Les boucles

Exemple pour la réflexion

On veut calculer et afficher la moyenne de 2 stagiaires à partir de 2 notes:

```
Var a,b,M: réel :
Début:

        Afficher("saisir 2 notes pour le stagiaire n 1 :")
        Lire(a,b)
        M← (a+b)/2
        Afficher(M)

        Afficher("saisir 2 notes pour le stagiaire n 2 :")
        Lire(a,b)
        M← (a+b)/2
        Afficher(M)
        ……
        ……
        Afficher("saisir 2 notes pour le stagiaire n 20:")
        Lire(a,b)
        M← (a+b)/2
        Afficher(M)
Fin
```

Analyse de l'exemple

- Dans l'exemple précédent , on peut remarquer qu'il y a un groupe de lignes qui se répètent exactement 20 fois,
- On peut facilement remarquer que cette écriture est lourde car il y a des répétitions.
- Question: est ce qu'il n'y a pas de remède pour éviter de faire ces répétitions?
- La réponse c'est oui, dans la programmation il existe ce qu'on appelle **les actions répétitives** ou bien les **boucles.**

C'est quoi une boucle?

- La boucle est l'une des structures de base de la programmation.
- Une boucle va nous permettre de **répéter des actions (ou instructions) plusieurs fois.**
- Donc une **action** qui se répète, on doit l'écrire comme suit:

```
Boucle
        Action qui se répète
Fin boucle
```

Types de boucles

- Il existe deux types de boucles:

1. Un type de boucle pour lequel **on connait à l'avance** le **nombre de répétitions** à effectuer. (boucle **POUR**)
2. Un type de boucle pour lequel on ne connait pas à l'avance le nombre de répétitions à effectuer , dans ce cas la boucle **dépend d'une condition** (boucles **TANT QUE** et **REPETER**)

La boucle TANT QUE

- **Syntaxe**:

```
TANT QUE (condition) FAIRE
   Action
FIN TANT QUE
```

- **Explication:**
  - ✓ **Tant que la condition est vrai**, l'ordinateur va exécuter ACTION
  - ✓ **Une fois la condition devient fausse**, l'ordinateur **sort de la boucle** et passe à la suite du programme.
  - ✓ ACTION peut être une action d'affectation, d'affichage, de lecture, action conditionnelle, ou même une action répétitive.

La boucle TANT QUE

- Exemple: on suppose qu'on veut que l'utilisateur saisisse une nombre négatif :

```
Var x: entier
Début
Afficher("veuillez saisir une valeur négative svp ")
Lire(x)

TANT QUE (x>=0) FAIRE

Afficher(" Erreur, veuillez refaire la saisie  ")
Lire(x)

FIN TANT QUE
Afficher (" le nombre que vous avez fournit est : ", x)
Fin
```

La boucle TANT QUE

Explication de l'exemple précédent:

- On demande à l'utilisateur un nombre obligatoirement négatif. S'il donne une valeur correcte dès la première saisie, on va l'afficher.

- Si l'utilisateur saisisse une valeur positive, dans ce cas on va pas l'accepter, donc on va lui demander de refaire la saisie TANT QUE la valeur est positive. une fois il fournit une valeur correcte c'est-à-dire négatives, on va sortir de la boucle et l'afficher.

La boucle répéter

- **Syntaxe**:

```
Répeter
  ACTION
Tant que (condition)
```

- **Explication**:

Après le mot répéter il n'y a pas de condition, donc **ACTION** va être d'abord exécutée puis on vérifie si une condition est vrai.

La boucle répéter

- Exemple:

```
Var x: entier
Debut

Repeter
        Afficher("saisir un nombre négatif svp")
        Lire(x)
Tant que (x>=0)
Afficher("la valeur que vous avez fournie est :", x)
Fin
```

La boucle répéter

Explication de l'exemple précédent:

- Après le mot répéter on a demandé à l'utilisateur de saisir une valeur négative.
- L'utilisateur va saisir une valeur, et va être vérifié : est ce qu'elle est positive? si c'est le cas l'utilisateur va refaire la saisie, s'il saisie une autre fois une valeur négative il va refaire la saisie une autre fois et ainsi de suite.
- Si l'utilisateur saisie une valeur négative, on va sortir de la boucle et afficher ce nombre.

La différence entre TANT QUE et REPETER

- La boucle REPETER est comme la boucle TANT QUE mais à l'envers.
- Dans la boucle TANT QUE il faut que la condition soit vrai pour entrer dans la boucle et exécuter les actions, par contre la boucle REPETER ne possède pas de condition d'entrée dans la boucle. donc les actions dans la boucle REPETER vont être exécuté au moins 1 fois (bien sûre si on ne fait pas intervenir d'autres actions)

Observation

- Regardons l'exemple avec **Tant que** et l'exemple avec **répéter**. il y'a une petite différence: dans le premier exemple on a fait deux affichages différents. Par contre dans l'exemple 2 on a fait un seul affichage. Si on veut faire exactement la même chose que l'exemple 1 on dois écrire:

```
Var x: entier
Debut
        Afficher(" veuillez saisir une valeur négative svp ")
        Lire(x)
SI (x>=0) ALORS
        Repeter
        Afficher(" Erreur, veuillez refaire la saisie  ")
        Lire(x)
        Tant que (x>=0)
FIN SI
Afficher("la valeur que vous avez fournie est :", x)
Fin
```

La boucle POUR

- Syntaxe:

```
POUR   variable ← Valinit    à  ValFin  ,[PAS]  faire

ACTION

FIN POUR
```

- Variable va servir pour compter le nombre de répétions réalisée. On l'appelle **compteur** de la boucle.
- **Valinit**: est une valeur initiale du compteur, c'est-à-dire la valeur avec laquelle il va commencer.
- **ValFin**: c'est la valeur finale du compteur
- **PAS:** c'est le pas avec lequel le compteur de la boucle va être **incrémenté** ou **décrémenté**, sa valeur par **défaut est 1.** on est pas obligé de l'écrire, sauf si sa valeur est différente de 1.

La boucle POUR

- Exemple:

```
Var i,N: entier
début
N← 5

POUR  i← 1  à  5  FAIRE

      N←N+3
      Afficher(N)

FIN POUR
Fin
```

La boucle POUR

Explication de l'exemple précédent:

- Ici le pas n'est indiqué donc sa valeur est 1. alors **i va être incrémenté de 1.**
- N est égale au début à 5, puis on va lui rajouter 3 dans chaque itération de la boucle pour.

```
i=1     N=8     //première itération
i=2     N=11    //deuxième itération
i=3     N=14    ...
i=4     N=17    ...
i=5     N=20    //5 ème et denière itération
```

- Les valeurs qui vont être affichées sont: 8,11,14,17 et 20