Tecnicatura Superior en Análisis de Sistemas

PROGRAMACIÓN V

Tema 3- Estructuras de control.

Las estructuras de control nos permiten elegir diferentes caminos en función de los datos que evaluamos en cada momento, nos permiten tomar decisiones, realizar acciones repetitivas, dependiendo de unas condiciones que nosotros mismos establecemos previamente.

Estas estructuras de control pueden ser:

- Estructuras de control condicionales.
- Estructuras de control repetitivas.

Las estructuras condicionales permiten evaluar una condición o varias y elegir el camino correcto. Las estructuras repetitivas repiten un número determinado de veces un conjunto de instrucciones.

Estructuras condicionales

```
<?php
    if(expresión) {

        sentencias 1, 2, 3;

    } else {

        sentencias A, B, C;

    }
?>
```

En la sentencia **if**, si la expresión se cumple se ejecutan las sentencias 1, 2, 3…, y en caso de que no se cumpliese la expresión, se ejecutarían las sentencias que están dentro del **else**, es decir, las sentencias A, B, C.

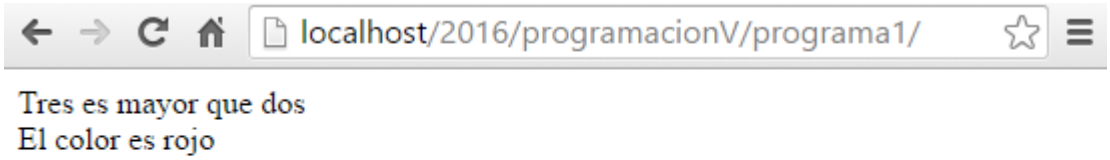
Ejemplo práctico:

```php
<?php
    if (3>2){
      echo "Tres es mayor que dos";
    }else{
      echo "Tres no es mayor que dos";
    }
}?>
```

```php
<?php
    $miVariable = "Rojo ";

    if ($miVariable == "Rojo"){
        echo "El color es rojo";
    }else{
        echo "No es rojo";
    }
?>
```

Y en el navegador veríamos lo siguiente:



Tres es mayor que dos
El color es rojo

La sentencia **switch** es similar a una serie de sentencias **IF** en la misma expresión. En muchas ocasiones, es posible que se quiera comparar la misma variable (o expresión) con muchos valores diferentes, y ejecutar una parte de código distinta dependiendo de a qué valor es igual. Para esto es exactamente la expresión **switch**.

```php
<?php
    switch (expresión) {
        case valor1:
           instrucciones;
        break;
        case valor2:
           instrucciones;
        break;
        case valor3:
           instrucciones;
        break;
    }
?>
```

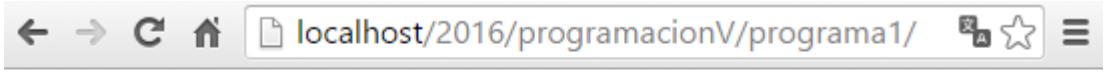
En este ejemplo la expresión será normalmente una variable cuyo contenido queremos evaluar, pero puede ser una operación matemática, una expresión booleana entre otras. Se evalúa cada caso y de ser positivo entra y ejecuta las instrucciones.

Ejemplo práctico:

```php
<?php
    $variable = 1;

    switch($variable){
        case(1):
        echo "La variable es igual a 1";
    break;
        case(2):
        echo "La variable es igual a 2";
    break;
}
?>
```

Testea la variable y la compara, si es verdadero ejecuta el código, como la variable es igual a 1, en el navegador veríamos lo siguiente:



La variable es igual a 1

Estructuras repetitivas

El bucle **while**, esta instrucción ejecuta un bloque de programa mientras se cumpla una cierta condición. Si la condición es verdadera, ingresa en el ciclo del **while**, y ejecuta la sentencia. Cuando deja de cumplirse la condición, sale del ciclo y continúa ejecutándose el resto del programa.

Si por el contrario, la condición no se cumple de entrada, las líneas en el interior del while no se ejecutarán ni una vez.

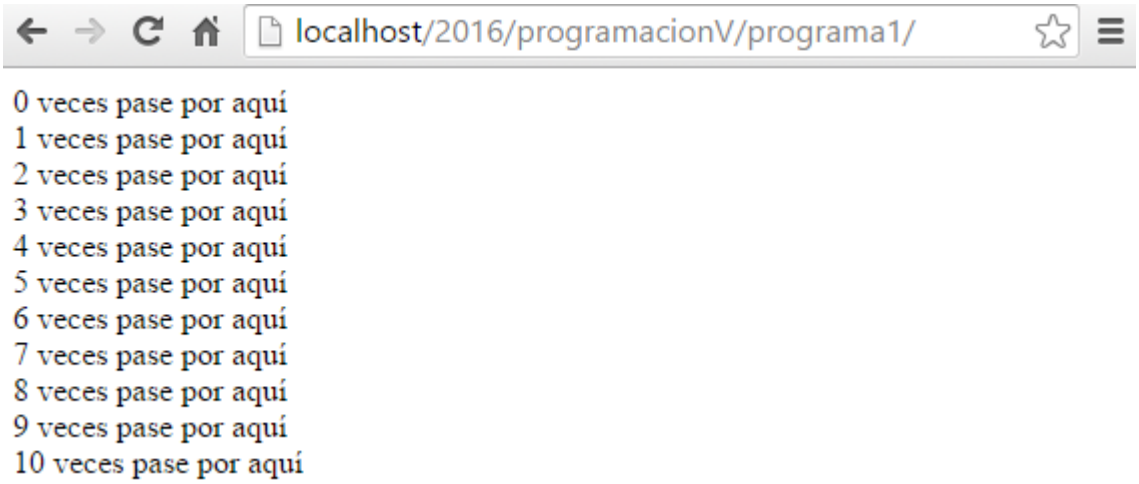
```
<?php
    while (Condición){
       Instrucción 1;
       Instrucción 2;
       Instrucción 3;
    }
?>
```

Ejemplo práctico:

```
<?php
    $miVariable = 0;

    while ($miVariable <= 10){
        echo $miVariable." veces pase por aquí
<br>";
        $miVariable = $miVariable +1;
    }
?>
```

Y lo que veríamos en el navegador es la entrada de 0 a 10 que pasa la variable, hasta llegar a ser mayor o igual a 10, en donde sale del bucle while:



Los bucles **do-while** son muy similares a los bucles while, excepto que la expresión verdadera es verificada al final de cada iteración en lugar que al principio. La diferencia principal con los bucles **while** es que está garantizado que corra **la primera iteración** de un bucle do-while (la expresión verdadera sólo es verificada al final de la iteración), mientras que no necesariamente va a correr con un bucle while regular (la expresión verdadera es verificada al principio de cada iteración, si se evalúa como FALSE justo desde el comienzo, la ejecución del bucle terminaría inmediatamente).

```
<?php
     do {
        sentencia 1;
        sentencia 2;
        sentencia 3;

    } while(condición);
?>
```

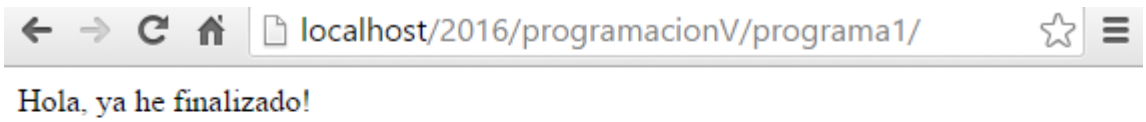
Ejemplo práctico:

```
<?php
$miVariable = 1;

    do{
        echo "Hola";

        }
    while ($miVariable >2);
        echo ", ya  he finalizado!<br>";
?>
```

Primero ejecuta y después va al **while** y como 1 no es mayor que 2, sale.



Con la instrucción **for**, se tienen que introducir tres expresiones que, mientras devuelvan "TRUE" permiten la ejecución de las sentencias.

```
<?php
    for (expresión_inicial;
condición_continuación; expresión_paso) {
    sentencia 1;
    sentencia 2;
    sentencia 3;
}
?>
```

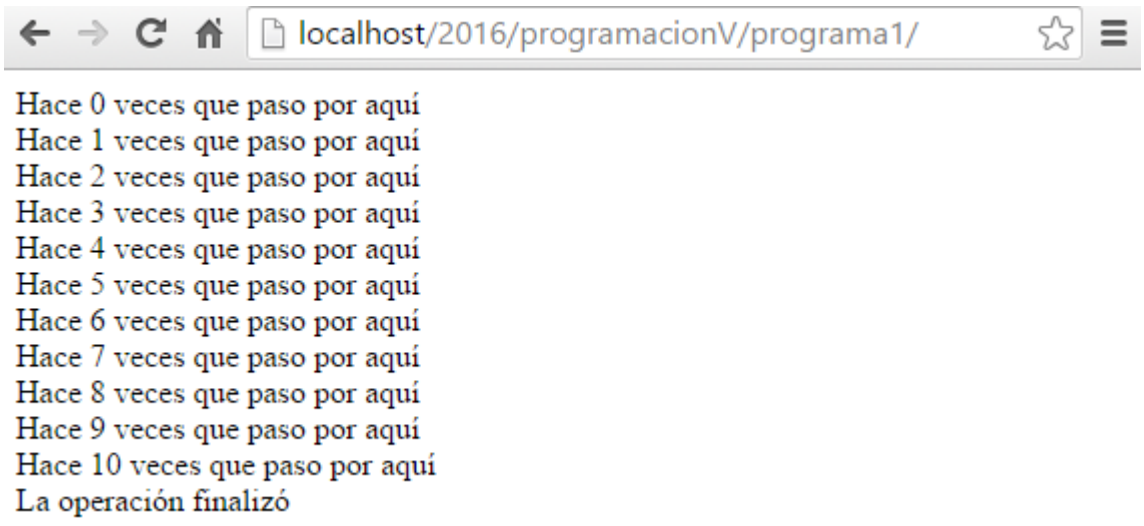
La expresión inicial se evalúa siempre. La condición de continuación se evalúa al principio de cada iteración: si el resultado es true se ejecuta primero el bloque de sentencias, después la expresión de paso y finalmente se evalúa nuevamente la condición de continuación; si el resultado es false el bucle se termina.

Ejemplo práctico:

```
<?php
for ($numero = 0; $numero<=10; $numero++){
        echo "Hace ".$numero." veces que paso por
aquí <br>";
        }
        echo "La operación finalizó";
?>
```

Mientras la variable número sea igual a 0 y menor o igual a 10 que la variable aumente 1 caso contrario que muestre el mensaje "La operación finalizo".

En el navegador veríamos lo siguiente:



El bucle **foreach** es muy útil para recorrer matrices cuyo tamaño se desconoce o matrices cuyos índices no son correlativos o numéricos (matrices asociativas).

El constructor **foreach** funciona sólo sobre arrays y objetos, y emitirá un error al intentar usarlo con una variable de un tipo diferente de datos o una variable no inicializada.

```
<?php
    foreach (expresión_array as $valor)
        sentencia 1;
        sentencia 2;

    foreach (expresión_array as $clave => $valor)
        sentencia 1;
        sentencia 2;
?>
```

La primera forma recorre el array dado por expresión_array. En cada iteración, el valor del elemento actual se asigna a $valor y el puntero interno del array avanza una posición (así en la próxima iteración se estará observando el siguiente elemento).

La segunda forma además asigna la clave del elemento actual a la variable $clave en cada iteración.

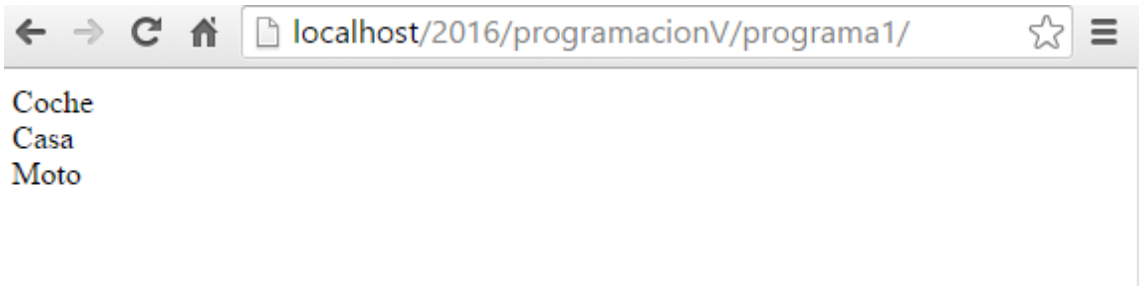
Ejemplo práctico:

```
<?php
    $matriz = array ("Coche", "Casa", "Moto");

    foreach ($matriz as $valor){
        echo $valor."<br>";
    }
?>
```

Recoge cada índice y lo introduce en la variable valor.

Así lo veríamos en nuestro navegador:

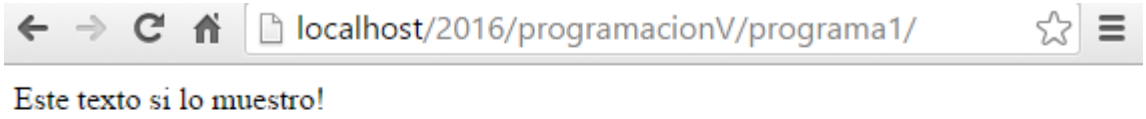


Goto

El operador **goto** puede ser usado para saltar a otra sección en el programa. El punto de destino es especificado mediante una etiqueta seguida de dos puntos y la instrucción es dada como **goto** seguida de la etiqueta del destino deseado. La etiqueta de destino debe estar dentro del mismo fichero y contexto, lo que significa que no se puede saltar fuera de una función o método.

Ejemplo práctico:

```
<?php
    goto marca;
        echo "este texto me lo salto…";
    marca:
        echo "este texto si lo maestro!";
?>
```



Actividades

1. Instalar un servidor web que tenga Apache, MySQL y PHP, de acuerdo a su sistema operativo.
2. Instalar en su sistema operativo un editor de código acorde al mismo.
3. Realizar un script en PHP que contenga el siguiente código:

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

4. Crear un directorio dentro de la carpeta **htdocs** de su servidor llamado “ejercicio1” e incluir el archivo del punto 3 en el mismo; y ejecutar en el navegador.
5. Leer toda la información que se detalla al ejecutar el punto 4 (módulos instalados en nuestro servidor).
6. Realizar un script que tenga una variable del tipo **string** con un mensaje al usuario que le dé un mensaje de bienvenida y mostrarla utilizando el comando echo.
7. Mostrar al usuario el siguiente mensaje utilizando el comando **echo**, *Hola a todos soy estudiante de “sistemas”*.

8. Realizar un programa que ingresando un número cualquiera, almacenado en una variable, me diga si es par o impar.
9. Realizar un programa que ingresando tres lados de un triángulo, almacenados en tres variables distintas, me diga qué tipo de triangulo es (equilátero, isósceles o escaleno).
10. Realizar la tabla de multiplicar del número 7, del 1 al 10, y mostrar los resultados por pantalla.
11. Utilizando el operador **goto**, realizar una suma de dos variables, y una multiplicación, saltarse la suma y mostrar en pantalla el resultado de la multiplicación.
12. Realizar la suma de los 100 primeros números naturales, y mostrar el resultado en pantalla.

Autoevaluación

1. ¿Qué necesitamos para comenzar a programar en PHP o algún lenguaje web?
2. ¿Qué significa MAMP?
3. ¿Qué es un lenguaje de scripting?
4. ¿Hace falta declarar el tipo de variables en PHP?
5. ¿Qué diferencia hay entre un ciclo while y do-while?