



Conceptos Básicos PHP

Introducción a la Programación
Facultad de Informática
Univ.Nac. del Comahue



Temas

1° PARTE:

- **Identificadores**
- **Buenas prácticas de programación**
- **Traducir de Pseudocódigo a PHP**

2° PARTE:

- **Herramientas para programar en computadora**

3° PARTE:

- **Variables**
- **Tipos de Datos**
- **Expresiones**



Temas

1º PARTE:

- **Traducir de Pseudocódigo a PHP**
- **Identificadores**
- **Buenas prácticas de programación**



Identificadores

Combinación de caracteres que nombran:

variables, palabras reservadas, constantes, métodos* y **clases****, de forma que puedan ser identificados de forma única.

* Métodos: tema de la unidad 4 de modularización

** Clases: se estudiará en la materia Introd. A la Prog. Orientada a Objetos (IPOO) de la TUDW



Palabras Reservadas PHP

Las palabras reservadas son palabras predefinidas del lenguaje que se utilizan para reconocer diferentes estructuras de un programa. Son necesarias para dar la orden a la computadora.

<http://php.net/manual/es/reserved.keywords.php>

<u>__halt_compiler()</u>	<u>abstract</u>	<u>and</u>	<u>array()</u>	<u>as</u>
<u>break</u>	<u>callable</u> (a partir de PHP 5.4)	<u>case</u>	<u>catch</u>	<u>class</u>
<u>clone</u>	<u>const</u>	<u>continue</u>	<u>declare</u>	<u>default</u>
<u>die()</u>	<u>do</u>	<u>echo</u>	<u>else</u>	<u>elseif</u>
<u>empty()</u>	<u>enddeclare</u>	<u>endfor</u>	<u>endforeach</u>	<u>endif</u>
<u>endswitch</u>	<u>endwhile</u>	<u>eval()</u>	<u>exit()</u>	<u>extends</u>
<u>final</u>	<u>finally</u> (a partir de PHP 5.5)	<u>for</u>	<u>foreach</u>	<u>function</u>
<u>global</u>	<u>goto</u> (a partir de PHP 5.3)	<u>if</u>	<u>implements</u>	<u>include</u>
<u>include_once</u>	<u>instanceof</u>	<u>insteadof</u> (a partir de PHP 5.4)	<u>interface</u>	<u>isset()</u>
<u>list()</u>	<u>namespace</u> (a partir de PHP 5.3)	<u>new</u>	<u>or</u>	<u>print</u>
<u>private</u>	<u>protected</u>	<u>public</u>	<u>require</u>	<u>require_once</u>
<u>return</u>	<u>static</u>	<u>switch</u>	<u>throw</u>	<u>trait</u> (a partir de PHP 5.4)
<u>try</u>	<u>unset()</u>	<u>use</u>	<u>var</u>	<u>while</u>
<u>xor</u>	<u>yield</u> (a partir de PHP 5.5)			

En Pseudocódigo:
ESCRIBIR



No se puede usar palabras reservadas como identificadores de variables.



Palabras Reservadas PHP

En la materia de Introducción a la Programación incorporaremos distintas palabras reservadas.

Muchas de estas palabras reservadas nos permitirán ampliar la cantidad de órdenes o instrucciones que podemos ejecutar en una computadora.



Identificadores de Constantes

- Una constante es un identificador para un valor. Como el nombre sugiere, este valor no puede variar durante la ejecución del programa.
- Una constante tendrá un nombre, un valor y un tipo de dato, pero a excepción de las variables, una vez declarado su valor no se puede modificar.
- En general y por convención, como nombre de constante se utilizan palabras en mayúsculas

Un ejemplo muy conocido es el número **PI** cuyo valor es **3,1415**



Veremos que PHP provee constantes predefinidas como es el caso de PI (<https://www.php.net/manual/es/function.pi.php>)



Identificadores de Variables

- Comienzan con una letra o un guión bajo (underscore), seguidos de cualquier cantidad de letras, números y guiones bajo. No permitimos vocales acentuadas, ñ, símbolos extraños, espacios en blanco.
- Es válido utilizar letras mayúsculas, pero en el caso de Variables y Métodos (tema de unidad 4) por **convención** sólo utilizaremos letras mayúsculas para inicializar las palabras, excepto la 1ª letra. (utilizaremos notación lowerCamelCase https://es.wikipedia.org/wiki/Camel_case)

Ejemplos válidos: **areaRectangulo**, **unNumeroPar**, **num1**...

Ejemplos No válidos: **area Rectangulo**, **1NumeroPar**, **porc%**, ...



- El caracter guión bajo (underscore),_, puede aparecer en un nombre. Sin embargo, en la materia trataremos de evitarlo para iniciar un nombre de variable.



Buenas Prácticas de Programación

Identificadores de Variables

Una buena práctica de programación comprende elegir:

- * **Utilizar una convención de nombrado**
- * **nombres significativos para las variables,**
- * **documentar su tipo y**
- * **describir en pocas palabras su uso**



Identificador de Variables

	Pseudocódigo	PHP
Identificador de Variables	<p>Identificador: siempre empieza con un una letra, luego sigue una combinación de letras y números. Utilizaremos notación lowerCamelCase: cuando la primera letra de cada una de las palabras es mayúscula con la excepción de que la primera letra que es minúscula.</p> <p>ejemplos: ladoMenor, ladoMayor, nota1, perimetroRectangulo</p>	<p>Identificador (con la misma convención que pseudocódigo) al que antepondremos el signo \$</p> <p>ejemplos: \$ladoMenor, \$ladoMayor, \$nota1, \$perimetroRectangulo</p>



PHP es case sensitive, es decir, sensible a las mayúsculas y minúsculas. Las variables \$var , \$Var, \$VAR, ... son variables distintas. Conviene respetar la notación lowerCamelCase para evitar errores⁴⁰



Comentarios

Toda línea que se encuentra comentada, no se ejecuta.

	Pseudocódigo	PHP
Comentarios	<p>En pseudocódigo utilizaremos:</p> <p>(* simbolos de comentarios *)</p>	<p>En PHP utilizaremos:</p> <p>/* simbolos de comentarios de muchas líneas */</p> <p>// comentarios de una línea</p>

Los comentarios son anotaciones legibles para el programador, y tienen el propósito de hacer el código fuente (especificación del programa) más fácil de entender, mejorando su mantenimiento y/o reuso.

Los comentarios son “invisibles” para la computadora, no son instrucciones, simplemente sirven para que, cuando otro programador lea el código fuente, pueda comprender mejor lo que lee.



Buenas Prácticas de Programación

Comentarios

Utilizar comentarios es una buena práctica de programación:

son útiles para los programadores, facilitan la legibilidad de un programa . Por ejemplo, podemos utilizar un comentario para explicar el por qué del uso de una instrucción o expresión, o cuál es el objetivo de crear una variable determinada.

Consigna de la materia: “programar para trabajar en equipo con otros programadores”





Traducción de Instrucciones de Pseudocódigo a PHP

- Instrucción de Asignación
- Instrucción de Entrada
- Instrucción de Salida



Instrucción de Asignación

	Pseudocódigo	PHP
Asignación	<p>nombreVariable \leftarrow Expresion</p> <p>Ejemplo:</p> <p>ladoMenor \leftarrow 20 ladoMayor \leftarrow ladoMenor*4</p> <p><i>Un instrucción debajo de la otra.</i></p>	<p>\$nombreVariable = Expresion ;</p> <p>Ejemplo:</p> <p>\$ladoMenor = 20; \$ladoMayor = \$ladoMenor * 4;</p> <p><i>Las instrucciones en PHP finalizan con ; Por legibilidad del código fuente: una instrucción debajo de la otra</i></p>



Instrucción de Salida

	Pseudocódigo	PHP
Salida / Escritura	<p>Ejemplo:</p> <p>ESCRIBIR("El perímetro del rectángulo es " , perimetroRectangulo)</p>	<p>Ejemplo:</p> <p>echo "El perímetro del rectángulo es " . \$perimetroRectangulo ;</p> <p><i>Las instrucciones en PHP finalizan con ;</i></p>



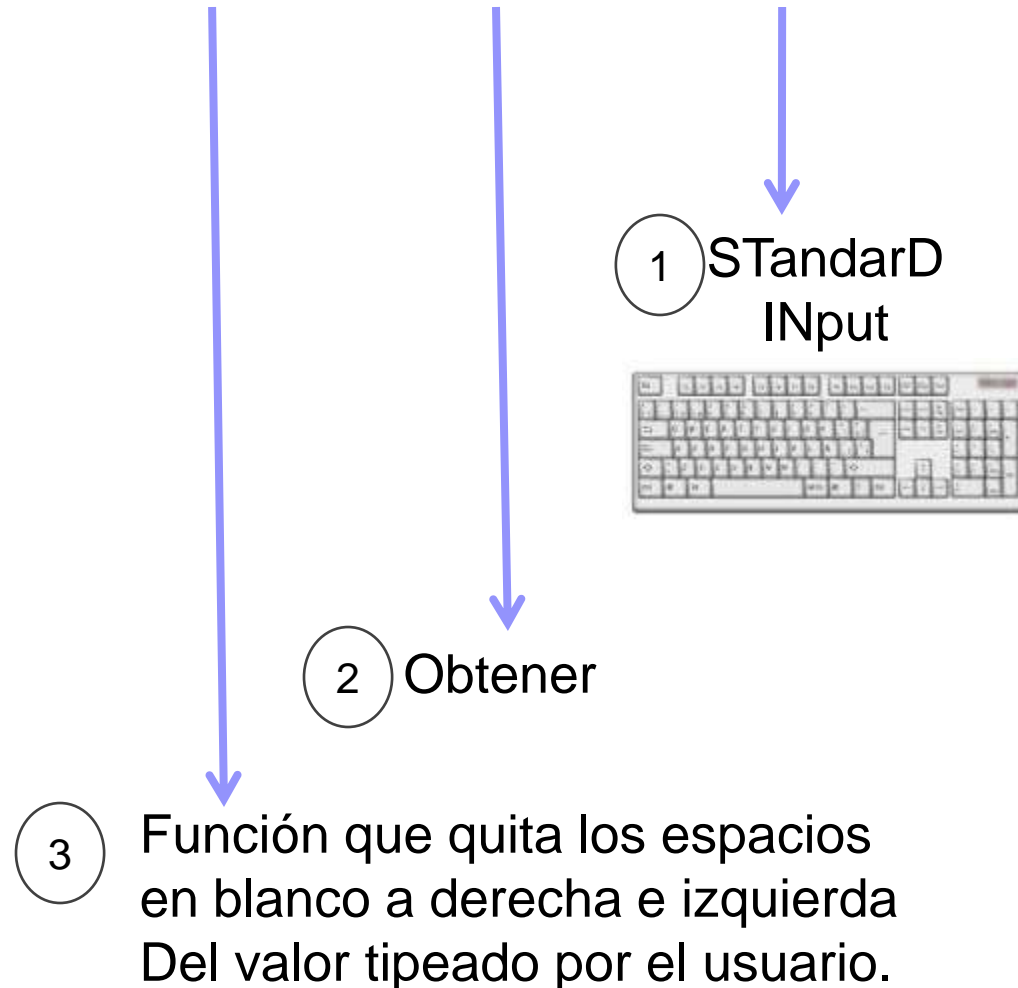
Instrucción de Entrada

	Pseudocódigo	PHP
Entrada / Lectura	<p>Ejemplo:</p> <p>LEER (ladoMenor)</p>	<p>Ejemplo:</p> <pre>\$ladoMenor = trim(fgets(STDIN)) ;</pre> <p><i>Las instrucciones en PHP finalizan con ;</i></p>



Instrucción de Entrada

```
$ladoMenor = trim( fgets ( STDIN ) );
```





Traducción de Pseudocódigo a PHP

Pseudocódigo	PHP
<p>PROGRAMA nacimiento</p> <p>(*Calcula el año de nac. de una persona *)</p> <p>String nombre, Entero edad, anio, anioNac</p> <p>ESCRIBIR("Ingrese su nombre:")</p> <p>LEER(nombre)</p> <p>ESCRIBIR("Ingrese edad:")</p> <p>LEER(edad)</p> <p>anioActual \leftarrow 2018</p> <p>anioNac \leftarrow anio - edad</p> <p>ESCRIBIR (nombre, " naciste en ", anioNac)</p> <p>FIN PROGRAMA</p>	<pre><?php /*Etiqueta que indica que comienza un programa PHP*/ /* PROGRAMA nacimiento */ /*Calcula el año de nac.de una persona */ /*String nombre, Int edad, anioActual, anioNac*/ echo "Ingrese su nombre: "; \$nombre = trim(fgets(STDIN)); echo "Ingrese edad: "; \$edad = trim(fgets(STDIN)); \$anioActual = 2018; \$anioNac = \$anio - \$edad; echo \$nombre . " naciste en ". \$anioNac;</pre> <p>?></p> <p>Si el código fuente sólo especifica instrucciones PHP Se recomienda no incluir el tag de finalización ">"</p>



Notar que en PHP El tipo de una variable **no se declara**, sino que **se establece durante la ejecución del programa y depende del valor que se le asigne a cada variable**. Como programadores sabemos los valores que se esperan asuman las variables. Por eso las declaramos en pseudocódigo



Temas

2º PARTE:

- Herramientas para programar en computadora**



¿Qué es el Intérprete?

Un intérprete **es un programa** capaz de analizar y ejecutar otros programas

```
holaMundo.php
1 <?php
2
3 echo "Estoy ejecutando codigo PHP! \n";
4 echo "hola Mundo!";
5
6 ?>
```

Console

Estoy ejecutando codigo PHP!
hola Mundo!

Programa Fuente

php
Intérprete

Traduce y ejecuta
línea a línea

En caso que
se produzca
un

Error de
sintaxis

Editar, corregir





Console

PHP Parse error: syntax error, ...
... in holaMundo.php on line 4



Herramientas para programar

- a) **Intérprete PHP**: traduce cada instrucción de un programa escrito en PHP, a lenguaje máquina para que pueda ser ejecutado. 
- b) **IDE (Entorno de desarrollo Integrado)**: Utilizaremos VSCode 

Ventajas:

- 1- Organiza la forma de trabajo en proyectos
 - 2- Tiene integrado un editor de texto que resalta la sintaxis php. Además instantáneamente vemos errores de sintaxis sin necesidad de ejecutar el programa.
 - 3- tiene autocompletado y formateo de código. (por ejemplo identa el código)
 - 4- Integramos el intérprete PHP al IDE.
 - 5- facilita la navegación del código fuente utilizando atajos del teclado
- Entre otras a medida que aprendamos a utilizarlo.



Temas

3º PARTE:

- **Variables**
- **Tipos de Datos**
- **Expresiones**



Variables



Variable

De manera intuitiva una variable es un contenedor (caja) donde es posible almacenar un dato. Puede ir cambiando su contenido y tiene los siguientes atributos:

Nombre: Permite identificar o referenciar a una variable.
(identificador de la variable)

Valor/dato: Es el contenido almacenado en una variable.

Almacenar un dato, implica la destrucción de cualquier otro dato almacenado en esa variable.

Tipo de Dato: Describe el tipo de información que almacena la variable (entero, float/real, booleano/lógico y string). **Determina las operaciones que podrán realizarse con cada variable.**



Variable

Conceptos:

Declaración: es el momento en que se define el Tipo de Dato y el nombre de la variable: <tipo> <identificador>

Ejemplo: **String** mensaje,
int indice

Inicialización: Es la asignación del primer valor a una variable. En lenguajes como PHP *la variable se considera definida cuando es inicializada.*

(Utilizar una variable no definida en una expresión produce un error de programación: “undefined variable”)



Ejemplo:

mensaje ← “Bienvenido a IP”
indice ← 0



Utilizar una variable no definida en una expresión produce un error de programación

PROGRAMA Velocidad

(* muestra la velocidad de un auto identificado por su patente*)

String patente, Float velocidad → **Declaración**

patente ← "AA 234 BB" → **Inicialización variable patente**

ESCRIBIR(patente. " viaja a ", velocidad , "km/h")

FIN PROGRAMA



→ **Uso de Variable no definida en una expresión**

<?php

/* velocidad */

/* muestra la velocidad de un auto identificado por su patente*/

/* String \$patente, Float \$velocidad */

\$patente = "AA 234 BB";

echo \$patente. " viaja a " . \$velocidad . "km/h";



→ **Como PHP cuenta con tipo de datos dinámicos dejamos comentada la declaración a modo de documentación del programa**



Los Lenguajes de Programación según el momento en que se determina el Tipo de Dato de las Variables

Lenguajes de programación

• con tipos de datos

estáticos:

El Tipo de Dato se determina antes de la ejecución del programa (en la compilación), se declaran los tipos de datos de las variables en la especificación del programa.

Java, C, C++, Pascal

• con tipos de datos

dinámicos:

El Tipo de Dato se determina durante la ejecución del programa.
El tipo de la variable depende del valor que se le asigna.
Una variable puede asumir distintos tipos durante la ejecución del programa

PHP, Python, Java Script 29



Los Lenguajes de Programación según el momento en que se determina el Tipo de Dato de las Variables

Cómo **PHP** es un lenguaje donde **el tipo de dato de las variables se determina dinámicamente** (durante la ejecución del programa),

en la materia de Introducción a la Programación trabajaremos de la siguiente manera:



Los Lenguajes de Programación según el momento en que se determina el Tipo de Dato de las Variables

- 1°- Se debe diseñar el algoritmo en pseudocódigo, donde **es obligación** declarar el tipo de la variable y asegurar que el valor que se le asigne a la variable corresponde con el tipo de dato declarado.
- 2°- Se traducirá el algoritmo de Pseudocódigo a PHP respetando los especificado en el paso 1°.
- 3° - En el **parcial** se considerará error si se declara una variable de un determinado Tipo de Dato y se le asigna un valor que no corresponde con dicho tipo de dato. (La excepción es que a una variable float, le asigne un entero)





Tipo de Datos



Tipo de dato de una variable

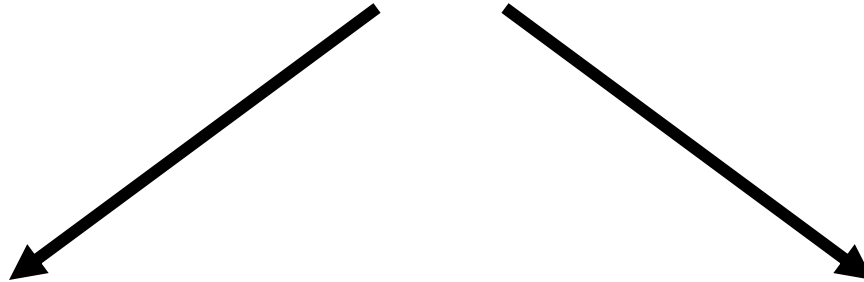
Determina los valores que puede tomar una variable y las operaciones que pueden realizarse con ella:

Tipo	Ejemplos de valores	Algunas operaciones que permite realizar el tipo de dato
Entero	-25 45 0 11247	Suma, Resta, Multiplicación, División, Resto Comparación por mayor, igual, menor ...
Float o Real	-25.02 0.24 3.142 124.58	Suma, Resta, Multiplicación, División Comparación por mayor, igual, menor ...
Booleano o lógico	true false	or, and, not ...
String (cadena de caracteres)	"hola mundo" "abcd" "124" "tengo 32 años"	Concatenación Comparación por mayor, igual, menor ...



Tipo de Dato

Hay dos **tipos de datos** fundamentales



- **Primitivos:**
los proporciona
el lenguaje de
programación.

- **Definidos por el usuario:**
tipos que define el
programador.



Tipo de Dato Primitivos

PHP soporta 8 tipos de datos primitivos

Cuatro tipos escalares:

- **boolean**: expresa un valor de verdad, puede ser TRUE (verdadero) o FALSE (falso). (<http://php.net/manual/es/language.types.boolean.php>)
- **integer**: con frecuencia denominados enteros o int, son números enteros positivos o negativos -sin punto decimal-.
- **Float**: representan números reales y son escritos con un punto decimal que divide la parte entera de la parte fraccionaria. Hay otro tipos de datos dependiendo de la precisión, también conocido como **double**. Aunque dependiendo del lenguaje de programación float y double tienen la misma precisión. En la materia los usaremos como sinónimos.
- **string**: cadenas de caracteres. No olvidemos que en PHP una cadena de caracteres se indica entre comillas simples o entre comillas dobles.



Tipo de Dato Primitivos

Dos tipos compuestos:

- **array** : Este tipo puede tener varios usos diferentes; puede ser usado como una matriz, una lista (vector), una tabla asociativa, etc.
- **object**

Dos tipos especiales:

- **Resource**: una variable de este tipo contiene referencias a un recurso externo (archivo abierto, conexión con una base de datos,...)
- **NULL**: El valor especial NULL representa una variable sin valor. NULL es el único valor posible del tipo null.



Expresiones

(Operaciones entre datos)



EXPRESIONES

- Operadores y operandos
- Operadores Aritméticos
- Operadores Lógicos
- Operadores comparación
- Precedencia de Operadores



EXPRESIONES

Son una parte fundamental de la programación ya que sirven para realizar una o varias operaciones sobre un dato o un conjunto de datos (Operandos), obteniéndose otro dato como resultado.

Una expresión es una combinación de valores, variables y operadores.

<?php

echo 1+1 ;



El intérprete:

1° resuelve la expresión 1+1

2° Muestra el resultado en pantalla

?>

Pantalla:

2



EXPRESIONES

<?php

echo **EXPRESION** ;



Es decir: El intérprete resuelve la expresión y luego imprime el resultado

?>

En la materia de Introducción a la Programación:

- vamos a mostrar en pantalla los resultados de expresiones enteras, float y string,
- vamos a **evitar** mostrar el resultado de expresiones booleanas . *(pensemos que a un usuario no le interesa ver en pantalla un valor true o false, sino un mensaje claro de la respuesta a su problema)*



Paciencia!... Ya veremos más adelante cómo vamos a mostrar un mensaje a partir de la evaluación de una expresión Booleana. (diapositiva 47)



EXPRESIONES

Orden de evaluación en una asignación

Para ejecutar una instrucción de Asignación, primero se evalúa y resuelve la expresión del lado derecho del operador de asignación '='.

Luego, el resultado se utiliza para inicializar el valor de la variable a la izquierda del operador de asignación '='.

<?php

/* int \$huevosPorCanasta */

\$huevosPorCanasta = 24 ;

\$huevosPorCanasta = \$huevosPorCanasta - 2 ;

?>

Asignación

Expresión Lado
Derecho del op. de Asig.



EXPRESIONES

<?php

\$variable = EXPRESION; → Es decir: El intérprete resuelve la expresión y luego almacena el resultado en la variable.

?>

La expresión puede resultar ser un float, un entero, un string o un booleano.

En la materia de Introducción a la Programación:



- Si declaramos que la variable es numérica, le asignaremos una expresión cuyo resultado sea numérico
- Si declaramos que la variable es string, le asignaremos una expresión cuyo resultado sea string.
- Si declaramos que la variable es booleana, le asignaremos una expresión cuyo resultado sea booleano.



EXPRESIONES

Operadores y Operandos

Un operador es un carácter o una secuencia de caracteres (+, *, &&).

Los operadores definen las operaciones que van a realizarse con los datos u operandos (literales, constantes, variables, funciones).



EXPRESIONES

Operadores y Operandos

Los **operadores** se pueden clasificar según

• **Número De Operandos:**

- unarios,
- binarios,
- ternarios

• **Tipo de operandos y de su resultado :**

- aritméticos,
- de cadenas de caracteres,
- de comparación,
- lógicos o booleanos



EXPRESIONES

Operadores

Según Número De Operandos:

El operador **unario** sólo aplica a un operando, por ejemplo:

- ! (el operador lógico de negación)
- (el operador aritmético de negación)

Ejemplos:

Pseudocodigo	PHP
NOT aprobo	!\$aprobo
- temperatura	-\$temperatura

```
<?php
/*boolean $esMayor
float $gradosCentigrados */
$esMayor = false;
$esMenor = ! $esMayor;  —————> ! false resulta ser true

$gradosCentigrados = -5;
$gradosCentigrados = - $gradosCentigrados; —————> -(-5) resulta ser 5
```



EXPRESIONES

Operadores

Según Número De Operandos:

El operador **binario** opera sobre dos operandos, como los operadores aritméticos + (suma), - (resta), MOD (resto de división) o los operadores de comparación < (menor) y <= (menor igual). La mayoría de los operadores entran en esta categoría.

Ejemplos:

Pseudocodigo	PHP
a + b	\$a + \$b
a <= b	\$a <= \$b
a / 4	\$a / 4
a MOD 4	\$a % 4



EXPRESIONES

Operadores

Según Número De Operandos:

Finalmente, hay sólo un operador **ternario**, que opera sobre tres operandos;

Ejemplos:

Pseudocodigo	PHP
SI E1 ENTONCES E2 SINO E3	E1? E2 : E3

E1 es una expresión booleana (al evaluarse su resultado es True o False)

E2 es la expresión booleana, string, float o entera que se resuelve si E1 es True

E3 es la expresión, del mismo tipo de E2, que se resuelve si E1 es False

Ejemplo:

String calificacion, Boolean aprobo, Int nota

LEER(nota)

aprobo ← nota > 60

calificacion ← **SI** aprobo **ENTONCES** “Felicitaciones” **SINO** “La próxima será”

ESCRIBIR (calificacion)



EXPRESIONES

Operadores según Tipo de operandos y de su resultado

Operadores Aritméticos

Ejemplo	Nombre	Resultado
$-\$a$	Negación	Opuesto de $\$a$.
$\$a + \b	Adición	Suma de $\$a$ y $\$b$.
$\$a - \b	Sustracción	Diferencia de $\$a$ y $\$b$.
$\$a * \b	Multiplicación	Producto de $\$a$ y $\$b$.
$\$a / \b	División	Cociente de $\$a$ y $\$b$.
$\$a \% \b	Módulo	Resto de $\$a$ dividido por $\$b$.

Los operandos son Numéricos y el resultado es Numérico

<?php

```
/*int $a, $b , $sum , $resto */
```

```
$a = 15;
```

```
$b = 10;
```

```
$sum = $a + $b ; /* la suma resulta ser 25 */
```

```
$resto = $a % $b ; /* el resto o módulo resulta ser 5*/
```



EXPRESIONES

Operadores según Tipo de operandos y de su resultado

Operadores Lógicos

| Pseudocodigo | PHP |
|----------------|---------------------------|
| NOT a | ! \$a |
| a AND b | \$a && \$b |
| a OR b | \$a \$b |

Los operandos son Booleanos y el resultado es Booleano

Tablas de verdad para recordar los resultados de las operaciones NOT, AND y OR:

| a | NOT a |
|-------|--------------|
| true | false |
| false | true |

| a | b | a AND b |
|-------|-------|----------------|
| true | true | true |
| true | false | false |
| false | true | false |
| false | false | false |

| a | b | a OR b |
|-------|-------|---------------|
| true | true | true |
| true | false | true |
| false | true | true |
| false | false | false |

<http://php.net/manual/es/language.operators.logical.php>



EXPRESIONES

Operadores según Tipo de operandos y de su resultado

Operadores Lógicos

Tablas de verdad para recordar los resultados de las operaciones NOT, AND y OR:

| a | NOT a |
|-------|-------|
| true | false |
| false | true |

| a | b | a AND b |
|-------|-------|---------|
| true | true | true |
| true | false | false |
| false | true | false |
| false | false | false |

| a | b | a OR b |
|-------|-------|--------|
| true | true | true |
| true | false | true |
| false | true | true |
| false | false | false |

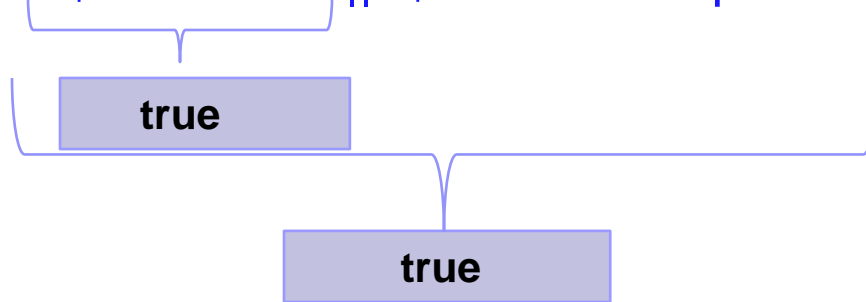
<?php

```
/*boolean $esMenor, $estaEmancipado , $puedeVotar*/
```

```
$esMenor = false;
```

```
$estaEmancipado = false;
```

```
$puedeVotar = ! $esMenor || $estaEmancipado ;
```





EXPRESIONES

Operadores según Tipo de operandos y de su resultado

Operadores de Comparación

| Pseudocodigo | PHP | Resultado |
|---------------------|-------------------------|-------------------------|
| a = b | \$a == \$b | True si a igual b |
| a <> b | \$a <> \$b | True si a distinto b |
| a < b | \$a < \$b | True si a menor b |
| a > b | \$a > \$b | True si a mayor b |
| a <= b | \$a <= \$b | True si a menor igual b |
| a >= b | \$a >= \$b | True si a mayor igual b |

Los operandos pueden ser numéricos y el resultado es Booleano

Los operandos pueden ser string y el resultado es Booleano

<?php

```
/*boolean $esMenor, $estaOrdenado, $edad*/
```

```
$edad = 45;
```

```
$esMenor = $edad <= 18;
```

```
$estaOrdenado = "garcia" < "martinez" /* compara por orden alfabetico */
```



EXPRESIONES

Operadores de Comparación

Los operadores de comparación se utilizan para hacer comparaciones entre valores enteros, reales o de cadena (literales o almacenados en variables).

El resultado de utilizar un operador de Comparación es Verdadero o Falso (True o False)

<?php

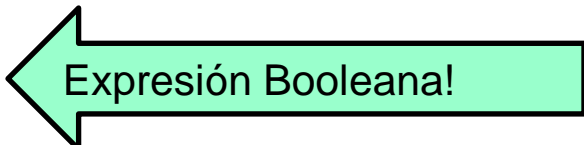
\$nombre1 = "Patricia";

\$nombre2 = "Maria"

\$ordenado = \$nombre1 > \$nombre2;

echo \$ordenado? "Estan ordenados" : "No estan ordenados";

?>



Expresión Booleana!



EXPRESIONES

Operadores según Tipo de operandos y de su resultado

Operadores de Cadena de Caracteres (string)

Pseudocodigo	PHP
a , b	\$a . \$b

El operador de concatenación ('.'), retorna un string como resultado de concatenar sus operandos derecho e izquierdo.

<?php

\$cadena = "Mundo!";

echo "Hola " . \$cadena ;



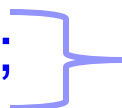
Hola Mundo!

?>

<?php

\$var = 3 + 4;

echo "Resultado: " . \$var ;



Resultado: 7

?>



Resumen según el tipos de datos de operandos y resultado

Operador	Tipo de operandos	Tipo de resultado
Aritméticos	Numéricos (float o Int)	Numérico (float o Int)
Booleanos o lógicos	Booleanos	Booleano
De comparación	Numéricos String	Booleano
Concatenación	String / numérico	String



EXPRESIONES

Operadores y Operandos

Clasificamos los **operadores** según

• **Número De Operandos:**

- unarios,
- binarios,
- ternarios

• **Tipo de operandos y de su resultado :**

- aritméticos,
- de cadenas de caracteres,
- de comparación,
- lógicos o booleanos



EXPRESIONES

¿En qué orden el intérprete resuelve las expresiones?

- Reglas de Precedencia
- Reglas de Asociatividad
- Utilización de Paréntesis



EXPRESIONES

¿En qué orden el intérprete resuelve las expresiones?

Por ejemplo:

$$1 + 5 * 3$$

¿Cuál es el resultado?

16

18





EXPRESIONES

Precedencia de Operadores

Cuando hay más de un operador en una expresión, **el orden de evaluación depende de las reglas de precedencia de los operadores**.

Si una expresión **no contiene paréntesis** será evaluada de acuerdo a las reglas de precedencia.

La precedencia de operadores se refiere al orden en que se resuelven las operaciones de una expresión.

Por ejemplo, en la expresión **1 + 5 * 3**, la respuesta es 16 y no 18 porque el operador de **multiplicación** ("*") tiene **precedencia mayor** que el operador de **adición** ("+").



EXPRESIONES

Asociatividad de Operadores

Cuando los operadores tienen igual precedencia su asociatividad decide cómo se agrupan.

Por ejemplo "-" tiene asociatividad a izquierda, es decir que se resuelve primero el operador más a la izquierda, así $1 - 2 - 3$ se agrupa como $(1 - 2) - 3$ y se evalúa a -4.

Los operadores de igual precedencia que NO SON asociativos no pueden usarse unos junto a otros, por ejemplo, $1 < 2 > 1$ es ilegal en PHP. (**la expresión correcta es: $1 < 2 \text{ AND } 2 > 1$**)



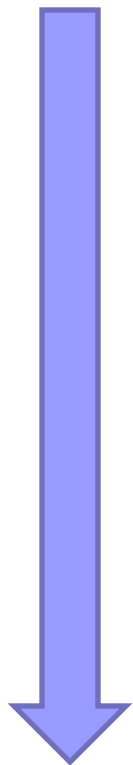
El **uso de paréntesis** aumenta la legibilidad del código haciendo grupos explícitamente en lugar de confiar en la precedencia y asociatividad implícitas del operador.



EXPRESIONES

Precedencia de Operadores

Mayor
Precedencia



Menor
Precedencia

Asociatividad	Operadores	Información adicional
no asociativo	<i>clone new</i>	<u>clone</u> and <u>new</u>
izquierda	<i>[</i>	<u>array()</u>
derecha	<i>**</i>	<u>aritmética</u>
derecha	<i>++ -- ~ (int) (float) (string) (array) (object) (bool) @</i>	<u>tipos</u> e <u>incremento/decremento</u>
no asociativo	<i>instanceof</i>	<u>tipos</u>
derecha	<i>!</i>	<u>lógico</u>
izquierda	<i>* / %</i>	<u>aritmética</u>
izquierda	<i>+ - .</i>	<u>aritmética</u> y <u>string</u>
izquierda	<i><< >></i>	<u>bit a bit</u>
no asociativo	<i>< <= > >=</i>	<u>comparación</u>
no asociativo	<i>== != === !== <></i>	<u>comparación</u>
izquierda	<i>&</i>	<u>bit a bit</u> y <u>referencias</u>
izquierda	<i>^</i>	<u>bit a bit</u>
izquierda	<i> </i>	<u>bit a bit</u>
izquierda	<i>&&</i>	<u>lógico</u>
izquierda	<i> </i>	<u>lógico</u>
izquierda	<i>? :</i>	<u>ternario</u>
derecha	<i>= += -= *= **= /= .= %= &= = ^= <<= >>= =></i>	<u>asignación</u>
izquierda	<i>and</i>	<u>lógico</u>
izquierda	<i>or</i>	<u>lógico</u>

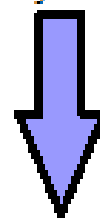


EXPRESIONES

Precedencia de Operadores

Asociatividad Operadores	
izquierda	* / %
izquierda	+ - .
no asociativo	< <= > >=
izquierda	or

Mayor Prec.



Menor Prec.

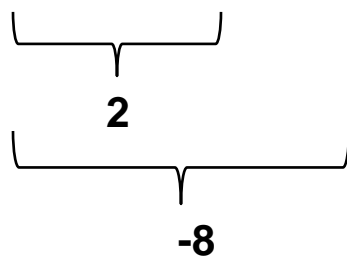
<?php

\$min = 4;

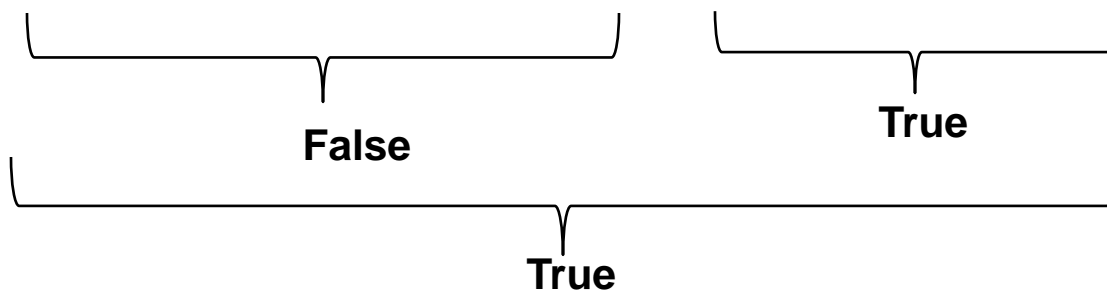
\$score = 2;

\$resultado = \$score < \$min / 2 - 10 or \$score < 90;

?>



a	b	a OR b
true	true	true
true	false	true
false	true	true
false	false	false





EXPRESIONES

Precedencia de Operadores

Asociatividad Operadores	
izquierda	* / %
izquierda	+ - .
no asociativo	< <= > >=
izquierda	or

Mayor Prec.



Menor Prec.

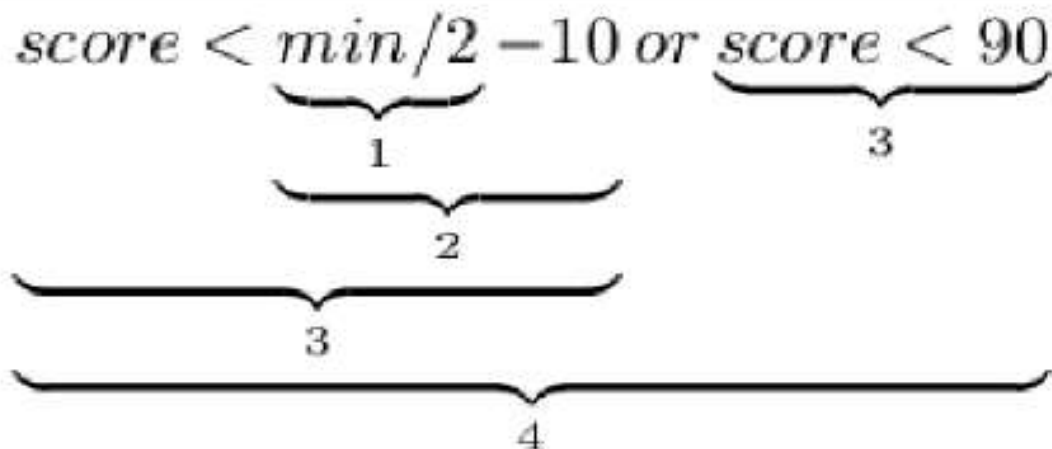
<?php

\$min = 4;

\$score = 2;

\$resultado = \$score < \$min / 2 - 10 or \$score < 90;

?>





EXPRESIONES

Precedencia de Operadores

<?php

\$min = 4;

\$score = 2;

\$resultado = (\$score < ((\$min / 2) - 10)) or (\$score < 90);

?>

