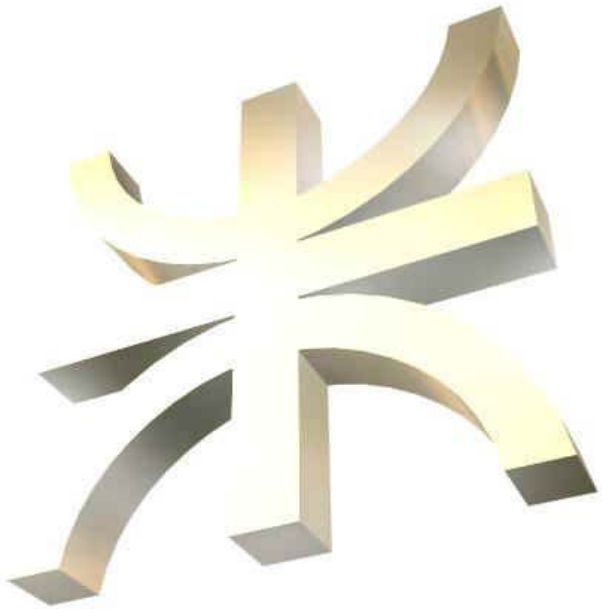
Modulo II – PHP y MySQL inicial

U.T.N.



Contenidos del módulo

- Estructuras de control selectivas
- Estructuras de control repetitivas
- Arrays
- Funciones

Estructuras de control selectivas

- **El condicional if**

Sintaxis:

```
if (condición) {
    //Código a ejecutar si la condición es verdadera
    }
else {
    //Otro código a ejecutar en caso contrario
}
```

Estructuras de control selectivas

- **El condicional switch**

Sintaxis:

```
switch(expresión) {
  case "a": //Código a ejecutar;
            break;
  case "b": //Código a ejecutar;
            break;
  case "c": //Código a ejecutar;
            break;
 default: //Código a ejecutar por default.
}
```

Estructuras de control selectivas

- Un switch busca dentro de los “case” el valor de la variable o expresión evaluada (generalmente se evalúan variables).
- Si lo encuentra, ejecuta el código correspondiente.
- Si no lo encuentra, ejecuta el código por default.
- *Observaciones:* El case por default es opcional.
- Es importante poner “break;” al final de cada bloque de código dentro de cada case para que el switch no siga comparando al valor de la variable con los case que le siguen al correcto.

Estructuras de control repetitivas

- Los bucles son estructuras que controlan el flujo de ejecución del código, pero ya no en forma "condicional" o "selectiva" como en el caso de los if/else y switch, sino que determinan la ejecución "repetitiva" de un código.

- Debido a su característica repetitiva, son ideales para recorrer vectores, o conjuntos de filas o registros traídos de una base de datos.

- Existen dos funciones básicas en PHP (no son las únicas) para las estructuras de tipo bucle: **for y while.**

Los bucles for

- **Cómo repetir una acción una cantidad fija de veces: el bucle "for".**

- Este tipo de instrucción llamada **for** permite **ejecutar una orden "a repetición" un número "exacto" de veces**, es decir, el número de repeticiones debe ser conocido con anterioridad.

- El bucle **for** trabaja modificando el valor de una variable que utiliza como "**índice**" o indicador del **número de vuelta** en el que se encuentra el bucle (un "cuentavueltas")

Los bucles for

- Esta variable $indice debe ir en primer lugar, como primer parámetro dentro de los paréntesis del for, fijando el valor original que tendrá esa variable al **inicio** del bucle.

- Luego, como segundo argumento, se fija el valor **tope** (o final) de esa variable -es el número hasta el cual se repetirán las instrucciones-.

- En tercer lugar, se especifica qué **incremento** o decremento sufrirá esa variable luego de terminar de dar cada vuelta del bucle.

```
for (valorinicial; valorfinal; incremento){
        bloque a ejecutar;
}
```

Los bucles for

Ejemplo:

```
for ($indice=0; $indice<10; $indice=$indice + 1) {

Aquí van las instrucciones que ejecutará 10 veces
(mientras $indice valga menos que 10);

}
```

Los bucles while

- Los bucles While ejecutan un bloque de código mientras se cumpla la condición especificada.

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

- Muchas veces nos sucederá que debemos realizar una acción hasta que pase algo, pero no sabemos a ciencia cierta cuándo sucederá.

- Ese algo puede ser una condición que se de en el programa por ejemplo voy a leer mientras haya registros en el archivo cuando no haya mas registros saldré del bucle. O puede darse por una acción del ususario, por ejemplo mientras el usuario no presione una determinada tecla seguiré ejecutando las instrucciones del bucle.

Los bucles while

Ejemplo:

```
while ($condicion !=“Salir”) {

Aquí van las instrucciones se que ejecutarán mientras el
    valor de la variable $ condicion sea distinto a “Salir”
}
```

Vectores o arrays

Los vectores (también llamados arrays, matrices de una sola dimensión o tablas) son un lugar donde almacenar datos, como las variables, pero con la posibilidad de almacenar "varios" datos y no uno solo.

Son realmente útiles e imprescindibles para trabajar con bucles y con bases de datos, ya que todos los datos que se traen de una base suelen terminar "acomodados" en un vector.

Vectores o arrays

Este código declara y adjudica valor a una **variable**:

```
<?php
$numero = 514;
print ($numero); //imprime un 514.
?>
```

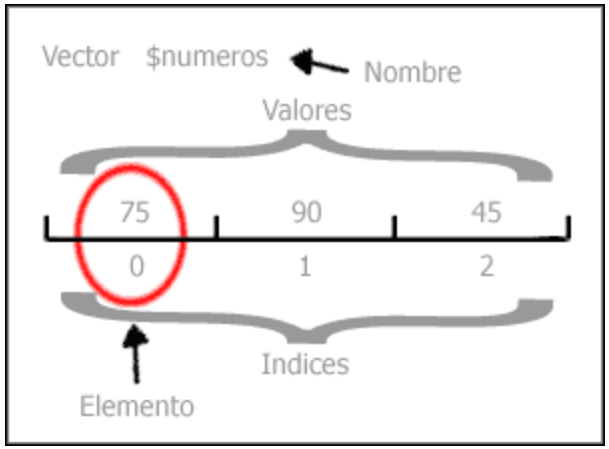
El nombre de la variable es $numero**,** y el valor almacenado en este momento es el número 514.
Podríamos decir que la variable es como "una cajita" con un único compartimiento, donde colocamos caracteres (letras, números), y les asignamos un "sobrenombre" para referirnos a "eso" que guarda la variable.

Vectores o arrays

Ahora veamos un vector:

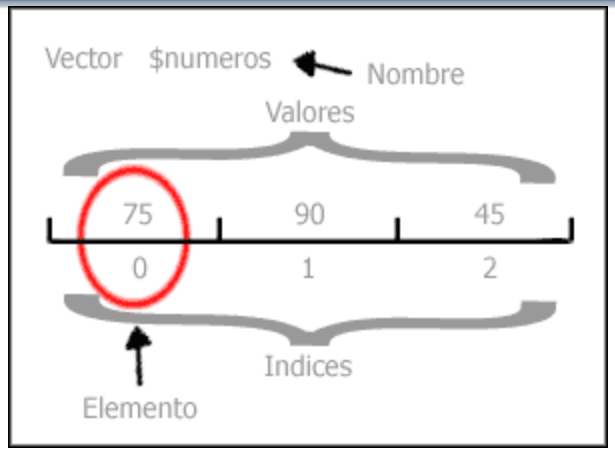
En el siguiente caso, nuestro código declarará un vector de 3 elementos o celdas al que llamaremos $numeros:

```
<?php
$numeros[0] = 75;
$numeros[1] = 90;
$numeros[2] = 45;
print ($numeros[0]."<br
/>".$numeros[1]."<br />".$numeros[2]);
//imprime 75<br />90<br />45
?>
```



En este caso, el nombre del vector es $numeros, y tiene 3 "subdivisiones internas" llamadas "elementos", "celdas" o "posiciones" (cada subdivisión sería lo envuelto en rojo en el gráfico anterior): cada elemento es un par que consta de un "índice" (su nombre) y un "valor".

Vectores o arrays



El primer elemento de este vector tiene como índice el número "0" y como valor un "**75**".
El segundo elemento, con índice "1", almacena un "90", y el tercer elemento de índice "2" guarda un número "45" en su interior.
A diferencia de una variable, un vector almacena varios datos, cada uno de ellos con un nombre de índice diferente.
Los índices de cada celda de un vector no se numeran a partir de 1 en adelante, sino que a partir de cero.

La primera celda es la [0] (se lee "sub cero"). La segunda, sub 1, y la tercera sub 2 y así sucesivamente.

Vectores o arrays

También se puede utilizar una forma mucho más simple y más breve de declarar vectores, que es mediante el constructor "array" , aquí PHP define de manera implícita los índices del array de 0 en adelante

```
<?php
$paises = array("Argentina", "Uruguay", "Chile", "Perú");
// crea un vector llamado $paises de cuatro elementos de tipo string.

$loteria = array(23,8,36,12,99);
// crea un vector de cinco elementos de tipo integer.

$usuario = array("Juan Pérez", 24, "casado", 800);
// crea un vector que alterna datos string con datos integer.
?>
```

Vectores o arrays

En muchos casos, en especial cuando trabajemos con bases de datos, nos será de enorme utilidad definir los índices del vector con cadenas de texto en vez de con números.

```
<?php
$datos["nom"] = "Juan Pérez";
$datos["ed"] = 24;
$datos["est"] = "casado";
$datos["suel"] = 800;
print ($datos["nom"]);
//imprime Juan Pérez
?>
```

Funciones

Las funciones son un conjunto de sentencias o instrucciones, que nos permiten pasarles variables (o parámetros) y recibir un resultado de vuelta.

Todas las funciones se definen con la palabra **function** delante del nombre de la función, luego paréntesis (), que pueden o no contener parámetros dentro, y por último las instrucciones de la función que van entre llaves {}

```
<?php
function mifuncion(){
instrucciones;
}
?>
```

Funciones

Las funciones en general son usadas para resumir procesos que son utilizados muchas veces, por lo que es conveniente tenerlos resueltos una sola vez en una función y luego simplemente llamar a dicha función.

Hay dos cosas importantes que debemos saber sobre las funciones, para pasarle datos a una función, esta función debe aceptarlos entre los paréntesis, y para que una función nos devuelva un resultado debemos usar la sentencia return.
Pongamos un ejemplo de una función que acepte dos parámetros, los multiplique entre si y nos devuelva el resultado:

Funciones

```php
<?php
function producto($num1,$num2){
$producto = $num1 * $num2;
return $producto;
}
$variable1 = producto(10,3); // $variable1 valdrá 30 (10*3)
$variable2 = producto(17,3); // $variable2 valdrá 51
echo $variable1."<br>";
echo $variable2;
?>
```

Funciones

- **Otra función.**

  funcion1.php:

```
<?php
function suma ($a,$b){
$resultado = $a + $b;
return $resultado;
}
echo suma(5,2)."<br>";
?>
```

Funciones propias del lenguaje

Aparte de poder definir nosotros nuestras funciones según nuestras necesidades, PHP nos brinda una cantidad de funciones predefinidas que veremos mas adelante, las cuales pueden utilizare de la misma manera. Las mismas se encuentran clasificadas dentro del manual de PHP en funciones matemáticas, funciones de string, funciones para manejo de archivos, funciones para arrays, funciones para manejo de sql, entre otras.

En la última unidad del módulo veremos algunas de las funciones para manejo de sql y en el nivel intermedio vemos en detalle algunas mas, de string, matemáticas y para manejo de archivos.