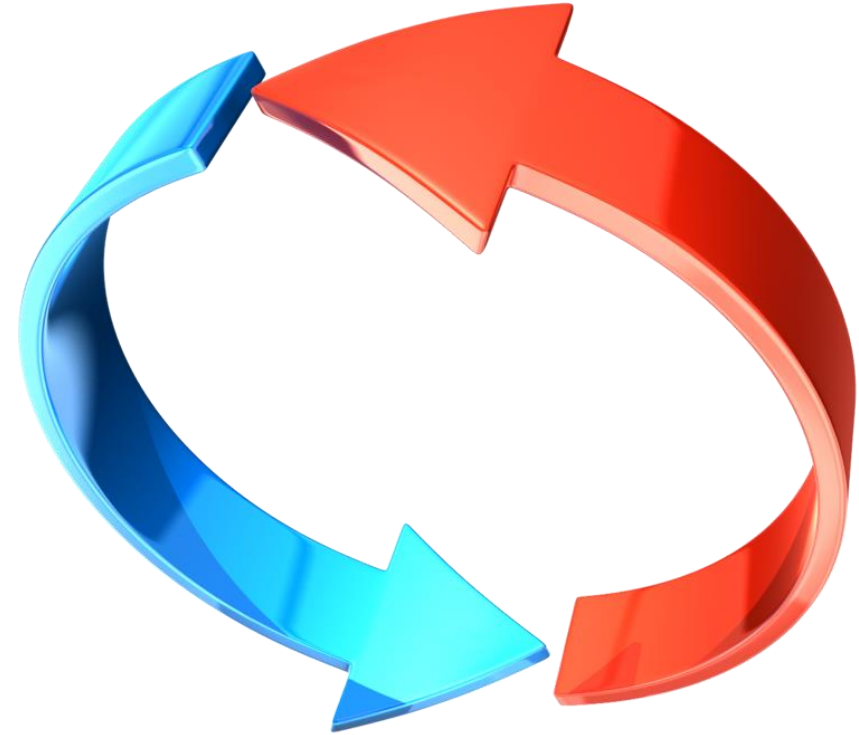


INFORMATIQUE 1

VII. LES BOUCLES



I. Principe

Problématique

- Dans l'algorithme suivant, que peut-on faire pour s'assurer que l'utilisateur rentre une note valide ?

```
VARIABLE
  n1: Réel
DEBUT
  ECRIRE ("Donnez la note")
  LIRE(n1)
  ECRIRE("Le résultat est" + n1)
FIN
```

Problématique

- Première idée : à l'aide d'une condition, on redemande à l'utilisateur si la valeur saisie est incorrecte :

```
VARIABLE
  n1: Réél
DEBUT
  ECRIRE ("Donnez la note")
  LIRE(n1)
  // condition de note non valide
  SI (n1 EST STRICTEMENT INFERIEUR A 0 OU n1 EST STRICTEMENT SUPERIEUR A 20) ALORS
    // on redemande la saisie
    ECRIRE("Donnez la note")
    LIRE (n1)
  FIN SI
  ECRIRE("Le résultat est" + m)
FIN
```

Problématique

- Si l'utilisateur donne plusieurs fois une valeur mauvaise ?

```
VARIABLE
  n1: Réel
DEBUT
  ECRIRE ("Donnez la note")
  LIRE(n1)
  // condition de note non valide
  SI (n1 EST STRICTEMENT INFÉRIEUR A 0 OU n1 EST STRICTEMENT SUPÉRIEUR A 20) ALORS
    ECRIRE("Donnez la note") // on redemande la saisie
    LIRE (n1)
    SI (n1 EST STRICTEMENT INFÉRIEUR A 0 OU n1 EST STRICTEMENT SUPÉRIEUR A 20) ALORS
      ECRIRE("Donnez la note") // on redemande la saisie
      LIRE (n1)
    FIN SI
  FIN SI
  ECRIRE("Le résultat est" + m)
FIN
```

Problématique

- Dans cette situation, on ne sait pas à l'avance combien de fois il va falloir vérifier la validité de la valeur saisie : utiliser des branchements conditionnels ne suffit pas.
- Il faudrait pouvoir **répéter** des instructions tant que la valeur n'est pas valide.
- C'est le rôle des **boucles** : les boucles permettent de répéter des instructions un certain nombre de fois pré-déterminé, ou, répéter en fonction d'une condition.

Principe

- Ce qu'il se passe:
 1. l'ordinateur lit les instructions de haut en bas (comme d'habitude) ;
 2. puis, une fois arrivé à la fin de la boucle, dans certaines conditions il repart à la première instruction ;
 3. il recommence alors à lire les instructions de haut en bas...
 4. ... et ça recommence !



- Il existe deux types de boucles :
 1. les boucles à nombre de tours déterminé par avance (**pour**)
 2. les boucles à nombre de tours indéterminé (**tant que ... faire ... / répéter ... tant que ...**)

II. Les différentes boucles

La boucle TANT QUE

- La boucle **TANT QUE** permet de répéter un ensemble d'instructions tant qu'une condition vaut VRAI.

```
TANT QUE (expression booléenne) FAIRE  
    ... // liste d'instructions à répéter  
FIN TANT QUE
```

La boucle TANT QUE

- La boucle **TANT QUE** permet de répéter un ensemble d'instructions tant que une condition est VRAI.

```
TANT QUE (expression booléenne) FAIRE  
    ... // liste d'instructions à répéter  
FIN TANT QUE
```




Le FIN TANT QUE indique la fin des instruction à répéter (comme le FIN SI)

La boucle TANT QUE

- La boucle **TANT QUE** permet de répéter un ensemble d'instructions tant que une condition est VRAI.

```
TANT QUE (expression booléenne) FAIRE  
    █... // liste d'instructions à répéter  
FIN TANT QUE
```



Indentation obligatoire pour les instructions appartenant au tant que !

La boucle TANT QUE

- Exemple :

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  c <- 0
  TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
    c <- c + a + b
    a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

```
VARIABLE
  a,b,c : entier
DEBUT
  → a <- 9
  b <- 3
  c <- 0
  TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
    c <- c + a + b
    a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

Variable	Valeur
a	
b	
c	

La boucle TANT QUE

- Exemple :

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  → b <- 3
  c <- 0
  TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
    c <- c + a + b
    a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

Variable	Valeur
a	9
b	
c	

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	9
b	3
c	

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  → c <- 0
  TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
    c <- c + a + b
    a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	9
b	3
c	0

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  c <- 0
  → TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
      c <- c + a + b
      a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```


La boucle TANT QUE

- Exemple :

Variable	Valeur
a	9
b	3
c	0

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  c <- 0
  TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
  →      c <- c + a + b
         a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

$a \geq 0$ est VRAI : on rentre dans la boucle

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	9
b	3
c	12

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  c <- 0
  TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
    c <- c + a + b
    → a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	6
b	3
c	12

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  c <- 0
  TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
    c <- c + a + b
    a <- a - b
  → FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	6
b	3
c	12

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  c <- 0
  → TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
      c <- c + a + b
      a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	6
b	3
c	12

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  c <- 0
  TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
  →      c <- c + a + b
         a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

$a \geq 0$ est VRAI : on rentre à nouveau dans la boucle

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	6
b	3
c	21

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  c <- 0
  TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
    c <- c + a + b
    → a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	3
b	3
c	21

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  c <- 0
  TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
    c <- c + a + b
    a <- a - b
  → FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	3
b	3
c	21

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  c <- 0
  → TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
      c <- c + a + b
      a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```


La boucle TANT QUE

- Exemple :

Variable	Valeur
a	3
b	3
c	21

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  c <- 0
  TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
  →      c <- c + a + b
         a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

$a \geq 0$ est VRAI : on rentre à nouveau dans la boucle

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	3
b	3
c	27

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  c <- 0
  TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
    c <- c + a + b
    → a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	0
b	3
c	27

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  c <- 0
  TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
    c <- c + a + b
    a <- a - b
  → FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	0
b	3
c	27

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  c <- 0
  → TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
      c <- c + a + b
      a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	0
b	3
c	27

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  c <- 0
  TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
  →      c <- c + a + b
         a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

$a \geq 0$ est VRAI : on rentre à nouveau dans la boucle

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	0
b	3
c	30

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  c <- 0
  TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
    c <- c + a + b
    → a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

Variable	valeur
a	-3
b	3
c	30

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  c <- 0
  TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
    c <- c + a + b
    a <- a - b
  → FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  c <- 0
  → TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
      c <- c + a + b
      a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

Variable	Valeur
a	-3
b	3
c	30

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	-3
b	3
c	30

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  c <- 0
  TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
    c <- c + a + b
    a <- a - b
  → FIN TANT QUE
  ECRIRE (c)
FIN
```

$a \geq 0$ est FAUX : on sort de la boucle

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	-3
b	3
c	30

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  c <- 0
  TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
    c <- c + a + b
    a <- a - b
  FIN TANT QUE
  → ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	-3
b	3
c	30

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 9
  b <- 3
  c <- 0
  TANT QUE (a EST SUPERIEUR OU EGAL A 0) FAIRE
    c <- c + a + b
    a <- a - b
  FIN TANT QUE
  ECRIRE (c)
→ FIN
```

30

La boucle TANT QUE

- Remarques :
 - Le nombre de tours de boucle n'est pas toujours connu à l'avance
 - Attention ! Il est possible de rester “coincé” dans la boucle : c'est **une boucle infinie**

La boucle TANT QUE

- Exemple :

```
VARIABLE
  a,b,c : entier
DEBUT
→ a <- 5
  b <- 3
  c <- 0
  TANT QUE (a EST DIFFERENT DE 0) FAIRE
    c <- c + a + b
    a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

Variable	Valeur
a	
b	
c	

La boucle TANT QUE

- Exemple :

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 5
  → b <- 3
  c <- 0
  TANT QUE (a EST DIFFERENT DE 0) FAIRE
    c <- c + a + b
    a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

Variable	Valeur
a	5
b	
c	

La boucle TANT QUE

- Exemple :

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 5
  b <- 3
  → c <- 0
  TANT QUE (a EST DIFFERENT DE 0) FAIRE
    c <- c + a + b
    a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

Variable	Valeur
a	5
b	3
c	

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	5
b	3
c	0

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 5
  b <- 3
  c <- 0
→ TANT QUE (a EST DIFFERENT DE 0) FAIRE
    c <- c + a + b
    a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```


La boucle TANT QUE

- Exemple :

Variable	Valeur
a	5
b	3
c	0

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 5
  b <- 3
  c <- 0
  TANT QUE (a EST DIFFERENT DE 0) FAIRE
  →      c <- c + a + b
          a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	5
b	3
c	8

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 5
  b <- 3
  c <- 0
  TANT QUE (a EST DIFFERENT DE 0) FAIRE
    c <- c + a + b
    → a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	2
b	3
c	8

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 5
  b <- 3
  c <- 0
  TANT QUE (a EST DIFFERENT DE 0) FAIRE
    c <- c + a + b
    a <- a - b
  → FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 5
  b <- 3
  c <- 0
→ TANT QUE (a EST DIFFERENT DE 0) FAIRE
    c <- c + a + b
    a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

Variable	Valeur
a	2
b	3
c	8

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	2
b	3
c	8

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 5
  b <- 3
  c <- 0
  TANT QUE (a EST DIFFERENT DE 0) FAIRE
  →      c <- c + a + b
         a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	2
b	3
c	13

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 5
  b <- 3
  c <- 0
  TANT QUE (a EST DIFFERENT DE 0) FAIRE
    c <- c + a + b
    → a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	-1
b	3
c	13

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 5
  b <- 3
  c <- 0
  TANT QUE (a EST DIFFERENT DE 0) FAIRE
    c <- c + a + b
    a <- a - b
  → FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 5
  b <- 3
  c <- 0
  → TANT QUE (a EST DIFFERENT DE 0) FAIRE
      c <- c + a + b
      a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

Variable	Valeur
a	-1
b	3
c	13

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	-1
b	3
c	13

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 5
  b <- 3
  c <- 0
  TANT QUE (a EST DIFFERENT DE 0) FAIRE
  →      c <- c + a + b
         a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	-4
b	3
c	15

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 5
  b <- 3
  c <- 0
  TANT QUE (a EST DIFFERENT DE 0) FAIRE
    c <- c + a + b
    a <- a - b
  → FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	-4
b	3
c	15

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 5
  b <- 3
  c <- 0
→ TANT QUE (a EST DIFFERENT DE 0) FAIRE
    c <- c + a + b
    a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	-4
b	3
c	15

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 5
  b <- 3
  c <- 0
  TANT QUE (a EST DIFFERENT DE 0) FAIRE
  →      c <- c + a + b
          a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	-4
b	3
c	14

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 5
  b <- 3
  c <- 0
  TANT QUE (a EST DIFFERENT DE 0) FAIRE
    c <- c + a + b
    → a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

Variable	Valeur
a	-7
b	3
c	14

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 5
  b <- 3
  c <- 0
  TANT QUE (a EST DIFFERENT DE 0) FAIRE
    c <- c + a + b
    a <- a - b
  → FIN TANT QUE
  ECRIRE (c)
FIN
```

La boucle TANT QUE

- Exemple :

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 5
  b <- 3
  c <- 0
→ TANT QUE (a EST DIFFERENT DE 0) FAIRE
    c <- c + a + b
    a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

Variable	Valeur
a	-7
b	3
c	14

La boucle TANT QUE

- Exemple :

```
VARIABLE
  a,b,c : entier
DEBUT
  a <- 5
  b <- 3
  c <- 0
  TANT QUE (a EST DIFFERENT DE 0) FAIRE
  →      c <- c + a + b
          a <- a - b
  FIN TANT QUE
  ECRIRE (c)
FIN
```

Variable	Valeur
a	-7
b	3
c	14

a n'aura jamais la valeur 0: la condition vaudra toujours VRAI. On reste bloqué dans la boucle.

La boucle FAIRE TANT QUE

- La boucle FAIRE ... TANT QUE est très similaire à la boucle TANT QUE

FAIRE

... // liste d'instructions à répéter

TANT QUE (condition)

La boucle REPETER TANT QUE

- Les deux type de boucles TANT QUE :

TANT QUE (expression booléenne) FAIRE

... // liste d'instructions à répéter

FIN TANT QUE

FAIRE

... // liste d'instructions à répéter

TANT QUE (expression booléenne)

La boucle *FAIRE ... TANT QUE* permet d'exécuter **au moins une fois** la liste d'instructions

La boucle POUR

- On utilise cette boucle lorsque l'on sait combien de fois on doit répéter les instructions.
- Une boucle POUR se base sur la valeur d'une variable qui sert de compteur.
- Syntaxe:

```
POUR var_iter DE val_debut À val_fin [PAS DE n] FAIRE  
    //Liste d'instructions à répéter  
    ...  
FIN POUR
```

La boucle POUR

```
POUR var_iter DE val_debut À val_fin [PAS DE n] FAIRE
    //Liste d'instructions à répéter
    ...
FIN POUR
```

- La ligne POUR contient 3 instructions condensées :
 - L'initialisation de la variable compteur ou **variable d'itération** (ici à `val_debut`)
 - La condition **d'arrêt** de la boucle: lorsque la **variable d'itération** \geq `val_fin`
 - L'incrémentation (le pas) de la variable d'itération. À chaque « tour » de boucle :

```
var_iter ← var_iter + n
```

La boucle pour

- Exemple

```
VARIABLE
  i: entier
DEBUT
  // lorsque le pas n'est pas précisé il est par défaut à 1
  // la valeur de fin est toujours exclue par défaut
  POUR i DE 0 à 10 FAIRE
    ECRIRE(i)
  FIN POUR
FIN
```

La boucle pour

- Exemple

```
VARIABLE
  i: entier
DEBUT
  // lorsque le pas n'est pas précisé il est par défaut à 1
  // la valeur de fin est toujours exclue par défaut
  POUR i DE 0 à 10 FAIRE
    ECRIRE(i)
  FIN POUR
FIN
```

```
0
1
2
3
4
5
6
7
8
9
```

La boucle s'arrête pour $i=10$, donc la dernière valeur affichée sera 9.

La boucle pour

- Exemple :

```
VARIABLES
    i,n : entier
DEBUT
→ n ← 10
    POUR i DE 2 À n PAR PAS DE 2 FAIRE
        écrire(i*i)
    FIN POUR
FIN
```

Variable	Valeur
n	
i	



La boucle pour

- Exemple :

```
VARIABLES
    i,n : entier
DEBUT
    n ← 10
→ POUR i DE 2 À n PAR PAS DE 2 FAIRE
    écrire(i*i)
    FIN POUR
FIN
```

Variable	Valeur
n	10
i	



La boucle pour

- Exemple :

```
VARIABLES
    i,n : entier
DEBUT
    n ← 10
    POUR i DE 2 À n PAR PAS DE 2 FAIRE
        → écrire(i*i)
    FIN POUR
FIN
```

Variable	Valeur
n	10
i	2



La boucle pour

- Exemple :

```
VARIABLES
    i,n : entier
DEBUT
    n ← 10
    POUR i DE 2 À n PAR PAS DE 2 FAIRE
        écrire(i*i)
    → FIN POUR
FIN
```

Variable	Valeur
n	10
i	2

```
4
```

La boucle pour

- Exemple :

```
VARIABLES
    i,n : entier
DEBUT
    n ← 10
→ POUR i DE 2 À n PAR PAS DE 2 FAIRE
    écrire(i*i)
FIN POUR
FIN
```

Variable	Valeur
n	10
i	2

4

La boucle pour

- Exemple :

```
VARIABLES
    i,n : entier
DEBUT
    n ← 10
→ POUR i DE 2 À n PAR PAS DE 2 FAIRE
    écrire(i*i)
FIN POUR
FIN
```

Variable	Valeur
n	10
i	4

```
4
```

La boucle pour

- Exemple :

```
VARIABLES
    i,n : entier
DEBUT
    n ← 10
    POUR i DE 2 À n PAR PAS DE 2 FAIRE
    →     écrire(i*i)
    FIN POUR
FIN
```

Variable	Valeur
n	10
i	4

```
4
```

La boucle pour

- Exemple

```
VARIABLES
    i,n : entier
DEBUT
    n ← 10
    POUR i DE 2 À n PAR PAS DE 2 FAIRE
        écrire(i*i)
    → FIN POUR
FIN
```

Variable	Valeur
n	10
i	4

```
4
16
```

La boucle pour

- Exemple :

```
VARIABLES
    i,n : entier
DEBUT
    n ← 10
→ POUR i DE 2 À n PAR PAS DE 2 FAIRE
    écrire(i*i)
FIN POUR
FIN
```

Variable	Valeur
n	10
i	6

```
4
16
```

La boucle pour

- Exemple :

```
VARIABLES
    i,n : entier
DEBUT
    n ← 10
    POUR i DE 2 À n PAR PAS DE 2 FAIRE
        → écrire(i*i)
    FIN POUR
FIN
```

Variable	Valeur
n	10
i	6

```
4
16
```


La boucle pour

- Exemple :

```
VARIABLES
    i,n : entier
DEBUT
    n ← 10
    POUR i DE 2 À n PAR PAS DE 2 FAIRE
        écrire(i*i)
    → FIN POUR
FIN
```

Variable	Valeur
n	10
i	6

```
4
16
36
```

La boucle pour

- Exemple :

```
VARIABLES
    i,n : entier
DEBUT
    n ← 10
→ POUR i DE 2 À n PAR PAS DE 2 FAIRE
    écrire(i*i)
FIN POUR
FIN
```

Variable	Valeur
n	10
i	8

```
4
16
36
```

La boucle pour

- Exemple :

```
VARIABLES
    i,n : entier
DEBUT
    n ← 10
    POUR i DE 2 À n PAR PAS DE 2 FAIRE
    →   écrire(i*i)
    FIN POUR
FIN
```

Variable	Valeur
n	10
i	8

```
4
16
36
```

La boucle pour

- Exemple :

```
VARIABLES
    i,n : entier
DEBUT
    n ← 10
    POUR i DE 2 À n PAR PAS DE 2 FAIRE
        écrire(i*i)
    → FIN POUR
FIN
```

Variable	Valeur
n	10
i	8

```
4
16
36
64
```

La boucle pour

- Exemple :

```
VARIABLES
    i,n : entier
DEBUT
    n ← 10
→ POUR i DE 2 À n PAR PAS DE 2 FAIRE
    écrire(i*i)
FIN POUR
FIN
```

Variable	Valeur
n	10
i	10

```
4
16
36
64
```

La boucle pour

- Exemple :

```
VARIABLES
```

```
    i,n : entier
```

```
DEBUT
```

```
    n ← 10
```

```
    POUR i DE 2 À n PAR PAS DE 2 FAIRE
```

```
        écrire(i*i)
```

```
    FIN POUR
```

```
→ FIN
```

Variable	Valeur
n	10
i	10

```
4  
16  
36  
64
```

La boucle pour

ATTENTION :

- On ne modifie jamais les bornes de début et de fin dans la boucle
- On ne modifie jamais la valeur de l'itérateur dans la boucle
- On n'arrête jamais un pour avant sa fin prévue

Si vous avez besoin de transgresser un de ces principes, c'est que vous ne devez pas utiliser une boucle pour.

Comparaison des boucles

	Nombre de tours connu	Au moins un tour	Peut être infinie
TANT QUE	Non	Non	Oui
REPETER TANT QUE	Non	Oui	Oui
POUR	Oui	Non	Oui mais plus facile à éviter !

- Une boucle POUR peut toujours être transformée en boucle TANT QUE, mais pas l'inverse ! (voir TD)
- La boucle POUR est plus courte à écrire => si on connaît le nombre de tours de boucle à réaliser, il faut utiliser une boucle POUR !

Application

- Comment gérer le problème du début de cours ?
- Quelle boucle est la plus adaptée ?

```
VARIABLE
  n1: Réel
DEBUT
  ECRIRE (“Donnez la note”)
  LIRE(n1)
  ECRIRE(“Le résultat est” + n1)
FIN
```

Application

- La boucle POUR ne fonctionne pas : on ne peut pas prévoir le nombre de fois où l'utilisateur va se tromper !

```
VARIABLE
  n1: Réél
  i: entier
DEBUT
  POUR i de 0 à 10
    ECRIRE (“Donnez la note”)
    LIRE(n1)
  FIN POUR
  ECRIRE(“Le résultat est” + n1)
FIN
```

Application

- La boucle POUR ne fonctionne pas : on ne peut pas prévoir le nombre de fois où l'utilisateur va se tromper !

```
VARIABLE
  n1: Réél
  i: entier
DEBUT
  POUR i de 0 à 10
    ECRIRE ("Donnez la note")
    LIRE(n1)
  FIN POUR
  ECRIRE("Le résultat est" + n1)
FIN
```

Ici on va demander 10 fois à l'utilisateur de saisir la note indépendamment de si les saisies sont correctes ou non !

Application

- Boucle TANT QUE

```
VARIABLE
  n1: Réel
DEBUT
  ECRIRE (“Donnez la note”)
  LIRE(n1)
  TANT QUE (          )
    ECRIRE (“Donnez la note”)
    LIRE(n1)
  FIN TANT QUE
  ECRIRE(“Le résultat est” + n1)
FIN
```

Application

- Boucle TANT QUE

```
VARIABLE
  n1: Réel
DEBUT
  ECRIRE (“Donnez la note”)
  LIRE(n1)
  TANT QUE (n1 STRICTEMENT INFÉRIEUR A 0 OU n1 STRICTEMENT SUPÉRIEUR A 20)
    ECRIRE (“Donnez la note”)
    LIRE(n1)
  FIN TANT QUE
  ECRIRE(“Le résultat est” + n1)
FIN
```

Application

- Boucle TANT QUE

```
VARIABLE
  n1: Réel
DEBUT
  // il faut donner une valeur à n1 avant de rentrer dans la boucle
  ECRIRE ("Donnez la note")
  LIRE(n1)
  TANT QUE (n1 STRICTEMENT INFÉRIEUR A 0 OU n1 STRICTEMENT SUPÉRIEUR A 20)
    ECRIRE ("Donnez la note")
    LIRE(n1)
  FIN TANT QUE
  ECRIRE("Le résultat est" + n1)
FIN
```

Application

- Boucle REPETER TANT QUE

```
VARIABLE
  n1: Réel
DEBUT
  // Pas besoin d'initialiser n1
  REPETER
    ECRIRE ("Donnez la note")
    LIRE(n1)
  TANT QUE(n1 STRICTEMENT INFÉRIEUR A 0 OU n1 STRICTEMENT SUPÉRIEUR A 20)
  ECRIRE("Le résultat est" + n1)
FIN
```

III. Langage C

Boucle TANT QUE

```
Tant que (condition) faire  
...  
Fin tant que
```



```
while (condition) {  
    ...  
}
```

```
Répéter  
...  
Tant que (condition)
```



```
do {  
    ...  
} while (condition);
```

Boucle POUR

- Syntaxe :

```
for(initialisation; condition; itération) {  
    instructions;  
}
```

- Déroulement :
 1. exécute *initialisation*
 2. vérifie *condition*, si faux -> sortie
 3. exécute *instructions*
 4. exécute *itération*
 5. retour à 2

```
for (i=1; i<=n; i++) {  
    instructions;  
}
```

Boucle POUR

```
Pour i de 1 à n faire  
    ...  
Fin pour
```

```
for (i=1; i<n; i++) {  
    ...  
}
```

```
Pour i de 1 à n+1 par pas de 2  
faire  
    ...  
Fin pour
```

```
for (i=1; i<=n; i=i+2) {  
    ...  
}
```

Boucle POUR

- Contrairement au pseudo-code, l'initialisation et l'itération de la boucle **for** peut contenir des **blocs** d'instructions.
- Les instructions de ces blocs doivent être séparés par des virgules.
- Exemple :

```
for (i=1, j=-5; i<=n; i=i+2, j=j*3) {  
    ...  
}
```

Application

```
VARIABLE
  n1: Réel
DEBUT
  // Pas besoin d'initialiser n1
  REPETER
    ECRIRE ("Donnez la note")
    LIRE(n1)
  TANT QUE(n1 STRICTEMENT
INFERIEUR A 0 OU n1 STRICTEMENT
SUPPERIEUR A 20)
    ECRIRE("Le résultat est" + n1)
FIN
```

Application

```
VARIABLE
  n1: Réel
DEBUT
  // Pas besoin d'initialiser n1
  REPETER
    ECRIRE ("Donnez la note")
    LIRE(n1)
  TANT QUE(n1 STRICTEMENT
  INFÉRIEUR A 0 OU n1 STRICTEMENT
  SUPÉRIEUR A 20)
    ECRIRE("Le résultat est" + n1)
FIN
```



```
int main(){
  float n1;
  do{
    printf("Donnez la note");
    scanf("%f",&n1);
  } while(n1<0 || n1 >20);
  printf("Le résultat est %f",n1);
}
```

Application

```
int main(){
    int i,n;
    printf("Donnez un nombre");
    scanf("%d",&n)
    for (i=1; i<=n;i++){
        printf("%d", i*i);
    }
    return 0;
```

Application

```
int main(){
    int i,n;
    printf("Donnez un nombre");
    scanf("%d",&n)
    for (i=1; i<=n;i++){
        printf("%d", i*i);
    }
    return 0;
```

Ce code permet d'afficher le carré des n premiers nombres entiers.

Conclusion

- Les boucles sont très utilisées en algorithmie : elles permettent de répéter des instructions plusieurs fois.
- Il existe plusieurs types de boucles différentes : **POUR** (for), **TANT QUE** (while) et **FAIRE ... TANT QUE** (do ... while). A chaque situation correspond le bon type de boucle.
- Les boucles associées aux branchements conditionnels constituent la base de l'algorithmie !

