Les instructions répétitives

Dans plusieurs cas nous sommes obligés à répéter un ensemble de traitements plusieurs fois suivant certains paramètres.

Une structure d'itération permet de répéter plusieurs fois le même traitement. La répétition d'un bloc d'actions est définie par des actions spéciales que nous appelons des **boucles**.

La boucle Tantque

C'est une structure de traitements répétitifs conseillée quand le nombre d'itération est inconnu. Elle a la forme générale suivante :

Syntaxe :

```
Tantque Condition faire

ACTIONS

FinTantque;
```

Tant que la condition est vérifiée on exécutera le corps de la boucle, on s'arrêtera dès que la condition n'est plus vérifiée.
En Fortran :

```
Do while(condition)
   action
enddo
```

Exemple

```
Algorithme saisie
Var N : Entier
Debut
N ← 0
Ecrire ("Entrez un nombre entre 1 et 3")
TantQue (N < 1 ou N > 3) faire
    Lire (N)
    Si (N < 1 ou N > 3) Alors
      Ecrire ("Saisie erronée. Recommencez")
    FinSi
FinTantQue
Fin
```

```
Program saisie
Implicit none
Integer::N
N=0
Print*,' Entrez un nombre entre 1 et 3'
Do while (N < 1 ou N > 3)
Read*,N
If(N < 1 ou N > 3) then
```

```
Print*,' Saisie erronée. Recommencez'
Endif
Enddo
End program saisie
```

N.B: Les structures **TantQue** sont employées dans les situations où l'on doit procéder à un traitement systématique sur les éléments d'un ensemble dont on ne connaît pas d'avance la quantité, comme par exemple :

- le contrôle d'une saisie
- la gestion des tours d'un jeu (tant que la partie n'est pas finie, on recommence)
- la lecture des enregistrements d'un fichier de taille inconnue.

La boucle Répéter :

C'est une structure de traitement répétitif conseillée quand le nombre d'itération est inconnu. Elle a la forme générale suivante :

Syntaxe

Répéter
ACTIONS
Jusqu' à Condition;

La boucle **Répéter** permet de rentrer dans la boucle quel que soit la condition et répète l'exécution jusqu'à ce que la condition soit vérifiée

```
Algorithme saisie
Var N :Entier
Debut
N ← 0
Ecrire "Entrez un nombre entre 1 et 3"
Repeter
    Lire (N)
    Si N < 1 ou N > 3 Alors
      Ecrire "Saisie erronée. Recommencez"
    FinSi
  jusqu'à N > 1 et N < 3
Fin
```

Remarque:

- Quel que soit l'état de la condition, dans la boucle **répeter** on exécutera au moins une fois le corps de la boucle. Tandis que dans la boucle **tant que**, si la condition n'étant pas vérifiée au départ on n'exécutera pas la boucle.
- Dans le corps de la boucle **répéter** et **tant que**, il doit exister une variable - dite de contrôle ; qui sera modifiée pour faire évoluer l'état de la condition. En général, cette *variable de contrôle* doit être initialisée avant l'entrée dans la boucle.
- Le choix de la forme de la boucle doit répondre au problème à résoudre.

La boucle Pour :

C'est une structure de traitements répétitifs conseillée quand le nombre d'itération est connu à l'avance. Elle a la forme générale suivante :

Syntaxe :

Pour *compteur* ←Initial à Final **Pas** ValeurDuPas *faire*

ACTIONS

Fin Pour ;

Où :

- ***Initial*** : La valeur initiale
- ***Final***: La valeur finale
- ***ValeurDuPas*** : Le pas de progression

N.B: Les structures Pour sont employées dans les situations où l'on doit procéder à un traitement systématique sur les éléments d'un ensemble dont le programmeur connaît d'avance la quantité.

En Fortran

```
Do compteur= Initial , Final , ValeurDuPas
```

Exemple

```
Algorithme somme
Variable N,i :Entier
Debut
Lire(N)
S ← 0
Pour i ←1 à N faire
S←S+i
finpour;
Ecrire ('La somme S=',S)
 Fin
```

```
program somme
implicit none
integer ::N,i
read*,N
s=0
do i=1,N
s=s+i
enddo
```

```
print*,'la somme S=',N
```

Boucles imbriquées

Nous voulons construire la table des valeurs de z tel que: z= xy

Exemple

```
algorithme  xytab
var x,y,z:réel
Debut

Pour x=1,2 faire
     Pour y=1,4,0.5 faire
        Z=x/y
        Ecrire(x,y,z)
     Finpour
Finpour
Fin
```

```
Program xytab
Implicit none
Real ::x,y,z
Do =1,2
  Do y=1,4,0.5
     Z=x/y
     Print*,x,y,z
  Enddo
enddo
```

Execution

x	y	z
1.00000	**1.00000**	**1.00000**
1.00000	**1.50000**	**0.666667**
1.00000	**2.00000**	**0.500000**
1.00000	**2.50000**	**0.400000**
1.00000	**3.00000**	**0.333333**
1.00000	**3.50000**	**0.285714**
1.00000	**4.00000**	**0.250000**
2.00000	**1.00000**	**2.00000**
2.00000	**1.50000**	**1.33333**
2.00000	**2.00000**	**1.00000**
2.00000	**2.50000**	**0.800000**
2.00000	**3.00000**	**0.666667**
2.00000	**3.50000**	**0.571429**
2.00000	**4.00000**	**0.500000**

Press RETURN to close window...