Introduction

Ce cours est une première ébauche d'un futur chapitre sur la boucle for. Il n'est pas (encore) accompagné d'exercices d'application mais un certain nombre d'exemples sont fournis.

Son but est de vous présenter le fonctionnement de la "boucle for" qui va désormais remplacer la "boucle de répétition" que vous utilisiez jusqu'à présent.

La "vraie" boucle

Jusqu'à présent, pour répéter certaines instructions un nombre fixe de fois, vous utilisiez une boucle de répétition, par exemple pour afficher les nombres de 0 à 9 :

```
int nombre = 0;
repeat (10)
{
   cout << nombre << " ";
   nombre = nombre + 1;
}
```

Comme nous vous l'avions dit, la construction `repeat()` n'existe en réalité pas en C++, elle vous avait été enseignée afin de faciliter votre découverte de la programmation. Il était possible de l'utiliser grâce à la ligne suivante, que nous vous demandions de mettre au début de vos programmes :

```
#define repeat(nb) for (int _loop = 1, _max = (nb) ; _loop <= _max ; _loop++)
```

Sans expliquer tout le détail de cette instruction, nous allons vous expliquer comme utiliser ce `for` pour faire de "vraies" boucles.

Cas général : suites croissantes

On cherche à faire prendre à une variable `nombre` une suite de valeurs espacées régulièrement, par exemple les nombres de 1 à 100 ou les multiples de 2 entre -20 et 20. Il existe deux manière de parcourir ces valeurs, soit de manière croissante ou de manière décroissante. Nous allons commencer par les suites croissantes. Le fonctionnement de `for` est alors le suivant :

```
for (int nombre = <debut>; nombre <= <fin>; nombre = nombre + <saut>)
{
   ...
}
```

ce qui va faire prendre à la variable `nombre` toutes les valeurs entre `<debut>` (inclus) et `<fin>` (inclus) en faisant des sauts de `<saut>` à chaque fois. Ce code peut donc se lire ainsi :

```
Pour nombre allant de <debut> à <fin> en faisant des sauts de <saut>
   ...
```

Pour bien comprendre, le mieux est de regarder quelques exemples :

Afficher les nombres de 0 à 9

```
for (int nombre = 0; nombre <= 9; nombre = nombre + 1)
{
   cout << nombre << " ";
}
```

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

On va donc de 0 (inclus) à 9 (inclus) en faisant "+1" entre chaque valeur.

Afficher les nombres de 10 à 20

```
for (int nombre = 10; nombre <= 20; nombre = nombre + 1)
{
   cout << nombre << " ";
}
```

```
10 11 12 13 14 15 16 17 18 19 20
```

On va donc de 10 (inclus) à 20 (inclus) en faisant "+1" entre chaque valeur.

Afficher les nombres pairs de 100 à 120

```
for (int nombre = 100; nombre <= 120; nombre = nombre + 2)
{
   cout << nombre << " ";
}
```

```
100 102 104 106 108 110 112 114 116 118 120
```

On va donc de 100 (inclus) à 121 (inclus) en faisant "+2" entre chaque valeur.

Afficher les nombres de 0 à 9 (variante)

A la place du `nombre <= <fin>` il est aussi possible d'utiliser `nombre < <fin>`, par exemple :

```
int valeurMaximum = 10;
for (int nombre = 0; nombre < valeurMaximum; nombre = nombre + 1)
{
   cout << nombre << " ";
}
```

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

On va donc de 0 (inclus) à 10 (non inclus, c'est-à-dire 9 inclus) en faisant "+1" entre chaque valeur.

Interprétation

La variable `nombre` prend la valeur `<debut>`, on regarde alors si la condition `nombre <= <fin>` (ou `nombre < <fin>`) est vraie :

- Si c'est le cas, on rentre dans la boucle. Arrivée à la fin de la boucle, l'instruction `nombre = nombre + <saut>` est executée puis on regarde à nouveau si la condition est vraie.
- Si ce n'est pas le cas, on sort de la boucle.

Cas général : suites décroissantes

Dans le cas d'une suite décroissantes de nombres, le fonctionnement est fortement similaire. Au lieu d'ajouter quelque chose à la valeur pour passer à la prochaine valeur, on doit simplement soustraire quelque chose :

```
for (int nombre = <debut>; nombre >= <fin>; nombre = nombre - <saut>)
```

```
{
    ...
}
```

Le saut est donc négatif et la condition pour continuer à boucler est qu'on est pas encore arrivé à la valeur `<fin>` donc qu'on est plus grand que elle, étant donné que la suite de nombres est décroissante.

Sur un exemple :

Afficher les multiples de 5 de 20 à -20

On veut donc afficher des nombres de manière décroissante, donc le "saut" est négatif

```
for (int nombre = 20; nombre >= -20; nombre = nombre - 5)
{
    cout << nombre << " ";
}
```

↳

```
20 15 10 5 0 -5 -10 -15 -20
```

On va donc de 20 (inclus) à -20 (inclus) en faisant "-5" entre chaque valeur.

Conclusion

Désormais, n'utilisez plus la boucle de répétition, mais une vraie boucle for !