (" Il gèle"); Fselon End.	Program TypeClimat ; Var Temp : Real ; Begin Write(" Combien est la température Aujourd'hui?"); Read(Temp); Case (Temp) of 30..60 : Write (" Alerte à la canicule "); -50..0 : Write (" Il gèle ") ; End; End.	Program TypeClimat Real Temp Write(*,*) 'Combien est la température Aujourd'hui?' Read*, Temp Select Case (Temp) Case(30:) Write(*,*)' Alerte à la canicule' Case(:-1) Write(*,*) 'Il gèle' End Select End

Exercice 13

Algorithme	Pascal	Fortran
Algorithme Valabsolue ; Var A : Real; Début Ecrire (" Donnez un nombre"); Lire(A); Si (A < 0) Alors A ← -A; Fsi Ecrire (" La valeur Absolue est", A) ; End.	Program Valabsolue ; Var A : Real ; Begin Write(" Donnez un nombre ?"); Read(A); If (A < 0) Then A := -A; Write (" La valeur Absolue est ",A); End.	Program Valabsolue Real A Write(*,*) ' Donnez un nombre ?' Read*, A If (A < 0) Then A = -A Endif Write(*,*) 'La valeur Absolue est ',A End

Exercice 14

Algorithme	Pascal	Fortran
Algorithme Valb ; Var a,b : Real; Début Ecrire (" Donnez un nombre"); Lire(a); Si (a ≠ 1) Alors b ← 5/(a-1); Fsi Ecrire (b) ; End.	Program Valb ; Var a,b : Real ; Begin Write(" Donnez un nombre ?"); Read(a); If (a < > 1) Then b := 5/(a-1); Write (b); End.	Program Valb Real a,b Write(*,*) ' Donnez un nombre ?' Read*, a If (a <>1) Then b = 5/(a-1) Endif Write(*,*) b End

Exercice 15

Algorithme	Pascal	Fortran
Algorithme Parite ; Var N : Entier; Début Ecrire (" Donnez un nombre"); Lire(N); Si (N Mod 2 = 0) Alors Ecrire ("Nombre Pair") ; Sinon Ecrire ("Nombre Impair") ; Fsi End.	Program Parite ; Var N : Integer ; Begin Write(" Donnez un nombre"); Read(N); If (N Mod 2 == 0) Then Write ('Nombre Pair ') Else Write ('Nombre Impair '); End.	Program Parite Integer N Write(*,*) ` Donnez un nombre` Read*, N If (Mod (N ,2) = 0) Then Write (*,*) ` Nombre Pair` Else Write(*,*) ` Nombre Impair` Endif End

Exercice 16

Algorithme	Pascal	Fortran
Algorithme TriangleRect ; Var A,B,C,X : Entier; Début Ecrire ("Introduire les dimensions du triangle"); Lire(A,B,C); X ← SQRT(A**2 + B**2) ; Si (C = X) Alors Ecrire ("Le Triangle est Rectangle") ; Fsi End.	Program TriangleRect ; Var A,B,C,X : Integer ; Begin Write(' Introduire les dimensions du triangle'); Read (A,B,C); X := SQRT (A*A + B*B); If (C == X) Then Write ('Le Triangle est Rectangle ') End.	Program TriangleRect Integer A,B,C,X Write(*,*) ' Introduire les dimensions du triangle' Read*, A,B,C X = SQRT (A*A + B*B); If (C = X) Then Write(*,*) ` Le Triangle est Rectangle` Endif End

Exercices sur boucles

Exercice 17

Algorithme	Pascal	Fortran
Algorithme Somme ; Var S,i : Entier; Début S ← 0 ; Pour i allant de 0 à 50 Faire S ← S +1 ; Fpour Ecrire ("La Somme est :)") ; Ecrire (S) ; End.	Program Somme ; Var S,i : Integer ; Begin S := 0 ; For i := 0 To 50 Do S := S +1 ; Write ('La Somme est :)') ; Write (S) ; End.	Program Somme Integer S,i S = 0 For i = 0, 50 S = S +1 Enddo Write (*,*) 'La Somme est :' Write (*,*) S End

Exercice 18

Algorithme	Pascal	Fortran
Algorithme Produit ; Var P,i : Entier; Début P ← 1 ; Pour i allant de 1 à 10 Faire P ← P * i ; Fpour Ecrire ("Le Produit est :") ; Ecrire (P) ; End.	Program Produit ; Var P,i : Integer ; Begin P := 1 ; For i := 1 To 10 Do P := P * i ; Write ('Le Produit est ',P) ; End.	Program Produit Integer P,i P = 1 For i = 1, 10 P = P * i Enddo Write (*,*) 'Le Produit est ',P End

Exercice 19

Algorithme	Pascal	Fortran
Algorithme SOMIMPAIR ; Var S,i : Entier; Début S ← 0 ; Pour i allant de 1 à 20 Faire Si ((i mod 2) ≠ 0) Then S ← S + i ; Fsi Fpour Ecrire ("La Somme des Nb impairs est :",S) ; End.	Program SOMIMPAIR ; Var S,i : Integer ; Begin S := 0 ; For i := 1 To 20 Do Begin If (i Mod 2) <> 0) Then S := S + i ; End ; Write ('La Somme des Nb impairs est ',S) ; End.	Program SOMIMPAIR Integer S,i S = 0 For i = 1, 20 If (Mod(i, 2) <> 0) Then S = S + i Endif Enddo Write (*,*) ' La Somme des Nb impairs est ',S End

Exercice 20

Algorithme	Pascal	Fortran
Algorithme SOMPAIR ; Var S,i : Entier; Début S ← 0 ; Pour i allant de 1 à 20 Faire Si ((i mod 2) = 0) Then S ← S + i ; Fsi Fpour Ecrire ("La Somme des Nb pairs est :",S) ; End.	Program SOMPAIR ; Var S,i : Integer ; Begin S := 0 ; For i := 1 To 20 Do Begin If (i Mod 2) == 0) Then S := S + i ; End ; Write ('La Somme des Nb pairs est ',S) ; End.	Program SOMPAIR Integer S,i S = 0 For i = 1, 20 If (Mod(i,2) = 0) Then S = S + i Endif Enddo Write (*,*) ' La Somme des Nb pairs est ',S End

Exercice 21

Algorithme	Pascal	Fortran
Algorithme PGCD ; Var A,B,P : Entier; Début Lire(A,B) ; Tant que (A≠B) Faire Si (A >B) Alors A ← A - B ; Sinon B ← B - A ; Fsi FTantque P ← A ; Ecrire ("Le PGCD est:",P) ; End.	Program PGCD ; Var A,B,P : Integer ; Begin Read (A,B); While (A<>B) do Begin If (A >B) Then A := A - B Else B := B - A ; End ; P := A ; Write ('Le PGCD est :',P) ; End.	Program PGCD Integer A,B,P Read *,A,B Do While (A <> B) If (A >B) Then A = A - B Else B = B - A Endif Enddo P = A Write (*,*) ' Le PGCD est :',P End

Exercice 22

Solution 1 (En utilisant la boucle POUR)

Algorithme	Pascal	Fortran
Program factorielle ; var i, fact ,N: entier ; Début Ecrire (" Donner une valeur de N"); Lire (N); fact ← 1; Pour i Allant de 1 à N Faire fact ← fact *i; Fpour ; Ecrire ('N ! =', fact) ; Fin.	Program factorielle ; var i, fact, N: integer ; begin writeln (' Donner une valeur de N '); readln (N); fact :=1; for i:=1 to N do begin fact := fact *i; end ; writeln ('N ! =', fact); End.	Program factorielle integer i, fact, N writel(*,*) 'Donner une valeur de N ' Read (*,*) N fact =1 Do i= 1,N fact = fact *i Enddo writel (*,*) ' N ! =', fact End

Solution 2 (En utilisant la boucle TantQue)

Algorithme	Pascal	Fortran
Program factorielle ; var i, fact ,N: entier ; Début Ecrire (" Donner une valeur de N"); Lire (N); fact ← 1; i ← 1; Tantque (i <= N) Faire fact ← fact *i; i ← i+1; Ftantque ; Ecrire ('N ! =', fact); Fin.	Program factorielle ; var i, fact, N: integer ; begin writeln (' Donner une valeur pour N '); readln (N); fact :=1; i := 1; While (i<= N) do begin fact := fact *i; i:=i+1; end ; writeln ('N ! =', fact); end.	Program factorielle integer i, fact, N writel(*,*) 'Donner une valeur pour N ' read (*,*) N fact =1 i = 1 Do While (i<= N) fact = fact * i i =i+1 Enddo writel (*,*) ' N ! =', fact End

Solution 3 (En utilisant la boucle REPETER)

Algorithme	Pascal	Fortran
Program factorielle ; var i,N: entier ; fact : réel ; Début Ecrire (" Donner une valeur de N"); Lire (N); fact ← 1; i ← 1; Repeter fact ← fact *i; i ← i+1; Jusqu'à (i>N) ; Ecrire ('N ! =', fact); Fin.	Program factorielle; var i,N: integer ; fact : real ; begin writeln(' Donner une valeur pour N '); readln (N); fact :=1; i := 1; Repeat fact := fact * i; i:=i+1; Untill (i>N) ; writeln ('N ! =', fact); readln End.	Il n'existe pas d'instruction REPETER. On utilise donc une syntaxe alternative : Program factorielle integer i, N real fact write(*,*)'Donner une valeur pour N ' read(*,*) N fact =1 i = 1 6: If (i >N) Goto 5 fact = fact * i i = i+1 Goto 6 5: Write(*,*) 'N ! =', fact End

Exercice 23

Solution 1 (En utilisant la boucle POUR)

Algorithme	Pascal	Fortran
Program puissance; var i, x,N: entier ; Var Puiss : réel ; Début Ecrire ("choisir une valeur pour x = "); Lire (x); Ecrire (" choisir une valeur pour N "); Lire (N); Puiss ← 1; Pour i Allant de 1 à N Faire Puiss ← Puiss *x; Fpour ; Ecrire ("x a la puissance N = ", Puiss); Fin.	Program puissance ; var i,N,x: integer ; Puiss : real ; begin writeln (' choisir une valeur pour x = '); readln (x); writeln (' choisir une valeur pour N = '); Readln (N); Puiss :=1; for i:=1 to N do begin Puiss := Puiss *x; end ; writeln ('x a la puissance N = ', Puiss); end.	Program puissance integer i, x, N Real Puiss writel(*,*) 'Donner une valeur de x ` read (*,*) x writel(*,*) ` Donner une valeur de N ` read (*,*) N Puiss =1 Do i= 1,N Puiss = Puiss * x Enddo write(*,*) `x a la puissance N =', Puiss End

Solution 2 (En utilisant la boucle TantQue)

Algorithme	Pascal	Fortran
Program puissance; var i, x,N: entier ; Var Puiss : réel ; Début Ecrire ("choisir une valeur pour x "); Lire (x); Ecrire (" choisir une valeur pour N "); Lire (N); Puiss ← 1; i ← 1; Tanque (i <= N) Faire Puiss ← Puiss * x; i ← i+1; Ftantque ; Ecrire ("x a la puissance N =", Puiss); Fin.	Program puissance; var i, x, N: integer ; Var Puiss : real ; begin writeln ('choisir une valeur pour x'); readln (x); writeln ('Donner une valeur pour N '); readln (N); Puiss :=1; i := 1; While (i<= N) do begin Puiss:= Puiss * x; i:=i+1; end ; writeln ('x a la puissance N = ', Puiss); End.	Program puissance Integer i, x, N Real Puiss write(*,*) 'Donner une valeur pour x ` Read (*,*) x write(*,*) 'Donner une valeur pour N ` read (*,*) N Puiss =1 i = 1 Do While (i <= N) Puiss = Puiss * x i =i +1 Enddo write(*,*) `x a la puissance N =', Puiss End

Solution 3 (En utilisant la boucle REPETER)

Algorithme	Pascal	Fortran
Program puissance; var i, x,N: entier ; Var Puiss : réel ; Début Ecrire ("choisir une valeur pour x "); Lire (x); Ecrire (" choisir une valeur pour N "); Lire (N); Puiss ← 1; i ← 1 ; Repeter Puiss ← Puiss *x; i ← i+1; Jusqu'à (i > N) ; Ecrire ("x a la puissance N=", Puiss); Fin.	Program puissance; var i, x, N: integer ; Var Puiss : real ; begin write('choisir une valeur pour x'); Readln (x); write('Donner une valeur pour N '); readln (N); Puiss :=1; i := 1; Repeat Puiss:= Puiss * i; i:=i+1; Untill(i>N) ; write('x a la puissance N = ',Puiss); End.	Il n'existe pas d'instruction REPETER. On utilise donc une syntaxe alternative : Program puissance Integer i, x, N Real Puiss write(*,*) 'Donner une valeur pour x ' Read (*,*) x write(*,*) 'Donner une valeur pour N' read (*,*) N Puiss =1 i = 1 6: If (i > N) Goto 5 Puiss = Puiss * x i =i +1 Goto 6 5 :write(*,*) 'x a la puissance N =', Puiss End

Exercices sur les Tableaux

Exercice 24

Algorithme	Pascal	Fortran
Algorithme IndiceX ; Var V : Tableau V(N) de réel ; Var I, N, X, Indice : entier; Début Lire (N,X) ; Pour I allant de 1 à N Faire Si (V(I)=X) alors Indice = I ; Fin Si Fin Pour Ecrire ("l'indice de X dans V est :", Indice) ; Fin.	Program IndiceX ; Var V : Array V[1..N] of real ; Var I, N, X, Indice : Integer; Begin Read (N,X) ; For I := 1 to N Do Begin If (V[I]==X) Then Indice := I ; End; Write('indice de X dans V est :', Indice) ; End.	Program IndiceX Real, Dimension V(100) Integer, I,Indice,N,X Read(*,*) N,X For I = 1, N If (V(I) = X) Then Indice = I Endif Enddo Write(*,*) 'indice de X dans V est :', Indice End

Exercice 25

Algorithme	Pascal	Fortran
Algorithme InvTableau ; Var i, X, N : entier; Var T : Tableau (10) d'entiers ; Début Lire (N) ; Pour i de 1 à N Lire (T(i)) ; Fin Pour i Allant de 1 à N/2 Faire X ← T(i) ; T(i) ← T (N -i +1) ; T (N -i +1) ← X ; Fpour Fin.	Program IndiceX ; Var T : Array[1..10] of Integer; Var i, N, X: Integer; Begin Read (N) ; For i := 1 to N Do Read(T[i]); For i := 1 to N/2 do Begin X := T[i] ; T[i] := T [N -i +1] ; T [N -i +1] := X ; End End.	Program IndiceX Integer, Dimension T(10) Integer, i, ,N,X Read(*,*) N Do i = 1, N Read(*,*)T(i) Enddo Do i = 1, N/2 X = T(i) T(i) = T (N -i +1) T (N -i +1) = X Enddo End

Exercice 26

Algorithme	Pascal	Fortran
Algorithme Som2vect ; Var V1 ,V2 ,V3: Tableau (10) de réel ; Var i: Entier ; Début Pour i Allant de 1 à 5 Faire Lire (V1(i)); Fpour Pour i Allant de 1 à 5 Faire Lire (V2(i)); Fpour Pour i Allant de 1 à 5 Faire V3(i) ← V1(i) + V2(i); Fpour Pour i Allant de 1 à 5 Faire Ecrire (V3(i)); Fpour Fin.	Program Som2vect ; Var V1 ,V2 ,V3: array [1..10] of real ; Var i: integer ; begin for i:=1 to 5 do Readln (V1[i]); for i:=1 to 5 do Readln (V2[i]); for i:=1 to 5 do V3[i]:= V1[i]+ V2[i]; for i := 1 to 5 do Write ('V3[' ,i, ']= ',V3[i]); Readln End.	Program Som2vect Real, Dimension V1(10),V2(10),V3(10) Integer, i Do i=1, 5 Read(*,*) V1(i) Enddo Do i =1, 5 Read(*,*) V2(i) Enddo Do i=1, 5 V3(i) = V1(i) + V2(i) Enddo Do i=1, 5 Write(*,*) V3(i) End

Exercice 27

Algorithme	Pascal	Fortran
Algorithme TriVectCrois ; Var V: Tableau (N) d'Entier ; Var i,j,N,Min: Entier ; Début Lire (N) Ecrire(" Introduire les éléments du vecteur ") ; Pour i allant de 1 à N faire Lire(V(i)) ; Fpour Pour i allant de 1 à N-1 faire Min ← V(i) ; Pour j allant de (i+1) à N faire Si (V(j) < Min) alors Min ← V(j) ; Fsi Fpour V(i) ← Min ; Fpour Pour i allant de 1 à N faire Ecrire(V(i)) ; Fpour Fin.	Program TriVectCrois ; Var V: array [1..100] of Integer ; Var i,j,N,Min: integer ; begin Read (N) Write (' Introduire les éléments du vecteur ') ; For i := 1 to N Do Read (V[i]) ; For i := 1 to N-1 Do Begin Min := V[i] ; Fo j := (i+1) to N Do Begin If (V[j] < Min) Then Min := V[j] ; End ; V(i) := Min ; End ; For i := 1 to N Do Write (V[i]) ; End.	Program TriVectCrois Integer, Dimension V(100) Integer i,j ,N,Min Read (*,*) N Write (*,*) ` Introduire les éléments du vecteur ` Do i=1, N Read(*,*) V(i) Enddo Do i = 1, N-1 Min = V(i) Do j = (i+1), N If (V(j) < Min) Then Min = V(j) Endif Enddo V(i) = Min Enddo Do I =1, N Write(*,*) V (i) End

Exercices sur les Matrices

Exercice 28

Algorithme	Pascal	Fortran
Algorithme somdeuxmat; Const N=3, P=4 ; Var M1, M2, M: Tableau (N,P) de Réel ; Var i, j,N,P: Entier ; Début Ecrire ("Introduire les éléments des deux matrices M1 et M2 ") ; Pour i Allant de 1 à N Faire Pour j Allant de 1 à P Faire Lire (M1(i,j)) ; Fpour Fpour Pour i Allant de 1 à N Faire Pour j Allant de 1 à P Faire Lire (M2(i,j)) ; Fpour Fpour	program somdeuxmat ; Const N=3; P=4; Var M1 ,M2 ,M: array [N,P] of real ; Var i,j,N,P: integer ; begin Write (' Introduire les éléments des deux matrices M1 et M2 ') ; For i:=1 to N do For j:=1 to P do Readln (M1[i,j]); For i:=1 to N do For j:=1 to P do readln (M2[i,j]); For i:=1 to N do For j:=1 to P do	Program somdeuxmat Parameter N=3 , P=4 Real, Dimension M1(100),M2(100),M (100) Integer i,N,P Write (*,*) ` Introduire les éléments des deux matrices M1 et M2 ` Do i=1, N Do j=1, P Read(*,*) M1(i,j) Enddo Do i=1, N Do j=1, P Read(*,*) M2(i,j) Enddo Do i=1, N

<p>Fpour Pour i Allant de 1 à N Faire Pour j Allant de 1 à P Faire M(i,j) ← M1(i,j)+ M2(i,j); Fpour Fpour Pour i Allant de 1 à N Faire Pour j Allant de 1 à P Faire Ecrire (M(i,j)); Fpour Fpour Fin.</p>	<p>M[i,j]:= M1[i,j]+ M2[i,j]; For i:=1 to N do For j:=1 to P do writeln (M[i,j]); End.</p>	<p>Do j=1, P M(i,j)= M1(i,j)+ M2(i,j) Enddo Enddo Do i=1, N Do j=1, P Write(*,*) M(i,j) Enddo Enddo End</p>
--	--	--

Exercice 29

Algorithme	Pascal	Fortran
<p>Algorithme MatTransp; Var AT, A: Tableau (n,n) de Réel ; Var i, j,n: Entier ; Début Ecrire('Donner la taille de la matrice A :') ; Lire(n) ; Ecrire (' Introduire les éléments de la matrice A '); Pour i Allant de 1 à n faire Pour j Allant de 1 à n faire Lire (A(i,j)) ; Fpour Fpour Pour i Allant de 1 à n faire Pour j Allant de 1 à n faire AT(i,j)← A(j,i); Fpour Fpour Ecrire (' Affichage des éléments de la matrice transposée TA '); Pour i Allant de 1 à n faire Pour j Allant de 1 à n faire Ecrire (AT(i,j)); Fpour Fpour Fin.</p>	<p>program MatTransp ; var A: array [1..n ,1.. n] of real ; AT: array [1..n ,1.. n] of real ; i,j,n: integer ; begin write('Donner la taille de la matrice A :') ; read(n) ; writeln ('Introduire les éléments de la matrice A '); for i:=1 to n do for j:=1 to n do readln (A[i,j]); for i:=1 to n do for j:=1 to n do AT [j,i]:= A[i,j]; writeln (' Affichage des éléments de la matrice transposée AT '); for i:=1 to p do for j:=1 to n do writeln (AT [i,j]); End.</p>	<p>Program MatTransp Integer i,j,n Dimension A(100,100), AT (100,100) Write(*,*)'Donner la taille de la matrice A :' read(*,*) n Write(*,*)'Donner les éléments de la matrice A ' Do i=1,n Do j=1,n read(*,*) A(i,j) Enddo Enddo Do i=1,n Do j=1,n AT (i,j)=A(j,i) Write(*,*)AT (i,j) Enddo Enddo write(*,*) ` Affichage des éléments de la matrice transposée AT ` Do i=1,n Do j=1,n Write(*,*)AT (i,j) Enddo Enddo End</p>

Exercice 30

Algorithme	Pascal	Fortran
Algorithme Tracemat; Var A: Tableau (n ,n) d'Entiers; Var i, j,n, TraceA : Entier ; Début Ecrire('Donner la taille de la matrice A ') ; Lire(n) Ecrire ('Donner les éléments de la matrice A'), Pour i Allant de 1 à n Faire Pour j Allant de 1 à n Faire Lire(A(i,j)); Fpour Fpour TraceA ← 0 ; Pour i Allant de 1 à n Faire TraceA ← TraceA + A(i,i); Fpour Ecrire ("La trace de la matrice A est:", TraceA) ; Fin.	Program Tracemat; var A: array [1..n ,1.. n] of Integer ; Var i,j,n,TraceA: integer ; begin write('Donner la taille de la matrice A :') ; Read(n) ; writeln (' Donner les éléments de la matrice A'); for i:=1 to n do for j:=1 to n do readln (A[i,j]); TraceA :=0; for i:=1 to n do TraceA := TraceA+ A[i,i]; writeln (' La trace de la matrice A est:",TraceA); End.	Program Tracemat Integer i,j,n,TraceA Dimension A(100,100) Write(*,*)'Donner la taille de la matrice A ' Read(*,*) n Write(*,*) 'Donner les éléments de la matrice A' Do i=1,n Do j=1,n read(*,*) A(i,j) Enddo Enddo TraceA = 0 Do i=1,n TraceA=TraceA +A(i,i) Enddo Write(*,*)'La trace de la matrice A est:', TraceA End