



# UNIVERSIDAD DON BOSCO

## FACULTAD DE INGENIERIA

### ESCUELA DE COMPUTACION

CICLO 1	GUIA DE LABORATORIO #4	
	Nombre de la Practica:	Funciones en PHP
	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

Que el estudiante:

- Adquiera el dominio de la sintaxis y el uso de funciones con el lenguaje PHP.
- Logre la habilidad necesaria para declarar encabezados de funciones con o sin argumentos y con devolución o no de valores.
- Haga uso de características avanzadas de las funciones como la lista variable de argumentos, las funciones variables y recursividad.
- Haga uso de la modularidad para crear scripts que serán incluidos dentro de un script principal.

## II. INTRODUCCION TEORICA

### Funciones en PHP

Una **función** es un bloque de código independiente y autónomo que contiene un grupo de instrucciones que se identifican con un nombre. Las funciones pueden invocarse todas las veces que se requiera desde cualquier punto de un *script*. El código de las funciones puede aparecer dentro del *script* que se esté realizando o puede ser parte de un *script* independiente que será llamado como archivo de inclusión (otro *script* PHP invocado).

Generalmente, cuando se realizan *scripts* con PHP las instrucciones se ejecutan conforme van siendo procesadas por el intérprete del lenguaje. Sin embargo, a la hora de realizar *scripts* más complejos, hay ocasiones en las que el mismo código ha de ser ejecutado más de una vez. Para estos casos sería útil que el lenguaje de programación permitiera dividir el código en segmentos más pequeños, de forma que cada bloque de código PHP pudiera ser agrupado bajo un identificador para que pudiera ser accedido de forma independiente.

PHP permite la creación de funciones que permiten recoger todos los aspectos planteados anteriormente. Una **función** permite desarrollar una **tarea concreta y bien definida**. Se encuentra separada del resto de instrucciones del programa y se le asigna un nombre para que posteriormente pueda ser llamada desde otro punto del *script*, o incluso, desde otro *script*. Es a través del nombre de la función que se pueden ejecutar las instrucciones contenidas en la función tantas veces como sea necesario.

La utilización de funciones permite que un script aparezca escrito como una lista de referencias a las tareas que se deben hacer para crear una página de respuesta.

Las funciones también pueden o no, aceptar parámetros o argumentos externos de los que dependa el proceso que deba realizarse para posteriormente devolver un valor.

### Sintaxis para definir encabezado de una función

La sintaxis se muestra a continuación:

```
function nombre_funcion([$param1, $param2, ... , $paramn]){  
    //bloque de instrucciones PHP  
}
```

En su forma más compleja, una función se declara con la palabra reservada *function*. A continuación, se debe declarar el nombre de la función y posteriormente, y entre paréntesis, cada uno de los argumentos que recibirá separados por coma. En el caso más simple, sólo se colocan los paréntesis vacíos inmediatamente después del nombre de la función. Después de cerrar los paréntesis debe abrir llaves y colocar las instrucciones que realizará la función para luego cerrar la llave.

Como se puede ver, son necesarios los siguientes componentes en la declaración de una función:

- ✓ La palabra reservada ***function*** que debe utilizarse para indicar al intérprete de PHP que se va a crear una nueva función
- ✓ ***nombre\_funcion*** indica el nombre con el que se va a identificar la función dentro del *script* PHP para su posterior llamada.
- ✓ ***\$param1, \$param2, ... , \$paramn*** representan los parámetros necesarios para que la función pueda ser ejecutada. Los parámetros han de expresarse siempre entre paréntesis y separados por comas. Incluso si la función no necesitara ningún parámetro, deberán utilizarse los paréntesis.
- ✓ **Bloque de instrucciones PHP**, representa el conjunto de sentencias o instrucciones que se van a ejecutar cada vez que se haga una llamada a la función.

### Argumentos o parámetros de la función

El diseño de una función puede o debería incluir la aceptación opcional de uno o más argumentos o parámetros que representarán valores que se pasan desde el exterior a la función. Esto permite que la función sea independiente del exterior, haciéndola más portable.

Se puede entender un argumento o parámetro como una variable con ámbito local a la función que sirve para almacenar valores que son pasados a la función.

Los argumentos o parámetros se especifican colocando nombres de variables entre los paréntesis de la definición de la función:

```
function myfunction($parametro1, $parametro2){  
    //Código de la función  
}
```

### Devolución de valores

Casi siempre será importante que la función creada devuelva un valor antes de finalizar la ejecución.

Aunque también existirán casos en los que la función pudiera realizar una tarea sin que devuelva valor

alguno. La palabra reservada utilizada para devolver valores en las funciones de PHP es *return*, igual que en otros lenguajes que usted ya conoce. Veamos los siguientes ejemplos:

```
function square ($num) {  
    return $num * $num;  
}  
  
echo square(4); // obtiene '16'.  
-----  
function residuo($dividendo, $divisor){  
    $res = $dividendo % $divisor;  
    return $res;  
}
```

No se puede devolver múltiples valores desde una función, pero un efecto similar se puede conseguir devolviendo una lista.

```
function small_numbers(){  
    return array(0, 1, 2);  
}  
  
list($zero, $one, $two) = small_numbers();
```

### Apunte importante:

- Recuerde que a los **valores que se le aportan a una función desde el exterior** se les denomina **parámetros** o **argumentos** de la función. Los **parámetros** actúan como variables dentro de la función. Estas variables no pueden utilizarse fuera de ese ámbito.
- El valor que devuelva la función debe ser enviado utilizando la palabra reservada ***return*** y, a continuación, **el nombre de la variable que contiene el resultado** que se desea retornar al punto de llamada.

## Ventajas del uso de funciones

### a) Simplificación de código

Las funciones hacen que el código de un script sea más fácil de leer, por tanto, lo hacen más entendible y más agradable a la vista del que tenga que leer el código. Esto es, porque evita el tener que repetir varias veces un segmento de código dentro de un mismo script. En lugar de ello se escribe una sola vez y luego dentro del script solamente hacemos la llamada a la función, en lugar de volverlo a escribir.

### b) Reutilización de código

Permite que otros scripts, a parte del script donde se digitó el código de la función, puedan hacer uso de dicha función. Esto evita la carga del programador de tener que escribir la misma funcionalidad en diferentes scripts.

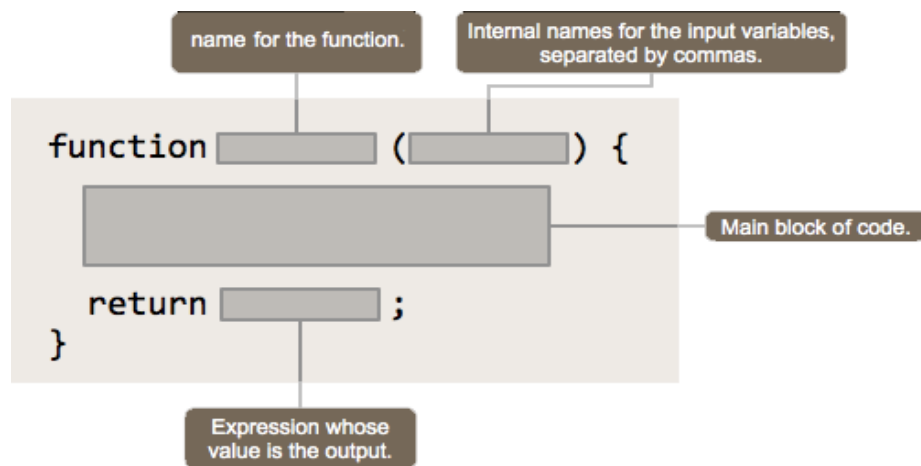
### c) Modularidad

El uso de funciones permite que las modificaciones al código sean menos trabajosas para el programador, porque en lugar de tener que cambiar el mismo código en varios lugares de un script, solamente se modifica el código de la función y los scripts que la llaman no sufren modificación alguna (al menos, en la mayor parte de los casos).

### Funcionamiento de una función

Aunque el comportamiento puede variar en algunos casos, la mayoría de las funciones siguen un proceso como el siguiente, una vez que son invocadas:

1. Aceptan uno, o más, valores, conocidos como argumentos, desde el script que la llama,
2. Realizan un proceso utilizando dichos valores (argumentos o parámetros), y
3. Devuelven el resultado y el control del programa al script que hizo la llamada.



Aspecto de una función en PHP

### Características de las funciones de PHP (versión 4.2.X en adelante)

1. PHP no hace distinción entre mayúsculas y minúsculas para los nombres de las funciones. Pese a ello se recomienda ser consistente en el código que se escribe de tal forma que si se nombró a una función Cambiar, utilizar en el código siempre Cambiar, y no utilizar indistintamente, cambiar, CAMBIAR, CAMbiar, etc.
2. Al igual que con las variables, podemos hacer referencia (una llamada) a una función antes de definirla en el código. Existe una excepción, que se da cuando la función ha sido definida condicionalmente. En ese caso la definición debe preceder a la llamada.
3. No se puede redefinir una función o eliminar su definición. Esto se debe a que PHP no soporta sobrecarga de funciones como otros lenguajes de programación. Por ejemplo C++.
4. Las últimas versiones de PHP, (4 o superior) soportan el uso de lista variable de argumentos en las funciones.
5. Se pueden utilizar parámetros por defecto en las funciones asignándole un valor inicial dentro de la definición de los parámetros de la función.

## Tipos de funciones

En PHP podemos utilizar las **funciones incorporadas del lenguaje** (abs(), list(), array(), chr(), intval(), trim(), substr(), printf(), time(), date(), etc.) o pueden realizarse **funciones definidas por el usuario**.

Las **funciones incorporadas del lenguaje** se comportan de la forma predeterminada para la que fueron creadas. Existen muchas funciones incorporadas o predefinidas en PHP. Algunas ya las hemos utilizado en guías anteriores. Por ejemplo: gettype(), settype(), print(), printf(), substr(), date(), etc.

Las **funciones definidas por el usuario**, o personalizadas, son creadas por el programador y su funcionalidad y utilidad dependen en gran medida de la habilidad del programador. La ventaja de las segundas es que el programador posee un control completo sobre ellas, ya que puede hacer que una función se comporte exactamente del modo que se desea.

## Parámetros de las funciones

La información puede suministrarse a las funciones mediante la lista de parámetros, una lista de variables y/o constantes separadas por comas.

PHP soporta pasar parámetros por valor (el comportamiento por defecto), por referencia, y parámetros por defecto. Listas de longitud variable de parámetros sólo están soportadas en PHP4 y posteriores. Un efecto similar puede conseguirse en PHP3 pasando un array de parámetros a la función:

```
function takes_array($input) {  
    echo "$input[0] + $input[1] = ", $input[0]+$input[1];  
}
```

## Pasar parámetros por referencia

Por defecto, los parámetros de una función se pasan **por valor** (de manera que si se cambia el valor del argumento dentro de la función, el valor pasado fuera de ella no se ve alterado). Si se desea permitir a una función modificar el valor de sus parámetros, deben ser pasados por referencia.

Para hacer que una función pase sus parámetros por referencia existen dos métodos:

- ★ Anteponiendo un símbolo de ampersand (&) al nombre del parámetro o argumento en la definición de la función. Esto hará que siempre que se mande un parámetro a la función en la llamada, este será enviado por referencia. Por ejemplo:

```
function add_some_extra(&$string){  
    $string .= ' y algo más.';  
}  
$str = 'Esto es una cadena, '  
add_some_extra($str);  
echo $str; // Saca 'Esto es una cadena, y algo más.'
```

- ★ Anteponer el símbolo de ampersand (&) al nombre del parámetro en la llamada a la función. Este método se debe aplicar si se desea pasar una variable por referencia a una función que no toma el parámetro por referencia por defecto; es decir, si no se especificó en la definición de la función.

```
$num = 10;  
function called_function($number){ //En la declaración no indica que  
    //el parámetro sea por referencia  
    $number = $number +1;
```

```

    echo $number, "\n";
}
called_function(&$num); //Se le pasa argumento por referencia a la función
echo $num, "\n";

```

## Parámetros por defecto

Una función puede definir valores por defecto para los parámetros escalares estilo C++. Esto significa que si al llamar a una función, que requiere parámetros, estos no se especifican, se le asignará a estos un valor por defecto. Veamos el siguiente ejemplo:

```

function makecoffee($type = "cappucino"){
    return "Hacer una taza de $type.\n";
}
echo makecoffee(); //Se llama a la función sin indicar parámetro
echo makecoffee("espresso"); //Se llama a la función usando
parámetro

```

La salida del fragmento anterior es:

```

Hacer una taza de cappucino.
Hacer una taza de espresso.

```

El valor por defecto tiene que ser una **valor literal**, y no una variable o miembro de una clase.

En PHP 4.0 también es posible especificar **unset** como parámetro por defecto. Esto significa que **el argumento no tomará ningún valor** en absoluto si el valor no es suministrado.

Destacar que cuando se usan parámetros por defecto, estos tienen que estar a la derecha de cualquier parámetro sin valor por defecto; de otra manera las cosas no funcionarán de la forma esperada. Considera el siguiente fragmento de código:

```

function makeyogurt ($type = "acidophilus", $flavour) {
    return "Haciendo un bol de $type $flavour.\n";
}
echo makeyogurt ("mora"); // No funcionará de la manera esperada

```

La salida del ejemplo anterior es:

```

Warning: Missing argument 2 in call to makeyogurt() in usr/local/etc/httpd/htdocs/php3test/functest.html on line 41
Haciendo un bol de mora.

```

Y ahora, compáralo con:

```

function makeyogurt ($flavour, $type = "acidophilus") {
    return "Haciendo un bol de $type $flavour.\n";
}
echo makeyogurt ("mora"); // funciona como se esperaba

```

La salida de este ejemplo es:

Haciendo un bol de acidophilus mora.

### Funciones variables

En la mayoría de casos las llamadas a función se realizan con el identificador de la función de forma directa, sin embargo, en PHP es posible asignar el identificador a una variable de cadena y hacer que en la llamada a función se utilice esta variable en lugar del nombre o identificador de la función. Por ejemplo:

```
$funcion = "potencia";  
$valor = 3;  
echo "$valor elevado al cuadrado es: " . $funcion(3) . "<br />";  
  
//Definición de la función sqrt  
function sqrt($numero){  
    return $numero * $numero;  
}
```

Aunque este ejemplo es bastante trivial, el verdadero potencial de las funciones variables se puede experimentar al realizar retrollamadas (callbacks), definiendo una matriz (array) con los nombres de varias funciones haciendo que un ciclo o lazo recorra los nombres de las funciones para que realice cada una el cálculo con uno o varios valores con las distintas funciones definidas. Por ejemplo:

```
$trigFunctions = array("sin", "cos", "tan"); //Funciones predefinidas PHP  
$degrees = 30;  
foreach($trigFunctions as $myfunction){  
    echo "\$myfunction(\$degrees) = " . $myfunction(deg2rad($degrees))  
    . "<br />";  
}
```

### Alcance de las variables en PHP

La palabra alcance o ámbito hace referencia a las partes del programa que pueden tener acceso a una variable. La vida de una variable es el tiempo durante el cual existe.

En PHP las variables pueden tener tres tipos de alcance:

a) **Local.**

Las **variables locales** son visibles dentro de una función y su vida acaba justo cuando la ejecución de la función termina.

Por defecto, todas las variables que se declaran dentro de una función son locales en la función. Esto significa que estas variables no son accesibles desde fuera de la función y mueren cuando la función termina.

### Ejemplo:

```
<?php
//Variable global, visible afuera de las funciones
$friend = "Carlos";
function who(){
    //Variable local que muere cuando acaba la función
    $friend = "Juan";
    printf("<p>Dentro de la funci&ocute;n tu amigo es: %s.</p>", $friend);
}
//Llamada a la función who()
who();
printf("<p>Fuera de la funci&ocute;n tu amigo es: %s.</p>", $friend);
?>
```

Al ejecutar el script obtenemos lo siguiente:

Dentro de la función tu amigo es: Juan.

Fuera de la función tu amigo es: Carlos.

#### b) Global.

Las **variables globales** son visibles en todo un script, y normalmente no lo son para una función. Dentro de una función no se puede acceder a las variables que se han declarado fuera de ella, a menos que la pasemos por referencia a la función en forma de argumento. Otra forma de hacer que una función acceda a una variable externa a la función es utilizando la palabra reservada global antes del nombre de la variable.

#### Ejemplo: Usando argumento por referencia

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Argumento a funci&ocute;n</title>
</head>
<body>
<?php
    //function raise_sal(&$salary){
    function raise_sal(&$salary){
        //global $salary;
        $salary *= 1.1;
    }
    $salary = 50000;
    //raise_sal();
    raise_sal($salary);
?>
&iexcl;Felicitades!, su nuevo salario es: . <?= "$" . $salary ?><br />
</body>
</html>
```



Al ejecutar el script obtenemos lo siguiente:

¡Felicidades!, su nuevo salario es: . \$55000

### Ejemplo

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Argumento a funci&ocirc;n</title>
</head>
<body>
<?php
    function raise_sal(){
        $GLOBALS['salary'] *= 1.1;
    }
    $salary = 50000;
    raise_sal();
?>
&excl;¡Felicidades!, su nuevo salario es: . <?= "$" . $salary ?><br />
</body>
</html>
```

El resultado es:

¡Felicidades!, su nuevo salario es: . \$55000

### c) Estático.

Las **variables estáticas** son locales en una función, pero retienen su valor entre distintas llamadas a la misma. Una variable estática no es visible fuera de la función, pero no muere cuando la función acaba. Cuando se inicializa una variable estática, esta no pierde su valor entre las distintas llamadas, recuerda el valor que tenía en la llamada anterior. Estas variables mueren hasta que termina el script.

Ejemplo:

```
<?php
function trackname(){
    static $count = 0;
    $count++;
    echo "<p>Has ingresado a esta p&aacute;gina $count veces</p>";
}
//Invocar varias veces a la función trackname()
trackname();
trackname();
trackname();
trackname();
?>
```

El resultado es:

Has ingresado a esta página 1 veces

Has ingresado a esta página 2 veces

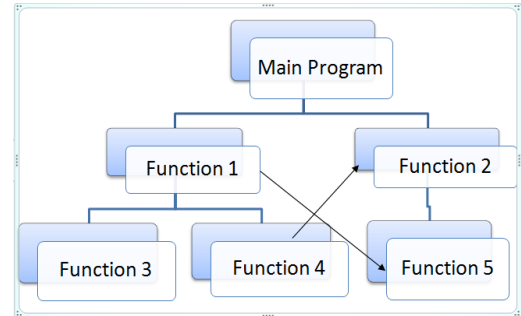
Has ingresado a esta página 3 veces

Has ingresado a esta página 4 veces

## Funciones incorporadas de PHP

PHP proporciona un gran número de funciones incorporadas, que ya están listas para su utilización inmediata.

Lo único que debe hacerse es invocarlas, para lo que necesitará conocer su sintaxis, los argumentos que se requieren para hacer la llamada a la función y el tipo de valor que devuelve para asignarlo a alguna variable u operar con el resultado en el punto de llamada a la función dentro del script principal.



Entre los tipos de funciones más utilizados están:

- Funciones para manejo de caracteres.

En PHP existen funciones que permiten trabajar con cadenas de caracteres que podemos dividir en varias subcategorías.

Por ejemplo, funciones para **limpiar espacios en blanco**, como trim(), ltrim(), rtrim() o chop(), funciones para cambiar de mayúsculas a minúsculas y viceversa, como strtolower() y strtoupper(), funciones para obtener partes de una cadena o un indicador de posición de coincidencia de una subcadena dentro de otra cadena; es decir obtener subcadenas de una cadena más larga, como substr(), strpos(), strstr(), explode(), etc.

- Funciones de búsqueda, reemplazo y comparación en cadenas.

Estas funciones permiten realizar búsqueda de una cadena dentro de otra más larga, reemplazar ciertos caracteres de una cadena por otros o comparar una cadena con otra, como htmlentities(), substr\_replace(), str\_replace(), strcmp(), strncmp(), strcasecmp(), etc.

- Funciones de fecha y hora.

Estas funciones permiten trabajar con fechas en PHP, ya sea para obtener algo conocido como la marca de tiempo de la era UNIX o para mostrar datos específicos de una fecha y hora, como el día, el mes, el año, la hora, los minutos y hasta los segundos.

Las funciones de fecha de PHP son, entre otras: time(), mktime(), date(), checkdate(), date\_default\_timezone\_get(), date\_default\_timezone\_set(), etc.

- Funciones para manejo de archivos y directorios.

Otro conjunto de funciones útiles, en este caso para el manejo de archivos, son las funciones: fopen(), fread(), fwrite(), feof(), fgetc(), fgets(), fputs(), fseek(), rewind(), ftell(), fclose(), etc.

- Funciones para manejo de expresiones regulares.

También existen algunas funciones de PHP relacionadas con el manejo de expresiones regulares. Entre estas se pueden mencionar: preg\_match(), preg\_match\_all(), preg\_replace(), preg\_split(), etc.

- Funciones para trabajo con bases de datos.

Las funciones para trabajar con bases de datos en PHP son diversas, dependiendo del tipo de gestor de bases de datos con el que se desea conectar. Es así que PHP proporciona un amplio número de funciones para trabajar con bases de datos MySQL, PostgreSQL, MSSQL e incluso Oracle, entre otras. En el caso de MySQL se puede trabajar usando las extensiones de MySQLi que son las más recomendadas por su rapidez como `mysqli_connect()`, `mysqli_query()`, `mysqli_fetch_array()`, `mysqli_fetch_assoc()`, `mysqli_fetch_object`, `mysqli_errno()`, etc.

## Modularidad

En PHP es posible hacer uso de la característica de modularidad utilizando las sentencias `include()` o `require()` para implementar la reutilización del código definido en archivos con código PHP. La sintaxis de estas sentencias son las siguientes:

```
include('nombre_de_archivo');  
-----  
require('nombre_de_archivo');
```

**NOTA:** Los paréntesis son opcionales. Si no se incluyen debe dejar espacio entre la palabra reservada `include|require` y el delimitador de cadena, que puede ser comilla simple o doble.

En la sintaxis anterior ‘`nombre_de_archivo`’ es el nombre del archivo que se desea incluir o evaluar dentro del script que utiliza esta sentencia.

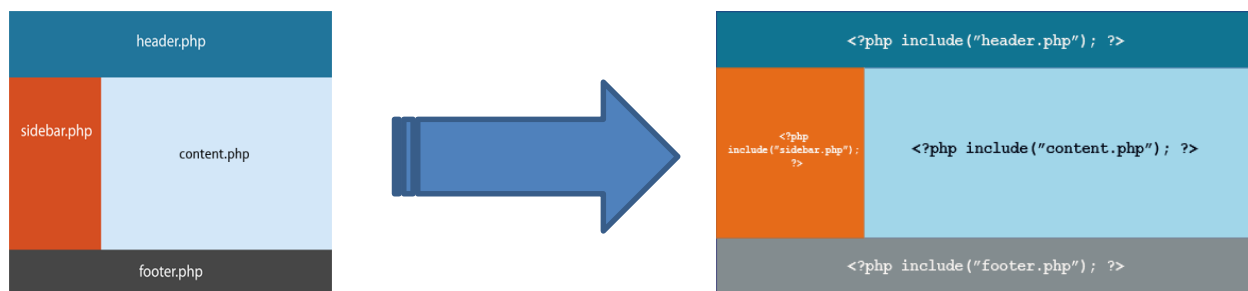
Las instrucciones `include()` y `require()` son prácticamente idénticas, lo único que las diferencia es la forma en que procesan los errores; es decir, mientras que `include()` devuelve una advertencia (`warning`) cuando se produce un fallo por no encontrar el archivo que se pretende incluir, `require()` devolverá un error fatal al producirse el mismo problema.

Existen dos variantes de estas funciones que se denominan `include_once()` y `require_once()`, respectivamente. Lo que las diferencia de las anteriores es que evitan un posible conflicto en caso de intentar incluir varias veces el mismo archivo.

Un de las aplicaciones más útiles de estas funciones o de sus variantes es proporcionar modularidad en los programas o secuencias de comando PHP. También se pueden utilizar para incluir librerías de funciones o bibliotecas de clase.

A manera de ejemplo concreto, puede imaginar una página que se divide en varias secciones lógicas, que luego son invocadas por un script principal que invoca a todas las partes para armar toda la página.

Observe la imagen:



La modularidad permite que se pueda dividir un *script* en *scripts* más pequeños y manejables que permitan realizar modificaciones de forma más rápida y ordenada; es decir, si en algún momento debe hacerse un cambio importante en un sitio creado con PHP, la modularidad permitiría modificar lo menos posible los scripts de la aplicación. Debería ser en la mayoría de los casos un sólo script el que debería de modificarse. Esto es una enorme ventaja para el programador.

### III. MATERIALES Y EQUIPO

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

No.	Material	Cantidad
1	Guía de práctica #4: Funciones en PHP	1
2	Computadora con WampServer, Sublime Text y PHP Designer instalados	1
3	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* cargando la página en el navegador de su preferencia.

**Ejercicio #1:** El siguiente ejemplo muestra cómo generar una tabla de multiplicar solicitando el número de la tabla al usuario a través de un formulario, luego, haciendo uso de una función se obtiene el valor ingresado y se crea la tabla de multiplicar de ese número.

Note que esta función no recibe argumentos ni retorna valor alguno. Se valida el valor numérico que el usuario ingresa haciendo uso de validación en el cliente con JavaScript.

#### Archivo 1: entrada.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8" />
  <title>Introducción de datos</title>
  <link rel="stylesheet" href="css/form.css" />
  <link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Days+One" />
```

```

    <script src="js/modernizr.custom.lis.js"></script>
    <script src="js/validar.js"></script>
</head>
<body>
<header id="D3">
    <h1>Ejemplo de uso de Función</h1>
</header>
<section>
<article>
<form method="POST" action="tabla.php" name="frmingreso" id="frmingreso" class="elegant-aero">
    <h1>
        Cálculo del
        <span>factorial de un número</span>
    </h1>
    <ul>
        <li class="campo">
            <label for="txtnumero">Número: </label>
            <input type="text" name="txtnumero" id="txtnumero" maxlength="3"
placeholder="(Ingresa un número entero)" pattern="[0-9]{1,3}" required="required" /><br />
            <span id="numbersOnly">Ingresa números enteros</span>
        </li>
        <li class="campo">
            <input type="submit" name="enviar" value="Mostrar" class="button" />&nbsp;
            <input type="reset" name="limpiar" value="Cancelar" class="button" />
        </li>
    </ul>
</form>
</article>
</section>
</body>
</html>

```

## Archivo 2: tabla.php

```

<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <title>Tablas de multiplicar</title>
    <link rel="stylesheet" href="css/tabla.css" />
    <link rel="stylesheet" href="css/links.css" />
</head>
<body>
<section>
    <article>
<?php
    function escribir_tabla(){
        $num = (isset($_POST['txtnumero']) && is_int(intval($_POST['txtnumero']))) ?
$_POST['txtnumero'] : 0;

```

```

        if($num != 0){
            echo "<div id=\"tablamulti\">\n";
                echo "<table>\n";
            echo "\t<caption>Tabla del $num</caption>\n";
            echo "\t<thead>\n\t\t<tr>\n\t\t\t<th colspan=\"3\">\n";
            echo "\t\t\t\tTabla del número $num\n";
            echo "\t\t\t\t<th>\n\t\t\t\t<tr>\n\t\t\t\t</thead>\n";
            echo "\t<tbody>\n";
            for($i=1; $i<=20; $i++){
                echo "\t\t\t<tr>\n";
                echo "\t\t\t\t<td>\n\t\t\t\t\t", $num, "&nbsp;*&nbsp;", $i, "\t\t\t\t</td>\n";
                echo "\t\t\t\t<td>\n\t\t\t\t\t = \t\t\t\t</td>\n";
                echo "\t\t\t\t<td>\n\t\t\t\t\t", $num*$i, "\t\t\t\t</td>\n";
                echo "\t\t\t\t</tr>\n";
            }
            echo "\t</tbody>\n";
            echo "</table>\n";
            echo "</div>\n";
        }
        else{
            die(
                "<p class=\"error\">El valor ingresado debe ser un número entero mayor que
cero.</p>\n
                <div id=\"enlace\">\t\n<a href=\"entrada.html\">Regresar</a>\n</div>\n"
            );
        }
    }
    escribir_tabla();
    $link = "<a href=\"entrada.html\" class=\"a-btn\">";
    $link .= "\t<span class=\"a-btn-symbol\">i</span>";
    $link .= "\t<span class=\"a-btn-text\">Volver</span>";
    $link .= "\t<span class=\"a-btn-slide-text\">al formulario</span>";
    $link .= "\t<span class=\"a-btn-slide-icon\"></span>";
    $link .= "</a>";
    echo $link;
?>
</article>
</section>
</body>
</html>

```

El resultado visible en su navegador deber lucir así:

# Ejemplo de uso de Función

**Cálculo del**  
factorial de un número

Número:


5

Mostrar

Cancelar

### Tabla del 5

Tabla del número 5		
5 * 1	=	5
5 * 2	=	10
5 * 3	=	15
5 * 4	=	20
5 * 5	=	25
5 * 6	=	30
5 * 7	=	35
5 * 8	=	40
5 * 9	=	45
5 * 10	=	50
5 * 11	=	55
5 * 12	=	60
5 * 13	=	65
5 * 14	=	70
5 * 15	=	75
5 * 16	=	80
5 * 17	=	85
5 * 18	=	90
5 * 19	=	95
5 * 20	=	100

 **Volver**  
AL FORMULARIO

Resultado personal:

## Tabla del 5

Tabla del número 5		
5 * 1	=	5
5 * 2	=	10
5 * 3	=	15
5 * 4	=	20
5 * 5	=	25
5 * 6	=	30
5 * 7	=	35
5 * 8	=	40
5 * 9	=	45
5 * 10	=	50
5 * 11	=	55
5 * 12	=	60
5 * 13	=	65
5 * 14	=	70
5 * 15	=	75
5 * 16	=	80
5 * 17	=	85
5 * 18	=	90
5 * 19	=	95
5 * 20	=	100



**Volver**  
AL FORMULARIO

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



**Ejercicio #2:** El siguiente ejemplo muestra cómo generar la serie de Fibonacci solicitando al usuario el número de términos de la serie generada. Este dato es validado en el cliente y en el servidor.

**Archivo 1: fibonacci.php**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8" />
  <title>Serie de Fibonacci</title>
  <link rel="stylesheet" href="css/fonts.css" />
  <link rel="stylesheet" href="css/form.css" />
  <link rel="stylesheet" href="css/niko.css" />
</head>
<body>
<header id="vintage">
```

```

    <h1>Serie de Fibonacci con recursividad</h1>
</header>
<section>
<article>
<?php
    //Función de Fibonacci recursiva
    function fibonacci($n){
        if(($n == 0) || ($n == 1)) return $n;
        return fibonacci($n - 2) + fibonacci($n - 1);
    }

    if(isset($_GET['enviar'])):
        //Verificando el correcto ingreso de los datos
        $numero1 = isset($_GET['txtnumero']) ? $_GET['txtnumero'] : -1;
        if($numero1 == -1):
            $backlink = "\t<p>El valor del campo no es válido</p>\n";
            $backlink .= "\t<a class=\"a-btn\" href=\"{\$_SERVER['PHP_SELF']}\" 
title=\"Regresar\">\n";
            $backlink .= "\t\t<span class=\"a-btn-symbol\">J</span>\n";
            $backlink .= "\t\t<span class=\"a-btn-text\">Ingresar dato</span>\n";
            $backlink .= "\t\t<span class=\"a-btn-slide-text\">nuevamente</span>\n";
            $backlink .= "\t</a>\n";
            echo $backlink;
            exit(0);
        endif;
        $tabla = "<table>\n";
        $tabla .= "<caption>Generando serie de Fibonacci</caption>\n";
        $tabla .= "<thead>\n\t";
        $tabla .= "<tr>\n\t\t<th scope=\"col\">\n\t\t\tSecuencia\n\t\t\t</th>\n\t\t";
        $tabla .= "<th scope=\"col\">\n\t\t\tValor\n\t\t\t</th>\n\t";
        $tabla .= "<thead>\n";
        $tabla .= "<tbody>\n\t";
        $i = 0;
        //Con este contador se verifica que se generen el número
        //de términos en la serie que indicó el usuario en el formulario
        while($i <= $numero1){
            $class = $i%2 == 0 ? "odd" : "even";
            $tabla .= "<tr class=\"{$class}\">\n\t\t\t<td>\n\t\t\t\tF<sub>$i</sub>\n\t\t\t</td>\n\t";
            $tabla .= "\n\t\t\t<td>\n\t\t\t\t" . fibonacci($i) . "\n\t\t\t</td>\n\t</tr>\n";
            $i++;
        }
        $tabla .= "<tbody>\n</table>\n";
        echo $tabla;
        echo "<br />\n<a href=\"{\$_SERVER['PHP_SELF']}\" title=\"Regresar\" class=\"a-btn\">";
        echo "\t\t<span class=\"a-btn-symbol\">J</span>\n";
        echo "\t\t<span class=\"a-btn-text\">Ingresar</span>\n";
        echo "\t\t<span class=\"a-btn-slide-text\">nuevos datos</span></a>";
    else:
?>

```

```

<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="GET" class="elegant-aero">
<fieldset>
    <h1>
        Serie de
        <span>Fibonacci</span>
    </h1>
    <ul>
        <li class="campo">
            <label for="numero1">Valores para serie: </label>
            <input type="text" name="txtnumero" id="txtnumero" size="4" maxlength="3"
pattern="[0-9]{1,3}" required />
            <span id="numbersOnly">Ingrese números</span>
        </li>
        <li class="campo">
            <input type="submit" name="enviar" value="Generar" class="button" />&nbsp;
            <input type="reset" name="limpiar" value="Cancelar" class="button" />
        </li>
    </ul>
</fieldset>
</form>
<script src="js/validar.js"></script>
<?php endif; ?>
</article>
</section>
</body>
<script src="js/modernizr.custom.lis.js"></script>
</html>

```

Resultado de la ejecución en el navegador:



# SERIE DE FIBONACCI CON RECURSIVIDAD

Generando serie de Fibonacci

Secuencia	Valor
F <sub>0</sub>	0
F <sub>1</sub>	1
F <sub>2</sub>	1
F <sub>3</sub>	2
F <sub>4</sub>	3
F <sub>5</sub>	5
F <sub>6</sub>	8
F <sub>7</sub>	13
F <sub>8</sub>	21
F <sub>9</sub>	34
F <sub>10</sub>	55
F <sub>11</sub>	89
F <sub>12</sub>	144
F <sub>13</sub>	233
F <sub>14</sub>	377
F <sub>15</sub>	610

Resultado personal:

## SERIE DE FIBONACCI CON RECURSIVIDAD

Generando serie de Fibonacci

Secuencia	Valor
F <sub>0</sub>	0
F <sub>1</sub>	1
F <sub>2</sub>	1
F <sub>3</sub>	2
F <sub>4</sub>	3
F <sub>5</sub>	5
F <sub>6</sub>	8
F <sub>7</sub>	13
F <sub>8</sub>	21
F <sub>9</sub>	34
F <sub>10</sub>	55

Técnico  **Ingresar** NUEVOS DATOS **Urián López Medina - LM242664**  
Escuela de Computación - Escuela de Computación Ciclo I - 2025

**Ejercicio #3:** En este ejemplo se ingresan varias edades con validación en el cliente con JavaScript que permite ingresar solo edades numéricamente correctas. Al enviar los datos ingresados en un cuadro de lista se calcula en el servidor el promedio de las edades ingresadas y cada una de las edades que se ingresaron.

**Archivo 1: edades.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Edades de personas</title>
  <link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Oswald" />
  <link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Andika" />
  <link rel="stylesheet" href="css/edades.css" />
</head>
<body>
<section>
<article>
<form name="frmedades" action="promedioedad.php" method="POST">
<fieldset>
  <legend><span>Edades de personas:</span></legend>
  <ul>
    <li>
      <label for="edad">Ingreso de edades:</label>
      <div class="campo">
        <input type="text" name="edad" id="edad" maxlength="60" />
        <input type="reset" name="agregar" id="agregar" value="Agregar" /><br />
        <span id="numbersOnly">
          El campo sólo acepta números.
        </span>
      </div>
    </li>
  </ul>
</fieldset>
</article>
</section>
</body>
</html>
```

```

        </span>
    </div>
</li>
<li>
    <label for="ingresados">Edades ingresadas:</label>
    <div class="campo">
        <select name="ingresados[]" id="ingresados" size="6"
multiple="multiple"></select>
    </div>
</li>
<li>
    <div class="campo">
        <input type="submit" name="enviar" id="enviar" value="Enviar"
class="boton" />
    </div>
</li>
</ul>
</fieldset>
</form>
</article>
</section>
</body>
<script src="js/agregar.js"></script>
<script src="js/validatefield.js"></script>
<script src="js/modernizr.custom.lis.js"></script>
</html>

```

## Archivo 2: promedioedad.php

```

<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <title>Promedio de edades</title>
    <link rel="stylesheet" href="css/fonts.css" />
    <link rel="stylesheet" href="css/sticky-table.css" />
</head>
<body>
<section>
<article class="component">
<table>
    <thead>
        <th class="head">Edades ingresadas</th>
    </thead>
    <tbody>
<?php
    if(isset($_POST['enviar'])){
        if(isset($_POST['ingresados'])){
            calcularedadprom($_POST['ingresados']);

```

```

    }
    else{
        $msgerr = "No hay edades que procesar.";
        $edades = array($msgerr);
        calcularedadprom($edades);
    }
}

// Función para calcular el promedio de las edades
function calcularedadprom($edades){
    $promedades = 0;
    $contador = 0;
    $celdas = "";
    if(is_array($edades)){
        foreach($edades as $edad){
            $celdas .= "\n\t<tr>\n\t\t<td class=\"user-
name\">\n\t\t\t\t$edad\n\t\t</td>\n\t</tr>\n</tbody>\n";
            $promedades += $edad;
            $contador++;
        }
        $promedades /= $contador;
        $promedades = number_format($promedades, 2, ".", "");
        $celdas .= "<tfoot>\n";
        $celdas .= "<tr>\n\t\t<th>\n\t\t\t\tLa edad promedio es:
$promedades\n\t\t</th>\n\t</tr>\n";
        $celdas .= "</tfoot>\n";
        echo $celdas;
    }
    else{
        $celdas .= "\n\t<tr>\n\t\t<td class=\"user-
name\">\n\t\t\t\t$edades\n\t\t</td>\n\t</tr>\n</tbody>\n";
        echo $celdas;
    }
}
?>
</table>
</article>
<!-- Librerías de jQuery para hacer el efecto sticky-headers -->
<script src="//ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>
<script src="http://cdnjs.cloudflare.com/ajax/libs/jquery-throttle-debounce/1.1/jquery.ba-
throttle-debounce.min.js"></script>
<script src="js/modernizr.custom.lis.js"></script>
<script src="js/jquery.stickyheader.js"></script>
</section>
</body>
</html>

```

El resultado que se puede observar al ejecutar el script con cualquier navegador sería el siguiente:

**Edades de personas:**

Ingreso de edades:

Edades ingresadas:

- 45
- 28
- 32
- 37
- 21
- 26

**Edades de personas:**

Ingreso de edades:

Edades ingresadas:

- 28
- 32
- 37
- 21
- 26
- 46

Edades ingresadas
45
28
32
37
21
26
46
La edad promedio es: 33.57

**Resultado personal:**

Edades ingresadas
20
40
La edad promedio es: 30.00

---

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

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



**Ejercicio #4:** El siguiente ejemplo ilustra cómo manejar el envío de campos de formulario de tipo select y checkbox cuando en ambos casos se maneja el nombre del campo como una matriz dentro del script PHP.

Se crea una función en script del lado del servidor que procesará los datos seleccionados en ambos tipos de campos, teniendo en cuenta que en el caso de los datos seleccionados en el campo de formulario de tipo select se creará una lista desordenada (ol), mientras que en el caso de los checkbox que utilizan el mismo nombre en todos los campos creados con ciudades, se creará una lista ordenada (ul).

En la función se utilizan dos argumentos, el primero que debe ser la matriz con los datos de la lista y el segundo que indica el tipo de lista que se quiere crear donde los únicos valores posibles deberían ser: "ul", "ol" .

**Archivo 1: selectfields.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Estilo de checkboxes, radios y multi select</title>
  <link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Bitter" />
  <link rel="stylesheet" href="css/fonts.css" />
  <link rel="stylesheet"
href="http://ajax.googleapis.com/ajax/libs/jqueryui/1/themes/ui-lightness/jquery-ui.css" />
  <link rel="stylesheet" href="css/jquery.multiselect.css" />
  <link rel="stylesheet" href="css/fields.css" />
```

```

</head>
<body>
<header>
    <h1 class="inset">Ciudades y destinos de vacación</h1>
</header>
<section>
<article>
<div class="form-style">
<form name="frmdestinos" action="destinos.php" method="POST">
<h1>
    Selección de
    <span>países y ciudades</span>
</h1>
<select name="location[]" multiple="multiple" size="5" id="location">
    <option value="Guatemala">Guatemala</option>
    <option value="El Salvador">El Salvador</option>
    <option value="Costa Rica">Costa Rica</option>
    <option value="Honduras">Honduras</option>
    <option value="Panamá">Panamá</option>
    <option value="Nicaragua">Nicaragua</option>
    <option value="Belice">Belice</option>
</select><br />
<div id="places">
    <input type="checkbox" name="place[]" id="place1" value="Ciudad de Guatemala" />
    <label for="place1"><span></span>Ciudad de Guatemala</label><br />
    <input type="checkbox" name="place[]" id="place2" value="Antigua Guatemala" />
    <label for="place2"><span></span>Antigua Guatemala</label><br />
    <input type="checkbox" name="place[]" id="place3" value="Panajachel" />
    <label for="place3"><span></span>Panajachel</label><br />
    <input type="checkbox" name="place[]" id="place4" value="San Salvador" checked="checked"
/>
    <label for="place4"><span></span>San Salvador</label><br />
    <input type="checkbox" name="place[]" id="place5" value="Santa Ana" />
    <label for="place5"><span></span>Santa Ana</label><br />
    <input type="checkbox" name="place[]" id="place6" value="Santa Miguel" />
    <label for="place6"><span></span>Santa Miguel</label><br />
    <input type="checkbox" name="place[]" id="place7" value="San José" />
    <label for="place7"><span></span>San José</label><br />
    <input type="checkbox" name="place[]" id="place8" value="Puntarenas" />
    <label for="place8"><span></span>Puntarenas</label><br />
    <input type="checkbox" name="place[]" id="place9" value="San Pedro Sula" />
    <label for="place9"><span></span>San Pedro Sula</label><br />
    <input type="checkbox" name="place[]" id="place10" value="Tegucigalpa" />
    <label for="place10"><span></span>Tegucigalpa</label><br />
    <input type="checkbox" name="place[]" id="place11" value="Granada" />
    <label for="place11"><span></span>Granada</label><br />
    <input type="checkbox" name="place[]" id="place12" value="Ciudad de Panamá" />
    <label for="place12"><span></span>Ciudad de Panamá</label><br />
    <input type="checkbox" name="place[]" id="place13" value="Colón" />

```

```

        <label for="place13"><span></span>Colón</label><br />
        <input type="checkbox" name="place[]" id="place14" value="Colón" />
        <label for="place14"><span></span>Belmopán</label><br />
    </div>
    <input type="submit" name="submit" id="enviar" value="Enviar" class="button" />
    <input type="reset" name="cancel" value="Cancelar" class="button" />
</form>
</div>
</article>
</section>
</body>
<script src="js/checkFields.js"></script>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.js"></script>
<script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1/jquery-ui.min.js"></script>
<script src="js/jquery.multiselect.js"></script>
<script>
    $(function() {
        $("#location").multiselect();
    });
</script>
</html>

```

## Archivo 2: destinos.php

```

<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ciudades de destino</title>
    <link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Bitter" />
    <link rel="stylesheet" href="css/fields.css" />
</head>
<body>
<header>
    <h1 class="inset">Ciudades de destino</h1>
</header>
<section>
<div class="form-style">
<?php
    //La función espera una matriz con una lista ($lista)
    //ya sea de países o ciudades y un parámetro opcional
    //con el tipo de lista que tendrá un valor por defecto "ul"
    function createList($lista, $tipo="ul"){
        //Inicializando variables para ambos tipos de listas HTML
        $ullist = "";
        $ollist = "";
        switch($tipo):
            case "ul":

```

```

        $ullist .= "<article id=\"countries\">\n";
        $ullist .= "\t<h1>\n";
        $ullist .= "\t\tPaíses y ciudades\n";
        $ullist .= "\t\t<span>seleccionadas</span>\n";
        $ullist .= "\t</h1>\n";
        $ullist .= "\t<ul class=\"imglist\">\n";
        foreach($lista as $key => $value):
            $ullist .= "\t\t<li><a href=\"javascript:void(0)\">$key =>
$value</a></li>\n";
        endforeach;
        $ullist .= "\t</ul>\n";
        $ullist .= "</article>\n";
        print $ullist;
        break;
    case "ol":
        $ollist .= "<article id=\"cities\">\n";
        $ollist .= "\t<h1><span>Ciudades</span></h1>\n";
        $ollist .= "\t<div class=\"numberlist\">\n";
        $ollist .= "\t\t<ol>\n";
        foreach($lista as $key => $value):
            $ollist .= "\t\t\t<li><a href=\"javascript:void(0)\">$key =>
$value</a></li>\n";
        endforeach;
        $ollist .= "\t\t</ol>\n";
        $ollist .= "\t</div>\n";
        $ollist .= "</article>";
        print $ollist;
        break;
    default:
        print "<p>No está definido este tipo de lista</p>";
        break;
endswitch;
}

//Inicia el procesamiento del formulario
if(isset($_POST['submit'])):
    //Análisis de los elementos de campo select
    if(is_array($_POST['location'])):
        //Si se tiene una matriz invocar a la función
        //createList para crear la lista UL
        createList($_POST['location']);
    else:
        echo "Se esperaba una lista, no un valor escalar.";
        exit(0);
    endif;

    //Análisis de los elementos checkbox
    extract($_POST);
    if(is_array($place)):

```

```

        createList($place, "ol");
    endif;
else:
    print "<p>El procesamiento del formulario requiere que se haya presionado el botón
Enviar.</p>";
    print "<a href=\"selectfields.html\">Regresar</a>";
    exit(0);
endif;
?>
</div>
</section>
</body>
</html>

```

El resultado que podrá visualizar si ejecuta el script selectfields.html en el navegador de su preferencia, debería ser el siguiente:

The image displays two screenshots of a web application interface for selecting countries and cities. Both screenshots feature a blue header with the title "Selección de países y ciudades".

The top screenshot shows a list of countries with checkboxes. The "3 selected" dropdown at the top indicates that three items are selected. The selected items are Guatemala, Costa Rica, and Nicaragua. The interface includes "Check all" and "Uncheck all" buttons.

The bottom screenshot shows a list of cities with checkboxes. The "3 selected" dropdown at the top indicates that three items are selected. The selected items are Antigua Guatemala, Panajachel, San José, Puntarenas, and Granada. The interface includes "ENVIAR" and "CANCELAR" buttons at the bottom.

# Ciudades de destino

Países y ciudades  
seleccionadas

0 => Guatemala  
1 => Costa Rica  
2 => Nicaragua

Ciudades

1 0 => Antigua Guatemala  
2 1 => Panajachel  
3 2 => San José  
4 3 => Puntarenas  
5 4 => Granada

Resultado personal:

# Ciudades de destino

Países y ciudades  
seleccionadas

0 => Honduras  
1 => Nicaragua

Ciudades

1 0 => San Salvador  
2 1 => Tegucigalpa  
3 2 => Colón

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

**Ejercicio #5:** El siguiente ejemplo muestra cómo construir una aplicación modular creando un sitio web en partes, primero con un archivo denominado header.php, luego el footer.inc.php y el archivo header.inc.php. Ambos archivos son integrados en home.php para armar la página principal de un sitio web. Esta técnica de construcción es llamada modularidad, ya que se utilizan archivos .inc.php

**Archivo 1: sitio.php**

```
<?php
    require('header.inc.php');
?>
<body>
<div id="pagewrap">
<!-- #header y #nav -->
<?php
    require('nav.inc.php');
?>
<!-- /#header y #nav -->
<!-- #content -->
<?php
    require('content.php');
?>
<!-- /#content -->
<!-- #sidebar -->
<?php
    require('sidebar.php');
?>
<!-- /#sidebar -->
<!-- #footer -->
<?php
```

```

        require('footer.inc.php');
    ?>
<!-- /#footer -->
</div>
<!-- /#pagewrap -->
</body>
</html>

```

## Archivo 2: header.inc.php

```

<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width; initial-scale=1.0">
    <title>Web adaptable con Media Queries</title>
    <!--css -->
    <link rel="stylesheet" href="css/style.css" />
    <!-- <link href="media-queries.css" rel="stylesheet" type="text/css"> -->
    <!-- html5.js - IE 9 -->
    <!--[if lt IE 9]>
        <script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->
    <!-- css3-mediaqueries.js - IE 9 -->
    <!--[if lt IE 9]>
        <script src="http://css3-mediaqueries-js.googlecode.com/svn/trunk/css3-
mediaqueries.js"></script>
    <![endif]-->
</head>

```

## Archivo 3: nav.inc.php

```

<header id="header">
    <hgroup>
        
        <h1 id="site-description">Responsive Design con CSS3 Media Queries</h1>
    </hgroup>
    <nav>
        <ul id="main-nav" class="clearfix">
            <?php
                $menuoptions = array("Home", "Quiénes somos", "Servicios", "Portafolio",
"Blog", "Contacto");
                $listitems = "";
                foreach($menuoptions as $item):
                    $listitems .= "<t<li>\n\n<a
href=\"javascript:void(0);\">$item</a>\n</li>\n";
                endforeach;
                echo $listitems;
            ?>
        </ul>
    </nav>

```



```

</nav>
<!-- nav -->
<form id="searchform" action="javascript:void(0);">
    <input id="s" placeholder="Search" type="search">
</form>
</header>

```

#### Archivo 4: content.php

```

<div id="content">
<article class="post clearfix">
    <h2 class="post-title"><a href="#">HTML5</a></h2>
    <figure class="post-imagen">
        
    </figure>
    <p>
        <strong>HTML5</strong>
        (<em><strong>H</strong>yper<strong>T</strong>ext <strong>M</strong>arkup
        <strong>L</strong>anguage</em>, versión 5) es la quinta revisión importante
        del lenguaje básico de la
        <a href="http://es.wikipedia.org/wiki/World_Wide_Web" title="World Wide Web">World
Wide Web</a>,
        <a href="http://es.wikipedia.org/wiki/HTML" title="HTML">HTML</a>.
        HTML5 especifica dos variantes de sintaxis para HTML:
        un «clásico» HTML (text/html), la variante conocida
        como <em>HTML5</em> y una variante
        <a href="http://es.wikipedia.org/wiki/XHTML" title="XHTML">XHTML</a>
        conocida como sintaxis <em>XHTML5</em> que deberá
        ser servida como XML (XHTML) (application/xhtml+xml).
        <a href="http://es.wikipedia.org/wiki/HTML5#cite_note-w3cSpec-0">1</a>
        <a href="http://es.wikipedia.org/wiki/HTML5#cite_note-franganilloHTML5-1">2</a>
        Esta es la primera vez que HTML y XHTML se han desarrollado
        en paralelo.
    </p>
    <p>
        Todavía se encuentra en modo experimental, lo cual indica la misma W3C; aunque ya
es usado
        por múltiples desarrolladores web por sus avances, mejoras y ventajas.
    </p>
    <p>
        Al no ser reconocido en viejas versiones de navegadores por sus nuevas etiquetas,
se
        le recomienda al usuario común actualizar a la versión más nueva, para poder
disfrutar
        de todo el potencial que provee HTML5.
    </p>
</article>
<article class="post clearfix">
    <h2 class="post-title"><a href="#">CSS3</a></h2>

```

```

<figure class="post-imagen">
    
</figure>
<p>
    El nombre <strong>hojas de estilo en cascada</strong> viene del
    <a href="http://es.wikipedia.org/wiki/Idioma_ingl%C3%A9s" title="Idioma
inglés">inglés</a>
    <em>Cascading Style Sheets</em>, del que toma sus siglas.
    CSS es un lenguaje usado para definir la presentación de
    un documento estructurado escrito en
    <a href="http://es.wikipedia.org/wiki/HTML" title="HTML">HTML</a>
    o <a href="http://es.wikipedia.org/wiki/XML" title="XML">XML</a>
    <a href="http://es.wikipedia.org/wiki/CSS3#cite_note-1">2</a>
    (y por extensión en <a href="http://es.wikipedia.org/wiki/XHTML"
title="XHTML">XHTML</a>).
</p>
<p>
    El <a href="http://es.wikipedia.org/wiki/W3C" title="W3C">W3C</a>
    (World Wide Web Consortium) es el encargado de formular la especificación de las
    <a href="http://es.wikipedia.org/wiki/Hojas_de_estilo" title="Hojas de
estilo">hojas de estilo</a> que servirán de estándar para los
    <a href="http://es.wikipedia.org/wiki/Agentes_de_usuario" title="Agentes de
usuario">agentes de usuario</a> o
    <a href="http://es.wikipedia.org/wiki/Navegadores"
title="Navegadores">navegadores</a>.<br>
    La idea que se encuentra detrás del desarrollo de CSS es separar la
    <em>estructura</em> de un documento de su <em>presentación</em>.
</p>
</article>
<!-- /.post -->
</div>

```

### Archivo 5: sidebar.php

```

<aside id="sidebar">
<section class="widget">
    <h4 class="widgettitulo">Noticias</h4>
    <ul>
    <?php
        $submenuoptions = array(
            "Diseño Web",
            "Diseño Gráfico",
            "WordPress",
            "Joomla",
            "Drupal",
            "Redes Sociales",
            "SEO"
        );
        $listsubmenu = "";

```

```

        foreach($submenuoptions as $itemsub):
            $listsubmenu .= "\n<li>\n\n<a
href=\"javascript:void(0);\">$itemsub</a>\n</li>\n";
        endforeach;
        echo $listsubmenu;
    ?>
</ul>
</section>
<section class="widget">
    <h4 class="widgettitulo">Sitos Web</h4>
    <ul>
    <?php
        $externallinks = array(
            "http://www.tutosytips.com/"      => "Tutosytips.com",
            "http://www.wordpress.org.com/"  => "Wordpress",
            "http://www.Joomla.org/"         => "Joomla",
            "http://www.twitter.com/"        => "Twetter",
            "http://www.facebook.com/"       => "Facebook",
            "http://www.youtube.com/"        => "YouTube",
            "javascript:void(0);"             => "SEO"
        );
        $listext = "";
        foreach($externallinks as $extlink => $extitem):
            $listext .= "\n<li>\n\n<a href=\"$extlink\">$extitem</a>\n</li>\n";
        endforeach;
        echo $listext;
    ?>
    </ul>
</section>
</aside>

```

#### Archivo 6: footer.inc.php

```

<footer id="footer">
    <p>Tutorial por <a href="http://tutosytips.com/">tutosytips</a></p>
</footer>
<!-- footer -->

```

El resultado:



## Resultado personal:



## V. DISCUSION DE RESULTADOS

1. Realice un script PHP que utilice una función para que dada una lista de números enteros ingresada por el usuario (como mínimo 2 números) permita encontrar el número mayor y menor de dicha lista utilizando dos funciones con lista variable de argumentos, una para encontrar el número mayor y otra para encontrar el número menor. Los números deben ser ingresados mediante un campo input de tipo número (type="number") que mediante un botón agregar se añadirán en un campo select de múltiple selección, del mismo modo que el ejemplo de las edades. Cuando se hayan ingresado varios número, el usuario debe seleccionar todos o una cantidad de valores ingresados que no debe ser menor que dos. Si se intenta enviar uno solo o ninguno de los números debe lanzarse un mensaje de alerta con JavaScript. La solución debe permitir visualizar la lista de números ingresados e indicar cuáles han sido el número mayor y el número menor encontrados por sus funciones.

### Resultado personal:

#### Encuentra el número mayor y menor

Ingrese un número:

Agregar

Lista de números:

Calcular

Números seleccionados: 10, 30, 20

Mayor: 30

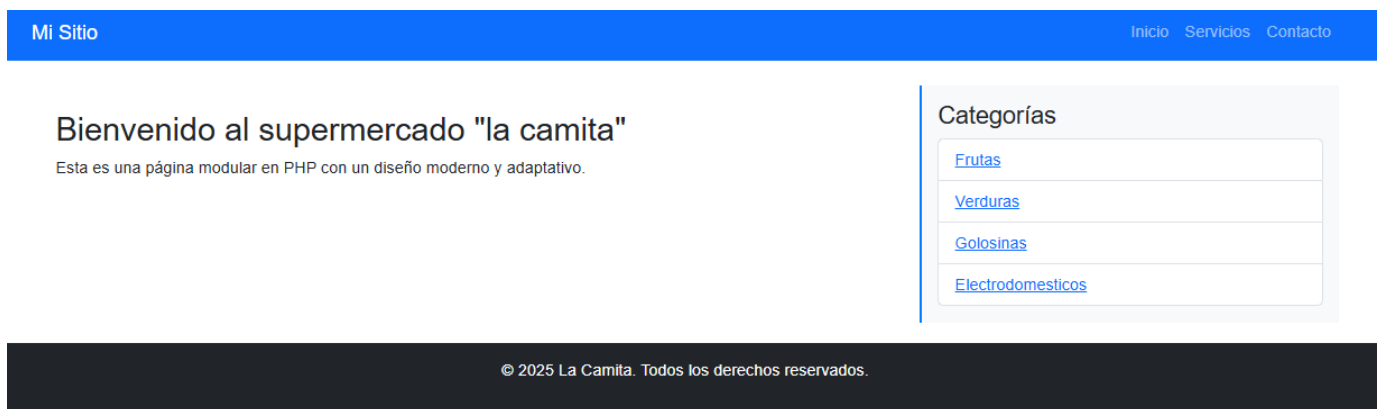
Menor: 10

---

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

- Haciendo uso de la modularidad descrita en el ejemplo 7 del procedimiento, cree una página web que incluya un archivo PHP para la sección head de la página que incluya la Definición de Tipo de Documento, otro para la sección footer, el menú de navegación principal, la sección content y la sección sidebar. Tiene que tomar un diseño distinto del que se mostró en el ejemplo. Puede ser la misma distribución en las mismas posiciones, pero tiene que utilizar una hoja de estilo completamente diferente y con manejo de adaptabilidad para distintos tipos de dispositivo y resoluciones de pantalla.

## Resultado :



## VI. INVESTIGACIÓN 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()`.

- **`include('archivo.php');`** : Incluye y ejecuta el archivo especificado. Si el archivo no se encuentra, PHP lanza una advertencia (warning) pero el script sigue ejecutándose.
- **`require('archivo.php');`** : También incluye y ejecuta el archivo, pero si no se encuentra, genera un error fatal y detiene la ejecución del script.
- **`include_once('archivo.php');`** : Funciona como `include()`, pero **solo incluye el archivo una vez**. Si ya ha sido incluido, PHP lo ignora.
- **`require_once('archivo.php');`** : Funciona como `require()`, pero evita incluir el archivo más de una vez.

Diferencias claves:

- Evitan incluir el mismo archivo varias veces, lo que es útil para evitar errores de redeclaración de funciones, clases o variables.
- `require_once()` es más estricto, si el archivo no se encuentra, el script se detiene.

2. Investigue sobre el uso de funciones anónimas en PHP, indicando en qué casos pueden ser utilizadas e implemente un ejemplo simple de cada caso. Que no sean los mismos que ya salen en los sitios web que consulte. Modifique para mostrar esa utilidad con sus propios ejemplos.

Las funciones anónimas en PHP (también llamadas closures o funciones lambda) son funciones que no tienen un nombre y se pueden asignar a variables, pasar como argumentos o usar en funciones de orden superior.

Dichas funciones tienen diferentes usos, entre ellos:

- ❖ Callback en `array_map()`: Aplicamos una función anónima para elevar al cuadrado cada número de un array.

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

$cuadrados = array_map(function($num) {
    return $num * $num;
}, $numeros);

print_r($cuadrados); // Salida: [1, 4, 9, 16, 25]
```

- ❖ Clausuras con use: Las funciones anónimas pueden capturar variables del ámbito externo usando use.

```
$factor = 3;
$triple = function($num) use ($factor) {
    return $num * $factor;
};

echo $triple(5); // Salida: 15
```

- ❖ Asignar Función a Variable: Se usa una función anónima como una variable para reutilizarla.

```
$saludar = function($nombre) {
    return "¡Hola, $nombre!";
};

echo $saludar("Carlos"); // Salida:
¡Hola, Carlos!
```

- ❖ Función anónima en array\_filter(): Filtramos números pares usando una función anónima.

```
$numeros = [10, 15, 20, 25, 30];

$pares = array_filter($numeros,
function($num) {
    return $num % 2 === 0;
});

print_r($pares); // Salida: [10, 20, 30]
```



- ❖ Uso en Rutas Simples: Se usa una función anónima para manejar rutas en una app simple.

```
$rutas = [
    "/inicio" => function() { echo "Bienvenido a
la página de inicio"; },
    "/contacto" => function() { echo "Esta es la
página de contacto"; }
];

$rutaActual = "/inicio";
if (isset($rutas[$rutaActual])) {
    $rutas[$rutaActual](); // Llama a la función
    anónima correspondiente
}
```

3. Investigue el uso de las funciones de filtros de saneamiento y validación, concretamente filter\_input() y filter\_var, explique las diferencias, muestre su sintaxis, indique los distintos tipos de filtro que se pueden utilizar en uno de sus argumentos y muestre ejemplos concretos de su utilización, al menos dos ejemplos con filter\_input() y dos con filter\_var().

filter\_input() obtiene y filtra datos directamente desde \$\_GET, \$\_POST, etc, en cambio filter\_var() filtra una variable ya almacenada en PHP.

**Sintaxis de filter\_input() y filter\_var():**

```
// filter_input(tipo de entrada, nombre de la
variable, tipo de filtro, opciones)
filter_input(INPUT_GET, 'nombre',
FILTER_SANITIZE_STRING);

// filter_var(variable, tipo de filtro, opciones)
filter_var($email, FILTER_VALIDATE_EMAIL);
```

## Tipos de filtros en PHP

### *Filtros de validacion*

- ❖ **FILTER\_VALIDATE\_INT**: Verifica si el valor es un número entero.
- ❖ **FILTER\_VALIDATE\_FLOAT**: Verifica si el valor es un número decimal.
- ❖ **FILTER\_VALIDATE\_EMAIL**: Verifica si el valor es un correo válido.
- ❖ **FILTER\_VALIDATE\_URL**: Verifica si el valor es una URL válida.
- ❖ **FILTER\_VALIDATE\_IP**: Verifica si el valor es una IP válida.

### *Filtros de saneamiento*

- ❖ **FILTER\_SANITIZE\_STRING**: Elimina caracteres peligrosos.
- ❖ **FILTER\_SANITIZE\_EMAIL**: Elimina caracteres inválidos en emails.
- ❖ **FILTER\_SANITIZE\_URL**: Elimina caracteres inválidos en URLs.
- ❖ **FILTER\_SANITIZE\_NUMBER\_INT**: Elimina caracteres que no sean números.
- ❖ **FILTER\_SANITIZE\_SPECIAL\_CHARS**: Escapa caracteres especiales en HTML.

## Ejemplos de uso de dichos filtros en PHP:

```
# | Validar y Sanear un Número con filter_input()
// Suponiendo que el usuario envía ?edad=23 en la URL
$edad = filter_input(INPUT_GET, 'edad',
    FILTER_VALIDATE_INT);

if ($edad === false) {
    echo "Edad no válida";
} else {
    echo "Edad válida: $edad";
}
```

```

# Sanear y Validar un Email con filter_input()

// Suponiendo que el usuario envía un email por
formulario POST
$email = filter_input(INPUT_POST, 'correo',
FILTER_SANITIZE_EMAIL);

if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
    echo "Correo válido: $email";
} else {
    echo "Correo no válido";
}

```

```

//Sanear una Cadena con filter_var()
$texto = "<script>alert('Hackeado');</script> Hola
Mundo!";
$seguro = filter_var($texto,
FILTER_SANITIZE_STRING);

echo $seguro; // Salida: " Hola Mundo!"

```

## 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>