Clase 03

Análisis de Sistemas

Materia:
Programación Web II

Docente contenidista: ROLDÁN, Hernán

Revisión: Coordinación

Contenido

Clase 3

¡Te damos la bienvenida a la materia **Programación Web II**!

En esta clase vamos a ver los siguientes temas:

- Constantes.
- Operadores.
- If...Else...Elseif.
- Switch.
- Loops.
- Break y Continue.

Constantes

Una **constante** en PHP es una variable que no puede ser reasignada o redefinida a lo largo del script. Una vez que se ha definido una constante, su valor no puede ser cambiado. Las constantes se definen con la función define().

Las constantes se usan en PHP para almacenar valores que deben ser accedidos globalmente y no cambiados a lo largo del tiempo. Algunos ejemplos de uso de constantes son: valores de configuración, valores constantes utilizados en múltiples partes del código, etc.

Sintaxis

define(name, value, case-insensitive)

```php
<?php

   define('BD_SERVIDOR', 'localhost');
   define('BD_USUARIO', 'elon');
   define('BD_CLAVE', 'argentinacampeon2022');
   define('BD_NOMBRE', 'cursos');

  $conn = mysqli_connect(BD_SERVIDOR, BD_USUARIO, BD_CLAVE,
BD_NOMBRE);

   if(!$conn){
       die("Error de conexión: " . mysqli_connect_error());
   }
   echo "Conexión exitosa";
   mysqli_close($conn);

?>
```

Las constantes **BD_SERVIDOR**, **BD_USUARIO**, **BD_CLAVE** y **BD_NOMBRE** se definen para almacenar información sobre la conexión a la base de datos. Luego, se utiliza la función mysqli_connect para conectarse a la base de datos utilizando estas constantes como argumentos. Si la conexión es exitosa, se imprimirá "Conexión exitosa", de lo contrario, se mostrará un mensaje de error. Finalmente, se cierra la conexión utilizando mysqli_close.

Operadores

En programación, los **operadores** son símbolos o palabras clave que representan una acción o una operación matemática que se realiza en datos o variables. Por ejemplo, el operador + es utilizado para sumar dos números, el operador - para restar, * para multiplicar, y / para dividir.

PHP tiene varios tipos de operadores, incluyendo:

- **Aritméticos:** +, -, *, /, % (módulo), ** (potencia).
- **de Asignación:** =, +=, -=, *=, /=, %=, .= (concatenación).
- **de Comparación:** == (igual), != (diferente), < (menor que), > (mayor que), <= (menor o igual que), >= (mayor o igual que), === (idéntico), !== (no idéntico).
- **Lógicos:** && (y lógico), || (o lógico), ! (negación lógica).
- **de Incremento/Decremento:** ++$x (incremento), --$x (decremento).
- **de Cadena:** . (concatenación).
- **de Array:** + (unión de arrays), == (igualdad de arrays), === (identidad de arrays), != (desigualdad de arrays), <> (desigualdad de arrays), !== (desidentidad de arrays).
- **de Control de errores:** @ (supresión de errores).
- **de Tipos:** instanceof (verificación de tipo de objeto).
- **de Expresión ternaria:** ?: (operador ternario).

Estos son los tipos de operadores más comunes en PHP, y cada uno de ellos tiene una función específica y una sintaxis particular para su uso en el código.

Para más información, podés consultar la sección PHP: Operadores en la documentación oficial del lenguaje.

A continuación, algunos ejemplos prácticos de los operadores anteriormente citados.

Operadores aritméticos:
Los operadores aritméticos son símbolos matemáticos utilizados en programación para realizar operaciones aritméticas básicas como la suma, resta, multiplicación, división y módulo. Estos operadores permiten realizar cálculos y modificar variables en el código.

Ejemplo de uso del operador de adición

```php
<?php

   $x = 10;
   $y = 20;
   $z = $x + $y;

   // $z ahora es igual a 30

?>
```

Ejemplo de uso del operador de sustracción

```php
<?php

   $x = 10;
   $y = 20;
   $z = $y - $x;

   // $z ahora es igual a 10

?>
```

Ejemplo de uso del operador de multiplicación

```php
<?php

   $x = 10;
   $y = 20;
   $z = $x * $y;

   // $z ahora es igual a 200

?>
```

Ejemplo de uso del operador de división

```php
<?php

   $x = 10;
   $y = 20;
   $z = $y / $x;

   // $z ahora es igual a 2

?>
```

Ejemplo de uso del operador de módulo

```php
<?php

   $x = 10;
   $y = 20;
   $z = $y % $x;

   // $z ahora es igual a 0

?>
```

Ejemplo de uso del operador de potencia

```php
<?php

   $x = 2;
   $y = 10;
   $z = $x ** $y;

   // $z ahora es igual a 1024

?>
```

Operadores de asignación:
Los operadores de asignación son símbolos utilizados en programación para asignar valores a variables.

```php
<?php

   // Asignación (=)
   $x = 5;
   echo $x; // 5

   // Asignación suma (+=)
   $x = 5;
   $x += 3;
   echo $x; // 8

   // Asignación resta (-=)
   $x = 5;
   $x -= 3;
   echo $x; // 2

   // Asignación multiplicación (*=)
   $x = 5;
   $x *= 3;
   echo $x; // 15

   // Asignación división (/=)
   $x = 6;
   $x /= 2;
   echo $x; // 3

   // Asignación módulo (%)
   $x = 7;
   $x %= 3;
   echo $x; // 1

   // Asignación de concatenación (.=)
   $x = "Hola";
   $x .= " mundo";
   echo $x; // Hola mundo

?>
```

Operadores de comparación:
Los operadores de comparación en programación son símbolos utilizados para comparar dos valores y determinar si se cumplen o no ciertas condiciones. Estos operadores devuelven un valor booleano (verdadero o falso) dependiendo del resultado de la comparación.

```php
<?php

    $a = 5;
    $b = 10;

    // Operador igual a
    if($a == $b){
        echo "$a es igual a $b\n";
    }
    else{
        echo "$a no es igual a $b\n"; // 5 no es igual a 10
    }

    // Operador no igual a
    if($a != $b){
        echo "$a no es igual a $b\n"; // 5 no es igual a 10
    }
    else{
        echo "$a es igual a $b\n";
    }

    // Operador mayor que
    if($a > $b){
        echo "$a es mayor que $b\n";
    }
    else{
        echo "$a no es mayor que $b\n"; // 5 no es mayor que 10
    }
```

```php
    // Operador menor que
    if($a < $b){
        echo "$a es menor que $b\n"; // 5 es menor que 10
    }
    else{
        echo "$a no es menor que $b\n";
    }

    // Operador mayor o igual que
    if($a >= $b){
        echo "$a es mayor o igual que $b\n";
    }
    else{
        echo "$a no es mayor o igual que $b\n"; // 5 no es mayor o
igual que 10
    }

    // Operador menor o igual que
    if($a <= $b){
        echo "$a es menor o igual que $b\n"; // 5 es menor o igual que
10
    }
    else{
        echo "$a no es menor o igual que $b\n";
    }

?>
```

Operadores lógicos:
Los operadores lógicos son simbolos que se utilizan para realizar operaciones lógicas y comparaciones en un programa de computadora. Estos operadores permiten evaluar expresiones lógicas y determinar si son verdaderas o falsas, su uso más común es en estructuras de control de flujo, como if-else, para tomar decisiones en el código basadas en el resultado de una comparación o operación lógica.

En PHP, los operadores lógicos son similares a los de otros lenguajes de programación, estos operadores incluyen:

- **AND (&&):** Devuelve verdadero si ambas expresiones son verdaderas.

```php
<?php

   if($a == 5 && $b == 10){
       // hacer algo
   }

?>
```

- **OR (||):** Devuelve verdadero si al menos una de las expresiones es verdadera.

```php
<?php

   if($a == 5 || $b == 10){
       // hacer algo
   }

?>
```

- **NOT (!):** Invierte el valor booleano de la expresión.

```php
<?php

   if(!($a == 5)){
       // hacer algo
   }

?>
```

Operadores de incremento/decremento:
Los operadores de incremento y decremento son operadores unarios que se utilizan en la programación para aumentar o disminuir el valor de una variable en una unidad. Hay dos operadores de incremento y dos operadores de decremento en la mayoría de los lenguajes de programación, y PHP no es la excepción:

- **Incremento (++):** Aumenta el valor de una variable en una unidad.

- **Decremento (--):** Decrementa el valor de una variable en una unidad.

- **Pre-incremento (++x):** Aumenta el valor de una variable antes de ser utilizado en una expresión.

- **Pre-decremento (--x):** Decrementa el valor de una variable antes de ser utilizado en una expresión.

Operador pre-incremento:

```php
<?php

   $x = 5;
   ++$x;
   echo $x; // Salida: 6

?>
```

Operador post-incremento:

```php
<?php

   $x = 5;
   $y = $x++;
   echo $y; // Salida: 5
   echo $x; // Salida: 6

?>
```

Operador post decremento:

```php
<?php

   $x = 5;
   --$x;
   echo $x; // Salida: 4

?>
```

Operador post decremento:

```php
<?php

   $x = 5;
   $y = $x--;
   echo $y; // Salida: 5
   echo $x; // Salida: 4

?>
```

Los operadores de incremento y decremento se utilizan a menudo en bucles for y while para controlar el número de iteraciones. Por ejemplo, en un bucle for, se puede utilizar el operador de incremento para aumentar el valor de un contador en una unidad en cada iteración:

```php
<?php

   $x = 15;

   for($i=1; $i<=$x; $i++){
       echo $i . "\n"; // imprime los números 1 al 15
    }

?>
```

Operadores de cadena:
Los operadores de cadena son operadores que permiten unir o concatenar dos o más cadenas de texto en una sola. En muchos lenguajes de programación, incluyendo PHP, se utiliza el operador de concatenación "." para unir dos o más cadenas de texto.

```php
<?php

   $nombre = "Bart";
   $apellido = "Simpson";
   $nombre_completo = $nombre . " " . $apellido;

   echo $nombre_completo; // Salida: "Bart Simpson"

?>
```

Operadores de array:
Los operadores de array en programación son operadores específicos que se utilizan para trabajar con arreglos, que son estructuras de datos que almacenan una secuencia ordenada de valores. Algunos ejemplos de operadores de array incluyen:

Ejemplo de operador de unión de arrays:

```php
<?php

   $array1 = array(1, 2, 3);
   $array2 = array(4, 5, 6);
   $union = $array1 + $array2;
   // $union será [1, 2, 3, 4, 5, 6]

?>
```

Ejemplo de operador de igualdad de arrays:

```php
<?php

   $array1 = array(1, 2, 3);
   $array2 = array(1, 2, 3);
   $equal = ($array1 == $array2);
   // $equal será true

?>
```

Ejemplo de operador de identidad de arrays:

```php
<?php

   $array1 = array(1, 2, 3);
   $array2 = array(1, 2, 3);
   $identical = ($array1 === $array2);
   // $identical será true

?>
```

Ejemplo de operador de desigualdad de arrays:

```php
<?php

   $array1 = array(1, 2, 3);
   $array2 = array(4, 5, 6);
   $not_equal = ($array1 != $array2);
   // $not_equal será true

   $not_equal = ($array1 <> $array2);
   // $not_equal será true

?>
```

Ejemplo de operador de desidentidad de arrays:

```php
<?php

   $array1 = array(1, 2, 3);
   $array2 = array(4, 5, 6);
   $not_identical = ($array1 !== $array2);
   // $not_identical será true

?>
```

Operadores de control de errores:
Los operadores de control de errores en PHP son un conjunto de símbolos que permiten controlar los errores que ocurren en un script PHP.

En PHP, el operador '@' es conocido como el operador de supresión de errores y permite suprimir la visualización de un error en tiempo de ejecución. Al colocar el operador @ antes de una instrucción en un script PHP, se evita que cualquier mensaje de error generado por esa instrucción se muestre en la salida. Este operador es útil cuando se quiere ocultar errores no críticos o para evitar que los errores públicos estén disponibles para los usuarios finales. Sin embargo, suprimir los errores no es la forma recomendada de manejar errores en PHP, ya que puede dificultar la depuración y el seguimiento de problemas en la aplicación. En su lugar, se recomienda usar una técnica de manejo de errores adecuada, como la función set_error_handler() o la gestión de excepciones.

```php
<?php

   @$archivo = fopen('alta_de_usuarios.txt', 'r');

   if(!$archivo){
       echo 'Error al abrir el archivo "alta_de_usuarios.txt".';
   }
   else{
       echo 'Lectura del archivo exitosa.';
       fclose($archivo);
   }

?>
```

Operadores de Tipos:
Los operadores de tipo se utilizan para verificar el tipo de una variable. Hay dos operadores de tipo en PHP: instanceof y gettype.

- **instanceof:** El operador instanceof en PHP se utiliza para verificar si un objeto es una instancia de una determinada clase o de una clase que hereda de ella.

Ejemplo de uso del operador instanceof:

```php
<?php

   Class Animal{}
   Class Perro extends Animal{}

   $obj_perro = new Perro();

   if($obj_perro instanceof Animal){
       echo '"obj_perro" es instancia de la clase Animal.';
   }
   // Salida: La instancia "obj_perro" es instancia de la clase
Animal.


    echo "\n";

   if($obj_perro instanceof Perro){
       echo '"obj_perro" es instancia de la clase Perro.';
   }
   // Salida: La instancia "obj_perro" es instancia de la clase
Animal.

?>
```

- **gettype:** Este operador se utiliza para obtener el tipo de una variable en forma de cadena.

Ejemplo de uso del operador gettype:

```php
<?php

   Class Animal{}
   $obj_animal = new Animal();
   echo "\n";
   echo gettype($obj_animal);

   $nombre = 'John Wick';
   echo "\n";
   echo gettype($nombre);

   $edad = 10;
   echo "\n";
   echo gettype($edad);

   $precio = 100.5;
   echo "\n";
   echo gettype($precio);

   $luces_encendidas = true;
   echo "\n";
   echo gettype($luces_encendidas);

?>
```

Operadores ternarios:
Un operador ternario es un tipo de operador en programación que permite evaluar una expresión y elegir un valor o bloque de código en función de si la expresión es verdadera o falsa. Estos operadores son una forma abreviada de escribir una sentencia if-else y se representan mediante el símbolo ? seguido de :. En su forma más básica, la sintaxis de un operador ternario es la siguiente:

expresión ? valor_si_verdadera : valor_si_falsa

Donde expresión es cualquier expresión que puede evaluarse como verdadera o falsa, y valor_si_verdadera y valor_si_falsa son los valores o bloques de código que se deben elegir en función del resultado de la evaluación de expresión.

Como dijimos, el operador ternario en PHP es una forma abreviada de escribir una sentencia if-else en una sola línea. Aquí hay algunos ejemplos de su uso:

```php
<?php

   $a = 42;
   $b = 23;

   // Uso básico
   echo ($a > $b) ? '$a es mayor que $b' : '$a no es mayor que $b';
   // Salida: $a es mayor que $b

   // Asignación de valores
   $result = ($a > $b) ? $a : $b;
   echo $result;
   // Salida: 42

   // Uso con funciones
   $array = [1, 2, 3];
   $count = count($array);
   echo ($count > 0) ? "Hay $count elementos en el array" : 'El
array está vacío';
   // Salida: Hay 3

?>
```

If...Else...Elseif

Un bloque if-else-elseif es un tipo de estructura de control en programación que permite ejecutar diferentes bloques de código en función de si una o más condiciones son verdaderas o falsas. La sintaxis básica de un bloque if-else-elseif es la siguiente:

```
if(condición){
  // Código a ejecutar si la condición es verdadera
}
elseif(otra_condición){
  // Código a ejecutar si la primera condición es falsa y la segunda es verdadera
}
else{
  // Código a ejecutar si ninguna de las condiciones anteriores es verdadera
}
```

El bloque if es el bloque principal y contiene la condición que se debe evaluar.

Si la condición es verdadera, se ejecutará el bloque de código correspondiente.

Si la condición es falsa, se evaluarán las condiciones en los bloques elseif adicionales, si existen.

Si ninguna de las condiciones es verdadera, se ejecutará el bloque else, si existe.

Ejemplo de uso del bloque if

```php
<?php

   $num = 5;

   if($num > 10){
       echo "El número es mayor a 10.";
   }
   elseif($num < 10){
       echo "El número es menor a 10.";
   }
   else{
       echo "El número es igual a 10.";
   }

   // Salida: El número es menor a 10.

?>
```

El ejemplo de arriba evalúa la variable $num y determina si es mayor que 10, menor que 10 o igual a 10. En función de eso, se imprime un mensaje correspondiente.

Switch

En programación, un bloque switch es una estructura de control que permite la ejecución de un bloque de código específico de entre varios posibles, basado en el valor de una expresión o variable. Es similar a una serie de sentencias if-else, pero a menudo resulta más legible y eficiente para grandes conjuntos de casos.

La sintaxis básica de un bloque switch en algunos lenguajes de programación populares incluye la palabra clave "switch" seguida por la expresión o variable que se va a evaluar y varios bloques "case" que especifican los posibles valores y el código correspondiente a ejecutar en cada caso. También se puede utilizar un bloque "default" para especificar un comportamiento por defecto en caso de que ningún otro "case" coincida con la expresión evaluada.

Sintaxis

```
switch(expression){

   case value1:

     // código a ejecutar si expression es igual a value1
     break;

   case value2:

     // código a ejecutar si expression es igual a value2
     break;
   ...
   default:
     // código a ejecutar si expression no coincide con ningún
otro case
     break;

 }
```

La sentencia switch en PHP está compuesta por las siguientes partes:

1. La palabra clave ‘switch’, seguida de una expresión que se evalúa y su valor se compara con los valores especificados en los casos.

2. Un bloque de casos definidos con la palabra clave ‘case’, seguidos del valor que se desea comparar con la expresión evaluada.

3. Un bloque de código que se ejecutará si la expresión coincide con el valor especificado en algún caso.

4. Una sentencia ‘break’ para detener la ejecución después de ejecutar el bloque de código correspondiente.

5. Una sentencia opcional ‘default’ que se ejecutará si ninguno de los casos coincide con la expresión evaluada.

Debajo algunos ejemplos de la sentencia switch en PHP:

Ejemplo 1

```php
<?php

   $dia = "lunes";

   switch($dia){
      case "lunes":
        echo "Hoy es lunes.";
        break;
      case "martes":
        echo "Hoy es martes.";
        break;
      case "miércoles":
        echo "Hoy es miércoles.";
        break;
      default:
        echo "Hoy no es ni lunes, ni martes, ni miércoles.";
        break;
      }

      // Salida: Hoy es lunes.

?>
```

Ejemplo 2

```php
<?php

   $personajes_simpsons = ['Homer', 'Marge', 'Bart', 'Lisa',
'Maggie'];

   // print_r($personajes_simpsons);

   $personaje_actual = 'Lisa';

   echo "\n";

   switch($personajes_simpsons){

    case $personaje_actual == $personajes_simpsons[0]:
      echo 'Homer';
      break;

    case $personaje_actual == $personajes_simpsons[1]:
      echo 'Marge';
      break;

    case $personaje_actual == $personajes_simpsons[2]:
      echo 'Bart';
      break;

    case $personaje_actual == $personajes_simpsons[3]:
      echo 'Lisa';
      break;

    case $personaje_actual == $personajes_simpsons[4]:
      echo 'Maggie';
      break;

    default:
      echo 'No es un personaje de la familia Simpson.';

   }

   // Salida: Lisa

?>
```

Es importante mencionar que, si no se coloca una sentencia break después de cada caso, la ejecución continuará hasta encontrar una sentencia break o hasta el final de la sentencia switch. Esto puede ser útil para ejecutar varios bloques de código seguidos en el mismo caso, pero también puede ser una fuente de errores y debes ser cuidadoso al usar esta técnica.

Loops

Los loops en programación son estructuras de control que permiten ejecutar un bloque de código varias veces, hasta que se cumpla una condición específica. En otras palabras, los loops te permiten repetir un proceso hasta que se cumpla una determinada condición.

Hay dos tipos principales de loops en programación:

1. **For loops:** Este tipo de loop se utiliza para repetir un bloque de código un número determinado de veces. Por ejemplo, puedes utilizar un for loop para imprimir los números del 1 al 10.

2. **While loops:** Este tipo de loop se utiliza para repetir un bloque de código mientras se cumpla una condición determinada. Por ejemplo, puedes utilizar un while loop para imprimir los números mientras una variable sea menor que 10.

Los loops son una herramienta muy útil en programación y te permiten automatizar tareas repetitivas y realizar cálculos complejos de manera más eficiente.

En PHP hay 4 tipos de loops: for, while, do...while y foreach.

1. **For Loop:** El for loop se utiliza para repetir un bloque de código un número determinado de veces. Tiene las siguientes partes:
    - La primera parte, es donde se inicializa la variable del contador.
    - La segunda parte, es la condición que debe ser cumplida para que el loop continúe ejecutándose.
    - La tercera parte, es donde se incrementa o decrementa la variable del contador en cada iteración del loop.

Sintaxis

for(init counter; test counter; increment counter){

código a ejecutar por cada iteración;

}

Ejemplo:

```php
<?php

  for ($i = 1; $i <= 5; $i++){
    echo $i . "\n";
  }

  /*

    Salida:
    1
    2
    3
    4
    5

  */

?>
```

2. **While Loop:** El while loop se utiliza para repetir un bloque de código mientras se cumpla una determinada condición. Tiene solo una parte:
    - La parte, es la condición que debe ser cumplida para que el loop continúe ejecutándose.

Sintaxis

while(condition is true){

código a ejecutar;

}

Ejemplo:

```php
<?php

  $i = 1;

  while($i <= 5){

    echo $i . "\n";

    $i++;

  }

  /*

  Salida:
  1
  2
  3
  4
  5

  */

?>
```

3. ***Do...While Loop:*** El do...while loop es similar al while loop, pero se asegura de que el bloque de código se ejecute al menos una vez antes de comprobar la condición. Tiene las siguientes partes:
    - El bloque de código que se ejecuta en cada iteración del loop.
    - La parte, es la condición que debe ser cumplida para que el loop continúe ejecutándose.

Sintaxis

do{

código a ejecutar;

}while(condition is true);

Ejemplo

```php
<?php

  $i = 1;

  do{
    echo $i . "\n";
    $i++;
  }while($i <= 5);

  /* Salida:

    1
    2
    3
    4
    5

  */

?>
```

4. **Foreach:** El foreach loop en PHP es un tipo de loop diseñado específicamente para recorrer arrays y objetos iterables. Es muy útil para realizar tareas repetitivas con los elementos de un array, como imprimirlos en pantalla o modificarlos. El foreach loop consta de dos partes:
    - La primera parte, es el array u objeto que se quiere recorrer.
    - La segunda parte, es la variable que se utilizará para acceder a los elementos del array u objeto en cada iteración.

Sintaxis

foreach($array as $valor){

code a ejecutar;

}

Ejemplo:

```php
<?php

   $escritores = array('Edgar Allan Poe', 'Howard Phillips Lovecraft',
'Julio Cortázar');

   foreach($escritores as $escritor){
       echo $escritor .  '.' . "\n";
   }

   // Salida
   // Edgar Allan Poe
   // Howard Phillip Lovecraft
   // Julio Cortázar

   echo "\n";

   echo $escritores[0] . '.'; // Salida: Edgar Allan Poe.
   echo "\n";
   echo $escritores[1] . '.'; // Salida: Howard Phillips Lovecraft.
   echo "\n";
   echo $escritores[2] . '.'; // Salida: Julio Cortázar.

?>
```

Break y Continue

Las sentencias "break" y "continue" en PHP son instrucciones de control de flujo que permiten modificar el comportamiento de los bucles y estructuras de control.

La sentencia "break" se utiliza para salir de un bucle o una estructura de control antes de que se complete su ejecución normal. Cuando se encuentra la instrucción "break", el flujo de control se mueve fuera del bucle o estructura de control, y continúa ejecutando la siguiente instrucción después del bucle o estructura de control.

Ejemplo de la sentencia "break" en un bucle "while":

```php
<?php

  $i = 0;

  while ($i < 10){
    echo $i . "\n";
    $i++;
    if($i == 5){
      break;
    }
  }

   /*

    Salida:
    0
    1
    2
    3
    4

  */

?>
```

En este ejemplo, el bucle "while" imprimirá los números del 0 al 4, y luego se detendrá debido a la instrucción "break".

La sentencia "continue" se utiliza para omitir la ejecución de una iteración particular de un bucle o estructura de control, y pasar a la siguiente iteración. Cuando se encuentra la instrucción "continue", el flujo de control salta a la próxima iteración del bucle o estructura de control.

Ejemplo de la sentencia "continue" en un bucle "for":

```php
<?php

  for($i = 0; $i < 10; $i++){
    if ($i % 2 == 0){
      continue;
    }
    echo $i . "\n";
  }

?>
```

En este ejemplo, el bucle "for" imprimirá los números impares del 1 al 9, ya que la instrucción "continue" omitirá la ejecución de las iteraciones donde el valor de "$i" es par.

Hemos llegado así al final de esta clase en la que vimos:

- Constantes.
- Operadores.
- If...Else...Elseif.
- Switch.
- Loops.
- Break y Continue.

Te esperamos en la **clase en vivo** de esta semana.
No olvides realizar el **desafío semanal**.

¡Hasta la próxima clase!

Bibliografía

Documentación Oficial de PHP:
https://www.php.net/manual/es/index.php

W3Schools: PHP Tutorial:
https://www.w3schools.com/php/default.asp

Cabezas Granado, L., González Lozano, F., (2018). Desarrollo web con PHP y MySQL. Anaya Multimedia.

Heurtel, O., (2016). Desarrolle un sitio web dinámico e interactivo. Eni Ediciones.

Welling, L., Thompson, L., (2017). Desarrollo Web con PHP y MySQL. Quinta Edición. Editorial Anaya.

Para ampliar la información

PHP: Constantes

PHP: Operadores

PHP: If... else... elseif

PHP: Switch

PHP: Estructuras de control

Guía de HTML

Todo sobre PHP

PHP: The Right Way

PHP Programming Language Tutorial - Full Course

PHP Full Course for non-haters

CURSO de php desde cero