

Structures itératives

1. Principe des itérations

Dans la plupart des problèmes, certaines actions doivent être exécutées plusieurs fois.

Quand le nombre de répétitions est grand, il est fastidieux de réécrire n fois la même séquence de code. Cette réécriture est même impossible dans le cas où le nombre de répétitions (itérations) est inconnu a priori (traiter un ensemble de données jusqu'à ce qu'il n'y en ait plus).

Il faut pouvoir exprimer la répétition d'une action, qui une fois initialisée se continuera jusqu'à ce qu'un certain événement se produise. Cet événement d'arrêt sera spécifié dans l'algorithme par l'intermédiaire d'une condition.

2. Structures itératives de base

Étudions quatre types d'itérations (boucles) dont vous trouverez l'équivalent dans les principaux langages de programmation.

Boucle "Tant que" avec un test en début d'itération :

Tantque Condition **Faire**

Actions

Refaire

Boucle "Jusqu'à" avec un test en début d'itération :

Jqa Condition **Faire**

Actions

Refaire

Boucle "Tant que" avec un test en fin d'itération :

Faire

Actions

Tantque Condition **Refaire**

Boucle "Jusqu'à" avec un test en fin d'itération :

Faire

Actions

Jqa Condition **Refaire**

Ces quatre boucles ont un certain nombre de points communs :

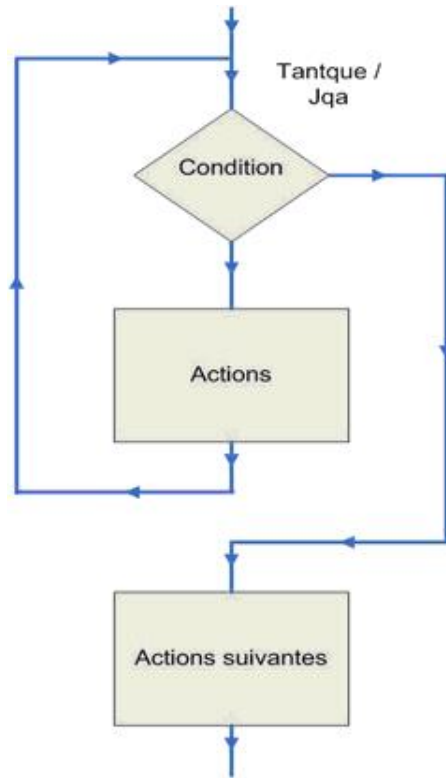
- **Jqa** est une abréviation pour Jusqu'à
- **Tantque** est une abréviation pour Tant que

- Condition est une expression booléenne de résultat Vrai ou Faux
- Actions représente une séquence d'actions

Avec les deux dernières itérations pour lesquelles le test se fait en fin de boucle, les Actions sont effectuées au moins une fois.

Les boucles "Tant que" et "Jusqu'à" ont un fonctionnement très proche dans la mesure où il suffit d'inverser la condition pour passer d'une syntaxe à l'autre. Par exemple $COMPTEUR > 10$ dans le cas d'une boucle "Tant que" deviendra $COMPTEUR \leq 10$ dans le cas d'une boucle "Jusqu'à".

L'ordinogramme suivant correspond aux structures "Tant que" et "Jusqu'à" (avec un test en début d'itération) :



Dans le cas d'une boucle **Jqa** :

- la flèche reliant "Condition" à "Actions" correspond à l'état Faux de la Condition,
- la flèche reliant "Condition" à "Actions suivantes" correspond à l'état Vrai de la Condition.

Dans le cas d'une boucle **Tantque** :

- la flèche reliant "Condition" à "Actions" correspond à l'état Vrai de la Condition,
- la flèche reliant "Condition" à "Actions suivantes" correspond à l'état Faux de la Condition.

3. Exercice n°9 : Moyenne de 10 nombres

Sujet

Calculer et afficher la moyenne de 10 nombres saisis au clavier

Corrigé

Début

Co Calcul de la moyenne de 10 nombres saisis au clavier **Fco**

Co Déclarations **Fco**

Réel NBLU, CPT, SOMME, MOYENNE

Co Initialisations **Fco**

CPT <- 1

SOMME <- 0

Co Itération **Fco**

Tantque CPT < 11 **Faire**

Ecrire("Nombre n° ", CPT, " : ")

 NBLU <- **Lire**

 SOMME <- SOMME + NBLU

 CPT <- CPT + 1

Refaire

Co Affichage du résultat **Fco**

MOYENNE <- SOMME / 10

Ecrire("Moyenne : ", MOYENNE)

Fin

4. Exercice n°10 : Moyenne d'une série de n nombres

Sujet

Calculer et afficher la moyenne de n nombres réels saisis au clavier (liste terminée par un zéro)

Profitons de cet énoncé un peu plus complexe pour montrer qu'une même problématique peut être résolue par de multiples algorithmes.

Corrigé n°1

Début

Co Déclarations **Fco**

Réel NBLU, CPT, SOMME, MOYENNE

Co Initialisations **Fco**

CPT <- 0

SOMME <- 0

NBLU <- 1 **Co** Pour permettre le passage dans l'itération au moins une fois **Fco**

Co Boucle de lecture **Fco**

Jqa NBLU = 0 **Faire**

Ecrire("Nombre (0 pour finir) : ")

 NBLU <- **Lire**

 CPT <- CPT + 1

 SOMME <- SOMME + NBLU

Refaire

Co Affichage du résultat **Fco**

MOYENNE <- SOMME / (CPT-1)

Ecrire("Moyenne des ", CPT - 1, "nombres = ", MOYENNE)

Fin

Dans ce corrigé, la saisie du zéro final ne provoque pas la sortie immédiate de la boucle. Il faut donc réduire la valeur de la variable CPT (compteur du nombre de nombres saisis au clavier) dans le calcul de la moyenne. Il faut aussi penser à affecter une valeur initiale à la variable NBLU (ici 1) avant l'itération afin que celle-ci puisse débiter. Enfin avec cette version de corrigé, un problème se produirait (division par zéro) si le premier nombre saisi au clavier est un 0.

Corrigé n°2

Début

Co Déclarations **Fco**

Réel NBLU, CPT, SOMME, MOYENNE

Co Initialisations **Fco**

CPT <- 0

SOMME <- 0

Co Boucle de lecture **Fco**

Faire

Ecrire("Nombre (0 pour finir) : ")

NBLU <- **Lire**

CPT <- CPT + 1

SOMME <- SOMME + NBLU

Tantque NBLU <> 0 **Refaire**

Co Affichage du résultat **Fco**

Si CPT-1=0

Alors

Ecrire("Aucun chiffre n'a été saisi")

Sinon

MOYENNE <- SOMME / (CPT-1)

Ecrire("Moyenne des ", CPT - 1, "nombres = ", MOYENNE)

Finsi

Fin

Dans ce corrigé, une boucle **Tantque** avec un test en fin de boucle a été utilisée. Il n'est pas ici nécessaire d'affecter une valeur factice à la variable NBLU avant la boucle. Par ailleurs une structure conditionnelle après l'itération permet d'éviter une éventuelle division par 0.

Corrigé n°3

Début

Co Déclarations **Fco**

Réel NBLU, CPT, SOMME, MOYENNE

Co Initialisations **Fco**

CPT <- 0

SOMME <- 0

Co 1ère saisie **Fco**

Ecrire("Nombre (0 pour finir) : ")

NBLU <- **Lire**

```

Co Boucle de lecture Fco
Tantque NBLU <> 0 Faire

    CPT <- CPT +1
    SOMME <- SOMME + NBLU
    Ecrire("Nombre (0 pour finir) : ")
    NBLU <- Lire

Refaire

Co Affichage du résultat Fco
Si CPT=0
Alors

    Ecrire("Aucun chiffre n'a été saisi")

Sinon

    MOYENNE <- SOMME / CPT
    Ecrire("Moyenne des ", CPT, "nombres = ", MOYENNE)

Finsi

Fin

```

Dans ce corrigé, une boucle **Tantque** avec un test en début de boucle a été retenue. La particularité est qu'une première saisie clavier (variable NBLU) est faite avant l'itération, ce qui oblige ensuite à prévoir la saisie des nombres suivants de la suite en fin d'itération. L'intérêt de cette solution est que le zéro final n'est pas pris en compte dans la variable CPT.

5. Exercice n°11 : Plus Grand Commun Diviseur par la méthode des divisions successives

Sujet

Calculer le PGCD de deux nombres entiers lus au clavier (Méthode de la division)

La démarche de calcul est la suivante :

- Saisir les nombres A et B au clavier
- Calculer le RESTE de la division entière de A par B
- Quand ce RESTE est nul B est le PGCD cherché
- Remplacer A par B
- Remplacer B par ce RESTE
- Recommencer au niveau du calcul du RESTE

Profitons de cet énoncé pour mettre en évidence que les boucles **Tantque** et **Jqa** peuvent s'utiliser indifféremment dans la résolution des problèmes itératifs, il suffit d'inverser logiquement la condition testée. Il faut noter que la boucle **Tantque** est systématiquement implémentée dans les langages de programmation (nommée en général While ou Do While). La boucle **Jqa** (qui se traduirait par un Until ou Do Until) n'est pas, par contre, systématiquement implémentée.

Corrigé n°1

Début

```

Co Déclarations Fco
Ent A, B, RESTE

Co Saisie des deux nombres Fco
Ecrire("Premier nombre : ")
A <- Lire
Ecrire("Deuxième nombre : ")
B <- Lire

Co Boucle de traitement Fco
RESTE <- A Mod B
Tantque RESTE <> 0 Faire

    A <- B
    B <- RESTE
    RESTE <- A Mod B

Refaire

Co Fin de traitement Fco
Ecrire("PGCD = ", B)

Fin

```

Corrigé n°2

Début

```

Co Déclarations Fco
Ent A, B, RESTE

Co Saisie des deux nombres Fco
Ecrire("Premier nombre : ")
A <- Lire
Ecrire("Deuxième nombre : ")
B <- Lire

Co Boucle de traitement Fco
RESTE <- A Mod B
Jqa RESTE = 0 Faire

    A <- B
    B <- RESTE
    RESTE <- A Mod B

Refaire

Co Fin de traitement Fco
Ecrire("PGCD = ", B)

Fin

```

6. Structure itérative Pour

Les structures itératives **Jqa Faire ... Refaire** et **Tantque Faire ... Refaire** peuvent être remplacées par la structure itérative **Pour Faire ... Refaire** dans le cas où le nombre d'itérations est connu avant le début de la structure et uniquement dans ce cas.

Cette itération, un peu déroutante au premier abord, a la particularité de présenter un certain automatisme quant à la gestion du compteur d'itérations effectuées.

Sa syntaxe générale est :

Pour VARIABLE **De** EXPRESSION_DEBUT **A** EXPRESSION_FIN [**Pas** EXPRESSION_PAS] **Faire**

Actions

Refaire

Notons ses particularités :

- VARIABLE est une variable entière à déclarer en début d'algorithme.
- EXPRESSION_DEBUT est soit une constante entière (par exemple 1) soit une variable entière à déclarer en début d'algorithme et à initialiser avant la structure itérative.
- EXPRESSION_FIN est soit une constante entière (par exemple 10) soit une variable entière à déclarer en début d'algorithme et à initialiser avant la structure itérative.
- EXPRESSION_PAS est soit une constante entière (par exemple 2) soit une variable entière à déclarer en début d'algorithme et à initialiser avant la structure itérative. Le plus souvent **Pas** EXPRESSION_PAS n'est pas prévu dans ce type d'itération.
- EXPRESSION_PAS peut être négatif et dans ce cas EXPRESSION_DEBUT doit être > (ou éventuellement =) à EXPRESSION_FIN.
- La structure **Pour** gère elle-même l'incrémentation du compteur de boucle (VARIABLE). Il ne faut donc pas intervenir sur ce compteur dans les Actions, bien que cela ne soit pas interdit.
- En informatique un paramètre facultatif est noté comme suit [paramètre]. Quand le paramètre est obligatoire, les [] ne sont pas à prévoir.

7. Exercice n°12 : Calcul de la moyenne de 10 nombres

Sujet

Calculer la moyenne de 10 nombres saisis au clavier par l'intermédiaire d'une boucle **Pour**

Corrigé

Début

Co Calcul de la moyenne de 10 nombres saisis au clavier **Fco**

Co Déclarations **Fco**

Ent CPT

Réel NBLU, SOMME, MOYENNE

Co Initialisations **Fco**

SOMME <- 0

Co Itération **Fco**

Pour CPT **De** 1 **A** 10 **Faire**

Ecrire("Nombre n° ", CPT, " : ")

 NBLU <- **Lire**

 SOMME <- SOMME + NBLU

Co NB : Il ne faut surtout pas modifier la valeur de la variable CPT dans la boucle **Fco**

Refaire

Co Affichage du résultat **Fco**

MOYENNE <- SOMME / 10

Ecrire("Moyenne : ", MOYENNE)

Co NB : Il faut éviter de diviser ici la SOMME par CPT **Fco**

Co dans la mesure où dans certains langages **Fco**

Co CPT vaudra 11 en sortie **Fco**

Fin

Comme indiqué en commentaires dans l'algorithme ci-avant, il est préférable de ne pas utiliser directement la valeur de la variable CPT (qui a servi à compter les nombres saisis au clavier) car selon les variantes d'implémentation dans les langages de programmation, CPT pourrait valoir 10 ou 11.

8. Exercice n°13 : Décompte du nombre de voyelles dans un mot

Sujet

Compter le nombre de voyelles d'un mot saisi au clavier

Rappel de cours : une fonction **Longueur**(CHAINE) sera utilisée pour déterminer le nombre de caractères du mot saisi. La fonction **Sous_chaine**(CHAINE, POSITION_DEBUT, [POSITION_FIN]) sera également nécessaire.

Corrigé

Début

Co Déclarations **Fco**

Car MOT

Ent NB_VOYELLES, COMPTEUR

Co Saisie du mot au clavier **Fco**

Ecrire("Mot : ")

MOT <- **Lire**

Co Mise en majuscules du mot **Fco**

MOT <- **Majuscules**(MOT)

Co Initialisation des variables **Fco**

NB_VOYELLES <- 0

Co Détermination du nombre de voyelles **Fco**

Pour COMPTEUR **De** 1 **A** **Longueur**(MOT) **Faire**

Co Analyse de la lettre en cours **Fco**

Suivant **Sous_chaine** (MOT, COMPTEUR, COMPTEUR)

 "A" : NB_VOYELLES <- NB_VOYELLES + 1

 "E" : NB_VOYELLES <- NB_VOYELLES + 1

 "I" : NB_VOYELLES <- NB_VOYELLES + 1

 "O" : NB_VOYELLES <- NB_VOYELLES + 1

 "U" : NB_VOYELLES <- NB_VOYELLES + 1

 "Y" : NB_VOYELLES <- NB_VOYELLES + 1

Finsuivant


Refaire

Co Affichage du résultat **Fco**

Ecrire(MOT, "contient : ", NB_VOYELLES, " voyelle(s)")

Fin

Pour la première fois, nous voyons une structure conditionnelle (**Suivant ... Finsuivant**) intégrée dans une structure itérative (**Pour**). Bien évidemment cette structure conditionnelle aurait pu être basée sur un **Si ... Finsi**.

 En fonction des problématiques à résoudre, vous l'aurez compris il faudra combiner au mieux les structures conditionnelles (**Si ... Finsi** ou **Suivant ... Finsuivant**) avec les différentes variantes de boucles (**Tantque**, **Jqa** et **Pour**) en les imbriquant, séquençant...