Universidad de Cantabria

Introducción al PHP

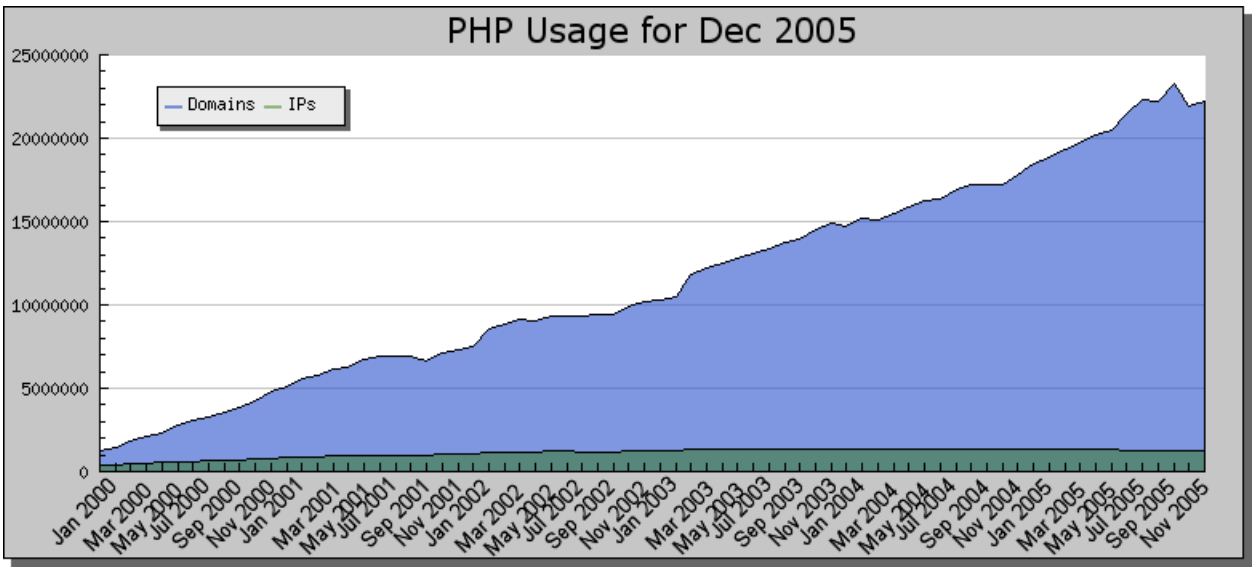
Ricardo Sáez
Marta Zorrilla

Febrero-2006

I- Aspectos Generales

Orígenes

- PHP (Personal Home Page), 1994, inventado por Rasmus Lerdof.
- Basado en “scripts” desarrollados en perl, luego reescritos en C.
- Versión actual PHP5 (sept 2004), presente en 20% de servidores web en internet. 6º lenguaje mas utilizado (C,Java, C++, VB, Perl)



I- Aspectos Generales
Características

- Lenguaje interpretado del lado Servidor. Los programas son ejecutados a través de un intérprete antes de transferir al cliente el resultado en forma de HTML puro.
- Programas embebidos directamente en el código HTML.
- Sigue corriente Open Source. Tanto intérprete como código fuente accesibles de forma gratuita en la red (http://www.php.net).
- Sintaxis muy similar a C, Perl o Java. Lenguaje fácil de aprender

I- Aspectos Generales
Características

- Lenguaje multiplataforma (trabaja sobre la mayoría de servidores Web y está preparado para interactuar con más de 20 tipos de bases de datos).
- Uso de mayúsculas, minúsculas indiferente (!!! Excepción: identificadores de variables)
- Formato libre. Las instrucciones deben terminar con “;”
- Comparado con otro tipo de tecnologías similares resulta mas rápido, independiente de la plataforma y más sencillo de aprender y utilizar.

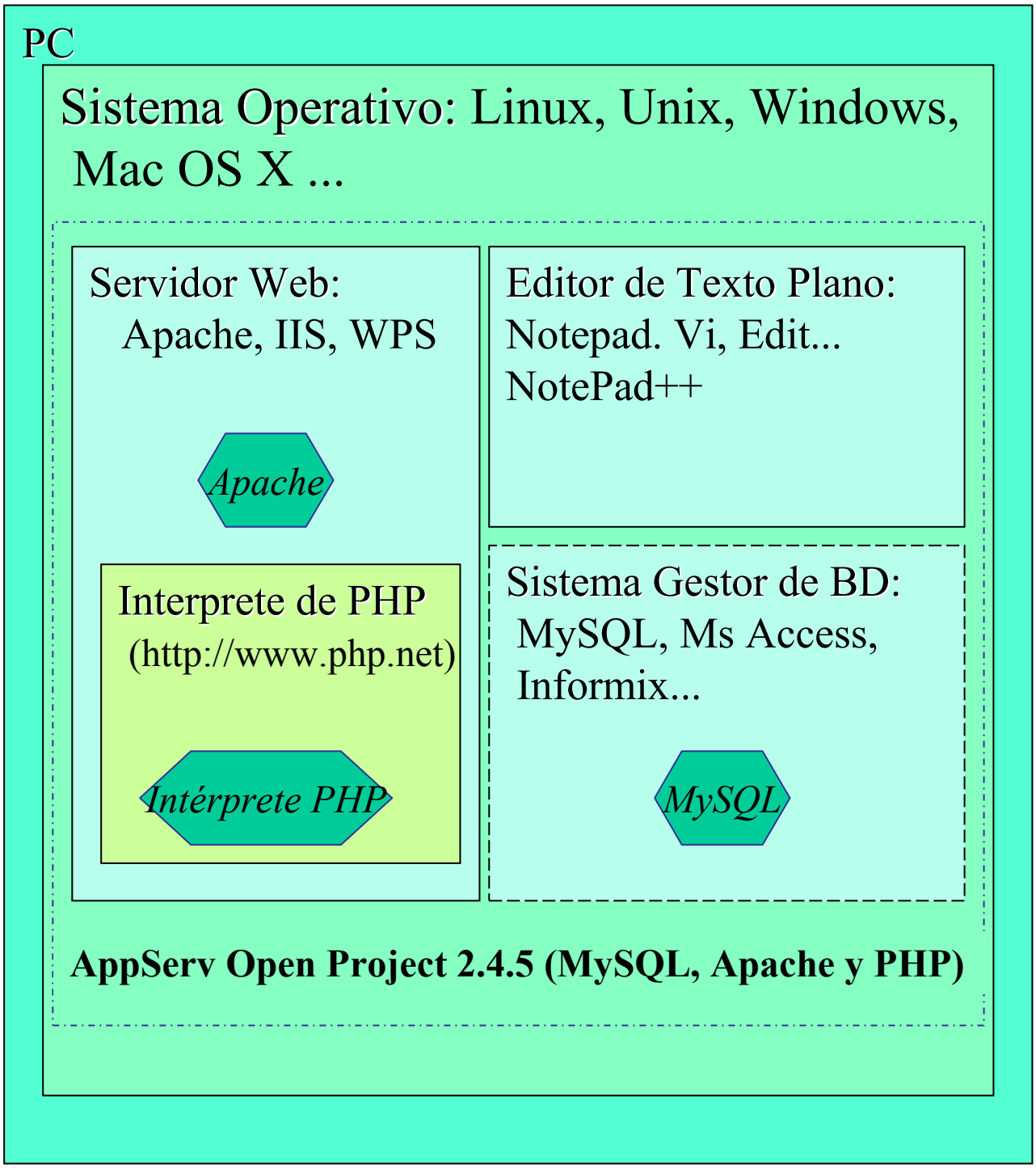
I- Aspectos Generales
Características

- Innumerable cantidad de funciones predefinidas

  - *Correo electrónico.*
  - *Administración gestión de base de datos.*
  - *Gestión de directorios y ficheros.*
  - *Tratamiento de imágenes.*
  - *Generación y lectura de cookies.*
  - *Generación de ficheros PDF...*

I- Aspectos Generales

Necesidades para programar en PHP



I- Aspectos Generales

Primer programa en PHP

```
< HTML>
<HEAD>
        <TITLE>Primer Programa</TITLE>
</HEAD>
<BODY>
        <?PHP
        echo "Mi primer programa";
        ?>
</BODY>
</HTML>
```

Programa1.php

I- Aspectos Generales

Cómo incrustar código en PHP

Forma más recomendable

```
<?PHP
.....
.....
.....
?>
```

- *Siempre disponible (no depende de configuración del Servidor.*
- *Específica de PHP.*
- *Única que permite incrustar código PHP en XML y XHTML*

Otras formas

```
<SCRIPT
LANGUAGE="php">
.....
.....
.....
.....
</SCRIPT>
```

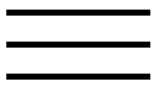
```
<?
.....
.....
.....
.....
?>
```

```
<%
.....
.....
.....
.....
%>
```

I- Aspectos Generales

Cómo incrustar código en PHP

```
<?php
  if ($x<0) {
     $y=1;
     echo "Número
negativo";
  }
  else {
     $y=2;
     echo "Número
positivo";
  }
?>
```



```
<?php
  if ($x<0) {
    $y=1;
?>
Número negativo
<?php
  }
  else {
    $y=2;
?>
Número positivo
<?php
  }
?>
```

Los scripts pueden ser divididos en bloques y escribir entre ellos HTML puro.

I- Aspectos Generales

Algunos ejemplo PHP

```
<?php
phpinfo();
?>
```

*La función **phpinfo()** para obtener información acerca de su sistema y configuración como las variables predefinidas disponibles*

```
<?php
echo
$_SERVER["HTTP_USER_AGENT"];
?>
```

Chequeo de navegador web usado

I- Aspectos Generales
Elementos básicos

***Identificador*:** Nombre utilizado para designar variables, etiquetas, funciones, y demás objetos del programa. Combina letras, dígitos y subrayado. Significativos los 6 ó 31 primeros.

***Comentario*:** Texto explicativo en el código fuente. Se enmarca entre secuencias /* ..*/, b) // , c) # ..... Son ignorados por el compilador.

I- Aspectos Generales
Elementos básicos

TIPOS DE DATOS **:** Naturaleza de los datos que maneja el programa

tipos escalares:

- boolean
- integer
- float (número de punto-flotante, también conocido como 'double')
- string

tipos compuestos:

- array
- object

tipos especiales:

- resource
- NULL

pseudo-tipos por razones de legibilidad:

- mixed
- number
- callback

I- Aspectos Generales
Elementos básicos

BOOLEANO

Un boolean expresa un valor de verdad. Puede ser **TRUE** or **FALSE**.

```
<?php
$foo = True; // asignar el valor TRUE a $foo
?>
```

Conversión a booleano: usar el molde (bool) o (boolean). Normalmente no necesario, el valor se convierte automáticamente si se requiere

Son considerados FALSE:
- el *boolean* FALSE mismo
- el *integer* 0 (cero)
- el *float* 0.0 (cero)
- el valor *string* vacío, y el *string* "0"
- un *array* con cero elementos
- un *object* con cero variables miembro
- el tipo especial NULL (incluye variables no definidas)

Cualquier otro valor es considerado TRUE

I- Aspectos Generales

Ejemplo

```
<?php
var_dump((bool) "");        // bool(false)
var_dump((bool) 1);         // bool(true)
var_dump((bool) -2);        // bool(true)
var_dump((bool) "foo");     // bool(true)
var_dump((bool) 2.3e5);     // bool(true)
var_dump((bool) array(12)); // bool(true)
var_dump((bool) array());   // bool(false)
var_dump((bool) "false");   // bool(true)
?>
```

I- Aspectos Generales
Elementos básicos

ENTEROS

Un integer es un número del conjunto Z = {..., -2, -1, 0, 1, 2, ...}.

```
<?php
$a = 1234; // numero decimal
$a = -123; // un numero negativo
$a = 0123; // numero octal (equivalente al 8
3 decimal)
$a = 0x1A; // numero hexadecimal (equival
ente al 26 decimal)
?>
```

No hay un operador de división de enteros en PHP. Puede moldear el valor a un entero para asegurarse de redondearlo hacia abajo, o usar la función round().

```
<?php
var_dump(25/7); // float(3.5714285714286)
var_dump((int) (25/7)); // int(3)
var_dump(round(25/7));  // float(4)
?>
```

I- Aspectos Generales
Elementos básicos

Conversión a integer: usar el molde *(int)* o *(integer).* Normalmente no necesario, el valor se convierte automáticamente si se requiere. También se puede usar funcion *intval()*

- desde *boolean* FALSE es 0, TRUE es 1
- ver manual..

I- Aspectos Generales
Elementos básicos

COMA FLOTANTE

también conocidos como "flotantes", "dobles" o "números reales")

Formalmente:

LNUM [0-9]+

DNUM ([0-9]*[\.]{LNUM}) | ({LNUM}[\.][0-9]*)

EXPONENT_DNUM (({LNUM} | {DNUM}) [eE][+]? { LNUM})

```
<?php
$a = 1.234;
$b = 1.2e3;
$c = 7E-10;
?>
```

Conversión a flotante. Ver sección en documentación complementaria

I- Aspectos Generales
Elementos básicos

CADENA

Un valor **string** es una serie de caracteres. En PHP, un caracter es lo mismo que un byte, es decir, hay exactamente 256 tipos de caracteres diferentes.

Un literal de cadena puede especificarse en tres formas diferentes.

- comillas simples
- comillas dobles
- sintaxis heredoc

```
<?php
echo 'esta es una cadena simple';
echo "Este cuadro tiene $cuadro metros de
 ancho.";
$cadena = <<<FIN
?>
```

I- Aspectos Generales
Elementos básicos

Tipo de dato	***Ejemplos de valores***
entero decimal	1 123 2100 -12 90
entero hexadecimal	0x80 /* 128 en dec.*/ 0x21B /* 539 en dec.l */
entero octal	012; /* 10 en decimal */
coma flotante	123.23, -76.00001, 1.4e3
cadena	“programa en PHP” ‘programacion OO’

```
decimal : [1-9][0-9]* | 0
hexadecimal : 0[xX][0-9a-fA-F]+
octal : 0[0-7]+ integer : [+-
]?decimal | [+-]?hexadecimal | [+-
]?octal
```

I- Aspectos Generales
Elementos básicos

***Variables*: Posición de memoria con nombre.**

- Mantiene valores que pueden variar en el programa.
- No necesitan declararse.
- No tienen tipo predefinido, puede asignarse cualquier valor.
- Nombres de variables comienzan con "$"
- En nombres de variables si se distingue mayúsculas-minúsculas.
- Se forman por combinación de letras, dígitos y subrayado. Válidas letras acentuadas y "ñ"
- Una variable puede reutilizarse asignándose datos, incluso de tipos distintos.

I- Aspectos Generales
Elementos básicos

- Se pueden reutilizar en el tiempo, incluso dándoles valores de diferentes tipos.
- Además de variables definidas por el programador, existen variables predefinidas a las que se tiene acceso desde los scripts.
- Puede haber nombre de variables que sean a su vez variables

```
<?PHP
$variable='x';
....
$$variable=150;          //equivale a
$x=150
?>
```

I- Aspectos Generales
Elementos básicos

Ejemplos

Variables válidas:

$x, $día_de_mañana, $variable2

Variables no válidas:

x, $día-de-mañana, $2variable, $dto%

Asignación de variables:

```
<?PHP
$x=5;
$y=3;
echo “x=$x<BR>y=$y”;
$x=5*$x*$y;
echo “x=$x<BR>y=$y”;
?>
```

I- Aspectos Generales
Elementos básicos

Funciones para conocer el tipo de dato que guarda una variable:

- *gettype()* *devuelve el tipo de la variable.*
- *is_array()* *determina si contiene un array.*
- *is_float()* *determina si contiene un núm. en coma flotante.*
- *is_int()* *determina si contiene un número entero.*
- *is_object()* *determina si hace referencia a un objeto.*
- *is_string()* *determina si contiene una cadena de caracteres.*

I- Aspectos Generales

Ejemplo "is_array()"

```
<?php
$si = array('esto', 'es', 'una matriz');

echo is_array($si) ? 'Matriz<br>': 'no es Ma
triz';
echo "\n";

$no = 'esto es una cadena';

echo is_array($no) ? 'Matriz' : 'no es Matriz'
;
?>
```

Matriz

no es Matriz

I- Aspectos Generales
Elementos básicos

Otras funciones de interés:

- *isset()*

Permite conocer si una variable ha sido definida. Importante para saber si se han recibido todos los datos de un formulario.

- *unset()*

Permite eliminar una variable. Borra el contenido de la misma y libera el espacio en memoria que ocupaba.

- *empty()*

Permite saber si una variable que existe tiene un valor nulo o vacío

I- Aspectos Generales

Ejemplo "isset(), unset()"

```
<?php

$var = '';
if (isset($var)) {
   echo "Esta variable esta definida, asi
que se imprimira esto.";
}
$a = "prueba";
$b = "otraprueba";
var_dump(isset($a));      // TRUE
var_dump(isset($a, $b));  // TRUE
unset ($a);
var_dump(isset($a));     // FALSE
var_dump(isset($a, $b)); // FALSE
$foo = NULL;
var_dump(isset($foo));   // FALSE
?>
```

I- Aspectos Generales
Elementos básicos

- La primera vez que se usa una variable y se le da un valor, automáticamente PHP le asigna un tipo a dicha variable.
- Si el valor asignado cambia de tipo, implícitamente se produce una conversión de tipo en la variable.
- El programador puede forzar la conversión de una variable de dos formas

✓ *Mediante el uso de la función* ***settype()***
$b=settype($a, "string");

✓ *Mediante el operador de conversión*
$b=(integer)$a;
también llammado cast o molde

I- Aspectos Generales
Elementos básicos

- 3 localizaciones:
  - fuera de todas las funciones, *en el script del documento (globales)*
  - en cuerpo de las funciones *(locales)*
  - en definición de parámetros *(parámetros formales)*

- *Ambito de una variable*; en que lugares una variable es conocida y puede ser usada

  (*) Por defecto tienen ámbito global

Variables globales:

- Se conocen en todo el programa ó documento
- Se declaran fuera de las funciones
- Existen en todo el documento, incluso si hay varios scripts.
- Pueden ser referenciadas dentro de una función solo si se declaran como “global” dentro de la funcion.

I- Aspectos Generales
Elementos básicos

Ejemplo1: Uso de variables globales dentro de una función

```
<?php
$iva=1.17;
function importe_total ($bruto) {
    global $iva;
    $i=$bruto*$iva;
    return $i;
}
?>
```

Ejemplo2

```
<?php
$a = 1; /* area global */
function Test()
{
    echo $a; /* referencia a local variable */
}
Test();
?>
```

I- Aspectos Generales
Elementos básicos

Variables locales:

- Se declaran dentro de la función
- Sólo pueden ser referenciadas dentro de la función o bloque donde fueron declaradas
- Sólo existen mientras se ejecuta el bloque donde están declaradas.
- el uso de variables locales ayuda a prevenir fenómenos no deseados
- las variables locales no pueden retener sus valores entre llamadas a la función

Parámetros formales:

- Si una función utiliza argumentos, debe declarar las variables que van a aceptar los valores de los argumentos (parámetros formales). Se comportan como var. locales

I- Aspectos Generales
Elementos básicos

Ejemplo-1: uso de variables locales y parametros

```
.....
function importe_final ($precio)  {
$x=$precio*1.17;
return $x;
}
function descuento ($importe)  {
$x=$importe;
if ($importe > 20000)
    $x=$importe-$importe*0.25;
return $x;
}
.....
```

Ejemplo-2: uso de parametros

```
function cuadrado ($valor)  {
return ($x*$x);
}
```

I- Aspectos Generales
Elementos básicos

Constantes: Valores fijos no alterables por el programa. Son los valores que asignamos a las variables.

- Solo pueden definirse constantes con valores escalares (números o cadenas)
- La definición de constantes se realiza con la función *define()*
- Los identificadores de constantes no comienzan con "$" de las variables
- Por convenio se usan mayúsculas para los identificadores de constantes.
- Las constantes no tienen restricción de ámbito.

I- Aspectos Generales
Elementos básicos

Ejemplo-1: uso de constantes

```
define("NOMBRE_EJERCICIO", "Ej-1");
define ("DESCUENTO", 0.25);
define("TIPO_IVA",1.17);
define ("IMPORTE_MAXIMO", 20000);

echo NOMBRE EJERCICIO;

$importe=$importe*(1+TIPO_IVA);
if ($importe >IMPORTE_MAXIMO) {
    $importe =$importe*(1-DESCUENTO);
}
echo $importe;
.....
```

I- Aspectos Generales
Elementos básicos

Secuencias de Escape: permiten representar caracteres no imprimibles (retorno de carro, tabulación, etc), y confiere portabilidad al código.

- \n salto de línea
- \r retorno de carro
- \" comilla doble
- \' comilla simple
- \0 nulo
- \t tabulación horiz.
- \\ barra invertida
- \N carácter cuyo codigo ASCII en octal es N
- \xN carácter cuyo codigo ASCII en hexadecimal es N

Ejemplo:

```
echo "El radio=$radio\tArea=$area\nFin";
```

El radio=1 Area=3.14
Fin

I- Aspectos Generales
Elementos básicos

Cadenas de caracteres:

- Las cadenas son arrays de caracteres por lo que se puede acceder a los diferentes caracteres que la forman de manera individual mediante su índice.
  cadena[1] → accede al segundo carácter guardado en la variable cadena

- Para su definición se pueden utilizar como delimitadores las comillas simples o las dobles pero no ambas a la vez.

- La diferencia entre el uso de comillas dobles y simples como delimitadores de cadenas es, además de las secuencias de escape que acepta, que **las variables dentro de una cadena con comillas dobles se expanden** (se sustituyen por su valor).

I- Aspectos Generales
Elementos básicos

Cadenas de caracteres y variables:

- Existe otra forma de expandir variables dentro de cadenas que PHP hereda de Perl
  <<<NOMBRE_IDENTIFICADOR
  .....
  NOMBRE_IDENTIFICADOR

```
<?PHP $edad=54;
echo “mi edad es $edad<BR>”;
echo ‘mi edad es ‘ . $edad .
‘<BR>’;
echo <<<TEXTO
   mi edad es $edad<BR>
TEXTO; ?>
```

I- Aspectos Generales
Elementos básicos

***Operadores*:**

En PHP hay 5 clases

- aritméticos
- relacionales
- lógicos
- a nivel de bits
- especiales

I- Aspectos Generales
Elementos básicos

aritméticos:

pueden aplicarse a casi todos los tipos de datos numéricos.

operadores

-	resta, también monario
+	suma
*	multiplicación
/	división (si operandos enteros, división entera)
%	división en módulo (resto división entera)
--	decremento (precede/sigue a operando)
++	incremento (precede/sigue a operando)

Ejemplo1:

```
$x=10;
$y=++$x;    /* x=11, y=11 */
```

Ejemplo2:

```
x=10;
$y=$x++;
 /*x=11, y=10 */
```

Ejemplo3:

```
$x=10; $y=2*--$x; $z= ++$x-$y--;
/* x=10, y=17, z=-8 */
```

I- Aspectos Generales
Elementos básicos

relacionales :

pueden aplicarse sobre los datos predefinidos y dan como resultado un valor lógico.

operadores

> mayor que

>= mayor o igual que

< menor que

<= menor o igual que

== igual que

=== igual que + son del mismo tipo

!= distinto de

!== distinto que + son de distinto tipo

I- Aspectos Generales
Elementos básicos

Ejemplo-1:

```
<?PHP
    $x=10;
    $y=30;
    if ($x==$y)          /* ← → $x ==$y */
        echo “los dos valores son iguales”;
    else
        echo “los dos valores son diferentes”;
?>
```

Resultado-1: los dos valores son diferentes

Ejemplo-2:

```
<?PHP
$x=1000;
$y="1000";
        if ($x==$y)
                echo "los dos valores son
iguales";
        else
                echo "los dos valores son
diferentes";
?>
```

I- Aspectos Generales
Elementos básicos

lógicos:

se aplican sobre operandos lógicos y dan como resultado un valor lógico.

operadores

&& Y

$x&&$y →cierto si tanto $x como $y son cierto

and Y

$xand$y →cierto si tanto $x como $y son cierto

|| O

$x || $y → cierto si $x ó $y son cierto

or O

$x or $y → cierto si $x ó $y son cierto

xor O exclusivo

$x xor $y →cierto si $x ó $y cierto, no ambos

! negación

!$x →cierto si $x tiene valor falso

I- Aspectos Generales
Elementos básicos

a nivel de bits

válidos para tipo entero.

operadores

&	AND	$x & $y
|	OR	$x | $y
^	O exclusiva	&x ^ &
-	complemento a 1	-$x
>>	desplazamiento a derecha	
<<	desplazamiento a izquierda	

nota:

variable >> número posiciones en bits
/* dividir */

variable << número posiciones en bits
/*multiplicar */

I- Aspectos Generales
Elementos básicos

Ejemplo-1:

```
<?PHP
$x=12.5;
$y=20.4;
    echo ($x&&$y);          /*1 */
    echo"<br>";
    echo ($x||$y);          /*1 */
?>
```

Ejemplo-2:

```
<?PHP
$x=10;
$y=20;
        echo ($x&$y);          /*  0  */
        echo"<br>";
        echo ($x|$y);          /*  30  */
?>
Resultado:
```

I- Aspectos Generales
Elementos básicos

otros operadores:

- *operador ?* (condicional) se utiliza con sentencias condicionales

  formato

  exp1?exp2:exp3;

  evalúa “exp1”, si cierto, se evalúa “exp2” y toma ese valor para la expresión”. Si “exp1” es falsa, evalúa “exp3” tomando ese valor para la expresión.

  Ejemplo

  $par= ($n %2 ==0) ? 1:0;

- *operador .* (concatenación) permite concatenar cadenas

  formato

  cadena1.cadena2

  Ejemplo

  $titulo1= “Lenguaje”;

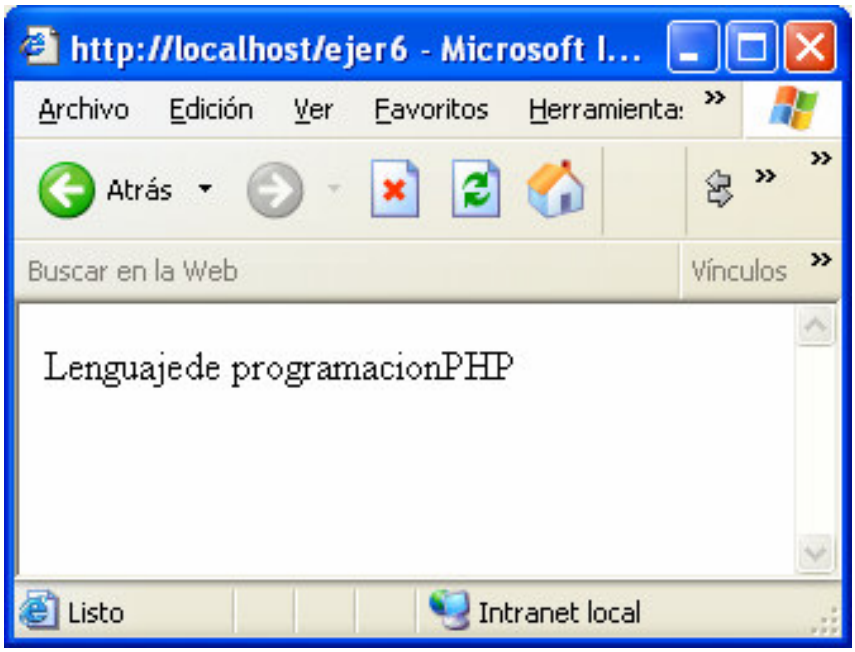
  $titulo2=“PHP”;

  $titulo=$titulo1.” de programacion”.$titulo2;

I- Aspectos Generales
Elementos básicos

Ejemplo:

```
<?PHP
$titulo1= "Lenguaje";
$titulo2="PHP";
$titulo=$titulo1."de programacion".$titulo2;
echo $titulo;
?>
```



I- Aspectos Generales
Elementos básicos

- *operador "@"* (supresión de error)

  el interprete PHP no genera mensajes de error aunque detecte situaciones erróneas

Ejemplo

```
$y= @(1/0);
```

- *operador "`"* (ejecución)

  la expresión encerrada entre acentos debe ser tratada como una sentencia a ejecutar directamente por el sistema operativo

Ejemplo

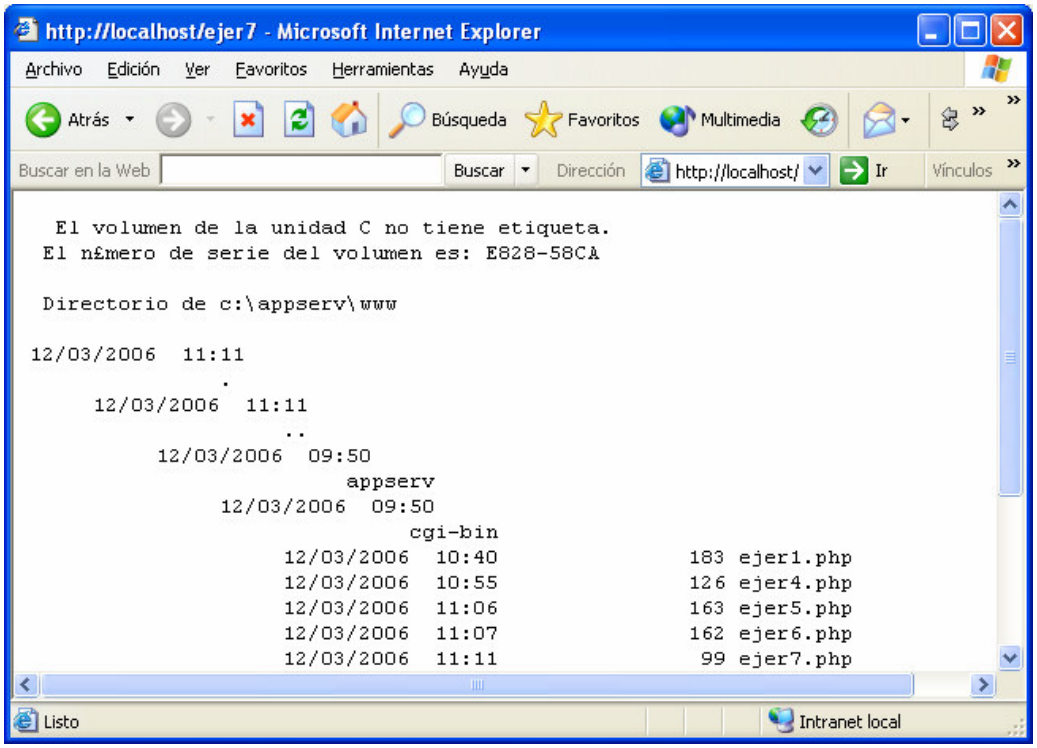
```
<?php
$x=`dir`;
echo "<pre> $x </pre>";
?>
```

/* genera una pagina web con el contenido del directorio donde está */

I- Aspectos Generales
Elementos básicos

Ejemplo:

```
<?php
        $x=`dir`;
        echo "<pre> $x </pre>";
?>
```



I- Aspectos Generales
Elementos básicos

- *operador "cast"* (conversión de tipo)

  formato

  (tipo) expresion;

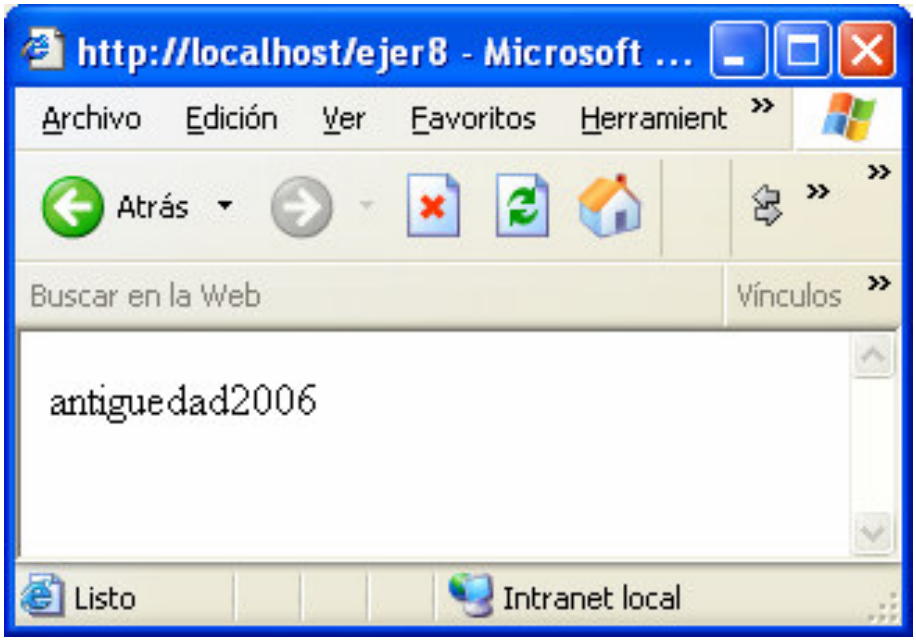
Ejemplo

```
$x=(int) 15.65;
$p= (string) 2006;
```

I- Aspectos Generales
Elementos básicos

Ejemplo:

```
<?php
        $p=(string)2006;
        print("antiguedad".$p);
?>
```



I- Aspectos Generales
Elementos básicos

operadores paréntesis y corchetes

los primeros cambian la precedencia natural de los operadores, los segundos llevan a cabo la indexación de arrays (proporcionan el índice del array)

***Resumen de precedencias*:**

Mayor

() [-> .	I-D
++ --(cast), @	D-I
* / %	I-D (aritméticos)
+ -	I-D (aritméticos)
. (concatenacion)	I-D
<< >>	I-D (relacionales)
<< = >> =	I-D (relacionales)
==, ! =, ===	I-D (relacionales)
&, ^, |,&&,||	I-D (lógicos)
condicional	I-D
asignacion	
and, or, xor	I-D (lógicos)

Menor

I- Aspectos Generales
Elementos básicos

***Expresiones*:** Es cualquier combinación válida de operadores, variables y constantes.

II- *PHP y los Formularios HTML*

Formularios HTML:

- Forma principal de interactuar con el usuario.
- *Cada dato que el usuario introduzca en el correspondiente campo del mismo*
- *El parámetro NAME de los campos de un formulario sirven para poder identificar cada campo dentro del formulario*

Envío de datos a programas PHP

1. *Indicar en el **parámetro ACTION**, de la etiqueta FORM, el nombre del documento PHP al que deben ser enviados los datos.*

2. *El documento PHP recibirá variables con los datos que el usuario introdujo en los campos del formulario. El nombre de las variables coincidirá con el nombre de los campos del formulario.*

II- *PHP y los Formularios HTML*

Método de acceso a las variables del formulario

Primer método (válido desde php 4.1):

$_REQUEST['nombre_control'],
$_POST[nombre_control],
$_GET[nombre_control].

Segundo método o estilo corto:

- *Necesidad de configurar en archivo **php.ini** del intérprete de php el siguiente parámetro : register_globals=on.*

- *$nombre_control.*

Tercer método (válido independientemente del servidor):

$HTTP_POST_VARS['nombre_variable'],
$HTTP_GET_VARS['nombre_variable'].

I- Aspectos Generales
Elementos básicos

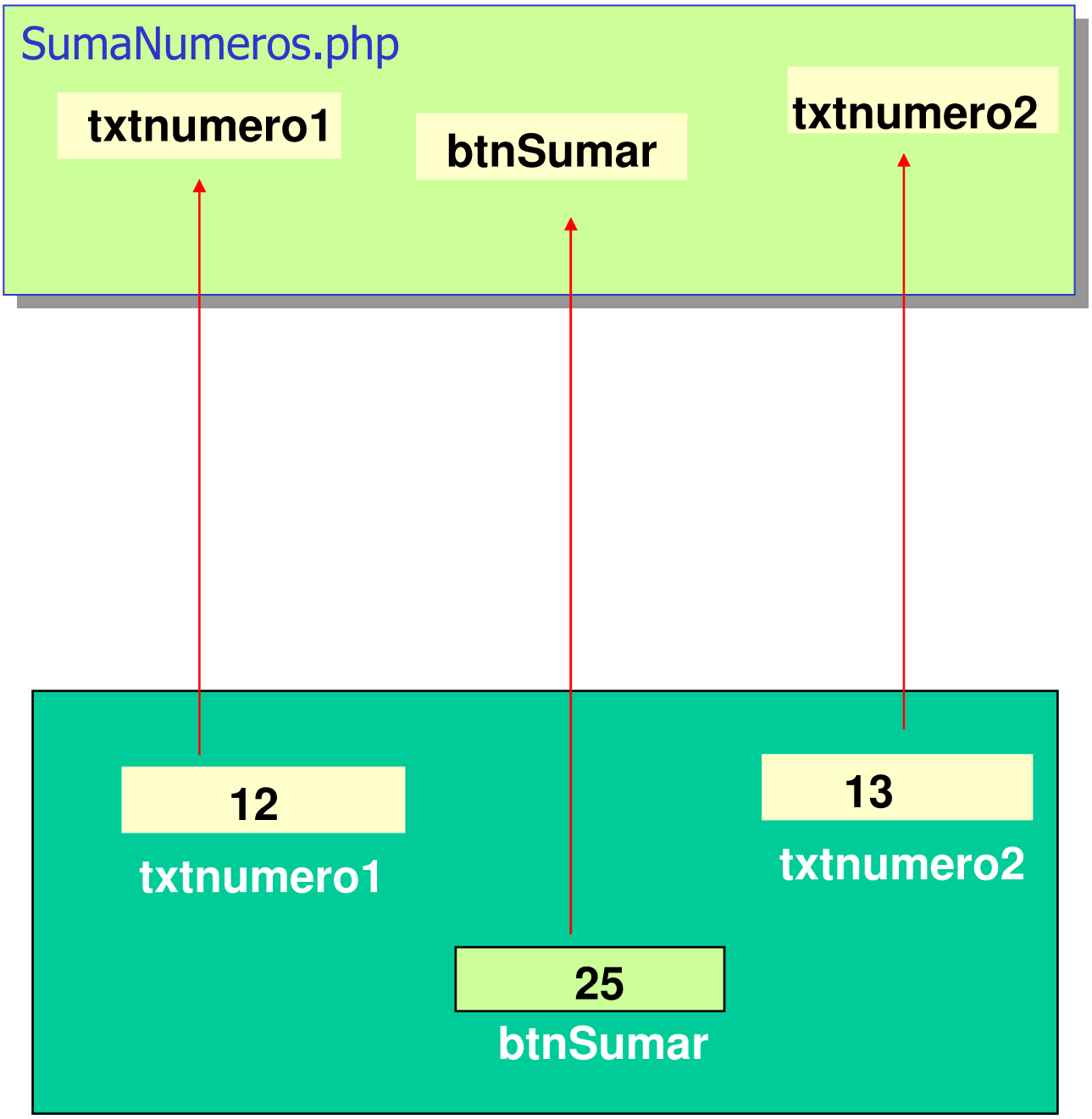
Ejemplo de Tratamiento de formularios con listas de selección multiples

Ejemplo-1: SumaNumeros (SumaNumeros.php)

Ejemplo-2: Procesar par/impar (procesar.php)

II- *PHP y los Formularios HTML*

Tratamiento de formularios con listas de selección múltiples



I- Aspectos Generales
Elementos básicos

Ejemplo:

```
<html>
<head>
      <title>procesar un numero</title>
</head>
<body>
<?php
if (isset($_REQUEST['btnAceptar'])) {
$num1=$_REQUEST['txtNumero'];
              if (!is_numeric($num1) ) {
                 echo is_int($num1);
        print ("Valor introducido no es numerico");
                              }
     elseif ($num1%2 == 0)
        print ("El número es par");
     else
        print ("El número es impar");  }
?>
<form action="procesar.php" method="post">

       Introduzca un numero entero:
       <INPUT type="text" name="txtNumero" size=4>
      <br>
      <INPUT type="submit" name="btnAceptar"
value="Aceptar" size=4>
</form>
</body> </html>
```

III- Sentencias de Control
introducción

- Son aquellas que permiten alterar el orden secuencial habitual en la ejecución de sentencias.
  - condicionales (if, switch)
  - iterativas (while, for, do-while, foreach)
  - de salto (break, continue, goto, return)
  - de etiquetado (case, default, label)
  - de expresión (expresiones válidas)
  - de bloque ({})
- En php cualquier valor distinto de 0 se considera “true”, 0 se considera “falso”.
- Sentencia. Puede ser simple, bloque o ninguna.

III- Sentencias de Control *condicionales*

if

forma general

if (expresión) sentencia;
else sentencia;

Descripción

si expresión es cierta (cualquier valor distinto de 0), se ejecuta la/s sentencia/s objeto del if, sino las del else

if anidado

Es una variante del if consistente en una sentencia if objeto de otro if o else.

En PHP, al igual que en C, el “else” está asociado al if más próximo que no tenga sentencia else asociada

III- Sentencias de Control *condicionales*

escala if-else-if

Es una variante del if consistente en una sentencia if objeto de otro if o else.

Si todas las condiciones fallan se ejecuta la sentencia “else” final (o nada si no hay)

Formato

```
if (expresión1) sentencia;
  else
      if (expresion2) sentencia;
      else
          if (expresión3) sentencia;
.
.
                  else sentencia;
```

III- Sentencias de Control *condicionales*

- Si en lugar de tratar una sentencia simple en una cláusula “if”, es preciso tratar un conjunto de ellas se empleará el bloque

```
{
  sentencia1;
  ...
  sentencian;
  }
```

- Sintaxis alternativa: PHP admite sintaxis alternativa para delimitar el bloque (útil para fraccionar una estructura en dos scripts)

```
if (condicion) :
        sentencia1;
        ..
        sentencian;
  endif;
```

III- Sentencias de Control *condicionales*

```
if (condición1) :
    sentencia(s) a ejecutar si la condición 1 se cumple
elseif (condición2) :
    sentencia(s) a ejecutar si la condición 2 se cumple
...
else :
    sentencia(s) a ejecutar si ninguna de las condiciones
    anteriores se cumplen
endif;
```

III- Sentencias de Control *condicionales*

switch sentencia de selección multiple

forma general

```
switch (expresión) {
    case constante1:
       secuencia de sentencias
       break;
    case constante2:
       secuencia de sentencias
       break;
    ....
    default:
       secuencia de sentencias;
    }
```

Descripción

Se comprueba el valor de expresión y se compara por orden con las constantes especificadas en las claúsulas. Cuando se encuentra correspondencia se ejecuta la secuencia de sentencias asociadas a ese case hasta el break o hasta el final de switch. Si no hay correspondencia se ejecuta las correspondientes a default.

III- Sentencias de Control *condicionales*

- La sentencia switch se diferencia de la if en que sólo puede comprobar la igualdad
- No puede haber dos constantes case en el mismo switch que tengan igual valor.
- Si se utilizan constantes carácter en las claúsulas switch, se transforman a sus valores enteros
- Puede haber case con sentencias vacías
- Si no hay break, la ejecución de un case continua en el siguiente, hasta que se encuentre o termine el switch
- La sentencia switch dispone de una sintaxis alternativa *(: ....end switch)*

III- Sentencias de Control
condicionales

```
switch (expresión) {
    case valor1:
        sentencia(s) a ejecutar si se cumple
la condición
        break;
    case valor2a:
    case valor2b:
        sentencia(s) a ejecutar si se cumple
la condición
        break;
    ...
    default:
        sentencia(s) a ejecutar si ninguna de
las condiciones
        anteriores se cumplen  }
```

III- Sentencias de Control *iterativas*

Permiten que un conjunto de instrucciones sea ejecutado hasta que se alcance una condición (predefinida como en el *for*, o sin final determinado como en *while* o *do-while*)+ foreach

for

forma general

for (inicialización; condición; incremento)
    sentencia;

Descripción

inicialización-> sentencia de asignación para iniciar variable de control de ciclo.

condición -> expresión relacional.

incremento -> determina como cambia la variable de control.

III- Sentencias de Control *iterativas*

while

forma general

while (condicion) sentencia;

Descripción

se evalúa condición, si es cierta se ejecuta sentencia. Si condición es falsa se sale del bucle

do-while

analiza la condición a final del ciclo.

forma general

```
do {
    sentencia;
} while (condicion);
```

Descripción

el bucle itera hasta que la condición se hace falsa.

III- Sentencias de Control *iterativas*

foreach

forma general

```
foreach (nombre_array as nombre_variable) {
    sentencia(s) a ejecutar para cada elemento del array
}
```

Sentencia incorporada en la versión php 4.0.

(*) Permite recorrer todos los elementos de un array de una forma muy simple.

En la primera iteración la variable contendrá el valor del primer elemento del array nombre_array en las siguientes iteraciones los sucesivos valores hasta recorrer todos los elementos del array

III- Sentencias de Control *iterativas*

```
while (expresión) {
    sentencia(s) a ejecutar si se cumple la condición
}
```

```
while (expresión) :
    sentencia(s) a ejecutar si se cumple la condición
endwhile;
```

```
for (inicialización; condición; modificación) {
    sentencia(s) a ejecutar si se cumple la condición
}
```

```
for (inicialización; condición; modificación) :
    sentencia(s) a ejecutar si se cumple la condición
endfor;
```

```
do {
    sentencia(s) a ejecutar si se cumple la condición
}
while (expresión);
```

III- Sentencias de Control salto

Sentencia **BREAK y CONTINUE**

• Permiten alterar la ejecución prevista de una estructura iterativa

• **BREAK**:
- Cancela completamente la ejecución del bucle.
- Es utilizada, además, en la sentencia SWITCH.

• **CONTINUE**:
- Cancela una de las iteraciones pasando directamente a la siguiente.

Dentro de un bucle, ambas sentencias admite el uso de un parámetro opcional que indica: Número de estructuras de control de las que hay que salir o Número de niveles a saltar para continuar la ejecución (break n; continue n;).

III- Sentencias de Control otras

de expresión

cualquier expresión válida C seguida de punto y coma

Ejemplo:

```
func();
a=b+c;
b+f();
;
```

de bloque

grupo de sentencias relacionadas que se tratan como una unidad

Los bloques comienzan con “{“ y concluyen con “}”

Se pueden colocar en cualquier punto de programa

IV- Funciones
Introducción

Es la base de toda actividad de programa

forma general
function NombreFuncion (argumentos) {
sentencias;
return ValorDevuelto;
}

Ejemplo-3.1 /* calcula base^n */

```
<?php
function potencia ($base, $n) {
$p=1;
for ($i=1;$i<=$n; $i++)
            $p=$p*$base;
return $p;
}
$numero=50;
$exponente=5;
echo potencia($numero,exponente);
?>
```

IV- Funciones
llamada

Llamada a las funciones:

- La función debe de estar definida antes de realizar la llamada (en PHP 4 y 5, no)
- En la llamada, los argumentos deben pasarse en el orden en que aparecen en la definición de parámetros.
- Si se pasan menos argumentos que parámetros definidos, los últimos se asumirán nulos
- Si la función devuelve un valor, este puede asignarse a yuan variable o ser usado en otra expresión.

Ejemplo-3.2 con la función anterior, las siguientes llamadas son validas

```
$x=2;
$y=3;
$z=potencia($x,$y);
$z=potencia(2,3);
$x=2;
Echo POTENCIA($x,3);
$z=suma(“2”,”3”);
```

IV- Funciones
Argumentos

***argumentos*:** es una de las formas en que dos funciones intercambian información.

argumento: valor usado en una llamada a función (también parámetro actual)

parámetro: nombre de las variables locales que aparecen entre paréntesis en la declaración de la función (también parámetro formal). En PHP todos los argumentos pueden ser considerados opcionales

IV- Funciones
Argumentos

Ejemplo-3.3 /* Potencias */

```
<HTML>
<HEAD>
<TITLE> Ejemplo de potencias </TITLE>
</HEAD>
<BODY>
<? PHP
function Potencia ($base, $exp) {
$p=1;
for ($i=1;$i<=$exp;$i++)
    $p=$p*$base;
return $p;
 }
$b1=2; $b2=-2;
for ($i=0;$i<10; $i++) {
  echo potencia($b1,$i);
  echo potencia($b2,$i);
  echo potencia(potencia($b1,$b2),$i);  }
?>
</BODY>
</HTML>
```

IV- Funciones
Argumentos

- los parámetros formales y actuales deben ser del mismo tipo.

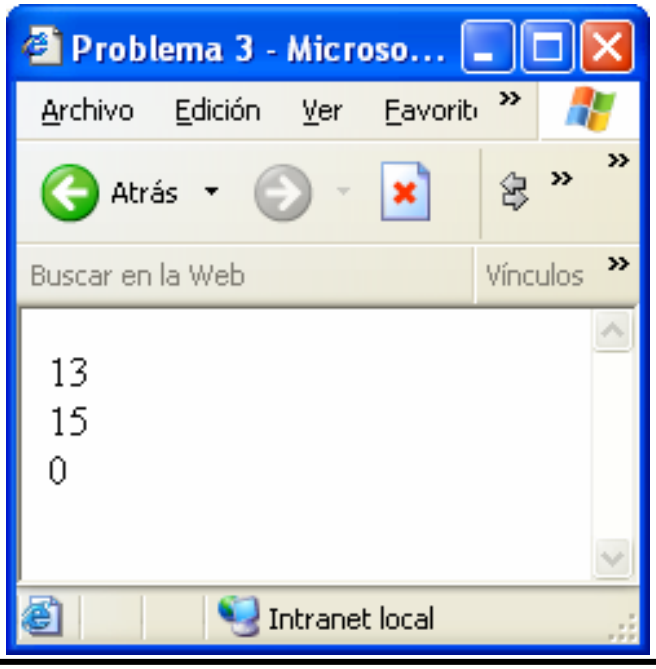
- al definir funciones pueden darse valores por defecto a los argumentos: si no está presente en la llamada se le asigna ese valor defecto (asignar parámetro a valor)

- listas de argumentos de longitud variable: permitido a partir de PHP4 (*ejemplo 3.4*)

IV- Funciones
Argumentos

Ejemplo-3.4 /* Listas de argumentos de longitud variable */

```
<? PHP
function suma () {
$n=func_num_args();
$aux=0;
for ($i=0;$i<$n;$i++)
    $aux += func_get_arg($i);
return aux;
}
print suma (2,5,6). “<BR>”;
print suma (1,5,7,2). “<BR>”;
print suma ().”<BR>”;
?>
```



IV- Funciones
Paso de parámetros

***Paso de parámetros*:** definen la forma en que pasan argumentos a las funciones.

- Llamada por valor: se copia el valor del argumento en el parámetro formal de la función. Los cambios en los parámetros de la función *no afectan* a las variables que se usan en la llamada.

- Llamada por referencia: se copia la dirección del argumento en el parámetro formal de la función. Los cambios en los parámetros de la función *afectan* a las variables que se usan en la llamada.

IV- Funciones
Paso de parámetros

- Para indicar que un argumento se pasa por referencia, se debe anteponer “&” al nombre de la variable en la lista de parámetros.

Ejemplo-3.5: /* ppv-cuadrado de un valor dado */

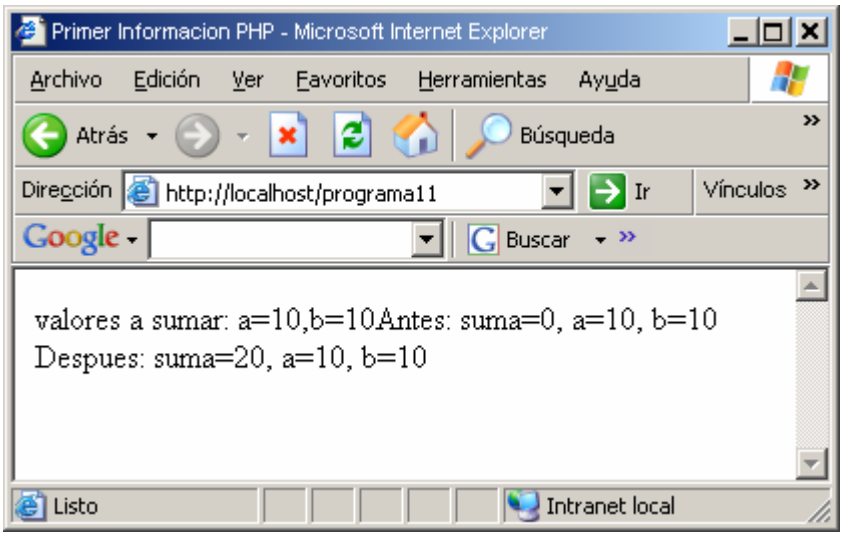
```
function  cuadrado ($x) {
$x=$x*$x;
return $x;
}
.....
$t=10;
print (“$t,cuadrado($t)”);
....
```

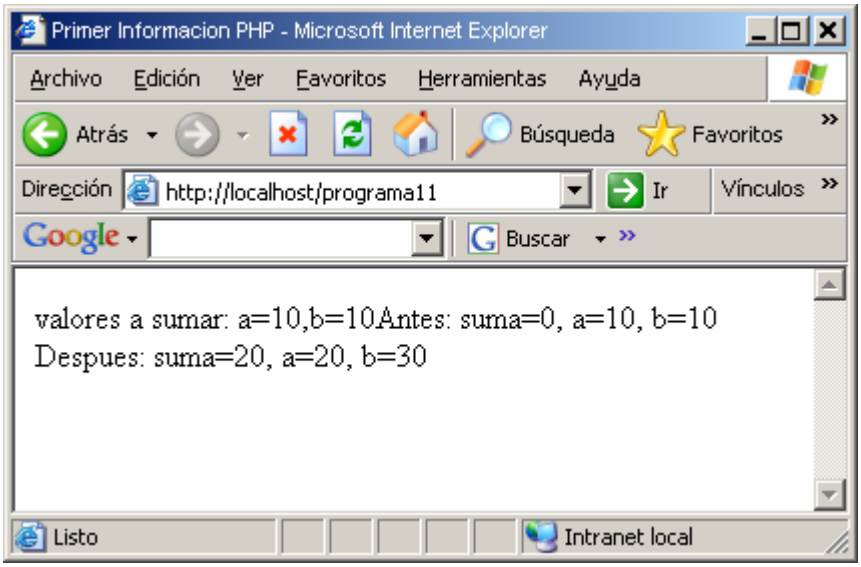
IV- Funciones
Paso de parámetros

Ejemplo-3.6: /* suma de dos números */

```
<? PHP
Opción-1:     function suma($x,$y)  {
Opción-2:     function suma(&$x,&$y)  {
$c=$x+$y;
$x+=10;
$y*=3;
return ( $c);
}
?>
<HTML>
<HEAD>
<TITLE> Ejemplo de SUMAS </TITLE>
</HEAD>
<BODY>
<? PHP
$a=10; $b=10; $c=0;
echo “valores a sumar: a=$a,b=$b”);
print (“Antes:
          suma=$c, a=$a, b=$b <BR>”);
$c=suma($a,$b);
print (“Despues:
          suma=$c, a=$a, b=$b <BR>”);
?>
</BODY>
</HTML>
```

IV- Funciones
Paso de parámetros





IV- Funciones
Paso de parámetros

Ejemplo-3.7 intercambio de dos valores

```
<? PHP
function  intercambio (&$x, &$y)  {
$temp=$x;
$x=$y;
$y=$temp;
}
?>
<HTML>
<HEAD>
<TITLE> Ejemplo INTERCAMBIO </TITLE>
</HEAD>
<BODY>
<? PHP
$x=10;$y=20;
print("valores iniciales x=$x,y=$y  <BR>");
inter($x,$y);
printf("valores finales x=$x,y=$y  BR>");
?>
</BODY>
</HTML>
```

IV- Funciones
Ambito de las variables

***reglas de ámbito*:** controlan si un fragmento de código conoce o tiene acceso a otro segmento de código o datos.

- Toda variable definida dentro de una función tiene ámbito “local”.

- Una variable declarada fuera de una función tiene un ámbito “global” y no puede ser usada dentro si no se declara “global”

- Variable estática local: tipo especial de variable local que no pierde su valor cuando termina la ejecución de la función. Precisa usar “static” en su declaración

IV- Funciones
conceptos varios

***sentencia return*: tiene dos usos:**

- forzar la salida inmediata de la función
- devolver valor

***claúsulas "include" y "require"*:**
permiten insertar en un programa código que se encuentra en un fichero externo.

sintaxis

include("mifichero.php");
require("mifichero.php");

diferencias

En caso de no localizar "mifichero.php", "require" produce -Error Fatal- y no ejecuta el resto del programa. "Include" genera aviso y continúa con ejecución del resto de programas.

IV- Funciones
conceptos varios

***funciones variables*:** el nombre de una función puede ser almacenado en una variable durante la ejecución del programa *(ejemplo-3.8)*

***valores retornables por una funcion*:** puede ser de cualquier tipo: cadena de caracteres, valor numérico, valor booleano, array, ...Siempre irá asociado con la sentencia “return”

IV- Funciones
conceptos varios

Ejemplo-3.8 Funciones Variables

```
<? PHP
function  suma ($x, $y)  {
return ($x+$y);
}
function  resta ($x, $y)  {
return ($x-$y);
}
function  producto ($x, $y)  {
return ($x*$y);
}
function  cociente ($x, $y)  {
return ($x/$y);
}
?>
<HTML>
<HEAD>
<TITLE> Operaciones aritméticas </TITLE>
</HEAD>
<BODY>
<? PHP
$operador=array(“+”,”-”,”*”,”/”);
$a=10;
$b=100;
```

IV- Funciones
conceptos varios

```
for ($i=0;$i<4;$i++) {
switch ($operador[$i])  {
    case ‘+’:
        $accion=“suma”;
        break;
    case ‘-’:
        $accion=“resta”;
        break;
    case ‘*’:
        $accion=“producto”;
        break;
    case ‘/’:
        $accion=“cociente”;
        break;
    }
$resultado=$accion($a,$b);
print(“Resultado:
 $a $operador[i]  $b=$resultado <BR>”);
}
?>
</BODY>
</HTML>
```

IV- Funciones
conceptos varios

***recursividad*: es la propiedad que en C tienen las funciones de llamarse a si mismas**

características de la recursividad:

- en las funciones recursivas existirá un “if” que permite a la función volver sin ejecutar de nuevo la función.

- las versiones recursivas no minimizan tamaño de código ni ahorran espacio de almacenamiento de variables.

- las versiones recursivas son mas lentas que las iterativas equivalentes.

- Un número elevado de llamadas a función pueden desbordar la pila.

- Determinados algoritmos de ordenación (quitsort) son idóneos para tratarlos con funciones recursivas. También problemas de inteligencia artificial.

IV- Funciones
conceptos varios

Ejemplo-3.9 Funcion Factorial

```
 /* version factorial-iterativa */

function factorial($n)
{
$respuesta=1;
for ($t=1; $t<=$n;$t++)
     $respuesta *=$t;
return $respuesta;
}

/* version factorial-recursiva */
function factorial($n)
{
 if ($n==1) return 1;
 $respuesta=factorial($n-1)*$n;
return $respuesta;
 }
```

IV- Funciones *predefinidas*

PHP cuenta con mas de 700 funciones predefinidas que pueden ser utilizadas en el programa sin necesidad de invocar librería alguna.

Una breve clasificación:

- Matemáticas
- Trabajo con fechas
- Trabajo con cadenas
- Trabajo con arrays
- Ficheros y directorios (*)
- Acceso a bases de datos (*)
- Comunicaciones *(*)*

(*) Se verán en los correspondientes capítulos

Todo en ***http://www.php.net***

Otras referencias específicas de interés:

http://www.interec.com/tutoriales/manuales/php.phtml

IV- Funciones *predefinidas*

- Matemáticas

abs (n)	***exp (a)***
round (n)	***log (n)***
ceil (n)	***min (n1, n2, ...) ; min (array)***
floor (n)	***max (n1, n2, ...) ; max (array)***
cos (n)	***pow (a, b)***
sin (n)	***rand (min, máx), mtrand (min, máx)***
tan (n)	***sqrt (a)***

- Trabajo con fecha y hora

time ()
mktime (hora, minuto, segundo, mes, día año)
checkdate (mes, día año)
date (formato, instante)

IV- Funciones *predefinidas*

- Trabajo con cadenas

Reconocimiento de caracteres

ctype_alnum (cadena)
ctype_space (cadena)
ctype_alpha (cadena)
ctype_digit (cadena)
ctype_lower (cadena)
ctype_upper (cadena)
ctype_print (cadena)
ctype_punct (cadena)

Conversión cadena-array

explode (separador, cadena, límite)
implode (separador, array)
join (separador, array)
str_word_count (cadena, formato)

Prolongación de cadena

str_pad (cadena, longitud, cadena_añad, clase_ad)
str_repeat (cadena, número)

IV- Funciones *predefinidas*

Modificación de cadena

str_replace (buscar, reemplazar, cadena)
strrev (cadena)
strtolower (cadena)
strtoupper(cadena)
substr_replace (cadena, reemplazar, posic, long)
ucfirst (cadena)
ucwords (cadena)

Comparación de cadenas

strcasecmp (cadena1, cadena2)
strcmp (cadena1, cadena2)
strnatcasecmp (cadena1, cadena2)
strnatcmp (cadena1, cadena2)

Búsqueda de datos

strchr (cadena, carácter)
strrchr (cadena, carácter)
stristr (cadena, fragmento)
strstr (cadena, fragmento)
strpos (cadena, fragmento, inicio_búsqueda)
strrpos (cadena, carácter)
substr (cadena, longitud_inicial, longitud)
substr_count (cadena, fragmento)

IV- Funciones *predefinidas*

Subdivisión de cadenas

wordwrap (cadena, ancho, separación)
strtok (cadena, delimitador)

Longitud de una cadena

strlen (cadena)

- Trabajo con arrays

Transformación de los índices

array_change_key_case (array, caso)

Subdivisión

array_chuck (array, dimensión, valor_booleano)

Contabilización de elementos

array_count_values (array)
count (array)

IV- Funciones *predefinidas*

Búsqueda de datos

array_filter (array, función)
array_keys (array, elementos)
array_key_exists (indice, array)
array_rand (array, num_elementos)
array_search (elemento_buscado, array, v_logico)
array_values (array)
in_array (elemento, array)

Generación de arrays

array_combine (indices, elementos)
compact (ListaVariables)
extract (array, extracción)
range (inicio, final, aumento)
array_slice (array, fragmento, dimensión)
array_splice (array, frag, dim, nuevo)

Combinación de arrays

array_merge (array1, array2, array3, ...)
array_diff_assoc (array1, array2, array3, ...)
array_diff (array1, array2, array3, ...)
array_intersect_assoc (array1, array2, array3, ...)
array_intersect (array1, array2, array3, ...)

IV- Funciones *predefinidas*

Prolongación/truncamiento de array

array_fill (indice, cantidad, elemento)
array_pad (array, tamaño_final, elemento_nuevo)
array_pop (array)
array_push (array, elemento1, elemento2, ...)
array_shift (array)
array_unique (array)
array_unshift (array, elemento1, elemento2, ...)

Aplicación de funciones

array_map (función, array1, array2, array3, ...)
array_walk (array, nombreFunción, argumentos)
asort (array), arsort (array)
ksort (array), krsort (array)
natsort (array) , natcasesort (array)
array_reverse (array)
sort (array), rsort (array)

V- ARRAYS
Introducción

- Un array es una colección de datos que se referencian con un nombre común
- En PHP los datos pueden ser de diferente tipo (cadenas, valores numéricos, otros arrays, ...)
- A cada elemento especifico del array se accede mediante un índice
- No es necesario determinar la dimensión antes de inicializarlo
- Existen multitud de funciones que permiten la gestión y manipulación de arrays.
- Constan de posiciones de memoria consecutivas; la dirección mas baja es la del primer elemento

V- ARRAYS
conceptos

Arrays unidimensiolanes

- son en esencia listas de información *heterogénea* guardadas en posiciones contiguas de memoria

formatos y ejemplos de uso

$nombre_variable[indice]=valor;

valor= contenido del elemento

indíce= elemento del array

nombre_variable= identificador del arrary

V- ARRAYS
conceptos

Ejemplo.4.1.1- *//Array lineal de 4 elementos*

```
$X[0]=1;
$X[1]="una cadena";
$X[2]=3;
$X[ ]="ultimo elemento";
```

Ejemplo.4.1.2- *//Array lineal de 4 elementos. Uso de la funcion array)*

```
$X= array (1,"una cadena",3,"ultimo elemento");
```

Ejemplo.4.1.3- *//gestión de elementos*

```
$z= $x[0] + 5*$x[2];
```

Ejemplo.4.1.4- *//mi cuenta bancaria*

```
$micuenta["numero_cuenta"]=20001;
$micuenta["titular"]="Jose Perez";
$micuenta["saldo"=12000;
$micuenta["fecha_pago"]=01012006;
```

Ejemplo.4.1.5- *//mi cuenta bancaria*

```
$micuenta=array(["numero_cuenta"=>200,
"titular"=>"Jose Perez",
   "saldo"=>12000,"u_pago"=>01012006) ;
```

V- ARRAYS
conceptos

Arrays multidimensionales

es un array cuyos elementos son a su vez nuevos arrays

formatos y ejemplos de uso

$nombre_variable[indice1][indice2]..[indiceN]= valor;

Ejemplo.4.2.1- *//Array de 2x2 elementos*

```
$X[0][0]=1;
$X[0][1]=0;
$X[1][0]=-1;
$X[1][1]=-2;
```

Ejemplo.4.2.2- *//Array de 2x2 elementos*

```
$X=array(array(1,0),array(-1,-2));
```

V- ARRAYS
conceptos

Aspectos varios:

- La funcion “print_r ( )” muestra los elementos de un array

- La manipulación de los elementos requiere uso de sentencias iterativas (foreach, for, etc.)

- Funciones para manipulación de arrays (ver capítulo III)

VI- Conectividad a Base de Datos MySQL

Objetivo:

Utilización del lenguaje de consultas de bases de datos relacionales (SQL) embebido en un lenguaje de programación (PHP) que permita generar páginas web dinámicas.

- PHP tiene funciones de conectividad a las bases de datos, que son específicas del SGBD:
  - mysql_nombrefuncion
  - Ora_nombrefuncion
  - ifx_nombrefuncion
  - dbase_nombrefuncion
  - dngres_ombrefuncion

Ejemplo: mysql_connect()

VI- Conectividad a Base de Datos MySQL

Pasos a seguir en la conexión a base de datos:

1. Conexión con MYSQL desde PHP
    1. Apertura de conexión
    2. Cierre de conexión
2. Selección de la base de datos
3. Ejecución de sentencias SQL sobre la BD seleccionada
4. Otras funciones de manipulación de datos

VI- Conectividad a Base de Datos MySQL

Conexión con MySQL:

- Apertura de conexión

mysql_connect (servidor, usuario, contraseña)

La función devuelve un valor entero que servirá como identificador de la conexión en las operaciones posteriores. Si se produce un error en la conexión la función devuelve 0.

mysql_connect (“localhost”, root, “”)

- Cierre de conexión

mysql_close (identificador)

En caso de no poner el identificador se cerrará la última conexión abierta. La función devuelve un valor entero, que será nulo si se produce algún error en el proceso.

VI- Conectividad a Base de Datos MySQL

Conexión con MysSQL:

Esquema general de una conexión

```
<html>
<head>
                    <title>Problema 8_1</title>
</head>
<body>
<?php
$c = mysql_connect ("localhost", "root", "");
 if (!$c) {
    die ("Conexion no realizada");
 }
 else {
   print ("conexion establecida correctamente");
   }
mysql_close($c);
?>
 </body>
</HTML>
```

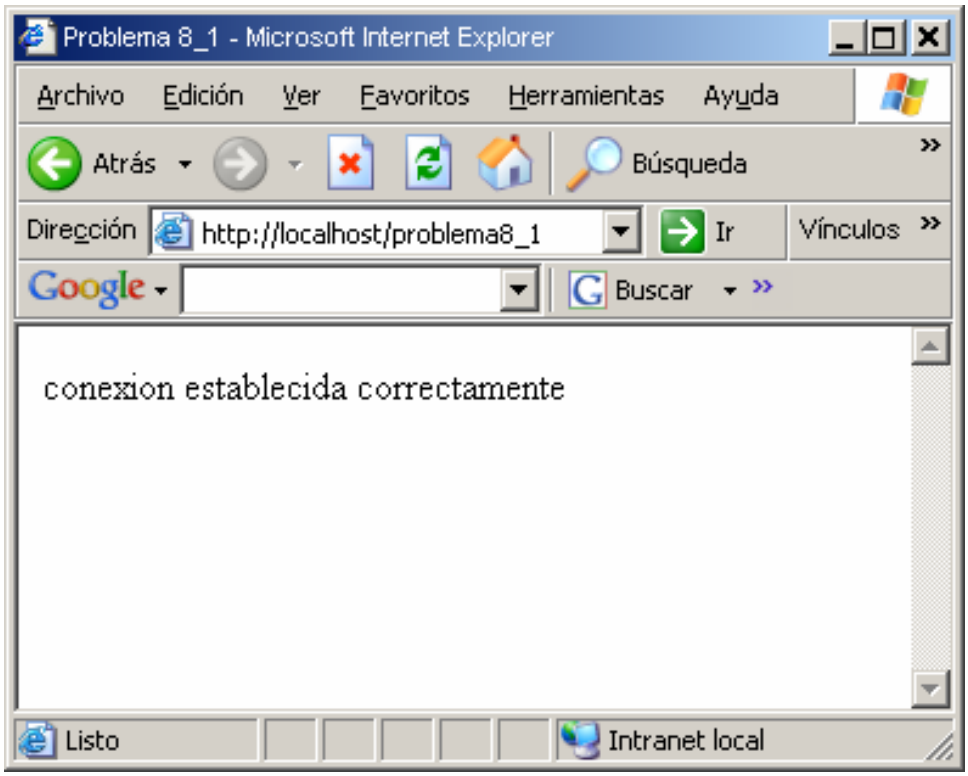
*En caso de no utilizar la función **mysql_close()**, la conexión se cerrará al finalizar el script.*

*la función **die ()**, sirve para mostrar el aviso de error si $c no nulo, y abortar ejecución del resto de programa*

VI- Conectividad a Base de Datos MySQL

Conexión con MysSQL:

Esquema general de una conexión



VI- Conectividad a Base de Datos MySQL

Selección de la Base de Datos:

Una vez abierta la conexión con el Servidor MySQL, el siguiente paso es la selección de la(s) base(s) de datos sobre la que queremos actuar.

mysql_select_db (nombreBD, identificador)

- *El identificador de la conexión es opcional. En caso de no utilizarlo se utilizará la última conexión abierta.*

- *Si se produce un error, la función devolverá valor 0.*

- *Una vez seleccionada la base de datos, ésta pasa a ser la base de datos activa, y cualquier operación posterior se dirigirá a ella.*

- *Para cambiar de base de datos se deberá volver a utilizar la función antes de cerrar la conexión.*

VI- Conectividad a Base de Datos MySQL

Selección de la Base de Datos:

```
<html>
<head>
                    <title>Problema 8_2</title>
</head>
<body>
<?php
$c = mysql_connect ("localhost", "root", "");
$bd= mysql_select_db("compras",$c);
 if (!$bd) {
    die ("Base de datos no existente");
 }

mysql_close($c);
?>
 </body>
</HTML></HTML>
```

Aborta el programa si no se puede seleccionar la base de datos ***compras.***

Probar a cambiar ***compras*** *por otros nombres*

VI- Conectividad a Base de Datos MySQL

Ejecución de sentencias SQL sobre la base de datos seleccionada

- *Realización de una consulta sobre la base de datos activa.*

mysql_query (sentencia, identificador)

El identificador de la conexión es opcional. En caso de no utilizarlo se utilizará la última conexión abierta.

Si se produce un error, la función devolverá el valor 0 en caso de una ejecución correcta se devuelve un entero.

En caso de ser una consulta de selección, este entero se utilizará posteriormente para acceder al resultado de la consulta.

En el caso de sentencias de creación, inserción, eliminación o actualización, este valor devuelto sirve para comprobar si la sentencia se ejecutó correctamente.

VI- Conectividad a Base de Datos MySQL

Ejecución de sentencias SQL sobre la base de datos seleccionada

```
<html>
<head>
                    <title>Problema 8_3</title>
</head>
<body>
<?php
$c = mysql_connect ("localhost", "root", "");
if (!$c) {
   die ("Conexion fallida");
 }
$ok= mysql_select_db("compras",$c);
 if (!$ok) {
   die ("Error en la selección de la base de
datos");
 }
 $consulta="SELECT * from articulos";
 $resultado=mysql_query($consulta, $c);
 if ($resultado) {
print "<H2 ALIGN=center>
.........
```

VI- Conectividad a Base de Datos MySQL

Ejecución de sentencias SQL sobre la base de datos seleccionada

mysql_query (sentencia, identificador)

mysql_db_query (nombreBD, sentencia, identificador)

sentencia idéntica a la anterior pero que permite ejecutar la sentencia sobre cualquier base de datos.

VI- Conectividad a Base de Datos MySQL

Obtención de errores en el acceso a BD

mysql_error ( )

Devuelve una cadena de caracteres explicando el último error producido en la operación de acceso a datos.

Inserción de datos mediante formulario

La verdadera utilidad de las funciones de conectividad a bases de datos desde PHP radica en la posibilidad de construir sentencias SQL de forma dinámica.

VI- Conectividad a Base de Datos MySQL

Recuperación de los resultados de consultas

Para acceder a los resultados de una consulta de selección.

mysql_fetch_row (resultado)

•*Para acceder al resultado de la consulta de selección se utiliza el valor que devuelve la función mysql_query().*

• *Cada vez que se produce la llamada a la función se genera un array con los valores de una fila de la tabla de resultados.*

•*Cuando no hay más filas que mostrar la función devuelve un valor nulo.*

Para saber cuantas filas o registros incluye el resultado de la consulta.

mysql_num_rows (resultado)

VI- Conectividad a Base de Datos MySQL

Recuperación de los resultados de consultas

Para conocer el número de registros afectados en una consulta de inserción, actualización o borrado

mysql_affected_rows ( )

Ejemplo de acceso a los datos de una consulta de selección.

```
$c = mysql_connect ("localhost", "root", "");
mysql_select_db ("Compras", $c);
$strSQL= "SELECT * FROM Articulos;";
$rs = mysql_query ($strSQL, $c);
while ($registro = mysql_fetch_row($rs)) {
   print("Codigo: $registro[0].<br>");
   print("Articulo: $registro[1].<br>");
   print("Descripción: $registro[2].<br>");
   print("Precio: $registro[3] euros.<br>");
}
mysql_close ($c);
```

VI- Conectividad a Base de Datos MySQL

Recuperación de los resultados de consultas

En el ejemplo anterior se necesitó saber el orden de los campos dentro de la tabla de la base de datos. Para facilitar el acceso a los datos PHP dispone de otra función.

mysql_fetch_array (resultado)

- *Función similar a mysql_fetch_row().*
- *Principal ventaja:datos son devueltos en un array asociativo en el que en lugar de utilizar índices numéricos de pueden utilizar los propios nombres de los campos.*

```
...
while ($registro = mysql_fetch_array($rs)) {
   print("Codigo: $registro['codigart'].<br>");
   print("Descripción:
$registro['descrart'].<br>");
   print("Precio: $registro['precart']
euros.<br>");
}
...
```

EJEMPLO 8.3