CHAPITRE 5

Les boucles : TANTQUE et POUR

Répéter sans réécrire



AU PROGRAMME

1 Pourquoi des boucles ?
2 La boucle TANTQUE
3 Compter et accumuler
4 Les pièges à éviter
5 La boucle POUR

Nom : ..... **Classe :** .....

AVANT DE COMMENCER

Ce que tu vas apprendre

Redemander une réponse tant qu'elle n'est pas correcte, afficher une table de multiplication, faire durer un jeu tant que la partie n'est pas finie : tout cela demande de répéter des instructions. C'est le rôle des boucles, la quatrième et dernière famille d'instructions.

1. **Expliquer à quoi sert une boucle.**
2. **Écrire une boucle `TANTQUE … FINTANTQUE` et préparer sa variable avant d'entrer.**
3. **Contrôler une saisie avec une boucle.**
4. **Utiliser un compteur et un accumulateur.**
5. **Repérer une boucle infinie et une boucle qui ne démarre jamais.**
6. **Écrire une boucle `POUR` quand le nombre de tours est connu.**

PARTIE 1

Pourquoi des boucles ?

Un programme demande de répondre par « O » (oui) ou « N » (non). L'utilisateur se trompe et tape autre chose. Avec un `SI`, on peut redemander **une** fois… mais s'il se trompe encore ? Il faudrait un deuxième `SI`, un troisième… sans jamais savoir combien en prévoir.

DÉFINITION

Une boucle

Une structure qui fait exécuter **plusieurs fois** la même série d'instructions, tant qu'une condition le demande. On parle aussi de structure **répétitive** ou **itérative**.

À quoi servent les boucles ?

Le contrôle de saisie
Redemander tant que la réponse n'est pas valable.

Les tours de jeu
Recommencer tant que la partie n'est pas finie.

Les calculs répétés
Additionner, compter, afficher une table de multiplication.

POUR ALLER PLUS LOIN

Dans un tableur, tu connais déjà les variables (les cellules) et les tests (la fonction SI). Les boucles, elles, sont propres à la programmation : c'est elles qui rendent un programme vraiment dynamique.

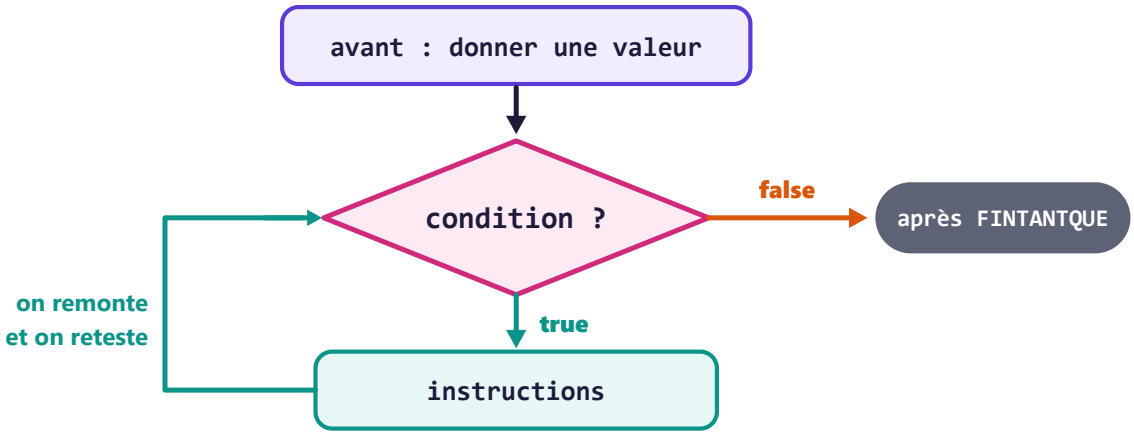
PARTIE 2

La boucle TANTQUE

PSEUDOCODE La structure

```
TANTQUE condition
    instructions
FINTANTQUE
```

Le programme teste la condition. Si elle est **vraie**, il exécute les instructions, puis **remonte** et teste à nouveau. Dès que la condition est **fausse**, il saute après le `FINTANTQUE`.



La condition est testée AVANT chaque passage : si elle est fausse dès le départ, on n'entre jamais.

Figure 5.1 — La boucle TANTQUE : on teste, on exécute, on remonte.

ATTENTION

Pas de FAIRE

La ligne du `TANTQUE` ne contient que la condition : pas de `FAIRE`, pas de `ALORS`. Dans VS Code, un `FAIRE` ajouté casse la traduction.

Le contrôle de saisie

PSEUDOCODE

```
variable reponse en chaine de caracteres
DEBUT
    ECRIRE "Veux-tu continuer ? (O/N)"
    LIRE reponse
    TANTQUE reponse <> "O" ET reponse <> "N"
        ECRIRE "Réponds par O ou par N :"
        LIRE reponse
    FINTANTQUE
    ECRIRE "Merci, tu as répondu " & reponse & "."
FIN
```

À L'ÉCRAN

```
Veux-tu continuer ? (O/N)
peut-être
Réponds par O ou par N :
oui
Réponds par O ou par N :
N
Merci, tu as répondu N.
```

À RETENIR

Préparer la condition avant d'entrer

- La variable testée doit **avoir une valeur** avant le `TANTQUE` : une première lecture, ou une valeur de départ.
- À l'intérieur, une instruction doit **pouvoir changer** cette variable, sinon on ne sort jamais.

PARTIE 3

Compter et accumuler

DÉFINITION

Un compteur

Une variable qui augmente de 1 à chaque tour : `i <- i + 1`. On lit : « i prend l'ancienne valeur de i, plus 1 ».

PSEUDOCODE **Une table de multiplication (les 5 premières lignes)**

```
variables n, i en entier
DEBUT
    ECRIRE "Quelle table veux-tu ?"
    LIRE n
    i <- 1
    TANTQUE i <= 5
        ECRIRE i & " x " & n & " = " & i * n
        i <- i + 1
    FINTANTQUE
FIN
```

À L'ÉCRAN

```
Quelle table veux-tu ?
7
1 x 7 = 7
2 x 7 = 14
3 x 7 = 21
4 x 7 = 28
5 x 7 = 35
```

DÉFINITION

Un accumulateur

Une variable qui **additionne** des valeurs au fil des tours : `total <- total + note`. Elle commence à 0.

PSEUDOCODE **Additionner des notes jusqu'à -1**

```
variables note, total, nb en entier
DEBUT
    total <- 0
    nb <- 0
    ECRIRE "Note (ou -1 pour terminer) ?"
    LIRE note
    TANTQUE note <> -1
        total <- total + note
        nb <- nb + 1
        ECRIRE "Note (ou -1 pour terminer) ?"
        LIRE note
    FINTANTQUE
    ECRIRE nb & " notes, total : " & total
FIN
```

À L'ÉCRAN

```
Note (ou -1 pour terminer) ?
12
Note (ou -1 pour terminer) ?
15
Note (ou -1 pour terminer) ?
9
Note (ou -1 pour terminer) ?
-1
3 notes, total : 36
```

Suivre une boucle à la trace

Pour la table de 7, une ligne par tour : on note la valeur de `i` au moment du test, le résultat du test et ce qui s'affiche.

Tour	i (au test)	i <= 5 ?	Affiché
1	1	true	1 x 7 = 7
2	2	true	2 x 7 = 14
…	…	…	…
5	5	true	5 x 7 = 35
fin	6	false	on sort de la boucle

PARTIE 4

Les pièges à éviter

La boucle infinie

La condition de sortie n'est **jamais** remplie : le programme tourne sans fin. Cause habituelle : on a oublié de modifier la variable dans la boucle.

La boucle qui ne démarre pas

La condition est **fausse dès le départ** : les instructions ne sont jamais exécutées, même une fois.

PSEUDOCODE **Boucle infinie : où est l'erreur ?**

```
i <- 1
TANTQUE i <= 5
    ECRIRE i
FINTANTQUE
```

`i` vaut 1 et ne change jamais : `i <= 5` reste vrai pour toujours. Il manque `i <- i + 1` dans la boucle. Dans VS Code, une boucle infinie s'arrête avec **Ctrl + C** dans le terminal.

PSEUDOCODE **Boucle qui ne démarre jamais**

```
i <- 10
TANTQUE i <= 5
    ECRIRE i
    i <- i + 1
FINTANTQUE
```

À RETENIR

Avant de lancer une boucle, vérifie

- La valeur de départ : la condition est-elle vraie au premier test ?
- Le changement : une instruction modifie-t-elle la variable testée ?
- La sortie : cette modification finit-elle par rendre la condition fausse ?

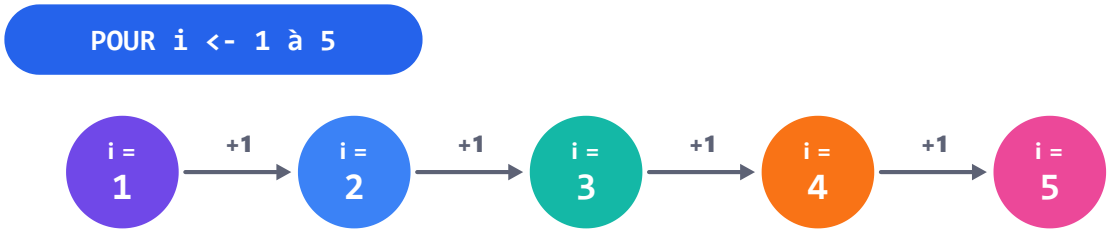
PARTIE 5

La boucle POUR

Quand on sait **à l'avance** combien de tours faire, la boucle `POUR` est plus simple : elle gère elle-même le compteur. C'est un cas particulier du `TANTQUE`.

PSEUDOCODE **La structure (comme dans VS Code)**

```
POUR i <- debut à fin
    instructions
POUR SUIVANT
FINPOUR
```



Le compteur avance tout seul de 1 à chaque passage : 5 passages, de i = 1 à i = 5.

Figure 5.2 — Le compteur part de la valeur de début et avance de 1 à chaque tour.

PSEUDOCODE **La table de multiplication complète**

```
variables n, i en entier
DEBUT
    ECRIRE "Quelle table veux-tu ?"
    LIRE n
    POUR i <- 1 à 10
        ECRIRE i & " x " & n & " = " & i * n
    POUR SUIVANT
    FINPOUR
FIN
```

À L'ÉCRAN

```
Quelle table veux-tu ?
3
1 x 3 = 3
2 x 3 = 6
3 x 3 = 9
4 x 3 = 12
5 x 3 = 15
6 x 3 = 18
7 x 3 = 21
8 x 3 = 24
9 x 3 = 27
10 x 3 = 30
```

À RETENIR

Les règles du POUR

- Le compteur est **déclaré** comme une autre variable (`entier`).
- On ne modifie **jamais** le compteur à l'intérieur : le `POUR` s'en charge.
- Le début et la fin sont un nombre ou une variable : `POUR i <- 1 à n`.

Situation	Boucle à choisir
Le nombre de tours est connu (10 lignes, n élèves…)	`POUR`
On ne sait pas combien de tours (saisie, jeu, « jusqu'à -1 »)	`TANTQUE`
Compter de 2 en 2, à rebours…	`TANTQUE` avec `i <- i + 2`, `i <- i - 1`…

POUR ALLER PLUS LOIN

Sur papier, certains livres écrivent un « pas » : `Pour i de 10 à 1 Pas -1`. L'extension de VS Code ne le connaît pas : le `POUR` avance toujours de 1. Pour un autre pas, utilise un `TANTQUE`.

BILAN DU CHAPITRE 5

L'essentiel à retenir

1. Une **boucle** répète une série d'instructions ; c'est la quatrième famille d'instructions.
2. `TANTQUE condition … FINTANTQUE` répète **tant que** la condition est vraie ; elle est testée avant chaque tour.
3. La variable testée doit avoir une valeur **avant** la boucle, et changer **dans** la boucle.
4. Le **contrôle de saisie** redemande tant que la réponse n'est pas valable.
5. Un **compteur** augmente de 1 (`i <- i + 1`) ; un **accumulateur** additionne (`total <- total + note`), en partant de `0`.
6. Pièges : la **boucle infinie** (rien ne change) et la boucle qui **ne démarre jamais** (fausse dès le départ).
7. `POUR i <- 1 à n … POUR SUIVANT … FINPOUR` quand le nombre de tours est connu ; sinon `TANTQUE`.

Le lexique

Boucle
Structure qui répète des instructions (`TANTQUE`, `POUR`).

Itération
Un tour de boucle.

Contrôle de saisie
Boucle qui redemande une valeur tant qu'elle n'est pas valable.

Compteur
Variable qui augmente de 1 à chaque tour.

Accumulateur
Variable qui additionne des valeurs au fil des tours.

Boucle infinie
Boucle dont la condition reste toujours vraie : elle ne s'arrête jamais.

Valeur sentinelle
Valeur spéciale qui arrête la saisie, comme -1.

Je fais le point

Coche une case par ligne : où en es-tu avant l'interro ?

Je suis capable de…	Oui	Presque	Pas encore
Écrire un `TANTQUE` correct, sans `FAIRE`.	☐	☐	☐
Contrôler une saisie avec une boucle.	☐	☐	☐
Utiliser un compteur et un accumulateur.	☐	☐	☐
Suivre une boucle tour par tour dans un tableau de trace.	☐	☐	☐
Repérer et corriger une boucle infinie.	☐	☐	☐
Choisir entre `POUR` et `TANTQUE`.	☐	☐	☐

CHAPITRE 5 · EXERCICES

Teste tes connaissances

avant l'interro

COMMENT T'Y PRENDRE

1. **Relis la synthèse** : l'essentiel, le lexique et « Je fais le point ».
2. **Fais les exercices sans le cours**, comme à l'interro.
3. **Corrige-toi** avec la correction vue en classe ou sur le site.
4. **Reviens sur tes erreurs** : relis la partie du cours concernée.

LES 5 EXERCICES

1. **Les boucles**
2. **Le tableau de trace d'une boucle**
3. **Les pièges**
4. **Corrige la boucle**
5. **À toi de boucler**

Exercice 1 Les boucles

▸ **Coche la bonne réponse.**

1. Quand la condition d'un `TANTQUE` est-elle testée ?

☐ après chaque tour ☐ avant chaque tour ☐ une seule fois, au début

2. Quelle boucle choisir pour afficher les nombres de 1 à 100 ?

☐ POUR ☐ TANTQUE ☐ SI

3. Quelle boucle choisir pour redemander un mot de passe jusqu'à ce qu'il soit correct ?

☐ POUR ☐ TANTQUE

4. Que vaut un accumulateur avant la boucle, en général ?

☐ 1 ☐ 0 ☐ rien

Exercice 2 Le tableau de trace d'une boucle

▸ **Suis la boucle tour par tour.**

PSEUDOCODE

```
variables i, total en entier
DEBUT
    total <- 0
    i <- 1
    TANTQUE i <= 4
        total <- total + i
        i <- i + 1
    FINTANTQUE
    ECRIRE total
FIN
```

À L'ÉCRAN

```
10
```

Tour	i (au test)	total (après le tour)
1		
2		
3		
4		
fin		

Exercice 3 Les pièges

- **Coche vrai ou faux. Si c'est faux, corrige la phrase sur la ligne.**

Affirmation	Vrai	Faux
Une boucle `TANTQUE` s'exécute toujours au moins une fois.	☐	☐
Oublier `i <- i + 1` dans une boucle `TANTQUE i <= 10` provoque une boucle infinie.	☐	☐
On peut écrire `TANTQUE i <= 10 FAIRE` dans VS Code.	☐	☐
Dans un `POUR`, on ne modifie pas le compteur soi-même.	☐	☐

Exercice 4 Corrige la boucle

- **Cet algorithme devrait afficher un compte à rebours de 10 à 1.**

```pseudocode
variable compte en entier
DEBUT
    compte <- 10
    TANTQUE compte > 0
        ECRIRE compte
    FINTANTQUE
    ECRIRE "Décollage !"
FIN
```

1. Que se passe-t-il quand on lance cet algorithme ?

.....

.....

2. Quelle instruction faut-il ajouter, et où ?

.....

.....

Exercice 5 À toi de boucler

- **Écris l'algorithme complet : déclarations, DEBUT, instructions, FIN.**

1. Demande une note entre 0 et 20 ; tant qu'elle n'est pas dans cet intervalle, redemande-la. Affiche ensuite « Note enregistrée ».

EXEMPLE D'ÉCRAN

```
Note sur 20 ?
25
Erreur : entre une note entre 0 et 20.
14
Note enregistrée : 14
```

2. Demande combien de nombres l'utilisateur veut additionner, puis demande-les un par un avec une boucle `POUR` et affiche leur somme.

EXEMPLE D'ÉCRAN

```
Combien de nombres ?
3
Nombre 1 ?
4
Nombre 2 ?
10
Nombre 3 ?
6
Somme : 20
```