Développement d'Applications Web

PHP et la Programmation Impérative:

Programmation Procédurale

Présenté par :

Me Derkaoui Orkia

Université Docteur Moulay TAHAR de Saida

derkaouiorkia@gmail.com

Plan

- Introduction: la programmation impérative
- Les variables
- Les fonctions
- La récursivité
- La répétition: les boucles

Classification des Langages de Programmation

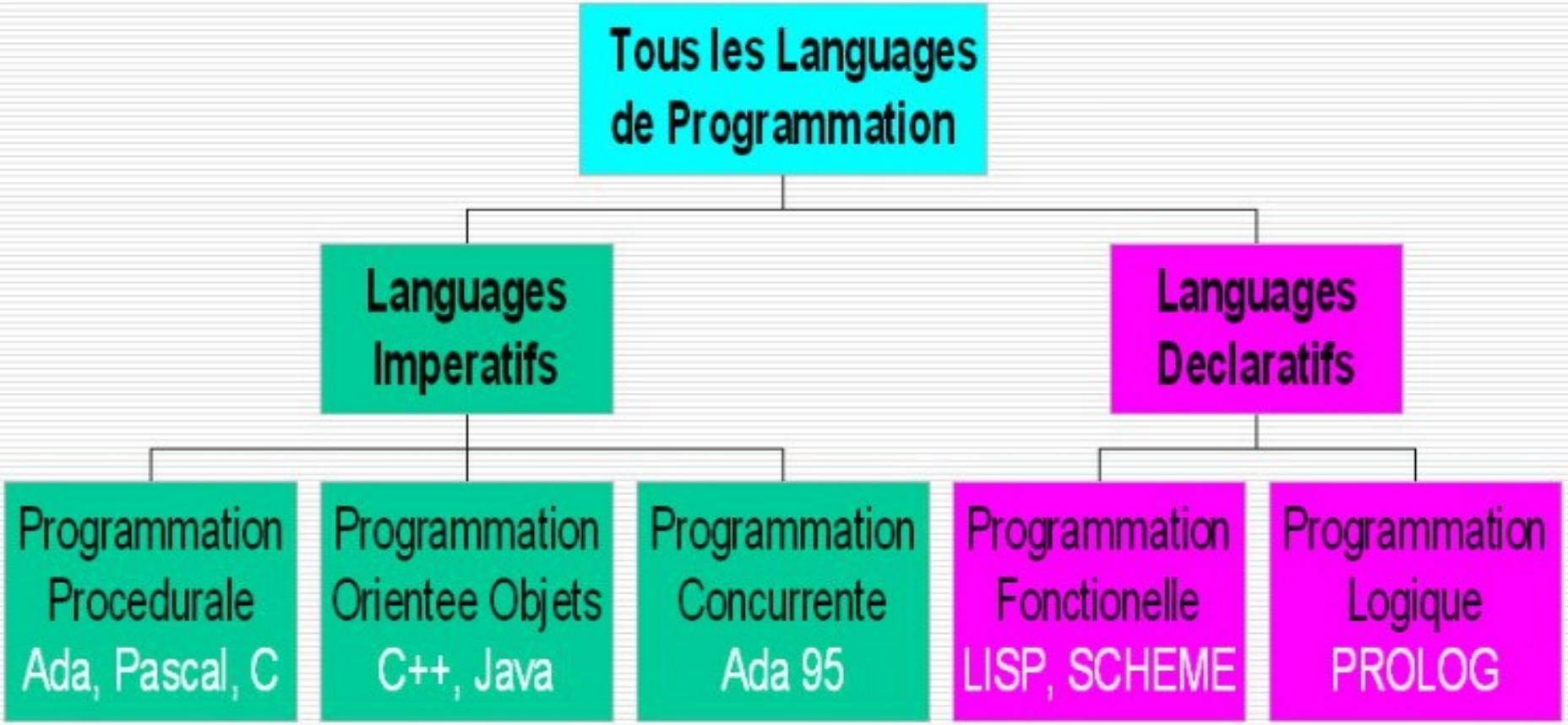

Image empruntée à [6]

La programmation impérative

Qu'est-ce que la programmation impérative ?

- La programmation impérative est l'un des deux principaux paradigmes de programmation de l'informatique.
- La programmation impérative

(du latin imperare = ordonner) est le paradigme de programmation le plus ancien. Ce paradigme définit un programme comme une **séquence clairement définie d'instructions** informatiques.

- Le code source des langages impératifs énonce donc des séquences d'ordres, déterminant quand l'ordinateur doit exécuter quelle action pour atteindre le résultat souhaité. Les valeurs utilisées dans les variables sont alors transformées en durée d'exécution du programme.

La programmation impérative

- Pour piloter les ordres, des **structures de contrôle, telles que des boucles ou embranchements, sont intégrées au code**.
- Le paradigme de programmation impérative est très concret et fonctionne en étroite collaboration avec le système.
- Le code est aisément compréhensible, mais l'écriture **de nombreuses lignes de code source** est nécessaire pour décrire ce que les langages de programmation déclarative obtiennent en quelques mots.

La programmation impérative

Les langages de programmation impérative les plus connus :

- **Fortran**
- **Java**
- **Pascal**
- **ALGOL**
- **C**
- **C#**
- **C++**
- **Assembler**
- **BASIC**
- **COBOL**
- **Python**
- **Ruby**

La programmation impérative

Les différents langages de programmation impérative peuvent encore être subdivisés en trois autres styles de programmation secondaires :

1. le style de programmation structuré,
2. procédural
3. modulaire.

La programmation impérative

- Le **style de programmation structuré** étend le principe impératif de base à des **structures de contrôle** concrètes : séquences, sélection et itération (répétition).
- La maîtrise ou l'évitement complet des instructions de saut, qui rendent le code impératif inutilement compliqué, se font en arrière-plan.

La programmation impérative

- L'**approche procédurale** divise la tâche à exécuter en plusieurs sous-tâches, que le code doit alors décrire séparément.
- Les blocs de programmation ainsi créés peuvent ensuite être réutilisés au sein d'autres programmes.
- **Cette approche, entre autres est utilisé en programmation avec le langage PHP:**

fonction

La programmation impérative

- Le **modèle de programmation modulaire** va encore plus loin en concevant, développant et testant les différents composants du programme indépendamment les uns des autres.
- Ce n'est qu'à la fin que la combinaison des différents modules forme le logiciel proprement dit.

Les variables en PHP

- En PHP, une **variable est un conteneur qui va nous permettre de stocker des informations** de différents types (texte, entier, booléen, etc.).
- Elles ne servent qu'à **stocker des informations temporairement**.
- En effet, elles n'existent que durant l'exécution du code PHP ; elles ne persistent pas dans le temps.
- Si on souhaite sauvegarder durablement des informations, il faut se tourner vers une autre méthode : base de données, fichiers, cookies par exemple.
- Cependant, la valeur d'une variable peut évoluer au fil de l'exécution d'un code PHP.

Les variables en PHP

CRÉER OU DÉCLARER UNE VARIABLE EN PHP

- Comme énoncé précédemment, une variable nous permet de stocker temporairement des informations. Chaque variable est identifiée au sein du code PHP par un **nom unique**. Cela permet de les distinguer les unes des autres et de les utiliser.
- A l'instar des identifiants en HTML, on peut identifier nos variables avec n'importe quel nom.
- Cependant, il y a tout de même **quelques règles à respecter**. Ainsi, pour déclarer une variable en PHP, il faut préciser le signe $ (dollar) avant chaque nom de variable, exemple : $nom = "Mohamed";
- Le nom ne doit pas contenir d'espace, ni de caractère spécial (-, !, %, etc.) hormis le _ (underscore).

Les variables en PHP

- De plus, une variable peut seulement commencer par une lettre ou un underscore.
- Les chiffres sont autorisés mais pas en premier caractère.

```
<?php
$test = 1;
$_test = 2;
$test_1 = 3;
?>
```

- Les **noms de variables sont sensibles à la casse**. Donc les variables $test, $TEST et $Test sont bien trois variables distinctes.

Les variables en PHP

Types des variables :

Les variables en PHP vont pouvoir stocker 8 grands types de données différents :

- Le type « chaine de caractères » ou String en anglais ;
- Le type « nombre entier » ou Integer en anglais ;
- Le type « nombre décimal » ou Float en anglais ;
- Le type « booléen » ou Boolean en anglais ;
- Le type « tableau » ou Array en anglais ;
- Le type « objet » ou Object en anglais ;
- Le type « NULL » qui se dit également NULL en anglais ;
- Le type « ressource » ou Resource en anglais ;

Les fonctions en PHP

Les fonctions ont plusieurs buts :

- Éclaircir le code en regroupant dans une même fonction

Certaines fonctionnalités d'un programme qui se répètent.

- Les fonctions sont

utilisées dans d'autre programme ce qui évite de répéter

pour chaque projet le même code.

On distingue 2 types de fonctions :

- **Les fonctions intégrées ou built-in** qui sont inclues par défaut avec les distributions de PHP comme print, echo et
- **Les fonctions définies par le programmeur** selon son besoin.

Les fonctions

Contenu d'une fonction

```
<?php
Function ma_fonction($paramtre1, $parametre2,…)
        //déclaration
{

        // code de la fonction
        // …

return($une_variable);    // facultatif


}


$retour = ma_fonction(2, 5); //appel
?>
```

Les fonctions intégrées ou built-in

La Liste de toutes les fonctions et méthodes du PHP

https://www.php.net/manual/fr/indexes.functions.php

Change language: French

Submit a Pull Request Report a Bug

Liste des fonctions et des méthodes

Liste de toutes les fonctions et méthodes du manuel

a b c d e f g h i j k l m n o p q r s t u v w x y z _

- a
  - abs - Valeur absolue
  - acos - Arc cosinus
  - acosh - Arc cosinus hyperbolique
  - addcslashes - Ajoute des slash dans une chaîne, à la mode du langage C
  - addslashes - Ajoute des antislashs dans une chaîne
  - AllowDynamicProperties::__construct - Construit une nouvelle instance d'attribut AllowDynamicProperties
  - apache_child_terminate - Termine le processus Apache après cette requête
  - apache_getenv - Lit une variable de processus Apache
  - apache_get_modules - Retourne la liste des modules Apache chargés
  - apache_get_version - Récupère la version d'Apache

Les fonctions définies par le programmeur

Selon notre besoin, le programmeur peut créer des fonctions qui réalise des fonctions spécifiques.

Exemple: fonction qui affiche une chaine de caractères en commençant par le dernier caractère :

```
<html><body>
<?php
$str = "Hello World !";

function reverse_i($str)
{
 for($i=1;$i<=strlen($str);$i++)
 echo substr($str, -$i, 1); //affiche un caractère en partant de la fin
 return;
}
reverse_i($str);

?>
</body></html>
```

Les fonctions et Utilisation des paramètres

Les paramètres d'une fonction peuvent être passés de 2 façons différentes :

- **Par valeur**, c'est a dire que s'ils ont une valeur à l'extérieur de la fonction, seule la valeur est transmise à la fonction, si la variable subit des modifications à l'intérieur de la fonction, ces modifications ne seront pas perçues dans le programme principal
- **Par référence**, avec le signe & avant la variable

  (ex : &$cpt).

Dans ce cas, l'adresse mémoire de la variable dans le programme est passée à la fonction et toute modification de cette variable dans la fonction aura des répercutions à l'extérieur du programme.

Par valeurs

```php
$var1=5;
$var2=9;
echo "Avant la permutation des valeurs des variables ","<br/>";
echo "var1=$var1 <br/>";
echo "var2=$var2 <br/>";

// L'échange attendu ne se produira pas car la fonction à reçu uniquement la valeur de la variable.

echange1($var1,$var2);
echo "Après l'appel de la fonction de permutation des valeurs des variables appel: par valeurs (pas de permutation)","<br/>";
echo "var1=$var1 <br/>";
echo "var2=$var2 <br/>";

function echange1($a,$b) {
    $tmp=$a;
    $a=$b;
    $b=$tmp;
}
```

Par référence

```php
$var1=5;
$var2=9;

    // L'échange attendu se produira pas car la fonction à reçu l'adresse de la variable.

echange2($var1,$var2);
echo "Après l'appel de la fonction de permutation des valeurs des variables : appel par référence (il y'a permutation)","<br/>";
echo "var1=$var1 <br/>";
echo "var2=$var2 <br/>";


function echange2(&$a,&$b) {
    $tmp=$a;
    $a=$b;
    $b=$tmp;
}
```

Références Bibliographiques

[1] http://icps.u-strasbg.fr/~genaud/courses/webtech/php/ch02.html

[2] http://ivmad.free.fr/pi/PHP/php-1.html

[3] https://www.php.net/manual/fr/indexes.functions.php

[4] https://www.ionos.fr/digitalguide/sites-internet/developpement-web/programmation-imperative/

[5] https://www.invivoo.com/paradigme-de-programmation/

[6] https://slideplayer.fr/slide/2736836/