PHP – COURS 03 – Les structures répétitives : suite

do... while

La seconde structure de boucle est "`do...while`", que l'on pourrait, en français, appeler "faire... tant que".

FAIRE
…
les INSTRUCTIONS
…
TantQue la condition est « VRAIE »

Ici encore, les instructions constituant la boucle sont exécutées tant que la condition de boucle est vraie. Cela dit, contrairement à `while`, avec `do...while`, la condition est évaluée à la fin de la boucle ; cela signifie que les instructions correspondant au corps de la structure de contrôle seront toujours exécutées au moins une fois, même si la condition est toujours fausse !

La syntaxe correspondant à cette structure répétitive est la suivante :

```
$a=0;
do {
   echo("$a \n");
   a++;
} while($a<=2);
```

Le résultat sera exactement le même que celui que nous avions précédemment obtenu, lorsque nous utilisions un `while`.

Par contre, si nous essayons de faire la même pour notre second exemple, en écrivant un code tel celui-ci :

```
$a=0;
do {
  echo("On est dans la boucle \n");
}while($a > 10);
```

... nous pourrons constater que l'on rentre dans la boucle, alors que la condition n'est pas vraie... Ce qui nous montre bien que la condition de boucle est testée en fin de boucle, et que les instructions correspondant à celle-ci sont toujours exécutées, au moins une fois.
Certes, dans un cas tel celui-ci, c'est plus un inconvénient qu'autre chose... Mais il est des cas lorsque l'on programme, où ce comportement correspond à ce que l'on recherche, et, dans ces occasions, il est plus simple d'utiliser un `do...while` qu'un `while` !

Par exemple, lorsque l'on souhaite afficher un menu proposant un choix à l'utilisateur , ou pour exécuter plusieurs fois un même programme.

```
do{
      ….....Programme principal
      echo "voulez vous recommencez (oui/non) \n";
      $reponse = lire();

}while ($reponse == "oui");
```

for...

En théorie, la boucle **while** permet de réaliser toutes les boucles que l'on veut. Toutefois, tout comme le **switch** pour les conditions, il est dans certains cas utile d'avoir un autre système de boucle plus « condensé », plus rapide à écrire.

Les boucles **for** sont juste une autre façon de faire une boucle **while.**

Voici un exemple de boucle **while** pour afficher les nombres de 1 à 100

```
$nombre = 1 ;
while ($nombre <= 100){
   echo ("$nombre \n");
   $nombre++;
}
```

Voici maintenant l'équivalent en boucle **for** :

```
for ($nombre = 1 ; $nombre <= 100 ; $nombre++){
   echo ("$nombre \n");
}
```

Quelles différences ?

- Vous noterez que l'on n'a pas initialisé la variable nombre à 1 dès sa déclaration.
- Il y a beaucoup de choses entre les parenthèses après le for (nous allons détailler ça après).
- Il n'y a plus de compteur++; dans la boucle.

Intéressons-nous à ce qui se trouve entre les parenthèses, car c'est là que réside tout l'intérêt de la boucle for. Il y a trois instructions condensées, chacune séparée par un point-virgule.

- La première est **l'initialisation** : cette première instruction est utilisée pour préparer notre variable compteur. Dans notre cas, on initialise la variable à 1.
- La seconde est **la condition** : comme pour la boucle while, c'est la condition qui dit si la boucle doit être répétée ou non. Tant que la condition est vraie, la boucle for continue.
- Enfin, il y a **l'incrémentation** : cette dernière instruction est exécutée à la fin de chaque tour de boucle pour mettre à jour la variable nombre. La quasi-totalité du temps on fera une incrémentation, mais on peut aussi faire une décrémentation (variable--) ou encore n'importe quelle autre opération (variable += 2; pour avancer de 2 en 2 par exemple).

Bref, comme vous le voyez la boucle for n'est rien d'autre qu'un condensé de la boucle while.

EXERCICES : utiliser la boucle for dans les exemples suivants

Exercice 1 : Compteur

Écrire un programme **compteur.php** qui permet de compter et d'afficher de 1 jusqu'à 20.

Exercice 2 : Compteur

Améliorer le programme pour faire saisir à l'utilisateur jusqu'à quel chiffre il souhaite que le programme compte.

Exercice 3

Écrire un programme qui calcule la somme des entiers de 1 à 100.

Exercice 4

Écrire un programme qui demande un nombre de départ, et qui calcule la somme des entiers jusqu'à ce nombre. Par exemple, si l'on entre 5, le programme doit calculer :
1 + 2 + 3 + 4 + 5 = 15
NB : on souhaite afficher uniquement le résultat, pas la décomposition du calcul.

Exercice 5

Modifier le programme précédent pour qu'il demande un nombre de départ et un nombre de fin, afin de calculer la somme des entiers entre ces deux nombres

Exemple :

Entrez le nombre de départ : 12
Entrez le nombre de fin : 14
Le programme doit calculer :
12+13+14 = 39

Exercice 6

Écrire un programme qui demande un nombre de départ, et qui ensuite écrit la table de multiplication de ce nombre, présentée comme suit (cas où l'utilisateur entre le nombre 7) :
Table de 7 :
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
…
7 x 10 = 70

Exercice 7

Écrire un programme qui demande successivement 10 nombres à l'utilisateur, et qui lui dise ensuite quel était le plus grand parmi ces 10 nombres :

Entrez le nombre numéro 1 : 12
Entrez le nombre numéro 2 : 14
etc.
Entrez le nombre numéro 10 : 6
Le plus grand de ces nombres est : 14

Modifiez ensuite le programme pour qu'il affiche de surcroît en quelle position avait été saisie ce nombre :

C'était le nombre numéro 2

Exercice 8

Écrire un programme qui affiche les nombres de 20 à 1 de trois en trois. Exemple :

20 19 18
17 16 15
14 13 12
...

(faire deux fois l'exercice : avec "for" et avec "while").