§ 3 Structuration des données et des programmes

3-0 L'analyse descendante, ou fragmentation descendante

Descartes affirmait que la logique est composée de quatre préceptes. Nous en extrayons deux pour introduire notre propos :

> *[...] de faire partout des dénombrements si entiers et des revues si générales que je fusse assuré de ne rien omettre.*
>
> *[...] de diviser chacune des difficultés que j'examinerais en autant de parcelles qu'il se pourrait et qu'il serait requis pour les mieux résoudre.*
>
> *Discours sur la Méthode (1637)*

Un des problèmes majeurs dans le développement de programmes réside dans la quantité de détails à considérer. Pour surmonter cette difficulté, on décompose la tâche à réaliser en sous-tâches, en précisant l'ordre de leur enchaînement et les conditions de leur activation. Les étapes suivantes consistent à décomposer ces sous-tâches en d'autres sous-tâches jusqu'à atteindre le niveau de détails exigé. On part donc d'un point de vue général et global et on n'introduit les détails que progressivement. Cette façon d'aborder la résolution de problèmes est appelée *analyse descendante* ou *fragmentation descendante*.

La programmation consiste essentiellement à maîtriser la complexité des problèmes. Seule une approche méthodique permet d'y parvenir. Ne vous lancez jamais dans l'écriture d'un programme sans avoir préalablement dégagé la structure des données et la structure de l'algorithme, au moins dans leurs grandes lignes.

3-1 Sous-programmes

Chaque sous-tâche correspond habituellement à un sous-programme. En PHP, chaque sous-programme est une « function ».

Sous-programme avec un ou plusieurs arguments par valeurs :

```
function sp($x, $y){
        $z= ...;
        return $z ;
} ;
```

$x et $y sont des ***arguments par valeurs*** ; à l'appel de la fonction sp, chaque argument est remplacé par la valeur d'une expression, par exemple sp(4*$t, $b-$a) et le sous-programme sp sera exécuté avec les valeurs d'arguments $x = 4*$t ; $y = $b-$a.

$z est une ***variable locale*** : elle n'a pas d'existence en dehors du sous-programme sp. Les arguments $x et $y sont aussi des variables locales.

Exemple 3-1-1 : Premier exemple de sous-programmes

Sous-programme avec une ou plusieurs variables globales :

```
function sp(){
        global $a, $b ;
        $z= ...;
        return $z ;
} ;
```

Les variables $a et $b sont ***globales*** ; ce qui signifie que $a représente la même variable

dans sp et hors de sp ; de même pour $b.
$z est une variable locale : elle n'a pas d'existence en dehors du sous-programme sp.
Les sous-programmes sans argument sont appelés « procédures ».

Exemple 3-1-2 : Variables locales, variables globales.

Sous-programme avec un ou plusieurs arguments par référence :

```
function sp(&$r, &$s){
        = ...;
        return $s ;
} ;
```

Les arguments $r et $s sont des ***arguments par référence*** ; à l'appel de la fonction sp, chaque argument se réfère au nom d'une variable, par exemple sp($p, $q) et le sous-programme sera exécuté avec $r qui est une référence à la variable $p, et $s qui est une référence à la variable $q.

Un argument par référence peut être modifié par l'exécution du programme.

Exemple 3-1-3 : Argument par référence.

Un sous-programme peut prendre la forme plus générale :

```
function sp($x, $y, &$r, &$s){
        global $a, $b ;
        $z= ...;
        return $z ;
} ;
```

Les arguments $x, $y sont passés par valeurs ;
les arguments $r, $s sont passés par référence ;
les variables $a, $b sont globales ;
les variables $x, $y, $z sont locales.

Exemple 3-1-4 : Table de valeurs numériques d'une fonction

Tabulation de la fonction $f(x)=\frac{4*x^4}{1+x^6}$ sur l'intervalle [0, 2].

Exemple 3-1-5 : Code secret de César - Exemple de fonctions imbriquées

Vers 50 avant J. C., César codait les messages qu'il envoyait à Cicéron de la manière suivante : les espaces entre les mots et les signes de ponctuation étaient supprimés ; chaque lettre du texte était remplacée par un autre au moyen de la table suivante (rotation de 3 à droite opérant sur les 26 lettres de l'alphabet) :

ABCDEFGHIJKLMNOPQRSTUVWXYZ
DEFGHIJKLMNOPQRSTUVWXYZABC

c'est-à-dire la lettre A est remplacée par la lettre D, la lettre B par E, ..., la lettre Z par C.
Remarquer que les fonctions sont imbriquées : *codecar* est un sous-programme local de *codechaine*, et *decodecar* est un sous-programme local de *decodechaine*.

Exemple 3-1-6 : Fonctions pour arrondir à 5 ct

Exemple 3-1-7 :
Fonction à output multiple: ensemble des solutions réelles d'une équation de degré <= 2

Si l'output d'un sous-programme est formé de plusieurs valeurs, il faut les organiser en un tableau.

3-2 Structures de données « PHP array() ; »

En PHP, les tableaux permettent de représenter des structures de données complexes. Les tableaux peuvent être imbriqués : on peut bâtir des tableaux de tableaux, des tableaux de tableaux de tableaux, ..., qui peuvent ainsi constituer une arborescence multiforme.

Exemple 3-2-1 : Tableaux PHP comme structures de données