



Análisis y desarrollo de software.

ID: 2556456



www.sena.edu.co

@SENAComunica

Instructor

Diego Fernando Calderón Silva
Correo: dfcalderon@sena.edu.co

Introducción a PHP

¿Qué es PHP?

PHP, es un lenguaje de programación de **código abierto** que permite el desarrollo de aplicaciones WEB o aplicaciones WEB dinámicas, el cual es apto para incrustar en lenguaje HTML

Tomado de:
<https://www.php.net/manual/es/index.php>



Contenido

1. Conceptos básicos.
2. Estructuras condicionales.
3. Ciclos y bucles.
4. Envío de datos mediante formularios.
5. Conexión a base de datos.
6. CRUD.

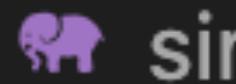
1. Conceptos básicos

- **Requisitos:**
 - Editor de texto.
 - Servidor Local
 - Navegador WEB



1. Conceptos básicos

- **Sintaxis:**



```
sintaxis.php
1 <?php
2
3 echo "Hola mundo!!!";
4
5 ?>
```

1. Conceptos básicos

- **Como hacer comentarios en PHP:**

Como su nombre lo indica, "comentarios" son aquellas porciones de código que hacen referencia a lo que se esta haciendo en la siguiente línea.

Usarlos es una buena practica.

1. Conceptos básicos

- Como hacer comentarios en PHP:

```
❶ syntaxis.php
❷ <?php
❸ //Mi primer hola mundo en PHP
❹ /*
❺   voy
❻   a
❼   comentar
❼   varias
❼   lineas
❼ */
❽ echo "Hola mundo ! ! !";
❾
❿ ?>
```

1. Conceptos básicos

- **Tipos de datos:**

- booleanos (Boolean)
- Enteros (Integer)
- Flotante (Double)
- Cadena de caracteres (String)
- Array
- Iterable
- Objetos
- Recursos
- NULO



1. Conceptos básicos

- **Tipos de datos (var_dump):**

A large magnifying glass is positioned over the list of data types, focusing on the text inside the lens.

- booleanos (Boolean)
- Enteros (Integer)
- Flotante (Double)
- Cadena de caracteres (String)

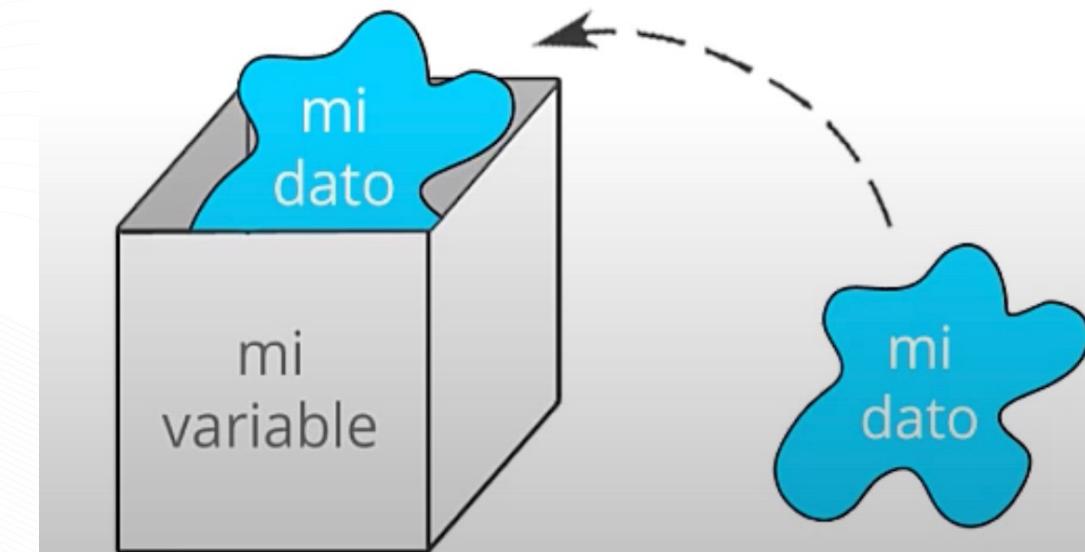
1. Conceptos básicos

- ¿Qué es una variable?

Una variable es un espacio en memoria con un nombre reservado para almacenar un valor correspondiente a un tipo de dato

El valor de una variable, puede cambiar durante la ejecución del código.

Tomado de:
<https://www.php.net/manual/es/language.variables.basics.php>



1. Conceptos básicos

- **Reglas para definir una variable.**
 1. En PHP las variables se presentan con un signo de dólar seguido por el nombre de la variable
 2. El nombre de una variable debe empezar con una letra un un guion bajo(_), seguido de cualquier número, letra. **NO PUEDE EMPEZAR CON NUMEROS.**
 3. El nombre de las variables es sensible a mayúsculas y minúsculas

1. Conceptos básicos

- ¿Qué es una constante?

En programación una constante es un valor que no puede ser alterado o modificado durante la ejecución de un programa.

Una constante corresponde a una longitud fija de un área reservada en memoria.

Ej: PI, DNI

1. Conceptos básicos

• Reglas para definir una constante.

1. El nombre de una constante sigue las mismas reglas de una variable de PHP, es decir que un nombre valido de una constante empieza por una letra o un guion bajo, seguido por la asignación de nombre que se le quiera dar.
2. Por defecto una constante distingue mayúsculas y minúsculas.
3. Por convención, los identificadores de constantes siempre se declaran en **MAYUSCULA**

 sintaxis.php X

clas >  sintaxis.php

```
1  <?php  
2  
3  define("CONSTANTE1", "Hola Mundo");  
4  
5  echo CONSTANTE1;  
6  
7  
8  ?>
```

 localhost/clas/sintaxis.php

Hola mundo



localhost/clas/sintaxis.php



Hola mundo

 sintaxis.php X

clas >  sintaxis.php

```
1  <?php  
2  
3  $const = 'Hola mundo';  
4  
5  echo $const;  
6  
7  ?>
```

1. Conceptos básicos

- ¿Qué es un Array?

Un array en PHP es un tipo de dato especial que representa los conocidos mapas de datos ordenados, ya que asocia valores con claves, también llamados “Array asociativo”.

Los valores de un array no tienen que ser del mismo tipo.

1. Conceptos básicos

- ¿Qué es un Array?

PHP soporta arrays:

1. Escalares (numérico)
2. Asociativos (índice por clave)
3. Multidimensionales

Y la forma de acceder a ellos es mediante corchetes [] y a diferencia de otros lenguajes no es necesario declararlo antes de usarlo.

Tomado de:

<https://www.php.net/manual/es/intro.array.php>

1. Conceptos básicos

- ¿Cómo definir un Array?

En php podemos usar el constructor `array()` o la notación de corchetes `[]`

Si a la hora de definir un array no se le asigna un valor será un array vacío.

Tomado de:

<https://www.php.net/manual/es/array.sorting.php>

1. Conceptos básicos

- ¿Qué es concatenación?

Unión de variables mediante el comando punto (.)

Eje: \$concat = \$minombre.\$miapellido;

1. Conceptos básicos

- ¿Qué es interpolación?

Es cuando se muestra el valor de una variable dentro de un string, pero solo se puede hacer en textos con comillas dobles “”

Eje:

1. echo “Mi nombre es: \$nombre y tengo \$edad años.”;
2. echo ‘Mi nombre es: \$nombre y tengo \$edad años.’;

1. Conceptos básicos

- Operadores aritméticos.

Nombre	Símbolo en PHP
Suma	+
Resta	-
División	/
Multiplicación	*
Modulo	%
Exponenciación	**

1. Conceptos básicos

- Jerarquía de los operadores aritméticos.

Jerarquía	Operador
1	()
2	**
3	*, /
4	+, -
5	%

1. Conceptos básicos

- **Asignación por referencia.**

Ambas variables terminan apuntando a los mismos datos y nada es copiado en ninguna parte. Lo que traduce que al usar el símbolo “&” antes del nombramiento de la variable, esta guardara el espacio en memoria ocupado, por lo que no guardara su valor actual si no cualquier valor que se ocupe en ese espacio de memoria a lo largo del tiempo.



1. Conceptos básicos

- Operadores de comparación.

Nombre	Símbolo en PHP	Respuesta
Igual	<code>==</code>	Booleano
Idéntico	<code>===</code>	Booleano
Diferente	<code>!=</code>	Booleano
Menor que	<code><</code>	Booleano
Mayor que	<code>></code>	Booleano
Menor o igual que	<code><=</code>	Booleano
Mayor o igual que	<code>>=</code>	Booleano

1. Conceptos básicos

- Operadores lógicos.

Símbolo	Nombre
AND, and	And (y)
OR, or	Or (ó)
!	Not (!)
&&	And (y)
	Or (o)

1. Conceptos básicos

- Operadores lógicos (AND).
- True = 1; False = 0

V1	V2	Resultado
0	0	0
0	1	0
1	0	0
1	1	1

1. Conceptos básicos

- Operadores lógicos (OR).
- True = 1; False = 0

V1	V2	Resultado
0	0	0
0	1	1
1	0	1
1	1	1

1. Conceptos básicos

- Operadores lógicos (NOT).
- True = 1; False = 0

V1	Resultado
!0	1
!1	0

1. Conceptos básicos

- Operadores de incremento/decremento

Símbolo	Nombre
<code>++</code>	Incremento
<code>--</code>	Decremento

EJEMPLOS

<code>++\$variable</code>	Pre-Incremento
<code>\$variable++</code>	Post-Incremento
<code>--\$variable</code>	Pre-Decremento
<code>\$variable--</code>	Post-Decremento

2. Estructuras condicionales

2. Estructuras condicionales



- Las estructuras condicionales son utilizadas para tomar decisiones en función de que se cumpla o no una determinada condición.

2. Estructuras condicionales



• Estructura condicional simple (if)

2. Estructuras condicionales



- Estructura condicional if

Ejercicio:

En una fabrica de computadores se plantea ofrecer a los clientes un descuento que dependerá del numero de computadores que compre. Si los computadores son menos de 5, se le dará un descuento del 10%, si el numero de computadores comprados es mayor o igual a 5 pero menos de 10 se le otorga un 20% de descuento; y si son mas de 10 se les da un 40% de descuento. El precio de cada computador es de \$700 USD.

2. Estructuras condicionales



- Estructura condicional multiple if-else-elseif

Método 1	Método 2
<pre>If(\$a>\$b){ echo "\$a es mayor que \$b"; }elseif (\$a == \$b) { echo "\$a es igual que \$b"; } Else { Echo "Ninguna es correcta"; }</pre>	<pre>If(\$a>\$b): echo "\$a es mayor que \$b"; elseif (\$a == \$b): echo "\$a es igual que \$b"; Else : Echo "Ninguna es correcta"; endif;</pre>

Ejercicios

1- Ejercicio de comparación numérica:
Escribe un programa que solicite al usuario que ingrese un número. Luego, el programa debe imprimir si el número ingresado es mayor, menor o igual a 10 utilizando la sentencia if.

2-Ejercicio de comparación de cadenas:
Escribe un programa que solicite al usuario que ingrese su nombre. Si el nombre ingresado es "Juan", el programa debe imprimir "¡Hola Juan!" en la pantalla. Si el nombre es diferente, el programa debe imprimir "Lo siento, no te conozco".

Ejercicios

3- Ejercicio de comparación múltiple:

Escribe un programa que solicite al usuario que ingrese un número del 1 al 7. Luego, el programa debe imprimir el día de la semana correspondiente al número ingresado. Por ejemplo, si el usuario ingresa "1", el programa debe imprimir "Lunes". Si el número ingresado no está en el rango válido, el programa debe imprimir "Número inválido".

4- Ejercicio de anidación de sentencias if:

Escribe un programa que solicite al usuario que ingrese su edad. Si la edad es mayor o igual a 18 años, el programa debe preguntar si tiene licencia de conducir. Si la respuesta es "sí", el programa debe imprimir "Puedes conducir". Si la respuesta es "no", el programa debe imprimir "Debes obtener una licencia de conducir primero". Si la edad es menor de 18 años, el programa debe imprimir "No puedes conducir".

5- Ejercicio de uso de operadores lógicos:

Escribe un programa que solicite al usuario que ingrese dos números. Luego, el programa debe imprimir si ambos números son mayores que 10 utilizando la sentencia if y los operadores lógicos "&&" y ">". Si ambos números son mayores que 10, el programa debe imprimir "Ambos números son mayores que 10". Si solo uno de los números es mayor que 10, el programa debe imprimir "Solo uno de los números es mayor que 10". Si ninguno de los números es mayor que 10, el programa debe imprimir "Ninguno de los números es mayor que 10".

2. Estructuras condicionales



- Operador ternario

El operador ternario puede considerarse como una estructura if de una sola línea.
Este está compuesto por tres partes:

```
<OPERADOR> ? <TRUE VALUE> : <FALSE VALUE>;
```

Se usa para operaciones lógicas por lo que el “echo” está rotundamente prohibido.

Ej:

```
(9>7) ? $total=10*7 : $total= 10*5;  
echo $total;
```

2. Estructuras condicionales



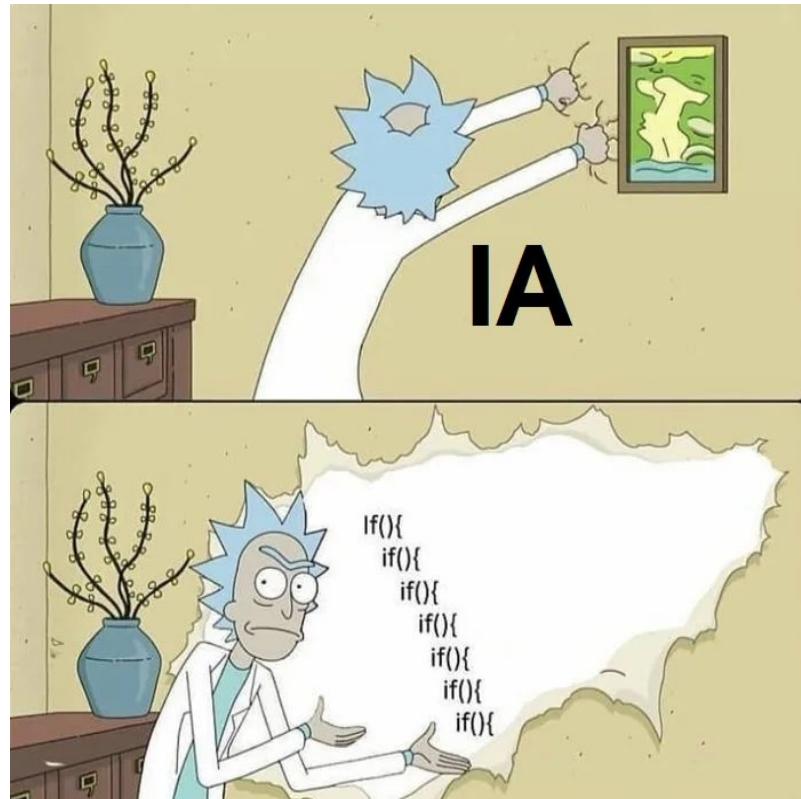
- Operador ternario

Ejercicio:

Hacer un programa que calcule el total a pagar por la compra de camisas. Si se compran 3 camisas o mas se debe aplicar un descuento del 20% sobre el total de la compra y si son menos de 3 camisas un descuento del 10%

2. Estructuras condicionales

- # • Estructura condicional anidada



2. Estructuras condicionales



- Estructura condicional anidada

```
if (condicion) {  
    if(condicion){  
        Bloque de código a ejecutar si la condición es TRUE  
    }else{  
        Bloque de código a ejecutar si la condición es FALSE  
    }  
} elseif (condicion) {  
    Bloque de código a ejecutar si la condición es TRUE  
} else {  
    Bloque de código a ejecutar si la condición es FALSE  
}
```

2. Estructuras condicionales



- Estructura condicional anidada
- Ejercicio:

Pida a un usuario la edad y genero, para que el sistema le indique si ya esta en la edad para pensionarse. Tenga en cuenta que un Hombre se puede pensionar a los 60años o mas, mientras que las mujeres lo pueden hacer después de los 54 años.

2. Estructuras condicionales



- ### Estructura condicional SWITCH

- La sentencia switch en PHP es una estructura de control que permite ejecutar diferentes bloques de código dependiendo del valor de una expresión o variable. La sintaxis de la sentencia switch consiste en una expresión que se evalúa y se compara con diferentes casos utilizando la palabra clave “**case**”, seguida de un bloque de código que se ejecutará si el valor de la expresión coincide con el caso. Si ninguno de los casos coincide con el valor de la expresión, se puede incluir un bloque de código “**default**” que se ejecutará en su lugar. La sentencia switch es útil cuando se tienen múltiples opciones de ejecución basadas en el valor de una variable, lo que simplifica el código en lugar de tener una serie de sentencias if encadenadas.

2. Estructuras condicionales



• Estructura condicional SWITCH

```
<?php  
$opcion = 2;  
  
switch ($opcion) {  
    case 1:  
        echo "Has seleccionado la opción 1";  
        break;  
    case 2:  
        echo "Has seleccionado la opción 2";  
        break;  
    case 3:  
        echo "Has seleccionado la opción 3";  
        break;  
    default:  
        echo "Opción inválida";  
        break;  
}  
?>
```

2. Estructuras condicionales



- Estructura condicional SWITCH
- Ejercicios:

Diseñe un software que escriba los nombre del día de la semana en función de una variable llamada “día”.

Los días de la semana son 7, por lo que, el rango de los valores de “dia” debe ser de 1 a 7 y dado el caso que “dia” tome un valor fuera de rango se deberá producir un mensaje de alerta!!!

2. Estructuras condicionales



- Estructura de selección múltiple
MATCH

- Esta estructura se introdujo en PHP 8, es similar a la estructura SWITCH pero acá las operaciones tienen en cuenta el tipo de dato a iterar.
- Permite doble comparación y distingue entre tipos de datos
- Permite hacer condicionales.

2. Estructuras condicionales



- Estructura de selección múltiple
MATCH

```
$opcion = 2;

switch ($opcion) {
    case 1:
        echo "Seleccionaste la opción 1";
        break;
    case 2:
        echo "Seleccionaste la opción 2";
        break;
    case 3:
        echo "Seleccionaste la opción 3";
        break;
    default:
        echo "Opción inválida";
        break;
}
```

```
$color = 'rojo';

$action = match ($color) {
    'rojo' => 'Detenerse',
    'amarillo' => 'Precaución',
    'verde' => 'Avanzar',
    default => 'Color desconocido'
};

echo $action;
```

2. Estructuras condicionales



- Estructura de selección múltiple
MATCH

```
<?php

$a="7";

$x=10;
$y=9;
$z=7;

$resultado = match($a){
    $x => "Valor igual a X",
    $y => "Valor igual a Y",
    $z => "Valor igual a Z",
    default => "No coincide con ninguna variable"
};

echo $resultado;
```

2. Estructuras condicionales



- Estructura de selección múltiple
MATCH

```
<?php
$edad = 18;

$resultado = match(true){
    $edad >= 60 => "Eres de la tercer edad",
    $edad >= 30 => "Eres adulto",
    $edad >= 10 => "Eres un niño"
};

echo $resultado;
```

2. Estructuras condicionales



• Desarrolle

- 1. Escribe un programa que tome un número entero como entrada y muestre un mensaje diferente dependiendo de si el número es 1, 2, o 3 utilizando el ciclo `match`.
- 2. Crea un programa que reciba el nombre de un color como entrada y muestre un mensaje diferente según el color utilizando el ciclo `match`. Por ejemplo, si el color es "rojo", mostrar "El color es cálido".

2. Estructuras condicionales



• Desarrolle

- 3. Diseña un programa que tome una letra como entrada y muestre un mensaje según la categoría a la que pertenezca la letra utilizando el ciclo `match`. Por ejemplo, si la letra es una vocal, mostrar "Es una vocal".
- 4. Desarrolla un programa que tome una fecha como entrada en formato "día-mes-año" y muestre un mensaje diferente según el mes utilizando el ciclo `match`.
- 5. Escribe un programa que tome un número del 1 al 7 como entrada y muestre el nombre del día de la semana correspondiente utilizando el ciclo `match`.

2. Estructuras condicionales



• Desarrolle

- 6. Crea un programa que reciba un número del 1 al 12 como entrada y muestre el nombre del mes correspondiente utilizando el ciclo `match`.
- 7. Diseña un programa que tome una nota numérica del 0 al 100 como entrada y muestre un mensaje diferente según el rango de la nota utilizando el ciclo `match`. Por ejemplo, si la nota está entre 90 y 100, mostrar "Excelente".
- 8. Desarrolla un programa que tome una cadena de texto como entrada y muestre un mensaje según la longitud de la cadena utilizando el ciclo `match`. Por ejemplo, si la longitud es mayor a 10 caracteres, mostrar "La cadena es larga".

2. Estructuras condicionales



- Desarrolle

- 9. Escribe un programa que tome una hora en formato "HH:MM" como entrada y muestre un mensaje diferente según la franja horaria utilizando el ciclo 'match'.
- 10. Crea un programa que reciba una opción del 1 al 5 como entrada y realice una acción diferente según la opción utilizando el ciclo 'match'.

3. Ciclos y bucles

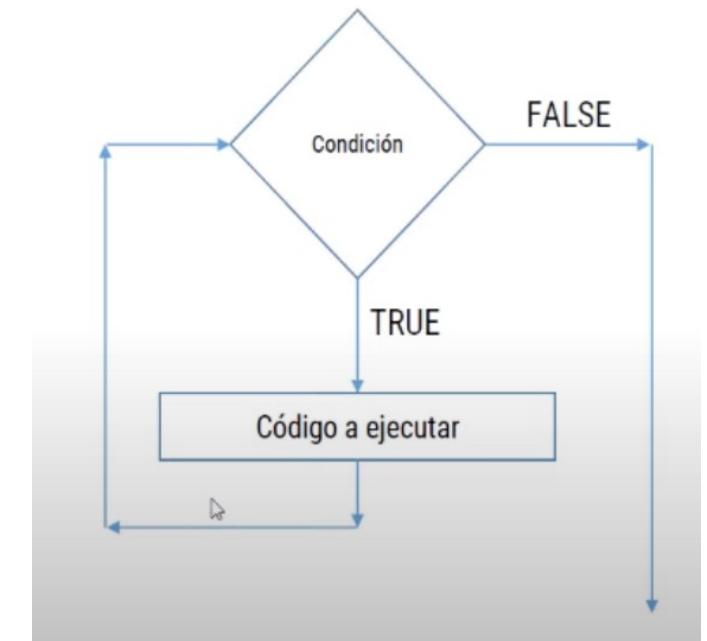
- Estructura repetitivas

- **WHILE**

While (mientras), ejecuta una porción de código designado mientras la condición que se le indica en su constructor sea **VERDADERA**. Una vez se sale del ciclo, es porque la condición deja de cumplirse.

```
While(Condicion){  
    Código a ejecutar;  
}
```

```
$num = 1;  
  
while ($num <= 10) {  
    echo $num . "<br>";  
    $num++;  
}
```



3. Ciclos y bucles



- Estructura repetitivas

- **WHILE**

Ejercicio:

1. Diseñe un código que imprima los números de -10 a 20 usando incrementos y decrementos.
2. Diseñe un código que imprima la tabla de multiplicar de un numero entre el 1 y 12.
3. Diseñe un código que muestre los números pares entre 0 y 100.

3. Ciclos y bucles



- Estructura repetitivas

- **DO - WHILE**

Do - while (hacer mientras mientras), es similar a la estructura while, excepto que la expresión verdadera es verificada al final de cada iteración en lugar de al principio.

```
do {  
    Código a ejecutar;  
}while(Condicion)
```

```
$num = 1;  
  
do {  
    echo $num . "<br>";  
    $num++;  
} while ($num <= 10);
```

3. Ciclos y bucles



- Estructura repetitivas

- **DO-WHILE**

Ejercicio:

1. Diseñe un código que imprima los números de -10 a 20 usando incrementos y decrementos.
2. Diseñe un código que imprima la tabla de multiplicar de un numero entre el 1 y 12.
3. Diseñe un código que muestre los números impares entre 0 y 100.

3. Ciclos y bucles



- Estructura repetitivas

- **FOR(para)**

La estructura repetitiva “for” se utiliza cuando se tienen definidas la cantidad de iteraciones a realizar

```
for($var; Condicion; (incre | decre) ){
```

Código a ejecutar;

```
}
```

3. Ciclos y bucles



- Estructura repetitivas
- **FOR**
Ejercicios
 - 1. Diseñe un código que muestre los numero del -10 al 10.
 - 2. Escriba un código que muestre la suma de los primeros 10 números enteros.
 - 3. Diseñe un código que genere un ciclo for anidado para imprimir las tablas de multiplicar del 1 al 10.

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

3. Ciclos y bucles

- Estructura repetitivas

- **FOR**

```
echo "<table>";

for ($i = 1; $i <= 10; $i++) {
    echo "<tr>";

    for ($j = 1; $j <= 10; $j++) {
        $mult = $i * $j;
        echo "<td>" . $mult . "</td>";
    }

    echo "</tr>";
}

echo "</table>";
```

3. Ciclos y bucles

- Estructura repetitivas

- **FOREACH**

Este ciclo proporciona un modo bastante sencillo de iterar sobre arrays. Se debe tener en cuenta que **foreach solo funciona sobre arrays** y de intentar usarlo sobre otro tipo de variable emitirá error. Para esta estructura existen dos sintaxis.

```
foreach($array as $valor){  
    $valor tendrá en cada iteración  
    un valor del array  
}
```

```
foreach($array as $clave => $valor){  
    $clave tendrá en cada iteración  
    una clave del array  
  
    $valor tendrá en cada iteración  
    un valor del array  
}
```

3. Ciclos y bucles

- Estructura repetitivas

- **FOREACH**

```
$array = array('uno', 'dos', 'tres');

foreach ($array as $valor) {
    echo $valor;
}
```

```
<?php

$frutas = ["Mora", "Naranja", "Fresa", "Maracuya", "Mandarina"];

$computadores = [
    "marca" => "Acer",
    "Modelo" => "a2338",
    "Color" => "Rojo",
    "Ram" => "256GB"
];

foreach($computadores as $clave => $com){
    echo "clave=".$clave." valor=".$com."<br>";
}
```

3. Ciclos y bucles



- ¿Cómo salir o detener un ciclo?



4. Envío de datos mediante formularios

4. Funciones

- **Funciones:** Es un conjunto de instrucciones a la que podemos recurrir siempre que lo deseemos, Estas pueden recibir parámetros externos y realizar cualquier tipo de tarea.
- **Declaración:** Para que sea un nombre valido debe comenzar con una letra o guion bajo seguido de cualquier letra pero antes debe existir la palabra reservada “function”.

```
function nombre_de_la_funcion(tipo_de_dato $parametros){  
    Código de la función;  
}
```

4. Funciones



Ejemplo de función.

- Crea una función llamada "calcularDescuento" que reciba dos parámetros: el precio original de un producto y el porcentaje de descuento a aplicar. La función debe calcular y devolver el precio final después de aplicar el descuento.

4. Funciones



Funciones pre-definidas para strings.

- Estas funciones ya hacen parte de PHP y se usan de forma pre-definida:
- Strlower => Todo a minúscula
- Strtoupper => Todo a mayúsculas.
- Ucfirst => La primer letra a mayúsculas.
- Ucwords => La primer letra de cada palabra a mayúscula.
- Strlen => Contar cantidad de caracteres.
- Str_word_count => Devuelve la cantidad de palabras de una frase

4. Funciones



DESARROLLE:

1. Crea una función llamada "esPalindromo" que reciba una cadena de texto como parámetro y devuelva true si la cadena es un palíndromo y false si no lo es. Un palíndromo es una palabra o frase que se lee igual de izquierda a derecha y de derecha a izquierda, ignorando los espacios y signos de puntuación.
2. Escribe una función llamada "generarFibonacci" que reciba un número entero como parámetro y genere y devuelva una lista con la secuencia de Fibonacci hasta ese número. La secuencia de Fibonacci comienza con 0 y 1, y cada número subsiguiente es la suma de los dos anteriores (0, 1, 1, 2, 3, 5, 8, ...).
3. Crea una función llamada "ordenarArray" que reciba un arreglo de números como parámetro y lo ordene de menor a mayor utilizando el algoritmo de ordenamiento burbuja.
4. Escribe una función que calcule la suma de todos los dígitos de un número entero dado. Por ejemplo, si el número es 1234, la suma sería $1 + 2 + 3 + 4 = 10$.
5. Escribe una función llama "FierroAlv" que imprima los números del 1 al 100, pero para los múltiplos de 3 imprime "Peso" en lugar del número y para los múltiplos de 5 imprime "Pluma". Para los números que son múltiplos de ambos, imprime "PesoPluma".

4. Funciones



DESARROLLE:

Tu tarea es programar un juego de Poker básico utilizando funciones y ciclos en PHP. El juego será de una sola mano, donde se repartirán 5 cartas al jugador y se evaluará la mejor combinación posible de esas cartas según las reglas del Poker.

Instrucciones:

1. Define una función llamada "repartirCartas" que no reciba ningún parámetro. Esta función deberá generar y devolver un arreglo con 5 cartas aleatorias del mazo. Puedes representar cada carta como una cadena de texto, por ejemplo: "As de Corazones", "Reina de Picas", etc. Puedes utilizar un arreglo asociativo o una matriz para representar el mazo de cartas.
2. Define una función llamada "mostrarCartas" que reciba un arreglo de cartas como parámetro. Esta función deberá mostrar por pantalla las cartas que se le pasen como argumento, una por línea.
3. Define una función llamada "evaluarMano" que reciba un arreglo de cartas como parámetro. Esta función deberá evaluar la mejor combinación posible de cartas según las reglas del Poker y mostrar por pantalla el resultado obtenido.

4. Funciones



DESARROLLE:

- Puedes implementar la evaluación de la mano utilizando estructuras condicionales y funciones adicionales si es necesario.
 - Algunas combinaciones de manos posibles en orden de mayor a menor valor son: escalera real (A, K, Q, J, 10 del mismo palo), escalera de color, póker (4 cartas del mismo valor), full house (3 cartas del mismo valor y 2 cartas del mismo valor), color (5 cartas del mismo palo), escalera (5 cartas consecutivas de palos diferentes), trío (3 cartas del mismo valor), dos pares, par y carta alta.
4. En el programa principal, utiliza las funciones definidas anteriormente para repartir las cartas al jugador, mostrar las cartas repartidas y evaluar la mano del jugador.
5. Puedes agregar funcionalidades adicionales al juego, como permitir que el jugador descarte y reemplace ciertas cartas antes de evaluar la mano final.

Recuerda utilizar ciclos y funciones de manera adecuada para mantener tu código ordenado y reutilizable.

¡Diviértete programando el juego de Poker en PHP y desafiando las reglas del azar!



G R A C I A S

Línea de atención al ciudadano: 01 8000 910270

Línea de atención al empresario: 01 8000 910682



www.sena.edu.co