Índice

3. Programación de código embebido en linguaxes de marcas 2

3.1 Estruturas de control 2

3.1.1. Estruturas de decisión 2

3.1.2. Estruturas repetitivas 5

3. Programación de código embebido en linguaxes de marcas

3.1 Estruturas de control

Todo script PHP está construído en case a unha serie de sentencias ou instrucións. A orde en que se executarán as instrucións ven definido polo control de fluxo. De non existir as sentencias de control de fluxo os scripts executaríanse de forma secuencial, é dicir, empezarían pola primeira instrución e irían unha por unha ata a chegar á última.

As estruturas de control de fluxo poden ser de dous tipos:

- Estruturas de **decisión**.
- Estruturas **repetitivas**.

Ambos tipos de estruturas estudaranse nos seguintes apartados.

3.1.1. Estruturas de decisión

Permiten **seleccionar entre varios camiños** por onde seguirá a execución do programa. Con frecuencia é necesario realizar diferentes accións segundo se cumpran ou non determinadas condicións. En PHP existen os seguintes tipos de sentencias condicionais:

- **Sentencia if**: executa un bloque de código soamente se unha condición é verdadeira.

Sintaxe da sentencia if

```
if (condición)
 {
 Bloque de código que se executará se a condición é verdadeira
 }
```

O seguinte exemplo mostra "Ten un bo día" se a hora actual do sistema en anterior ás 20 horas.

Exemplo da sentencia if

```
<?php
$t=date("H");
if ($t<"20")
 {
 echo "Ten un bo día!";
 }
?>
```

- **Sentencia if…else**: executa un bloque de código si unha condición é verdadeira, e executa outro bloque de código se a condición é falsa.

Sintaxe da sentencia if...else

```
if (condición)
{
 Bloque de código que se executará se a condición é verdadeira
}
else
{
 Bloque de código que se executará se a condición é falsa
}
```

O seguinte exemplo mostra "Ten un bo día!" se a hora actual do sistema en anterior ás 20 horas, e mostra "Ten unha boa noite!" en caso contrario.

Exemplo da sentencia if...else

```
<?php
$t=date("H");
if ($t<"20")
 {
 echo "Ten un bo día!";
 }
else
 {
 echo "Ten unha boa noite!";
 }
?>
```

- **Sentencia if…else if…else**: selecciona un ou varios bloques de código para ser executados.

Sintaxe da sentencia if...else if...else

```
if (condition)
 {
 Bloque de código que se executará se a condición é verdadeira
 }
else if (condition)
 {
 Bloque de código que se executará se a condición é verdadeira
 }
else
 {
 Bloque de código que se executará se a condición é falsa
 }
```

O seguinte exemplo mostra "Ten un bo día!" se a hora actual do sistema en anterior ás 10 horas, e a mensaxe "Te unha boa xornada!" se a hora é anterior ás 20 horas. En outra caso mostra "Ten unha boa noite!".

Exemplo da sentencia if...else if...else

```
<?php
$t=date("H");
if ($t<"10")
 {
 echo "Ten un bo día!";
 }
else if ($t<"20")
 {
 echo "Ten unha boa xornada!";
 }
else
 {
 echo "Ten unha boa noite!";
 }
?>
```

- **Sentencia elseif:** Como o seu nome suxire, é unha combinación de if e else. Como else, estende unha sentenza if para executar unha sentenza diferente no caso de que a expresión if orixinal avalíase como FALSE. No entanto, a diferenza de else, executará esa expresión alternativa soamente se a expresión condicional elseif avalíase como TRUE. Por exemplo, o seguinte código mostraría "a é maior que b", "a é igual a b" ou "a é menor que b":

```
<?php
 if ($a > $b) {
     print "a é maior que b";
 } elseif ($a == $b) {
     print "a é igual que b";
 } else {
```

```
        print "a é maior que b";
    }

    ?>
```

Pode haber varios *elseifs* dentro da mesma sentenza *if*. A primeira expresión *elseif* (se hai algunha) que se avalíe como TRUE executaríase. En PHP, tamén se pode escribir *'else if'* (con dúas palabras) e o comportamento sería idéntico ao dun *'elseif'* (unha soa palabra). O significado sintáctico é lixeiramente distinto (se estas familiarizado con C, é o mesmo comportamento) pero a liña básica é que ambos resultarían ter exactamente o mesmo comportamento.

- **Sentencia switch**: selecciona un entre varios bloques de código para ser executado.

Sintaxe da sentencia switch

```
switch (n)
{
case etiqueta1:
  código que se executará se n é igual á etiqueta1
  break;
case etiqueta2:
  código que se executará se n é igual á etiqueta2
  break;
default:
  código que se executará se n non coincide nin coa etiqueta1 nin coa etiqueta2
}
```

Neste caso, n é unha expresión (habitualmente unha variable), que se avalía unha vez. O valor da expresión n compárase cos valores de cada caso representados polas etiquetas. Se o valor coincide, executarase o bloque de código asociado ao case correspondente. A sentencia break evita que se continúe executando o código do seguinte case. A sentencia default úsase para o caso en que non hai coincidencia de valores.

O seguinte exemplo mostra unha mensaxe indicando a cor favorita se é unha das tres: vermello, azul ou verde. Noutro caso indica que non é ningunha delas.

Exemplo da sentencia switch

```
<?php
$favcolor="vermello";
switch ($favcolor)
{
case "vermello":
  echo "O teu color favorito é o vermello!";
  break;
case "azul":
  echo "O teu color favorito é o azul!";
  break;
case "verde":
  echo "O teu color favorito é o verde!";
  break;
default:
  echo "O teu color favorito non é nin vermello, nin azul, nin verde!";
}
?>
```

Exercicios.

3.1.2. Estruturas repetitivas

Frecuentemente, cando se está a programar deséxase que un determinado **bloque de código se execute varias veces**. En troques de engadir varias veces un bloque de liñas de código exactamente iguais consecutivamente, utilízanse los bucles para realizar tal tarefa. Polo tanto, as estruturas repetitivas executan un bloque de código un número de veces determinado, ou mentres se cumpra unha determinada condición.

En PHP existen as seguintes **sentencias repetitivas**:

- **while**: repite un bloque de código mentres unha condición especificada sexa verdadeira.

Sintaxe da sentencia while

```
while (condición)
 {
 Código que se executará
 }
```

No seguinte exemplo asígnaselle á variable $i o valor 1, e a continuación executarase un bucle while de maneira que en cada iteración se incrementará a variable $i unha unidade todo isto mentres $i teña un valor menor ou igual a 5. En cada iteración do bucle mostrarase unha frase que indica o valor de $i.

Exemplo da sentencia while

```
<?php
$i=1;
while($i<=5)
 {
 echo "O número é " . $i . "<br>";
 $i++;
 }
?>
```

O resultado da execución do anterior exemplo é:

Resultado da execución do exemplo anterior

```
O número é 1
O número é 2
O número é 3
O número é 4
O número é 5
```

- **do…while**: executa un bloque de código unha vez, e o repite mentres unha condición determinada sexa verdadeira.

Sintaxe da sentencia do…while

```
do
 {
 Bloque de código que se executará
 }
while (condition);
```

No seguinte exemplo iníciase á variable $i co valor 1, e a continuación executarase un bucle do…while de maneira que a primeira iteración executarase sempre. En cada unha das

iteracións incrementarase a variable $i unha unidade e se mostrara unha frase indicando o valor de $i, todo isto mentres $i teña un valor menor ou igual a 5.

Exemplo da sentencia do…while

```
<?php
$i=1;
do
 {
 $i++;
 echo "O número é " . $i . "<br>";
 }
while ($i<=5);
?>
```

O resultado da execución do anterior exemplo é:

Resultado da execución do exemplo anterior

```
O número é 2
O número é 3
O número é 4
O número é 5
O número é 6
```

- **for**: executa un bloque de código un número determinado de veces.

Sintaxe da sentencia for

```
for (inicio; condición; incremento)
 {
 Código que se executará
 }
```

O parámetro inicio é habitualmente inicia un contador. A condición avalíase en cada iteración de xeito que, se a condición é verdadeira o bucle continúa. O incremento úsase habitualmente para incrementar o contador.

No seguinte exemplo iníciase o contador a 5, e se incrementa unha unidade en cada iteración. No corpo do bucle imprímese o valor do contador en cada iteración.

Exemplo da sentencia for

```
<?php
for ($i=1; $i<=5; $i++)
 {
 echo "O número é " . $i . "<br>";
 }
?>
```

O resultado da execución do anterior exemplo é:

Resultado da execución do exemplo anterior

```
O número é 1
O número é 2
O número é 3
O número é 4
O número é 5
```

- **foreach**: repite un bloque de código para cada elemento nun array.

Sintaxe da sentencia foreach

```
foreach ($array as $valor)
 {
```

```
Código que será executado
}
```

Para cada iteración, asígnase á variable $valor o valor dun elemento do array $array. Deste xeito, o bucle terminará cando se procesen todos os elementos do array.

No seguinte exemplo imprímense os valores dun array.

Exemplo da sentencia foreach

```
<?php
$x=array("un","dous","tres");
foreach ($x as $valor)
 {
 echo $valor . "<br>";
 }
?>
```

O resultado da execución do anterior exemplo é:

Resultado da execución do exemplo anterior

```
un
dous
tres
```

- **break**

break escapa das estruturas de control iterante (bucle) actuais *for*, *while*, ou *switch*.

break acepta un parámetro opcional, o cal determina cantas estruturas de control hai que escapar. Desde PHP 5.4.0 isto foi eliminado.

```
<?php
$arr = array ('one', 'two', 'three', 'four', 'stop', 'five');
while (list (, $val) = each ($arr)) {
    if ($val == 'stop') {
        break;    /* Poderíase escribir 'break 1;' */
    }
    echo "$val<br>\n";
}
/* Usando o argumento opcional. */
$i = 0;
while (++$i) {
    switch ($i) {
    case 5:
        echo "At 5<br>\n";
        break;  /* Sae do switch. */
    case 10:
        echo "Ao 10; saíndo<br>\n";
        break;  /* Sae do switch. */
    default:
        break;
    }
}
?>
```

- **continue**

continue úsase dentro da estrutura do bucle para saltar o resto da iteración actual do bucle e continuar a execución ao comezo da seguinte iteración.

continue acepta un parámetro opcional, o cal determina cuantos niveis (bluces) hai que saltar antes de continuar coa execución. Desde PHP 5.4.0 isto foi eliminado.

```
<?php
for ($i = 0; $i < 5; ++$i) {
    if ($i == 2){
        continue;
    }
    print "$i\n";
}}
?>
```

Exercicios.