CHAPITRE 2 **les instructions de contrôle**

Introduction

Jusqu'à maintenant, on a vu des programmes Fortran simples, avec des instructions séquentielles (print, read, affectation =). Mais, ces instructions sont-elles suffisantes pour exprimer tous les cas des différents problèmes ?
Exemple : écrire le programme qui calcule l'inverse d'un nombre réel x.

```
 PROGRAM inverse
IMPLICIT NONE
REAL :: x, inv

Read *,  x
Inv = 1/x
Print *, ' inverse de x = ', inv
END PROGRAM inverse
```

Si le nombre tapé est zéro, le programme aura une fin anormale, car on ne peut exécuter 1/x que si x est différent de zéro.

Donc on a besoin de nouvelles instructions , ce sont les instructions conditionnelles.

LES CONDITIONS :

Pour aborder les conditions, il faut d'abord parler d'expression logique. Il existe en FORTRAN un type de variable LOGICAL. Une variable de type LOGICAL peut prend deux valeurs: .VRAI. ou .FAUX. FORTRAN définit les expressions logiques suivantes, dans lesquelles x et y sont deux variables :

- x == y signifie x égal y
- x / = y signifie x non égal à y
- x > y signifie x supérieur à y
- x < y signifie x inférieur à y
- x >= y signifie x supérieur ou égal à y
- x < = y signifie x inférieur ou égal à y

Il définit aussi les opérateurs logiques:

- .OR. OU
- .AND. ET
- .NOT. NON

Une condition simple est une expression de la forme : v1 op v2
Où : v1 et v2 sont des variables , des constantes…
OP : est un opérateur de comparaison (> < = ….)
Ex : x< 0 y= x-2 z > = 3*x+9 …..
Son résultat est toujours de type logique (vrai ou faux).

Une condition composée est formée de 2 ou plusieurs conditions simples, reliées avec des opérateurs logiques (et, ou). Son résultat est toujours de type logique.
Ex : (x<0) et (z > = 10) (a > b) ou (z /= 2*n)……..

REMARQUE : en maths on écrit : $x \in [1..10]$ ou $1 \leqslant x \leqslant 10$
Mais ceci est interdit en programmation !!!! on écrit : (x> = 1) et (x< = 10)

2) L'instruction conditionnelle IF ……..THEN ENDIF (SI ……ALORS FINSI)

Munis de ces expressions logiques, on peut définir les instructions conditionnelles suivantes:

```
IF (condition) THEN

   <liste d'instructions>

   ENDIF
```

La signification de ce jeu d'instruction est simple: si l'expression condition est vraie, alors la liste d'instructions contenue entre le IF et le ENDIF est exécutée. Sinon, le programme continu et exécute l'instruction immédiatement suivante le ENDIF.

Elle peut être simplifiée si le bloc <liste d'instructions> contient une seule instruction.
Exemple : if (x < 2) then
Max = x
Endif

On peut l'écrire: if (x < 2) max = x (on supprime then et endif).
exemple 1 : écrire le programme FORTRAN qui lit un réel x au clavier , et calcule son inverse s'il n'est pas nul.

```
 PROGRAM inverse
implicit none
real :: x, inv
print*, ‘entrer le nombre x’
read*, x
if (x/=0) then
inv = 1/x
print*, ‘inverse de x = ‘, inv
endif
print*, ‘fin du programme’
ENDPROGRAM inverse
```

EXERCICE 2 : ecrire le programme FORTRAN qui calcule la racine carrée d’un nombre réel x, si ce nombre est positif ou nul.

```
 PROGRAM racine
implicit none
real :: x, rac
print*, ‘entrer le nombre x’
read*, x
if (x>= 0) then
rac = SQRT (x)
print*, ‘racine carrée  de x = ‘, rac
endif
print*, ‘fin du programme’
ENDPROGRAM racine
```

3) L’instruction IF……THEN …….ELSE……..ENDIF
(SI……..SINON ……FINSI)

Sa syntaxe est : SI (condition) alors
Bloc1…….
SINON
Bloc2……
FINSI

Explication : si la condition est vraie on exécute le bloc d’instructions 1, si la condition est fausse on exécute le bloc d’instructions 2. L’algorithme continue après finsi.

```
IF (condition) THEN

<liste d'instructions 1>

ELSE

<liste d'instructions 2>

ENDIF
```

Exemple 1 : réécrire l'algorithme inverse avec SI…….SINON. dans le cas où x = 0, afficher un message d'erreur.

```
SI ( x #  0)  alors
  Inv = 1/x
   PRINT* , ' inverse de x = ', inv
SINON
  Print*, 'erreur !!! le nombre tapé est nul '
FINSI
```

En FORTRAN :

```
      if   (x /= 0)   then

             Inv = 1/x

             Print*, 'inverse de x = ', inv

         Else

                Print*, 'erreur !!! le nombre tapé est nul'

   endif
```

Exercice 2 : reprendre l'exemple de la racine carrée d'un nombre x . Dans le cas où le nombre est nul, afficher un message d'erreur.

```
 PROGRAM racine
implicit none
real :: x, rac
print*, 'entrer le nombre x'
read*, x
if (x>= 0) then
rac = SQRT (x)
print*, 'racine carrée  de x = ', rac
else
print*, 'erreur !! le nombre tapé
est négatif'
 endif
print*, 'fin du programme'
ENDPROGRAM racine
```

3) les tests imbriqués:

les structures conditionnelles peuvent être imbriquées, c'est-à-dire incluses les unes dans les autres, dans le cas où il y'a plusieurs choix ou alternatives.

```
Cas général :
IF (condition1) THEN
<liste d'instructions 1>
ELSE IF (condition2) THEN
<liste d'instructions 2>
ELSE IF (condition3) THEN
<liste d'instructions 3>
ELSE
<liste d'instructions n>
ENDIF
```

Vous en devinez l'utilisation! Supposons que je doive exécuter un traitement différent selon qu'une variable x est égale à 1, 2, 3 ou différente de 1, 2, et 3. Je poserai:

```
IF (x == 1) THEN
<liste d'instructions 1>
ELSE IF (x == 2) THEN
<liste d'instructions 2>
ELSE IF
(x == 3) THEN
<liste d'instructions 3>
ELSE
C cas de x différent de 1,2 et 3
<liste d'instructions 4>
ENDIF
```

Si x = 1, seule la liste d'instructions 1 est exécutée. Si x=2, seule la liste d'instructions 2 est exécutée, etc...

C'est un jeu d'instruction très employé en traitement des données, mais qui peut générer des programmes un peu fouillis! Les commentaires sont indispensables.

Exemple : Ecrire le programme fortran qui lit la moyenne d'un étudiant et lui affiche l'observation : moyenne >= 10→ « admis »

9.5 <= Moyenne < 10→ « racheté »

Moyenne < 9.5 → « ajourné »

```
 Program observation
implicit none
real :: moy
print*,'donner la moyenne'
read*, moy
if  (moy>=10) then
  print*,'admis'
else
  if (moy>=9.5) then
    print*,'racheté'
  else
    print*,'ajourné'
endif
end program observation
```

5) L'instruction SELECT CASE :

Utile pour choisir entre différents blocs d'instructions , en fonction de la valeur d'une expression.

type :: i

ATTENTION !!! i doit être de type integer ou logical ou character.

SELECT CASE (i)

Case (0) ……..! S'exécute si i= 0

Case (:-1)……..! s'exécute si i<= -1.

Case (5 :10) ……. !s'exécute si 5<= i <= 10

Case default …….s'exécute dans tous les autres cas

End select

L'expression testée doit vérifier au plus 1 case.

Le CASE DEFAULT n'est pris en compte que si l'expression ne vérifie aucun CASE.

On note : case (:1) ------→ i <=1

Case (10 :) ------→ a >= 10

Case (5:10) -------→ 5<= I <= 10

Exercice 1 : programme qui lit le numéro d'un mois au clavier, et affiche le nombre de jours correspondant. Si le mois # {1…..12} écrire erreur.

```
program jours
implicit none
integer :: num
print*,' donner le numéro du mois'
read*, num
select case (num)
case (1,3,5,7,8,10,12)      print*,'ce mois a 31 jours'
case (2)                    print*, 'ce mois a 28 ou 29 jours'
case (4 ,6, 9,11)           print*,'ce mois a 30 jours'
case default                print*,' erreur , mois inexistant'
end program jours
```

EXERCICE 2 :

Ecrire le programme fortran qui lit 2 réels x et y , et un opérateur (+ - * /)
Selon l'opérateur tapé, exécuter l'opération correspondante (addition, soustraction, multiplication, division) et afficher le résultat.
si un autre caractère est tapé, afficher un message d'erreur.