



Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les structures de contrôle](#) ► [Boucles et conditions](#)

Boucles et conditions

PHP nous propose plusieurs structures pour gérer le flot de notre programme :

- des structures conditionnelles :
 - if-else : équivalente à la structure SI-SINON d'AlgoBox
 - switch
- des boucles ou itérations :
 - for : équivalente à la structure POUR d'AlgoBox
 - while : équivalente à la structure TANT_QUE d'AlgoBox
 - do-while

Fin

NAVIGATION



Accueil

■ [Ma page](#)

Pages du site

Mon profil

Cours actuel

[Intro PHP](#)

Participants

Le langage PHP : introduction

Les types et les variables

Les opérateurs

Les structures de contrôle

 **[Boucles et conditions](#)**


 [La condition if](#)

 [L'opérateur ternaire](#)

 [T.P. conditions](#)

 [La structure switch](#)

 [La boucle for](#)

 [T.P. boucles for](#)

 [La boucle while](#)

 [La boucle do-while](#)

 [T.P. boucles while et do-while](#)

 [Break et continue](#)

 [T.P. imbrications](#)

Les structure de données

Les fonctions

Les erreurs

Les fichiers

Les expressions rationnelles

PHP et HTML

Petite application

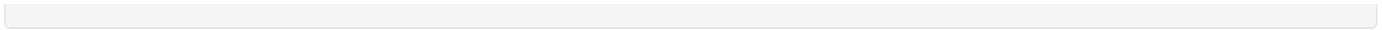
[Mes cours](#)

ADMINISTRATION



Administration du cours

Réglages de mon profil



Connecté sous le nom « [Arnaud Lemais](#) » ([Déconnexion](#))
[Intro PHP](#)

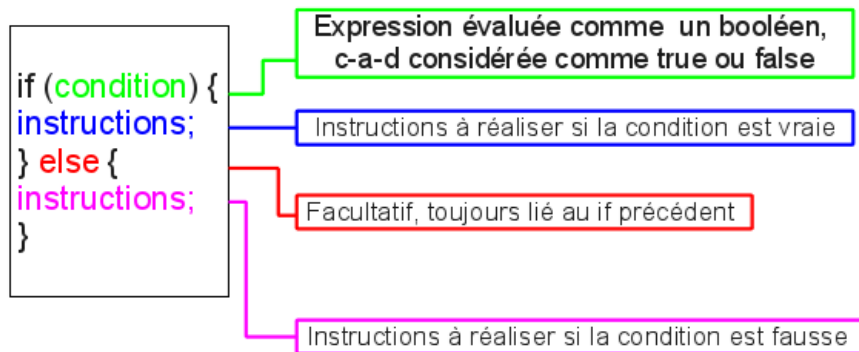


Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les structures de contrôle](#) ► [La condition if](#)

La condition if

La structure conditionnelle `if` de PHP a le même rôle que la structure SI d'AlgoBox : elle permet, si une condition est vraie, d'appliquer un traitement et, si elle est fausse, d'appliquer éventuellement un autre traitement. La syntaxe du `if` est :



Par exemple, un programme qui teste si un nombre est supérieur ou égal à 5 :

```

1  <?php
2
3  // if_1.php
4  $nb = 15;
5  if ($nb >= 5) {
6      echo "$nb est supérieur ou égal à 5";
7  } else {
8      echo "$nb est inférieur à 5";
9  }
10 echo PHP_EOL;
  
```

Cela affichera : 15 est supérieur ou égal à 5.

Le bloc `else` est facultatif :

```

1  <?php
2
3  // if_2.php
4  $nb = 15;
5  if ($nb >= 5) {
6      echo "$nb est supérieur ou égal à 5";
7  }
8  echo PHP_EOL;
  
```

Le bloc `else` est toujours lié au bloc `if` qui le précède. Ce qui peut parfois engendrer des comportements étranges si l'on ne fait pas attention. Illustrons cela par un exemple : je souhaite écrire un programme qui affiche si un nombre est positif, négatif, ou nul. Je pourrais être tenté d'écrire cet algorithme :

```

VARIABLES
nb EST_DU_TYPE NOMBRE
message EST_DU_TYPE CHAINE
DEBUT_ALGORITHME
nb PREND_LA_VALEUR -15
SI (nb > 0) ALORS
    DEBUT_SI
        message PREND LA VALEUR nb + " est positif"
    FIN_SI
FIN_ALGORITHME
  
```

```

AFFICHER message
FIN_SI
SI (nb < 0) ALORS
DEBUT_SI
message PREND_LA_VALEUR nb + " est négatif"
AFFICHER message
FIN_SI
SI (nb == 0) ALORS
DEBUT_SI
message PREND_LA_VALEUR nb + " est nul"
AFFICHER message
FIN_SI
FIN_ALGORITHME

```

Ce qui donnerait en PHP :

```

1 <?php
2
3 // if_3.php
4 $nb = -15;
5 if ($nb > 0) {
6     echo "$nb est positif";
7 }
8 if ($nb < 0) {
9     echo "$nb est négatif";
10 }
11 if ($nb == 0) {
12     echo "$nb est nul";
13 }
14 echo PHP_EOL;

```

Ce qui afficherait : -15 est négatif.

Maintenant si je souhaite utiliser le else, je serais tenté d'écrire cette bêtise (notez que la valeur de nb est 15) :

```

VARIABLES
nb EST_DU_TYPE NOMBRE
message EST_DU_TYPE CHAINE
DEBUT_ALGORITHME
nb PREND_LA_VALEUR 15
SI (nb > 0) ALORS
DEBUT_SI
message PREND_LA_VALEUR nb + " est positif"
AFFICHER message
FIN_SI
SI (nb < 0) ALORS
DEBUT_SI
message PREND_LA_VALEUR nb + " est négatif"
AFFICHER message
FIN_SI
SINON
DEBUT_SINON
message PREND_LA_VALEUR nb + " est nul"
AFFICHER message
FIN_SINON
FIN_ALGORITHME

```

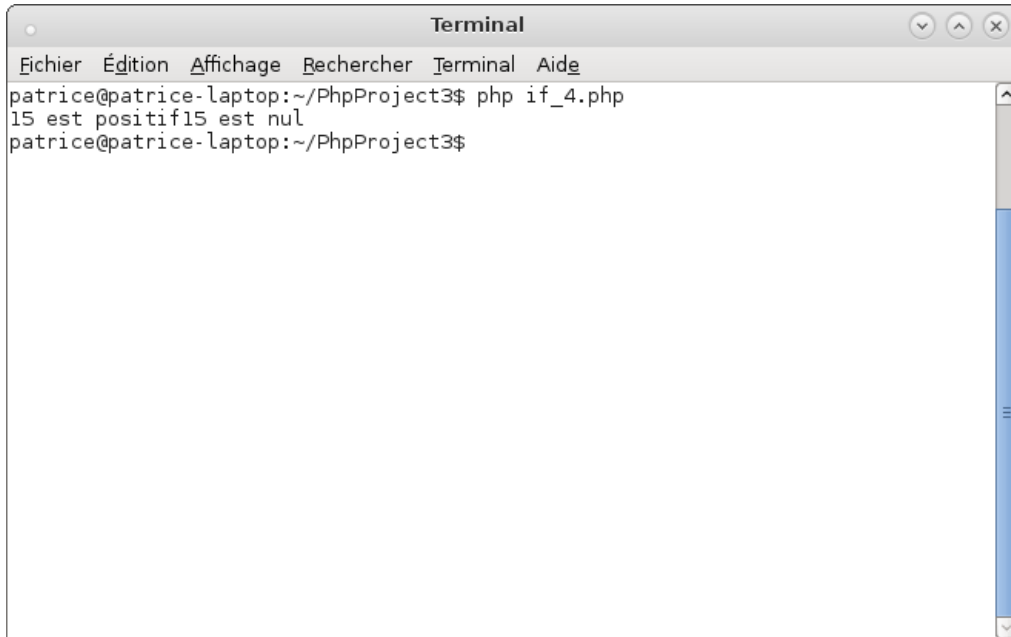
En PHP, cela donne :

```

1 <?php
2
3 // if_4.php
4 $nb = 15;
5 if ($nb > 0) {
6     echo "$nb est positif";
7 }
8 if ($nb < 0) {
9     echo "$nb est négatif";
10 } else {
11     echo "$nb est nul";
12 }
13 echo PHP_EOL;

```

Le résultat est logique mais peut surprendre :



```

Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php if_4.php
15 est positif15 est nul
patrice@patrice-laptop:~/PhpProject3$

```

Si nb vaut 15 alors l'expression `$nb > 0` du if est vraie, donc l'instruction qui se trouve dans le bloc du if (entre les accolades) sera exécutée. Le programme affiche "15 est positif" et continue son exécution. J'arrive au second if dont la condition est `$nb < 0`. Cette expression est fausse, l'instruction du bloc if ne sera pas exécutée. Mais après ce bloc if, il y a un bloc else. Comme nous l'avons dit, le bloc else est lié au bloc if qui le précède. Ce qui signifie que si je ne rentre pas dans le bloc if, je rentre nécessairement dans le bloc else qui le suit. Donc ici l'instruction du bloc else sera exécutée : le programme affiche "15 est nul". Finalement nous obtenons donc l'affichage "15 est positif15 est nul".

Corrigeons ce programme :

```

VARIABLES
nb EST_DU_TYPE NOMBRE
message EST_DU_TYPE CHAINE
DEBUT_ALGORITHME
nb PREND_LA_VALEUR 15
SI (nb > 0) ALORS
    DEBUT_SI
    message PREND_LA_VALEUR nb + " est positif"
    AFFICHER message
    FIN_SI
SINON
    DEBUT_SINON
    SI (nb < 0) ALORS
        DEBUT_SI
        message PREND_LA_VALEUR nb + " est négatif"
        AFFICHER message
        FIN_SI
    SINON
        DEBUT_SINON
        message PREND_LA_VALEUR nb + " est nul"
        AFFICHER message
        FIN_SINON
    FIN_SINON
FIN_ALGORITHME

```

Ce qui donne en PHP :

```

1 <?php
2
3 // if_5.php
4 $nb = 15;
5 if ($nb > 0) {
6     echo "$nb est positif";
7 } else {
8     if ($nb < 0) {
9         echo "$nb est négatif";
10    } else {
11        echo "$nb est nul";

```

```

12     }
13 }
14 echo PHP_EOL;

```

Ici nous avons imbriqué un if-else dans un autre if-else.

PHP nous propose une structure `elseif` afin d'éviter certaines imbrications. Le `elseif` permet des structures du genre :

```

SI nb > 0 ALORS
    AFFICHER nb + " est positif"
SINON SI nb < 0 ALORS
    AFFICHER nb + " est négatif"
SINON
    AFFICHER nb + " est nul"
FINSI

```

Le `else` sera lié au `elseif` qui le précède :

```

1 <?php
2
3 // if_6.php
4 $nb = 15;
5 if ($nb > 0) {
6     echo "$nb est positif";
7 } elseif ($nb < 0) {
8     echo "$nb est négatif";
9 } else {
10    echo "$nb est nul";
11 }
12 echo PHP_EOL;

```

Dans un `if`, il est possible dans la condition d'invertir l'opérateur de gauche et l'opérateur de droite. Par exemple si vous souhaitez tester un égalité vous pouvez écrire cela :

```

1 <?php
2
3 // if_7.php
4 $nb = 10;
5 if ($nb == 10) {
6     echo 'OK';
7 }
8 echo PHP_EOL;

```

ou encore :

```

1 <?php
2
3 // if_8.php
4 $nb = 10;
5 if (10 == $nb) {
6     echo 'OK';
7 }
8 echo PHP_EOL;

```

Cette inversion permet d'éviter les erreurs sémantiques dues à l'opérateur `=`. Que se passerait-il si, par étourderie, nous oublions un égal à l'opérateur d'égalité ? Celui-ci se transformerait en opérateur d'assignation. L'expression `$nb == 10` deviendrait `$nb = 10`. L'expression n'aurait donc plus le même sens, d'où l'erreur sémantique. L'expression `$nb == 10` est une expression booléenne qui vérifie l'égalité de valeur entre la variable `nb` et 10. Par contre `$nb = 10` est une assignation de valeur, et cette expression renvoie un résultat : la valeur assignée, ici 10. Les expressions `$nb == 10` et `$nb = 10` n'ont pas le même sens, mais elles renvoient bien toutes les deux une valeur : `true` ou `false` pour la première, 10 pour la seconde. En PHP, un nombre entier non

nul est considéré comme true. Donc dans le if la condition `$nb = 10` sera considérée comme vraie puisqu'elle renvoie un entier non nul, et non entrons dans le if... Voici un exemple de ce comportement :

```
1 <?php
2
3 // if_9.php
4 $nb = 5;
5 if ($nb = 10) {
6     echo 'OK';
7 }
8 echo PHP_EOL;
```

Dans le if nous avons une assignation et non une comparaison... Si nous inversons les deux opérateurs :

```
1 <?php
2
3 // if_10.php
4 $nb = 10;
5 if (10 = $nb) {
6     echo 'OK';
7 }
8 echo PHP_EOL;
```

L'éditeur nous signale une erreur de syntaxe à la ligne 5, nous devons corriger en :

```
1 <?php
2
3 // if_11.php
4 $nb = 10;
5 if (10 == $nb) {
6     echo 'OK';
7 }
8 echo PHP_EOL;
```

L'inversion des opérateurs évite donc l'erreur sémantique.

Fin

NAVIGATION

Accueil

■ Ma page

Pages du site

Mon profil

Cours actuel

Intro PHP

Participants

Le langage PHP : introduction

Les types et les variables

Les opérateurs

Les structures de contrôle

📁 Boucles et conditions

📁 **La condition if**

📁 L'opérateur ternaire

📁 T.P. conditions

📁 La structure switch

📁 La boucle for

📁 T.P. boucles for

📁 La boucle while

📁 La boucle do-while

📁 T.P. boucles while et do-while

 [Break et continue](#) [T.P. imbrications](#)[Les structure de données](#)[Les fonctions](#)[Les erreurs](#)[Les fichiers](#)[Les expressions rationnelles](#)[PHP et HTML](#)[Petite application](#)[Mes cours](#)

ADMINISTRATION

[Administration du cours](#)[Réglages de mon profil](#)

Connecté sous le nom « [Arnaud Lemais](#) » ([Déconnexion](#))
[Intro PHP](#)

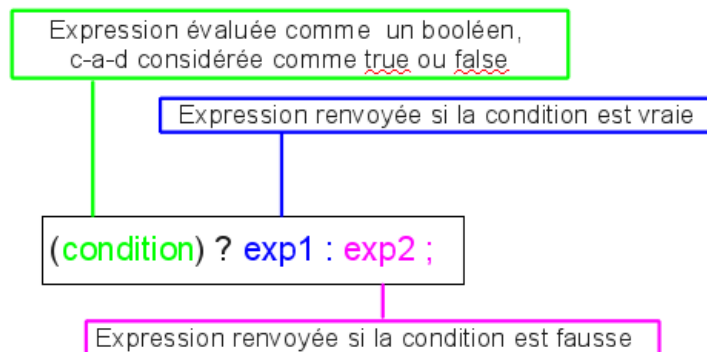


Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les structures de contrôle](#) ► [L'opérateur ternaire](#)

L'opérateur ternaire

Bien qu'étant un opérateur, je place l'opérateur ternaire dans les structures de contrôle car, bien souvent, cet opérateur sert à faire des assignments conditionnelles qui pourraient être effectuées par une structure if/else. La syntaxe de l'opérateur ternaire est :



Exemple :

```

1 <?php
2
3 // operateur_ternaire_1.php
4 $nb = 5;
5 echo ($nb == 5) ? 'yes' : 'no';
6 echo PHP_EOL;

```

Autre exemple avec une assignment :

```

1 <?php
2
3 // operateur_ternaire_2.php
4 define('MAJORITE', 18);
5 $age = 48;
6 $message = "";
7 if ($age >= MAJORITE) {
8     $message = 'majeur';
9 } else {
10    $message = 'mineur';
11 }
12 echo $message . PHP_EOL;

```

devient avec un opérateur conditionnel :

```

1 <?php
2
3 // operateur_ternaire_2.php
4 define('MAJORITE', 18);
5 $age = 48;
6 $message = ($age >= MAJORITE) ? 'majeur' : 'mineur';
7 echo $message . PHP_EOL;

```

[Fin](#)

NAVIGATION




Accueil

■ [Ma page](#)

[Pages du site](#)[Mon profil](#)[Cours actuel](#)

[Intro PHP](#)

[Participants](#)[Le langage PHP : introduction](#)[Les types et les variables](#)[Les opérateurs](#)[Les structures de contrôle](#) [Boucles et conditions](#) [La condition if](#) [L'opérateur ternaire](#) [T.P. conditions](#) [La structure switch](#) [La boucle for](#) [T.P. boucles for](#) [La boucle while](#) [La boucle do-while](#) [T.P. boucles while et do-while](#) [Break et continue](#) [T.P. imbrications](#)[Les structure de données](#)[Les fonctions](#)[Les erreurs](#)[Les fichiers](#)[Les expressions rationnelles](#)[PHP et HTML](#)[Petite application](#)

[Mes cours](#)

ADMINISTRATION

[Administration du cours](#)[Réglages de mon profil](#)

Connecté sous le nom « [Arnaud Lemais](#) » ([Déconnexion](#))
[Intro PHP](#)



Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les structures de contrôle](#) ► [T.P. conditions](#)

T.P. conditions

Implémentez en PHP les algorithmes qui permettront de tester si un nombre nb est

1. égal à 5
2. entre 5 inclus et 10 inclus
3. inférieur ou égal à 5
4. multiple de 3
5. impair
6. d'une part plus petit que 5 ou plus grand que 10, et d'autre part impair
7. entre 5 inclus et 10 inclus et impair

Vous afficherez un message si la condition est réalisée et un autre si la condition ne l'est pas.

Le nom des fichiers sera de la forme "if_num_nom_prenom.php", en minuscule et sans accent. Vous remettrez une archive zip contenant ces fichiers, le nom de l'archive sera de la forme "if_nom_prenom.zip".


État du travail remis

| | |
|--------------------------|---------------------------|
| Numéro de tentative | Ceci est la tentative 1. |
| Statut des travaux remis | Remis pour évaluation |
| Statut de l'évaluation | Noté |
| Dernière modification | jeudi 26 mars 2015, 08:56 |
| Remises de fichiers | |

Modifier le travail

Modifier votre travail remis

Feedback

| | |
|---------------------------|---|
| Note | 100,00 / 100,00 |
| Évalué le | jeudi 26 mars 2015, 09:27 |
| Évalué par |  Patrice Fernandez |
| Feedback par commentaires | C'est bon. Dans le 7, les parenthèses autour de <code>\$nb >= 5 && nb <= 10</code> sont inutiles. |

NAVIGATION

[Accueil](#)

■ [Ma page](#)

[Pages du site](#)



Mon profil

Cours actuel

[Intro PHP](#)

Participants

Le langage PHP : introduction

Les types et les variables

Les opérateurs

Les structures de contrôle

 [Boucles et conditions](#)


 [La condition if](#)

 [L'opérateur ternaire](#)

 **[T.P. conditions](#)**

 [La structure switch](#)

 [La boucle for](#)

 [T.P. boucles for](#)

 [La boucle while](#)

 [La boucle do-while](#)

 [T.P. boucles while et do-while](#)

 [Break et continue](#)

 [T.P. imbrications](#)

Les structure de données

Les fonctions

Les erreurs

Les fichiers

Les expressions rationnelles

PHP et HTML

Petite application

[Mes cours](#)

ADMINISTRATION



Administration du cours

Réglages de mon profil

Connecté sous le nom « [Arnaud Lemais](#) » ([Déconnexion](#))
[Intro PHP](#)



Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les structures de contrôle](#) ► [La structure switch](#)

La structure switch

La structure switch est comparable à une suite if/elseif/else, et va permettre d'effectuer un traitement au cas par cas. La structure switch va vérifier l'égalité de deux expressions, et ainsi déterminer le ou les cas à exécuter.

Par exemple si nous souhaitons effectuer un affichage si la valeur d'un entier nb est égale à 10, 15, 20 ou une autre valeur, nous écrivons avec un if/elseif/else :

Si nb est égale à 10 alors j'affiche "égale à 10"

Sinon si nb est égale à 15 alors j'affiche "égale à 15"

Sinon si nb est égale à 20 alors j'affiche "égale à 20"

Sinon j'affiche "nb a une autre valeur".

Ce qui donnerait en PHP :

```
1 <?php
2
3 // switch_1.php
4 $nb = 10;
5 if ($nb == 10) {
6     echo 'égale à 10';
7 } elseif ($nb == 15) {
8     echo 'égale à 15';
9 } elseif ($nb == 20) {
10    echo 'égale à 20';
11 } else {
12    echo 'nb a une autre valeur';
13 }
14 echo PHP_EOL;
```

Pour effectuer le même traitement nous pourrions écrire avec un switch :

la valeur à observer est nb :

- dans le cas où nb est égale à 10, j'affiche "égale à 10" et je sors de la condition
- dans le cas où nb est égale à 15, j'affiche "égale à 15" et je sors de la condition
- dans le cas où nb est égale à 20, j'affiche "égale à 20" et je sors de la condition
- dans le cas où la valeur de nb n'est ni 10, ni 15, ni 20 (cas par défaut), j'affiche "nb a une autre valeur" et je sors de la condition

Ce qui donnerait en PHP :

```
1 <?php
2
3 // switch_2.php
4 $nb = 10;
5 switch ($nb) {
6     case 10:
7         echo 'égale à 10';
8         break;
9     case 15:
10        echo 'égale à 15';
11        break;
12     case 20:
13        echo 'égale à 20';
14        break;
15     default:
16        echo 'nb a une autre valeur';
```

```
17         break;
18     }
19     echo PHP_EOL;
```

Dans chaque cas l'instruction break sert à "casser" la structure conditionnelle. Si l'on oublie le break, une fois que la structure entre dans un cas, elle exécute les cas suivants jusqu'à trouver un break.

Dans un switch, le cas par défaut est facultatif et il est tout à fait possible de "lier" plusieurs cas :

```
1 <?php
2
3 // switch_3.php
4 $nb = 15;
5 switch ($nb) {
6     case 10:
7     case 15:
8         echo '$nb est égale à 10 ou 15';
9         break;
10    case 20:
11        echo '$nb est égale à 20';
12        break;
13 }
14 echo PHP_EOL;
```

La structure switch vérifie l'égalité de l'expression entre parenthèses et de l'expression d'un cas en utilisant l'opérateur ==. Le switch ne vérifie donc pas l'égalité stricte.

Les expressions utilisées dans un switch peuvent être de n'importe quel type sauf de type Callable.

Autre exemple de switch :

```
1 <?php
2
3 // switch_4.php
4 $vin = 'Bordeaux';
5 switch ($vin) {
6     case 'Bourgogne':
7         echo 'est';
8         break;
9     case 'Bordeaux':
10        echo 'ouest';
11        break;
12    case 'Minervois':
13        echo 'sud';
14        break;
15    default:
16        echo 'ailleurs';
17        break;
18 }
19 echo PHP_EOL;
```

Fin

NAVIGATION



[Accueil](#)

■ [Ma page](#)

[Pages du site](#)

[Mon profil](#)

[Cours actuel](#)

[Intro PHP](#)

[Participants](#)

[Le langage PHP : introduction](#)

[Les types et les variables](#)

Les opérateurs

Les structures de contrôle

 [Boucles et conditions](#)


 [La condition if](#)

 [L'opérateur ternaire](#)

 [T.P. conditions](#)

 **[La structure switch](#)**

 [La boucle for](#)

 [T.P. boucles for](#)

 [La boucle while](#)

 [La boucle do-while](#)

 [T.P. boucles while et do-while](#)

 [Break et continue](#)

 [T.P. imbrications](#)

Les structure de données

Les fonctions

Les erreurs

Les fichiers

Les expressions rationnelles

PHP et HTML

Petite application

[Mes cours](#)

ADMINISTRATION



Administration du cours

Réglages de mon profil

Connecté sous le nom « [Arnaud Lemais](#) » ([Déconnexion](#))

[Intro PHP](#)

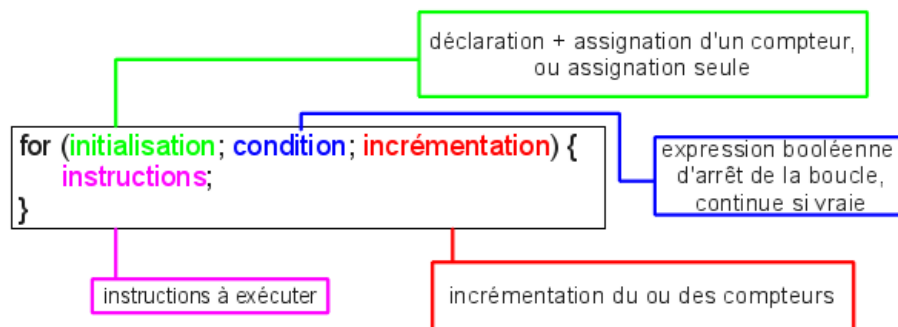


Introduction au langage de programmation PHP

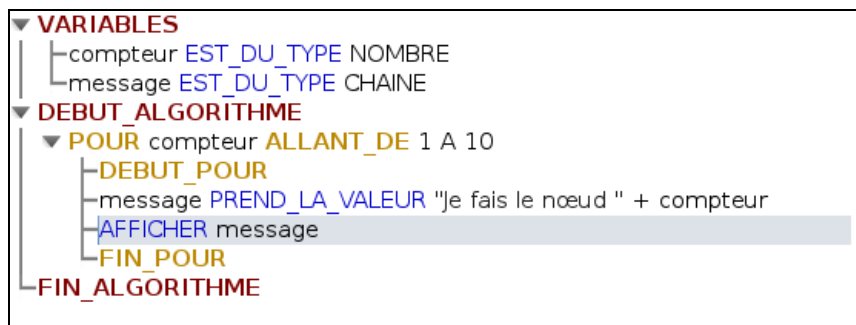
[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les structures de contrôle](#) ► [La boucle for](#)

La boucle for

La boucle `for` est équivalente à la structure POUR d'AlgoBox : elle permet la répétition d'instructions en utilisant un compteur. PHP va donner un peu de souplesse à cette structure et nous permettre, dans une boucle `for`, d'utiliser plusieurs compteurs ainsi que d'incrémenter ou de décrémenter à notre guise ces compteurs. Les déclarations des compteurs pourront se faire directement dans la boucle, ou à l'extérieur comme nous l'avons fait avec AlgoBox. Voici la syntaxe de la boucle `for` :



Nous allons implémenter en PHP cet algorithme :



Ce qui nous donne :

```

1 <?php
2
3 // for_1.php
4 for ($i = 1; $i <= 10; $i++) {
5     echo "je fais le noeud $i" . PHP_EOL;
6 }
  
```

Ici le compteur est incrémenté de 1 à chaque tour de boucle : vous avez noté l'utilisation de l'opérateur d'incrémentation (`++`). Nous pourrions tout aussi bien incrémenter le compteur d'une autre valeur, et même le décrémenter. Par exemple si je souhaite afficher les valeurs de 10 à 0 inclus en faisant "reculer" le compteur de 2 pas à chaque tour de boucle :

```

1 <?php
2
3 // for_2.php
4 for ($i = 10; $i >= 0; $i -=2) {
5     echo $i . PHP_EOL;
6 }
  
```


Cette fois nous avons utiliser l'opérateur d'assignation `-=`. À chaque tour de boucle `$i` prend la valeur `$i - 2`.

Nous pourrions aussi utiliser plusieurs compteurs, et incrémenter/décrémenter ces compteurs. Si je souhaite afficher les 10 premières additions dont la somme est 10, je peux écrire :

```
1 <?php
2
3 // for_3.php
4 for ($i = 0, $j = 10; $i < 10; $i++, $j--) {
5     $somme = $i + $j;
6     echo "$i + $j = $somme" . PHP_EOL;
7 }
```

Notez que dans l'initialisation et l'incrémentation, les instructions sont séparées par des virgules.

Dans une boucle `for`, l'initialisation, la condition ainsi que l'incrémentation sont facultatives. Seuls les points-virgules sont obligatoires. Je pourrais réécrire l'exemple précédent en déclarant mes compteurs à l'extérieur de la boucle, et incrémenter et décrémenter mes compteurs dans le "corps" de la boucle. Il ne me resterait plus entre parenthèses que la condition et les points-virgules :

```
1 <?php
2
3 // for_4.php
4 $i = 0;
5 $j = 10;
6 for (; $i < 10;) {
7     $somme = $i + $j;
8     echo "$i + $j = $somme" . PHP_EOL;
9     $i++;
10    $j--;
11 }
```

Cela ne ressemble plus vraiment à une boucle `for`...

Fin

NAVIGATION



Accueil

■ Ma page

Pages du site

Mon profil

Cours actuel

Intro PHP

Participants

Le langage PHP : introduction

Les types et les variables

Les opérateurs

Les structures de contrôle

Boucles et conditions

La condition if

L'opérateur ternaire

T.P. conditions

La structure switch

La boucle for

T.P. boucles for

La boucle while

La boucle do-while

T.P. boucles while et do-while

Break et continue

T.P. imbrications

Les structure de données

Les fonctions
Les erreurs
Les fichiers
Les expressions rationnelles
PHP et HTML
Petite application

[Mes cours](#)

ADMINISTRATION



Administration du cours

Réglages de mon profil

Connecté sous le nom « [Arnaud Lemais](#) » ([Déconnexion](#))
[Intro PHP](#)



Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les structures de contrôle](#) ► [T.P. boucles for](#)

T.P. boucles for

Implémentez en PHP les algorithmes qui permettront :

1. d'afficher les nombres de 1 à 50
2. d'afficher les nombres de 1 à n, n étant saisi par l'utilisateur
3. d'afficher les 10 premiers nombres de la table de multiplication de 3
4. d'afficher les 10 premiers nombres d'une table de multiplication choisie par l'utilisateur

Le nom de chaque fichier sera de la forme "for_num_nom_prenom.php". Vous remettrez une archive zip contenant ces fichiers, le nom de l'archive sera de la forme "for_num_prenom.zip".

Pour récupérer une saisie de l'utilisateur vous utiliserez la fonction fgets() sur STDIN et trim(), voici un exemple :

```
1 <?php
2
3 $nb = trim(fgets(STDIN));
4 echo "Vous avez saisi le nombre $nb.";
```

Lors de l'exécution de ce script PHP attend une saisie en console, puis affiche cette saisie après validation (touche Entrée).

État du travail remis

| | |
|--------------------------|---------------------------|
| Numéro de tentative | Ceci est la tentative 1. |
| Statut des travaux remis | Remis pour évaluation |
| Statut de l'évaluation | Noté |
| Dernière modification | jeudi 26 mars 2015, 10:14 |
| Remises de fichiers | |

Modifier le travail

Modifier votre travail remis

Feedback

| | |
|---------------------------|---|
| Note | 100,00 / 100,00 |
| Évalué le | jeudi 26 mars 2015, 10:26 |
| Évalué par |  Patrice Fernandez |
| Feedback par commentaires | C'est bon. |

NAVIGATION

[Accueil](#)■ [Ma page](#)[Pages du site](#)[Mon profil](#)[Cours actuel](#)[Intro PHP](#)[Participants](#)[Le langage PHP : introduction](#)[Les types et les variables](#)[Les opérateurs](#)[Les structures de contrôle](#) [Boucles et conditions](#) [La condition if](#) [L'opérateur ternaire](#) [T.P. conditions](#) [La structure switch](#) [La boucle for](#) [T.P. boucles for](#) [La boucle while](#) [La boucle do-while](#) [T.P. boucles while et do-while](#) [Break et continue](#) [T.P. imbrications](#)[Les structure de données](#)[Les fonctions](#)[Les erreurs](#)[Les fichiers](#)[Les expressions rationnelles](#)[PHP et HTML](#)[Petite application](#)[Mes cours](#)

ADMINISTRATION

[Administration du cours](#)[Réglages de mon profil](#)

Connecté sous le nom « [Arnaud Lemais](#) » ([Déconnexion](#))
[Intro PHP](#)

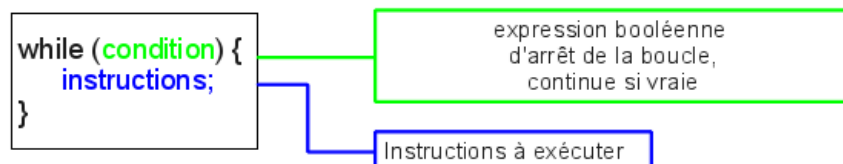


Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► Les structures de contrôle ► [La boucle while](#)

La boucle while

La boucle **while** est équivalente à la structure TANT_QUE d'AlgoBox : tant que la condition est vraie, la boucle continue. La boucle while a pour syntaxe :



Reprenons le dernier script du chapitre précédent :

```

1  <?php
2
3  // for_4.php
4  $i = 0;
5  $j = 10;
6  for (; $i < 10;) {
7      $somme = $i + $j;
8      echo "$i + $j = $somme" . PHP_EOL;
9      $i++;
10     $j--;
11 }

```

Dans ce script nous trouvons une boucle for qui a été dépouillée des parties d'initialisation et d'incrémentation des compteurs. Il ne reste plus dans ce for que la condition d'arrêt. Nous pourrions donc transformer cette boucle for en boucle while en la traduisant par :



Ce qui donnerait en PHP :

```

1  <?php
2
3  // while_1.php
4  $i = 0;
5  $j = 10;
6  while ($i < 10) {

```

```

7     $somme = $i + $j;
8     echo "$i + $j = $somme" . PHP_EOL;
9     $i++;
10    $j--;
11 }

```

Nous allons implémenter l'algorithme suivant :



Pour réaliser cette implémentation nous allons devoir récupérer une saisie clavier de l'utilisateur. Voici un exemple où je récupère une saisie clavier que j'affecte à la variable message avant d'afficher la valeur de message.

```

1 <?php
2
3 // while_2.php
4 $message = trim(fgets(STDIN));
5 echo $message;

```

Lorsque j'exécute ce script, PHP attends une saisie en console, je saisis une chaîne de caractères puis je valide, et PHP affiche ma saisie. Je peux donc maintenant implémenter mon algorithme :

```

1 <?php
2
3 // while_3.php
4 $message = '';
5 while ($message != 'stop') {
6     $message = trim(fgets(STDIN));
7     echo "Vous avez saisi : '$message'" . PHP_EOL;
8 }
9 echo 'Fin !' . PHP_EOL;

```

Fin

NAVIGATION

[Accueil](#)

■ [Ma page](#)

[Pages du site](#)

[Mon profil](#)

[Cours actuel](#)

[Intro PHP](#)

[Participants](#)

[Le langage PHP : introduction](#)











[Les types et les variables](#)

[Les opérateurs](#)

[Les structures de contrôle](#)

[Boucles et conditions](#)

[La condition if](#)

-  L'opérateur ternaire
-  T.P. conditions
-  La structure switch
-  La boucle for
-  T.P. boucles for
-  **La boucle while**
-  La boucle do-while
-  T.P. boucles while et do-while
-  Break et continue
-  T.P. imbrications

Les structure de données

Les fonctions

Les erreurs

Les fichiers

Les expressions rationnelles

PHP et HTML

Petite application

[Mes cours](#)

ADMINISTRATION



Administration du cours

Réglages de mon profil

Connecté sous le nom « [Arnaud Lemais](#) » ([Déconnexion](#))
[Intro PHP](#)



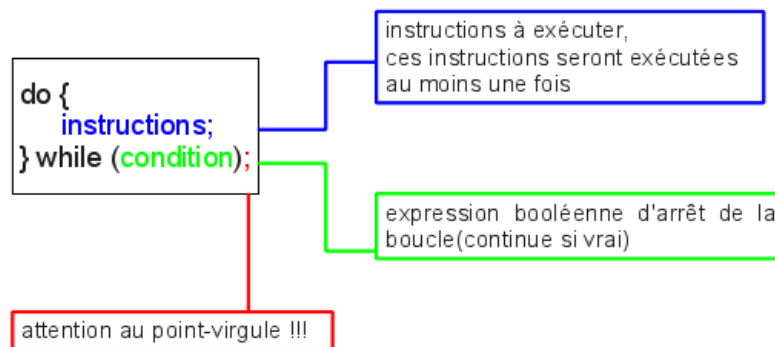
Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les structures de contrôle](#) ► [La boucle do-while](#)

La boucle do-while

La boucle **do-while** est très proche dans son principe de **la boucle while** : répéter des instructions tant que la condition est vraie. Pour le do-while nous dirions plutôt faire quelque chose tant que quelque chose est vraie. La boucle do-while implique qu'une action est exécutée avant le test de la condition. D'abord je fais, ensuite je regarde si je peux continuer, c'est-à-dire si la condition est vraie. Prenons un exemple trivial : tant que je gagne je joue. Dis comme cela nous pourrions écrire l'algorithme suivant : tant que gagné est vrai alors jouer. Mais le problème c'est que pour gagner, il faut d'abord jouer. Nous devrions donc dire : je joue tant que je gagne. D'abord j'exécute l'action de jouer, puis, si j'ai gagné, alors je recommence.

Voici la syntaxe du do-while :



Si nous implémentons l'algorithme de la page précédente avec un do-while nous obtenons :

```
1 <?php
2
3 // do_while_1.php
4 $message = '';
5 do {
6     $message = trim(fgets(STDIN));
7     echo "Vous avez saisi : '$message'" . PHP_EOL;
8 } while ($message != 'stop');
9 echo 'Fin !' . PHP_EOL;
```

Les instructions du bloc do seront toujours exécutées au moins une fois, même si la condition est fausse.

```
1 <?php
2
3 // do_while_2.php
4 $nb = 100;
5 do {
6     echo "\$nb vaut $nb" . PHP_EOL;
7     $nb++;
8 } while ($nb < 10);
9 echo 'C\'est fini!' . PHP_EOL;
```

Cela affichera :


```
C'est fini!  
patrice@patrice-laptop:~/PhpProject3$
```

Fin

NAVIGATION



Accueil

■ Ma page

Pages du site

Mon profil

Cours actuel

Intro PHP

Participants

Le langage PHP : introduction

Les types et les variables

Les opérateurs

Les structures de contrôle

[Boucles et conditions](#)

[La condition if](#)

[L'opérateur ternaire](#)

[T.P. conditions](#)

[La structure switch](#)

[La boucle for](#)

[T.P. boucles for](#)

[La boucle while](#)

[La boucle do-while](#)

[T.P. boucles while et do-while](#)

[Break et continue](#)

[T.P. imbrications](#)

Les structure de données

Les fonctions

Les erreurs

Les fichiers

Les expressions rationnelles

PHP et HTML

Petite application

Mes cours

ADMINISTRATION



Administration du cours

Réglages de mon profil

Connecté sous le nom « [Arnaud Lemais](#) » ([Déconnexion](#))
[Intro PHP](#)



Introduction au langage de programmation PHP

[Accueil](#) ▶ [Mes cours](#) ▶ [Développement logiciel](#) ▶ [Intro PHP](#) ▶ [Les structures de contrôle](#) ▶ [T.P. boucles while et do-while](#)

T.P. boucles while et do-while

Implémentez en PHP l'algorithme qui permettra de faire deviner un nombre à l'utilisateur. Le nombre à deviner sera un nombre entre 1 et 10 inclus généré aléatoirement par PHP. L'utilisateur devra saisir une proposition de nombre tant qu'il n'aura pas trouvé le nombre aléatoire.

Pour affecter à une variable un nombre aléatoire entre 1 inclus et 10 inclus, il faut affecter à la variable le "resultat" de la fonction `mt_rand(1, 10)` :

Ce qui donne :

```
1 <?php
2
3 $resultat = mt_rand(1, 10);
```

Le nom du script sera "tant_que_1_nom_prenom.php".

Complétez l'algorithme pour que la programme affiche si le nombre saisi par l'utilisateur est plus petit ou plus grand que le nombre aléatoire. Le nom script sera "tant_que_2_nom_prenom.php".

Complétez votre algorithme pour que l'utilisateur est 10 chances, une fois ces 10 chances épuisées le jeu s'arrête. Si l'utilisateur a trouvé le nombre aléatoire avant épuisement de ses chances le jeu s'arrête. Afficher le nombre de chances restantes. Le nom du script sera "tant_que_3_nom_prenom.php".

Complétez votre algorithme pour qu'à la fin d'une partie le programme propose à l'utilisateur de rejouer : "Voulez-vous rejouer ? (oui/non)". Le nom du script sera "tant_que_4_nom_prenom.php".

Vous remettrez une archive zip contenant ces fichiers, le nom de l'archive sera de la forme "tant_que_nom_prenom.zip".

État du travail remis

| | |
|--------------------------|---------------------------|
| Numéro de tentative | Ceci est la tentative 1. |
| Statut des travaux remis | Remis pour évaluation |
| Statut de l'évaluation | Noté |
| Dernière modification | jeudi 26 mars 2015, 12:01 |
| Remises de fichiers | |

Modifier le travail

Modifier votre travail remis

Feedback

| | |
|------|-----------------|
| Note | 100,00 / 100,00 |
|------|-----------------|

Évalué le

jeudi 26 mars 2015, 12:26

Évalué par



Patrice Fernandez

Feedback par commentaires

C'est bon.

NAVIGATION




Accueil

■ Ma page

[Pages du site](#)[Mon profil](#)[Cours actuel](#)

[Intro PHP](#)

[Participants](#)[Le langage PHP : introduction](#)[Les types et les variables](#)[Les opérateurs](#)[Les structures de contrôle](#) [Boucles et conditions](#) [La condition if](#) [L'opérateur ternaire](#) [T.P. conditions](#) [La structure switch](#) [La boucle for](#) [T.P. boucles for](#) [La boucle while](#) [La boucle do-while](#) [T.P. boucles while et do-while](#) [Break et continue](#) [T.P. imbrications](#)[Les structure de données](#)[Les fonctions](#)[Les erreurs](#)[Les fichiers](#)[Les expressions rationnelles](#)[PHP et HTML](#)[Petite application](#)

[Mes cours](#)

ADMINISTRATION

[Administration du cours](#)[Réglages de mon profil](#)Connecté sous le nom « [Arnaud Lemais](#) » ([Déconnexion](#))[Intro PHP](#)



Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les structures de contrôle](#) ► [Break et continue](#)

Break et continue

L'instruction [break](#), que nous avons déjà rencontrée dans le switch, permet de "casser" une boucle, c'est-à-dire d'y mettre fin. Par exemple la boucle suivante affiche les nombres de 0 à 5 inclus puis "c'est fini":

```
1 <?php
2
3 // break_continue_1.php
4 for ($i = 0; $i < 6; $i++) {
5     echo $i . PHP_EOL;
6 }
7 echo 'c\'est fini' . PHP_EOL;
```

Si je décide de "casser" ma boucle quand \$i vaut 4 avant l'affichage alors j'écrirai :

```
1 <?php
2
3 // break_continue_2.php
4 for ($i = 0; $i < 6; $i++) {
5     if ($i == 4) {
6         break;
7     }
8     echo $i . PHP_EOL;
9 }
10 echo 'c\'est fini' . PHP_EOL;
```

Lorsque i vaut 4, la boucle est cassée par l'instruction break et le script continue à l'instruction qui suit la boucle.

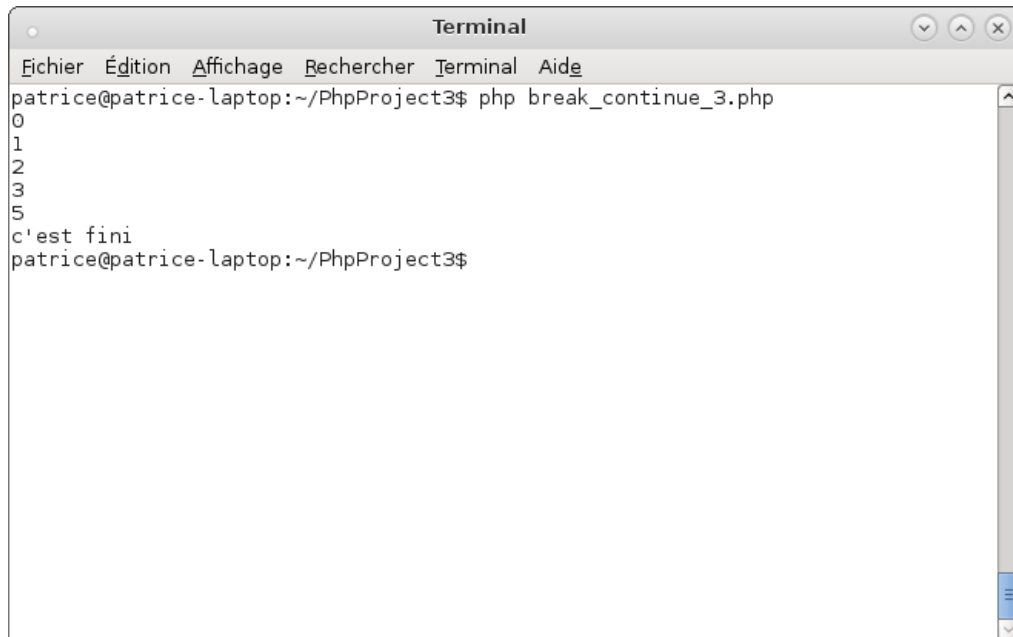
```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php break_continue_2.php
0
1
2
3
c'est fini
patrice@patrice-laptop:~/PhpProject3$
```

L'instruction [continue](#), elle, ne casse pas la boucle mais la fait passer au tour suivant.

```
1 <?php
2
3 // break_continue_3.php
4 for ($i = 0; $i < 6; $i++) {
```

```
5     if ($i == 4) {  
6         continue;  
7     }  
8     echo $i . PHP_EOL;  
9 }  
10 echo 'c'est fini' . PHP_EOL;
```

Ici lorsque i vaut 4, on passe tout de suite au tour suivant, sans exécuter l'affichage.



```
Terminal  
Fichier Édition Affichage Rechercher Terminal Aide  
patrice@patrice-laptop:~/PhpProject3$ php break_continue_3.php  
0  
1  
2  
3  
5  
c'est fini  
patrice@patrice-laptop:~/PhpProject3$
```

Fin

NAVIGATION

Accueil

- Ma page

Pages du site

Mon profil

Cours actuel

Intro PHP

Participants

Le langage PHP : introduction

Les types et les variables

Les opérateurs

Les structures de contrôle

 Boucles et conditions


 La condition if

 L'opérateur ternaire

 T.P. conditions

 La structure switch

 La boucle for

 T.P. boucles for

 La boucle while

 La boucle do-while

 T.P. boucles while et do-while

 **Break et continue**

 T.P. imbrications

Les structure de données

Les fonctions

Les erreurs

Les fichiers

Les expressions rationnelles

PHP et HTML

Petite application

[Mes cours](#)

ADMINISTRATION



Administration du cours

Réglages de mon profil

Connecté sous le nom « [Arnaud Lemais](#) » ([Déconnexion](#))
[Intro PHP](#)

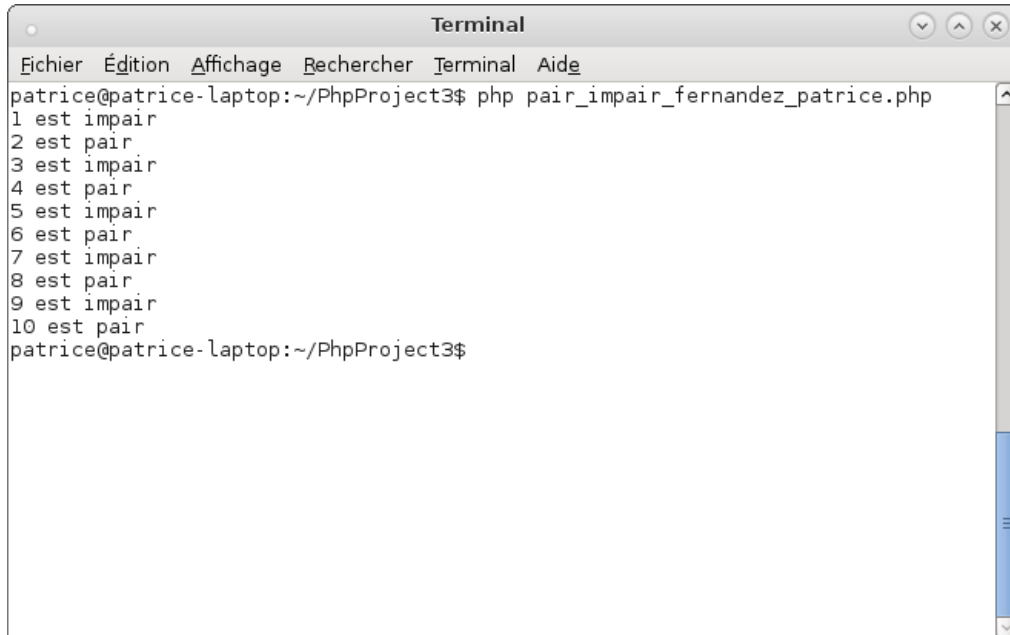


Introduction au langage de programmation PHP

[Accueil](#) ▶ [Mes cours](#) ▶ [Développement logiciel](#) ▶ [Intro PHP](#) ▶ [Les structures de contrôle](#) ▶ [T.P. imbrications](#)

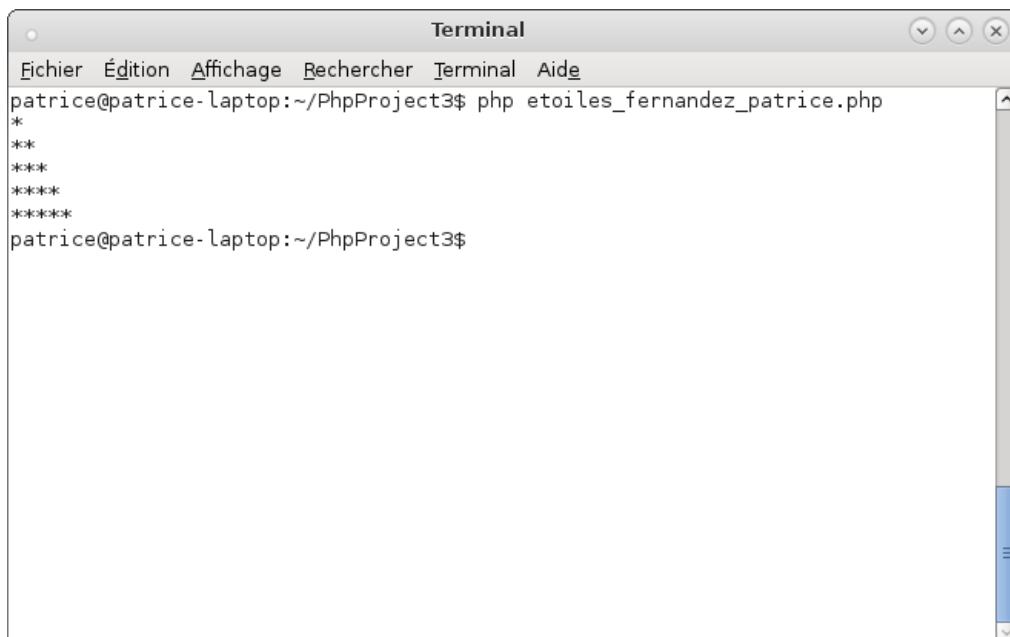
T.P. imbrications

Dans un fichier dont le nom sera de la forme "pair_impair_*nom_prenom*.php", implémentez un algorithme qui permettra d'afficher pour chaque nombre entre 1 et 10 inclus s'il est pair ou impair :



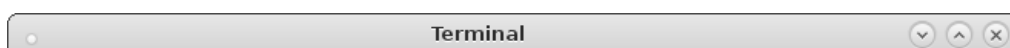
```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php pair_impair_fernandez_patrice.php
1 est impair
2 est pair
3 est impair
4 est pair
5 est impair
6 est pair
7 est impair
8 est pair
9 est impair
10 est pair
patrice@patrice-laptop:~/PhpProject3$
```

Dans un fichier dont le nom sera de la forme "etoiles_*nom_prenom*.php", implémentez un algorithme qui permettra d'afficher des étoiles sur 5 étages :

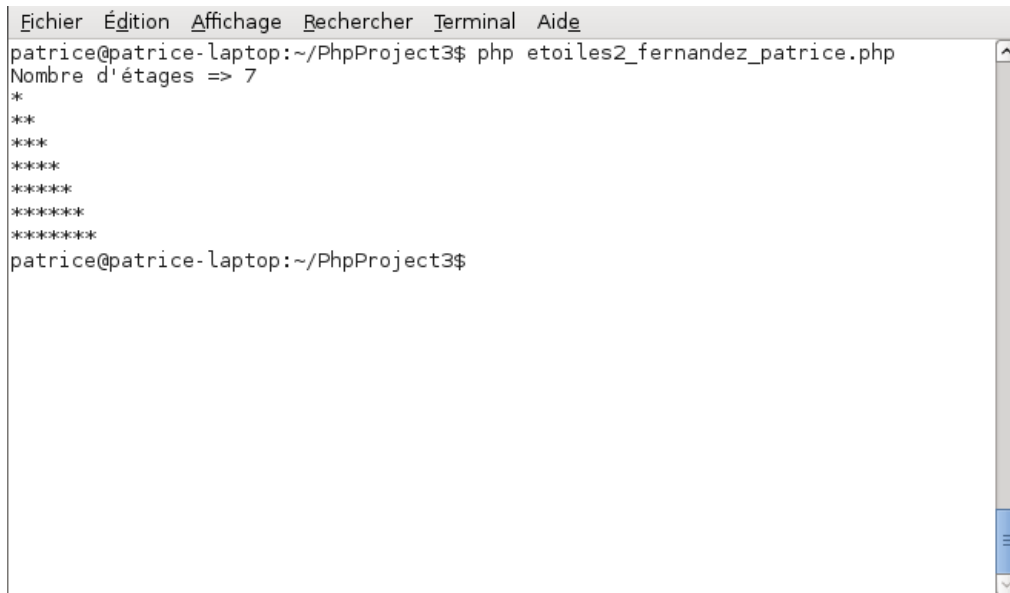


```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php etoiles_fernandez_patrice.php
*
**
***
****
*****
patrice@patrice-laptop:~/PhpProject3$
```

Dans un fichier dont le nom sera de la forme "etoiles2_*nom_prenom*.php", implémentez un algorithme qui permettra d'afficher des étoiles sur n étages, n étant saisi par l'utilisateur :

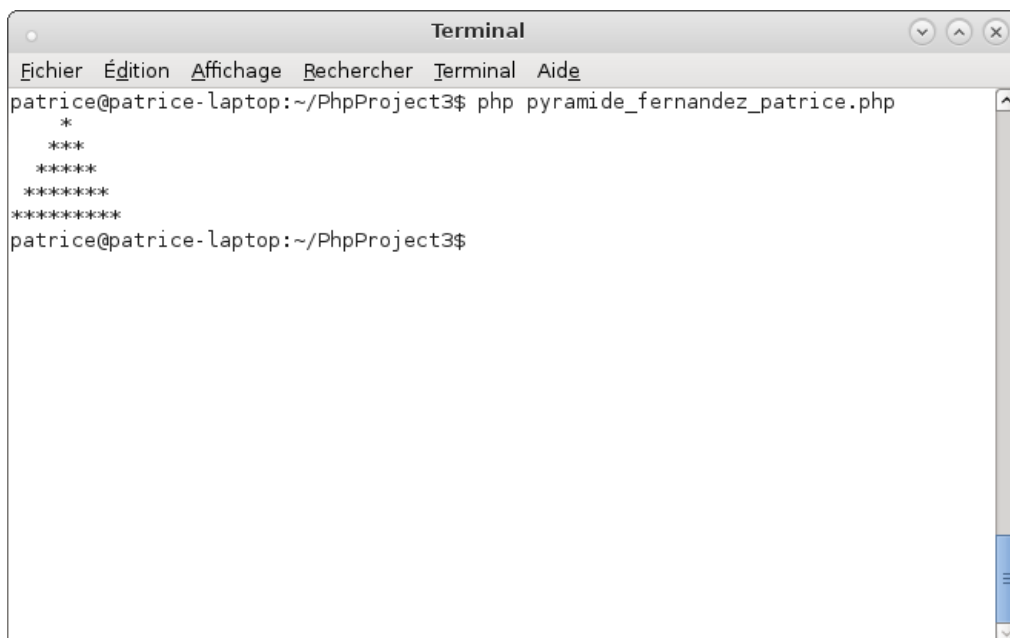


```
Terminal
```

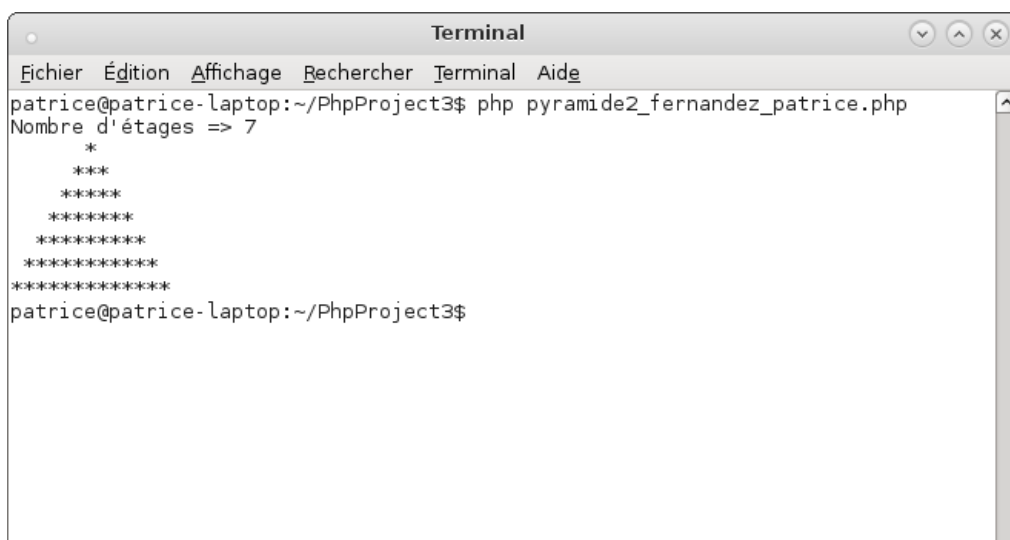
```
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php etoiles2_fernandez_patrice.php
Nombre d'étages => 7
*
**
***
****
*****
*****
*****
patrice@patrice-laptop:~/PhpProject3$
```

Dans un fichier dont le nom sera de la forme "pyramide_*nom_prenom*.php", implémentez un algorithme qui permettra d'afficher une pyramide d'étoiles de 5 étages :



```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php pyramide_fernandez_patrice.php
*
**
***
****
*****
patrice@patrice-laptop:~/PhpProject3$
```

Dans un fichier dont le nom sera de la forme "pyramide2_*nom_prenom*.php", implémentez un algorithme qui permettra d'afficher une pyramide d'étoiles de n étages, n étant un entier saisi par l'utilisateur :



```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php pyramide2_fernandez_patrice.php
Nombre d'étages => 7
*
**
***
****
*****
*****
*****
patrice@patrice-laptop:~/PhpProject3$
```



Vous remettrez une archive zip contenant les fichiers. Le nom de l'archive sera de la forme "imbrications_*nom_prenom*.zip".

État du travail remis





| | |
|--------------------------|---------------------------|
| Numéro de tentative | Ceci est la tentative 1. |
| Statut des travaux remis | Remis pour évaluation |
| Statut de l'évaluation | Noté |
| Dernière modification | jeudi 26 mars 2015, 15:25 |
| Remises de fichiers | |

[Modifier le travail](#)[Modifier votre travail remis](#)

Feedback

| | |
|---------------------------|---|
| Note | 100,00 / 100,00 |
| Évalué le | jeudi 26 mars 2015, 16:00 |
| Évalué par |  Patrice Fernandez |
| Feedback par commentaires | C'est bon. |

NAVIGATION

[Accueil](#)[Ma page](#)[Pages du site](#)[Mon profil](#)[Cours actuel](#)[Intro PHP](#)[Participants](#)[Le langage PHP : introduction](#)[Les types et les variables](#)[Les opérateurs](#)[Les structures de contrôle](#) [Boucles et conditions](#) [La condition if](#) [L'opérateur ternaire](#) [T.P. conditions](#) [La structure switch](#) [La boucle for](#) [T.P. boucles for](#) [La boucle while](#)

[La boucle do-while](#)[T.P. boucles while et do-while](#)[Break et continue](#)[T.P. imbrications](#)[Les structure de données](#)[Les fonctions](#)[Les erreurs](#)[Les fichiers](#)[Les expressions rationnelles](#)[PHP et HTML](#)[Petite application](#)[Mes cours](#)

ADMINISTRATION

[Administration du cours](#)[Réglages de mon profil](#)

Connecté sous le nom « [Arnaud Lemais](#) » ([Déconnexion](#))
[Intro PHP](#)