Boucles conditionnelles

Boucles conditionnelles

- **Exemple** : Donner l'algorithme qui demande à l'utilisateur d'entrer plusieurs nombres. La saisie des nombres s'arrête lorsque l'utilisateur entre un zéro
- Lorsqu'on souhaite **répéter une instruction** tant qu'une **certaine condition** est remplie, alors qu'il est à priori **impossible de savoir à l'avance** au bout de **combien d'itérations** cette condition cessera d'être satisfaite. Dans ce cas, on a deux possibilités :

```
TANT QUE (condition)
        Instructions à répéter
FIN TANT QUE
```

```
REPETER
        Instructions à répéter
JUSQU'À (condition)
```

Boucle Tant Que

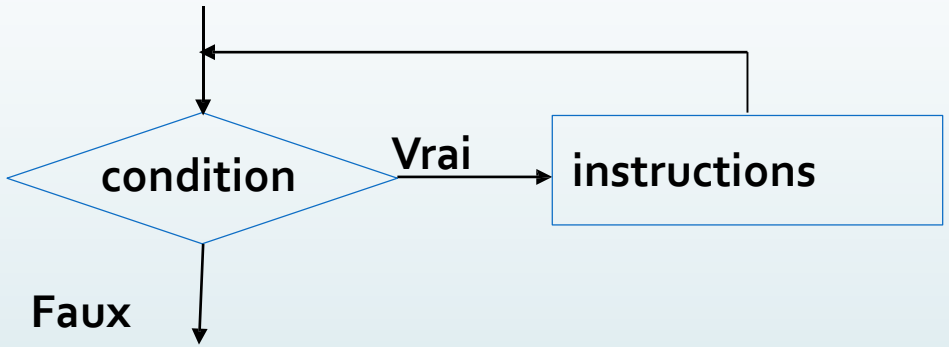
- **Syntaxe**

TANT QUE *(condition)*

Instructions à répéter

FINTANT QUE

Schéma itératif



- 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 pour 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**

Boucle Tant Que

- **Remarque**
  - 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
  - Exemple de boucle infinie :

```
i ← 1
TANT QUE (i > 0)

FIN TANT QUE
```

Correction

```
i ← 1
TANT QUE (i < 100)
        i ← i+1
FIN TANT QUE
```

Boucle Tant Que

- **Exemple 1: Ecrire un algorithme qui demande à l'utilisateur de saisir une valeur entre 1 et 10. Si la valeur n'est pas comprise entre 1 et 10, on redemande la saisie. Sinon on s'arrête.**

```
Algorithme SaisirNombre
VARIABLE N: REEL
Début
        Ecrire (" Saisir un nombre entre 1 et 10 ")
        Lire (N)
        TANT QUE    (N<1 OU N > 10)
                Ecrire (" Saisir un nombre entre 1 et 10: ")
                Lire (N)
        FIN TANT QUE
Fin
```

Boucle Tant Que

- **Exemple 2: Ecrire un algorithme qui calcule la factorielle d'un nombre saisi par l'utilisateur.**

```
Algorithme Factorielle
Variables N, i, Fact: Entier
Début
        Ecrire (" Saisir un nombre: ")
        Lire (N)
        Fact ← 1
        i ← 1
        TantQue (i < N)
                i ← i+1
                Fact ← Fact *i
        FinTantQue
        Ecrire (" La factorielle de ", N, " est : ", Fact)
Fin
```

Boucle Tant Que

- Q1. Simuler l'exécution des algorithmes suivants :

```
Algorithme Tanque2
Variables nb : entier
Début
   nb ← 10
   Tantque nb > 40
         écrire(nb)
         nb ← nb + 10
    FinTantQue
écrire ('le nombre vaut ', nb)
Fin
```

```
Algorithme Tanque3
Variables x : entier
Début
     lire(x)
     Tantque x <= 3
           écrire(x)
           x ← x + 1
      FinTantQue
      écrire (x)
Fin
```

Boucle Tant Que

- **Q2. Simuler l'exécution des algorithmes suivants :**

```
Algorithme Tanque4
Variables a,b : entier
Début
    a ← 1
    b ← a
    TantQue (a < 100)
        b ← b * 5
        écrire(b)
    FinTantQue
Fin
```

Boucle Tant Que

- Q3. Un algorithme qui demande un nombre de départ, et qui ensuite affiche les dix nombres suivants. Par exemple, si l'utilisateur entre le nombre 17, le programme affichera les nombres de 18 à 27.

- Q4. Un algorithme qui détermine le premier nombre entier N tel que la somme de 1 à N dépasse strictement 100

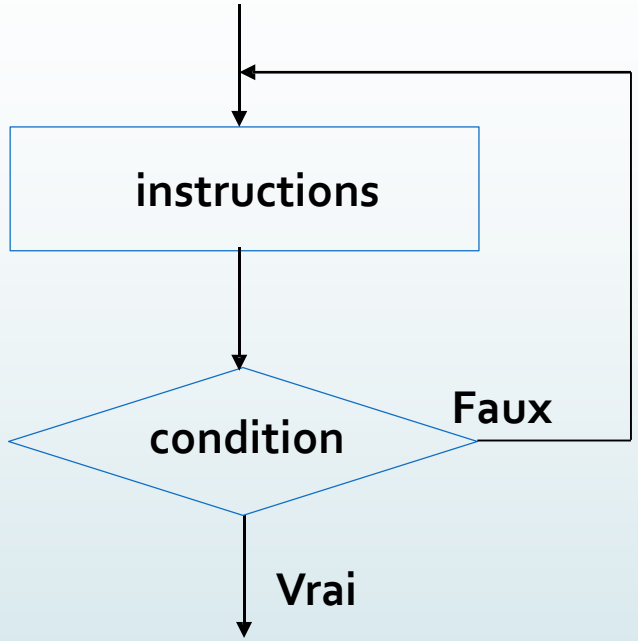
Boucle Répéter jusqu'à

- **Syntaxe:**

REPETER

Instructions à répéter

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 la condition soit vraie** (**tant qu'elle est fausse)**

Boucle Répéter jusqu'à

- **Mécanisme** :
  - Exécution **la séquence des instructions** **une première fois**
  - Evaluation de **condition** :
    - **Vraie** → **sortie** et exécution de la suite de l'algorithme
    - **fausse** → **retour** pour exécuter à nouveau la séquence et tester la condition
- **Remarque :**
  - la séquence des instructions à répéter doit **modifier l'expression booléenne ou condition de sortie**, sinon **la boucle est infinie**
  - Cette **séquence** est exécutée **au moins une fois**

Boucle Répéter jusqu'à

- **Exemple 1: Un algorithme pour la saisie d'un nombre positif non nul.**

```
Algorithme SaisieNbrePositif
Variables N : réel
Début
    Répéter
        écrire ("Introduisez un nombre positif non nul" )
        lire ( N )
    jusqu'à ( N<=0 )
Fin.
```

Boucle Répéter jusqu'à

Exemple 2: Ecrire un algorithme permettant de calculer la somme d'une suite d'entiers qui se termine par 0

```
Algorithme SommeSuite
VARIABLES N, S: Entier
Début
        S ← 0
        Répéter
                Ecrire (" Saisir un entier")
                Lire (N)
                S ← S +N
        Jusqu'à (N = 0)
        Ecrire (" La somme est: " , S)
Fin
```

Boucle Répéter jusqu'à

- **Q1. Ecrire un algorithme permettant de lire une suite de nombres réels sur le clavier. Le dernier élément à lire est un zéro. L'algorithme doit afficher la somme des éléments lus.**

- **Q2. Un algorithme qui détermine le premier nombre entier N tel que la somme de 1 à N dépasse strictement 100 (version avec répéter jusqu'à).**

Pour vs 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 cpt= val_ initiale A val_finale PAS de val_pas
        instructions
FinPour
```

```
cpt ← val_initiale
TantQue (cpt <= val_finale)
        instructions
        cpt ← cpt+val_pas
FinTantQue
```

Pour vs Tant Que

Exemple: Ecrire un algorithme qui calcule la factorielle d'un nombre saisi par l'utilisateur

Algorithme / TantQue

```
Algorithme Factorielle
Variable N, i, Fact: Entier
Début
    Ecrire (" Saisir un nombre: ")
    Lire (N)
    Fact ← 1
    i ← 1
    TantQue (i<=N)
        Fact ← Fact *i
        i ← i + 1
     FinTanQue
     Ecrire (" La factorielle de ", N, " est : ", Fact)
Fin
```

Algorithme / Pour

```
Algorithme Factorielle
Variables N, i, Fact: Entier
Début
        Ecrire (" Saisir un nombre: ")
        Lire (N)
        Fact ← 1
        Pour i=1 A N
            Fact ← Fact *i
        FinPour
    Ecrire (" La factorielle de ", N, " est : ", Fact)

Fin
```

Pour vs Tant Que

Pour

- Implicitement, la structure Pour:
  - ✓ Initialise un compteur
  - ✓ Incrémente le compteur à chaque pas
  - ✓ Vérifie que le compteur ne dépasse pas la borne supérieure
- Nombre d'itérations connu à l'avance

Tant que

- Explicitement, la structure tant que:
  - ✓ Initialise un compteur
  - ✓ Incrémente le compteur à chaque pas
  - ✓ Vérifie que le compteur ne dépasse pas la borne supérieure
- L'arrêt de la boucle dépend d'une condition

Répéter vs Tant Que

Exemple: Ecrire un algorithme permettant d'effectuer un contrôle de saisie sur l'âge saisi par un utilisateur

```
Algorithme SaisirAge
VARIABLE Age: Entier
Début

        Répéter
            Ecrire ("Saisir votre age")
            Lire (Age)

    Jusqu'à (Age > 0 ET Age <= 110)
     Ecrire (" Votre age est: " ,Age)
Fin
```

```
Algorithme SaisirAge
VARIABLE Age: Entier
Début
            Age ← 0
            TantQue (Age <=0 OU Age> 110)
                Ecrire ("Saisir votre age")
                Lire (Age)

            FinTantQue
            Ecrire (" Votre age est: " , Age)

Fin
```

Tant Que vs Répéter

Tant que

- Condition vérifiée avant chaque exécution du traitement
- Le traitement peut donc ne pas être exécuté
- La condition porte sur la saisie de nouvelles variables (relance)

Répéter…jusqu'à

- Condition vérifiée après chaque exécution du traitement
- Le traitement est exécuté au moins une fois
- La condition porte sur le résultat du traitement
- La boucle répéter est typique pour les saisies avec vérification

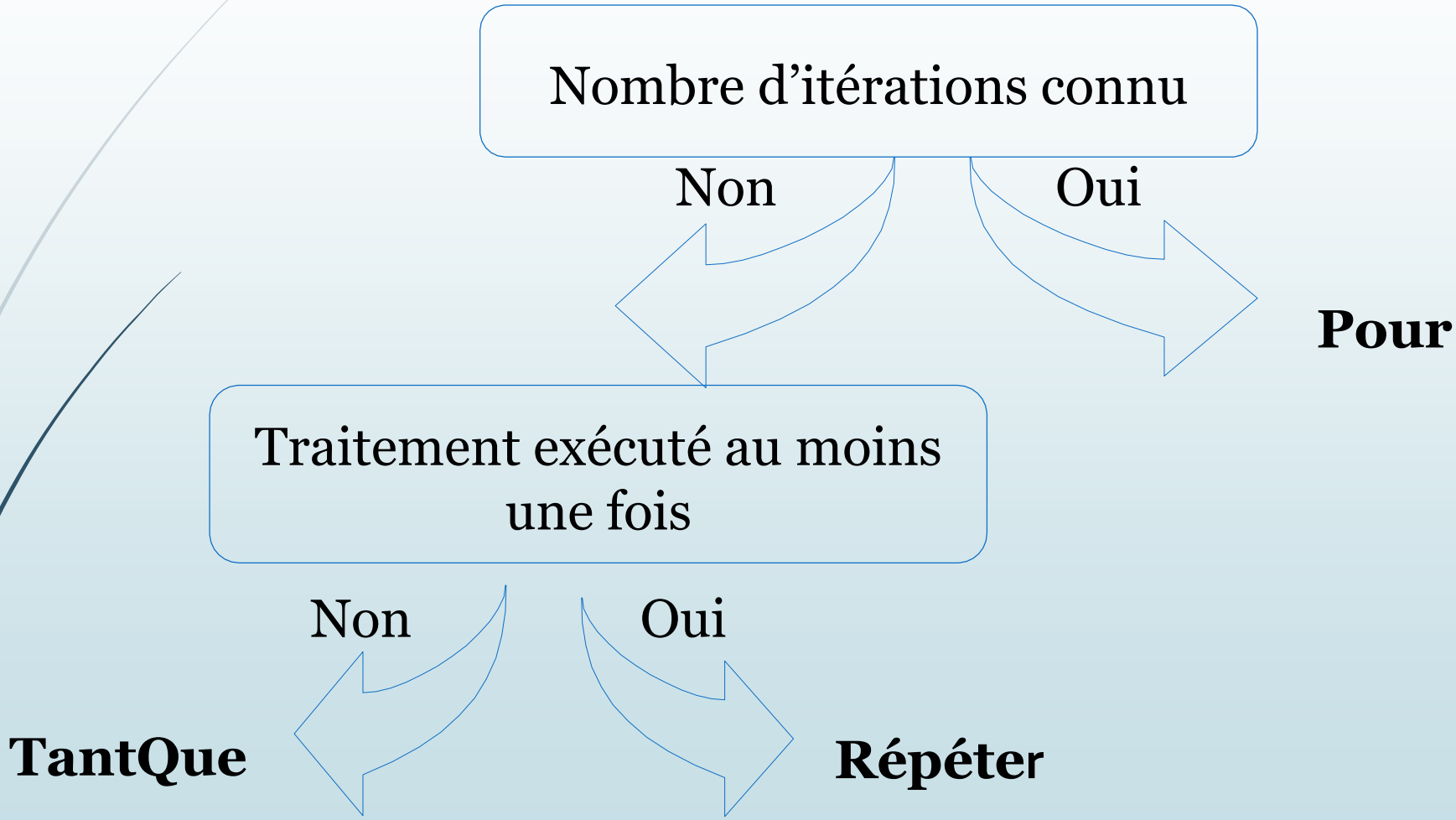
Choix d'un type de boucle

- Si on peut déterminer le nombre d'itérations avant l'exécution de la boucle, il est plus naturel d'utiliser **la boucle Pour**

- S'il n'est pas possible de connaître le nombre d'itérations avant l'exécution de la boucle, on fera appel à l'une des boucles : **TantQue ou répéter jusqu'à**

- Pour le choix entre **TantQue** et **jusqu'à** :
  - Si on doit tester la condition de contrôle avant de commencer les instructions de la boucle, on utilisera **TantQue**
  - Si la valeur de la condition de contrôle dépend d'une première exécution des instructions de la boucle, on utilisera **répéter jusqu'à**

Choix d'un type de boucle



Exercices

Exercice 1

Ecrire un algorithme qui demande successivement des nombres à l'utilisateur, et qui calcule le nombre de valeurs saisies. La saisie des nombres s'arrête lorsque l'utilisateur entre le caractère « n » ou « N ».

Exercice 2

Ecrire 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.

Exercice 3

Modifiez l'algorithme de la question 1, de façon qu'il nous renseigne sur le nombre des valeurs positives et sur le nombre des valeurs négatives. Ne comptez pas les valeurs nulles.

Exercices

Exercice 4

Ecrire un algorithme qui lit les caractères saisies par l'utilisateur. A la fin ce programme nous affichera la phrase saisie. La saisie des caractères s'arrête lorsqu'on tape point « . ». Pour l'utilisateur veut insérer un espace il lui suffit de tapez sur 0. Par exemple si l'utilisateur tape successivement les caractères « b » , « o », « n », « j », « o», « u », « r » , « t », « o », « u », « s », « . » , il nous affichera la chaîne « bonjourtous ».

Mais si il tape « b » , « o », « n », « j », « o », « u », « r » , « 0 », « t », « o », « u », « s », «.» , le programme affichera « bonjour tous ».