



# UNIVERSIDAD DON BOSCO

## FACULTAD DE INGENIERIA

### ESCUELA DE COMPUTACION

**CICLO 1**

#### GUIA DE LABORATORIO #3

**Nombre de la Practica:** Estructuras de Control (Sentencias repetitivas) y Matrices  
**Lugar de Ejecución:** Centro de Cómputo  
**Tiempo Estimado:** 2 horas con 30 minutos  
**MATERIA:** Desarrollo de Aplicaciones Web  
con Software Interpretado en el Servidor (DSS404)

## I. OBJETIVOS

En esta guía de práctica se espera que el estudiante:

- Adquiera un amplio dominio de la sintaxis de cada una de las sentencias repetitivas disponibles en PHP.
- Haga uso de sentencias para modificar el número de veces y la forma en qué se ejecuta un ciclo o lazo.
- Utilice eficazmente a matrices para almacenar valores y acceder a sus valores en los scripts PHP.
- Utilice estructuras de control repetitivas para trabajar con datos procedentes de una matriz escalar o asociativa.
- Adquiera el dominio de matrices unidimensionales y multidimensionales utilizando ciclos o lazos repetitivos anidados.
- Haga uso de diversas funciones para manejo de matrices con PHP.

## II. INTRODUCCION TEORICA

### 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*.

### Sentencia 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:

<pre>/* ejemplo 1 */  \$i = 1; while (\$i &lt;= 10) {     print \$i++;     /* el valor impreso sería        \$i antes del incremento        (post-incremento) */ }</pre>	<pre>/* ejemplo 2 */  \$i = 1; while (\$i &lt;= 10):     print \$i;     \$i++; endwhile;</pre>
--	--

## Sentencia 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);
```

En 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.

### Sentencia 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:

**inicializacion**, 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.

**condicion**, 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 **condicion**. 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([inicializacion]; [condicion]; [incremento]):  
    sentencia(s);  
endfor;
```

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

<pre>/* ejemplo 1 */  for (\$i = 1; \$i &lt;= 10; \$i++) {     print \$i; }</pre>	<pre>/* ejemplo 2 */  for (\$i = 1; ;\$i++) {     if (\$i &gt; 10) {         break;     }     print \$i; }</pre>	<pre>/* ejemplo 3 */  \$i = 1; for (;;) {     if (\$i &gt; 10) {         break;     }     print \$i;     \$i++; }</pre>
<pre>/* ejemplo 4 */  for (\$i = 1; \$i &lt;= 10; print \$i, \$i++);</pre>		
<pre>/* Ejemplo 5 */  for(\$i = 1; \$i &lt;= 10; \$i++):     print \$i; endfor;</pre>		

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";
    }
    // 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:

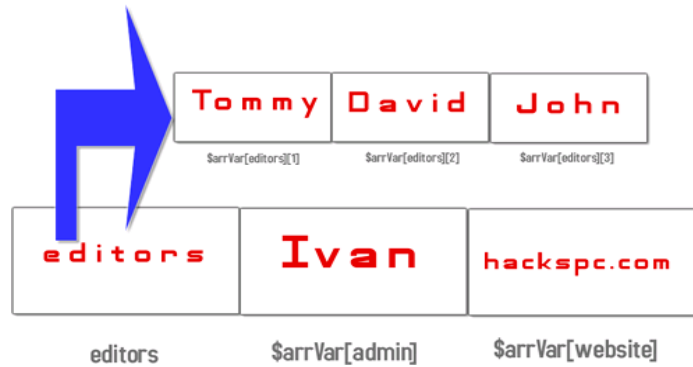
```
/* ejemplo sentencia continue */
$i = 1;
while ($i <= 10) {
    if($i%2 == 0){
        print "$i es par";
    }
    /*Si valor de $i es exactamente 5, el ciclo o lazo avanzará a la siguiente iteración*/
    elseif($i == 5) {
        continue;
    }
    else {
        print "$i es impar";
    }
}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

## Matrices

Una matriz o arreglo es un tipo de dato compuesto que permite almacenar bajo un mismo nombre a un conjunto de valores que pueden ser de tipos de datos diferentes.

Cuando se define una matriz se hace de forma similar a las variables escalares con la diferencia de que la asignación de valores se debe realizar a cada una de las posiciones del arreglo.

También se puede utilizar en lugar de la asignación directa estructuras de control como ciclos repetitivos o funciones para asignar varias posiciones de una vez, como veremos más adelante.



## Clasificación de las matrices

En PHP se pueden clasificar las matrices de dos formas:

- Por el tipo de índice que utilizan. Este índice se utiliza para hacer referencia a un elemento específico del arreglo— y,
- Por el número de dimensiones de la matriz. Esto es, cuántos índices son necesarios para hacer referencia a un elemento específico.

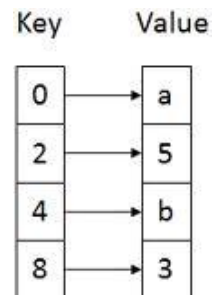
## Tipos de matrices según el tipo de índice

### a) *Matrices escalares*

También se les conoce como matrices enumeradas o indexadas, por el hecho que utilizan números enteros como índices. En PHP el índice para el primer elemento de un arreglo empieza en cero y los siguientes elementos van aumentando en uno este valor de índice, de modo que el segundo elemento tiene índice uno, el tercero índice dos, etc.

Por ejemplo, se puede tener una matriz llamada `$numeros` con cinco elementos. Para asignar un valor a cada uno de los elementos de este arreglo, habría que escribir:

```
$numeros[0] = 1;
$numeros[1] = 2;
$numeros[2] = 3;
$numeros[3] = 4;
$numeros[4] = 5;
```



Luego, si deseamos mandar a imprimir el tercer elemento de este arreglo, tendríamos que escribir:

```
echo $numero[2];
```

Ahora bien, si lo que deseamos es imprimir todos los elementos de la matriz, la mejor solución sería utilizar una sentencia repetitiva como esta:

```
for($i=0; $i<count($numeros); $i++){
    echo $numeros[$i] . "<br />\n";
}
```

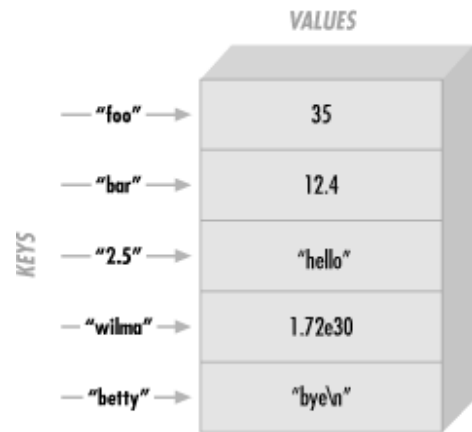
### b) *Matrices asociativas*

Estas matrices también son conocidas como tablas *hash*.

Se caracterizan por utilizar una cadena de caracteres como índice de la matriz. Lo normal es utilizar una cadena que permita asociar fácilmente el índice con el valor que almacena el elemento.

Por ejemplo, se puede crear una matriz llamada `$monedas` con cuatro elementos y asociar valores a cada uno de estos elementos de la siguiente forma:

```
$monedas["El Salvador"] = "dólar";  
$monedas["Guatemala"] = "quetzal";  
$monedas["Costa Rica"] = "colón";  
$monedas["Honduras"] = "lempira";
```



Si deseamos mandar a imprimir los valores de este arreglo, deberíamos escribir:

```
echo $monedas["El Salvador"];  
echo $monedas["Guatemala"];  
echo $monedas["Costa Rica"];  
echo $monedas["Honduras"];
```

O, haciendo uso de estructuras de control repetitivas, con la sentencia `foreach`:

```
foreach($monedas as $key => $value){  
    echo "Moneda de " . $key . ": " . $value . "<br />\n";  
}
```

## Tipos de matrices según el número de dimensiones

### a) *Matrices unidimensionales*

Es una matriz que utiliza un único índice para hacer referencia a todos los elementos de la misma. Este índice puede ser escalar o asociativo.

### b) *Matrices multidimensionales*

Es una matriz que necesita utilizar más de un índice para hacer referencia a un elemento específico. De todas las matrices multidimensionales, las más utilizados son las de dos dimensiones.

Los arreglos de tres o más dimensiones no se utilizan tanto debido a que son más difíciles de comprender y manejar.

El siguiente bloque de código muestra la definición de un arreglo multidimensional (con índices asociativos) y luego manda a imprimir en la ventana del navegador uno de sus elementos:

```
<?  
$pais=array  
(  
    "espana" =>array  
    (  
        "nombre"=>"España",  
        "lengua"=>"Castellano",  
        "moneda"=>"Peseta"    )  
)
```

```

    ),
    "francia" =>array
    (
        "nombre"=>"Francia",
        "lengua"=>"Francés",
        "moneda"=>"Franco"
    )
);
echo $pais["espana"]["moneda"]; //Imprime en la ventana del navegador: "Peseta"
?>

```

## Creación de matrices en PHP

En PHP, las matrices o arreglos se pueden crear asignando los valores a los elementos directamente, tal y como se ha hecho en los ejemplos anteriores, o se pueden utilizar funciones para ese propósito. Las funciones que permiten crear arreglos en PHP son: *array()* y *range()*.

La sintaxis de cada una de ellas se muestra a continuación:

```

$arreglo = array($valor1, $valor2, ... , $valorn);
$matriz = range($valorinicial, $valorfinal, $desplazamiento);

```

La función *array()* se utiliza para crear una variable de tipo matriz y asignarle los valores que va a contener en una sola instrucción. En tanto que, con *range()* se puede crear una matriz que comienza desde el *\$valorinicial* y termina con el *\$valorfinal*.

## Asignar los elementos de una matriz a variables escalares

PHP proporciona una función que permite asignar en una sola instrucción los elementos de una matriz a variables escalares independientes. Esta función es *list()*, que posee como argumentos las variables escalares en donde se almacenarán los valores almacenados en las posiciones del arreglo de forma ordenada. La sintaxis de la función es la que se muestra a continuación:

```
list ($variable1, $variable2, ... , $variablen) = $mi_array
```

En donde:

*\$mi\_array*, es la matriz o arreglo del que se extraerán los elementos.

*\$variable1, \$variable2, ..., \$variablen*, son las variables escalares independientes en las que se asignarán los diferentes elementos de la matriz.

## Funciones importantes para trabajar con matrices

El lenguaje PHP, brinda una cantidad considerable de funciones para facilitar el trabajo con matrices. A continuación se brinda información detallada de varias de ellas:

### Funciones para recorrer una matriz

Cuando se trabaja con matrices, PHP ofrece una serie de funciones que permiten realizar de forma más fácil y conveniente esta tarea.



**count()**: Permite obtener el número de elementos que componen la matriz. Es muy útil cuando se pretende realizar un recorrido sobre una matriz utilizando ciclos *for*, *while* o *do-while*. Por ejemplo:

```
for($i=0; $i<count($matriz); $i++)
```

**sizeof()**: También se utiliza para obtener el número de elementos de una matriz. Por ejemplo:

```
for($i=0; $i<sizeof($matriz); $i++)
```

**next()**: Devuelve el valor del elemento siguiente al actual (si existe), avanzando el puntero interno una posición. En caso de que el elemento actual sea el último de la matriz, devuelve *false*.

**prev()**: Devuelve el valor del elemento anterior al actual (si existe), retrocediendo el puntero interno una posición. En caso de que el elemento actual sea el primero, devuelve *false*.

**current()**: Devuelve el valor del elemento situado en la posición actual del cursor. Si la matriz está vacía o no hay más elementos, la función devolverá *false*.

**key()**: Devuelve el índice del elemento situado en la posición actual del cursor. Si la matriz está vacía, o no hay más elementos, devuelve la constante NULL.

**reset()**: Sitúa el cursor interno de la matriz en el primer elemento de la misma y devuelve el valor de dicho elemento. Si la matriz está vacía, la función devuelve el valor *false*.

**end()**: Avanza el cursor interno de la matriz hasta el último elemento de la misma y devuelve el valor de dicho elemento. Si la matriz está vacía devuelve *false*.

**each()**: Se usa para recorrer matrices (especialmente matrices asociativas), devolviendo simultáneamente el índice y el valor asociado al índice actual de un elemento. Además, avanza el puntero interno de la matriz hacia el siguiente elemento. Si el puntero interno apunta a la última posición de la matriz, la función devuelve *false*.

**list()**: Asigna una lista de variables en una sola operación. Suele utilizarse en combinación con la función *each()*.

**OBSERVACIÓN:** El argumento de todas las funciones anteriores es una matriz. A excepción de la función *list()*.

Ejemplo:

```
$matriz = array("Primavera", "Verano", "Otoño", "Invierno");  
do{  
    $index = key($matriz);  
    $value = current($matriz);  
    echo $index . ": " . $value . "<br>\n";  
}while(next($matriz));
```

## Sentencia 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 un matriz asociativa.

Hay dos sintaxis; la segunda es una extensión menor, pero útil de la primera:

```
foreach(expresion_array as $value) //sentencias;
```

```
foreach(expresion_array as $key => $value) //sentencias;
```

La primera forma recorre el array dado por ***expresion\_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 si, 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 #3: Sentencias repetitivas y matrices	1
2	Computadora con WampServer instalado y funcionando correctamente	1
3	Editor PHP Sublime Text o Eclipse PHP	1
4	Memoria USB o disco flexible	1

### IV. PROCEDIMIENTO

Indicaciones: Asegúrese de digitar el código de los siguientes ejemplos que se presentan a continuación. Tenga en cuenta que el PHP Designer no es compilador solamente un editor por lo tanto los errores de sintaxis los podrá observar únicamente hasta que se ejecute el script al cargar la página en el navegador de su preferencia.

### Nota:

- Crearemos la carpeta **guia3** dentro de la carpeta **www** o **htdocs** dependiendo de la herramienta que utilice.
- Será necesario guardar los siguientes documentos dentro de las carpetas proporcionadas como recursos. Asegurarse que cada uno sea guardado en la carpeta que tiene el nombre de cada ejemplo.

**Ejemplo 1: El siguiente ejemplo muestra cómo crear una tabla de conversiones de monedas haciendo uso de un ciclo while.**

#### Archivo 1: convmonedas.php

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Convertidor de monedas</title>
  <meta charset="utf-8" />
  <link rel="stylesheet" href="css/monedas.css" />
</head>
<body>
<header>
  <h2>Tabla de conversión de colones a dólares</h2><hr />
</header>
<section>
<article>
  <?php
    define("EQUIV", "8.75");
    $colones = 0.25;
    $tabla = "<table>\n<thead>\n";
    $tabla .= "<th>Colones</th>";
    $tabla .= "<th>Dólares</th>";
    $tabla .= "</thead>\n<tbody>\n";
    while($colones <= 8){
      $tabla .= "<tr>\n<td>&cent; ";
      $tabla .= number_format($colones, 2, '.', ',') . "</td><td>\$ ";
      $tabla .= number_format($colones / EQUIV, 2, '.', ',');
      $colones += 0.25;
      $tabla .= "</td>\n</tr>\n";
    } // fin del while
    $tabla .= "</tbody>\n</table>\n";
    echo $tabla;
  ?>
</article>
</section>
<script src="js/modernizr.custom.lis.js"></script>
</body>
</html>
```

Archivo 2:  
Resultado en el navegador:

**Tabla de conversión de colones a dólares**

Colones	Dólares
₡ 0.25	\$ 0.03
₡ 0.50	\$ 0.06
₡ 0.75	\$ 0.09
₡ 1.00	\$ 0.11
₡ 1.25	\$ 0.14
₡ 1.50	\$ 0.17
₡ 1.75	\$ 0.20
₡ 2.00	\$ 0.23
₡ 2.25	\$ 0.26
₡ 2.50	\$ 0.29
₡ 2.75	\$ 0.31
₡ 3.00	\$ 0.34
₡ 3.25	\$ 0.37
₡ 3.50	\$ 0.40
₡ 3.75	\$ 0.43
₡ 4.00	\$ 0.46
₡ 4.25	\$ 0.49
₡ 4.50	\$ 0.51
₡ 4.75	\$ 0.54

Resultado personal:

**Tabla de conversión de colones a dólares**

Colones	Dólares
₡ 0.25	\$ 0.03
₡ 0.50	\$ 0.06
₡ 0.75	\$ 0.09
₡ 1.00	\$ 0.11
₡ 1.25	\$ 0.14
₡ 1.50	\$ 0.17
₡ 1.75	\$ 0.20
₡ 2.00	\$ 0.23
₡ 2.25	\$ 0.26
₡ 2.50	\$ 0.29
₡ 2.75	\$ 0.31
₡ 3.00	\$ 0.34
₡ 3.25	\$ 0.37
₡ 3.50	\$ 0.40
₡ 3.75	\$ 0.43
₡ 4.00	\$ 0.46
₡ 4.25	\$ 0.49
₡ 4.50	\$ 0.51
₡ 4.75	\$ 0.54
₡ 5.00	\$ 0.57
₡ 5.25	\$ 0.60
₡ 5.50	\$ 0.63
₡ 5.75	\$ 0.66
₡ 6.00	\$ 0.69
₡ 6.25	\$ 0.71
₡ 6.50	\$ 0.74
₡ 6.75	\$ 0.77
₡ 7.00	\$ 0.80
₡ 7.25	\$ 0.83
₡ 7.50	\$ 0.86
₡ 7.75	\$ 0.89
₡ 8.00	\$ 0.91

**Ejemplo 2:** El siguiente ejemplo ilustra cómo utilizar ciclos o lazos do-while para acumular valores y obtener datos como el valor menor y mayor de una serie de números así como el total de números pares presentes en la misma.

**Archivo 1: intervalos.php**

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Bucle do-while</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="css/intervalos.css" />
    <link rel="stylesheet" href="css/jquery-responsiveTables.css" />
    <script src="js/modernizr.custom.lis.js"></script>
    <script src="//ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>
    <script>window.jQuery || document.write('<script src="js/jquery-
1.12.0.min.js"></script>')</script>
    <script src="js/jquery-responsiveTables.js"></script>
    <script src="js/jquery.responsiveText.js"></script>
    <script>
        $(document).ready(function() {
            $('.responsive').not('table').responsiveText();
            $('table.responsive').responsiveTables();
        });
    </script>
```

```

</head>
<body>
<header>
    <h1>Ciclos, lazos o bucles</h1>
</header>
<section>
<?php
    $min = 1;
    echo "<article>";
    echo "<div id=\"main\">";
    echo "<h2 class=\"subheading responsive\" data-compression=\"25\" data-min=\"14\"
data-max=\"40\">Ejemplos con el lazo o bucle <em>do-while</em></h2>";
    echo "</article>";
    echo "<p>\n";
    echo "Esta iteración sencilla con ";
    echo "<em>do-while</em> imprime una serie de ";
    echo "20 números enteros\n";
    echo "</p>\n";
    echo "<table class=\"responsive\" data-min=\"11\" data-max=\"30\" cellpadding=\"0\"
cellspacing=\"0\">\n";
    echo "\t<tbody>\n";
    echo "\t\t<tr>\n";
    do {
        echo "<td>$min</td>\n";
        $min++;
    } while ($min <= 20);
    echo "\t\t</tr>\n";
    echo "</tbody>\n";
    echo "</table>\n";
    $min=1;
    $max=95070;
    echo "<p>&nbsp;</p>\n<p>&nbsp;</p>\n";
    echo "<p>\nLa siguiente iteración muestra ";
    echo "el total de números pares en un ";
    echo "intervalo que va desde $min hasta $max\n</p>\n";
    $contador=0;
    echo "<table class=\"responsive\" data-min=\"11\" data-max=\"30\" cellpadding=\"0\"
cellspacing=\"0\">\n";
    echo "\t<tr>\n";
    echo "\t\t<td>El valor mínimo de la serie es:</td>\n";
    echo "\t\t<td>$min</td>\n";
    echo "\t\t<td>El valor máximo de la serie es:</td>\n";
    echo "\t\t<td>$max</td>\n";
    echo "\t</tr>\n</table>\n";
    do {
        if($min%2 == 0) $contador++;
        $min++;
    }while ($min <= $max);
    echo "<table class=\"responsive\" data-min=\"11\" data-max=\"30\" cellpadding=\"0\"


```

```

cellspacing="0">\n";
    echo "\t<tr>\n<td>\n";
    echo "El total de números ";
    echo "pares en este intervalo es:\n</td>\n";
    echo "\t\t<td>$contador</td>\n";
    echo "\t</tr>\n";
    echo "</table>\n";
?>
</section>
</body>
</html>

```

### Resultado en el navegador:



## CICLOS, LAZOS O BUCLES

### Ejemplos con el lazo o bucle *do-while*


Esta iteración sencilla con *do-while* imprime una serie de 20 números enteros

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----

La siguiente iteración muestra el total de números pares en un intervalo que va desde 1 hasta 95070

El valor mínimo de la serie es:	1	El valor máximo de la serie es:	95070
El total de números pares en este intervalo es:		47535	

### Resultado personal:



## CICLOS, LAZOS O BUCLES

### Ejemplos con el lazo o bucle *do-while*

Esta iteración sencilla con *do-while* imprime una serie de 20 números enteros

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----

La siguiente iteración muestra el total de números pares en un intervalo que va desde 1 hasta 95070

El valor mínimo de la serie es:	1	El valor máximo de la serie es:	95070
El total de números pares en este intervalo es:		47535	

Estudiante: José Adrián López Medina - LM242664

Técnico en Ingeniería en Computación - Escuela de Computación Ciclo I - 2025



**Ejemplo 3: El siguiente ejemplo muestra cómo utilizar un array asociativo bidimensional para llevar las notas de los alumnos de tres materias: LIS, POO y LPII**

**Archivo 1: promediomaterias.php**

```
<?php
$materias = array(
    "Lenguajes Interpretados en el Servidor" => array(
        'Claudia Maritza Salazar' => 8.6,
        'Sonia Lorena López' => 4.9,
        'Carlos Alberto Ávalos' => 7.9,
        'Jennifer Gabriela Vásquez' => 6.2,
        'Luisa Roxana Calderón' => 7.5,
        'Pedro Javier Hidalgo' => 9.8,
        'Morena Lissette Segovia' => 8.0,
        'Ligia Beatriz Hernández' => 7.6,
        'Luis Alberto Figueroa' => 9.5,
        'Julia María García' => 8.7,
        'Ileana Carolina Dominguez' => 9.4,
        'Oscar Rodrigo Reyes' => 8.5
    ),
    "Programación Orientada a Objetos I" => array(
        'Ramiro Leonel Zepeda' => 5.5,
        'Claudia Maritza Salazar' => 6.4,
```

```

        'Sonia Lorena López' => 8.3,
        'Carlos Alberto Ávalos' => 7.5,
        'José Luis Peñate' => 9.6,
        'Luisa Roxana Calderón' => 7.5,
        'Pedro Javier Hidalgo' => 5.8,
        'Morena Lissette Segovia' => 9.2,
        'Ligia Beatriz Hernández' => 5.0,
        'Marcelo José Menjivar' => 6.0,
        'Ileana Carolina Dominguez' => 8.2,
        'Oscar Rodrigo Reyes' => 8.5,
        'Claudia María Ponce' => 6.2
    ),
    "Lenguaje de Programación II" => array(
        'Javier Ernesto Fuentes' => 6.0,
        'Brenda Jocelyn Ponce' => 6.8,
        'Cristina Guadalupe Morales' => 5.3,
        'Ernesto Javier Hurtado' => 7.0,
        'Sonia Lorena López' => 4.3,
        'Carlos Alberto Ávalos' => 7.5,
        'José Luis Peñate' => 9.6,
        'Luisa Roxana Calderón' => 7.5,
        'Pedro Javier Hidalgo' => 5.8,
        'Lidia Esmeralda Cienfuegos' => 6.8,
        'Marcelo José Menjivar' => 6.0,
        'Ileana Carolina Dominguez' => 8.2,
        'Oscar Rodrigo Reyes' => 2.8,
        'Claudia María Ponce' => 6.2
    )
);

//Creando la página web con cadenas usando
//la sintaxis HERE DOC
echo "<!DOCTYPE html>";
echo "<html>\n";
echo "<head>\n";
echo "\t<title>Uso del foreach para recorrer una matriz</title>\n";
echo "\t<meta charset=\"utf-8\" />\n";
echo "\t<link rel=\"stylesheet\" href=\"css/fonts.css\" />\n";
echo "\t<link rel=\"stylesheet\" href=\"css/notasalumnos.css\" />\n";
echo "</head>\n";
echo "<body>\n";
echo "<header>\n";
echo "\t<h1>Notas de los estudiantes</h1><hr>\n";
echo "</header>\n";
echo "<section>\n";
echo "<article>\n";
foreach($materias as $materia => $notas){
    echo "\t<table>\n";
    echo "\t\t<thead>\n";

```



?>

### Resultado en el navegador:

Notas de los estudiantes	
Lenguajes Interpretados en el Servidor	
Nombre	Nota
Claudia Maritza Salazar	8.6
Sonia Lorena López	4.9
Carlos Alberto Ávalos	7.9
Jennifer Gabriela Vásquez	6.2
Luisa Roxana Calderón	7.5
Pedro Javier Hidalgo	9.8
Morena Lissette Segovia	8.0
Ligia Beatriz Hernández	7.6
Luis Alberto Figueroa	9.5
Julia María García	8.7
Ileana Carolina Domínguez	9.4
Oscar Rodrigo Reyes	8.5
CUM: 8.05	

### Resultado personal:

Lenguaje de Programación II	
Nombre	Nota
Javier Ernesto Fuentes	6.0
Brenda Jocelyn Ponce	6.8
Cristina Guadalupe Morales	5.3
Ernesto Javier Hurtado	7.0
Sonia Lorena López	4.3
Carlos Alberto Ávalos	7.5
José Luis Peñate	9.6
Luisa Roxana Calderón	7.5
Pedro Javier Hidalgo	5.8
Lidia Esmeralda Cienfuegos	6.8
Marcelo José Menjivar	6.0
Ileana Carolina Domínguez	8.2
Oscar Rodrigo Reyes	2.8
Claudia María Ponce	6.2
CUM: 6.41	

Estudiante: José Adrián López Medina - LM242664  
Técnico en Ingeniería en Computación - Escuela de Computación Ciclo I - 2025

**Ejercicio #4:** El siguiente ejercicio permite crear una serie de letras o números dependiendo de la selección del usuario en un control de menú desplegable. Para crear la serie generado se utiliza la función `range()`, proporcionando todos argumentos: el valor inicial y final de la serie, así como, el desplazamiento para la secuencia de cada elemento.

**Archivo 1: series.php**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Uso de la función range</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="css/series.css" />
  <link rel="stylesheet" href="css/formoid-solid-purple.css" />
  <script src="js/modernizr.custom.lis.js"></script>
</head>
</body>
<header>
  <h1>Generación de series con función de matrices</h1>
</header>
<section>
<article>
<form action="<?=$_SERVER['PHP_SELF']; ?>" method="POST" class="formoid-solid-purple">
  <h2>Tipos de serie</h2>
  <div class="element-select">
```

```

<label class="title"></label>
<div class="item-cont">
  <div class="large">
    <span>
      <select name="seltipo">
        <option value="AlfabetoMin" selected="selected">
          Alfabeto en minúsculas
        </option>
        <option value="AlfabetoMay">
          Alfabeto en mayúsculas
        </option>
        <option value="NumerosPares">
          Números pares
        </option>
        <option value="NumerosImpares">
          Números impares
        </option>
      </select>
      <i></i>
      <span class="icon-place"></span>
    </span>
  </div>
</div>
<div class="submit">
  <input type="submit" name="enviar" value="Enviar" />
</div>
</form>
<?php
if(isset($_POST['enviar'])){
  $tiposecuencia = isset($_POST['seltipo']) ? $_POST['seltipo'] : "";
  switch($tiposecuencia){
    case "AlfabetoMin":
      $inicio = 'a';
      $final = 'z';
      $salto = '1';
      break;
    case "AlfabetoMay":
      $inicio = 'A';
      $final = 'Z';
      $salto = '1';
      break;
    case "NumerosPares":
      $inicio = '0';
      $final = '50';
      $salto = '2';
      break;
    case "NumerosImpares":
      $inicio = '1';

```

```

        $final = '50';
        $salto = '2';
        break;
    }
    $secuencia = range($inicio,$final,$salto);
    echo "<div id=\"box\">";
    foreach($secuencia as $letra)
        echo "<span class=\"caracter\">" . $letra . "</span>&nbsp;&nbsp;&nbsp;";
    echo "</div>";
}
?>
</article>
</section>
</body>
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>
<script>window.jQuery || document.write('<script
src="js/jquery.min.js"></script>')</script>
<script src="js/formoid-solid-purple.js"></script>
</html>

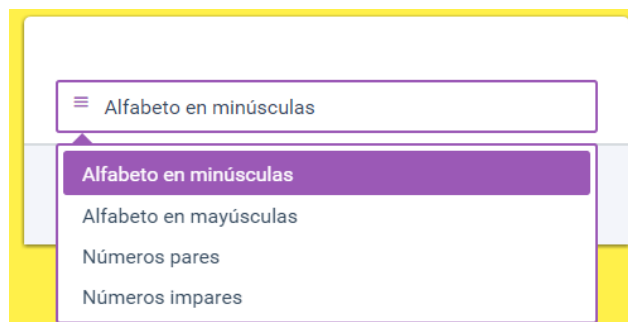
```

**Resultado en el navegador:**

## Generación de series con función de matrices



A screenshot of a web form on a yellow background. The form has a light gray border and contains a dropdown menu with the text "Alfabeto en minúsculas" and a purple button labeled "Enviar".



A screenshot of a web form on a yellow background, showing a dropdown menu with a list of options. The options are "Alfabeto en minúsculas", "Alfabeto en mayúsculas", "Números pares", and "Números impares". The "Alfabeto en minúsculas" option is highlighted in purple.

≡ Alfabeto en minúsculas

Enviar

0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50

Resultado personal:

# Generación de series con función de matrices

≡ Alfabeto en minúsculas

Enviar

A B C D E F G H I J K L M N O P Q R S T U V W X Y  
Z

**Estudiante: José Adrián López Medina - LM242664**  
**Técnico en Ingeniería en Computación - Escuela de Computacion Ciclo I - 2025**



**Ejercicio #5:** El siguiente ejemplo muestra el uso de la función `array_rand()` de PHP que permite generar un número de índices aleatorios para mostrar las imágenes de películas de la portada. Luego, se puede seleccionar un nombre de película y la cantidad de boletos de entrada que se desean adquirir. Al enviar el formulario y utilizando una matriz bidimensional asociativa se puede crear un boleto que indica el precio a pagar por el número de entradas adquirido.

**Archivo 1: cartelera.php**

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Cartelera de cine</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="css/cartelera.css" />
    <script src="js/modernizr.custom.lis.js"></script>
</head>
<body>
<header>
    <h1>Cartelera de cine</h1>
</header>
<section>
<?php
    //Creamos una variable con el directorio o carpeta que contiene las imágenes
    $dir = "img";
    //Creamos una matriz con los nombres de las imágenes que se almacenan en el directorio
    $peliculas = array(
        "al-diablo-con-el-diablo.jpg", "click.jpg",
        "cruzada.jpg", "efecto-mariposa.jpg",
        "busca-de-la-felicidad.jpg", "amenaza-fantasma.jpg",
        "la-milla-verde.jpg", "la-propuesta.jpg",
        "comunidad-del-anillo.jpg", "sexto-sentido.jpg"
    );
    //Obtener cuatro claves de la matriz $peliculas
    $claves = array_rand($peliculas, 4);
    //Creando una estructura de 3 columnas con elementos DIV
```

```

?>
<article>
    " width="200" /><br />
    " width="200" />
</article>
<article>
    <hgroup>
        <h2>CINE REFORMA</h2>
        <h3>Los mejores estrenos los encuentras aquí</h3>
        <h4>Estas son las películas en cartelera</h4>
    </hgroup>
    <div id="">
        <form action="pagoentrada.php" method="POST">
            <fieldset>
                <legend><span>Entrada al cine</span></legend>
                <label for="pelicula">Película</label>
                <select name="pelicula" id="pelicula">
                    <option value="mov0001" selected="selected">
                        Al diablo con el diablo
                    </option>
                    <option value="mov0002">
                        Click
                    </option>
                    <option value="mov0003">
                        Cruzada
                    </option>
                    <option value="mov0004">
                        El efecto mariposa
                    </option>
                    <option value="mov0005">
                        En busca de la felicidad
                    </option>
                    <option value="mov0006">
                        La amenaza fastasma
                    </option>
                    <option value="mov0007">
                        Milagros inesperados
                    </option>
                    <option value="mov0008">
                        La propuesta
                    </option>
                    <option value="mov0009">
                        La comunidad del anillo
                    </option>
                    <option value="mov0010">
                        El sexto sentido
                    </option>
                </select><br />
                <label for="cantidad">Cantidad</label>
            </fieldset>
        </form>
    </div>
</article>

```

```

        <input type="text" name="cantidad" id="cantidad" maxlength="1"
placeholder="Cantidad" pattern="\d{1}" required /><br />
        <input type="submit" name="comprar" value="Comprar" />
    </fieldset>
</form>
</div>
</article>
<article>
    " width="200" /><br />
    " width="200" />
</article>
</section>
</body>
</html>

```

## Archivo 2: pagoentrada.php

```

<!DOCTYPE html>
<html>
<head>
    <title>Boleto de entrada</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="css/tables.css" />
    <script src="js/modernizr.custom.lis.js"></script>
</head>
<body>
<?php
    $peliculas = array(
        "mov0001" => array(
            "titulo" => "Al diablo con el diablo",
            "precio" => 3.75,
            "sala" => "Sala 1",
            "hora" => "7:00 pm"
        ),
        "mov0002" => array(
            "titulo" => "Clic",
            "precio" => 3.75,
            "sala" => "Sala 3",
            "hora" => "5:00 pm"
        ),
        "mov0003" => array(
            "titulo" => "Cruzada",
            "precio" => 3.00,
            "sala" => "Sala 2",
            "hora" => "1:00 pm"
        ),
        "mov0004" => array(
            "titulo" => "El efecto mariposa",
            "precio" => 3.75,

```

```

        "sala" => "sala 3",
        "hora" => "6:00 pm"
    ),
    "mov0005" => array(
        "titulo" => "En busca de la felicidad",
        "precio" => 3.00,
        "sala" => "Sala 5",
        "hora" => "3:00 pm"
    ),
    "mov0006" => array(
        "titulo" => "La amenaza fastasma",
        "precio" => 3.75,
        "sala" => "sala 4",
        "hora" => "6:00 pm"
    ),
    "mov0007" => array(
        "titulo" => "Milagros inesperados",
        "precio" => 3.75,
        "sala" => "Sala 2",
        "hora" => "1:00 pm"
    ),
    "mov0008" => array(
        "titulo" => "La propuesta",
        "precio" => 3.75,
        "sala" => "sala 1",
        "hora" => "9:00 pm"
    ),
    "mov0009" => array(
        "titulo" => "La comunidad del anillo",
        "precio" => 3.75,
        "sala" => "Sala 5",
        "hora" => "7:00 pm"
    ),
    "mov0010" => array(
        "titulo" => "El sexto sentido",
        "precio" => 3.75,
        "sala" => "sala 3",
        "hora" => "8:45 pm"
    )
);

//Procesando los datos enviados del formulario
if(isset($_POST['comprar'])){
    $movie = isset($_POST['pelicula']) ? $_POST['pelicula'] : "";
    switch($movie){
        case "mov0001":
            $seleccionada = $peliculas["mov0001"];
            $cantidad = (isset($_POST['cantidad']) && is_numeric($_POST['cantidad'])) ?
$_POST['cantidad'] : 0;

```

```

        break;
        case "mov0002":
            $seleccionada = $peliculas["mov0002"];
            $cantidad = (isset($_POST['cantidad']) && is_numeric($_POST['cantidad'])) ?
$_POST['cantidad'] : 0;
            break;
        case "mov0003":
            $seleccionada = $peliculas["mov0003"];
            $cantidad = (isset($_POST['cantidad']) && is_numeric($_POST['cantidad'])) ?
$_POST['cantidad'] : 0;
            break;
        case "mov0004":
            $seleccionada = $peliculas["mov0004"];
            $cantidad = (isset($_POST['cantidad']) && is_numeric($_POST['cantidad'])) ?
$_POST['cantidad'] : 0;
            break;
        case "mov0005":
            $seleccionada = $peliculas["mov0005"];
            $cantidad = (isset($_POST['cantidad']) && is_numeric($_POST['cantidad'])) ?
$_POST['cantidad'] : 0;
            break;
        case "mov0006":
            $seleccionada = $peliculas["mov0006"];
            $cantidad = (isset($_POST['cantidad']) && is_numeric($_POST['cantidad'])) ?
$_POST['cantidad'] : 0;
            break;
        case "mov0007":
            $seleccionada = $peliculas["mov0007"];
            $cantidad = (isset($_POST['cantidad']) && is_numeric($_POST['cantidad'])) ?
$_POST['cantidad'] : 0;
            break;
        case "mov0008":
            $seleccionada = $peliculas["mov0008"];
            $cantidad = (isset($_POST['cantidad']) && is_numeric($_POST['cantidad'])) ?
$_POST['cantidad'] : 0;
            break;
        case "mov0009":
            $seleccionada = $peliculas["mov0009"];
            $cantidad = (isset($_POST['cantidad']) && is_numeric($_POST['cantidad'])) ?
$_POST['cantidad'] : 0;
            break;
        case "mov0010":
            $seleccionada = $peliculas["mov0010"];
            $cantidad = (isset($_POST['cantidad']) && is_numeric($_POST['cantidad'])) ?
$_POST['cantidad'] : 0;
            break;
    }
    //Iniciando la tabla con la información de la película seleccionada
?>
<section>
<article>

```

```

<table id="hor-zebra" summary="Datos recibidos del formulario">
  <caption>Información de formulario</caption>
  <thead>
    <tr class="thead">
      <th scope="col" colspan="2">Boleto de entrada</th>
    </tr>
  </thead>
  <tbody>
<?php
  //Obteniendo los datos de la película seleccionada que está almacenada en $seleccionada
  $table = "";
  $i = 1;
  foreach($seleccionada as $indice => $dato){
    if($i%2 != 0){
      $table .= "\t<tr class=\"odd\">\n";
    }
    else{
      $table .= "\t<tr class=\"even\">\n";
    }
    if($indice == "precio"){
      $dato *= $cantidad;
      $dato = "\$" . number_format((double)$dato, 2, ".", ",");
    }
    $table .= "\t\t<td>$indice</td>\n";
    $table .= "\t\t<td> $dato</td>\n\t</tr>\n";
    $i++;
  }
  echo $table;
}
?>
    </tbody>
  </table>
  <a href="cartelera.php">Ingresar otra película</a>
</article>
</section>
</body>
</html>

```

**Resultado en el navegador:**

**Cartelera de cine**

**CINE REFORMA**  
Los mejores estrenos los encuentras aquí  
Estas son las películas en cartelera

**Entrada al cine**

**Pelicula**  
El efecto mariposa ▼

**Cantidad**  
3

**COMPRAR**

**Información de formulario**

Boleto de entrada	
titulo	El efecto mariposa
precio	\$11.25
sala	sala 3
hora	6:00 pm

## Resultado personal:

### Cartelera de cine



#### CINE REFORMA

Los mejores estrenos los encuentras aquí  
Estas son las películas en cartelera

**Entrada al cine**

**Pelicula**

Milagros inesperados ▾

**Cantidad**

2

**COMPRAR**

**Estudiante:** José Adrián López Medina - LM242664  
**Técnico en Ingeniería en Computación - Escuela de Computacion Ciclo I - 2025**

Información de formulario	
Boleto de entrada	
título	Milagros inesperados
precio	\$7.50
sala	Sala 2
hora	1:00 pm

[ingresar otra película](#)

**Estudiante:** José Adrián López Medina - LM242664

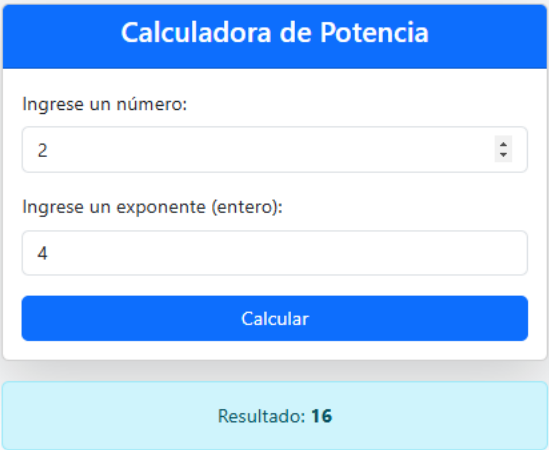
**Técnico en Ingeniería en Computación - Escuela de Computacion Ciclo I - 2025**



## V. DISCUSION DE RESULTADOS

1. Realice un script PHP que mediante un formulario que solicite dos números, el primero de ellos entero o con parte decimal y el segundo necesariamente entero, calcule la potencia de elevar el primer número ingresado a la potencia dada por el segundo número. No puede utilizar la función pow() para resolver este problema, debe resolverlo haciendo uso de un procedimiento que utilice un ciclo o lazo en donde aproveche la característica de que un número elevado a una potencia es igual a multiplicar ese número por si mismo tantas veces como indique la potencia. Por ejemplo:  $5^2 = 5 * 5 = 25$ , o  $3^4 = 3 * 3 * 3 * 3 = 81$ .

### Resultado personal:



The screenshot shows a web form titled "Calculadora de Potencia". It has two input fields: "Ingresa un número:" with the value "2" and "Ingresa un exponente (entero):" with the value "4". Below these is a blue "Calcular" button. At the bottom, a light blue box displays "Resultado: 16".

Estudiante: José Adrián López Medina - LM242664  
Técnico en Ingeniería en Computación - Escuela de Computacion Ciclo I - 2025

2. Cree un script que le permita ingresar un número del 1 al 10 a través de un formulario, que solamente deberá contener un campo de texto, su etiqueta y un botón de envío. El script PHP que realizará deberá mostrar la tabla de multiplicar de ese número de forma ordenada y utilizando hojas de estilo para una buena apariencia visual.

**Resultado:**

The screenshot shows a web interface for generating a multiplication table. At the top is a green header with the text "Tabla de Multiplicar". Below the header, there is a label "Ingrese un número (1 - 10):" followed by a text input field containing the number "3". A green button labeled "Generar Tabla" is positioned below the input field. Below the button, the resulting multiplication table is displayed, titled "Tabla del 3". The table lists the products of 3 multiplied by numbers from 1 to 10.

Tabla del 3	
3 x 1 =	3
3 x 2 =	6
3 x 3 =	9
3 x 4 =	12
3 x 5 =	15
3 x 6 =	18
3 x 7 =	21
3 x 8 =	24
3 x 9 =	27
3 x 10 =	30

**Estudiante: José Adrián López Medina - LM242664**  
**Técnico en Ingeniería en Computación - Escuela de Computacion Ciclo I - 2025**

3. Realice un script PHP que dadas tres notas de un listado de estudiantes organizadas en una matriz asociativa, le permita obtener el promedio de ellas sabiendo que las notas son de una tarea, una investigación y el examen parcial con la ponderación de 50%, 30% y 20% respectivamente. Por último, muestre las notas promediadas de cada alumno en una tabla bien presentable, mostrando el nombre del alumno, la nota que tiene en cada actividad y en una celda combinada su promedio.

SUGERENCIA: Haga una matriz con la siguiente estructura para facilitar el proceso:

```
$notas = array(  
    "nombre Alumno1" => array(  
        "Parcial" => 9.1,  
        "Investigación" => 8.5  
        "Tarea" => 4.8  
    ), ...
```

Resultado personal:

### Registro de Notas

#### Estudiante 1

Nombre:

Tarea (50%):

Investigación (30%):

Examen (20%):

Agregar Estudiante

Calcular Promedios

### Resultados

Nombre	Tarea (50%)	Investigación (30%)	Examen (20%)	Promedio
Adrian	8	7	10	8.1

**Estudiante: José Adrián López Medina - LM242664**

**Técnico en Ingeniería en Computación - Escuela de Computacion Ciclo I - 2025**

## VI. INVESTIGACION COMPLEMENTARIA

1. Investigue la utilización de las funciones `include_once()` y `require_once()` e indique la diferencia o diferencias que tienen con respecto a sus homólogas `include()` y `require()`.

Las funciones `include_once()` y `require_once()` en PHP son versiones mejoradas de `include()` y `require()`, respectivamente, con la principal diferencia de que garantizan que un archivo solo se incluya una vez, evitando duplicaciones que puedan causar errores en el código.

### Diferencias entre dichas funciones:

- **`include()`:** Incluye y evalúa el archivo especificado. Si el archivo no se encuentra, PHP genera un warning pero el script continúa. Puede incluir el mismo archivo múltiples veces.
- **`require()`:** Similar a `include()`, pero si el archivo no se encuentra, PHP genera un error fatal y detiene la ejecución del script. También permite múltiples inclusiones del mismo archivo.
- **`include_once()`:** Igual que `include()`, pero evita incluir el archivo más de una vez dentro del mismo script. Garantiza que el archivo se incluya solo una vez.
- **`require_once()`:** Igual que `require()`, pero evita la inclusión múltiple del mismo archivo. Asegura que el archivo solo se cargue una vez, evitando errores de redefinición.

2. Investigue sobre el uso de funciones anónimas en PHP, indicando los motivos por los cuáles pueden ser utilizadas e implemente un ejemplo simple de cada caso.

Las funciones anónimas (o closures) en PHP son funciones que no tienen un nombre asignado y generalmente se usan como valores de variables o argumentos de otras funciones. Se pueden emplear en varios casos, como callbacks, almacenamiento en variables, y funciones dentro de otras funciones.

### Ejemplos de uso de cada caso:

1. **Como callback:** Se utilizan comúnmente en funciones como `array_map()`, `array_filter()` o `usort()`.

```
$numeros = [1, 2, 3, 4, 5];

// Multiplicar cada número por 2
$duplicados = array_map(function($num) {
    return $num * 2;
}, $numeros);

print_r($duplicados);
```

2. **Almacenar una variable:** Permite reutilizar la función sin declararla explícitamente.

```
$suma = function($a, $b) {
    return $a + $b;
};

echo $suma(5, 3); // 8
print_r($duplicados);
```

3. **Uso dentro de otra función:** Útil para pasar lógica personalizada a funciones.

```
function operacion($a, $b, $funcion) {  
    return $funcion($a, $b);  
}  
  
$resultado = operacion(10, 5,  
    function($x, $y) {  
        return $x - $y;  
    });  
  
echo $resultado; // 5
```

4. **Uso de use para acceder a variables externas:** La variable \$factor es capturada dentro de la función anónima con use.

```
$factor = 3;  
  
$multiplicar = function($num) use  
($factor) {  
    return $num * $factor;  
};  
  
echo $multiplicar(4); // 12
```

## **VII. BIBLIOGRAFIA**

- Cabezas Granado, Luis Miguel. PHP 6 Manual Imprescindible. 1ra. Edición. Editorial Anaya Multimedia. 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.
- Gil Rubio, Francisco Javier/Villaverde, Santiago Alonso/Tejedor Cerbel, Jorge A. Creación de sitios web con PHP 5. Editorial McGraw-Hill. 1ra edición. Madrid, España, 2006.
- John Coggeshall. La Biblia de PHP 5. 1ra Edición. Editorial Anaya Multimedia. Madrid España.
- <http://www.php.net/manual/en>