TEMA 2.2

Estructuras de control en PHP

Condiciones: if

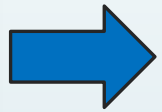
- En todo lenguaje de programación necesitamos poder expresar condiciones, no siempre querremos ejecutar determinadas sentencias.
- Una condición implica una comprobación lógica, si se cumple, se realiza la tarea, si no se cumple, no se realiza, o bien se realiza otra tarea diferente.
- En PHP las condiciones se implementan mediante la instrucción **if**. El **if** se traduce como "si" condicional. Si pasa esto, hacemos algo, si no pasa, hacemos otra cosa.
- Se utilizan paréntesis **( )** para englobar la expresión lógica de la condición, y se utilizan llaves **{ }** para englobar las sentencias de las tareas a realizar.
- Se escribiría algo así:

```
if ( pasa esto ) {
     hacemos algo
} else {
     hacemos otra cosa
}
```

- La palabra "**else**" indica "si no", en el caso de que no se cumpla la condición "pasa esto", el programa ejecutará las sentencias "hacemos otra cosa"

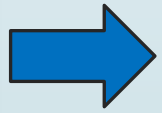
Condiciones: Ejemplos

```php
<?php
  $hora = 8; // La hora en formato de 24 horas
  if ($hora == 8) {
    echo "Suena el despertador";
  }
?>
```



Suena el despertador

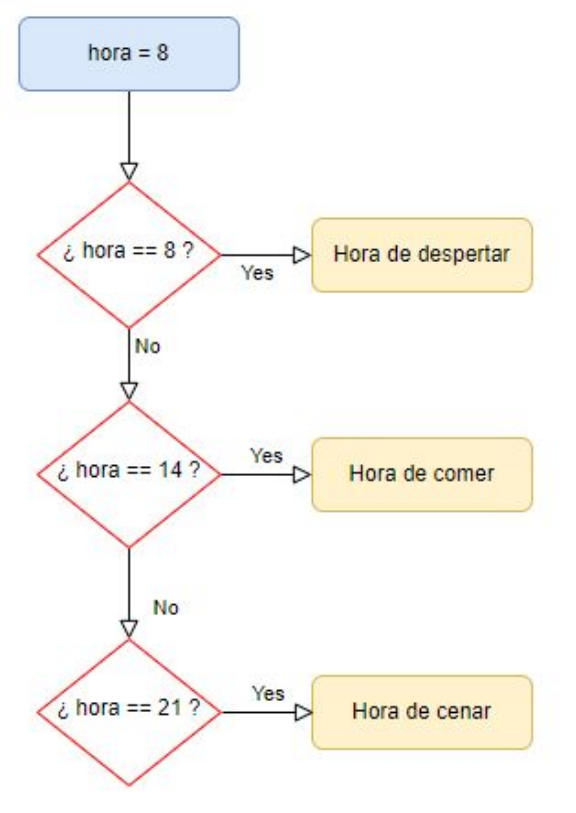
```php
<?php
  $numero = 3;
  if ($numero == 3) {
    echo "El número es 3 !!!";
  } else {
    echo "Resulta que el número no es 3!!!";
  }
?>
```



El número es 3 !!!

Condiciones: if anidados

- A veces, dependiendo del valor de una variable, queremos hacer cosas diferentes. Por ejemplo, según la hora del día, hacer cosas distintas.
- Una forma de programarlo es utilizando varios if anidados. Si somos organizados es posible que se visualice más o menos bien, como en el ejemplo siguiente:




```php
<?php
  $hora = 14; // La hora en formato de 24 horas
  if ($hora == 8) {
    echo "Es la hora de desayunar";
  } else if ($hora == 14) {
    echo "Es la hora de comer";
  } else if ($hora == 21) {
    echo "Es la hora de cenar";
  }
?>
```

Es la hora de comer

Condiciones: Abreviaturas

- Entre llaves ponemos un conjunto de sentencias a realizar.
- Si sólo hay una sentencia, no harían falta las llaves. No obstante, se recomienda utilizarlas, por claridad:

```php
<?php
  $hora = 14; // La hora en formato de 24 horas
  if ($hora == 8) {
    echo "Es la hora de desayunar";
  } else if ($hora == 14) {
    echo "Es la hora de comer";
  } else if ($hora == 21) {
    echo "Es la hora de cenar";
  }
?>
```

```php
<?php
  $hora = 14; // La hora en formato de 24 horas
  if ($hora == 8) echo "Es la hora de desayunar";
  else if ($hora == 14) echo "Es la hora de comer";
  else if ($hora == 21) echo "Es la hora de cenar";
?>
```

Es la hora de comer

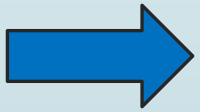
Condiciones: Operador ternario

- Una forma abreviada de expresar un if..else es el operador ternario
- Sirve para asignar valores a una variable según se cumpla o no una condición
- Su forma es la siguiente:

```
$variable = ( condicion ) ? valor1 : valor2 ;
```

- Los paréntesis **( )** engloban la **condición**, que podrá resultar en true o en false.
- El símbolo **?** indica que si se cumple la condición, el resultado de este operador ternario sería **valor1**.
- El **:** indica que, si no se cumple la condición, el resultado de este operador ternario sería el **valor2**.

```php
<?php
  $rica   = false;
  $guapa = true;
  $relacion = ($rica and $guapa)?"novia":"amiga";
  echo "Ella es mi $relacion";
?>
```

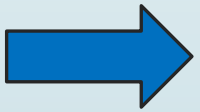


Ella es mi amiga

Condiciones: switch

- La sentencia **switch** permite ejecutar sentencias diferentes según el valor de una variable (o de una expresión).
- Los **break** son necesarios, si no se ponen seguiría comprobando el resto de condiciones.
- El **default** se ejecuta siempre que no se cumpla ninguna de las condiciones anteriores.
- El ejemplo anterior se escribiría así

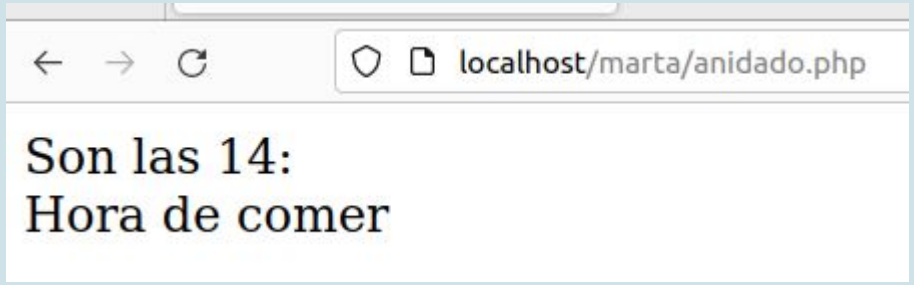
```php
<?php
  $hora = 14; // La hora en formato de 24 horas
  switch ($hora) {
    case 8:
      echo "Es la hora de desayunar";
      break;
    case 14:
      echo "Es la hora de comer";
      break;
    case 21:
      echo "Es la hora de cenar";
      break;
    default:
      echo "Ahora no toca comer";
  }
?>
```



Es la hora de comer

Ejercicio 1:

if anidado

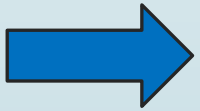


1. Crea un documento .php llamado anidado.php
2. Utiliza una variable que guarde una hora. Podrá ser un entero de 0 a 24.
3. Muestra la salida como la imagen de la izquierda, con las condiciones:
   - Si son entre las 7 y las 11, es hora de desayunar
   - Si son entre las 13 y las 15, es hora de comer
   - Si son entre las 20 y las 23, es hora de cenar
4. Recomendaciones: Utiliza un if anidado lo más abreviado y visual posible. Pruébalo varias veces en un navegador, con los valores de la variable: 6, 10, 12, 21
5. Modifica el programa para que si el valor no es uno de los indicados en las condiciones anteriores, el mensaje sea “Es hora de un tentempié”.

Bucles:

- Cuando queremos realizar una tarea muchas veces:
  - Podemos escribirla muchas veces: NO queremos hacer esto
  - Podemos utilizar un "bucle": Opción óptima
- Un bucle nos permite indicar cuántas veces queremos realizar la tarea
- Un bucle nos permite indicar que se realice la tarea una y otra vez, mientras se cumpla una condición. Por ejemplo, seguir comiendo mientras tengo hambre.
- Hay varias sentencias en PHP que nos permiten programar un bucle. Las veremos a continuación
- Un ejemplo de código repetitivo mal programado sería el siguiente:

```php
<?php
  $hora = 8;
  if ($hora == 8) echo "Suena el despertador<br>";
  if ($hora == 8) echo "Suena el despertador<br>";
  if ($hora == 8) echo "Suena el despertador<br>";
  if ($hora == 8) echo "Suena el despertador<br>";
  if ($hora == 8) echo "Suena el despertador<br>";
  if ($hora == 8) echo "Suena el despertador<br>";
  if ($hora == 8) echo "Suena el despertador<br>";
  if ($hora == 8) echo "Suena el despertador<br>";
  if ($hora == 8) echo "Suena el despertador<br>";
  if ($hora == 8) echo "Suena el despertador<br>";
?>
```



Suena el despertador
Suena el despertador
Suena el despertador
Suena el despertador
Suena el despertador
Suena el despertador
Suena el despertador
Suena el despertador
Suena el despertador
Suena el despertador

Bucle do...while:

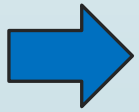
- Similar al bucle while, en do..while se ejecuta el código, y la condición se comprueba después

  `do { código que se repite } while ( condición )`

- La **condición** se comprueba *después* de ejecutar el código. Va entre paréntesis **( )**
- El **código** va dentro de las llaves **{ }**, es la tarea repetitiva. Se ejecutará una y otra vez, siempre que se cumpla la condición.
- **Diferencias** entre bucle **while** y **do..while**:
  - En el bucle **while**, la condición se comprueba antes de la ejecución del código.
  - La primera vez que se llega al bucle, **do..while** ejecuta el código sin condiciones. Es decir, siempre ejecuta el código una primera vez, y luego ya mira las condiciones.
  - La primera vez que se llega al bucle **while**, se comprueba la condición, si no se cumpliera podría no entrar nunca a ejecutar el código.

En este do..while, se ejecuta el código que hay entre llaves una primera vez, sin mirar ninguna condición.
Cuando termina el código, llega a la parte while y comprueba la condición ($numero <= 10).
Si se cumple, vuelve a ejecutar el código entre llaves. Si no se cumple termina el bucle.

```php
<?php
  $numero = 1;
  do {
    echo "$numero ";
    $numero++;
  } while ( $numero <= 10);
?>
```



1 2 3 4 5 6 7 8 9 10

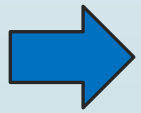
Bucle for:

- Si en vez de una condición, lo que queremos es que el código se repita un número de veces concreto, es mejor utilizar el bucle **for**.
- En el bucle for utilizaremos una variable que hace de contador. No es necesario contar en el código, ya que podemos indicarlo en el principio de la sentencia:

```
for ( $contador=1 ; $contador<=10 ; $contador++ ) {  echo $contador; }
```

- El **for** indica que vamos a repetir las sentencias que hay dentro de las llaves **{ }**
- Entre paréntesis **( )** ponemos tres informaciones separadas por **;**
  - La primera es el **valor inicial** del contador ($contador=1)
  - La segunda ($contador <= 10) indica la **condición** que tiene que cumplirse para ejecutar el código. Lo lógico en estos bucles es que se refiera a un valor límite para el contador utilizado.
  - La tercera ($contador++) indica que en cada iteración (repetición del código) se **modifica** el contador, y cómo. En este caso dice que tras cada vuelta le suma 1.

```php
<?php
  $numero = 1;
  for ($contador=1; $contador <= 10; $contador++) {
    echo "$contador ";
  }
?>
```



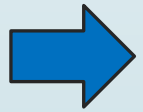
1 2 3 4 5 6 7 8 9 10

En este for, el valor inicial del contador es 1 ($contador=1), lo incrementamos en 1 cada vuelta del bucle ($contador++), y el bucle terminará en el momento que se deje de cumplir la condición ($contador <= 10), es decir, cuando lleguemos a 11 no se ejecutará el código y saldrá del bucle.

Bucle for ascendente y descendente:

- El contador utilizado en el bucle for puede contar de dos formas:
  1. **Ascendente**, por ejemplo 1,2,3,4,5... Para ello al modificar el contador lo incrementamos ($contador++). Era el caso del ejemplo anterior.
  2. **Descendente**, por ejemplo 10,9,8,7,6... Para ello al modificar el contador lo decrementamos ($contador--). En este caso la condición tendrá que ser un límite mínimo.
- Un ejemplo de contador descendente es iniciar el contador a 10, y luego ir restando 1 sucesivamente hasta llegar a 1, como la cuenta atrás de lanzamiento de un cohete.

```
<p>FOR DESCENDENTE<p>
<?php
  $numero = 1;
  for ($contador=10; $contador > 0; $contador--) {
    echo "$contador ";
  }
?>
```



FOR DESCENDENTE

10 9 8 7 6 5 4 3 2 1

En este for descendente, el valor inicial del contador es 10 ($contador=10), lo decrementamos en 1 cada vuelta del bucle ($contador--), y el bucle terminará en el momento que se deje de cumplir la condición ($contador > 0), es decir, cuando lleguemos a cero.

Bucles infinitos en el servidor

- Ya sabemos que un bucle repite (itera) un fragmento de código una y otra vez mientras se cumpla una condición. Si se cumple siempre, el bucle no para, es infinito.
- Si ejecutamos un bucle infinito en un script.php desde un cliente web , puede pasar lo siguiente:
  - El servidor web alcanza el tiempo límite configurado para ejecución de un script, y detiene el proceso enviando un error (HTTP 500)
  - El servidor web se queda sin recursos (memoria, CPU,...). El servidor web podrá detener el script para asegurar la estabilidad y rendimiento de otras webs alojadas en ese mismo servidor.
- En nuestra instalación XAMPP, el tiempo máximo de ejecución es de 120 segundos (2 minutos). Se configura en la variable max_execution_time del php.ini

En PHPinfo del dashboard podemos ver el valor de la variable en php.ini

max_execution_time	120

Ejercicio 2:

bucles

1. Crea un documento .php llamado juventud.php
2. Mostrará la salida de la izquierda, según lo indicado a continuación:
3. En el año 2000 tienes 30 años.
4. Vas a realizar un bucle que compruebe, todos los años, tu edad y tu juventud. El bucle se para cuando ya no eres joven.
5. Se considera joven una persona de menos de 40 años.
6. Recomendaciones: Utiliza un bucle while o do..while.
7. Utiliza variables para almacenar la edad, el año y la juventud.
8. Al programar el bucle, ¿qué pasa si te equivocas y haces un bucle infinito? Investiga la respuesta.

Ejercicio 3:

bucles

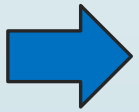
BUCLE WHILE
1, 2, 3, 4,
BUCLE DO..WHILE
1, 2, 3, 4,
BUCLE FOR
1, 2, 3, 4,

1. Crea un documento .php llamado forDescendente.php
2. Haz que muestre los números del 11 al 3, saltando de dos en dos, es decir:
3. 11,9,7,5,3
4. Para realizar este ejercicio utilizarás todos los bucles que hemos visto en PHP: while, do..while, y for.
5. Entrega un zip con el PHP y un pantallazo del resultado, que será similar a la salida de la izquierda.

Otra forma de salir de un bucle... No se recomienda!!

- Con la sentencia **break**, la ejecución se detendría, y saldría del bucle sin continuar el código ni volver a comprobar la condición
- Con la sentencia **continue**, la ejecución se detendría, no se terminaría de ejecutar el código, y se iría directamente a la condición para ver si vuelve a iterarse el código o por el contrario se sale del bucle.
- Hay que intentar no utilizar estas sentencias

```php
<?php
  $numero = 1;
  for ($contador=1; $contador <= 600; $contador++) {
    if ($contador == 5) break;
    echo "$contador ";
  }
?>
```



1 2 3 4

En este bucle quiero mostrar los números del 1 al 600 de forma ascendente.
La condición que "rompe" el bucle, terminándolo, hace el código confuso ya que pone condiciones adicionales fuera de las condiciones del bucle, esto es poco claro y difícil de mantener, sobre todo si el código es largo.

https://www.php.net/docs.php