CHAPITRE 1 INITIATION A FORTRAN

I) Pourquoi le FORTRAN ?

FORTRAN est le langage le plus utilisé dans la communauté scientifique, bien que C/C++ se rapproche. Les raisons de cette longévité sont nombreuses:

- FORTRAN est un langage structuré, de syntaxe simple et facile à apprendre, qui a su évoluer.
- C'est un langage utilisé depuis plus de 40 ans : les programmes et bibliothèques de routines disponibles se chiffrent en millions de lignes. On trouve toujours le bout de code que l'on cherche dans une bibliothèque existante.
- C'est un langage universel: la plupart des scientifiques mathématiciens, physiciens, chimistes, géologues, biologistes, ingénieurs, etc. connaissent ou ont côtoyé FORTRAN.

II) Historique :

Le premier compilateur FORTRAN fut conçu et implémenté par John Backus (IBM Lab, San José California) en 1954. Il s'agissait du FORTRAN I. L'objectif de Backus était de concevoir un langage de programmation qui permettrait de réaliser des programmes en utilisant une syntaxe la plus proche possible de la syntaxe mathématicienne. Le terme FORTRAN traduit cette ambition, puisqu'il signifie FORmula TRANslation.

- 1978, FORTRAN 77, encore largement utilisé aujourd'hui .
- 1990 , FORTRAN 90, devient un langage moderne (format libre, programmation modulaire).
- 1997, la dernière version: le FORTRAN 95.

Le FORTRAN est donc un langage qui a un demi-siècle d'existence, mais qui reste un des plus utilisés en calcul scientifique .

III) Introduction

Qu'est-ce qu'un programme informatique ?

– Tout commence par un **algorithme** : suite d'opérations (instructions) décrivant les étapes de calcul menant au final.

exemple : écrire un algorithme qui calcule la moyenne d'un étudiant au module INF1.

Ecrire un algorithme qui calcule le volume d'une sphère.

– L'algorithme est traduit en Programme, écrit avec un langage de programmation évolué. (pour nous, le fortran)

Il existe beaucoup de langages de programmation : Pascal, C, C++, Java, Matlab, Delphi, Python,….

→ Un programme est une suite d'opérations sur des **variables**, et éventuellement des échanges de données entre utilisateur et ordinateur (entrées-sorties);

→ Les **variables** sont des noms correspondant à de la place mémoire dans l'ordinateur, utilisés pour stocker les données et les manipuler.

VI) Premiers pas en Fortran

4.1 Structure d'un programme fortran

Program nom-du-programme

- Bloc Déclaration variables
- Bloc Instructions

End program nom-du-programme

4.2 Les variables

– variables : noms (chaines de caractères alphanumériques) permettant de manipuler des données en mémoire.

– opération essentielle : l'affectation (symbole =) . Exemple : variable n (entier) :

```
n=2    !-- prend la valeur 2
n=n+1  !-- augmente de 1
```

Ce n'est pas une égalité mais une opération : la valeur de n stockée en mémoire est remplacée par elle-même plus 1.

– le bloc déclaration des variables sert à indiquer à la machine quel est le type de chaque variable utilisée afin de réserver la place nécessaire en mémoire. exemple :

déclaration	signification	ex. d'affectation
`integer :: n,i`	entier	`n=10`
`real :: x,y`	réel	`x=1.0e-5`
		`y=1.0`
`complex :: z`	complexe	`z=cmplx(1.0,2.5e3)`
`character :: c`	caractère	`c='d'`
`logical :: b`	booléen	`b=.true.`

– déclaration implicite : bien que ce ne soit pas exigé, il est prudent de déclarer **toutes** les variables. Pour éviter les oublis, placer l'instruction **implicit none** au début du bloc déclaration. Les oublis sont alors détectés à la compilation.

Program exemple

Implicit none

Integer :: n , i

Real : : x,y …….

- une variable est un objet dont la valeur peut être changée tout le long de l'exécution d'un programme.

Le nom est fixe, le type est fixe, la valeur change.

Ex : la note de l'examen nom : note type : réel valeur : 12 11.25 13.75

4.3) Les constantes :

- Fortran permet de définir une constante symbolique, un symbole auquel on attribue une valeur, qui ne pourra pas être modifiée tout le long du programme.
- Nom fixe, type fixe, valeur fixe.
- Déclaration de constante en Fortran :

<type> , parameter : : nom-const = valeur

Ex : integer , parameter : : n1= 50, n2 = 120
Real , parameter : : pi = 3.14

4.4) Les noms des variables et des constantes : Les identificateurs

– Ils désignent les différents objets manipulés par le programme.
– ils sont composés d'une suite de caractères alphanumériques, limitée à 31 caractères, parmi : lettres sans accents majuscules et minuscules, chiffres, caractère _ (souligné) ;
– le 1er caractère doit être une lettre ;
– les majuscules et minuscules ne sont pas différentiées ; sauf dans les chaines de caractères.
- Racine et racine : la même chose, 'Racine' est différente de 'racine'.

– exemple : → les identificateurs suivants sont valides :

```
constante_gaz
pi2
Rk54
```

→ les identificateurs suivants ne sont pas valides :

```
accentué
avec espace
Il_y_a_plus_de_trente_et_un_caracteres
_souligne_devant
1_chiffre_devant
nom#alphanumerique
```

4.5) Les commentaires : symbole ! (point d'exclamation)

– tout ce qui est à droite du symbole **!** est ignoré par le compilateur

– cela permet d'écrire des commentaires dans le programme, c.-à-d. du texte qui en explique le fonctionnement

– le **!** peut être placé n'importe où dans la ligne (au début, au milieu, à la fin), et à n'importe quel endroit du programme

4.6) Les entrées-sorties écran/clavier

– pour lire au clavier la valeur d'une variable, quel que soit son type

read∗,variable

– pour écrire à l'écran un message et/ou la valeur d'une variable

print∗, 'bonjour' **print**∗,x **print**∗, ' il est ' ,h,'heures'

– exemple :

```
integer :: n
character(len=15) :: pays
...
print*,'quel age avez-vous ?'
read*,n
print*,'vous avez ',n,'ans'
print*,'de quel pays etes vous ?'
read*,pays
print*,'vous venez de ',pays
```

- **Les instructions multiples :**

La fin de ligne est une séparation naturelle entre 2 instructions :

```
print *, 'donnez un nombre'
read *, n
```

Mais il est possible de placer plusieurs instructions sur une même ligne, en les séparant par des (point virgule ;)

```
print *, 'donnez un nombre' ;read *, n
```

V) les expressions :

Elles désignent le calcul arithmétique ou logique.
Ex : (2*x+6)/ (5*z-12) expression arithmétique,
(A et B ou C) ET (non D) expression logique

Une expression est formée de : Constantes, variables, opérateurs et de parenthèses.

Quatre opérateurs :
• Addition : +
• Soustraction : -
• Multiplication : *

• Division : / (Pas de division par zéro !)

Ex : $\frac{3x+2}{5x-3}$ = (3 * X) + 2 / (5 * X) - 3

Le résultat de la division entre deux entiers est un entier (partie entière du quotient exact).

```
5/2 = 2     7/5 = 1     -8/3 = -2
```

- Opérateur de puissance : se note **
  Ex : a^b se note a**b
- Ordre des opérations : 1) puissance 2) multiplication et division
  3) addition et soustraction

En cas d'égalité de priorité, commencer de gauche à droite.
En cas de doute, utiliser les parenthèses.
Les expressions logiques : comparaisons

- Comparaison des expressions numériques avec une valeur logique pour résultat

Ancienne notation	Nouvelle notation	signification
.LT.	<	inferieur à
.LE.	<=	inférieur ou égal à
.GT.	>	supérieur à
.GE.	>=	supérieur ou égal à
.EQ.	==	égal à
.NE.	/=	différent de

Attention !!!! x = y : en fortran c'est une affectation, on lit : x reçoit la valeur de y.

X = = y est une comparaison logique, qui a pour valeur vrai ou faux.

Les opérateurs logiques : .NOT. non, .AND. et , .OR. ou

- **les fonctions intrinséques :**

la racine carrée : sqrt( ), valeur absolue : abs ( ), exp, log, log10, sin, cos, tan, asin (arcsin), acos….

Exemples de programmes fortran simples :

ex1) Ecrire le programme fortran qui lit au clavier 2 entiers a et b, calcule leur somme et leur produit, et les affiche à l'écran.

```
 Program calcul
Implicit none
!déclaration des variables
integer : : a, b , som, prod
!lecture des données
read*, a
read*, b
som = a + b
prod = a*b
print *,som
print*, prod
end program calcul
```

LA TRACE D'EXEXUTION :

La trace d'exécution permet de suivre le bon déroulement d'un programme pas à pas, instruction après instruction afin de s'assurer de son bon fonctionnement. L'exécution est manuelle. La trace d'exécution donne les valeurs des variables après exécution de chaque instruction. La trace d'exécution est un tableau qui contient en lignes les instructions, et en colonnes les variables.

Exemple 1: faire la trace d'exécution du programme précédent.

a	b	som	prod	écran
12				12
12	5			5
12	5	17		
12	5	17	60	
12	5	17	60	17
12	5	17	60	60

Exemple 2 : soit le programme fortran suivant :

```
 program pgm
implicit none
integer :: x,y,z
x = 3
y = 7
z = x
x = y
z = z+1
print*, x, y, z
end program pgm
```

Questions :
1) faire la trace d'exécution de ce programme.
2) quelles sont les valeurs des variables x, y , z à la fin du programme ?

trace d'exécution :

x	y	Z	écran
3			
3	7		
3	7	3	
7	7	3	
7	7	4	
7	7	4	7 7 4

donc valeurs finales de x, y, z :
x = 7
y = 7
z = 4

EXERCICE 3 : écrire le programme fortran qui lit au clavier le nom d'un étudiant et ses notes de tp et d'examen, et calcule sa moyenne puis l'affiche avec son nom.

```
 Program calcul_moyenne
implicit none
real :: ntp, nexam, moy
character (len=20) :: nom
read*, nom
read*,ntp
read*,nexam
moy= (nexam*0.7)+(ntp*0.3)
print*,nom,moy
end program calcul_moyenne
```