§ 2 Tableaux et structures de contrôle répétitives

2-1 Structure de contrôle de boucle for ($i=0 ; $i<20 ; $i++) {action ;}

Le structure « for » convient lorsque l'on sait à l'avance combien de fois un bloc d'instructions doit être répété.

Dans **l'itérateur ($i=0 ; $i<20 ; $i++)**,

- $i désigne la variable de contrôle ;
- $i<20 est la condition à remplir pour que l'action soit exécutée ;
- $i++ signifie $i=$i+1 et représente l'incrémentation de la variable de contrôle qui est effectuée après l'exécution de l'action.

Exemple 2-1-1 : Somme de n termes (valeur numérique du nombre e)

$$e = 1 + \frac{1}{1} + \frac{1}{1*2} + \frac{1}{1*2*3} + \frac{1}{1*2*3*4} + \frac{1}{1*2*3*4*5} + \frac{1}{1*2*3*4*5*6} + \ldots$$

Exemple 2-1-2 : Table d'intérêts composés

Explications de l'exemple 2-1-2 « HTML <table> » : Pour mettre en page des résultats sous la forme d'un tableau HTML, on fait appel aux balises suivantes :

- <table><tbody> .... </tbody></table> pour définir un tableau ;
- <tr> ... </tr> pour créer une nouvelle ligne du tableau ;
- <td style="text-align:right"> ... </td> pour créer une première cellule (qui correspond à la première colonne du tableau) dont le contenu sera aligné à droite ;
- <td style="text-align:right ; width:180px"> ... </td> pour créer une deuxième cellule (qui correspond à la deuxième colonne du tableau) dont le contenu sera aligné à droite et dont la largeur sera d'au moins 180 pixels, ce qui aura pour effet d'espacer les deux colonnes de nombres.

Attention : ces tableaux qu'on peut appeler « HTML <table> » n'ont rien à voir avec les tableaux que nous introduirons au § 2 et que nous pourrions plus précisément nommer « PHP array() ; ».

L'instruction PHP **number_format($valeur, 2, '.', ' ')** signifie que le nombre $valeur doit être affiché avec 2 chiffres décimaux, que le point décimal doit être représenté par le symbole '.' et que le séparateur de milliers est un espace symbolisé par ' '.

Exemple 2-1-3 « for » imbriqués : Déterminer tous les nombres de trois chiffres dont la somme des cubes des chiffres est égal au nombre lui-même. Equation $c^3 + d^3 + u^3 = c*100 + d*10 + u$

$$c \in \{1,2,3,4,5,6,7,8,9\} \wedge d \in \{0,1,2,3,4,5,6,7,8,9\} \wedge u \in \{0,1,2,3,4,5,6,7,8,9\}$$

Exemple 2-1-4 : Nombres parfaits

2-2 Tableaux ou variables indicées « PHP array() ; »

Considérons le cas courant où une liste de *n* valeurs doit être conservée en mémoire de travail sous la forme d'une variable indicée x_0, x_1, x_2, x_3, x_4, x_5, … , x_{n-1}.

Pour déclarer que la variable x est un tableau, on utilise l'instruction **$x = array() ;**
après quoi, on peut utiliser la variable indicée **$x[$i]** comme une variable usuelle.

Exemple 2-2-1 : Tableau : affichage, moyenne, maximum.

Tableaux bidimensionnels : pour déclarer que la variable t est un tableau à deux indices, on utilise

l'instruction **$t = array() ;** après quoi, on peut utiliser la variable indicée **$t[$i][$j]** comme une variable usuelle.

Exemple 2-2-2 : Table de multiplication modulo m

2-3 Structure de contrôle « tant que » while (condition) {action ;}

Il faut remarquer que, puisque la « condition » est testée au début, il est possible que le bloc d'instructions dénommé « action » ne soit pas du tout exécuté.

Exemple 2-3-1 : Somme de termes : valeur numérique approchée de sin(x)

$$\sin(x) = x - \frac{x^3}{1*2*3} + \frac{x^5}{1*2*3*4*5} - \frac{x^7}{1*2*3*4*5*6*7} + \frac{x^9}{1*2*3*4*5*6*7*8*9} - \ldots$$

Erreurs d'arrondi

Les algorithmes que l'on utilise doivent tenir compte des erreurs d'arrondi. Par exemple, le nombre 0.2 comporte, lorsqu'il est converti en base 2, un nombre infini de bits ce qui nécessite qu'il soit tronqué. C'est pourquoi, en ce qui concerne les variables de contrôle, il faut donner la préférence aux variables à valeurs entières. Si l'on doit utiliser une variable de contrôle de type réel (à valeurs non entières), il ne faut jamais la tester avec une égalité.

Exemple 2-3-2a : Erreur d'arrondi (exemple à ne pas suivre)

Exemple 2-3-2b : Erreur d'arrondi (exemple corrigé)

Exemple 2-3-3 : Décomposition d'un entier naturel en facteurs premiers

Commentaires de l'exemple 2-3-3

- Le nombre n donné est divisé par p=2 autant de fois que cela est possible ; durant cette activité, on compte le nombre e de fois que le facteur p est contenu dans n ; on affiche 2^e ;
- Le nombre n donné est divisé par p=3 autant de fois que cela est possible ; durant cette activité, on compte le nombre e de fois que le facteur p est contenu dans n ; on affiche 3^e ;
- Puisque maintenant n n'est divisible ni par 2, ni par 3, le reste de la division de n par 4 est nul ; ainsi seuls des diviseur p premiers peuvent apparaître ;
- Poursuivre avec p=5, etc, tant que n>1. Le processus s'arrête nécessairement car la plus grande valeur possible de p est n.
- Si e==0, le facteur p^0 n'est pas affiché ;
  si e == 1, affiche p au lieu de p^1 ;
  si e >= 2, affiche p^e.
- Au premier facteur p^e affiché, la variable « suivant » a la valeur booléenne « faux » ; l'expression p^e est affichée sans être précédée d'une étoile.
- Pour le deuxième facteur p^e et les suivants, la variable « suivant » a la valeur logique « vrai » ; alors une étoile est écrite avant d'afficher p^e.
- Il est préférable d'écrire « if ($suivant) » plutôt que « if ($suivant == true) ».

2-4 Structure de contrôle « répète ... jusqu'à ce que » do {action ;} while (condition) ;

Il faut remarquer que, puisque la « condition » est testée à la fin, le bloc d'instructions dénommé « action » est toujours exécuté au moins une fois.

Exemple 2-4-1 : Développement décimal du quotient de deux entiers.