Suite Chapitre 5 – les structures répétitives

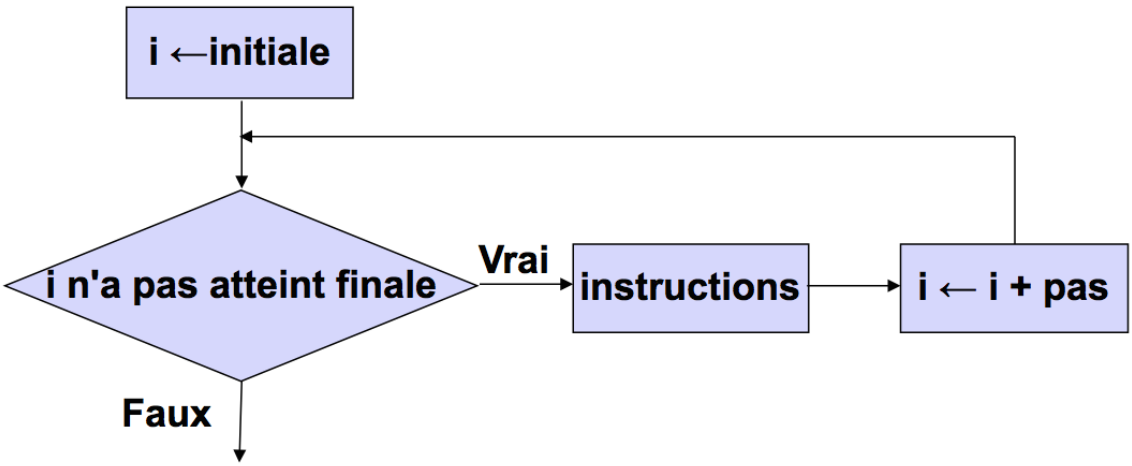
I. Boucle Pour

La syntaxe de la boucle Pour est :

Pour compteur **allant de** initiale **à** finale par **pas** valeur du pas

Liste des instructions

FinPour



Remarque : le nombre d'itérations dans une boucle Pour est connu avant le début de la boucle

- **Compteur** est une variable de type entier (ou caractère). Elle doit être déclarée.
- **Initiale et finale** peuvent être des valeurs, des variables définies avant le début de la boucle ou des expressions de même type que compteur.

- **Pas** est un entier qui peut être positif ou négatif. **Pas** peut ne pas être mentionné, car par défaut sa valeur est égal à 1. Dans ce cas, le nombre d'itérations est égal à finale - initiale+ 1.

Déroulement des boucles Pour :

1. La valeur initiale est affectée à la variable compteur
2. On compare la valeur du compteur et la valeur de finale :
    a. Si la valeur du compteur est > à la valeur finale dans le cas d'un pas positif (ou si compteur est < à finale pour un pas négatif), on sort de la boucle et on continue avec l'instruction qui suit FinPour
    b. Si compteur est <= à finale dans le cas d'un pas positif (ou si compteur est >= à finale pour un pas négatif), instructions seront exécutées
        i. Ensuite, la valeur de compteur est incrémentée de la valeur du pas si pas est positif (ou décrémenté si pas est négatif)
        ii. On recommence l'étape 2 : La comparaison entre compteur et finale est de nouveau effectuée, et ainsi de suite …

Exemple 1 : Pour avec pas positif

Algorithme exemple_1
Variables

```
        n, i : entier
Debut
    Ecrire (" Entrez la valeur de n ")
    Lire (n)
    Pour i allant de 1 à n
        Ecrire ("Bonjour tout le monde")
    FinPour
Fin
```

Exemple 2 : Pour avec pas négatif

```
Algorithme exemple_1
Variables
        n, i : entier
Debut
    Ecrire (" Entrez la valeur de n ")
    Lire (n)
    Pour i allant de n à 1 pas -1
        Ecrire ("Bonjour tout le monde")
    FinPour
Fin
```

II. Les boucles imbriquées

Les instructions d'une boucle peuvent être des instructions itératives. Dans ce cas, on aboutit à des **boucles imbriquées.**

```
Pour i allant de 1 à 5
```

```
        Pour j allant de 1 à i
            écrire("O")
        FinPour
        écrire("X")
FinPour
```

Exécution du programme :

OX

OOX

OOOX

OOOOX

OOOOOX

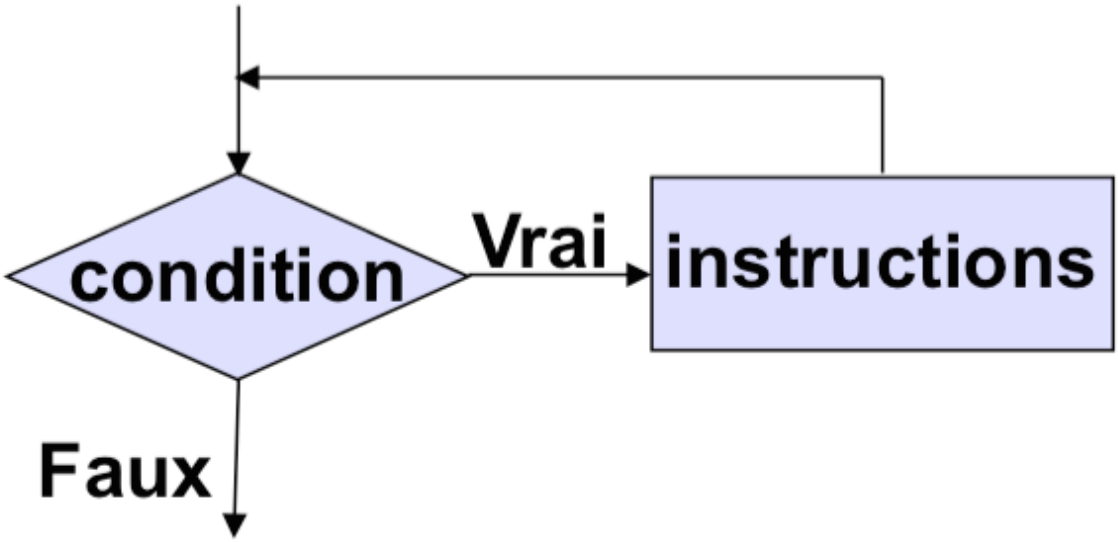
III. Les boucles Tant que

La syntaxe d'une boucle Tant que est :

```
Tant que (condition)
        Liste des instructions
FinTant que
```

- La condition (dite condition de contrôle de la boucle) est évaluée avant chaque itération.
- Si la condition est vraie, on exécute instructions (corps de la boucle), puis, on retourne tester la condition. Si elle est encore vraie, on répète l'exécution, …

- Si la condition est fausse, on sort de la boucle et on exécute l'instruction qui est après FinTantQue



Exemple 1 : Contrôle de saisie d'une lettre majuscule jusqu'à ce que le caractère entré soit valable.

```
Algorithme Contrôle_Saisie
Variable
        C : caractère
Debut
        Ecrire (" Entrez une lettre majuscule ")
        Lire (C)
        TantQue (C < 'A' OU C > 'Z')
                Ecrire ("Saisie erronée. Recommencez")
                Lire (C)
```

```
        FinTantQue
        Ecrire ("Saisie valable")
    Fin
```

Exemple 2 : Un algorithme qui détermine le premier nombre entier N tel que la somme de 1 à N dépasse strictement 100.

```
    Algorithme somme_à_100
    Variables
        som, i : entier
    Début
        i ← 0
        som← 0
        TantQue (som <=100)
            i ← i+1
            som ← som+i
        FinTantQue
        Ecrire (" La valeur cherchée est N=  ", i)
    Fin
```

Remarques :

- Le nombre d'itérations dans une boucle TantQue n'est pas connu au moment d'entrée dans la boucle. Il dépend de l'évolution de la valeur de condition

- Une des instructions du corps de la boucle doit absolument changer la valeur de condition de vrai à faux (après un certain nombre d'itérations), sinon le programme tourne indéfiniment

➔ **Il faut toujours faire attention aux boucles infinies**

- Exemple de boucle infinie :

```
i ← 2
TantQue  (i > 0)
        i ←  i+1
FinTantQue
```

Liens entre Pour et tant que

La boucle Pour est un cas particulier de Tant Que (cas où le nombre d'itérations est connu et fixé) . Tout ce qu'on peut écrire avec Pour peut être remplacé avec TantQue (la réciproque est fausse)

```
Pour compteur allant de initiale à finale par pas valeur du pas
        instructions
FinPour
```

peut être remplacé par : (cas d'un pas positif)

```
        compteur ← initiale
        TantQue  compteur <= finale
                        instructions
```

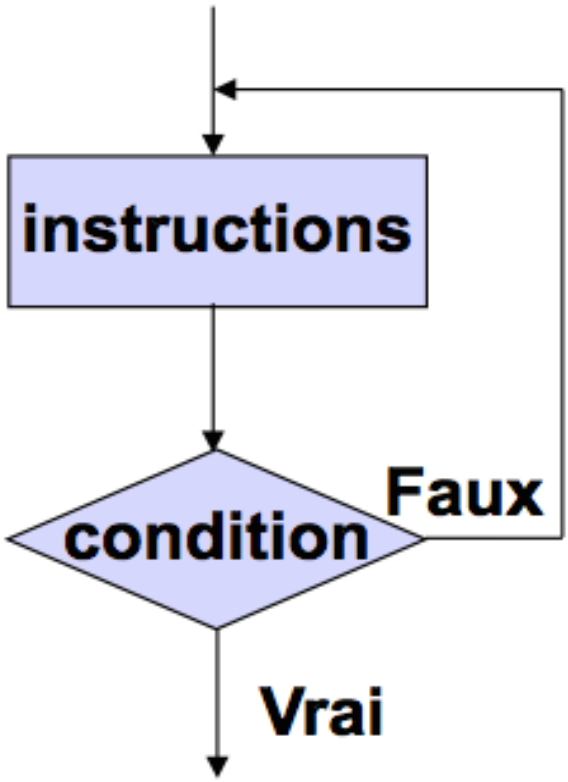
```
                compteur ← compteur+pas
    FinTantQue
```

IV. Les boucles Répéter … jusqu'à …

La syntaxe de l'instruction jusqu'à est :

```
Répéter
        instructions

Jusqu'à  condition
```



- Condition est évaluée après chaque itération

- Les instructions entre *Répéter* et *jusqu'à* sont exécutées au moins une fois et leur exécution est répétée jusqu'à ce que condition soit vrai (tant qu'elle est fausse.

V. Exercices d'application

Exercice 1 :

Écrire un algorithme qui demande un nombre de départ, et qui ensuite écrit la table de multiplication de ce nombre, présentée comme suit (cas où l'utilisateur entre le nombre 5) :

Table de 5 :

5 x 1 = 5

5 x 2 = 10

5 x 3 = 15

…

5 x 10 = 50

Exercice 2 :

Écrire un algorithme qui demande un nombre de départ, et qui calcule la somme des entiers jusqu'à ce nombre. Par exemple, si l'on entre 5, le programme doit calculer :

1 + 2 + 3 + 4 + 5 = 15

Exercice 3 :

Écrire un algorithme qui demande un nombre de départ, et qui calcule sa factorielle.

NB : la factorielle de 8, notée 8 ! vaut 1 x 2 x 3 x 4 x 5 x 6 x 7 x 8

Exercice 4 :

Écrire un algorithme qui demande successivement 10 nombres à l'utilisateur, et qui lui dise ensuite quel était le plus grand parmi ces 10 nombres et sa position :

Entrez le nombre numéro 1 : 12

Entrez le nombre numéro 2 : 14

…

Entrez le nombre numéro 10 : 6

Le plus grand de ces nombres est : 14, sa position : 2

Exercice 5

Écrire un algorithme qui demande successivement des nombres à l'utilisateur, et qui lui dise ensuite quel était le plus grand parmi ces nombres et quel était sa position. La saisie des nombres s'arrête lorsque l'utilisateur entre un zéro.

Exercice 6 :

Écrire un algorithme qui demande successivement des nombres à l'utilisateur, et qui calcule leur moyenne. La saisie des nombres s'arrête lorsque l'utilisateur entre un zéro.