1 Les structures de données en PHP (suite)

1.1 Les tableaux (dictionnaires)

- Le nom du type est `array`.
- Les tableaux sont les équivalents des dictionnaires en Python mais avec une limitation sur le type des éléments (pas de type `double`). Ils ne peuvent être indicés que par des entiers positifs ou par des chaînes de caractères. Les valeurs associées aux éléments sont de n'importe quel type en revanche, comme en Python.
- On crée ou écrase un élément avec `$tab[elem]=...` et on accède à la valeur d'un élément avec `$tab[elem]`. Le nombre d'éléments dans un tableau est donné par la fonction `count()`. On peut enlever un élément d'un tableau avec `unset($tab[elem])`. On peut trier un tableau dans l'ordre de ses éléments avec `sort($tab)` (`rsort()` pour un ordre décroissant).
  Il y a également des raccourcis bien pratiques :
  - « `array(valeur-0,valeur-1,...,valeur-n)` » est un tableau dont les éléments sont les entiers de `0` à `n` et les valeurs correspondantes sont évidemment les `valeur-`i successifs.
    On peut donc écrire `$tab = array(...);`.
  - « `array("chaine-0"=>valeur-0 , "chaine-1"=>valeur-1 , "chaine-n"=>valeur-n)` » est de même un tableau dont les éléments sont les chaînes `"chaine-`i`"` et les valeurs correspondantes les `valeur-`i.

  On peut aussi découper une chaîne de caractères en morceaux dans un tableau d'entiers : `explode(marque,chaine)` repère tous les endroits où la marque est contenue dans la chaine et met dans le tableau les parties situées entre les marques. Par exemple `$t=explode(' ',"salut bonjour hello")` est un tableau à 3 éléments de `0` à `2` et dont les valeurs correspondantes sont dans l'ordre `"salut"`, `"bonjour"` et `"hello"`.
  Il y a de nombreuses autres opérations sur les tableaux, voir le manuel PHP si besoin.

Dans les doubles quotes, faire évaluer des variables est source d'erreurs et tout particulièrement s'il s'agit d'une variable de type tableau. Par exemple

```
$prenom = "Pierre";
print("Son prénom est $prenom.");
```

écrira bien « `Son prénom est Pierre.` » parce que le point n'est pas un caractère valide dans un nom de variable mais bien sûr

```
print("Son prénom est $prenomtte.");
```

dira que la variable `$prenomtte` n'est pas définie. Il faut écrire

```
print("Son prénom est {$prenom}tte.");
```

pour obtenir « `Son prénom est Pierrette.` ».
Dès lors, puisqu'un élément d'un tableau peut être lui-même une chaîne de caractères, la manière d'écrire l'appel au tableau avec les accolades est lourdement source d'erreurs. Mieux vaut donc utiliser à la place la concaténation :

```
$prenoms = array("Pierre","Anne");
print("Bonjour ".$prenoms[0]." et ".$prenoms[1]." !");
```

pour obtenir « `Bonjour Pierre et Anne !` ».

Comme en Python, les tableaux ne sont pas multidimensionnels mais on peut faire des tableaux de tableaux.

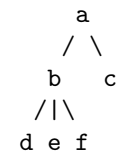
Enfin, il existe un itérateur sur les tableau :

```
foreach($tab as $elem=>$valeur)
  {
  ..ici le corps de la boucle..
  }
```

Pour finir cet inventaire des structures de données, signalons que la fonction `gettype()` renvoit le type de son argument (le résultat est une chaîne de caractères). *Les casts* : il est également possible de modifier ou d'imposer le type d'une variable en mettant simplement le type voulu entre parenthèse avant d'utiliser la variable. Par exemple « `$n="100"` » fait une variable `$n` de type `string` mais après « `$m = (integer)$n + 1;`, la variable `$m` est de type integer et vaut 101.

1.2 Exercice

Représentez l'arbre suivant sous forme d'un tableau :



Écrivez une fonction qui calcule le nombre de noeud d'un tel arbre : dans le style récursif puis dans le style itératif.

2 La gestion des formulaires

Les formulaires sont le moyen d'obtenir des informations de la part de l'utilisateur sur la machine cliente. On place un bouton de type « `submit` » dans le formulaire (voir plus loin) et lorsque l'utilisateur clique sur ce bouton, le fichier PHP « d'action » associé au formulaire est exécuté. On disposera, durant cette exécution, des informations que l'utilisateur a renseignées dans le tableau de nom `$_POST`.

Pour créer un formulaire, on place dans le fichier HTML envoyé au client :

```
<form method="post" action="unFichier.php"> //on n'utilise jamais method="get"
  ...un corps de texte en HTML, dont des éléments spécifiques aux formulaires...
</form>
```

Il est possible de structurer un formulaire en sous-parties s'il est un peu long. Dans ce cas, à l'intérieur du corps de texte HTML du <`form`> `...`</`form`> on peut créer des blocs avec :

```
<fieldset>
  <legend>Titre de cette partie du formulaire</legend>
  ...
</fieldset>
```

mais un bouton de soumission du formulaire enverra tous les champs du formulaire à l'action qui doit les traiter, jamais seulement ceux d'un fieldset.

L'élément de formulaire le plus courant est

```
<input type="..." name="..." ..et options diverses.. />
```

Lorsque le formulaire est soumis, le programme PHP d'action récupère les champs du formulaire dans le tableau `$_POST[]` et les éléments de ce tableau sont les différentes valeurs des `name="..."`, tandis que leur valeur est la réponse de l'utilisateur à cet <`input.../`>.

Il y a différents types d'`input` avec des comportements spécifiques :

- Pour le type `"text"`, l'utilisateur pourra entrer une ligne de texte et les options utiles sont
  - `size=".."` – la largeur de la zone de saisie en nombre de caractères
  - `maxlength=".."` – le nombre maximum de caractères autorisé en saisie pour l'utilisateur
  - `value="..."` – la réponse par défaut, si pas modifiée par l'utilisateur
- Pour le type `"password"`, l'utilisateur est invité à entrer un mote de passe (qui ne sera pas affiché à l'écran) et les options utiles sont `size` et `maxlength`.

— Pour le type `"radio"`, chaque `input` de type radio affiche généralement un rond cliquable. L'utilisateur devra choisir un et un seul « `input radio` » parmi ceux proposés sous le même nom. L'attribut `name` joue donc un rôle spécial puisqu'il il permet de grouper sous le même nom les boutons qui sont en choix exclusif : cliquer sur un rond désélectionne automatiquement les autres de même nom. Les options utiles sont :
  — `value="..."` qui est la valeur qu'on récupère si cet `input radio` est choisi. Si par exemple `name="toto"`, c'est donc la valeur de `$_POST["toto"]`.
  — `checked="checked"` permet de choisir celui des `input radio` qui est coché par défaut parmi ceux de même nom.
— Le type `"checkbox"` est similaire au type `radio` mais affiche généralement un carré cliquable et surtout `plusieurs` boutons peuvent être cochés. Il y a alors deux stratégies :
  — avoir un attribut `name` différent pour chaque bouton et dans ce cas on choisit généralement un attribut `value` de même valeur que le nom car une habitude en PHP est de considérer que si `$_POST['toto']=='toto'` alors c'est que `'toto'` est un champs booléen qui vaut `TRUE` et `$_POST['toto']` est vide si pas coché,
  — ou bien avoir un attribut `name` qui soit un tableau, ce qui permet de regrouper plusieurs `input checkbox` sous un même tableau. Dans ce cas il faut écrire « `name="tab[]"` » (i.e. il faut mettre les crochets derrière le nom) en donnant le nom qu'on veut au tableau à la place de `"tab"`. Dans ce cas, l'option `value` est une valeur que l'on retrouvera dans le tableau `tab[]` si le `input checkbox` est coché : `$_POST[tab][0]`, `$_POST[tab][1]`, *etc.*
— Pour le type `"file"`, l'utilisateur pourra envoyer un fichier vers le serveur PHP. L'`input` se présente comme le type `"text"`, mais avec un sélecteur de fichiers associé à la fin de la ligne, pour lui éviter d'avoir à taper l'adresse du fichier. On peut éventuellement limiter le type MIME du fichier qu'on accepte de recevoir avec l'option `accept="..."` dont la valeur doit alors être une liste de types MIME séparés par des virgules. Par exemple `"image/png,image/jpeg"`.
  Ce n'est plus le tableau `$_POST` qui donne les informations mais le tableau `$_FILES[]`. Supposons que name="toto" :
  — `$_FILES["toto"]["name"]` est le nom du fichier chez le client,
  — `$_FILES["toto"]["type"]` est son type MIME,
  — `$_FILES["toto"]["size"]` est sa taille,
  — `$_FILES["toto"]["tmp_name"]` est le nom temporaire sur le serveur du fichier envoyé par le client. Pour ne pas le perdre il faut le ranger quelquepart avec la commande :
  ```
  move_uploaded_file($_FILES["toto"]["tmp_name"] , "adresseChoisie")
  ```
  — `$_FILES["toto"]["error"]` dit ce qui a échoué si le fichier n'est pas parvenu jusqu'au serveur PHP...
— Le type `"reset"` permet de réinitialiser le formulaire et l'option `value="..."` est alors le nom écrit sur le bouton de réinitialisation.
— Enfin le type `"submit"` déclenche l'envoi au serveur des données saisies par l'utilisateur dans le formulaire. L'option `value="..."` est le nom écrit sur le bouton d'envoi.

Un autre élément de formulaire très courant est :

```
<textarea name=".." cols=".." rows="..">
  ..valeur par défaut..
</textarea>
```

C'est comme un `input` de type `"text"` mais avec plusieurs lignes possibles et la zone de saisie peut généralement être redimensionnée par l'utilisateur client.

Enfin on peut créer des menus avec l'élément :

```
<select name="..">
  <option value="valeur-1">texte-1 affiché dans le menu</option>
  <option value="valeur-2">texte-2 affiché dans le menu</option>
  ...
  <option value="valeur-n">texte-n affiché dans le menu</option>
</select>
```

et dans l'une des options on peut ajouter `selected="selected"` qui en fait le choix par défaut.

Enfin, il se peut qu'à l'issue de la réception d'un formulaire on souhaite envoyer un mail au client, en supposant qu'on dispose de son adresse mail bien sûr. La fonction `mail()` prend en arguments, dans cet ordre, l'adresse mail du destinataire, l'objet du mail et enfin le texte du mail. Naturellement ces arguments, surtout le texte du mail, peuvent être longs, et sont généralement calculés au préalable dans des variables de type `string`. *A noter* : cette fonction est `mail()` souvent désactivée sur les serveurs de pages web publiques, pour éviter les spams...

3 Les variables globales du serveur

— `$_SERVER['DOCUMENT_ROOT']` donne la racine du serveur,
— `$_SERVER['HTTP_HOST']` donne le nom du serveur,
— `$_SERVER['SERVER_ADDR']` donne l'adresse IP du serveur,
— `$_SERVER['HTTP_USER_AGENT']` donne le navigateur du client,
— `$_SERVER['REMOTE_ADDR']` donne l'adresse IP du client,

et bien d'autres...