Programación III

Tecnicatura Universitaria en Web
Tecnicatura Universitaria en Redes (Optativa)

- 2022 -

Programación III

T.U.W. – T.U.R.

Introducción a PHP

El lenguaje PHP

Extensión de los Archivos

- .php3 Indica código PHP 3.x.
- .php4 Indica código PHP 4.x.
- .php Indica código PHP.
- .phtml Actualmente en desuso.

Delimitadores

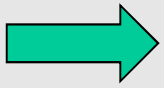
```
<? echo 'Primer método de delimitar código PHP'; ?>
```

```
<?php echo 'Segundo método, el más usado'; ?>
```

```
<% echo 'Método de compatibilidad con ASP'; %>
```

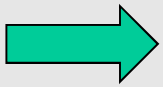
Delimitadores. Ejemplo.

```
<html>
<body>
<?php
if (date("H") >19 || date("H")<4)
{
    echo "<h1>Buenas noches.</h1>";
}
else
{
    echo "<h1>Buenos d&iacute;as. </h1>";
}
?>
</body>
</html>
```



Delimitadores. Ejemplo.

```
<html>
<body>

<?php if (date("H")>19 || date("H")<4) { ?>

<h1>Buenas noches.</h1>

<?php } else { ?>

<h1>Buenos d&iacute;as.</h1>

<?php }?>

</body>
</html>
```



Comentarios

```
/* Comentarios estilo C.
   Pueden extenderse durante varias líneas.
 */

// Comentarios estilo C++. Hasta fin de línea.

# Comentarios estilo Perl. Hasta fin de línea.
```

Variables. Declaración y Uso.

- NO hace falta declararlas
- Llevan delante el signo '$'.

```
$var_1 = 123;
$var_2 = 'hola';
$var_3 = $var_1 * 2;
```

Variables. Tipado.

Variables débilmente tipadas (tipo *mixed*).

```
$mi_variable = 'Inicializamos como texto';
$mi_variable = 3;                        // Entero.
$mi_variable = 3.14 * $mi_variable; // Float.
$mi_variable = new MiClase();        // Objeto.
```

Variables. Tipado. Conversión automática.

PHP realiza conversiones automáticas de tipo:

```
$mivar = 123;
echo $mivar;  // Se convierte a string
```

```
$mivar = '3';          // Se convierte a entero
$mivar = 2 + $mivar; // para realizar la suma
```

Variables. Tipado. Conversión explícita.

Operador cast:

```
$mivar = (string)123;
```

Cambiar el tipo de una variable:

```
$mivar = 12;
settype($mivar, "double");
```

Variables. Ámbito.

- En el cuerpo de un archivo, las variables son **GLOBALES** al archivo y archivos incluidos.
- En una función, son **LOCALES** a esa función.
- Dentro de una clase, sólo pueden ser accedidas a través del operador **"->"** sobre el nombre del objeto.

Referencias.

Se definen con el carácter '&':

```
$alias = &$variable
```

Se puede eliminar una referencia con la función *unset()*:

```
$a = 1;
$b = &$a;
unset ($a);
```

Tipos de datos.

- Enteros, en decimal, octal o hexadecimal.

  $MiVar = 123;

- Punto flotante.

  $MiVar = 1.3e4;

- Arrays.

  $MiVar[2] = 123;

- Strings.

  $MiVar = "Cadena de texto\n";

- Objetos:

  $MiVar = new MiClase();

Tipos de datos. Arrays.

```
$MiArray[0] = 1;

$MiArray[1] = “hola!!”;

$MiArray[] = 3;

echo $MiArray[2];  // 3
```

Tipos de datos. Arrays (2).

Funcionan como vectores o tablas hash al mismo tiempo:

```
$MiArray[“nombre”] = “Juan”;
echo $MiArray[0];              // 1
echo $MiArray[“nombre”]; // “Juan”
```

Y pueden tener más de una dimensión:

```
$MiOtroArray[1][“pepe”][4] = “3 dimensiones!”;
```

Tipos de datos. Arrays (3).

También se pueden definir con el constructor *array()* :

```
$OtroArrayMas = array( 1, "hola", 5);

$YOtroArray = array(
    0 => 1,
    1 => "hola",
    2 => 5,
    3 => 8,
    "nombre" => "Juan"
);
```

Tipos de datos. Strings. Comillas dobles.

- Si se delimitan entre comillas dobles ("), se expandirá cualquier variable que haya dentro de la cadena. Además, se pueden incluir ciertas secuencias de escape, al igual que en C:

Secuencia	Significado
\n	Nueva línea
\r	Retorno de carro
\t	Tabulació n horizontal
\\	Barra invertida
\$	Símbolo del dólar
\"	Dobles comillas
\[0-7]{1,3}	Carácter en octal
\x[0-9A-Fa-f]{1,2}	Carácter en hexadecimal

Tipos de datos. Strings (2). Comillas simples.

- Si se delimitan entre comillas simples (’), las variables no se expanden y además las únicas secuencias de escape que se reconocen son “\\” y “\'” (barra invertida y comillas simples.)

Tipos de datos. Strings (3).

Para concatenar cadenas se utiliza el operador '.' :

```
$cad = 'A esta cadena ';
$cad = $cad . 'le vamos a añadir más texto.';
```

Se puede acceder a cada caracter como si fuera un array:

```
$cad2 = "Tercer caracter de \$cad : '$cad[2]'";
```

Constantes.

Las constantes se definen con la función *define()*:

```
int define(string nombre, mixed valor [, int
                noMayusculas])

Ejemplo:
    define("SALUDO", "Hola, mundo!");
    echo "La constante SALUDO vale " . SALUDO;
```

Las constantes en PHP se diferencian de las variables en que:

- No llevan el símbolo '$' delante.
- Puede accederse a ellas desde cualquier parte del código donde han sido definidas, sin restricciones de ámbito como en las variables.
- No pueden ser redefinidas o borradas una vez definidas.
- Sólo pueden contener valores escalares, no vectores.

Mayúsculas y minúsculas.

Comportamiento mixto en variables y funciones:

- En las variables, las mayúsculas y minúsculas IMPORTAN.
- En los nombres de funciones y palabras reservadas, las mayúsculas NO IMPORTAN.

Operadores aritméticos.

Operación	Nombre	Resultado
$a + $b	Suma	Suma de $a y $b.
$a - $b	Resta	Diferencia entre $a y $b.
$a * $b	Multiplicación	Producto de $a y $b.
$a / $b	División	Cociente de $a y $b.
$a % $b	Módulo	Resto de la operación $a/$b.

Auto-incremento y Auto-decremento.

Operación	Nombre	Resultado
++$a	Pre-incremento	Incrementa $a en 1, y devuelve $a (incrementado).
$a++	Post-incremento	Devuelve $a, y después lo incrementa en 1.
--$a	Pre-decremento	Decrementa $a en 1, y después lo devuelve.
$a--	Post-decremento	Devuelve $a, y después lo incrementa en 1.

Operadores de bits.

Operación	Nombre	Resultado
$a & $b	Y	Se ponen a 1 los bits que están a 1 en $a y $b.
$a | $b	O	Se ponen a 1 los bits que están a 1 en $a o $b.
$a ^ $b	O Exclusivo	Se ponen a 1 los bits que están a 1 en $a o $b, pero no en ambos.
~ $a	No	Se invierten los bits (se cambian 1 por 0 y viceversa.)
$a << $b	Desp. Izq.	Desplaza $b posiciones a la izquierda todos los bits de $a.
$a >> $b	Desp. Drch.	Desplaza $b posiciones a la derecha todos los bits de $a.

Operadores lógicos.

Operación	Nombre	Resultado
$a and $b	Y	Cierto si $a y $b son ciertos.
$a or $b	O	Cierto si $a o $b es cierto.
$a xor $b	O Exclusivo.	Cierto si $a o $b es cierto, pero no ambos.
! $a	No	Cierto si $a es falso.
$a && $b	Y	Cierto si $a y $b son ciertos.
$a || $b	O	Cierto si $a o $b es cierto.

Operadores. Asignación, igualdad e identidad.

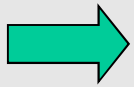
Operación	Nombre	Resultado
$a = $b	Asignación	Asigna el valor de una variable o expresión del segundo término a la variable del primer término.
$a == $b	Igualdad	Compara si el valor de los dos operandos es el mismo.
$a === $b	Identidad	Compara si el valor es el mismo y, además, el tipo coincide.

Operadores. Asignación, igualdad e identidad. Ejemplo.

```
$var1 = 1;          // Asignación
$var2 = 1;
$var3 = "1";
($var1 == $var2)   // Cierto, son iguales
($var1 == $var3)   // Son iguales (tras conversión)
($var1 === $var2) // Cierto, son idénticas
($var1 === $var3) // FALSO, el tipo no coincide
```

Operadores. Asignación, igualdad e identidad. Error.

```
$var1 = 1;
$var2 = 2;
if( $var1 = $var2 )
{
    echo 'iguales';
}
else
{
    echo 'distintas';
}
```

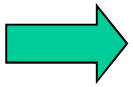


Comparaciones.

Operación	Nombre	Resultado
$a != $b	No igual	Cierto si el valor de $a no es igual al de $b.
$a !== $b	No idéntico	Cierto si $a no es igual a $b, o si no tienen el mismo tipo.
$a < $b	Menor que	Cierto si $a es estrictamente menor que $b.
$a > $b	Mayor que	Cierto si $a es estrictamente mayor que $b.
$a <= $b	Menor o igual que	Cierto si $a es menor o igual que $b.
$a >= $b	Mayor o igual que	Cierto si $a es mayor o igual que $b.

Operadores de cadenas.

```
<?php

$a = 1;
$b = 2;

echo "<h1>El resultado de $a + $b es: ".($a+$b)."</h1>";

?>
```



Atajos en la asignación.

+= -= *= /= %= &= ^= .=
>>= y <<=

```
$var1 += 3;              // $var1 = $var1 + 3;
$var2 /= 2;              // $var2 = $var2 / 2;
$var3 >>= 1;     // $var3 = $var3 >> 1;
```

Precedencia.

,
or
xor
and
print
= += -= *= /= .= %= &= |= ^= ~= <<= >>=
? :
||
&&
|

^
&
== != === !==
< <= > >=
<< >>
+ - .
* / %
! ~ ++ -- (int) (double) (string) (array) (object) @
[
new

Estructuras de control. if ... elseif ... else

```
if (expresión)
{
    comandos
}
```

```
if (expresión)
{
    comandos_cierto
}
else
{
    comandos_falso
}
```

Estructuras de control. if ... elseif ... else (2)

```
if (expresion1)
   {     comandos1    }
elseif (expresion2)
   {     comandos2    }
elseif (expresion3)
   {     comandos3    }
...
else
   {     comandosElse   }
```

while y do ... while

```
while (expresión)
{
    comandos
}
```

```
do
{
    comandos
}
while (expresión);
```

for

```
for (expresión1; expresión2; expresión3)
{
   comandos
}
```

```
$factorial5 = 1;
for ($i = 2; $i <= 5; $i++ )
{
    $factorial5 *= $i;
}
```

for (2)

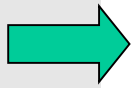
```
for ($factorial5 = 1, $i = 2; $i <= 5; $i++ )
{
    $factorial5 = $factorial5 * $i;
}
```

```
for ($factorial5=1, $i=2;
     $i<=5;
     $factorial5*=$i, $i++);
```

foreach

```
foreach (array as variable)
{
    comandos
}
```

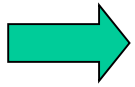
```
$a = array (1, 2, 3, 17);
foreach ($a as $v)
{
    print "Valor actual de \$a:
$v.\n<br>";
}

// Valor actual de $a: 1
// Valor actual de $a: 2
// Valor actual de $a: 3
// Valor actual de $a: 17
```



foreach (2)

```
foreach (array as indice => variable)
{
    comandos
}
```

```
<?php
    $a = array(
    "uno" => 1,
    "dos" => 2,
    "tres" => 3,
    "diecisiete" => 17
);

foreach ($a as $k => $v) {
    echo "\$a[$k] => $v.<br>";
}
?>
```



switch

```
switch (variable)
{
    case valor1:
        comandos1
    case valor2:
        comandos2
    ...
    case valorN:
        comandosN
    default:
        comandosDefault
}
```

switch (2)

```
switch ($i)
{
    case 1:
        echo “Código del 1”;

    case 2:
        echo “Código del 2”;

    case 3:
        echo “Código del 3”;
        break;

    case 4:
        echo “Código del 4”;
}
```

Verdadero o Falso. Valores numéricos.

```
$x = 1;  // $x
if( $x ) // se evalúa a cierto

$x = 0;  // $x definida como el entero 0
if( $x ) // se evalúa a falso
```

Verdadero o Falso. Strings.

```
$x = "hello"; // asignamos una cadena a $x
if( $x )      // se evalúa a cierto

$x = "";      // cadena vacía
if( $x )      // evalúa a falso

              // Excepción:
$x = "0";     // cero en una cadena
if( $x )      // evalúa a falso
              // (se convierte a entero)
```

Verdadero o Falso. Arrays.

```
$x = array();  // $x es un array vacío
if( $x )       // se evalúa como falso

$x = array( "a", "b", "c" );
if( $x )       // se evalúa a cierto
```

Verdadero o Falso. Objetos.

```
Class Yod {}    // clase vacía
$x = new Yod();
if( $x )        // se evalúa a falso

Class Yod {     // clase no vacía
    var $x = 1;
}
$x = new Yod();
if( $x )        // se evalúa a cierto
```

Verdadero o Falso. Constantes.

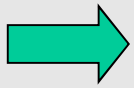
- TRUE es el valor entero decimal 1.
- FALSE es la cadena vacía.

Funciones.

```
function nombre ($arg_1, $arg_2, ..., $arg_n)
{
    comandos
    return $salida;
}
```

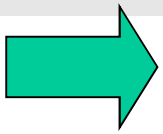
Funciones (2). Ejemplo.

```
function factorial ($valor) {
    if ($valor < 0) {
        return -1; // Error
    }
    if ($valor == 0 ) {
        return 1;
    }
    if ($valor == 1 || $valor == 2) {
        return $valor;
    }
    $ret = 1;
    for ($i = 2; $i <= $valor; $i++) {
        $ret = $ret * $i;
    }
    return $ret;
}
$factorial5 = factorial(5);
```



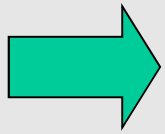
Funciones (3). Valores por defecto.

```
function enlace($url = ”www.php.net”)
{
    echo ‘<a href=”’ . $url . ‘”>Pulsa aquí</a>’;
}


enlace("http://www.unsl.edu.ar");
enlace("http://www.clarin.com.ar");
enlace();
```



Funciones (4). Argumentos por

```
function MiFuncion(&$var)
{
    $var++;
}

$a = 5;
echo 'Variable $a antes de la invocaci&oacute;n a la
funci&oacute;n: '.$a."<br>";

MiFuncion($a);

echo 'Variable $a Despu&eacute;s de la
invocaci&oacute;n a la funci&oacute;n: '.$a;

// Aquí $a == 6
```



Funciones (5). Devolución por referencia.

```
function &buscar_cliente($nombre)
{
    // ... buscamos ...
    return $registro;
}

$cliente =
&buscar_cliente("Juan");
echo $cliente->dni;
```