MEDAC
Instituto Oficial de Formación Profesional

DESARROLLO WEB EN ENTORNO SERVIDOR

TÉCNICO EN DESARROLLO DE APLICACIONES WEB

Programación basada en lenguaje de marcas con código embebido I

03

ÍNDICE

/ 1. Introducción y contextualización práctica 3

/ 2. Sentencias 4

2.1. Sentencias condicionales IF; IF-ELSE 5

2.2. Sentencia switch-case 6

/ 3. Bucles 7

3.1. For 7

3.2. Bucles while y do-while 8

3.3. Bucle foreach 9

/ 4. Otras sentencias en PHP 10

/ 5. Arrays 11

5.1. Operaciones con arrays 12

/ 6. Matrices 13

6.1. Acceso a las posiciones de la matriz 14

/ 7. Caso práctico 1: “Bucles en PHP” 15

/ 8. Caso práctico 2: “Creación de matrices en PHP” 17

/ 9. Comentarios 18

/ 10. Resumen y resolución del caso práctico de la unidad 19

/ 11. Bibliografía 19

/ 12. Webgrafía 19

OBJETIVOS

Utilizar mecanismos de decisión en la creación de bloques de sentencias.

Utilizar bucles y verificar su funcionamiento.

Utilizar arrays para almacenar y recuperar conjuntos de datos.

Añadir comentarios al código.

Utilizar bloques de sentencias embebidos en lenguaje de marcas.

/ 1. Introducción y contextualización práctica

Cada lenguaje de programación presenta una sintaxis concreta, así como unas reglas para su implementación. Se trata de pequeñas particularidades entre lenguajes que permiten el correcto funcionamiento de cada uno de ellos.

Las sentencias en PHP pueden ser de diferentes tipos: lineales, cuya ejecución se completa y pasa a la siguiente instrucción, condicionales, que permiten evaluar diferentes condiciones y en función del resultado de estas ejecutar un bloque de código u otro, y, finalmente, los bucles, que permiten ejecutar de forma reiterativa, en función de si se cumple (o no) una condición un bloque de código.

Además, en este tema, veremos uno de los datos compuestos más importantes en el desarrollo de aplicaciones, los arrays, un conjunto de datos del mismo tipo a los que se accede en función de su clave de posición.

Escucha el siguiente audio donde planteamos la contextualización práctica de este tema. Encontrarás su resolución en el apartado «Resumen y resolución del caso práctico de la unidad».

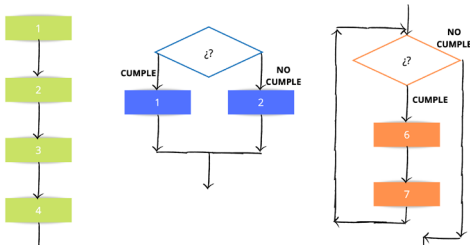

Fig. 1. Tipos de sentencias en PHP.

Audio intro. "¿Qué tipos de sentencias se implican en el diseño en PHP?"

https://bit.ly/3gI6Y9d

/ 2. Sentencias

Existen distintos tipos de sentencias, desde las que se ejecutan de forma lineal, hasta aquellas sentencias que permiten que el flujo de ejecución de un programa no sea lineal.

Las lineales nos serán de gran utilidad, puesto que, en función de ciertas condiciones, se producirá la toma de un camino durante la ejecución del programa o la de otro. En concreto, hablamos de las sentencias condicionales, también conocidas como **estructuras de control.**

Las instrucciones de ejecución lineal se ejecutan una tras otra. Si no existieran las sentencias condicionales o los bucles, no habría forma de variar el comportamiento.

Algunas de las sentencias lineales más conocidas son:

- **Expresiones de asignación (=):** $var1=$var2;
- **Formas postfijas o prefijas (++ y - -).** En las formas postfijas (varaible++), se realiza el incremento o decremento, pero no será hasta la siguiente iteración que se tome el nuevo valor, mientras que, en las prefijas (++variable), se toma el valor en cuanto se ejecuta la instrucción.
- **Llamadas a métodos: print(«Hola»);**
- **Expresiones de creación de objetos.**
- **Los bloques** de instrucciones se crean mediante las llaves { }, que permiten agrupar diferentes tipos de sentencias en un mismo bloque. También pueden utilizarse para agrupar todas las instrucciones asociadas a la ejecución de alguna de las ramificaciones de las sentencias condicionales o de los bucles de ejecución reiterativa.

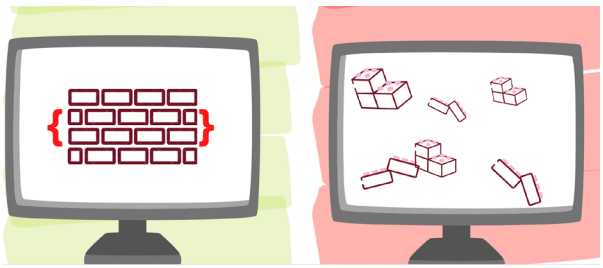

Fig. 2. Bloques de instrucciones con llaves.

Las **estructuras de control** permiten modificar el flujo de ejecución de las instrucciones de un programa, es decir, permiten realizar una serie de operaciones que incrementan las posibilidades de un programa. Si no existieran, el código sería completamente lineal.

Todos los lenguajes de programación presentan sus estructuras de control, similares entre ellas, añadiendo particularidades de funcionamiento en función del lenguaje escogido. Las sentencias condiciónales existentes en PHP se describen a continuación.

En este tema, nos vamos a ocupar de analizar en profundidad las sentencias, bucles y demás sintaxis propia del lenguaje para el desarrollo web en entorno servidor, PHP.

2.1. Sentencias condicionales IF; IF-ELSE

En la sentencia if se ejecutan las acciones codificadas si se cumple la condición evaluada; de lo contrario, el programa continuará con la ejecución de aquel código implementado a partir del bloque completo de la sentencia.

```
<?php
  if (condición){
    //acciones
      }
?>
```

Código 1. Sentencia if en PHP.

Fig. 3. Diagrama de funcionamiento if en PHP.

En la **sentencia if-else**, se ejecuta un grupo u otro de instrucciones en función de la condición evaluada. La diferencia principal con el caso anterior radica en que, ahora, se trata de una forma específica a la salida antes de volver a la ejecución del resto del programa en el que está incluida.

```
<?php
  if (condición){
            //acción1
    }else{
            //acción2
    }
?>
```

Código 2. Sentencia if-else en PHP

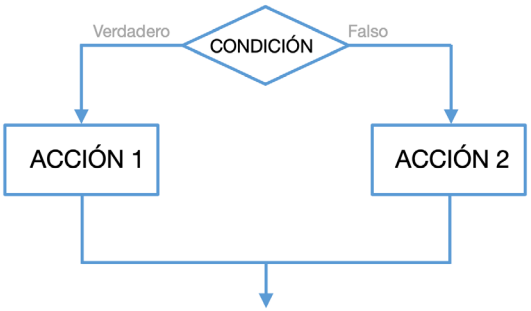

Fig. 4. Diagrama de funcionamiento if-else en PHP.

2.2. Sentencia switch-case

Este tipo de sentencias es similar a una de tipo if, pero, ahora, se evalúan más casos. Es apropiada para aquellos programas en los que el parámetro a evaluar puede tomar múltiples valores.

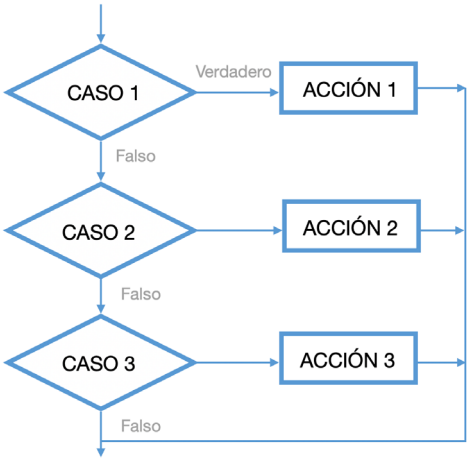

Fig. 5. Diagrama de funcionamiento switch en PHP.

Se ejecuta de forma lineal, de arriba hacia abajo, y cuando encuentra una **condición** que se cumple, ejecuta las acciones codificadas en ese bloque. Si, una vez finalizado el contenido del bloque, no se deja que se sigan evaluando el resto de condiciones, debe añadirse la palabra **break.**

Además, si ninguna de las condiciones se cumple, permite añadir un último bloque bajo el nombre **default**, dentro del cual se implementarán las acciones que se llevarán a cabo si se llega a ejecutar este último bloque.

```
<?php
  switch (condicion){
            case valor1:
                    //acción1
                    break;
            case valor2:
                    //acción2
                    break;
            case valorN:
                    //acciónN
                    break;
            default:
                    //acciónDefault
            }
?>
```

Código 3. Sentencia switch en PHP.

El bloque default no es obligatorio, del mismo modo que tampoco lo sería utilizar break si se desea seguir evaluando el resto de condiciones, para que la ejecución no se detenga tras haber cumplido una de las condiciones.

/ 3. Bucles

Los bucles son un tipo concreto de sentencias que permite la ejecución reiterativa de un bloque de instrucciones, si se cumplen las condiciones evaluadas. El número de veces que se ejecutan dependerá de la condición diseñada. Será posible combinar sentencias condicionales con bucles, consiguiendo, así, programas más complejos.

3.1. For

Un bucle de tipo for permite ejecutar un grupo de sentencias un número determinado de veces. En concreto, se fundamenta en una estructura que define el valor desde el que se va a comenzar a ejecutar hasta el valor en el que finaliza su ejecución.

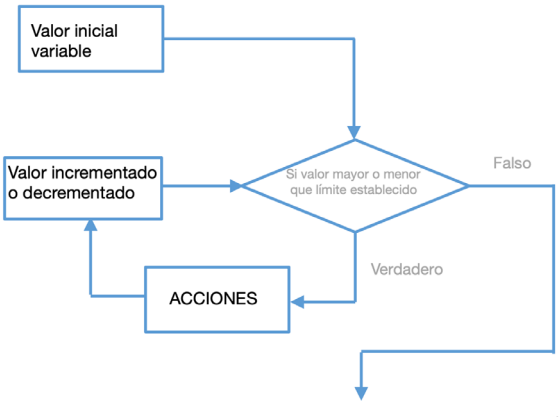

Fig. 6. Diagrama de funcionamiento for en PHP.

Para implementar este tipo de bucles, se utiliza la palabra **for**, seguida, entre paréntesis, del **valor inicial** en el que comienza la ejecución de las instrucciones que quedan contenidas entre llaves.

A continuación, se define el **valor final** hasta el que se realiza la acción reiterativa sobre este bucle. El último de los valores define cómo va a variar el valor inicial hasta llegar al final, y será posible tanto incrementar (**inc**) como decrementar (**dec**) su valor.

```
<?php
 for(valorInicial; valorFinal; inc/dec ){
            //acciones
 }
?>
```

Código 4. Bucle for en PHP.

En el siguiente ejemplo, se evalúa el valor de la variable desde que su valor es 0 hasta que alcanza el 10; para ello, incrementa su valor en 1 en cada iteración del código. Además, cuando el valor es mayor de 10, la ejecución del bloque de código se detiene.

```
for ($var1= 0; $var1<=10 ; $var1++) {
   if ($var1 > 10) {
      break;
   }
   echo $var1;
}
```

Código 5. Ejemplo completo con bucle for en PHP.

3.2. Bucles while y do-while

Los bucles **while** permiten ejecutar un mismo bloque de código, siempre que se cumpla la condición evaluada. Es la forma equivalente de decir, «mientras que una condición ocurra, se ejecutarán las instrucciones implementadas bajo este bloque». Este bucle puede no ejecutarse. En el caso de PHP, la sintaxis será:

```
<?php
 while(condición){
                //acciones
 }
?>
```

Código 6. Bucle while en PHP.

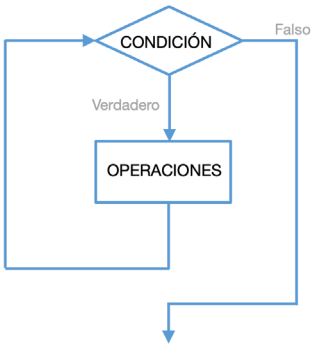

Fig. 7. Diagrama de funcionamiento while en PHP

Sin embargo, con do-while, a diferencia del anterior, al menos el conjunto de instrucciones implementadas dentro del bloque se ejecuta una vez, puesto que, en primer lugar, se ejecutan y, luego, se evalúa la condición; si ésta se cumple, se vuelve a ejecutar el mismo bloque. Y así de forma sucesiva hasta que la condición deja de cumplirse.

```
<?php
 do{
                //acciones
 }while(condición)
?>
```

Código 7. Bucle do-while en PHP.

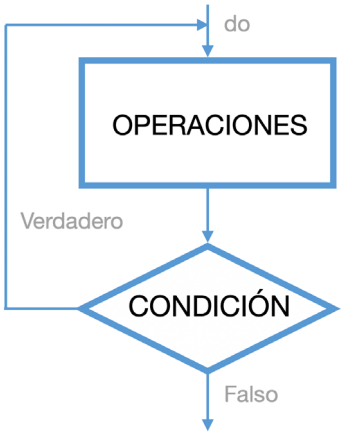

Fig. 8. Diagrama de funcionamiento do-while en PHP.

3.3. Bucle foreach

Este bucle se utiliza para realizar una iteración sobre los elementos de un array de implementación sencilla. De manera más exacta, un puntero avanza sobre cada una de las posiciones de un array, almacenado el valor en la variable definida.

```
<?php
  foreach(array as valor){
                //acciones
  }
?>
```

Código 8. Bucle FOREACH en PHP.

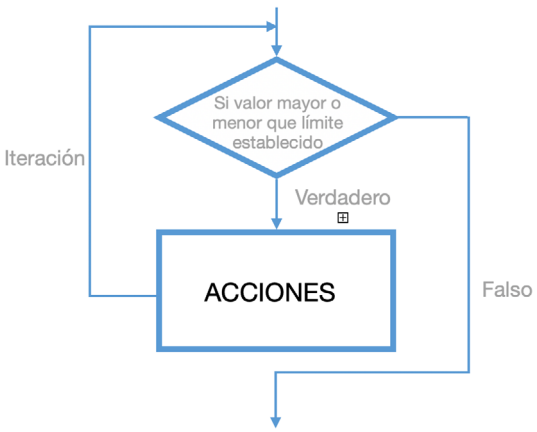

Fig. 9. Diagrama de funcionamiento FOREACH en PHP.

Tras el uso de este tipo de bucles, se aconseja utilizar la función unset(), que recibe como parámetro la variable donde se ha ido almacenando el valor del array en cada una de la iteraciones; de lo contrario, quedará almacenada la referencia a la última posición del array anterior.

Por ejemplo, en el siguiente código, se recorren cada una de las posiciones del array y se imprime su valor; finalmente, se elimina el valor de la variable utilizada como almacén auxiliar.

```
<?php
        $arrayValores = array(...);
        foreach ($arrayValores as &$valor) {
      print($valor);
        }
        unset($valor);
?>
```

Código 9. Ejemplo con bucle FOREACH en PHP.

Vídeo 1. "Uso de sentencias y bucles para arrays en PHP"
https://bit.ly/30EZhuO

/ 4. Otras sentencias en PHP

- **Break:** Como vimos, para el caso del condicional switch-case, existe una sentencia que permite detener la ejecución del bucle o estructura de control donde se encuentra implementada: break.

Este elemento permite salir de una estructura o de varias, es decir, si se coloca un break dentro de un if, su ejecución nos hará salirnos de este if, pero si, además, estamos situados dentro de otro bucle, por ejemplo, un while, no nos saldremos de este.

Para detener la ejecución de tantas estructuras anidadas circundantes como se desee, añadiremos un valor a continuación de la palabra break, que indica el número de sentencias o bucles que pararán su ejecución: break numeroDeEstructurasParaSalir;

- **Continue:** Esta sentencia permite «detener» la ejecución de las instrucciones bajo una determinada continuación y continuar la ejecución en la siguiente iteración de sentenciado condicional o bucle.

En concreto, se retoma en el fragmento encargado de evaluar de nuevo la condición para determinar si esta se cumple, y, seguidamente, ejecutar el contenido del bloque, si fuera el caso.

La sentencia en sí es la propia palabra 'continue': continue.

- **Otras sintaxis:** En algunos casos, es posible encontrar una sintaxis en PHP algo distinta a la implementada hasta el momento. La diferencia se basa en la sustitución de las llaves por el símbolo ":"; en concreto, de este modo, se sustituye a la llave de apertura del bloque.

Las llaves de cierre también cambian. En este caso, se utilizará la palabra endif, endwhile... en función del tipo de estructura.

```
<?php
        $arrayValores = array(a, b, c, d);
        foreach ($arrayValores as &$valor):
      print($valor);
        endforeach;
        unset($valor);
?>
```

Código 10. Ejemplo con otras sintaxis PHP.

Este tipo de sintaxis puede utilizarse en las estructuras de control: if (endif), if-else (endif), for (endfor), foreach (endforeach), while (endwhile) o switch (endswitch).

/ 5. Arrays

Este tipo de estructuras, muy común en todos los lenguajes de programación, permite organizar un conjunto de elementos en base a una clave de posición determinada. Los arrays sirven para almacenar varios datos del mismo tipo, por ejemplo, varios números enteros, varios objetos de una determinada clase...

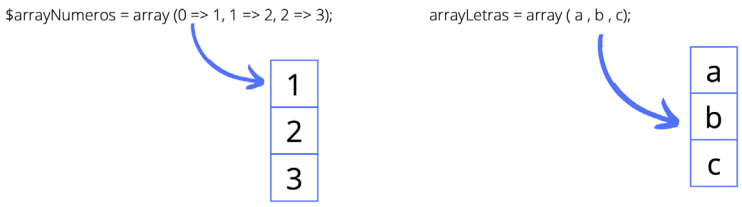

Fig. 10. Representación de arrays

Un array puede crearse de diferentes formas, en función de la implementación que se quiera llevar a cabo.

Sintaxis 1:

```
nombreArray = array ( posición1 => valor1, posición2 => valor2,...);
```

Código 11. Sintaxis 1 creación de arrays con PHP.

En este tipo se sintaxis, se indica la posición o la clave exacta en la que se van a almacenar cada uno de los valores. El nombre del array se define al inicio de la instrucción. Para crear un objeto de este tipo, se utiliza la función array() seguida de los argumentos que indican la posición y el contenido.

```
$arrayNumeros = array (“pos1”=> 1, “pos2”=> 2, “pos3”=> 3);
```

Código 12. Ejemplo creación de arrays con PHP.

Sintaxis 2:

```
nombreArray = array ( valor1, valor2,...);
```

Código 13. Sintaxis 2 creación de arrays con PHP

Esta segunda sintaxis es una adaptación de la primera; en este caso, no se indican las posiciones en las que se va a colocar cada valor, sino que se hace de manera consecutiva, desde la posición 0.

```
$arrayNumeros = array ( 1, 2, 3);
```

Código 14. Ejemplo creación de arrays con PHP

La definición de las claves de la primera sintaxis debe realizarse utilizando String o Integer, y son equivalentes expresiones tales como "1" y 1.

Hasta ahora, hemos visto cómo crear un array completo, pero, si tras la creación se desea seguir añadiendo más elementos, en concreto, en la posición siguiente al último almacenado, utilizaremos la siguiente instrucción: $nombreArray[]= valor;.

5.1. Operaciones con arrays

Acceso a los elementos del array

Para acceder a cualquiera de las posiciones de un array y obtener su valor para imprimirlo por pantalla, modificar el contenido, o llevar a cabo cualquier otra acción, se indica la posición exacta entre corchetes [ ].

```
$nombreArray[posicion];
```

Código 15. Acceso a una posición-clave de un array con PHP.

Eliminar los elementos del array

Para borrar un array de forma completa, deberemos eliminar el array entero o borrar solo uno de los elementos.

Borrar todo el array:

```
unset ($nombreArray);
```

Código 16. Borrado de todo el array en PHP.

Borrar un elemento del array:

```
unset ($nombreArray[posición]);
```

Código 17. Borrado de un solo elemento del array en PHP.

Otras funciones

Dada la presencia de este tipo de elementos compuestos, se desarrollan diferentes funciones que solo requieren de ser llamadas para poder utilizarlas dentro de cualquier nuevo proyecto:

- **Unión de arrays.** Se utiliza para concatenar un array con otro. El segundo se sitúa al final del primero y así de forma consecutiva.

```
array array_merge (array $array1, ...array $arrayN)
```

Código 18. Concatenación de arrays en PHP.

- **División de arrays.** Se permite obtener varios arrays, haciendo fragmentos más pequeños de otro array pasado por parámetro. También se le indica la posición desde la que va a comenzar la división (offset). El parámetro length determina la longitud del corte desde la posición marcada por offset.

```
array array_slice (array $array1, int offset, int length);
```

Código 19. División de arrays en PHP.

- **Ordenar arrays.** Se produce la clasificación de un array en base a diferentes criterios:

```
sort($nombreArray);// ordena de menor a mayor
rsort($nombreArray);// ordena de mayor a menor
```

Código 20. Ordenación contenido de arrays en PHP.

Vídeo 2. "Operaciones con arrays"
https://bit.ly/3fLIVoi

/ 6. Matrices

Las matrices o arrays bidimensionales permiten crear arrays de más de una dimensión, simulando estructuras similares a una matriz.

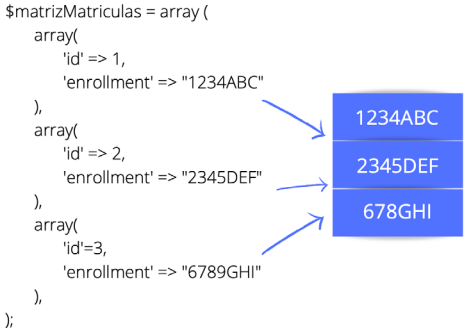

Fig. 11. Representación de matrices.

Para crear una matriz, se indica en cada posición del array el nuevo array que se almacenará en dicha posición. Dicho de otra forma, una matriz es un array de arrays, es decir, un array que, en cada nueva posición, almacena un nuevo array simulando, como resultado, una estructura similar a una matriz.

```
nombreMatriz = array [
                [array1], [array2], ... [arrayN]
]
```

Código 21. Implementación de matrices en PHP.

Una de las implementaciones posibles se basa en la creación del array principal. Dentro de este, vamos a ir colocando cada uno de los nuevos arrays. Para identificar cada posición, seguiremos usando una «clave» que apunta "=>" al array contenido.

```
"Clave" => array ( ... )
```

Código 22. Asociación posición – clave.

Por lo tanto, la sintaxis final quedaría de la siguiente forma:

```
$matriz = array (
                "Clave" => array ( ... ),
                "Clave" => array ( ... ),
                "Clave" => array ( ... )
);
```

Código 23. Implementación completa de matrices en PHP.

6.1. Acceso a las posiciones de la matriz

De forma similar a los arrays, para acceder a cada una de las posiciones de una matriz, indicamos, entre corchetes, la posición de la que vamos a obtener el valor. Esta posición puede ser 0-n, si no se la asociado ninguna clave a las posiciones del array, o utilizar la clave de posición.

La diferencia principal reside en que, ahora, serán necesarias dos claves o posiciones.

```
$nombreMatriz[posicion1][posicion2];
```

Código 24. Acceso a una posición-clave de una matriz con PHP.

En el siguiente ejemplo, se crea un nuevo array, que se modelará de tal forma que se genera una matriz. En este caso, se indica la clave identificadora para cada una de las posiciones; en caso contrario, se consideraría la clave desde el valor 0 en adelante.

```
<?php
$matriz = array (
      “Pos1” => array(“ 1 “, “ 2 “, “ 3 “),
      “Pos2” => array(“ 4 “, “ 5 “, “ 6 “),
      “Pos3” => array(“ 7 “, “ 8 “, “ 9 “)
  );
                  print($matriz [“Pos1”][0]);
                  print($matriz[“Pos1”][1]);
                  print($matriz[“Pos1”][2]);
                  print(“</br>”);
                  print($matriz[“Pos2”][0]);
                  print($matriz[“Pos2”][1]);
                  print($matriz[“Pos2”][2]);
                  print(“</br>”);
                  print($matriz[“Pos3”][0]);
                  print($matriz[“Pos3”][1]);
                  print($matriz[“Pos3”][2]);
?>
```

Código 25. Ejemplo de implementación de matrices en PHP.

En la siguiente imagen, podemos ver el resultado anterior, donde queda «simulada» una matriz.

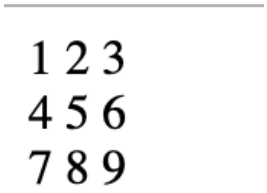

Fig. 12. Representación de matrices de ejemplo.

/ 7. Caso práctico 1: “Bucles en PHP”

Planteamiento: Para comenzar a practicar con las sentencias condicionales y los bucles vamos a realizar un pequeño programa en PHP que devuelva todos los valores comprendidos entre un número mínimo y otro máximo.

Nudo: Para implementar este programa, podemos utilizar un bucle de tipo while o uno de tipo for. El primero evaluará el valor de la variable mínima hasta que esta se iguale con la cifra máxima. Mientras que no supere la cifra, imprimirá el nuevo valor y actualizará el valor mínimo. Además, se va a imprimir un número en cada línea.

```
<?php
   $valorMinimo=1;
   $valorMaximo=10;
while ( $valorMinimo <= $valorMaximo )
{
                echo $valorMinimo . "</br>";
                $valorMinimo++;
     }
?>
```

Código 26. Código del Caso Práctico 1 con while.

También se podrá utilizar un bucle for con el mismo resultado:

```
<?php
   $valorMinimo=1;
   $valorMaximo=10;
for( $var = $valorMinimo; $var <= $valorMaximo; $var++ ){
echo $var. "</br>";
}
?>
```

Código 27. Código del Caso Práctico 1 con for.

Desenlace: En la barra del navegador, accedemos a la URL del servidor en la que se haya almacenado el fichero.

1
2
3
4
5
6
7
8
9
10

Fig. 13. Solución al código 26 y 27 del Caso Práctico 1.

/ 8. Caso práctico 2: “Creación de matrices en PHP”

Planteamiento: Los arrays y las matrices son datos compuestos que serán muy útiles para el desarrollo de aplicaciones web en entorno servidor. Por ejemplo, los datos relativos a las sesiones que se establecen para la autenticación de usuarios en un determinado sitio web, se almacenan en un array.

En este ejercicio vamos a construir una matriz donde se almacenan, en cada posición, los siguientes datos de un usuario:

- Login
- Pass
- Perfil

Crea una matriz con, al menos, 4 usuarios y todos los datos solicitados. Luego hay que imprimir de dos a tres posiciones de la matriz.

Nudo: Para almacenar la información de cada usuario en la matriz, utilizaremos como clave el nombre de usuario. De esta forma, cuando se extraiga la información para imprimirla, bastará con identificar, a través de la clave, el array de usuario, y luego la posición del array concreta a imprimir.

```
<?php
$datos = array (
“User1” => array(“User1”, “Pass1”, “admin”),
“User2” => array(“User2”, “Pass2”, “user”),
“User3” => array(“User3”, “Pass3”, “admin”)
  );
echo “El usuario 1 tiene perfil: “ . $datos[“User1”][2] . “</br>”;
echo “El usuario 2 tiene perfil: “ . $datos[“User2”][2] . “</br>”;
echo “El usuario 3 tiene perfil: “ . $datos[“User3”][2] . “</br>”;
?>
```

Código 28. Código del Caso Práctico 2.

Desenlace

El usuario 1 tiene perfil: admin
El usuario 2 tiene perfil: user
El usuario 3 tiene perfil: admin

Fig.14. Solución al código 28 del Caso Práctico 2.

/ 9. Comentarios

Mientras que en HTML la inclusión de comentarios se delimita utilizando <!-- -->, en PHP esto no tienen ninguna validez. Se permite el uso de tres tipos de delimitadores, algunos de los cuales coinciden con los empleados en lenguajes de programación como Java:

- **/* */.** Estos permiten insertar, entre los valores de apertura y cierre, todas las líneas de comentarios que sean necesarias. Este tipo de delimitadores son útiles cuando existen varias líneas de comentarios o cuando se quiere comentar algún fragmento de código.

```
  /* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  */
```

- **#.** Se utiliza para aquellos casos en los que el comentario no tiene una extensión superior a una línea. En concreto, se utilizan a continuación de una línea de código, para realizar alguna especificación sobre el código que le precede.

```
$var1="hola" #se crea una variable
```

- **//.** Al igual que el caso anterior, este tipo de delimitadores se utiliza, habitualmente, para comentar una línea completa.

El uso de comentarios en programación no es un hecho banal, sino que resulta clave para construir programas que resulten más fáciles de mantener en el futuro.

```
<head>
  <title>PHP con comentarios</title>
 </head>
 <body>
      <?php
        /*Este es el primer ejemplo con
        comentarios en PHP*/
        //echo lanza una ventana
        echo "Hola Mundo!!!"; #con mensaje
      ?>
</body>
</html>
```

Código 29. Ejemplo de uso comentarios con PHP.

Audio 1. "La importancia de los comentarios"

https://bit.ly/31DhpEn

/ 10. Resumen y resolución del caso práctico de la unidad

A lo largo del tema, hemos visto todas las sentencias que se pueden utilizar para la implementación en lenguaje PHP. La combinación de las sentencias lineales, las condicionales y los bucles permite modelar el comportamiento de cualquier algoritmo de programación.

Las sentencias lineales constituyen las principales líneas de acción en el desarrollo de programas: permiten la creación y asignación de variables, llamadas a funciones o la instalación de objetos. Recuerda que estas pueden agruparse en los bloques utilizando las llaves, { }.

Diferenciamos sentencias condicionales de bucles, porque las primeras se ejecutan una sola vez, comprobando si se cumple una función o no y, en función de esto, se ejecuta un bloque de programación u otro. Estas son: if, if-else y switch.

Por otro lado, los bucles son sentencias de tipo reiterativo. ¿Qué quiere decir esto? Significa que estas van a ejecutar el contenido del bloque que preceden tantas veces como se indique en la declaración del bucle (por ejemplo, for) o siempre que se cumpla (o no) una condición (como ocurre con while).

Finalmente, los arrays y las matrices son tipos de datos compuestos que agilizan el desarrollo con cualquier lenguaje de programación, cuyas funciones y formas de uso se utilizarán a lo largo de este módulo.

Resolución del caso práctico de la unidad.

Como se ha visto a lo largo del tema, las sentencias condicionales o estructuras de control nos permitirán dar respuesta a la pregunta planteada al inicio del tema. Si se realiza una pregunta a un usuario y, en función de la respuesta de este, la salida cambia, será posible utilizar alguna estructura de tipo if-else. Si el valor es correcto, se ejecutaría todo lo ontenido en el bloque implementado tras las llaves del if; de lo contrario, se ejecutará lo implementado tras las llaves del else.

El uso del if-else sería una de las opciones más recomendadas en este caso tan sencillo, pero, por ejemplo, también se podría utilizar un bucle de tipo do-while, el cuál lanzará la pregunta tantas veces como sea necesario, hasta que el usuario introduzca el valor correcto que se evaluará en la condición del while.

/ 11. Bibliografía

Ganzabal, X. (2019). Desarrollo web en entorno servidor. (1.a. ed.). Madrid: Síntesis.

Vara, J. M. (2012). Desarrollo en entorno Servidor. (1.a. ed.). Madrid: Rama.

/ 12. Webgrafía

PHP. Recuperado de: https://www.php.net/manual/es/intro-whatis.php