

Structures de Données

Romain GONÇALVES



epsi

Plan de cours

- I. Rappels d'algorithmie
- II. Structures complexes de données
- III. Algorithmes de tri
- IV. Récursivité
- V. Algorithmes de recherche
- VI. Corrigés

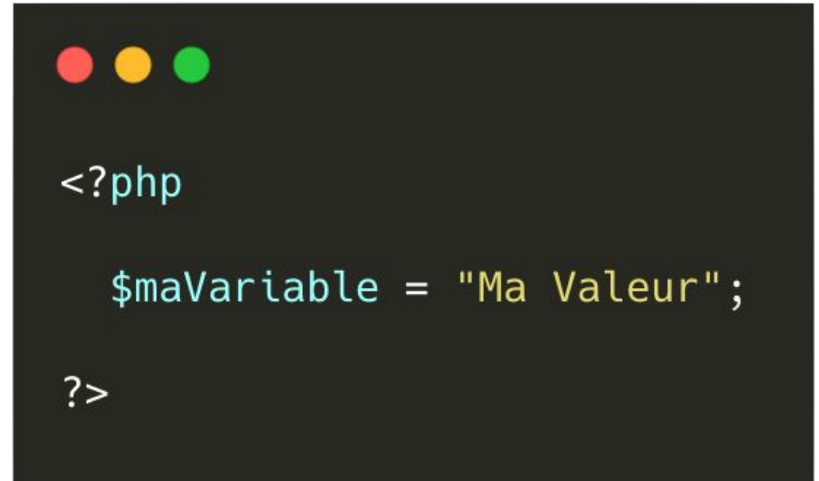


I. Rappels d'algorithmie



Syntaxe du PHP

- Au début de chaque fichier, il est nécessaire d'indiquer le tag d'ouverture de PHP : `<?php`
- La présence de ce tag permet la détection du code PHP par l'éditeur (pour la coloration syntaxique) et par l'interpréteur
- Le tag de fermeture en fin de fichier est facultatif
- Les variables sont préfixées par un `$`
- Toutes les instructions doivent se terminer par un point-virgule



```
<?php  
    $maVariable = "Ma Valeur";  
?>
```

Flux d'entrée/sortie




```
$userInput = trim(fgets(STDIN));  
  
$message = "L'utilisateur a tapé : $userInput" . PHP_EOL;  
fwrite(STDOUT, $message);  
echo $message;  
echo($message); // Identique à la ligne précédente  
  
fwrite(STDERR, "Erreur".PHP_EOL);
```

Constantes



```
define('VITESSE_LUMIERE_VIDE', 299792458);  
const VITESSE_SON_AIR = 340;  
  
echo VITESSE_LUMIERE_VIDE; // 299792458  
echo VITESSE_SON_AIR; // 340  
  
VITESSE_SON_AIR = 0; // Avertissement  
echo VITESSE_SON_AIR; // 340  
  
define('VITESSE_LUMIERE_VIDE', 0); // Avertissement  
echo VITESSE_LUMIERE_VIDE; // 299792458
```

Tests



```
$myAge = 29;
if($myAge < 18){
    echo "Vous êtes trop jeune";
}
else if($myAge > 99){
    echo "Vous êtes trop vieux";
}
else {
    echo "Bienvenue";
}
```



```
$color = "red";
switch($color){
    case "red":
        echo "Roses are red";
        break;
    case "blue":
        echo "The sky is blue";
        break;
    case "green":
        echo "Trees are green";
        break;
    default:
        echo "All the colors of the rainbow";
}
```

Condition ternaire

- La condition ternaire permet d'affecter ou retourner facilement l'une de deux valeurs en fonction d'un booléen (une condition) :



```
$majeur = ($age < 18) ? "Vous être mineur" : "Vous êtes majeur";  
echo $majeur . ($age == 17) ? ", mais plus pour longtemps" : "";
```

Syntaxe : (<CONDITION>) ? <VALEUR SI VRAI> : <VALEUR SI FAUX>

Boucles for et foreach



```
for($i = 0; $i < 5; $i++){  
    echo $i;  
}  
// Affichera "01234"
```



```
foreach(range(0,4) as $value){  
    echo $value;  
}  
// Affichera "01234";
```




```
foreach(["A", "B", "C"] as $key => $value){  
    echo $key . ":" . $value . ";";  
}  
// Affichera "0:A;1:B;2:C;"
```

for(<DEPART?>;<SORTIE?>;<ITERATION?>) { <CODE> }


foreach(<ENSEMBLE> as <VAR_VALEUR>) { <CODE> }

foreach(<ENSEMBLE> as <VAR_CLE> => <VAR_VALEUR>) { <CODE> }

Boucles while et do..while



```
$count = 0;
while($count < 5){
    echo $count;
    $count++;
}
// Affichera "01234"
```



```
$count = 5;
do {
    echo $count;
    $count++;
} while($count < 5);
// Affichera "5";
```

Syntaxe : **while**(<SORTIE?>) { <CODE> }
 do { <CODE> } **while**(<SORTIE?>) **;**

Tableaux

```
● ● ●  
  
$fruits = ['Fraise', 'Banane', 'Orange'];  
$couleurs = [  
    'Fraise' => 'Rouge',  
    'Banane' => 'Jaune',  
    'Orange' => 'Orange',  
];  
  
foreach($fruits as $key => $value){  
    echo $key . ":" . $value . " ";  
}  
// Affichera "0:Fraise;1:Banane;2:Orange;"  
  
foreach($couleurs as $key => $value){  
    echo $key . ":" . $value . " ";  
}  
// Affichera "Fraise:Rouge;Banane:Jaune;Orange:Orange;"
```

Les fonctions

```
function bissextile(int $annee): bool {  
    return ($annee % 4 == 0) && ($annee % 100 != 0 || $annee % 400 == 0);  
}  
echo "Saisir une année : ";  
$annee = intval(trim(fgets(STDIN)));  
  
if(bissextile($annee)){  
    echo "L'année $annee est bissextile" . PHP_EOL;  
} else {  
    echo "L'année $annee n'est pas bissextile" . PHP_EOL;  
}
```

```
function bonjour(string $nom = "Inconnu"){  
    echo "Bonjour $nom !";  
}
```

```
function prochainNoel(){  
    return [25, 12, 2018];  
}  
list($jour, $mois, $annee) = prochainNoel();
```

Les fonctions

En PHP, le scope des variables (leur portée au sein des différents blocs d'instructions) n'est pas perméable ; c'est à dire qu'une variable définie à l'extérieur d'une fonction ne sera pas accessible à l'intérieur de celle-ci, et inversement.



```
$a = "toto";

function testScope() {
    $b = "tata";
    echo $a . PHP_EOL; // Notice: Undefined variable: a
    echo $b . PHP_EOL; // tata
}

testScope();
echo $a . PHP_EOL; // toto
echo $b . PHP_EOL; // Notice: Undefined variable: b
```

Exercice 1 :

Développer une fonction qui retourne la valeur du $n^{\text{ème}}$ terme de la suite de Fibonacci

$$u_0 = 0, u_1 = 1, u_n = u_{n-1} + u_{n-2} \Rightarrow 0, 1, 1, 2, 3, 5, 8, \dots$$

Paramètres de la fonction : “n”, éventuellement “u0” et “u1”

```
php > echo fibonacci(10, 0, 1);  
55  
php > echo fibonacci(100, 0, 1);  
3.5422484817926E+20  
php > echo fibonacci(1000, 0, 1);  
4.3466557686937E+208  
php > echo fibonacci(10000, 0, 1);  
INF
```

Exercice 2 :

On souhaite monter un escalier de N marches, avec des déplacements d'un nombre spécifique de marches (ou plusieurs). Développer une fonction qui indique la première solution trouvée qui permet d'arriver exactement en haut (ou une erreur si pas de solution)

```
php > echo escalier(10, [2]);  
2,2,2,2,2  
php > echo escalier(10, [3]);  
Error  
php > echo escalier(10, [4]);  
Error  
php > echo escalier(10, [3, 4]);  
3,3,4  
php > echo escalier(10, [5]);  
5,5  
php > echo escalier(25, [2, 3, 4, 6]);  
3,4,6,6,6  
php > echo escalier(25, [7, 3]);  
7,3,3,3,3,3,3  
php > echo escalier(17, [2, 5]);  
2,5,5,5  
php > echo escalier(17, [2, 7]);  
2,2,2,2,2,7  
php > echo escalier(17, [5, 9]);  
Error
```

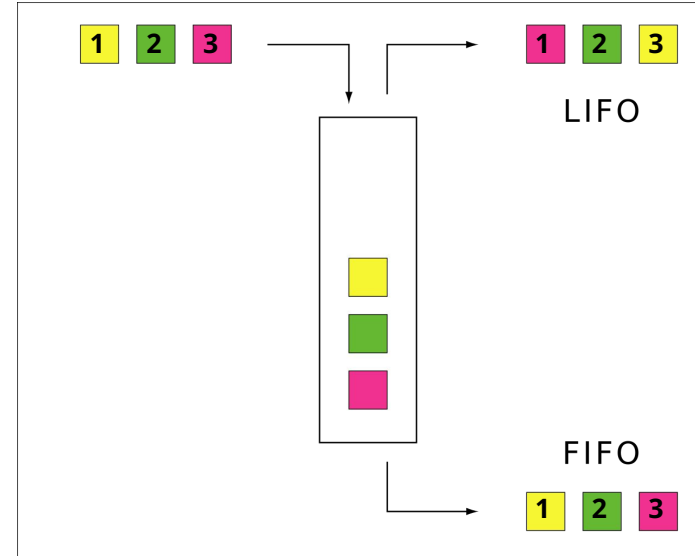


II. Structures complexes de données



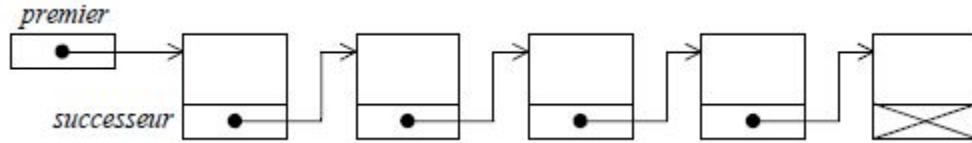
Files et Piles

- Les files (*queues*) et les piles (*stacks*) sont des tableaux avec des opérations limitées à :
 - L'ajout d'élément
 - Le retrait d'élément (suppression)
- La distinction entre les deux se fait uniquement sur l'ordre de traitement :
 - La pile est dite LIFO (**L**ast **I**n **F**irst **O**ut), c'est à dire que l'élément retiré est le plus récent
 - La file est dite FIFO (**F**irst **I**n **F**irst **O**ut), c'est à dire que l'élément retiré est le plus ancien
- Il existe dans certains langages une file "double" (*deque*, prononcé "deck"), dont les éléments peuvent être ajoutés et retirés des deux côtés

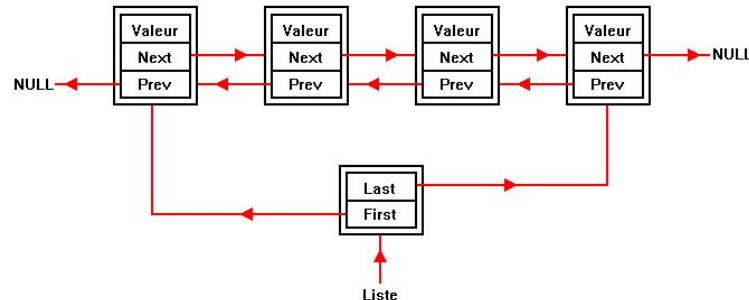


Listes Chaînées

- La liste chaînée est une structure de données où chaque élément dispose d'une fonction "successeur" qui stocke une référence à l'élément suivant.

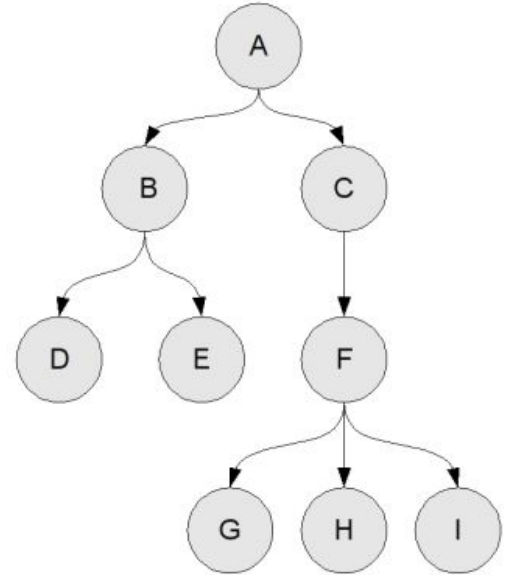


- Il existe également des listes doublement chaînées, dont les éléments disposent de deux fonctions "successeur" et "prédécesseur".



Arbres

- Un arbre est une structure de données où les éléments, nommés “noeuds”, peuvent avoir de zéro à N enfants. On parle de structure non linéaire.
- Deux types de noeuds sont particuliers :
 - Le noeud qui n’a pas de parent (A), que l’on appelle “racine”
 - Les noeuds qui n’ont pas d’enfants (D, E, G, H, I), que l’on appelle “feuille”
- Les arbres possèdent deux caractéristiques :
 - La “taille”, qui est le nombre total de noeuds
 - La “hauteur”, qui est la plus grande distance entre une feuille et la racine
- Un cas particulier courant est l’arbre “binaire” où chaque noeud peut avoir au maximum 2 enfants



IV. Algorithmes de tri

Algorithmes de tri

- **Tri par échange** : Pour chaque élément du tableau, on le compare à tous ses suivants. Lorsque l'ordre n'est pas respecté, on échange les éléments.
- **Tri à bulles** : On compare les éléments adjacents, et on effectue une permutation si le premier est supérieur au deuxième. On répète l'opération jusqu'à ce qu'il n'y ait plus de permutation
- **Tri à bulles amélioré** : On effectue les mêmes opérations, mais en déplaçant les plus grands éléments à la fin. A chaque itération, on réduit le parcours pour ignorer les éléments déjà mis en fin de tableau.
- **Tri par insertion** : On sépare le tableau en deux parties (de taille variable). Dans la première, on considère les données triées, et on vient y insérer chaque élément de la deuxième à la bonne place.

Pour visualiser les algorithmes de tri : <https://visualgo.net/en/sorting> et <https://www.toptal.com/developers/sorting-algorithms>

Exercice 3 :

Ecrire une fonction permettant de générer un tableau de N entiers aléatoires uniques. Appliquer ensuite chacun des 4 algorithmes de tri et mesurer les performances (**microtime**) pour N=10, N=100, N=1000, N=10000. Comparer avec la fonction **sort()**

Effectuer une copie du tableau afin d'avoir le même pour chaque algorithme

Attention : Ne pas utiliser de boucle foreach !

V. Récursivité

La récursivité

- On parle de récursivité lorsqu'une fonction s'appelle elle-même
- Par exemple, le calcul d'une factorielle ($7! = 7*6*5*4*3*2*1$) peut être écrit de la manière suivante :

```
function factorielle($n){  
    if($n == 1){  
        return 1;  
    } else {  
        return $n * factorielle($n - 1);  
    }  
}
```

- Il faut être vigilant à la condition de sortie, au risque de causer une erreur "Maximum call stack size exceeded" (Taille de pile d'appel dépassée => Boucle récursive infinie)

Le passage par référence

- Dans certains cas (dont celui de la récursivité), il peut être utile qu'une fonction PHP soit capable de modifier un de ses paramètres sans avoir à le retourner (car elle retourne déjà autre chose)
- C'est possible en ajoutant un `&` avant le paramètre dans la définition de la fonction
- **Attention de ne pas en abuser !**

```
<?php

function double(&$a, $b){
    $a = $a * 2;
    $b = $b * 2;
}

$nombre1 = 5;
$nombre2 = 3;
double($nombre1, $nombre2);
var_dump($nombre1, $nombre2);
// Affiche 10 et 3
```

```
function generationNombre(&$historique){
    $nombre = random_int(0, getrandmax());
    $historique[] = $nombre;
    return $nombre;
}

$resultats = []; $reference = [];

for($i = 0; $i < 5; $i++){
    $resultats[] = generationNombre($reference);
}

echo json_encode($resultats);
//=> [1647425027,374647373,39971773,1226731448,298571311]
echo json_encode($reference);
//=> [1647425027,374647373,39971773,1226731448,298571311]
```

Exercice 4 :

Développer les fonctions récursives suivantes :

- Quantité de chiffres dans un nombre (utiliser la fonction **intdiv**)
- Calcul de la puissance entière d'un nombre entier (uniquement avec des multiplications)

Exercice 5 (A):

Développer une fonction récursive qui retourne la valeur du $n^{\text{ème}}$ terme de la suite de Fibonacci, puis afficher les 50 premiers termes.

$$u_0 = 0, u_1 = 1, u_n = u_{n-1} + u_{n-2} \Rightarrow 0, 1, 1, 2, 3, 5, 8, \dots$$

Pour interrompre l'exécution : **Control+C**

Exercice 5 (B):

Améliorer la fonction précédente afin qu'elle conserve un historique des calculs effectués.

Afficher les 1500 premiers termes de la suite de Fibonacci

VI. Algorithmes de recherche

Algorithmes de recherche

En supposant un tableau d'entiers **trié**, on peut identifier les deux algorithmes de recherche suivants :

- **Séquentiel** : On parcourt le tableau jusqu'à trouver l'élément recherché
- **Dichotomie** : On divise le tableau en deux. Si la valeur médiane est celle recherchée, on arrête. Si la valeur recherchée est inférieure à la médiane, on recommence l'algorithme avec la première moitié du tableau. Sinon, on recommence l'algorithme avec la deuxième moitié.

On notera que la recherche par dichotomie est un algorithme **récuratif**.

Exercice 6 :

Générer un tableau de N entiers aléatoires uniques. Appliquer ensuite les algorithmes de recherche séquentiel et par dichotomie et mesurer les performances (**microtime**) pour N=10, N=100, N=1000, N=10000. Comparer les performances avec la fonction **array_search()**

Effectuer les recherches pour 5 entiers du tableau et 5 entier en dehors.

Exercice 7 :

```
Saisir le nombre à calculer (< 2000) : 1234
Tirage : 100, 50, 50, 8, 7, 2
100+50=150
150-2=148
148*8=1184
1184+50=1234
```

- Sur le principe du jeu “Des chiffres et des lettres”, développer un algorithme qui trouve un enchaînement d’opérations permettant d’obtenir le nombre cible à partir des 4 opérations élémentaires et d’un ensemble de 6 nombres choisis aléatoirement (doublons possible) parmi : **1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 25, 50, 75, 100**
- Chaque nombre ne peut être utilisé qu’une seule fois au maximum
- **Déroulement du jeu :**
 - On demande à l’utilisateur la saisie d’un nombre inférieur à 2000 (pour limiter les calculs)
 - L’algorithme analyse toutes les possibilités de calcul pour chercher une solution
 - Si une solution est trouvée, on affiche la succession d’opérations trouvée
 - S’il n’y a pas de solutions, on affiche une erreur
- **Indices :** Attention à la division par zéro ; trier les nombres tirés en ordre décroissant