UTN.BA FACULTAD REGIONAL BUENOS AIRES
SECRETARÍA DE EXTENSIÓN UNIVERSITARIA FRBA UTN

Centro de e-Learning

EXPERTO UNIVERISTARIO EN MySQL Y PHP NIVEL INICIAL

www.sceu.frba.utn.edu.ar/e-learning

Módulo 2

CONTINUAMOS CON MySQL Y PHP

Estructuras de control repetitivas: FOR, WHILE, DOWHILE.

Presentación de la Unidad:

Aprenderemos las estructuras de control repetitivas y como influyen en el flujo de nuestro programa.

Empezamos a integrar los contenidos de las unidades anteriores como variables, constantes, los distintos tipos de operadores. E incorporamos las estructuras de control repetitivas para armar programas un poco mas complejos.

Objetivos:

- **Conocer las estructuras de control repetitivas: FOR, WHILE y DOWHILE. Y como van a alterar éstas el flujo de nuestro programa.**

Temario:

Estructuras Control de Flujo - Bucles

Bucles (Ciclos o loops)

- **While**
- **do... while**
- **for**

Estructuras Control de Flujo - Bucles

Bucles (Ciclos o loops):

Las computadoras, como cualquier máquina, están diseñadas para realizar tareas repetitivas. Es por ello que nuestros programas pueden aprovecharse de este principio para realizar una determinada secuencia de instrucciones un cierto número de veces.

Para ello, utilizamos las estructuras llamadas bucles que nos ayudan a, usando unas pocas líneas, realizar una tarea incluida dentro del bucle un cierto número de veces definido por nosotros mismos.

Los bucles nos permiten iterar conjuntos de instrucciones, es decir repetir la ejecución de un conjunto de instrucciones mientras se cumpla una condición.

Hay 3 ciclos fundamentales:

While

El while ejecutará cierto bloque de código mientras una condición dada a evaluar resulte siempre verdadera.

Primero evalúa si es verdadera la expresión dada como condición, luego ejecuta el código, y cuando termina de ejecutar la última línea del mismo vuelve a evaluar la condición. Si sigue siendo verdadera, vuelve a ejecutar el código, si no, deja de ejecutarlo, sale del ciclo y sigue con el código que siga debajo.

Sintaxis:

```
while (expresión) {
//Código a ejecutar
}
```

Observación:

Hay que tener cuidado de no caer en un ciclo infinito, ya que provoca problemas en la computadora (y además no tendría sentido).

Para eso, tras cada vuelta del while hay que asegurarse que en la condición no

se esté evaluando lo mismo que la vuelta anterior, cambiando algo de ella a través del código del mismo while.

Los bucles while son los tipos de bucle más simples en PHP, actúan de forma muy parecida
al bucle FOR (que veremos en esta misma unidad), pero se diferencia de éste que no incluye en su declaración la inicialización de la variable de control del bucle, ni su incremento o decremento.

Nota:

Las llaves no son estrictamente necesarias para cuando dentro de los bucles while o for hay solo una sentencia, pero si recomendable ponerlas. Cuando hay más de una sentencia dentro del while o for, hay que poner las llaves.

Si intentamos traducir a 'lenguaje humano' que significa o que hace el while, tomando el ejemplo de sintaxis básica que he puesto arriba, sería algo así:
Mientras (while) expresión sea verdadero (mientras la expresión que pongas se cumpla y sea cierto) ejecuta la(s) sentencia(s).

**El significado de una sentencia *while* es simple. Le dice a PHP que ejecute la(s) sentencia(s)
anidada(s) repetidamente, mientras la expresión *while* se evalúe como TRUE.**

El valor de la expresión es comprobado cada vez al principio del bucle, así que incluso si este valor cambia durante la ejecución de la(s) sentencia(s) anidada(s), la ejecución no parará hasta el fin de la iteración (cada vez que PHP ejecuta las sentencias en el bucle es una iteración).

A veces, si la expresión *while* se evalúa como FALSE desde el principio de todo, la(s) sentencia(s) anidada(s) no se ejecutarán ni siquiera una vez.

Como con la sentencia *if*, se pueden agrupar multiples sentencias dentro del mismo bucle
while encerrando un grupo de sentencias con llaves, o usando la sintaxis alternativa.

Con el bucle while podremos ejecutar un conjunto de instrucciones un número indeterminado de veces, siempre y cuando el resultado de la expresión sea verdadera (debe ser una expresión que se evalúe a un valor lógico).

Si la expresión se evalúa y da **true** se ejecutan las sentencias del bucle, después de ejecutarlas vuelve a evaluar la expresión.
Esto se repetirá hasta que la expresión se evalúe y de **false** (no se cumpla la expresión), en cuyo caso dejaría de ejecutarse el bucle while y continuaría con la

ejecución del script (con lo que haya después de la llave } que cierra el while).

Ejemplo:

```
<?php
$cuenta = 0;
echo "Voy a entrar al bucle while <br>";
while ( $cuenta <= 10) {
echo "Cuenta vale $cuenta <br>";
$cuenta++;
}
echo "He salido del bucle while";
?>
```

El ejemplo es sencillo.

Simplemente es una cuenta de 0 a 10, ambos inclusive.

Analicemos el ejemplo.

- Lo primero que hago es inicializar a 0 la variable $cuenta (es recomendable inicializar las variables).

- A continuación imprimo por pantalla una línea antes de entrar al bucle, simplemente para que vean cuando está dentro y cuando no.

- Bien, llega el while. Como expresión a evaluar se ha puesto $cuenta <= 10, quiere decir que mientras el valor de $cuenta sea menor o igual a 10, las sentencias del bucle se ejecutarán hasta que ésta expresión sea falsa.

- Entonces, al principio, hemos inicializado $cuenta a 0, por lo que $cuenta (que vale 0) es menor o igual que 10 (esto da true, verdadero), así pues entra en el bucle e imprime mediante el echo el valor de $cuenta, recordemos siempre que podemos usar código HTML dentro de php, en este caso usamos
 para que ponga un valor en cada línea diferente.

Importante:

¿Que sentido tiene $cuenta++? sencillo, como hemos dicho al principio y a diferencia del for, el while no tiene operador de incremento / decremento en su expresión, por lo que tendremos

que ponerlo antes de la llave que cierra el while, esto quiere decir que cada vez que ejecute las sentencias, incremente en +1 el valor de $cuenta, a continuación vuelve a evaluar la expresión hasta que no se cumpla (sea false, falso).

- Finalmente, cuando el bucle haya dado varias vueltas y la variable $cuenta haya alcanzado el valor 11, evaluará la expresión, ésta dará false, y las sentencias del bucle no volverán a ejecutarse y se seguirá el curso normal del script, en este caso lo siguiente al bucle while es que imprima por pantalla “He salido del bucle while”.

Otro ejemplo:

```
<?php
$i = 1;
$suma = 0;
while ( $i < 10) {
$suma = $suma + $i;
$i ++;
}
echo "<strong>Suma total: $suma</strong>";
?>
```

En este ejemplo, similar al anterior, lo que hacemos es una cuenta de 1 a 10, y en cada iteración del bucle acumulo la suma de los valores del 1 al 10, dando como resultado 45.

Vamos a hacer un ejemplo un poco más elaborado, combinando el bucle while con estructuras de control, como el IF:

```
<?php
$numero = 15;
$pares = 0;
$impares = 0;
while ( $numero > 0) {
if ( $numero % 2 == 0 ) {
echo "El $numero es un número PAR <br> ";
$pares++;
}
else {
echo "El $numero es un número IMPAR <br>";
```

```
$impares++;
}
$numero--;
}
echo "En total he contado <strong>$pares números pares</strong> y
<strong>
$impares
impares<
/strong>"
;
?>
```

Bien, este último ejemplo es para que veamos que es posible meter dentro de un bucle, ya sea un WHILE, FOR... estructuras de control como puede ser un IF.

En el ejemplo recorreremos los números cuenta atrás de 15 al 1, con el if hacemos una comparación, si el resto de la división del numero entre 2 da cero, significa que es un número par, sino es impar, y acumulamos en dos variables diferentes si es par o impar.

Finalmente muestro el resultado de números pares e impares contados.

do... while:

Ejecuta una vez cierto bloque de código por primera vez, y luego actúa igual que el while.

Es decir, su uso es similar a while, pero aquí, las sentencias que siguen al do (Español: Hacer) se ejecutan por lo menos una vez y se comprueba la condición luego de la primera iteración; así, si es verdadera la condición se repite por segunda vez, si es falsa se continúa con las sentencias inmediatamente después de la instrucción while. **Tiene sólo una sintáxis.**

Sintaxis:

```
do {
//Código a ejecutar
} while (condición);
```

Todo lo explicado referente al bucle while se aplica también al **do...while** ya que son muy similares, excepto que en estos últimos, las condiciones se comprueban al final de cada iteración en vez de al principio.

La principal diferencia frente a los bucles regulares while es que se garantiza la

ejecución de la primera iteración de un bucle do..while (la condición se comprueba sólo al final de la iteración), mientras que puede no ser necesariamente ejecutada con un bucle while regular (la condición se comprueba al principio de cada iteración, si esta se evalúa como FALSE desde el principio la ejecución del bucle finalizará inmediatamente).

```
<?php
$i = 0; do { print $i;
} while ($i>0);
?>
```

El bucle de arriba se ejecutaría exactamente una sola vez, después de la primera iteración, cuando la condición se comprueba, se evalúa como FALSE ($i no es más grande que 0) y la ejecución del bucle finaliza.

for

El for es muy similar al while, pero se asegura de que los valores evaluados en la condición siempre cambien, y se usa cuando se quiere hacer un ciclo de determinado número fijo de veces.

Es decir, **FOR** nos permite hacer un conjunto de instrucciones o sentencias un número determinado de veces.

Los bucles for son los bucles más complejos en PHP.

Se comportan como en el lenguaje de programación C. La sintaxis es la siguiente:

```
<?php
for (inicialización; condición, incremento/decremento) {
sentencia (s);
}
?>
```

- Inicialización:

Normalmente se utiliza para inicializar y declarar la variable o variables que se van a utilizar como controladores del bucle, ésta Inicialización sólo se ejecuta una vez al principio del bucle.

- Condición:

Define la condición que ha de cumplirse para poder ejecutar las sentencia(s) que

hay entre las llaves { }, mientras la Condición sea cierta se ejecutarán las sentencias. La Condición se evalúa en cada iteración, y en el momento que la Condición no se cumpla, el bucle llega a su fin y no vuelve a ejecutar las sentencias. Tenemos que prestar especial atención a esta Condición ya que si esa condición siempre se cumple y no tiene fin nos encontraríamos con un bucle infinito.

- Incremento/decremento:

Modifica el valor de la variable del bucle. Se ejecuta en cada iteración del bucle, al igual que la Condición. Sirve para incrementar el valor (o decrementar) de la variable que controla el flujo del bucle.

Nota:

Estas tres expresiones que acabamos de explicar van separadas por punto y coma (;).

Ejemplo:

El primer ejemplo que haremos es mostrar en la página los números del 1 al 100:

```
<html>
<head>
<title>Problema</title>
</head>
<body>
<?php
for($f=1;$f<
=100;$f++)
{
echo $f;
echo "<br>";
}
?>
</body>
</html>
```

Vamos a detallar un poco más sobre la estructura for:

```
for($f=1;$f<=100;$f++)
{
```

```
echo $f;
echo "<br>";
}
```

- El primer argumento del for es la inicialización de una variable, en este caso se inicializa la variable $f con el valor 1. Este primer argumento del for se ejecuta solo una vez.

- Luego se ejecuta el segundo argumento que es la condición.
  Si la misma se verifica como verdadera se ejecuta todo el bloque comprendido entre las llaves de apertura y cerrado.

- Luego de haberse ejecutado el bloque repetitivo se ejecuta el tercer argumento del for que es el incremento de la variable, en este caso $f++ incrementa el contenido de la variable $f en 1 (también podemos poner en lugar de $f++ la asignación $f=$f+1).

- Luego del incremento de la variable se ejecuta nuevamente la condición del for (segundo argumento), de validarse nuevamente verdadero pasa a ejecutar el bloque repetitivo.

- Este ciclo se repite hasta que la condición del for se verifica false.

Otro ejemplo:

```
<?php
for ( $i = 1 ; $i <= 10 ; $i ++) {
print $i ;
}
?>
```

Veamos una breve explicación del ejemplo.
Como vemos, ha impreso en la pantalla 1 2 3 4 5 6 7 8 9 10.

Vayamos por partes. En lenguaje humano la linea for ($i = 1 ; $i <= 10; $i ++) significa: Empieza desde 1 y llega hasta 10, y suma en cada vuelta +1.

Es decir, iniciamos la variable $i a 1 ($i = 1), y le damos como condición para que finalice la ejecución de las sentencias, que $i sea menor o igual a 10, y en cada vuelta del bucle le sumamos uno a la variable $i.

Más ejemplos:

```
<?php
for ( $i = 10 ; $i >= 1 ; $i --) {
print $i;
echo ' ';}
?>
```

Ahora es el caso contrario.

Le decimos que tiene que ir desde el 10 ($i = 10) hasta el número 1 ($i >= 1) y le restamos uno cada vuelta del bucle, como resultado da: 10 9 8 7 6 5 4 3 2 1

Y ahora algo un poco más elaborado. Mediante un for vamos a calcular el factorial de un número (para todo número natural n, se llama n factorial o factorial de n al producto de todos los naturales desde 1 hasta n).

Ejemplo:

```
<?php
$numero = 5;
echo "Factorial de $numero = ";
$factorial = 1;
for ( $i = $numero ; $i >= 1 ; $i --) {
$factorial *= $i;
echo "$i"; if ($i == 1) echo " = "; else
echo " x ";
}
echo "$factorial";
?>
```

En este sencillo ejemplo calculamos el factorial de 5 ($numero = 5) y mostramos por pantalla el resultado obtenido, dando: Factorial de 5 = 5 x 4 x 3 x 2 x 1 = 120.