CHAPITRE 3 **Les structures itératives - Les boucles**

I) INTRODUCTION

On a déjà écrit le programme qui calcule la moyenne d'un étudiant au module INF1, connaissant sa note de TP et sa note d'examen .
Pour un étudiant, les instructions sont :

bloc 1

```
read*, nom
read*,ntp
read*,nexam
moy = (ntp*0.3) + (nexam*0.7)
print*,moy
```

Si on veut calculer la moyenne pour 2 étudiants, on répète le bloc précédent 2 fois :

```
read*, nom
read*,ntp
read*,nexam
moy = (ntp*0.3) + (nexam*0.7)
print*,moy
```

```
read*, nom
read*,ntp
read*,nexam
moy = (ntp*0.3) + (nexam*0.7)
print*,moy
```

On est obligé de répéter le même bloc d'instructions pour chaque étudiant. Imaginez une section de 1000 étudiants !!!
La solution : on a besoin d'une structure qui permet d'écrire le bloc d'instructions qui se répéte une seule fois, et de l'exécuter plusieurs fois, en changeant uniquement les notes de tp et d'examen.
Cette structure existe et s'appelle **structure itérative** ou **boucle**.

II) LA BOUCLE DO : BOUCLE AVEC COMPTEUR

syntaxe : DO <var> = <début> , <fin> [,pas]

.
.
. instructions à répéter
.
.

ENDDO

var : variable de type entier, joue le rôle d'un compteur.
début , fin : valeur initiale et valeur finale du compteur var . Leurs valeurs sont connues à l'avance.
pas : c'est la valeur avec laquelle avance le compteur var. Il est optionnel, s'il n'est pas écrit il est égal à un.

Exercice 1 :

soit la boucle :
```
DO  i = 1 , 100 , 1
    print * , i

ENDDO
```
pour i allant de 1 à 100, avançant par pas de 1, afficher la valeur de i.
le résultat sera l'affichage à l'écran de : 1 , 2 , 3 , 4 , 5 , 6……., 99 , 100.
c'est-à-dire l'affichage de tous les entiers compris entre 1 et 100.

Exercice 2 :

soit la boucle :
```
DO  i = 1 , 100 , 2
    print * , i

ENDDO
```
pour i allant de 1 à 100, avançant par pas de 2, afficher la valeur de i.
le résultat sera l'affichage à l'écran de : 1, 3, 5, 7, 9, 11, 13 , ……..,99
c'est-à-dire l'affichage de tous les entiers impairs compris entre 1 et 99.

Exercice 3 :

Ecrire la boucle do qui permet de calculer la moyenne du module INF1 pour 500 étudiants.

```
DO  I = 1 , 500
   read*, nom
    read*,ntp
    read*,nexam
   moy = (ntp*0.3) + (nexam*0.7)
   print*,moy

END DO
```

Exercice 4 :

Ecrire la boucle DO qui calcule la somme : S = 1 + 3 + 5 + 7 + ……. + n
n est un entier lu au clavier.

```
 read *, n
S = 0
DO   i = 1 , n , 2
 S  =  S  +  i
ENDDO

print *, S
```

Exercice 5 :

Ecrire la boucle DO qui calcule le produit : $P = 2^2 * 4^2 * 6^2 * 8^2 * \ldots\ldots * n^2$
n est un entier lu au clavier.

```
 read *, n
P = 1
DO   i = 2 , n , 2
 P  =  P  *  ( i*i)
ENDDO

print *, P
```

III) Forme infinie de la boucle DO :
On ne sort de cette boucle qu'avec l'instruction **exit.**

Syntaxe :

```
DO
  .
  . instructions
  .
 if (condition) then
       exit
 endif
    .
    .  instructions
ENDDO
```

On entre dans la boucle do sans condition, on execute les instructions, on évalue la condition if, lorsqu'elle devient vraie on sort de la boucle. On ne sait pas à l'avance, combien de fois on va exécuter la boucle.

exemple 1 : écrire la boucle do qui lit des nombres entiers au clavier et calcule leur somme. on s'arrête lorsque l'utilisateur tape zéro.

```
DO
  print *, 'donner un entier'
  read *, n
   if (n == 0 ) then
      exit
   endif
 som = som + n
ENDDO
print*, som
```

Exemple 2 : Ecrire la boucle DO qui permet d'afficher les carrés des nombres tapés au clavier. on s'arrête lorsque l'utilisateur tape zéro.

```
DO
  print *, 'donner un entier'
  read *, n
   if (n == 0 ) then
      exit
   endif
 print *, (n*n)
ENDDO
```

IV) LA BOUCLE DO WHILE :

Elle exprime une répétition conditionnelle, la poursuite de la boucle est contrôlée par une condition.

Syntaxe :
```
<initialisation>
do while (condition)
 :
 :  instructions à répéter
  <incrémentaion>
enddo
```

Condition : est une expression logique.
Fonctionnement : le compteur est initialisé, on entre dans la boucle si la condition est vraie, on reste dans la boucle et on exécute la liste des instructions à répéter, tant que la condition est vraie. lorsque la condition devient fausse, on sort de la boucle, et on continue l'exécution après le enddo.

Ex1 :

```
DO  i = 1 , 100 , 1
    print * , i

ENDDO
```

→

```
        I = 1
DO WHILE ( i <= 100)
        print * , i
        I = I +1
   ENDDO
```

EX2 :

```
DO  i = 1 , 100 , 2
    print * , i

ENDDO
```

→

```
        I = 1
DO WHILE ( i <= 100)
        print * , i
        I = I +2
    ENDDO
```

EX3 :

-

```
I = 1
DO  WHILE  (I < = 100)
   read*, nom
    read*,ntp
    read*,nexam
   moy = (ntp*0.3) + (nexam*0.7)
   print*,moy
         I = I + 1
END DO
```

EXO 4 :

```
 read *, n
S = 0
I = 1
DO  WHILE ( i <= n )
 S  =  S  +  i
I = I + 2
ENDDO

print *, S
```

REMARQUE IMPORTANTE :

Que se passe t'il si on oublie l'instruction de mise à jour (i = i +1)… ??
ex :

```
        I = 1
DO WHILE ( i <= 100)
        print * , i

    ENDDO
```

I	ecran
1	1
1	1
1	1
1	1
1	1

le compteur i aura toujours la valeur 1, il ne changera jamais de valeur. la condition (i <= 1) sera toujours vraie, donc on continue à afficher 1….jusqu'à l'infini.
C'est une grave faute en programmation, on l'appelle **une boucle infinie.**