Leçon 18
Les boucles while

Objectif

Les boucles sont comme des outils pour répéter des actions. Elles permettent de réaliser une tâche plusieurs fois sans avoir à écrire le même code encore et encore. Nous allons découvrir deux types de boucles en Kotlin : while et do while. Ces boucles nous aident à répéter des blocs de code jusqu'à ce qu'une condition spécifique soit remplie.

La Boucle while

La boucle while est utilisée lorsque vous voulez répéter une action tant qu'une condition est vraie. La condition est vérifiée avant chaque exécution du bloc de code. Si la condition est fausse dès le début, le bloc de code ne sera jamais exécuté.

Syntaxe :

```
while (condition) {
    // Bloc de code à répéter
}
```

Exemple :
Supposons que nous voulons afficher les nombres de 1 à 5 :

```
var i = 1
var resultats = ""
while (i <= 5) {
    resultats += "$i\n"  // Ajouter le nombre actuel à la chaîne de résultats
    i++  // Incrémenter le compteur
}
setContent {
    Text(resultats)
}
```

Dans cet exemple, nous commençons avec i égal à 1. Tant que i est inférieur ou égal à 5, nous ajoutons i à la chaîne de résultats et l'incrémenteons de 1 à chaque itération. La boucle continue jusqu'à ce que i soit supérieur à 5.

La Boucle do while

La boucle do while est utilisée pour exécuter un bloc de code au moins une fois avant de vérifier la condition. Cela signifie que le bloc de code sera toujours exécuté au moins une fois, même si la condition est fausse dès le départ. Cependant, ce type de boucle est rarement utilisé.

Syntaxe :

```
do {
    // Bloc de code à répéter
} while (condition)
```

Exemple :
Affichons les nombres de 1 à 5 en utilisant une boucle do while :

```
var i = 1
var resultats = ""
do {
    resultats += "$i\n"  // Ajouter le nombre actuel à la chaîne de résultats
    i++  // Incrémenter le compteur
} while (i <= 5)

setContent {
    Text(resultats)
}
```

Ici, nous affichons i et l'incrémenteons de 1 à chaque itération, puis vérifions si i est toujours inférieur ou égal à 5. La boucle se répète jusqu'à ce que i dépasse 5.

Utilisation de break avec for

Bien que nous ayons vu comment utiliser while et do while, il est aussi utile de savoir comment utiliser break avec une boucle for pour obtenir un comportement similaire à une boucle while.

Exemple :
Utilisons une boucle for avec break pour afficher les nombres de 1 à 5 :

```
var resultats = ""
for (i in 1 ≤ .. ≤ 10) {
    if (i > 5) break  // Sortir de la boucle si i est supérieur à 5
    resultats += "$i\n"  // Ajouter le nombre actuel à la chaîne de résultats
}
setContent {
    Text(resultats)
}
```

Dans cet exemple, nous utilisons une boucle for pour parcourir les nombres de 1 à 10. Cependant, nous utilisons break pour sortir de la boucle lorsque i devient supérieur à 5. Ainsi, seuls les nombres de 1 à 5 sont ajoutés à la chaîne de résultats.

Quand utiliser for vs while

- while : Utilisez while lorsque vous ne savez pas combien de fois vous devrez répéter l'action ou lorsque la condition de continuation est complexe. Par exemple, lorsque vous parcourez une liste de date jusqu'à la date du jour ou lorsque vous regardez chaque ligne d'un fichier.

- for : Utilisez for lorsque vous devez parcourir une séquence d'éléments, comme les éléments d'une liste ou les indices d'un tableau. Le for est plus lisible et plus simple pour ces cas.

Exercices

Exercice 1

1. Créez une variable resultats de type String.
2. Initialisez une variable i à 10.
3. Utilisez une boucle while pour décrémenter i de 1 jusqu'à ce qu'il atteigne 1.
4. Ajoutez chaque valeur de i à la variable resultats, avec un retour à la ligne après chaque nombre.
5. Affichez le contenu de resultats avec setContent { Text(resultats) }.

Exercice 2

1. Créez une variable resultats de type String.
2. Initialisez une variable i à 3.
3. Utilisez une boucle do while pour ajouter chaque multiple de 3 à resultats tant que i est inférieur ou égal à 30.
4. Incrémentez i de 3 à chaque itération.
5. Affichez le contenu de resultats avec setContent { Text(resultats) }.

Exercice 3

1. Créez une liste mutable d'éléments : val elements = mutableListOf("Élément A", "Élément B", "Élément C", "Élément D").
2. Créez une variable resultats de type String.
3. Initialisez une variable index à 0.
4. Utilisez une boucle while pour ajouter chaque élément de la liste à resultats, et arrêtez lorsque index est égal à 2.
5. Affichez le contenu de resultats avec setContent { Text(resultats) }.

Exercice 4

1. Créez une liste mutable d'éléments : val fruits = mutableListOf("Pomme", "Banane", "Cerise", "Datte", "Figue").
2. Créez une variable resultats de type String.
3. Utilisez une boucle for pour parcourir la liste et ajouter chaque élément à resultats.
4. Utilisez break pour arrêter la boucle après le deuxième élément.
5. Affichez le contenu de resultats avec setContent { Text(resultats) }.

Exercice 5

1. Créez une liste mutable de nombres : val nombres = mutableListOf(4, 7, 12, 15, 22).
2. Créez une variable resultats de type String.
3. Utilisez une boucle for pour parcourir la liste et rechercher le nombre 15.
4. Utilisez break pour sortir de la boucle lorsque le nombre 15 est trouvé.
5. Ajoutez un message indiquant que le nombre 15 a été trouvé à resultats.
6. Affichez le contenu de resultats avec setContent { Text(resultats) }.

Exercice 6

Contexte :
Vous gérez une application qui reçoit des commandes de produits. Chaque commande est stockée dans une liste et doit être traitée une par une. Vous devez créer un programme qui traite les commandes jusqu'à ce qu'il n'y en ait plus dans la liste.

Instructions :

1. Initialisation des Données :
   - Créez une liste mutable de commandes. Chaque commande est une chaîne de caractères représentant le produit commandé.
   - Par exemple : val commandes = mutableListOf("Commande 1: Livre", "Commande 2: Tasse", "Commande 3: Clavier").
2. Création d'une Variable pour le Résultat :
   - Créez une variable resultats de type String pour stocker les résultats du traitement des commandes.
3. Traitement des Commandes avec while :
   - Créez une boucle while qui continue tant que la liste de commandes n'est pas vide.
   - Dans chaque itération, récupérez et retirez la première commande de la liste.
   - Ajoutez un message à resultats indiquant que la commande a été traitée.
   - Par exemple, si la commande était "Commande 1: Livre", ajoutez "Traitement de Commande 1: Livre terminé.\n" à resultats.
4. Affichage des Résultats :
   - Utilisez setContent { Text(resultats) } pour afficher le contenu de resultats dans l'interface utilisateur de l'application.