



Universidad Don Bosco

FACULTAD DE INGENIERÍA

ESCUELA DE COMPUTACIÓN

Lenguaje Interpretado en el Servidor

Guía 2: Estructuras de Control en PHP



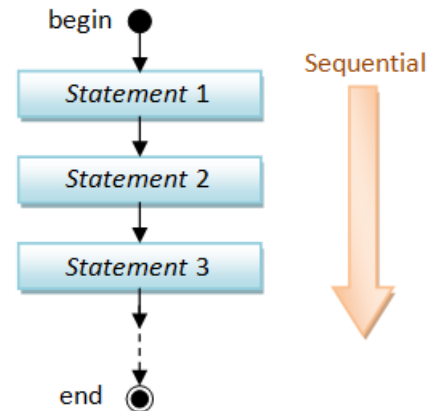
I. OBJETIVOS

En esta guía de práctica se espera que los estudiantes logren:

- Adquirir amplio dominio en el uso de las sentencias condicionales *if* y *switch*.
- Conseguir un uso correcto del operador condicional *?:*
- Adquirir un amplio dominio de la sintaxis de cada una de las sentencias repetitivas disponibles en PHP.
- Hacer uso de sentencias para modificar el número de veces y la forma en qué se ejecuta un ciclo o lazo.

II. INTRODUCCION TEORICA

Los primeros ejemplos de programación que suelen utilizarse están compuestos por una serie de instrucciones que se ejecutan de forma secuencial. A medida que se va avanzando en un curso uno cae en la cuenta que lo más habitual es que los problemas que se nos plantean casi nunca se pueden resolver con una ejecución secuencial. Es importante dominar y conocer cierto tipo de instrucciones que permitan modificar el flujo de ejecución de las instrucciones. Estas instrucciones permitirán controlar el flujo lógico en que se ejecutarán las instrucciones dentro de un script. Específicamente, podremos utilizar estructuras de control condicionales que nos permitirán bifurcar el flujo del programa dependiendo de si una condición se evalúa como verdadera o falsa, o repetir un bloque de instrucciones un número determinado de veces para realizar de forma más eficiente y elegante un proceso.



Las estructuras de control se dividen en sentencias condicionales, selectivas y repetitivas. Vamos a examinar todos estos tipos de instrucciones durante este curso. En esta guía de práctica trabajaremos con las primeras dos, dejando las últimas para la siguiente guía de práctica.

Se podrían clasificar las sentencias condicionales en dos:

- Sentencias condicionales *if*.**
 - Sentencia *if* simple.
 - Sentencia *if-else*.
 - Sentencia *if-elseif-else*.
- Sentencia selectiva *switch*.**

Sentencia *if*

La sentencia *if* permite evaluar una expresión condicional y decidir, en base al resultado, si se ejecuta o no un bloque de código. Si la evaluación resulta verdadera (*true*) se ejecuta el bloque de instrucciones y si resulta falsa (*false*) no se ejecuta.

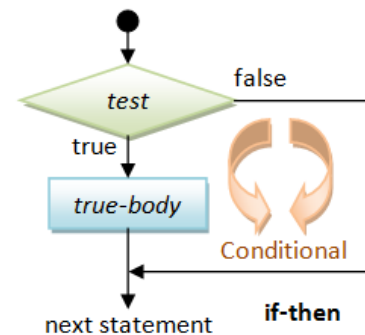
PHP proporciona una estructura *if* que es similar a la de C:

Sintaxis básica:

```
<?php
    if(expresion) //sentencia;
?>
```

Sintaxis extendida:

```
<?php
    if(expresion){
        //bloque de sentencias;
    }
?>
```



La sintaxis básica muestra la construcción de una sentencia *if* cuando esta contiene una sola instrucción. La sintaxis extendida muestra una sentencia *if* con varias instrucciones dentro del bloque que encierran las llaves del *if*.

Como se describe en la sección sobre expresiones, expresión se evalúa a su valor condicional (*boolean*). Si expresión se evalúa como TRUE, PHP ejecutará la sentencia, y si se evalúa como FALSE, la ignorará y continuará con la siguiente instrucción después del *if*.

El siguiente ejemplo mostraría: \$a es mayor que \$b, si \$a fuera mayor que \$b. De no ser así no mostraría nada:

```
<?php
    if($a > $b) print "$a es mayor que $b";
?>
```

A menudo, se desea tener más de una sentencia ejecutada de forma condicional. Por supuesto, no hay necesidad de encerrar cada sentencia con una cláusula *if*. En vez de eso, se pueden agrupar varias sentencias en un grupo de sentencias. Por ejemplo, este código mostraría \$a es mayor que \$b, si \$a fuera mayor que \$b, y entonces asignaría el valor de \$a a \$b:

```
<?php
    if($a > $b) {
        print "$a es mayor que $b";
        $b = $a;
    }
?>
```

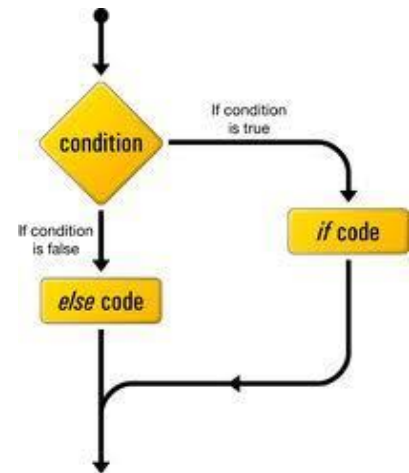
Las sentencias *if* se pueden anidar indefinidamente dentro de otras sentencias *if* o *else*, lo cual proporciona una flexibilidad completa para ejecuciones condicionales en las diferentes partes de tu programa.

Sentencia if-else

Hay situaciones en las que se desea ejecutar una o varias sentencias si se cumple una cierta condición, y ejecutar un bloque de sentencias distintas si la condición no se cumple. En estos casos es útil la sentencia *if ...else*. De esta forma se extiende una sentencia *if* para ejecutar una o varias sentencias en caso de que la expresión en la sentencia *if* se evalúe como FALSE. Por ejemplo, el siguiente código mostraría \$a (su valor) es

impar si $\$a \% 2$ resultara evaluado como 1 o $\$a$ es par si fuera evaluado como 0. Note que solamente se pueden obtener esos valores como resultado:

```
<?php
    if($a % 2){
        print "$a es impar";
    }
    else{
        print "$a es par";
    }
?>
```



La sentencia **else** se ejecuta solamente si la expresión **if** se evalúa como FALSE, y si hubiera alguna expresión **elseif** - sólo si se evaluara también como FALSE.

Sentencia if-elseif-else

La sentencia **elseif** es una combinación de **if** y **else**. Al igual que un **else**, extiende una sentencia **if** para ejecutar una sentencia diferente en caso de que la expresión **if** original se evalúa como FALSE. No obstante, a diferencia del **else**, ejecutará esa expresión alternativa solamente si la expresión condicional **elseif** se evalúa como TRUE. Un bloque **if** puede contener múltiples **elseif**, pero únicamente un **else**. El bloque de instrucciones bajo un **elseif** se ejecutará, únicamente si la condición **if** y todas las instrucciones **elseif** anteriores devuelven FALSE.

Por ejemplo, el siguiente código mostraría a es mayor que b, a es igual a b o a es menor que b:

```
<?php
    if($a > $b){
        echo "$a es mayor que $b";
    }
    elseif($a == $b){
        echo "$a es igual que $b";
    }
    else{
        echo "$a es mayor que $b";
    }
?>
```

Puede haber varios **elseif** dentro de la misma sentencia **if**. La primera expresión **elseif** (si hay alguna) que se evalúe como TRUE se ejecutaría. En PHP, también se puede escribir 'else if' (con dos palabras) y el comportamiento sería idéntico al de un 'elseif' (una sola palabra).

La sentencia **elseif** se ejecuta sólo si la expresión **if** precedente y cualquier otra expresión **elseif** precedente se evalúan como FALSE, y la expresión **elseif** actual se evalúa como TRUE.

Sintaxis alternativa de la sentencia condicional

PHP ofrece una forma alternativa de construcción de una sentencia condicional. Esta forma alternativa reemplaza el uso de la llave de apertura del bloque **if** por dos puntos (:) y la llave de cierre del bloque por la sentencia **endif**.

Ejemplo:

```
<?php
    if($num % 2 == 0):
        echo $num . " es par";
    else:
        echo $num . " es impar";
?>
```

Operador condicional

En PHP se puede utilizar un operador ternario utilizado con frecuencia en el lenguaje C++. Este es el operador "?", o también llamado **operador condicional**. Este es un operador ternario que se utiliza para simplificar la escritura de una sentencia condicional de tipo if...else, cuando esta contiene únicamente una sentencia en el bloque if y una sentencia en el bloque else.

La sintaxis de este operador es la siguiente:

```
(expresion_condicional) ? expresion1 : expresion2 ;
```

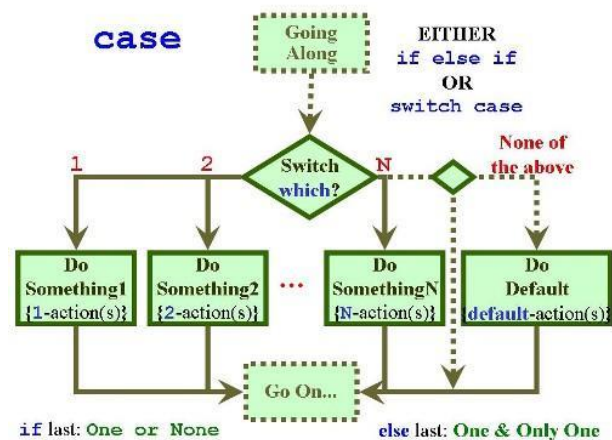
Debe tener en cuenta que si la expresión condicional se evalúa como **TRUE**, se ejecutará expresion1 y si se evalúa **FALSE**, se ejecutará expresion2. Como las expresiones devuelven un valor, usted puede perfectamente asignar la instrucción completa en una variable, como se muestra en el siguiente ejemplo:

```
$condicion = ($nota >= 6.0) ? "Aprobado" : "Reprobado" ;
```

Si la expresión condicional resulta verdadera la variable \$condicion tendrá el valor de "Aprobado", si se evalúa como falsa, entonces el valor asignado a \$condicion será "Reprobado".

Sentencia switch

La sentencia **switch** es similar a una sentencia **if** con varias instrucciones **elseif**. La diferencia está en que en un **if** la condición se evalúa tanto por la sentencia **if** como por los **elseif** que esta contenga. En cambio, en un **switch**, la condición se evalúa solamente una vez, luego el valor obtenido se compara con una serie de valores de prueba predefinidos. Cada uno de estos valores se presenta con una sentencia **case** dentro del **switch**. Cuando uno de estos valores coincide con el evaluado por la condición se ejecuta el bloque de instrucciones dentro de ese case de forma secuencial. En caso de que ninguno de los valores de prueba coincida con el devuelto por la condición se ejecutará el bloque de instrucciones bajo la sentencia **default**, si es que está presente, si no lo está, entonces no se ejecutará ninguna instrucción en el **switch**.



Nota: Tener en cuenta que al contrario que otros lenguajes de programación, **continue** se aplica a **switch** y funciona de manera similar a **break**. Si se tiene un **switch** dentro de un bucle y se quiere continuar con el paso siguiente en el bucle externo, usar **continue 2**.

Los siguientes dos ejemplos son dos modos distintos de escribir el mismo programa, uno usa una serie de sentencias **if**, y el otro usa la sentencia **switch**:

Utilizando if:

```
<?php
    if ($i == 0) {
        echo "$i es igual a 0";
    }
    elseif ($i == 1) {
        echo "$i es igual a 1";
    }
    elseif ($i == 2) {
        echo "$i es igual 2";
    }
?>
```

Utilizando switch:

```
<?php
    switch ($i) {
        case 0:
            print "i equals 0";
            break;
        case 1:
            print "i equals 1";
            break;
        case 2:
            print "i equals 2";
            break;
    }
?>
```

Es importante entender cómo se ejecuta la sentencia **switch** para evitar errores. La sentencia **switch** se ejecuta línea por línea (realmente, sentencia a sentencia). Al comienzo, no se ejecuta código. Sólo cuando se encuentra una sentencia **case** con un valor que coincide con el valor obtenido en la evaluación de la expresión **switch**, PHP comienza a ejecutar las sentencias. PHP continúa ejecutando las sentencias hasta el final del bloque switch, o la primera vez que se encuentre una sentencia **break**. Si no se escribe una sentencia break al final de una lista de sentencias case, PHP seguirá ejecutando las sentencias del siguiente **case**. Por ejemplo:

```
<?php
    switch ($i) {
        case 0:
            print "i es igual a 0";
        case 1:
            print "i es igual a 1";
        case 2:
            print "i es igual a 2";
    } ?>
```

Aquí, si \$i es igual a 0, ¡PHP ejecutaría todas las sentencias print! Si \$i es igual a 1, PHP ejecutaría las últimas dos sentencias print y sólo si \$i es igual a 2, se obtendría la conducta 'esperada' y solamente se mostraría 'i es igual a 2'. Así, es importante no olvidar las sentencias break (incluso aunque pueda querer evitar escribirlas intencionadamente en ciertas circunstancias).

En una sentencia switch, la condición se evalúa sólo una vez y el resultado se compara a cada sentencia case. En una sentencia elseif, la condición se evalúa otra vez. Si tu condición es más complicada que una comparación simple y/o está en un bucle estrecho, un switch puede ser más rápido.

La lista de sentencias de un case puede también estar vacía, lo cual simplemente pasa el control a la lista de sentencias del siguiente case.

```
<?php
switch ($i) {
    case 0:
    case 1:
    case 2:
        print "i es menor que 3, pero no negativo";
        break;
    case 3:
        print "i es 3";
}
?>
```

Un caso especial es el default. Este "case" coincide con todo lo que no coincidan los otros case. Por ejemplo:

```
<?php
switch ($i) {
    case 0:
        print "i es igual a 0";
        break;
    case 1:
        print "i es igual a 1";
        break;
    case 2:
        print "i es igual a 2";
        break;
    default:
        print "i no es igual a 0, 1 o 2";
}
?>
```

La expresión case puede ser cualquier expresión que se evalúe a un tipo simple, es decir, números enteros o de punto flotante y cadenas de texto. No se pueden usar aquí ni arrays ni objetos a menos que se conviertan a un tipo simple.

break

break escapa de las estructuras de control iterante (bucle) actual *for*, *while*, o *switch*.

break acepta un parámetro opcional, el cual determina cuántas estructuras de control hay que escapar.

Sintaxis alternativa de la sentencia switch

También la sentencia **switch** puede utilizarse con una sintaxis alternativa de dos puntos. La forma de construir una sentencia *switch* con sintaxis alternativa, se muestra a continuación:

```
<?php
switch($i):
    case 0:
        echo "$i es igual a 0";
        break;
    case 1:
        echo "$i es igual a 1";
        break;
    case 2:
```

```
        echo "$i es igual a 2";
        break;
    default:
        echo "$i no es igual a 0, 1 ni 2";
endswitch;
?>
```

SENTENCIAS REPETITIVAS

Las sentencias repetitivas son un mecanismo proporcionado por el lenguaje PHP para poder repetir varias veces un bloque de instrucciones. La utilidad que se le da a este tipo de sentencias en programas específicos es la realización de conteos, realizar acumulación de valores en una variable, concatenar cadenas, construir los elementos de listas o tablas HTML, recorrer los elementos de un arreglo o de un objeto, etc.

PHP soporta cuatro tipos de sentencias repetitivas que son las mismas utilizadas por varios lenguajes de programación. Estos son: ***while***, ***do-while***, ***for*** y ***foreach***.

WHILE

Los bucles, lazos o ciclos *while* son los tipos de bucles más simples en PHP. Se comportan exactamente como en la mayoría de lenguajes de programación, tales como C/C++, Java, etc. La forma básica de una sentencia *while* es la siguiente:

```
while (expresion){[}
    [sentencia(s);]
}]
```

Donde:

- **expresion**, es una expresión condicional que debe devolver un resultado lógico: *true* o *false* (verdadero o falso), y
- **sentencias**, es una instrucción o un bloque de instrucciones que se repetirán en tanto la expresión condicional se siga evaluando como verdadera.

NOTA: Cuando el ciclo o lazo *while* incluya una sola sentencia se pueden omitir las llaves. Sin embargo, es recomendable utilizarlas siempre.

El funcionamiento de una sentencia *while* es simple. Le dice a PHP que ejecute la(s) sentencia(s) que contiene repetidamente, mientras la expresión condicional del *while* se evalúe como verdadera (***true***). El valor de la expresión es comprobado cada vez al principio del bucle, así que incluso si este valor cambia durante la ejecución de la(s) sentencia(s) anidada(s), la ejecución no parará hasta el fin de la iteración (cada vez que PHP ejecuta las sentencias en el ciclo o bucle se da una iteración). Si la expresión *while* se evalúa como ***false*** desde el principio la(s) sentencia(s) anidada(s) no se ejecutarán ni siquiera una vez.

Al igual que con la sentencia *if*, existe una sintaxis alternativa para el ciclo o lazo *while*:

```
while (expresion):
    [sentencia(s);]
endwhile;
```

Los siguientes ejemplos son idénticos, y ambos imprimen números del 1 al 10:

/* ejemplo 1 */

```
$i = 1;
while ($i <= 10) {
    print $i++;
    /* el valor impreso sería
    $i antes del incremento
    (post-incremento) */
}
```

/* ejemplo 2 */

```
$i = 1;
while ($i <= 10):
    print $i;
    $i++;
endwhile;
```

DO ... WHILE

Los ciclos o bucles **do ... while** son muy similares a los bucles **while**, excepto que la condición se comprueba al final de cada iteración, en vez de al principio. La principal diferencia frente a los bucles regulares **while** es que en este, se garantiza la ejecución de la primera iteración de un bucle **do ... while** (la condición se comprueba sólo al final de la iteración), mientras que puede no ser necesariamente ejecutada con un bucle **while** regular (la condición se comprueba al principio de cada iteración, si esta se evalúa como **false** desde el principio la ejecución del bucle finalizará inmediatamente).

Hay una sola sintaxis para los bucles **do...while**:

```
do{
    sentencias;
} while (expresion);
```

Donde:

sentencias es el bloque de código a ejecutar un número indefinido de veces, dependiendo del resultado de la evaluación de la condición, y

expresion, es la expresión condicional que debe evaluarse para determinar si se seguirá ejecutando el bloque de sentencias del lazo.

El siguiente es un ejemplo:

```
$i = 0;
do {
    print $i;
} while ($i>0);
```

NOTA: Cuando el bucle **do-while** incluya una sola sentencia se pueden omitir las llaves. A pesar de ello es recomendable utilizarlas siempre.

FOR

Los ciclos o bucles **for** son los bucles más complejos en PHP por la construcción; sin embargo, son también los más utilizados porque su aplicación es más evidente. Se comportan exactamente igual que en lenguajes como C o Java. La sintaxis de un lazo **for** es:

```
for ([inicialización]; [condición]; [incremento]){  
    sentencia;  
}
```

Donde:

inicialización, es una expresión que inicializa una variable que funcionará como contador. Este valor representa el valor inicial de dicho contador a partir del cual comenzarán las iteraciones del lazo o bucle.

condición, es una expresión condicional que se evaluará al inicio antes de ejecutar el bloque de sentencias. Si la evaluación de esta condición da por resultado *true*, el bloque de instrucciones se ejecuta, y si el resultado es *false* el bloque de instrucciones no se ejecuta.

incremento, es una sentencia que permite incrementar o decrementar el valor de la variable contador definida en la inicialización. El incremento|decremento puede ser de una unidad (caso más frecuente) o de varias unidades.

NOTA: Cuando el **for** incluya una sola sentencia se pueden omitir las llaves. Sin embargo, es recomendable utilizarlas siempre.

La primera expresión (**inicialización**) se evalúa (ejecuta) incondicionalmente una vez al principio del bucle y solamente esa vez.

Al comienzo de cada iteración, se evalúa la **condición**. Si el resultado es *true*, el ciclo o lazo continúa y las sentencias anidadas se ejecutan. Si se evalúa como *false*, la ejecución del ciclo o lazo finaliza.

Justo, al final de cada iteración, se ejecuta la instrucción de **incremento|decremento**.

Cada una de las expresiones puede estar vacía. Que *expresion2* esté vacía significa que el bucle debería correr indefinidamente (PHP implícitamente lo considera como *true*, al igual que C). Esto puede que no sea tan inútil como se podría pensar, puesto que a menudo se quiere salir de un bucle usando una sentencia [*break*](#) condicional en vez de usar la condición de **for**.

Existe una sintaxis alternativa para la sentencia repetitiva **for**. Esta se muestra a continuación:

```
for([inicialización]; [condición]; [incremento]):  
    sentencia(s);  
endfor;
```

Considera los siguientes ejemplos. Todos ellos muestran números del 1 al 10:

```
<?php
```

```
/* ejemplo 1 */
```

```
for ($i = 1; $i <= 10; $i++) {  
    print $i;  
}
```

/* ejemplo 2 */

```
for ($i = 1; ;$i++) {  
    if ($i > 10) {  
        break;  
    }  
    print $i;  
}
```

/* ejemplo 3 */

```
$i = 1;  
for (;;) {  
    if ($i > 10) {  
        break;  
    }  
    print $i;  
    $i++;  
}
```

/* ejemplo 4 */

```
for ($i = 1; $i <= 10; print $i, $i++) ;
```

/* Ejemplo 5 */

```
for($i = 1; $i <= 10; $i++):  
    print $i;  
endfor;
```

?>

Por supuesto, el primer ejemplo parece ser el más elegante (o quizás el cuarto), pero uno puede descubrir que ser capaz de usar expresiones vacías en ciclos o lazos **for** resulta útil en algunas ocasiones.

INSTRUCCIONES BREAK Y CONTINUE.

Estas sentencias se utilizan dentro de los lazos, ciclos o bucles para provocar la terminación completa del lazo o el avance a la siguiente iteración del mismo.

Al utilizar la sentencia *break*, debe tener en cuenta que para que se ejecute debe producirse una situación en especial que amerite la terminación del ciclo o lazo. Para ello se puede utilizar una sentencia condicional, como se puede observar en el siguiente ejemplo:

/* ejemplo sentencia break */

```
$i = 1;  
while ($i <= 10) {  
    if($i%2 == 0){  
        print "$i es par";  
    }  
    $i++;  
}
```

```

}
// Si el valor de $i es exactamente 5 el ciclo o lazo se terminará
elseif($i == 5) {
    break;
}
else {
    print "$i es impar";
}
}

```

El comportamiento de la sentencia *continue* es ligeramente distinto, ya que, en lugar de terminar por completo el ciclo o lazo, finaliza la ejecución de la iteración actual, pero continúa la ejecución del ciclo con la iteración siguiente, como se ilustra en el siguiente ejemplo:

```

/* ejemplo sentencia continue */
$i = 1;
while ($i <= 10) {
    if($i%2 == 0){
        print "$i es par";
    }
    //Si el valor de $i es exactamente 5 el ciclo o lazo avanzará a la siguiente iteración
    elseif($i == 5) {
        continue;
    }
    else {
        print "$i es impar";
    }
}

```

FOREACH

PHP4 y PHP5 incluyen la instrucción ***foreach***, tal como Perl y algunos otros lenguajes. Esta instrucción permite recorrer los elementos de una matriz de una forma sencilla. La instrucción permite obtener cada uno de los valores almacenados en el arreglo y asignarlos automáticamente en una variable para trabajar con ellos dentro del bloque de instrucciones del ***foreach***. También es posible obtener el índice si acaso, se ha utilizado una matriz asociativa. Hay dos sintaxis; la segunda es una extensión menor, pero útil de la primera:

***foreach*(expresión_array as \$value) //sentencias;**

***foreach*(expresión_array as \$key => \$value) //sentencias;**

La primera forma recorre el array dado por ***expresión_array***. En cada iteración, el valor del elemento actual se asigna a ***\$value*** y el puntero interno del array se avanza en una unidad (así en la siguiente iteración, se estará apuntando de forma automática el elemento siguiente).

La segunda manera hace lo mismo, salvo que la clave del elemento actual será asignada a la variable ***\$key*** en cada iteración.

Notas:

- Cuando ***foreach*** comienza su primera ejecución, el puntero interno a la lista (*array*) se reinicia automáticamente al primer elemento del array. Esto significa que no se necesita llamar a [reset\(\)](#) antes de un bucle ***foreach***.
- Hay que tener en cuenta que ***foreach*** trabaja con una copia de la lista (*array*) especificada y no la lista en sí, por ello el puntero de la lista no es modificado como en la construcción *each*.

Algunos ejemplos más para demostrar su uso:

```
<?php
/* foreach ejemplo 1: sólo valor */
$a = array(1, 2, 3, 17);

foreach($a as $v) {
    print "Valor actual de \$a: $v.\n";
}

/* foreach ejemplo 2: valor (con clave impresa para ilustrar) */
$a = array(1, 2, 3, 17);
$i = 0; /* sólo para propósitos demostrativos */

foreach($a as $v) {
    print "\$a[$i] => $v.\n";
    $i++;
}

/* foreach ejemplo 3: clave y valor */
$a = array(
    "uno" => 1,
    "dos" => 2,
    "tres" => 3,
    "diecisiete" => 17
);
foreach($a as $k => $v) {
    print "\$a[$k] => $v.\n";
}

/* foreach ejemplo 4: matriz multi-dimensional */
$a[0][0] = "a";
$a[0][1] = "b";
$a[1][0] = "y";
$a[1][1] = "z";

foreach($a as $v1) {
    foreach ($v1 as $v2) {
        print "$v2\n";
    }
}

/* foreach ejemplo 5: matriz dinámica */

foreach(array(1, 2, 3, 4, 5) as $v) {
    print "$v\n";
}
?>
```

III. MATERIALES Y EQUIPO

Para la realización de la guía de práctica se requerirá lo siguiente:

No.	Material	Cantidad
1	Guía de práctica #2	1
2	Computadora con WampServer instalado y funcionando correctamente	1
3	Editor Visual Studio Code o DevPHP, Sublime, Notepad++ , etc.	1
4	Memoria USB	1

IV. PROCEDIMIENTO

Indicaciones:

1. Asegúrese de digitar el código de los siguientes ejemplos que se presentan a continuación. Tenga en cuenta que los editores de código PHP no son compiladores solamente editores por lo tanto los errores de sintaxis lo podrá observar únicamente hasta que se ejecute el script al cargar la página en el navegador de su preferencia.
2. Recuerde crear una carpeta en la carpeta del servidor que está utilizando (www, httdocs), en donde almacenara los recursos de la guía y sus páginas PHP

Ejemplo 1: En este ejemplo se evaluará si un año ingresado por un usuario a través de un formulario resulta ser bisiesto o no. El script PHP procesará el año ingresado para determinar si resulta ser bisiesto o no.

Archivo 1: bisiesto.php

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Año Bisiesto</title>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body class="bg-light">
  <div class="container py-5">
    <div class="row justify-content-center">
      <div class="col-md-6">
        <div class="card shadow">
          <div class="card-header text-center bg-primary text-white">
            <h4>Comprobador de Años Bisiestos</h4>
          </div>
          <div class="card-body">
            <?php
              $currentYear = date('Y');
              if (!isset($_POST['enviar'])): ?>
```

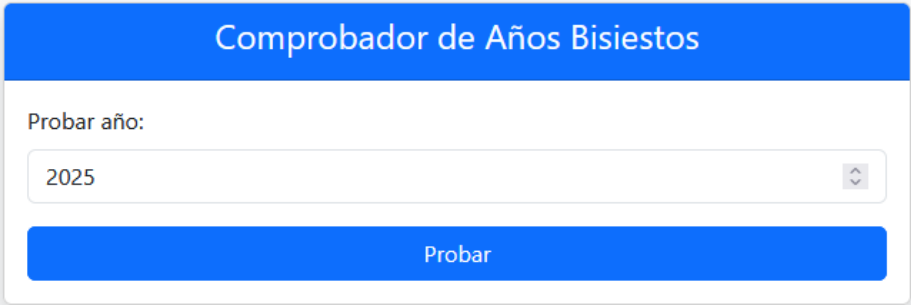
```

        <form action="<?= $_SERVER['PHP_SELF']; ?>" method="POST"
name="frmdatos" onsubmit="return validarAnio();">
            <div class="mb-3">
                <label for="year" class="form-label">Probar
año:</label>
                <input type="number" class="form-control" id="year"
name="year" placeholder="Ingrese el año" required>
                <div id="numbersOnly" class="form-text">Ingrese un
número entero mayor a 1900 y menor o igual a <?= $currentYear; ?></div>
            </div>
            <button type="submit" name="enviar" class="btn btn-
primary w-100">Probar</button>
        </form>
        <?php else:
            $year = intval($_POST['year']);
            if ($year <= 0 || $year > $currentYear) {
                echo "<div class='alert alert-danger text-center
mt-3'>Por favor, ingrese un año válido mayor a 1900 y menor o igual a
$currentYear.</div>";
                echo "<div class='text-center mt-3'><a href='" .
$_SERVER['PHP_SELF'] . "' class='btn btn-secondary'>Intentar de nuevo</a></div>";
            } elseif (($year % 4 === 0 && $year % 100 !== 0) ||
($year % 400 === 0)) {
                echo "<div class='alert alert-success text-center
mt-3'>El año $year es bisiestro.</div>";
            } else {
                echo "<div class='alert alert-danger text-center
mt-3'>El año $year no es bisiestro.</div>";
            }
            echo "<div class='text-center mt-3'><a href='" .
$_SERVER['PHP_SELF'] . "' class='btn btn-secondary'>Probar otro año</a></div>";
            endif;
        ?>
    </div>
</div>
</div>
</div>
<script>
    function validarAnio() {
        const yearInput = document.getElementById('year').value;
        const currentYear = new Date().getFullYear();
        if (yearInput <= 1900 || yearInput > currentYear
|| !Number.isInteger(Number(yearInput))) {

```

```
        alert(`Por favor, ingrese un año válido mayor a 1900 y menor o igual  
a ${currentYear}.`);  
        return false;  
    }  
    return true;  
}  
</script>  
<script  
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"  
></script>  
</body>  
</html>
```

Resultado del **script bisiesto.php** en el navegador usando el servidor web:

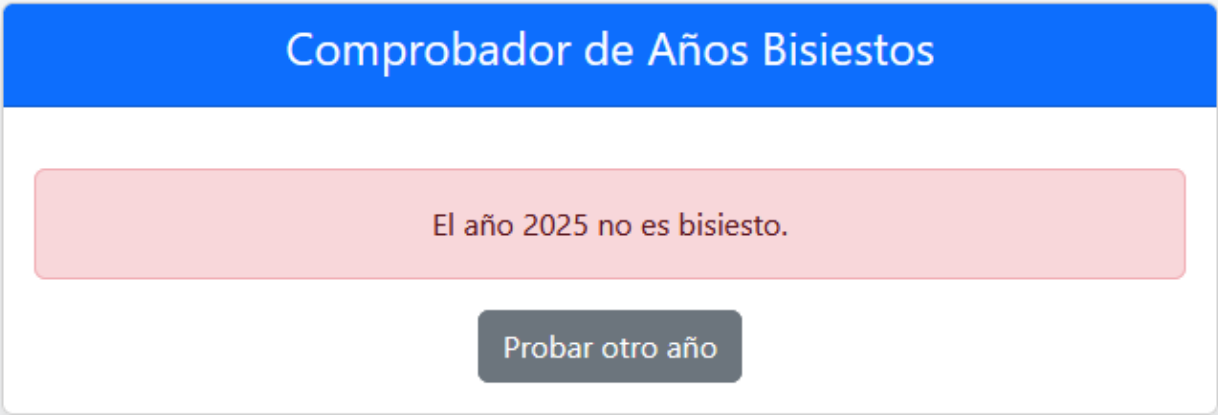


Comprobador de Años Bisiestos

Probar año:

2025

Probar



Comprobador de Años Bisiestos

El año 2025 no es bisiesto.

Probar otro año

Ejemplo 2: Se desea calcular el salario o sueldo neto de algún empleado mediante el uso de un formulario de ingreso de la información necesaria para realizar el cálculo. El formulario debe solicitar el nombre del empleado, su sueldo base y, si realiza horas extras, debe ingresarse el número de horas extras trabajadas y el salario por hora extra a pagar.

Al enviar los datos del formulario debe mostrarse de forma ordenada toda la información en una boleta de pago, mostrando como dato final el sueldo neto o líquido a pagar, tomando en cuenta las horas extras. En este formulario de ingreso se ha utilizado validación en el cliente de la entrada de datos en los campos de texto del formulario, haciendo uso de JavaScript no intrusivo (investigar sobre este tema, si no está claro).

Archivo 1: salario.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Calculo de Salario</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,user-scalable=no,initial-
scale=1.0,maximum-scale=1.0,minimum-scale=1.0" />
  <meta http-equiv="Content-Script-Type" content="text/javascript" />
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/css/bootstrap.min.css" />
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/js/bootstrap.bundle.min.js"></script>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <script src="js/disabledtextfields.js"></script>
</head>

<body>
<header class="container" id="inset">
  <h1 class="mb-4 text-center">Salario empleado</h1>
</header>
<section class="container py-4">
  <article>
    <div class="mb-3" id="empleado">
      <form action="salario.php" method="POST" name="salario" id="salario"
class="bg-light p-4 rounded shadow-sm">
        <div class="mb-4 text-center">
          <h2 class="text-primary">Cálculo de salario</h2>
        </div>

        <div class="mb-3">
          <label for="txtempleado" class="form-label">Nombre del empleado</label>
          <input class="form-control" type="text" name="empleado"
id="txtempleado" maxlength="50" placeholder="Nombre empleado" pattern="[A-Za-
záÉÍÓÚáéíóúÜü\s]+" title="Solo acepta caracteres alfabéticos" required />
          <div class="form-text">Solo acepta caracteres alfabéticos.</div>
        </div>

        <div class="mb-3">
          <label for="sueldobase" class="form-label">Sueldo base</label>
```



```

        <input type="text" class="form-control" name="sueldobase"
id="sueldobase" maxlength="12" placeholder="Sueldo base" pattern="\d+(\.\d{1,2})?"
title="Solo acepta números y punto decimal" required />
        <div class="form-text">Solo acepta números y punto decimal.</div>
    </div>

    <div class="mb-3 form-check">
        <input type="checkbox" class="form-check-input" name="hextra"
id="hextrasi" value="no" />
        <label class="form-check-label" for="hextrasi">Habilitar horas
extras</label>
    </div>

    <div class="mb-3">
        <label for="numhorasex" class="form-label">Número de horas
extra</label>
        <input type="text" class="form-control" name="numhorasex"
id="numhorasex" maxlength="2" placeholder="Número de horas" pattern="\d+"
title="Solo acepta números enteros" disabled />
        <div class="form-text">Solo acepta números enteros.</div>
    </div>

    <div class="mb-3">
        <label for="pagohextra" class="form-label">Pago por hora extra</label>
        <input type="text" class="form-control" name="pagohextra"
id="pagohextra" maxlength="6" placeholder="Pago hora extra"
pattern="\d+(\.\d{1,2})?" title="Solo acepta números y punto decimal" disabled />
        <div class="form-text">Solo acepta números y punto decimal.</div>
    </div>

    <div class="d-flex justify-content-between">
        <button type="submit" class="btn btn-primary" name="enviar"
id="enviar">Enviar</button>
        <button type="reset" class="btn btn-secondary" name="reset"
id="reset">Restablecer</button>
    </div>
</form>
</div>
</article>
</section>
</body>
</html>

```

Archivo 2: salario.php

```

<!DOCTYPE html>
<html lang="es">
<head>
  <title>Calcular Salario</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,user-scalable=no,initial-scale=1.0,maximum-
scale=1.0,minimum-scale=1.0" />
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/css/bootstrap.min.css" />
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>
<header id="inset">
  <h1 class="mb-4 text-center">Detalle del salario</h1>
</header>
<section class="container mt-5">
  <article class="card shadow p-4">
    <?php
    if (isset($_POST['enviar'])) {
      $empleado = isset($_POST['empleado']) ? $_POST['empleado'] : "";
      $sueldobase = isset($_POST['sueldobase']) ? $_POST['sueldobase'] : "";
      if (isset($_POST['hextra'])) {
        $horasextras = isset($_POST['numhorasex']) ? $_POST['numhorasex'] : "0";
        $pagohextra = isset($_POST['pagohextra']) ? $_POST['pagohextra'] : "0";
        $sueldohextras = $horasextras * $pagohextra;
      } else {
        $horasextras = 0;
        $sueldohextras = 0;
      }
      $sueldoneto = $sueldobase + $sueldohextras;

      echo "<div class='alert alert-primary'>\n<h3 class='text-center'>Boleta de pago</h3>\n";
      echo "<table class='table table-striped'>\n";
          echo "  <tr>\n\t\t<th colspan='2' class='text-center'>\n\t\t\tDetalle del
pago\n\t\t</th>\n\t\t</tr>\n";
          echo "  <tr>\n\t\t<td>\n\t\t\tEl empleado es: \n\t\t</td>\n\t\t<td>\n\t\t\t", $empleado,
"\n\t\t</td>\n\t\t</tr>\n";
          echo "  <tr>\n\t\t<td>\n\t\t\tEl sueldo base es: \n\t\t</td>\n\t\t<td>\n\t\t\t", $sueldobase,
"\n\t\t</td>\n\t\t</tr>\n";
          echo "  <tr>\n\t\t<td>\n\t\t\tLas horas extras son: \n\t\t</td>\n\t\t<td>\n\t\t\t", $horasextras,
"\n\t\t</td>\n\t\t</tr>\n";
          echo "  <tr>\n\t\t<td>\n\t\t\tEl pago por hora extra es: \n\t\t</td>\n\t\t<td>\n\t\t\t",
$sueldohextras, "\n\t\t</td>\n\t\t</tr>\n";
          echo "  <tr>\n\t\t<th>\n\t\t\tEl sueldo neto devengado es: \n\t\t</th>\n\t\t<th>\n\t\t\t",
$sueldoneto, "\n\t\t</th>\n\t\t</tr>\n";
    }
  </article>
</section>

```

```

        echo "</table>\n</div>\n";
    }
    ?>
    <a href="salario.html" class="btn btn-primary btn-block">
        <i class="bi bi-person-plus"></i> Ingresar otro empleado
    </a>
</article>
</section>

</body>
</html>

```

1- Resultado de la **página salario.html** en el navegador haciendo uso del servidor web:

Salario empleado

Cálculo de salario

Nombre del empleado

Solo acepta caracteres alfabéticos.

Sueldo base

Solo acepta números y punto decimal.

☒ Habilitar horas extras

Número de horas extra

Solo acepta números enteros.

Pago por hora extra

Solo acepta números y punto decimal.

2- Resultado del **script salario.php** al hacer clic en el **botón Enviar** del formulario del paso anterior:

Detalle del salario

Boleta de pago	
Detalle del pago	
El empleado es:	Karens Medrano
El sueldo base es:	\$ 100
Las horas extras son:	15
El pago por hora extra es:	\$ 37.5
El sueldo neto devengado es:	\$ 137.5

Ingresar otro empleado

Ejemplo 3: El siguiente ejemplo ilustra cómo utilizar el operador condicional para encontrar el mayor de tres números ingresados a través de un formulario.

Los campos de texto donde se ingresan los números son validados con JavaScript para que sólo acepten valores enteros (de 0 a 9), mostrando un mensaje en caso de querer introducir cualquier otro carácter que no sea numérico.

Archivo 1: compararnumeros.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Comparar números</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta http-equiv="Content-Script-Type" content="text/javascript" />
  <!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet">
  <script src="js/validar.js"></script>
</head>
<body>
<header class="bg-primary text-white text-center py-3">
  <h1>Comparación de números</h1>
</header>

<section class="container mt-4">
  <article id="formarea">
    <form name="frmpalabras" method="POST" action="compararnum.php" class="bg-light p-4 rounded shadow">
      <fieldset>
        <div class="mb-3">
          <label for="number1" class="form-label">Primer número:</label>
          <input type="text" name="numero1" id="number1" class="form-control" placeholder="Primer número" />
        </div>

        <div class="mb-3">
          <label for="number2" class="form-label">Segundo número:</label>
          <input type="text" name="numero2" id="number2" class="form-control" placeholder="Segundo número" />
        </div>

        <div class="mb-3">
          <label for="number3" class="form-label">Tercer número:</label>
          <input type="text" name="numero3" id="number3" class="form-control" placeholder="Tercer número" />
        </div>

        <div class="text-center">
```

```

        <button type="submit" name="conteo" id="count" class="btn btn-primary">Enviar</button>
    </div>
</fieldset>
</form>
</article>
</section>

<!-- Bootstrap JS (para funcionalidades como tooltips o modales, si se necesitan en el futuro) -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>

```

Archivo 2: compararnum.php

```

<!DOCTYPE html>
<html lang="es">
<head>
    <title>Contador de palabras</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
/>
</head>
<body>
<header class="bg-primary text-white text-center py-4">
    <h1>
        Resultados de la comparación de <?php echo $num1 = isset($_POST['numero1']) ? $_POST['numero1']
: 0; ?>,
        <?php echo $num2 = isset($_POST['numero2']) ? $_POST['numero2'] : 0; ?> y
        <?php echo $num3 = isset($_POST['numero3']) ? $_POST['numero3'] : 0; ?>
    </h1>
</header>
<section class="container alert alert-light mt-4" role="alert">
    <?php
        $elmayor = ($num1 > $num2) ? $num1 : $num2;
        $elmayor = ($elmayor > $num3) ? $elmayor : $num3;
        printf("El número mayor es: %d", $elmayor);
    ?>
</section>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>

```

1. Resultado del **archivo compararnumeros.html** en el navegador usando el servidor web:

Comparación de números

Primer número:
123

Segundo número:
567

Tercer número:
45

Enviar

- Resultado del **script compararnum.php**, después de haber hecho clic en el botón Enviar, del punto anterior

Resultados de la comparación de 123, 567 y 45

El número mayor es: 567

Ejemplo 4: El siguiente ejercicio muestra cómo realizar la conversión de un número en sistema decimal a uno en sistema binario utilizando para ello cadenas y un ciclo do-while.

Archivo: convertir.php

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Convertidor de Bases Numéricas</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
  <div class="container mt-5">
    <h1 class="text-center">Convertidor de Bases Numéricas</h1>
    <form method="POST" class="mt-4">
      <div class="mb-3">
        <label for="number" class="form-label">Número:</label>
        <input type="text" name="number" id="number" class="form-control" required>
      </div>
      <div class="mb-3">
        <label for="base_from" class="form-label">Base de origen:</label>
        <select name="base_from" id="base_from" class="form-select" required>
          <?php for ($i = 2; $i <= 36; $i++): ?>
            <option value="<?php echo $i; ?>">Base <?php echo $i; ?></option>
          <?php endfor; ?>
        </select>
      </div>
    </form>
  </div>
</body>
</html>
```

```

</div>
<div class="mb-3">
  <label for="base_to" class="form-label">Base de destino:</label>
  <select name="base_to" id="base_to" class="form-select" required>
    <?php for ($i = 2; $i <= 36; $i++): ?>
      <option value="<?php echo $i; ?>">Base <?php echo $i; ?></option>
    <?php endfor; ?>
  </select>
</div>
<button type="submit" class="btn btn-primary w-100">Convertir</button>
</form>

<?php if ($_SERVER['REQUEST_METHOD'] === 'POST'): ?>
  <div class="mt-4">
    <?php
      $number = $_POST['number'];
      $base_from = (int)$_POST['base_from'];
      $base_to = (int)$_POST['base_to'];

      // Validación y proceso de conversión
      try {
        echo "<div class='alert alert-info'><strong>Proceso de conversión:</strong><br>";

        // Convertir a decimal paso a paso
        $decimal = 0;
        $length = strlen($number);
        for ($i = 0; $i < $length; $i++) {
          $digit = base_convert($number[$i], $base_from, 10);
          $power = $length - $i - 1;
          $decimal += $digit * pow($base_from, $power);
          echo "Paso $i: Dígito: $digit, Potencia: $base_from^$power, Resultado Parcial: $decimal<br>";
        }

        echo "Número en decimal: $decimal<br>";

        // Convertir de decimal a base destino paso a paso
        $converted = "";
        $steps = [];
        while ($decimal > 0) {
          $remainder = $decimal % $base_to;
          $steps[] = $remainder;
          $decimal = intdiv($decimal, $base_to);
        }

        $steps = array_reverse($steps);
        foreach ($steps as $step) {
          $converted .= base_convert($step, 10, $base_to);
        }
      } catch (Exception $e) {
        echo "Error: " . $e->getMessage();
      }
    </?php>
  </div>
</?php>

```

```
        echo "Número convertido en base destino: $converted";
        echo "</div>";

        echo "<div class='alert alert-success'>El número <strong>$number</strong> en base
<strong>$base_from</strong> convertido a base <strong>$base_to</strong> es:
<strong>$converted</strong>.</div>";
    } catch (Exception $e) {
        echo "<div class='alert alert-danger'>Hubo un error en la conversión. Verifica los datos
ingresados.</div>";
    }
    ?>
</div>
<?php endif; ?>
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

Resultado en el navegador:

Convertidor de Bases Numéricas

Número:

Base de origen:

Base de destino:

Convertir

Proceso de conversión:

Paso 0: Dígito: 1, Potencia: 10^1 , Resultado Parcial: 10

Paso 1: Dígito: 0, Potencia: 10^0 , Resultado Parcial: 10

Número en decimal: 10

Número convertido en base destino: 1010

El número 10 en base 10 convertido a base 2 es: 1010.

Ejemplo 5: El siguiente ejemplo muestra cómo crear una tabla de conversiones de monedas haciendo uso de un ciclo while.

Archivo: convmonedas.php

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Tabla de Conversiones de Monedas</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
  <div class="container mt-5">
    <h1 class="text-center">Tabla de Conversiones de Monedas</h1>
    <form method="POST" class="mt-4">
      <div class="mb-3">
        <label for="amount" class="form-label">Monto en moneda base (USD):</label>
        <input type="number" name="amount" id="amount" class="form-control" required>
      </div>
      <div class="mb-3">
        <label for="currency" class="form-label">Selecciona la divisa:</label>
        <select name="currency" id="currency" class="form-select" required>
          <option value="EUR">Euro (EUR)</option>
          <option value="GBP">Libra Esterlina (GBP)</option>
          <option value="JPY">Yen Japonés (JPY)</option>
        </select>
      </div>
      <button type="submit" class="btn btn-primary w-100">Generar Tabla</button>
    </form>

    <?php if ($_SERVER['REQUEST_METHOD'] === 'POST'): ?>
      <div class="mt-4">
        <?php
          $amount = (float)$_POST['amount'];
          $currency = $_POST['currency'];

          // Tasas de conversión fijas (ejemplo, actualizar según tasas reales)
          $conversion_rates = [
            'EUR' => 0.85, // 1 USD = 0.85 EUR
            'GBP' => 0.75, // 1 USD = 0.75 GBP
            'JPY' => 110.00 // 1 USD = 110.00 JPY
          ];

          $rate = $conversion_rates[$currency];
          $limit = 10; // Número de filas en la tabla
          $counter = 1;
```

```
        echo "<table class='table table-striped'>";
        echo "    <thead><tr><th>#</th><th>Monto    (USD)</th><th>Convertido
($currency)</th></tr></thead><tbody>";

        while ($counter <= $limit) {
            $converted = $amount * $rate;
            echo "    <tr><td>$counter</td><td>" . number_format($amount, 2) . "</td><td>" .
number_format($converted, 2) . "</td></tr>";
            $amount += 10; // Incremento en el monto base
            $counter++;
        }

        echo "</tbody></table>";
        ?>
    </div>
    <?php endif; ?>
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

Resultado en el navegador:

Tabla de Conversiones de Monedas

Monto en moneda base (USD):

Selecciona la divisa:

Generar Tabla

#	Monto (USD)	Convertido (EUR)
1	1.00	0.85
2	11.00	9.35
3	21.00	17.85
4	31.00	26.35
5	41.00	34.85
6	51.00	43.35
7	61.00	51.85
8	71.00	60.35
9	81.00	68.85
10	91.00	77.35

V. DISCUSION DE RESULTADOS

1. Cree un script que a partir de la fecha de nacimiento ingresada en un campo tipo number de formulario calcule el número de días que la persona que la ha ingresado ha vivido. Tome en cuenta los años bisiestos para obtener el cálculo exacto de días.
2. Un supermercado calcula el precio final de un artículo considerando:
 - Un IVA del 13% aplicado sobre el precio base.
 - Un descuento del 10% si el precio base supera los \$100.Crea un formulario donde el usuario ingrese:
 - El precio base del artículo.El programa debe:
 - a. Calcular el IVA.
 - b. Aplicar el descuento si corresponde.
 - c. Mostrar el precio final después de aplicar el IVA y el descuento.

VII. BIBLIOGRAFIA

- Cabezas Granados, Luis Miguel. PHP 6 Manual Imprescindible. Editorial Anaya Multimedia. 1ª edición. Madrid, España. 2010.
- Doyle, Matt. Fundamentos de PHP Práctico. Editorial Anaya Multimedia. 1ª edición. Madrid, España. 2010.
- Gutiérrez, Abraham / Bravo, Ginés. PHP 5 a través de ejemplos. Editorial Alfaomega RAMA. 1ra edición. México. Junio 2005.
- Tim Converse / Steve Suehring. LA BIBLIA DE PHP 6 y MySQL. Editorial Anaya Multimedia. 1a. Edición. Madrid, España. 2009.
- John Coggeshall. La Biblia de PHP 5. 1ra Edición. Editorial Anaya Multimedia. Madrid España.
- <http://www.php.net/manual/en>