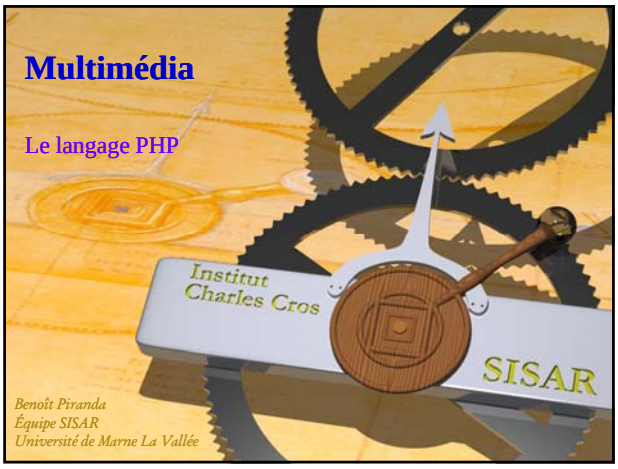
Multimédia

Le langage PHP



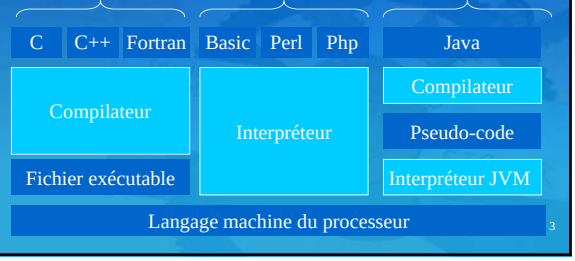
Benoît Piranda
Équipe SISAR
Université de Marne La Vallée

Niveau d'abstraction

- Langage de haut niveau
  - Développements logiciels
    - C, C++, Java
  - Applications scientifiques
    - C, C++, Fortran
  - Scripts systèmes
    - Shell, Perl, PHP
  - Application pour Internet
    - PHP, Java, JavaScript, C, C++
- Langage de bas niveau
  - Assembleur, langage machine

Différents types de langage

- Langages compilés
- Langages interprétés
- Langage Mixte



Langages interprétés

- L'analyse et la traduction du langage de haut niveau se fait pendant l'exécution
- Avantages
  - langage indépendant
  - exécution immédiate du code
  - recherche des erreurs d'exécution
    - état du programme au moment de l'erreur
    - poursuite du programme après correction
- Inconvénients
  - la même analyse est faite à chaque exécution
    - perte de temps
  - impossibilité de mélanger le code avec d'autres langages

Langages compilés

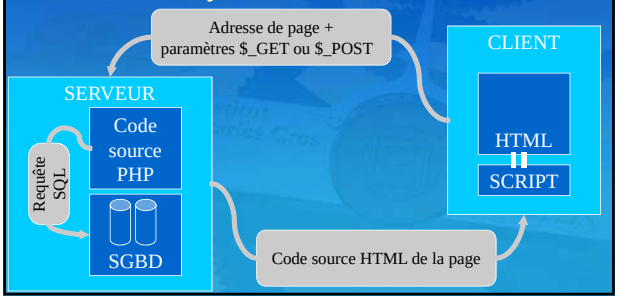
- Traitement du langage de haut niveau en deux étapes :
  - la compilation : traduction en langage machine
  - l'exécution par le système hôte
- Avantages
  - optimisation du code à priori
    - vitesse d'exécution plus grande
  - possibilité de faire coexister plusieurs modules
    - pouvant être produits par différents langages
- Inconvénients
  - temps de compilation
  - correction des erreurs plus complexes

Langage PHP : pour des sites dynamiques

- Langage
  - Multi plate-forme, intégré dans le HTML
  - Gratuit
- Dédié à :
  - La réalisation de site Internet interactif
    - Adaptation du code aux conditions d'exécution (date, lieu…)
    - Gestion de sessions, cookies…
  - La gestion de bases de données
    - MySQL, PostGres…
  - Modules graphiques

Langage PHP : pour des sites dynamiques

- **Langage interprété sur le serveur**
  - **Code non visible par le client (confidentialité)**
  - **Orienté objet**



Avantages et inconvénients

- **Interactivité**
  - **Au niveau du client (locale à la machine)**
  - **Durée de communication client/serveur non maîtrisée**
- **Contenu contextuel**
  - **Au niveau du serveur**
- **Sécurité**
  - **code du serveur**
  - **code du client**

Historique du langage PHP

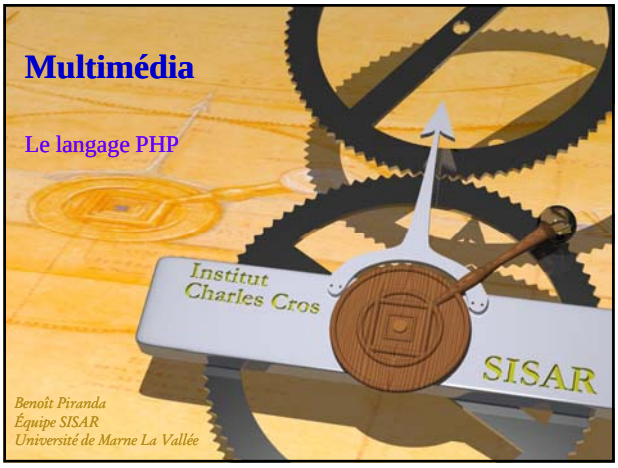
- **Langage récent 1996**
- **Auteur Rasmus Lerdorf**
  - **But : interactivité du site (comptage du nombre de visite)**
- **PHP2**
  - **Code inséré directement à l'intérieur du code HTML**
- **Dernière version : PHP5**
  - **Support XML**
  - **Extension MySQLi (interface objet)**
  - **Extension SOAP : Simple Object Access Protocol**
    - **Échanges standardisés entre acteur du web**

9

Structure d'un programme PHP

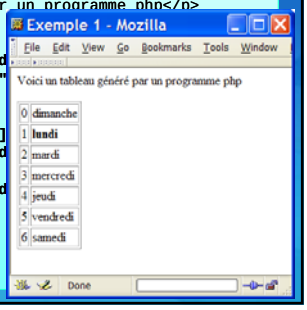
- **S'insère dans un document au format HTML**
- **Extension .php**
  - **Indique au serveur :**
    - **D'interpréter les commandes php placées à l'intérieur**
    - **De les remplacer par le résultat du programme**
- **Entre balises spécifiques**
  - **Style SGML: <? … ?>**
  - **Style XML : <?php … ?>**
  - **Style Active Server Pages : <% … ?>**
  - **Style JavaScript : <script … ?>**

10



Exemple de programme HTML/PHP

```
<html>
<head> <title>Exemple 1</titre> </head>
<body>
<p>Voici un tableau généré par un programme php</p>
<table border=1>
<?php
  $info=array("dimanche","lund
  "jeudi", "vendredi","samedi"
  $info_date = getdate();
  for ($i=0; $i<7; $i++) {
    if ($i==$info_date["wday"]
      echo "<tr><td>$i</td><td
    else
      echo "<tr><td>$i</td><td
  }
?>
</table>
</body>
</html>
```



Structure de langage

- Outils de programmation :
  - Manipulation de la mémoire
    - Affectation : variable=expression
    - Comparaison : expression comparateur expression
  - Traitement conditionnels
    - Réaliser des traitements suivant le résultat d'un comparaison
      - ✓ if (comparaison) traitements_si_vrai
      - ✓ if (comparaison) traitements_si_vrai else traitements_si_faux
      - ✓ …
  - Boucles
    - Répéter automatiquement des traitements
      - ✓ for(affectation ; comparaison ; affectation) traitements
      - ✓ …
- Séparés par des ';'

13

Les commentaires

- Informations complémentaires
  - Permettent
    - D'aérer le code
    - D'expliquer les étapes du programme
  - Syntaxe proche du C++ ou Java
    - Sur une ligne

```
// table des jours de la semaine
  $info=array("dimanche","lundi","mardi","mercredi",
  "jeudi", "vendredi","samedi");
```

    - Sur plusieurs lignes

```
/* teste si la ligne $i correspond au jour courant
   mémorisé dans $info_date */
    if ($i==$info_date["wday"])
     echo "<tr><td>$i</td><td><B>$info[$i] </B></td></tr>";
else echo "<tr><td>$i</td><td>$info[$i] </td></tr>";
  }
/*********************************************/
```

14

Fonctions d'affichage

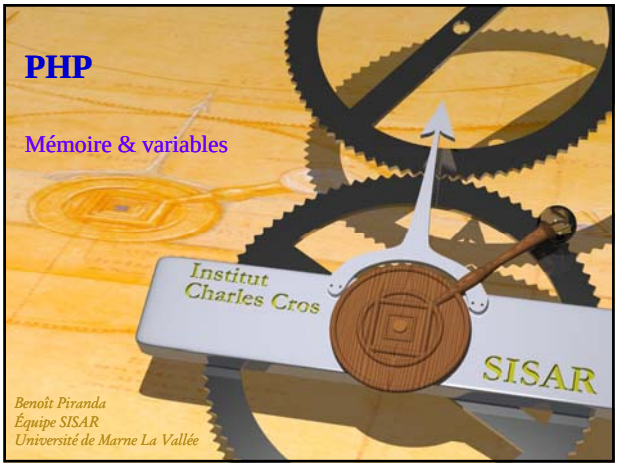
- echo chaîne1,chaîne2…
  - Si paramètre de type chaîne : la chaîne
  - Sinon conversion du contenu de la variable en chaîne
- print(chaîne);
- Version d'affichage sur plusieurs lignes…

```
print <<<BEGINHTML
Texte avec des apostrophes et
des guillements fini par le
mot clé suivit de ";"
BEGINHTML;
```

Fonctions d'affichage

- Sur le modèle du C
  - int printf(format,arguments)
  - $chaine sprintf(format,arguments)
  - int vprintf(format,argument)
  - $chaine vsprintf(format,argument)
  - Formats
    - %s : chaînes de caractères
    - %d : entiers signés
    - %u : entiers non signés
    - %f : réels à virgules flottantes
    - …

```
$data = array('Benoît', 'Piranda',2000);
vprintf("%s %s : année %d");
```



Les types de données de base

- Type de données
  - Signification donnée à une zone mémoire
  - Associé à un nombre d'octets
- Quelques types de base
  - boolean — booléen
  - int — entier long (32 bits)
  - double — réel double précision
  - string — chaînes de caractères

18

Notion de variable

- **Variable**
  - **Permet de mémoriser des informations sous un nom**
  - **Nom de variable**
    - Choisi pour reconnaître la notion associée à la variable
    - Formée de lettres (majuscule ou minuscule), de chiffre ou du caractère '_'
    - Doit commencer par un lettre
    - Apparaît dans le code PHP précédée de $

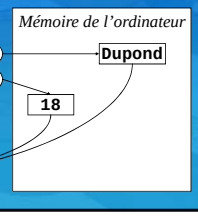
```
$nomEtudiant
$nom_etudiant
$compteur
$mot2passe
```


Notion de variable

- **Variable**
  - **Fait référence à un emplacement mémoire**
    - Permet de placer une valeur en mémoire : affectation
      - ✓ $nom = "Dupond";
    - Permet de connaître la valeur placée en mémoire : lecture
      - ✓ echo $nom;

```
$nom = "Dupond";
$numero = 18;
echo "$numero $nom";
```

```
18 Dupond
```



Affectation d'une valeur à une variable

- **Affectation**
  - **d'une valeur**
    - placer la valeur 10 dans la zone mémoire associée à $x
  - **d'une variable**
    - copier la valeur stockée dans la zone mémoire associée à $y vers la zone mémoire associée à $x
    - $y reste inchangé
  - **du résultat d'une expression**
    - évaluer l'expression et placer le résultat dans la zone mémoire associée à $x

$x = 10

$x = $y

$x = 2 + 3 × $y

Affectation d'une référence

- **Deux identifiants accèdent à la même zone mémoire**
  - **syntaxe : id2 =& id1**

```
$source = 1;
$source2 =& $source;
$source2++;
echo $source; // affiche 2
```

Constantes

- **Expression pouvant être affectée à une variable.**
- **Sa valeur est invariante.**
- **Types implicites**
  - **int**
    - Nombre sans virgule (18)
  - **double**
    - Nombre avec virgule ou notation scientifique (12.6, 5e7)
  - **boolean**
    - TRUE ou FALSE
  - **string**
    - Expression entre guillemets (" ") ou apostrophe (' ')

```
$nom = 'Dupond 20 $x 4';
$chaine = "Frais : $nom <br \>";
echo $chaine;
```

Frais : Dupond 20 $x 4

Vie d'une variable

- **Naissance : déclaration**
  - **Implicite lors de sa première utilisation**
  - **Le type de la variable**
    - Donné par le type du résultat de l'expression affectée
    - NULL : type à déterminer ultérieurement
    - Utilisation d'un opérateur de transtypage (cast)
      - ✓ (int), (double),…

```
<?php
$a = NULL;    // sans type
$b = 0;       // type int, valeur 0
$c = 0.0;     // type double, valeur 0
$d = '';      // type string, valeur chaîne vide
$e = (double)$b // type double, valeur de b : 0
?>
```

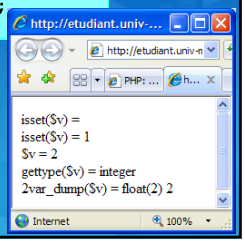
Vie d'une variable

- **Fonction de gestion des variables**
  - Savoir si une variable a été définie :
    - `isset($var);`
  - Savoir si une variable est à NULL :
    - `isnull($var);`
  - Vérifier le type d'une variable :
    - `isint($var);`
    - ...
  - Connaître le type d'une variable, et sa valeur :
    - `gettype($var);`
    - `var_dump($var);`
  - Forcer le type d'une variable :
    - `settype($var,"type");`

25

Exemple

```
<?php
  echo "isset(\$v) = ",isset($v),"<br \>";
  $v = 2;
  echo "isset(\$v) = ",isset($v),"<br \>";
  echo "\$v = $v<br \>";
  echo "gettype(\$v) = ",gettype($v),"<br \>";
  settype($v,"float");
  echo "var_dump(\$v) = ",var_dump($v);
?>
```

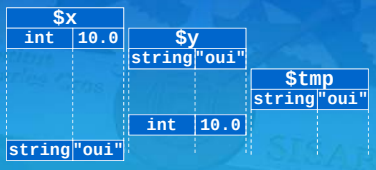


Vie d'une variable

- **Evolution : affectations**
  - Changement de valeur
  - Changement de type

```
<?php
$x = 10.0;
$y = "oui";

$tmp = $y;
$y = $x;
$x = $tmp;
?>
```



27

Vie d'une variable

- **Mort**
  - Automatique si variable plus utilisée ou fin de portée
  - Si utilisation de `unset($var);`
  - Suppression des données associées :
    - affectation à NULL

28

Portée des variables

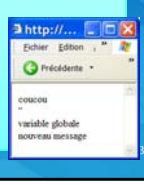
- **Portée d'une variable**
  - zone du programme dans laquelle elle est définie
- **Contexte local**
  - Dans une fonction (sous-programme)
  - Variable déclarée dans la fonction n'existe que pour le corps de la fonction
  - variable "static" : contexte maintenu dans la fonction
- **Contexte global**
  - A l'extérieur des fonctions
  - Variable déclarée pour tout le corps du programme
  - Utilisation d'une variable globale dans une fonction
    - Utilisation de la commande « `global` »

29

Portée des variables (exemple)

```
<?php
$message = "variable globale";
function affichage1()
{ // ici $message n'est pas définie
  $texte1 = "coucou"; // déclaration de $texte1 : locale
  echo $texte1;
} // fin de portée de $texte1

function affichage2()
{ global $message; // la variable globale $message
                   // est aussi définie dans la fonction
 echo $message;
}
affichage1(); echo "<br>'",$texte1,"'<br>";
affichage2();
$message = "nouveau message";
echo "<br>";
affichage2();
?>
```



30

Expressions mathématiques

- **Opérateurs arithmétiques**
  - **les 4 opérations mathématiques**
    - +, -, *, /
  - **Opérations spécifiques**
    - **sur les entiers % (modulo) : reste entier de la division**
    - **Sur les chaînes '.' : concaténation**

Opération	Opérateur	Résultat
$a+$b	Addition	Somme de $a et $b.
$a-$b	Soustraction	Différence de $a et $b.
$a*$b	Multiplication	Produit de $a et de $b.
$a/$b	Division	Quotient de $a et de $b, valeur entière si le résultat est entier, sinon un double.
$a%$b	Modulo	Reste entier de la division de $a par $b. 17 % 3 = 2
$a.$b	Concaténation	La chaîne composée de la chaîne $a suivie de $b.

31

Expressions mathématiques

- **Opérateurs d'incrémentation**

Opération	Opérateur	Résultat
++$a	Pré-incrémentation	Incrémente $a de 1 puis renvoie sa valeur
$a++	Post-incrémentation	Renvoie la valeur de $a puis l'incrémente de 1
--$a	Pré-décrémentation	Décrémente $a de 1 puis renvoie sa valeur
$a--	Post-décrémentation	Renvoie la valeur de $a puis le décrémente de 1

```
<html><body>
<?php
$a = 0;
echo "<p>",$a++,"</p>";
echo "<p>",$a,"</p>";
echo "<p>",--$a,"</p>";
?>
</body></html>
```

```
0
1
0
```

32

Expressions mathématiques

- **Opérateurs d'affectation**

Opération	Opérateur	Résultat
$a = $b	Affectation	$a reçoit la valeur de $b
$a += $b	Affectation et addition	$a est incrémenté de $b ($a = $a+$b)
$a -= $b	Affectation et soustraction	$a est décrémenté de $b ($a = $a-$b)
$a *= $b	Affectation et produit	$a est multiplié par $b, le résultat est placé dans $a. ($a = $a*$b)
$a /= $b	Affectation et division	$a est divisé par $b, le résultat est placé dans $a, son type peut changer. ($a = $a/$b)
$a %= $b	Affectation et modulo	$a est divisé par $b, le reste entier est placé dans $a. ($a = $a%$b)
$a .= $b	Affectation et Concaténation	L'expression $b est concaténée à la chaîne $a, le résultat est placé dans $a

33

Expressions mathématiques

- **Opérateurs de comparaison**
  - **Retourne Vrai (TRUE) si la comparaison de deux expressions se vérifie, Faux (FALSE) sinon.**

Opération	Opérateur	Résultat
$a > $b	Supérieur	Vrai si $a est strictement plus grand que $b
$a >= $b	Supérieur ou égal	Vrai si $a est plus grand ou égal à $b
$a == $b	Égal	Vrai si $a a la même valeur que $b
$a === $b	Identique	Vrai si $a a la même valeur et le même type que $b
$a != $b $a <> $b	Différent	Vrai si $a et $b ont des valeurs différentes
$a < $b	Inférieur	Vrai si $a est strictement plus petit que $b
$a <= $b	Inférieur ou égal	Vrai si $a est plus petit ou égal à $b

34

Expressions mathématiques

- **Opérateurs logiques**

Opération	Opérateur	Résultat
$a && $b $a AND $b	Et	Vrai si les expressions $a et $b sont vraies. ($a>=0 && $a<=20)
$a || $b $a OR $b	Ou	Vrai si au moins l'une des expressions $a ou $b est vraie. ($a<0 || $a>20)
$a ^ $b	Ou exclusif (XOR)	Vrai si une seule des expressions $a ou $b est vraie.
! $a	Non	Vrai si $a est faux.

35

Propriétés

- **a ou b ⇔ non ((non a) et (non b))**
  - **($a>=0 && $a<=20) ⇔ !($a<0 || $a>20)**
- **a et b ⇔ non ((non a) ou (non b))**

a et b

ET	0	1
0	0	0
1	0	1

(non a) et (non b)

ET	!0	!1
!0	1	0
!1	0	0

non((non a) et (non b))

!ET	!0	!1
!0	0	1
!1	1	1

a ou b

OU	0	1
0	0	1
1	1	1

36

Les chaînes de caractères

- Omniprésente dans les pages HTML
- Fonctions de haut niveau
  - But : Éviter de parcourir une chaîne
  - Lectures de variables dans une chaîne
  - Accéder à un caractère
  - Taille d'une chaîne
  - Lister les mots d'une chaîne
  - Position d'une sous-chaîne
  - Extraire une sous-chaîne
  - Remplacer un motif
  - Changement de la casse

Fonctions de haut niveau

- Lecture d'une variable dans une chaîne
  - Fonction à la mode C
  - tab=sscanf(source,masque);

```
$source = '3.14 pi pour 3.141592654';
list($abreg,$nom,$val) = sscanf($source,"%f %s pour %f");
```

- Accéder à un caractère
  - Chaîne de caractères se comportant comme un tableau

```
$source = 'master informatique';
echo $source[0],"<br \>";
echo ord($source[1]),"<br \>";
$source[6]=chr(123);
echo $source,"<br \>"
```

```
m
97
master{informatique
```

Fonctions de haut niveau

- Taille d'une chaîne
  - int strlen(string) : nombre de caractères
  - int str_word_count(string) : nombre de mots
- Lister les mots d'une chaîne
  - int str_word_count(string,codage)
    - Codage = 1 : numéro_mot => mot
    - Codage = 2 : position_mot => mot

```
$chaine = "master informatique premiere annee";
echo $chaine,"<br \>";
$tab = str_word_count($chaine,1);
var_dump($tab);
echo "<br \>";
$tab = str_word_count($cha
var_dump($tab);
```

```
array(4) { [0]=> string(6) "master" [1]=> string(12) "informatique"
[2]=> string(8) "premiere"  [3]=> string(5) "annee" }
array(4) { [0]=> string(6) "master" [7]=> string(12) "informatique"
[20]=> string(8) "premiere" [29]=> string(5) "annee" }
```

Fonctions de haut niveau

- Position d'une sous-chaîne
  - strpos(chaine,sous-chaine)
    - retourne FALSE si la chaine n'est pas trouvée
    - l'indice sinon
    - attention au type de la valeur retournée FALSE ou 0 ?
- Extraire une sous-chaîne
  - substr(chaine,position,longeur)

```
$chaine = 'benoit.piranda@univ-mlv.fr';
$pos = strpos($chaine,'@');
if ($pos===FALSE)
{ echo 'caractère non présent';
} else
{ echo "caractère en position $pos";
}
```

Fonctions de haut niveau

- Remplacer un motif
- Changement de la casse
- Protections de chaîne de caractères
  - Pour le code source PHP
    - addslashes(string) : préfixe automatiquement les guillemets
    - addcslashes(string) : converti aussi les caractères de fin de ligne
  - Pour les requêtes SQL
    - mysql_escape_string(string)
    - pg_escape_string(string)

Fonctions sur les dates et heures

- Très importantes en HTML dynamique
  - Traitement dépendant de l'instant d'appel de la page
    - Absolu
      - ✓ Afficher la date courante,
      - ✓ Gérer un calendrier
    - Relative à une autre date
      - ✓ Connexion à un site
      - ✓ Évolution d'une base de données

fonction de manipulation des dates

- formater une date dans une chaînes de caractère
  - date(format,timestamp)
    - si timestamp n'est pas renseigné : date courant
    - format donné par une chaîne de caractères avec jokers
      - ✓ d : jour du mois 01 à 31
      - ✓ m : mois 01 à 12
      - ✓ y : année sur deux chiffres
      - ✓ H : heure en format 24 heures
      - ✓ i : minutes
      - ✓ …

```
$format = 'd-m-y, H:i:s';
$d = date($format);
echo $d;
```

Fonctions de manipulation des dates

- Conversion chaine vers date
  - strtotime(chaine);
- Détermination d'un timestamp
  - mktime(heure,minute,seconde,mois,jour,année,is_dst)
  - les paramètres absents sont remplacés par les valeurs de l'heure courante

Les tableaux

- Notion de tableau en informatique
  - Regrouper plusieurs variables de même type
  - Un tableau est un bloc mémoire contigu
    - Une variable est associée à une position (un entier) dans le tableau
    - Chaque élément est directement accessible en indiquant son numéro d'ordre
  - Sa taille est fixe (définie lors de son allocation)

47

Les tableaux en PHP

- Notion encore plus riche, s'apparentant à la notion d'ensemble
- Ensemble de données (des variables)
  - De types hétérogènes
  - Associant une clé (identifiant unique) à une valeur
    - La clé peut être un indice, ou n'importe quel identifiant unique permettant d'accéder à une position du tableau

Clé	Variable
0	Dimanche
1	Lundi
2	Mardi
3	Mercredi
4	Jeudi
5	Vendredi
6	Samedi

Clé	Variable
Nom	Dupond
Prénom	Jean
Titre	Ma vie
Année	2000
Mois	04

48

Manipulation des tableaux

- **Déclaration / Affectation**
  - **Tableau = variable + suffixe '[' clé ']'**
    - `$tab[1]=10;`
    - `$tab["nom"]="dupond";`
  - **`count(tableau)` : nombre d'éléments dans le tableau**

```
$tab[]="Dupond";
$tab[]="Paul";
$tab[]="Ma vie";
$tab[]=2000;
```

Place la valeur "Dupond" dans la première place libre du tableau `$tab`. Si c'est une déclaration, le rang initial est 0

0	Dupond
1	Jean
2	Ma vie
3	2000

```
echo count($tab);
```

Affiche 4

49

Les tableaux associatifs

- **indexage par une clé quelconque dans un tableau**

```
$tab["Nom"]="Dupond";
$tab["prénom"]="Paul";
$tab["Titre"]="Ma vie";
$tab["Année"]=2000;
```

Nom	Dupond
Prénom	Jean
Titre	Ma vie
Année	2000

```
echo count($tab);
```

Affiche 4

50

Déclaration à l'aide de array

- **Outils permettant**
  - **de déclarer**
  - **et initialiser simultanément un tableau**
- **Liste de couples (clé => valeur)**
  - **Clé optionnelle, remplacée par des valeurs par défaut**
    - **0 si première**
    - **Dernière clé + 1 sinon**

```
$tab=array ("Nom"=>"Dupond", "Prénom"=>"Pierre",…);
$tab=array ("Dimanche","Lundi","Mardi",…);
$tab=array (1=>"Dimanche","Lundi",5=>"Mardi",…);
```

51

Parcours des tableaux

- **Boucle de parcours dédiée au tableau**
  - **Permet de passer en revue tous les éléments d'un tableau pour leur appliquer le même ensemble d'instructions**
  - **Instruction `foreach`**

```
foreach ($table as $cle=>$element) instructions;
foreach ($table as $element) instructions;
```

```
echo "<table border=1>";
echo "<tr><th>Id</th>";
echo "<th>Valeur</th></tr>";
$tab=array('Nom'=>'Dupond','prénom'=>'Paul',
           'Titre'=>'Ma vie','Année'=>2000);
foreach ($tab as $cle=>$elemt) {
  echo "<tr><td>$cle</td><td>$elemt</td></tr>";
}
echo "</table>";
```

Id	Valeur
Nom	Dupond
prénom	Paul
Titre	Ma vie
Année	2000

52

Tableau à deux dimensions

- **Tableau de tableaux (lignes)**

index	élément
0	[23,45,34,54]
1	["truc",4,5,3,4,5,4]
2	[4]
3	[23,45,34,6,75,84,93,45,34,54]

```
$tab=array( array(23,45,34,54),
            array("truc",4,5,3,4,5,4),
            array(4),
            array(23,45,34,6,75,84,93,45,34,54));
```

53

Exemple d'une matrice

```
$mat= array(array(1.0,0.5,0.2),
            array(0.1,1.0,0.3),
            array(0.2,0.6,1.0),
            array(0.0,0.1,1.0));
echo "<table>";
foreach ($mat as $ligne) {
  echo "<tr>";
  foreach ($ligne as $cel) {
    echo "<td>$cel</td>";
  }
  echo "</tr>\n";
}
echo "</table>";
```

1	0.5	0.2
0.1	1	0.3
0.2	0.6	1
0	0.1	1

54

Les fonctions sur les tableaux

- Les fonctions sont très nombreuses
  - Simplifient les accès aux tableaux
  - Permettent de les exploiter comme des bases de données
- Catégories :
  - Fonctions de créations
  - Fonctions de comparaison
  - Fonctions de tri
  - Fonctions de parcours
  - Fonctions d'extraction de données

55

Fonctions de créations

- Combinaison de deux tableaux
  - Fonction : `array_combine(key,values)`
  - Paramètres :
    - Un tableau de clés
    - Un tableau de données
  - Résultat :
    - Une liste de couples (clé=>donnée)

56

Fonctions de créations

- Concaténation de plusieurs tableaux
  - Fonction : `array_merge(tab1,tab2,…)`
  - Paramètres :
    - Liste de tableaux
  - Résultat :
    - Un tableau unique contenant consécutivement les données de tab1, tab2, …
- Remplissage
  - à partir d'une chaine
    - construit un tableau à partir d'éléments séparés par un caractère spéciale (style csv)
    - $tab = explode(';','Pierre;Paul,Jean');

57

Fonctions d'affichage

- Affichage des éléments d'un tableau
  - print_r(array…)
    - Pour le débugage uniquement
    - Pas de mise en forme
- Construction d'une chaîne à partir d'un tableau
  - implode(séparateur,tableau);

```
$tab=array('Nom'=>'Dupond','prénom'=>'Paul',
           'Titre'=>'Ma vie','Année'=>2000);
print_r($tab);
echo '<br \>',implode(';',$tab);
echo '<table><tr><td>',implode('</td><td>',$tab),'</td></tr></table>';
```

Array ([Nom] => Dupond [prénom] => Paul [Titre] => Ma vie [Année] => 2000)
Dupond;Paul;Ma vie;2000

Dupond	Paul	Ma vie	2000

Fonctions de tri

- Les tris peuvent être de natures multiples
  - Sur les données :
    - croissant : `sort(tableau)`, décroissant : `rsort(tableau)`
  - Sur les clés :
    - `ksort(tableau)`, `krsort(tableau)`
  - En tenant compte des valeurs numérique
    - `natsort(tableau)`
  - En ne tenant pas compte de la casse
    - `natcasesort(tableau)`
  - En programmant son propre critère de comparaison
    - `usort(tableau)`
  - Et bien d'autres…

59

Fonctions d'extraction de données

- Recherche dans un tableau
  - D'un élément : `in_array(expression, tableau)`
  - D'une clé : `array_search(expression, tableau)`
  - Nombre d'occurrences d'un élément : `array_count_values(tableau)`
    - Retourne un tableau de nombre d'apparitions

```
$tab = array('jean','paul','pierre','jean','pierre','jean',
'jacques');
print_r($tab);
echo '<br/>jacques :',in_array('jacques',$tab);
echo '<br/>roger :',in_array('roger',$tab);
echo '<br/>jacques :',array_search('jacques',$tab),'<br/>';
$cpt = array_count_values($
print_r($cpt);
```

Array ([0] => jean [1] => paul [2] => pierre [3] => jean [4] => pierre [5] => jean [6] => jacques)
jacques :1
roger :
jacques :6
Array ([jean] => 3 [paul] => 1 [pierre] => 2 [jacques] => 1)

Utilisation de tableau

- **Structure abstraite**
  - **Pile**
    - array_push(), Array_pop()
  - **File**
    - array_unshift(), array_shift()

```
$tab = array('jean','pierre', 'jacques');
print_r($tab);
array_push($tab,'robert');
echo '<br/>';print_r($tab);
echo '<br/>',array_pop($tab),',',array_pop($tab),'<br/>';
array_unshift($tab,'jules');
print_r($tab); echo '<br/>',array_shift($tab);
```

```
Array ( [0] => jean [1] => pierre [2] => jacques )
Array ( [0] => jean [1] => pierre [2] => jacques [3] => robert )
robert,jacques
Array ( [0] => jules [1] => jean [2] => pierre )
jules
```

Les tests

- **Réaliser des instructions différentes suivant la valeur d'une expression**
- **2 types de tests**
  - **Tests simples**
    - Évaluation d'une expression
    - Si l'expression est vraie (!=0) réaliser $instructions_1$
    - Si non réaliser $instructions_2$
  - **Tests multiples**
    - Suivant la valeur d'une variable
    - Cas v_i : $instructions_i$
    - Cas par défaut : $instruction_{défaut}$

63

Test si alors

- **Un traitement est réalisé uniquement si une expression est vraie (sinon pas de traitement)**
  - **Mot clé `if`**

```
if (expression)
    instructions
```

```
if ($x<$y) {
  $t=$x;
  $x=$y;
  $y=$t;
}
// x est supérieur ou égal à y
```

64

Test si alors sinon

- **Traitements différents pour les deux cas expression vrai ou expression fausse**
  - **Mots clés `if else`**

```
if (expression)
    instructions1     // cas vrai
else
    instructions2     // cas faux
```

```
if ($x<0.0) {
  echo "racine réelle de ", $x ," non définie";
}
else {
  echo "racine réelle de ", $x ," = " + Math.sqrt($x));
}
```

Imbrication de tests

- **Répondre à la question :**
  - **Si la première condition est vérifiée alors si la seconde est vérifiée faire les instructions 1 sinon les instructions 2**

```
if (cond1) {
  if (cond2) {
    instructions1
  }
  else {
    instructions2
  }
}
```

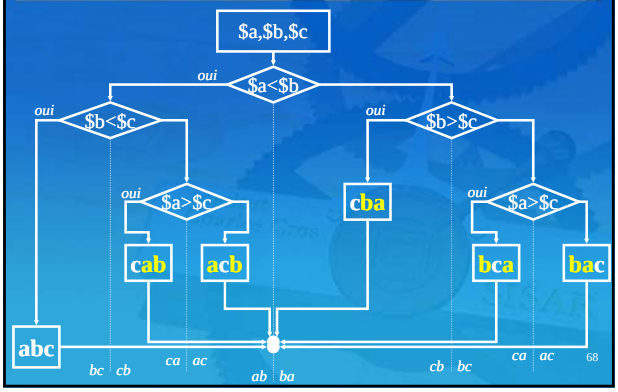
66

Exemple de tests imbriqués

- **Soit trois variables entières $a, $b et $c, échanger leur valeur de façon à ce que la relation $a<$b<$c soit vraie.**
- **Étude de cas :**
  - **Si $a < $b**
    - **Si $b < $c : (abc) il ne faut rien faire**
    - **Si $b > $c : (acb) intervertir $b ⇆ $c**
    - **Si $a > $c : (cab) $a ⇆ $c puis $b ⇆ $c**
  - **Sinon**
    - **Si $b > $c : (cba) intervertir $a ⇆ $c**
    - **Si $a > $c : (bca) $a ⇆ $c puis $a ⇆ $b**
    - **Si $a < $c : (bac) $a ⇆ $b**

67

Exemple de tests imbriqués



68

Test suivant valeurs … cas

```
switch (variable) {
  case constante1 : instructions1 [break;]
  case constante2 : instructions2 [break;]
: :
  default         : instructions
}
```

```
switch ($val) {
  case 0 : traitement(0);
           break;
  case 1 : traitement(1);
           break;
  default  : traitement(-1);
}
```

```
switch ($val) {
  case 0 :
  case 1 : …
       break;
  case 2 : …
       break;
}
```

69

Les boucles

- **Quatre types de boucles**
  - **On connaît à priori le nombre d'itérations**
    - **Avec compteur sur les itérations**
    - **Pour chaque élément d'un ensemble**
  - **On répète un traitement tant qu'une condition est vérifiée**
    - **Au moins une fois**
    - **Pas d'itération si la condition n'est pas initialement vérifiée**

71

Boucle « pour »

- **Opérateur très riche, trois éléments**
  - **Initialisation d'un compteur**
  - **Test de poursuite de l'itération**
  - **Modification de la variable compteur**
- **Si le test est faux au début de la boucle : pas d'itération**

72

Exemple de boucle pour

- **Exemple :**
  - **calculer la somme des n premiers entiers**

```
echo "<table border=1>";
echo "<tr><th>i</th><th>s</th></tr>";
$s=0;
for ($i = 1; $i <= 6; $i++)
{ $s+=$i;
  echo "<tr><td>$i</td><td>$s</td></tr>";
}
echo "</table>";
```

Réalise n itérations

i	s
1	1
2	3
3	6
4	10
5	15
6	21

73

Boucle « pour chaque »

- **Boucle de parcours dédiée au tableau**
  - **Permet de passer en revue tous les éléments d'un tableau pour leur appliquer le même ensemble d'instructions**
  - **Instruction `foreach`**

```
foreach ($table as $cle=>$element) instructions;
foreach ($table as $element) instructions;
```

```
echo "<table border=1>";
echo "<tr><th>Id</th>";
echo "<th>Valeur</th></tr>";
$tab=array('Nom'=>'Dupond','prénom'=>'Paul',
           'Titre'=>'Ma vie','Année'=>2000);
foreach ($tab as $cle=>$elemt) {
  echo "<tr><td>$cle</td><td>$elemt</td></tr>";
}
echo "</table>";
```

Id	Valeur
Nom	Dupond
prénom	Paul
Titre	Ma vie
Année	2000

74

Boucle « tant que… faire »

- **Répéter tant qu'une expression est vraie**
  - **Nombre de fois : 0 à ∞**
  - **Si le test est faux lors de la première itération : pas de traitement**
  - **L'expression d'arrêt doit être modifiée durant les instructions (boucle infinie)**

```
Tant que (expression) faire
  instructions
fait
```

```
while (expression)
  instructions
```

75

Exemple de boucle « tant que… faire »

- **Exemple**
  - **Chercher le premier élément d'une suite supérieur à une valeur**

$$\begin{cases} u_{0=1} \\ u_{1=1} \\ u_n = 2 \times u_{n-1} + u_{n-2} \end{cases}$$

```
echo "<table border=1>";
echo "<tr><th>Un_2</th>";
echo "<th>Un_1</th></tr>";
$Un_1=1;
$Un_2=1;
$max=100;
while ($Un_1<$max) {
  $t = $Un_1;
  $Un_1 = 2*$Un_1 + $Un_2;
  $Un_2 = $t;
  echo "<tr><td>$Un_2</td>";
  echo "<td>$Un_1</td></tr>";
}
echo "</table>";
```

Un_2	Un_1
1	3
3	7
7	17
17	41
41	99
99	239

76

Boucle « répéter…tant que »

- **Répéter tant qu'une expression est vraie**
  - **Nombre de fois : 1 à ∞**
  - **La première itération est réalisée avant le test**
  - **L'expression d'arrêt doit être modifiée durant les instructions (boucle infinie)**

```
Répéter
  instructions
Tant que (expression);
```

```
do
  instructions
while (expression);
```

77

Exemple de boucle « répéter…tant que »

- **Exemple :**
  - **Choisir au hasard, deux nombres différents compris entre 0 et 10**

```
$n1 = Math.random()*10.0;
do {
$n2 = Math.random()*10.0;
} while ($n2==$n1);
print("%d-%d", $n1, $n2);
```

```
$n1 = Math.random()*10.0;
$n2 = Math.random()*10.0;
while ($n2==$n1) {
  $n2 = Math.random()*10.0;
}
print("%d-%d", $n1, $n2);
```

78

Algorithmique pour le multimédia

Fonctions

Benoît Piranda
Équipe SISAR
Université de Marne La Vallée

Déclaration des fonctions utilisateurs

- Où : Partout dans le code php
  - Nom de la fonction précédée du mot clé `function`
- Paramètres
  - D'entrée ou d'entrée/sortie
  - Association d'une valeur par défaut
- Résultat
  - Aucun ou unique
  - Type peut être différent suivant l'exécution

```
function nom_fonction([param1[=defaut1], [param2, […]]])
{ return resultat;
}
```

Exemples de fonction

```
<?php
function htmlTabFromArray2D($tab,$ligneTitre=TRUE)
{ $str='<table border=1>';
  foreach ($tab as $ligne)
  { if ($ligneTitre)
    { $str.='<tr><th>'.implode('</th><th>',$tab[0]).'</th></tr>';
      $ligneTitre=false;
    } else
    { $str.='<tr><td>'.implode('</td><td>',$ligne).'</td></tr>';
    }
  }
  $str.='</table>';
  return $str;
}

$tableau = array(array('Jour','Intervenant')
,array('MARDI','Pierre'),array('JEUDI','Mari
,array('VENDREDI','Pierre'));
echo htmlTabFromArray2D($tableau);
?>
```

Jour	Intervenant
LUNDI	Paul
MARDI	Pierre
JEUDI	Marie
VENDREDI	Pierre

81

Passage de paramètres

- Paramètres d'entrée
  - Permettent d'initialiser les valeurs de la méthode
  - Ne sont pas modifiés à la sortie de la fonction
  - Peuvent être des constantes
- Paramètres de sorties
  - Transmettent un résultat à la fin de la fonctions
  - Sont forcément des modifiables
- Il existe trois types de passage de paramètres
  - Par valeur ou copie (paramètre d'entrée)
  - Par adresse (paramètre d'entrée & sortie) pas en PHP
  - Par référence (paramètre d'entrée & sortie)

82

Passage par valeur ou copie

- Paramètres d'entrée uniquement
  - mode par défaut dans les fonction PHP
  - Les variables peuvent être modifiée dans la méthode mais ne modifient pas le paramètre d'appel

```
<php
function decremente($valeur)
{ $valeur--;
}

$v = 3;
decremente($v);
echo $v;
?>
```

83

Passage par référence

- La variable en paramètre représente la variable d'appel pour le corps de la fonction
  - syntaxe de déclaration : &$var

```
<php
function decremente(&$valeur)
{ $valeur--;
}

$v = 3;
decremente($v);
echo $v;
?>
```

- Remarque :
  - en PHP 4 pas de valeur par défaut pour le passage par référence
  - fonctionne en PHP 5

84

Passage par référence

- **Valeur de retour par référence**
  - **à consommer avec modération**

```
<php
function &decremente(&$valeur)
{ $valeur--;
  return $valeur;
}

$v = 3;
$nouv =& decremente($v);
$nouv--;
echo $v;
?>
```

85

Fonction à plusieurs résultats

- **Une fonction peut retourner un tableau**
- **l'opérateur liste permet d'affecter le résultat à plusieurs variables**

```
<php
function plusieurs()
{ return array('v1','v2');
}

list($x1,$x2) = plusieurs();
?>
```