



## ISCC – J06

PHP 02 – Fonctions et boucles

# Fonctions

Bloc de code réutilisable et modulaire.

# Fonction d'affichage

Les fonctions sont des bouts de code réutilisables

Nom de la fonction



```
function affichage() {  
    echo "Hello World";  
}
```

# Fonction d'affichage

Les fonctions sont des bouts de code réutilisables

Nom de la fonction


```
function affichage() {  
    echo "Hello World";  
}
```

Contenu de la fonction

# Fonction d'affichage

Les fonctions sont des bouts de code réutilisables

**Appel de la fonction**



```
affichage();  
affichage();
```

# Paramètres de fonctions

"Argument" de la fonction



```
function affichage($variable) {  
    echo "$variable World";  
}
```

- Les fonctions peuvent avoir des "arguments"

# Paramètres de fonctions

- Les fonctions peuvent avoir des "arguments"
- On peut utiliser ces "arguments" dans le code de la fonction

```
function affichage($variable) {  
    echo "$variable World";  
}
```

"Argument" de la fonction

Utilisation de l'argument  
envoyé à la fonction

# Paramètres de fonctions

- Les fonctions peuvent avoir des "arguments"
- On peut utiliser ces "arguments" dans le code de la fonction

```
function affichage($variable) {  
    echo "$variable World";  
}
```

```
affichage("Bonjour"); /* résultat: Bonjour World */
```



Envoi de "Bonjour" en argument



# Paramètres de fonctions

- Les fonctions peuvent avoir des "arguments"
- On peut utiliser ces "arguments" dans le code de la fonction

```
function affichage($variable) {  
    echo "$variable World";  
}
```

```
affichage("Bonjour"); /* résultat: Bonjour World */
```

Envoi de "Bonjour" en argument



Résultat:

Bonjour World



# Paramètres de fonctions

- Les fonctions peuvent avoir des "arguments"
- On peut utiliser ces "arguments" dans le code de la fonction
- On peut envoyer des variables en tant qu'argument

```
function affichage($variable) {  
    echo "$variable World";  
}  
  
affichage("Bonjour"); /* résultat: Bonjour World */  
  
$mot = "Salut";  
affichage($mot); /* résultat: Salut World */
```

Envoi de la variable \$mot en argument qui contient "Salut"

# Paramètres de fonctions

- Les fonctions peuvent avoir des "arguments"
- On peut utiliser ces "arguments" dans le code de la fonction
- On peut envoyer des variables en tant qu'argument

```
function affichage($variable) {  
    echo "$variable World";  
}  
  
affichage("Bonjour"); /* résultat: Bonjour World */  
  
$mot = "Salut";  
affichage($mot); /* résultat: Salut World */
```

Envoi de la variable \$mot en argument qui contient "Salut"



Résultat:

Salut World



```
function addition($a, $b) {  
    return $a + $b;  
}
```



**Ce que "retourne" la fonction  
après le mot-clé return**

# Retours de fonction

Une fonction peut "retourner" une valeur.

```
function addition($a, $b) {  
    return $a + $b;  
}
```

Ce que "retourne" la fonction  
après le mot-clé return

Variable

```
$c = addition(5, 10);  
  
/* c est égal à 15 */
```

# Retours de fonction

Une fonction peut "retourner" une valeur.  
On peut stocker dans une variable le "retour" d'une fonction.

```
function addition($a, $b) {  
    return $a + $b;  
}
```

Ce que "retourne" la fonction  
après le mot-clé return

Variable

```
$c = addition(5, 10);
```

```
/* c est égal à 15 */
```

```
$o = addition($c, 10);
```

```
/* o est égal à 25 */
```

# Retours de fonction

Une fonction peut "retourner" une valeur.

On peut stocker dans une variable le "retour" d'une fonction.

On peut envoyer une variable en "argument" à une fonction.

# Fonctions intégrées au PHP

Le PHP a des fonctions dites "natives" qui sont utilisables de base comme:

- [echo](#) - Affiche une chaîne de caractères
- [count](#) - Compte tous les éléments d'un tableau ou quelque chose d'un objet

- Et plein [d'autres](#)...





# Fonction manipulation de chaines de caractères

Le PHP contient beaucoup de fonctions manipulant [les chaines de caractères](#) tel que :

- [strlen](#) — Calcule la taille d'une chaîne
- [str\\_word\\_count](#) — Compte le nombre de mots utilisés dans une chaîne
- [strtolower](#) — Renvoie une chaîne en minuscules
- Et plein [d'autres...](#)



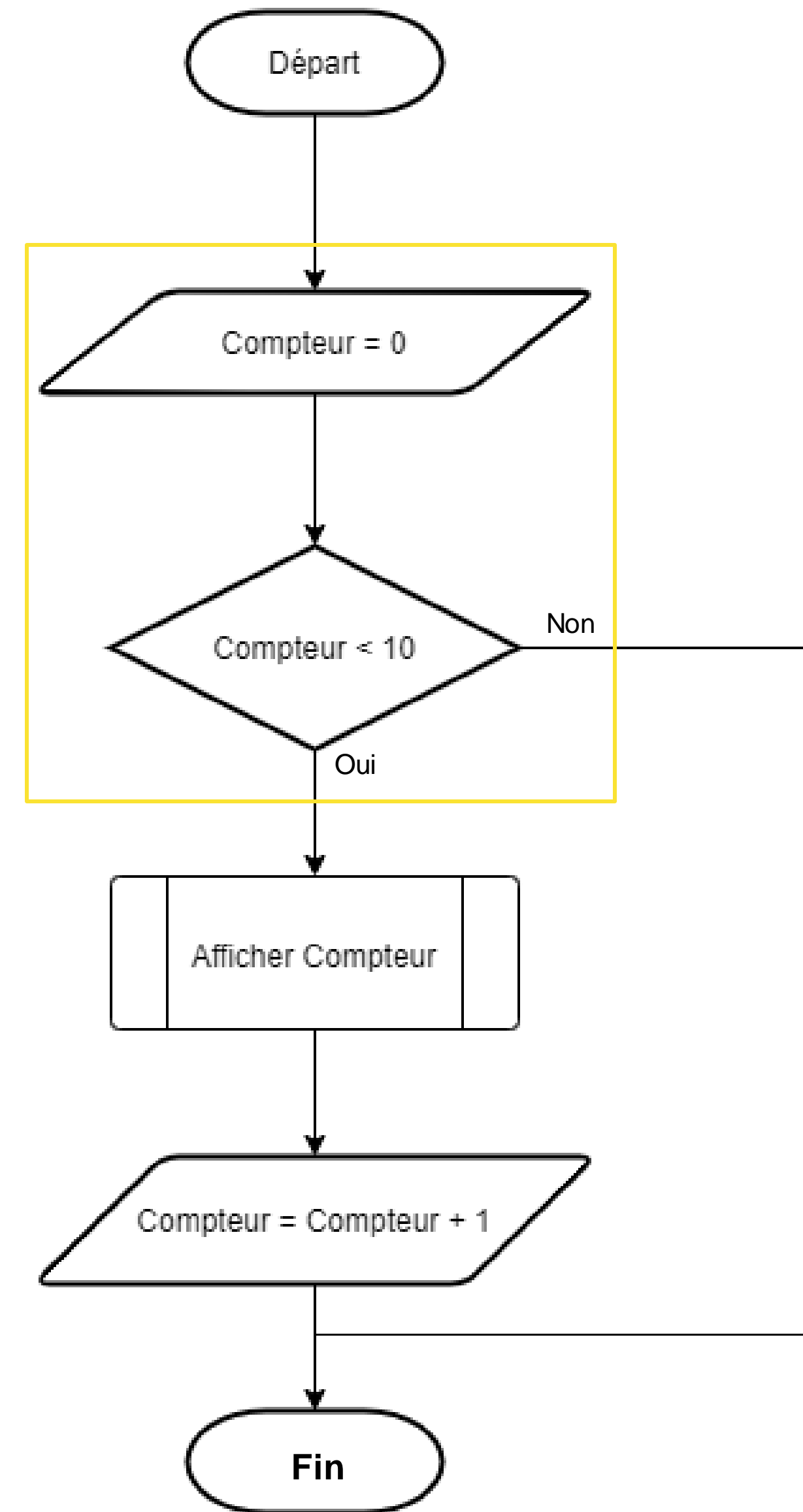
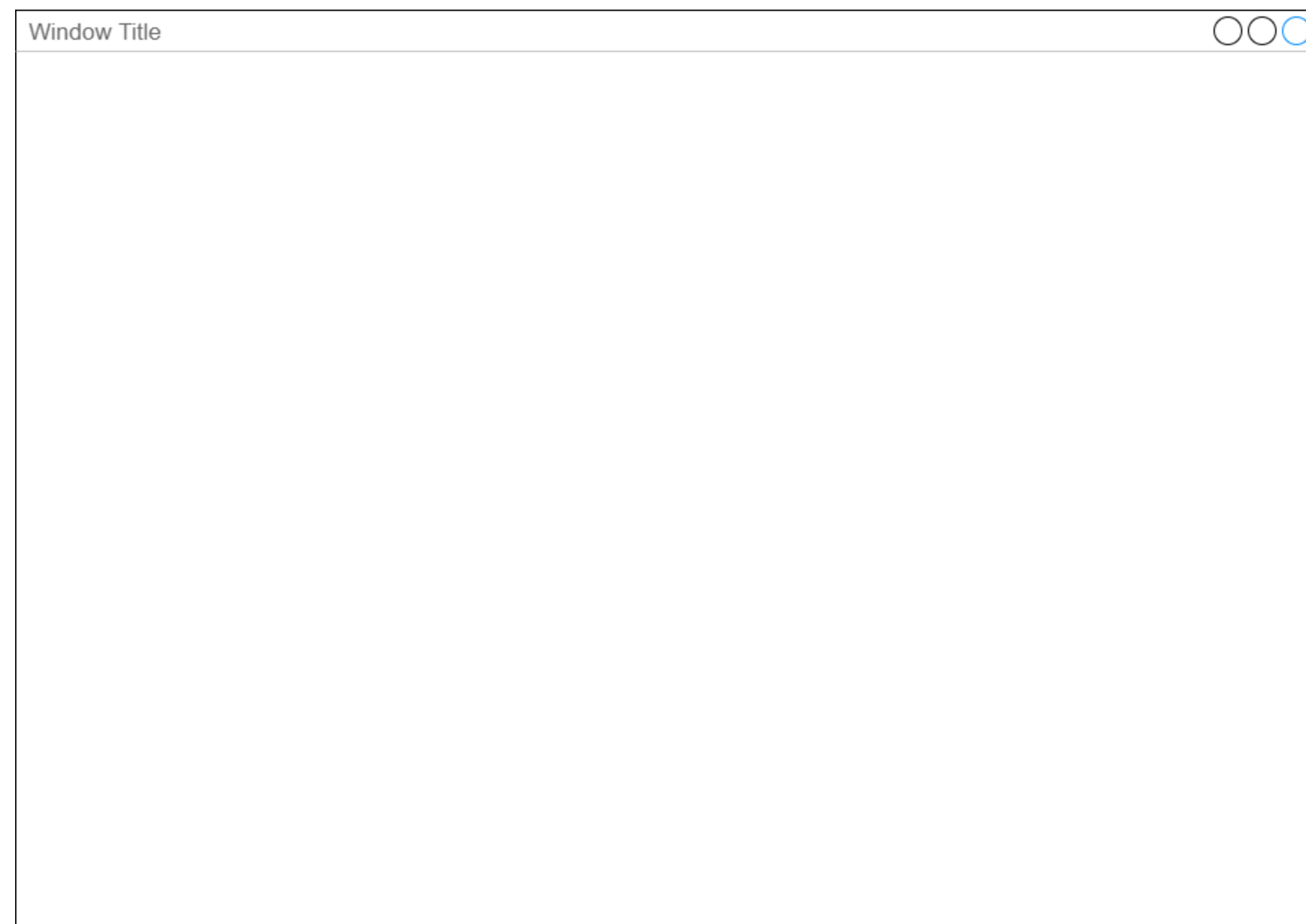


# Boucles

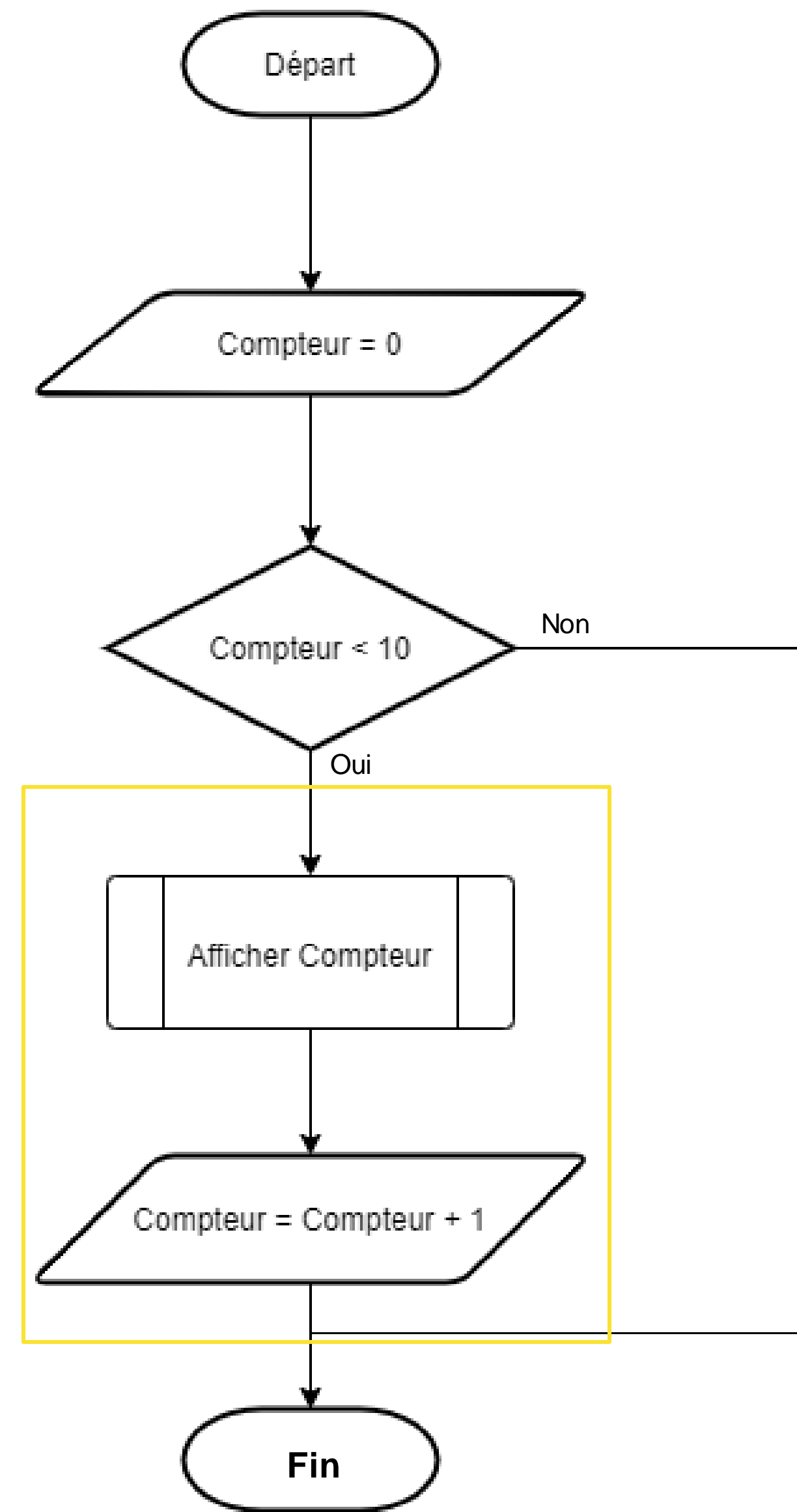
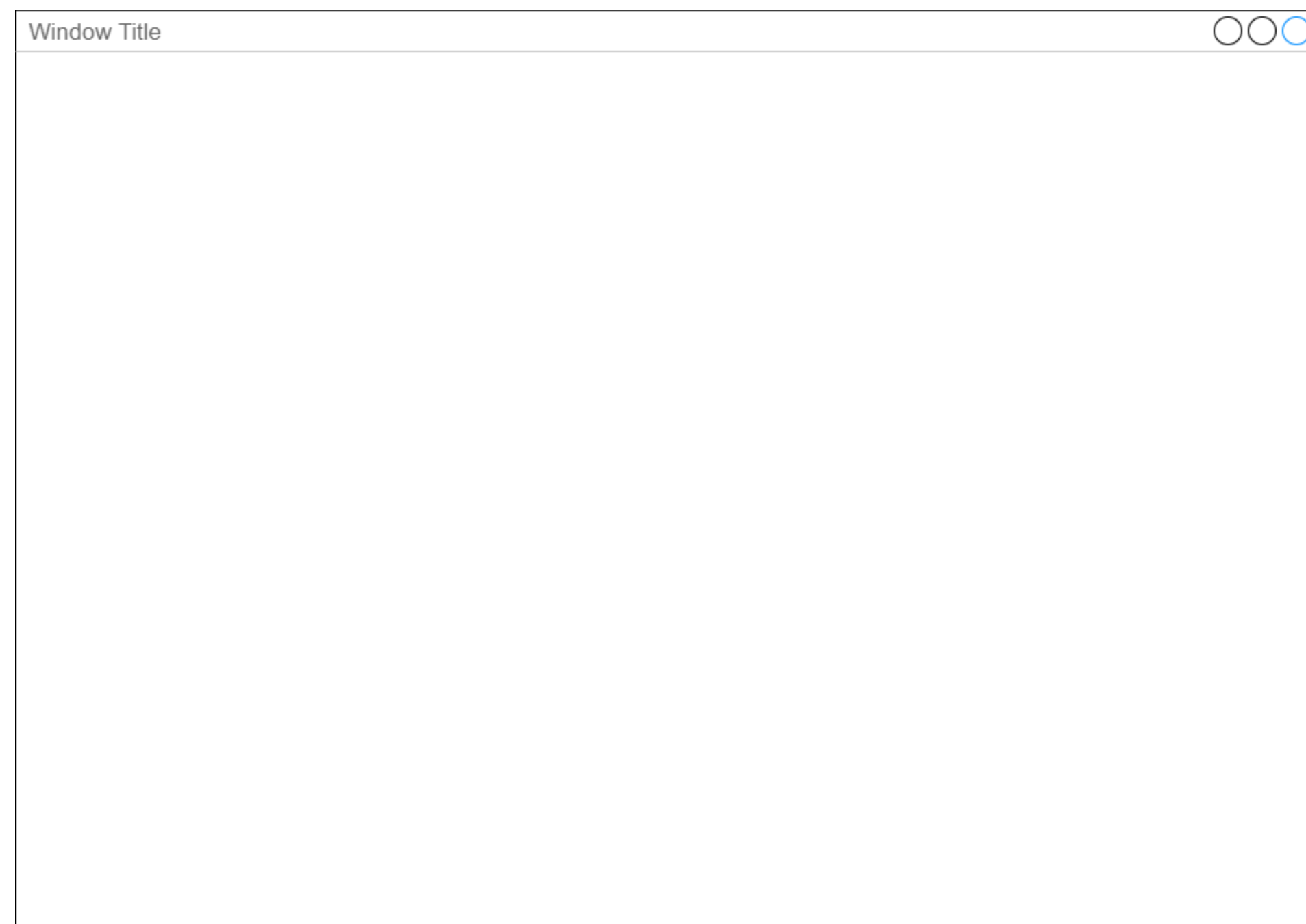
*I should have used loops*  
*I should have used loops*  
*I should have used loops*  
*I should have used loops*  
*I should have used loops*  
*I should have used loops*  
*I should have used loops*  
*I should have used loops*



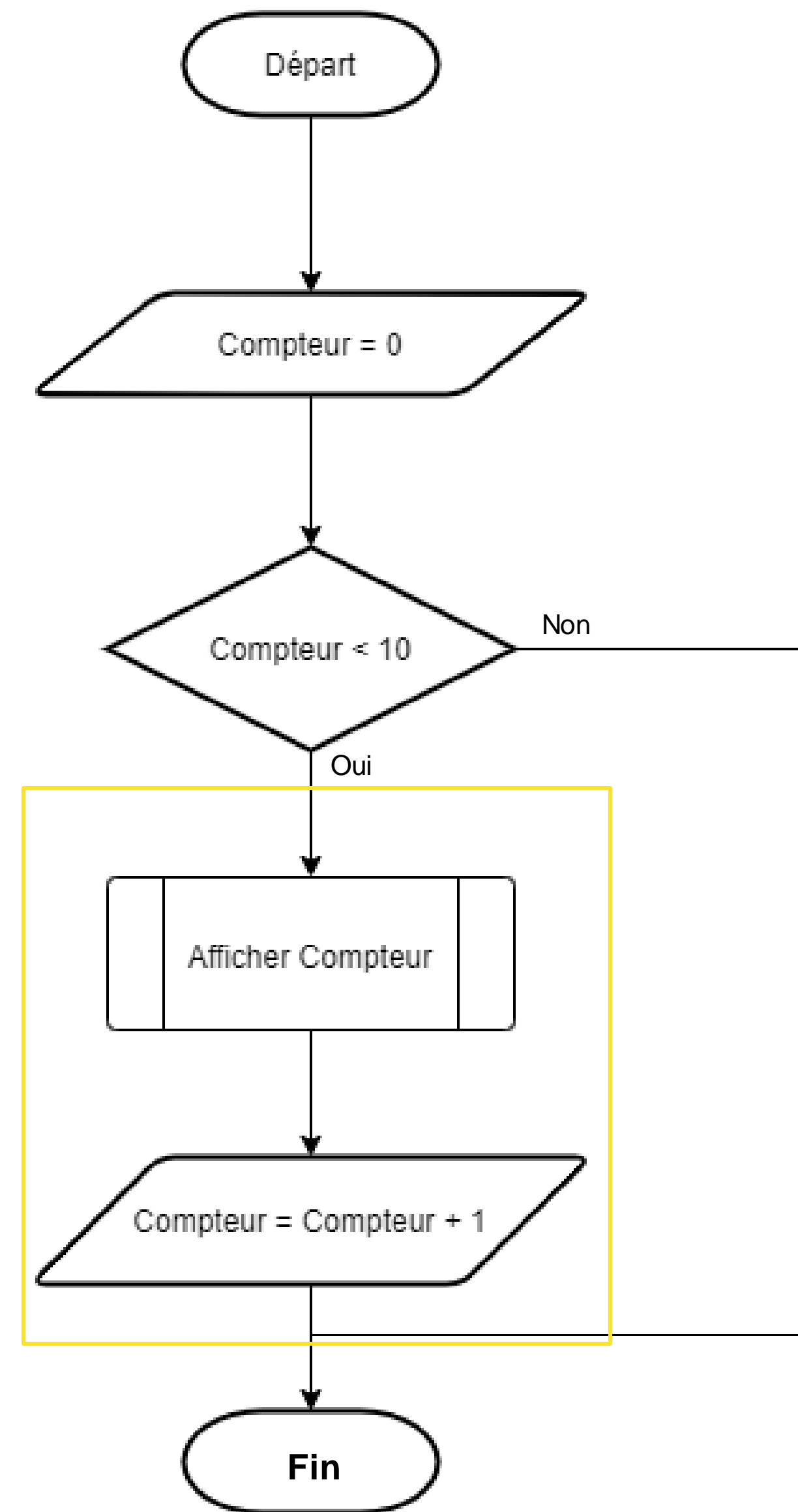
# Les conditions



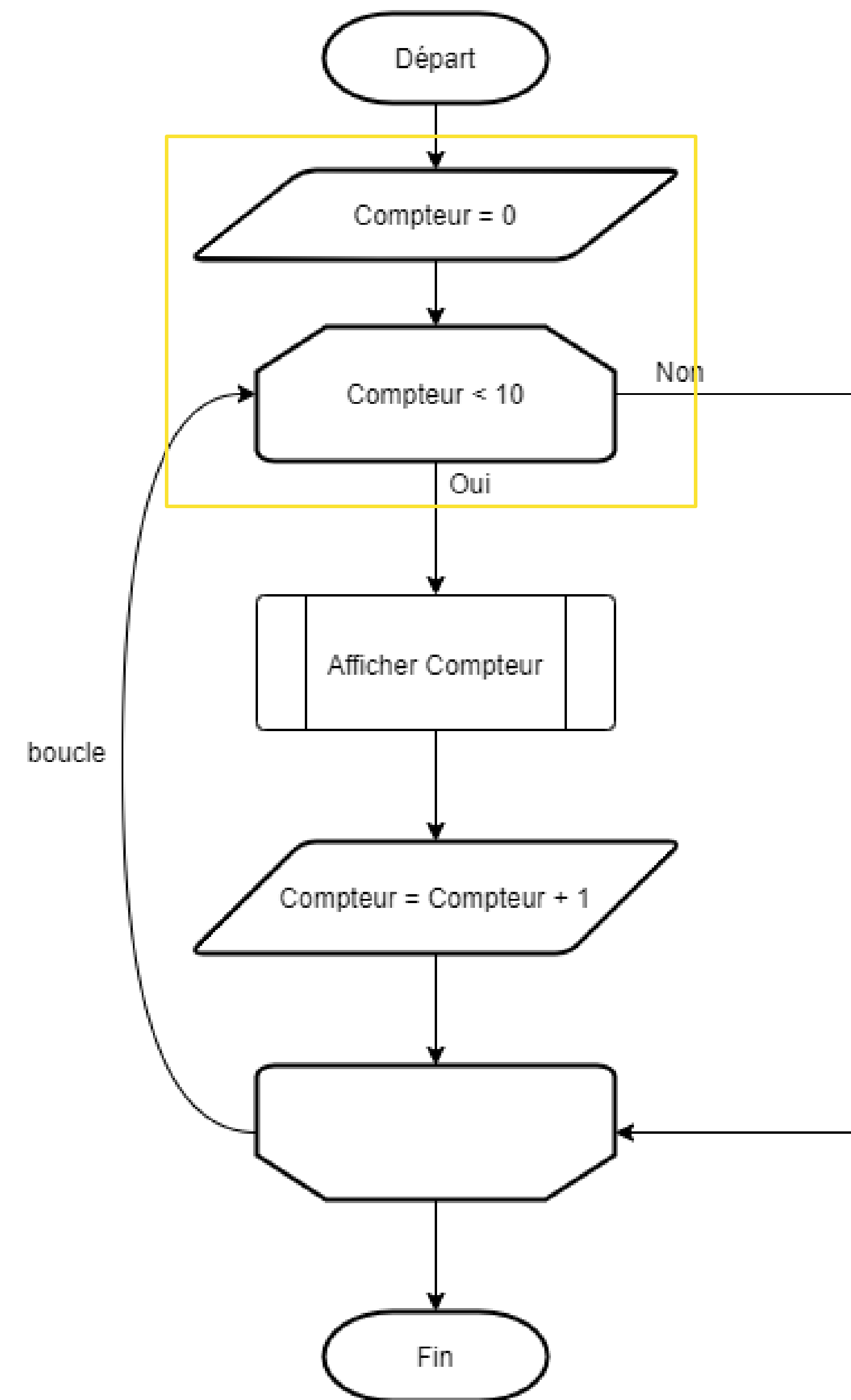
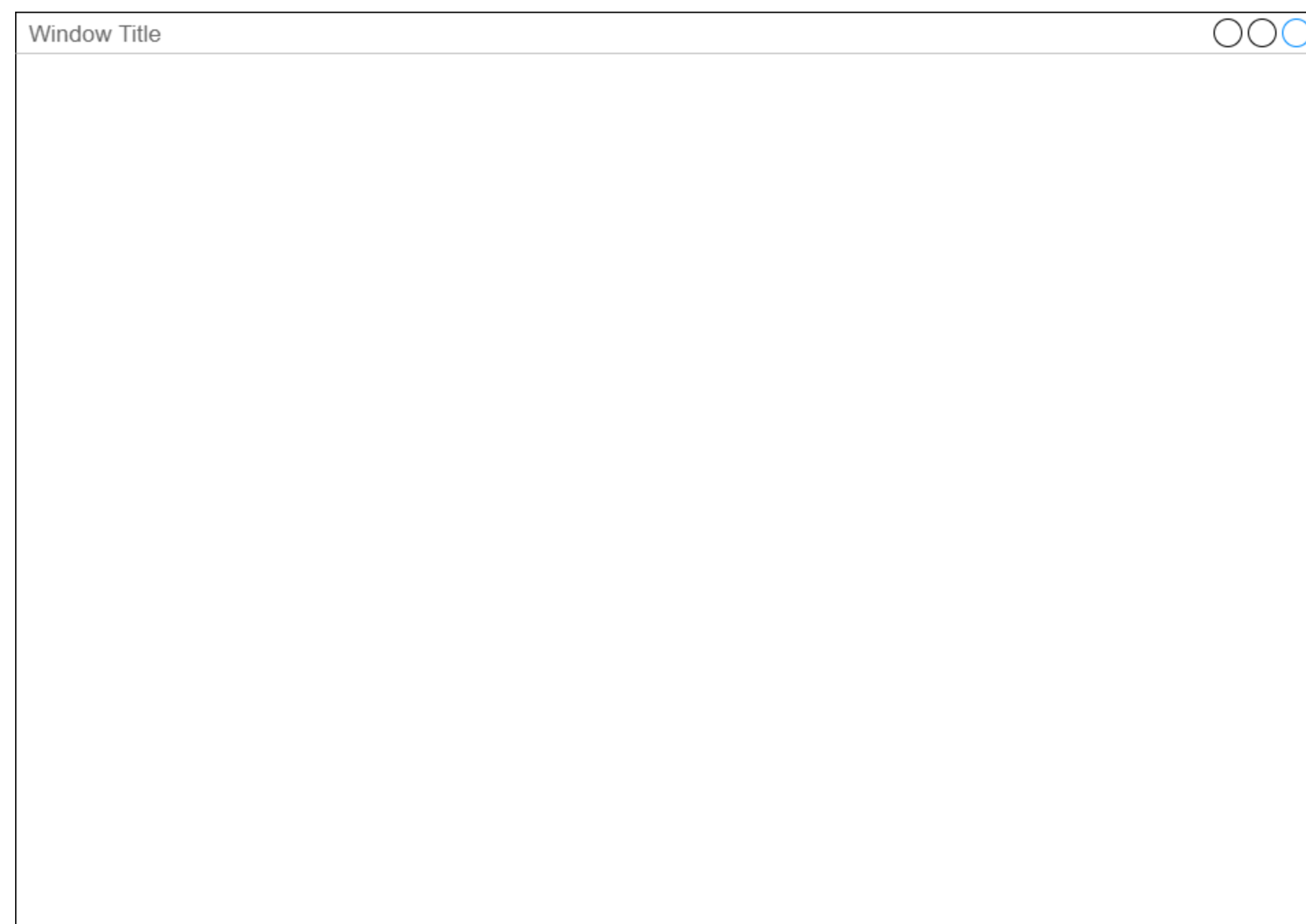
# Les conditions



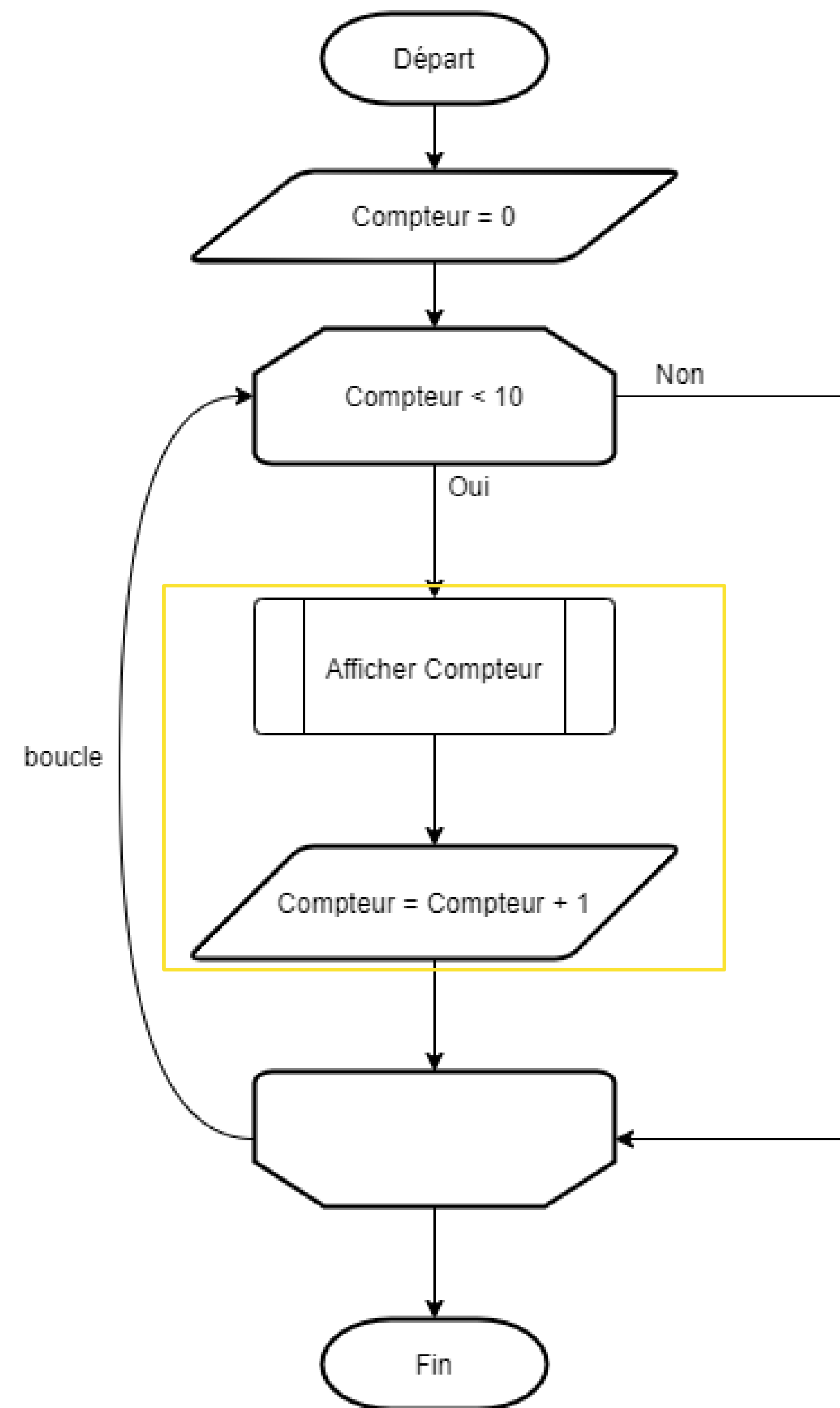
# Les conditions



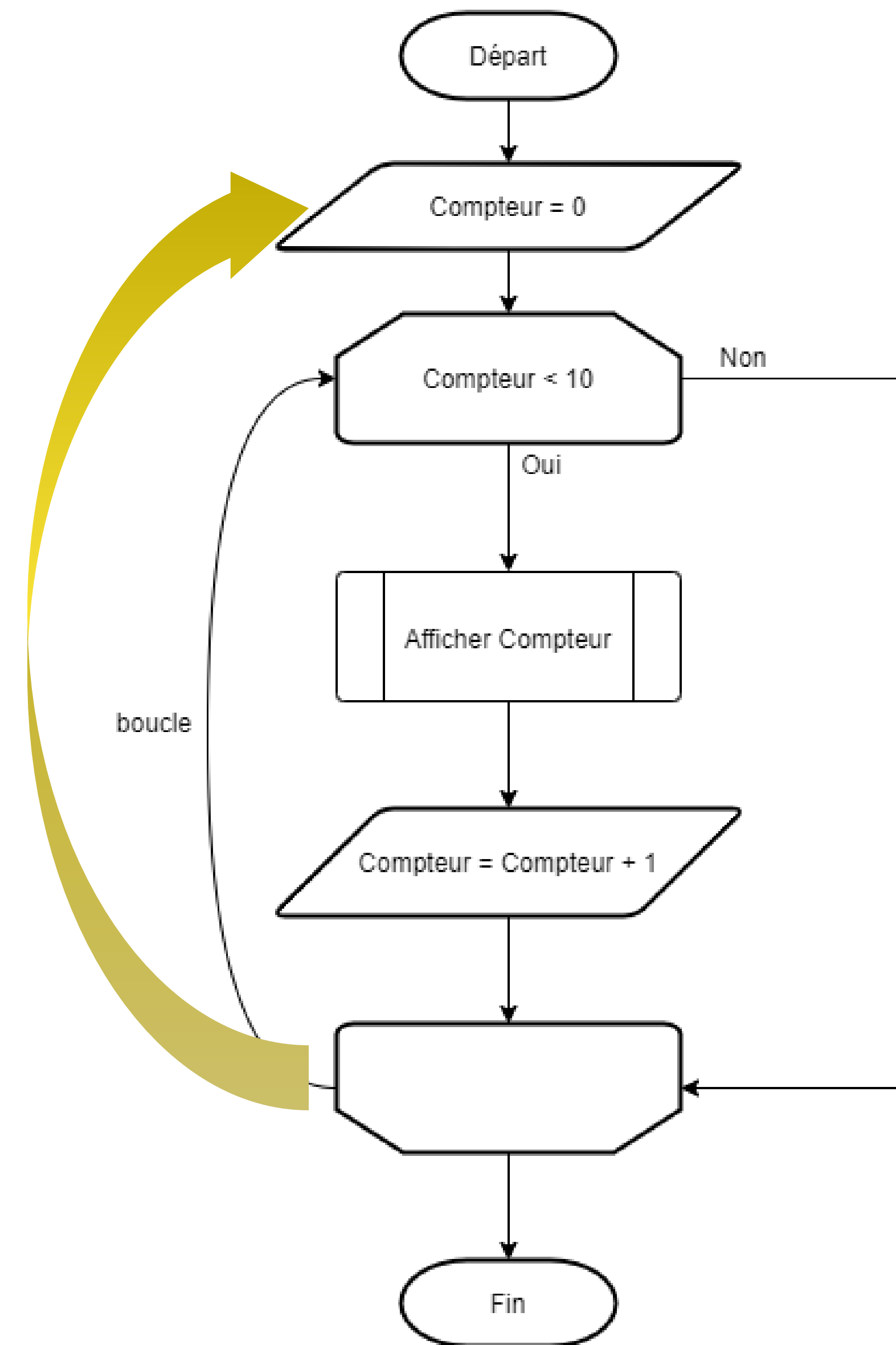
# Les boucles



# Les boucles

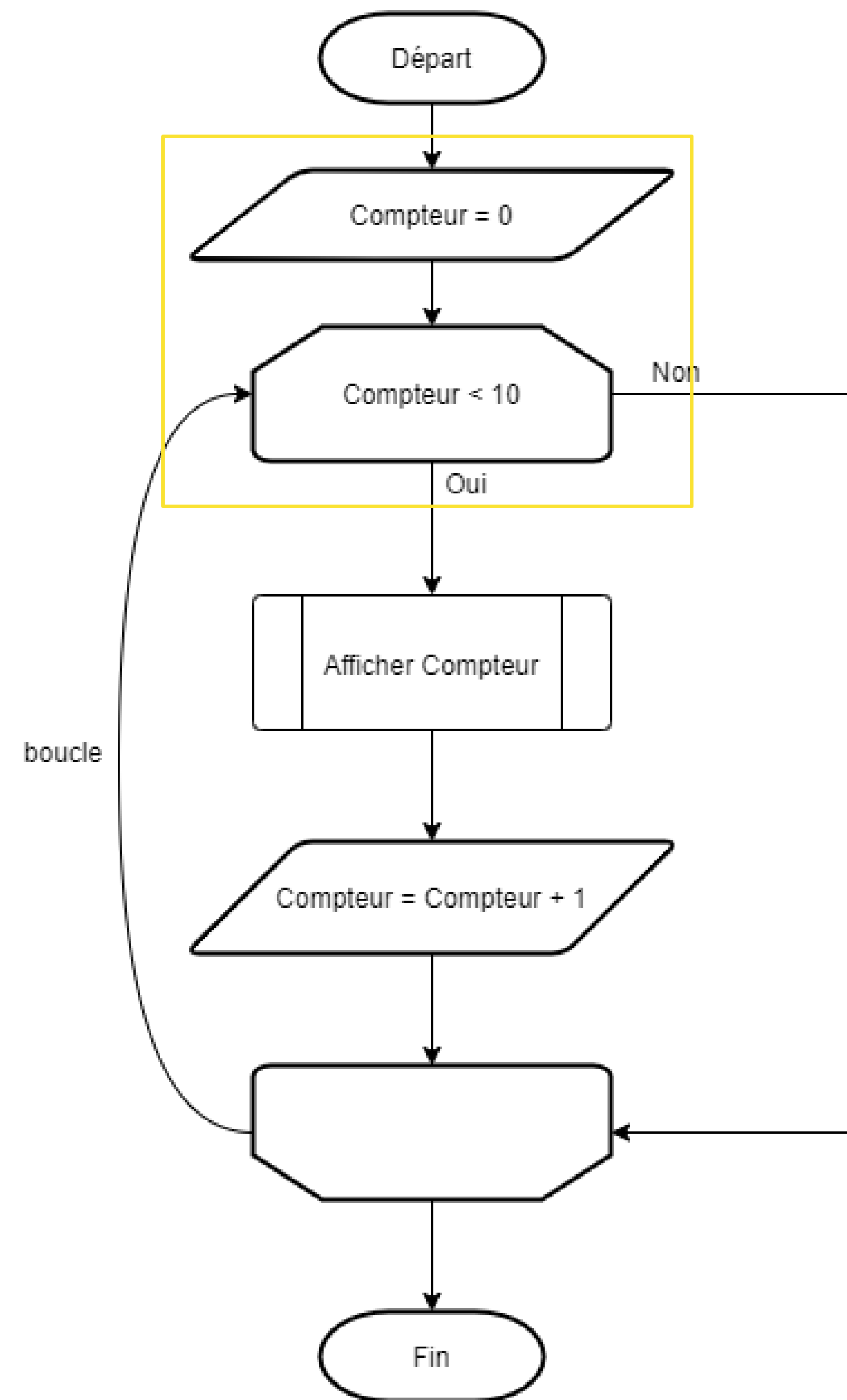


# Les boucles

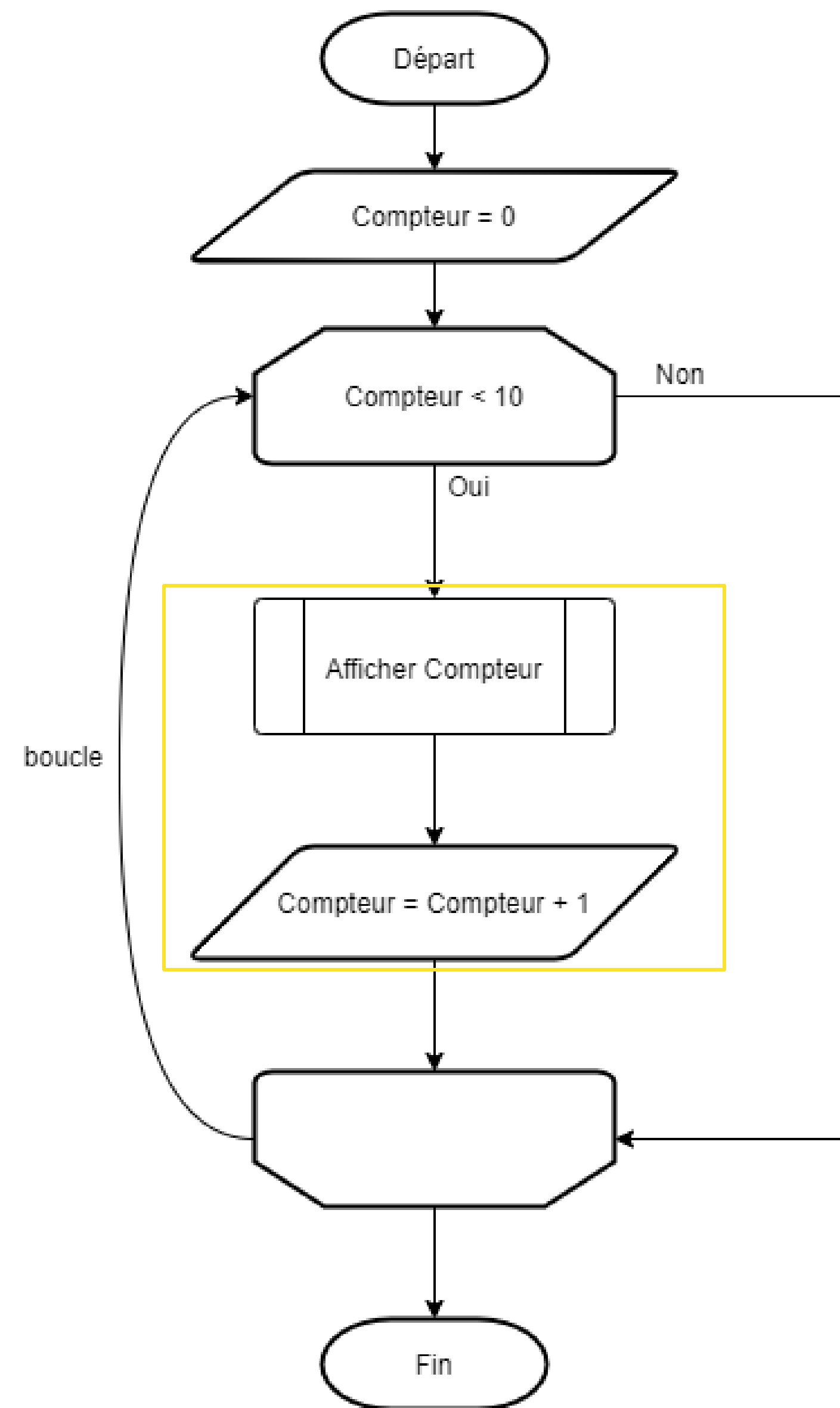
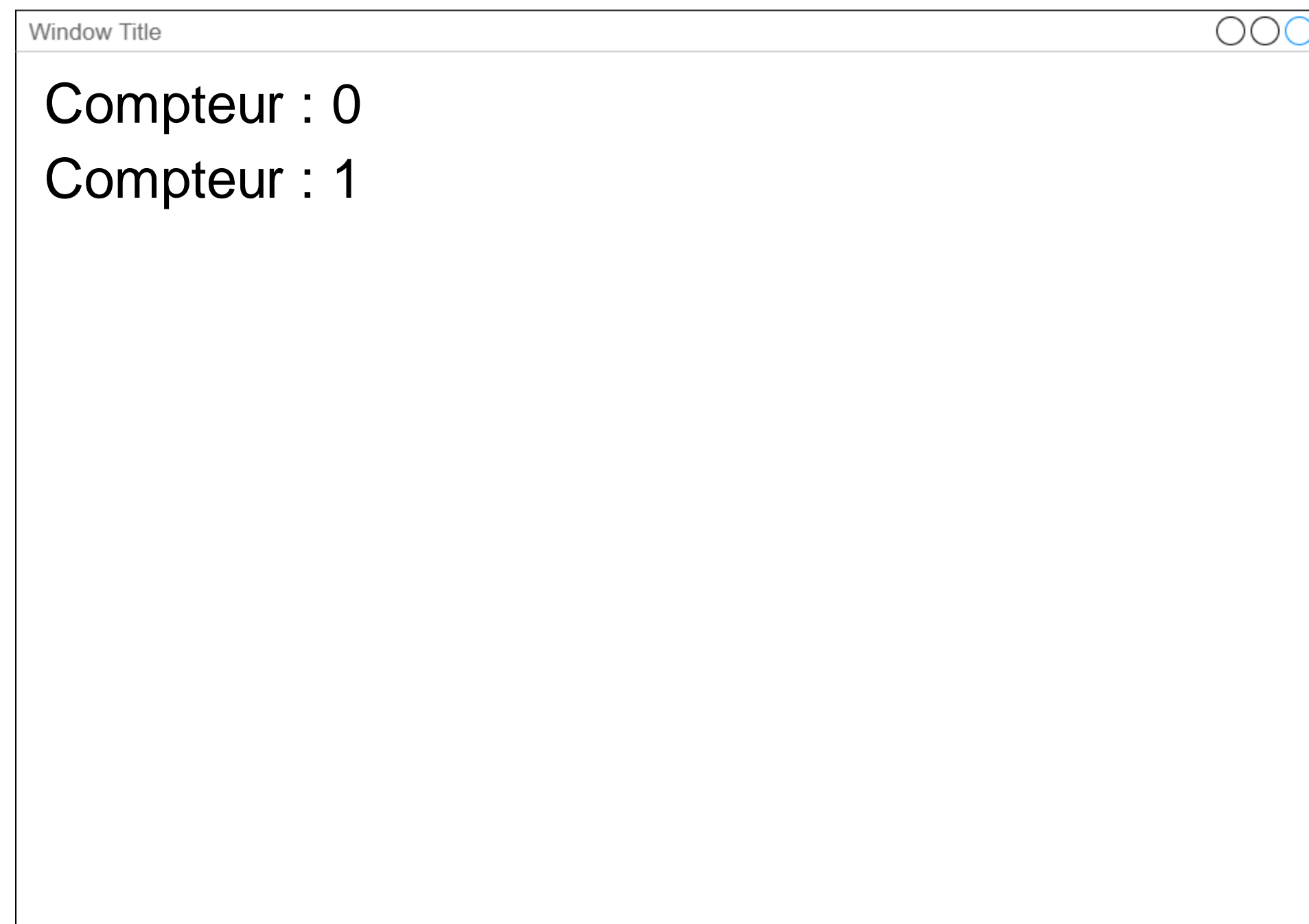




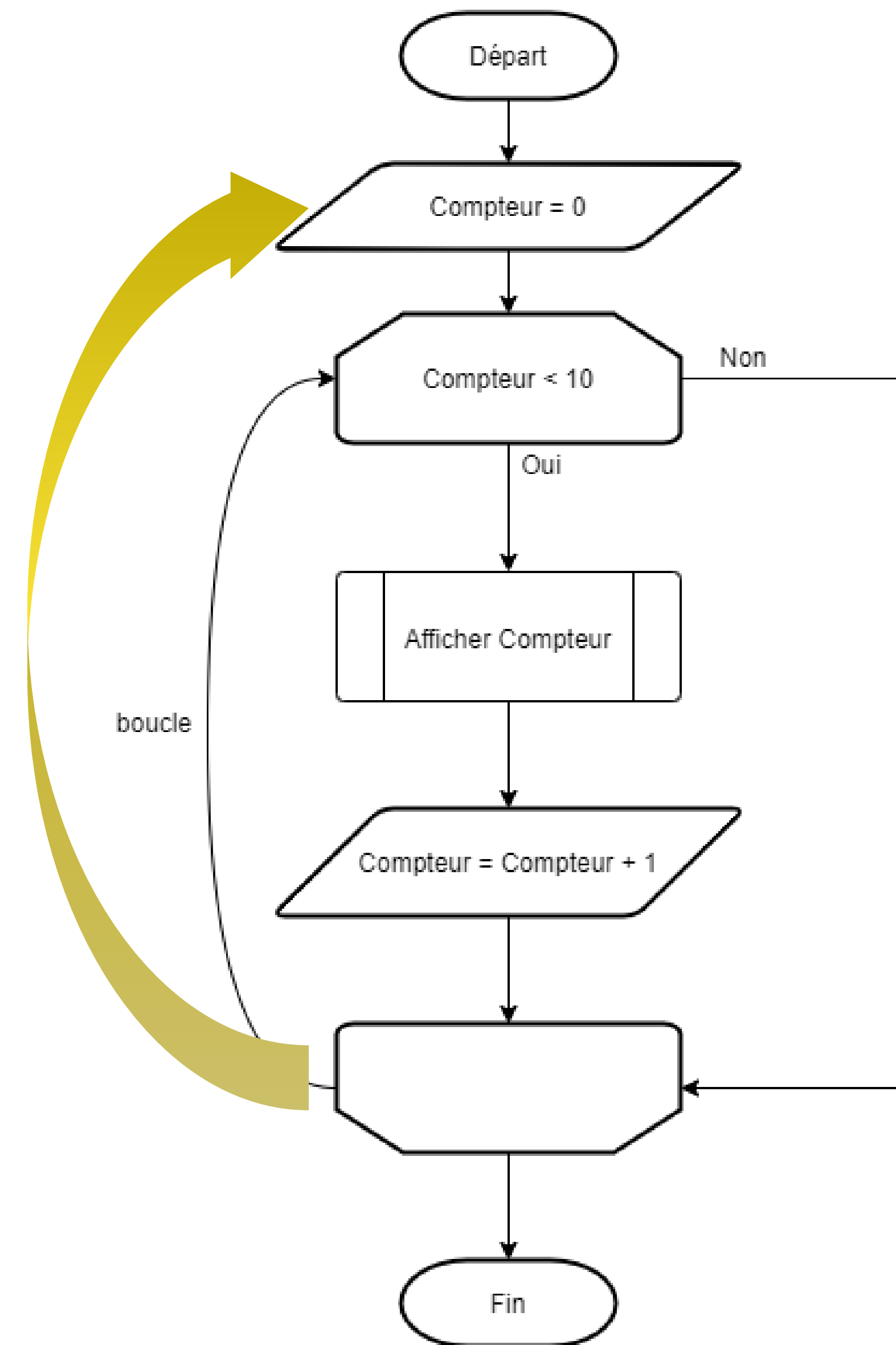
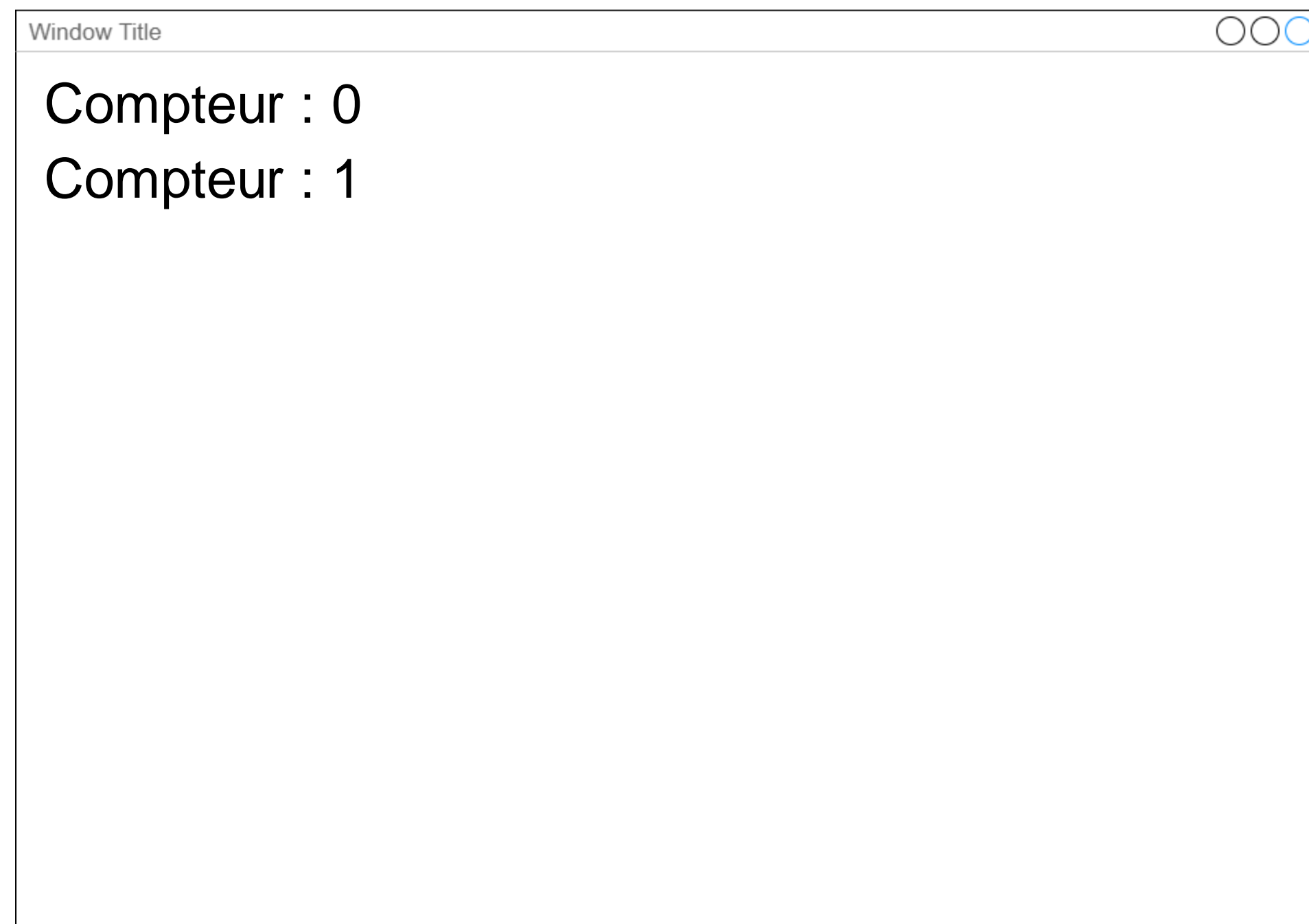
# Les boucles



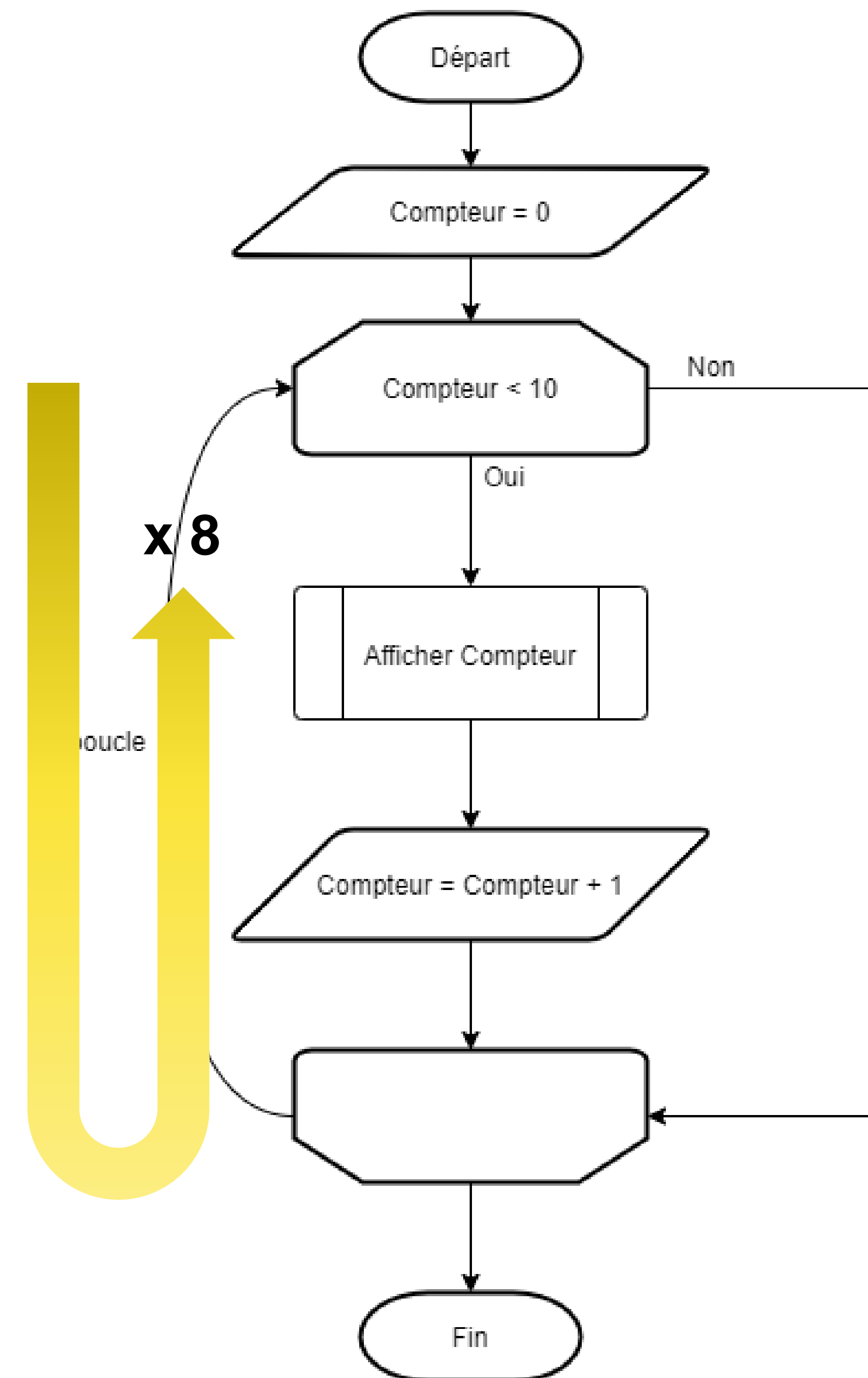
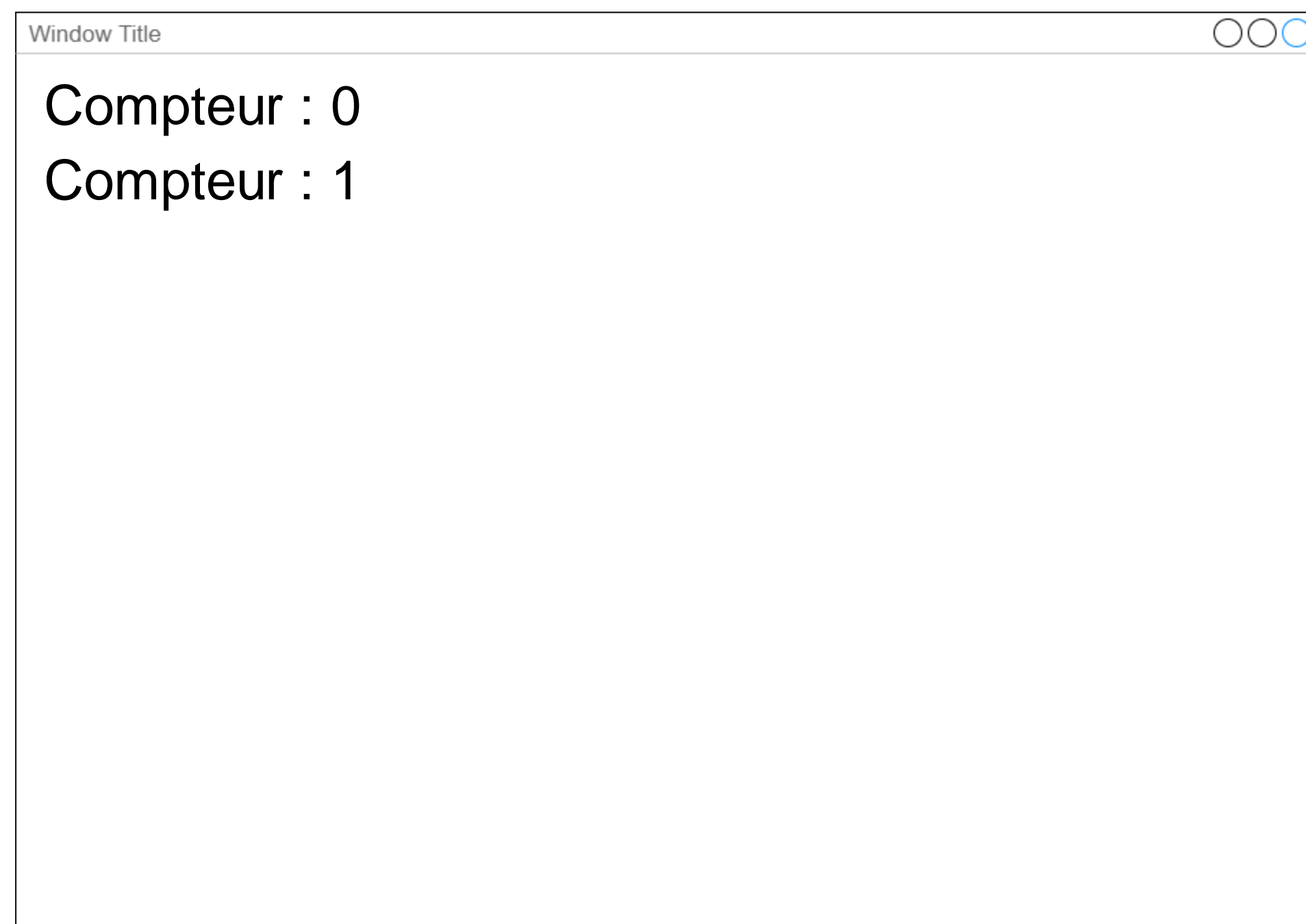
# Les boucles



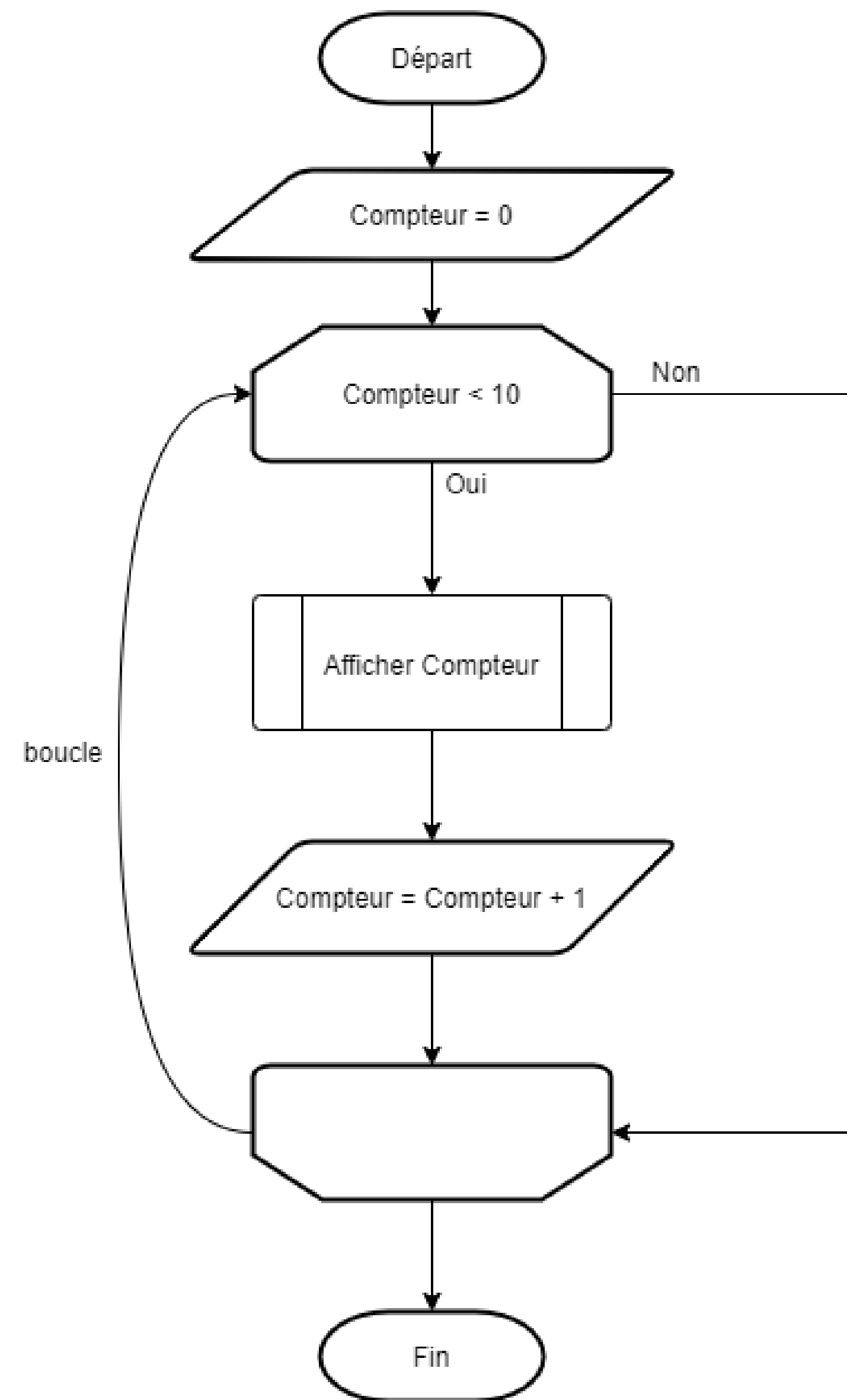
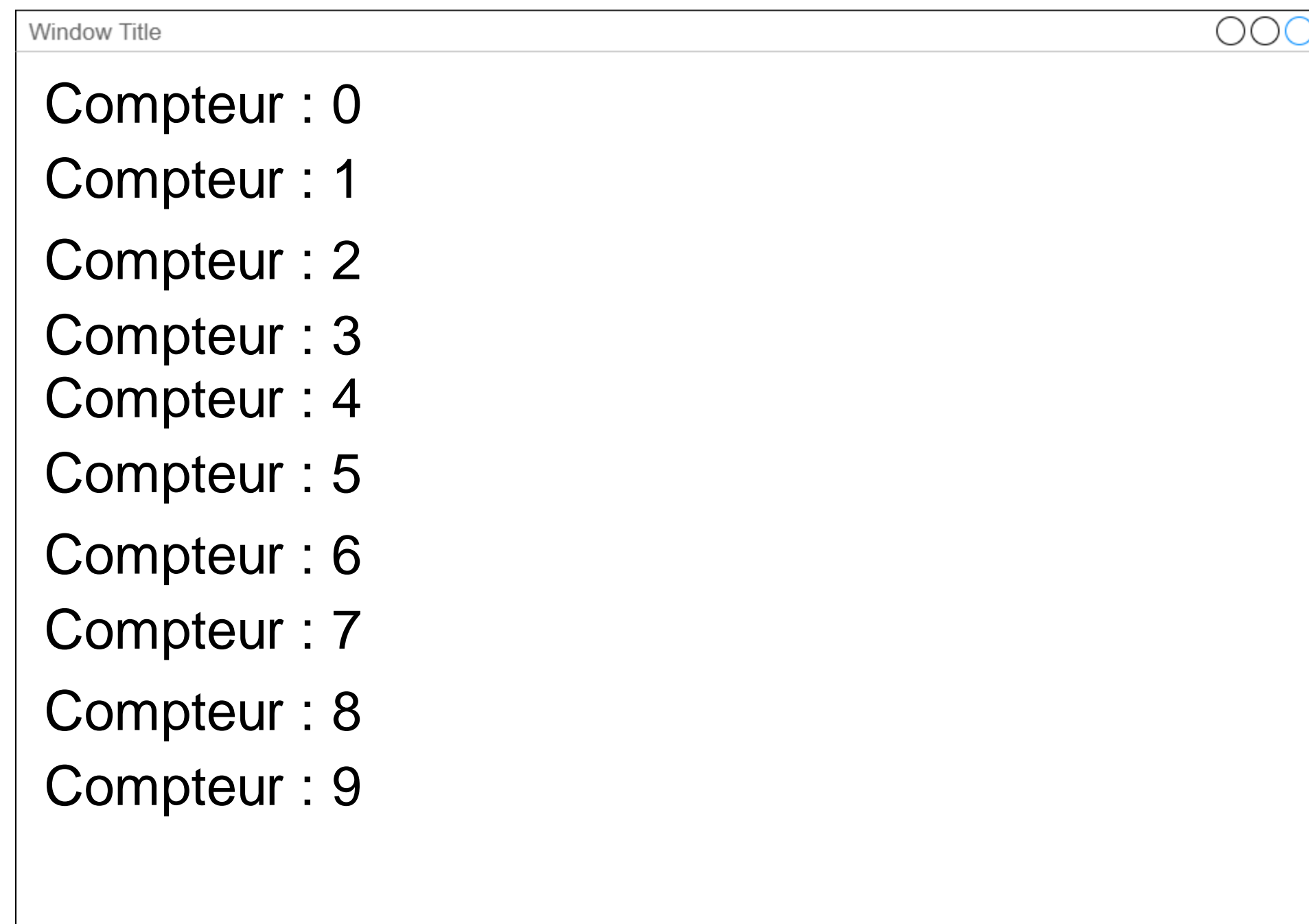
# Les boucles



# Les boucles



# Les boucles



# Boucle 'while'

Window Title

```
$compteur = 0;  
while ($compteur ≤ 10) {  
    . . . $compteur = $compteur + 1;  
}  
echo $compteur;  
  
?>
```

Résultat :  
10

# Boucle 'while'

Initialisation

```
Window Title○ ○ ○  
$compteur = 0;  
while ($compteur ≤ 10) {  
    . . . $compteur = $compteur + 1;  
}  
echo $compteur;  
  
?>
```

Résultat :  
10

# Boucle 'while'

Initialisation

Condition

```
Window Title
$compteur = 0;
while ($compteur <= 10) {
    . . . $compteur = $compteur + 1;
}
echo $compteur;

?>
```

Résultat :  
10



# Boucle 'while'

Initialisation  
Condition  
Incrémentation

```
Window Title
$compteur = 0;
while ($compteur < 10) {
    $compteur = $compteur + 1;
}
echo $compteur;

?>
```

Résultat :  
10

# Boucle 'for'

Window Title



```
for ($compteur = 0; $compteur ≤ 10; $compteur = $compteur + 1) {  
    ... echo "+ 1 tour";  
}  
echo $compteur;
```

### Résultat :

+ 1 tour

+ 1 tour

+ 1 tour

+ 1 tour

+ 1 tour

+ 1 tour

+ 1 tour

+ 1 tour

+ 1 tour

+ 1 tour

10

# Boucle 'for'

# Initialisation

Window Title



```
for ($compteur = 0; $compteur ≤ 10; $compteur = $compteur + 1) {  
    ... echo "+ 1 tour";  
}  
echo $compteur;
```

### Résultat :

[illegible]

# Boucle 'for'

# Initialisation

# Condition

[illegible]

# Boucle 'for'

# Initialisation

# Condition

# Incrémentation

```
Window Title
```

```
for ($compteur = 0; $compteur ≤ 10; $compteur = $compteur + 1) {  
    ... echo "+ 1 tour";  
}  
echo $compteur;
```

### Résultat :

+ 1 tour

+ 1 tour

+ 1 tour

+ 1 tour

+ 1 tour

+ 1 tour

+ 1 tour

+ 1 tour

+ 1 tour

+ 1 tour

10

# Boucle 'for'

# Initialisation

# Condition

# Incrémentation

Window Title



```
for ( $compteur = 0; $compteur ≤ 10; $compteur = $compteur + 1 ) {  
    ... echo "+ 1 tour";  
}  
echo $compteur;
```

### Résultat :

+ 1 tour

+ 1 tour

+ 1 tour

+ 1 tour

+ 1 tour

+ 1 tour

+ 1 tour

+ 1 tour

+ 1 tour

+ 1 tour

10

« Un programme sans boucle (...) ne vaut pas la peine d'être écrit. »

- Epigrams on Programming, Alan Jay Perlis