Estructuras de control.

Referencias:

http://www.php.net
http://www.w3schools.com

Bloque de código: Conjunto de instrucciones agrupado entre llaves: { }. Un bloque de una única instrucción no necesita ir entre llaves.

PHP ofrece una sintaxis altenativa para alguna de sus estructuras de control: if, while, for, switch. En cada caso, la forma básica de la sintaxis alternativa es cambiar la llave de apertura { por dos puntos : y la de cierre } por endif;, endwhile;, endfor;, o endswitch;.

```
...
if ($a == 5):
    print "a es igual a 5";
    print "...";
elseif ($a == 6):
    print "a es igual a 6";
    print "!!!";
else:
    print "a no es ni 5 ni 6";
en1dif;
...
```

Estructuras de control de selección.

Con ellas permitimos la ejecución condicional de bloques de código.
Conviene no olvidar como interpreta PHP los diferentes tipos de datos en una expresión lógica.

if (expLogica)
bloque

Ejecuta un bloque de código si la expresión lógica es TRUE

Ejemplo

```
...
if ($a > $b)
    print "a es mayor que b";
...
```

if (expLogica)
bloqueV
else
bloqueF

Ejecuta una sentencia si se cumple una la condición y una sentencia distinta si la condición no se cumple.

Ejemplo:

```
...
if ($a  >  $b) {
    print "a ES mayor que b";
}

else {
    print "a NO ES mayor que b";
}
...
```

```
if (expLogica1)
   bloque1
elseif (expLogica2)
   bloque2
```

Combinación de if y else. Permite ejecutar un bloque de código si la expresión lógica que acompaña a if es FALSE y TRUE la condición que acompaña a elseif.

Se recomienda utilizar else if

```
...
if ($a > $b) {
    print "a es mayor que b";
}
elseif ($a == $b) {
    print "a es igual que b";
}
else {
    print "a es mayor que b";
}
...
```

```
switch (expresion) {
   case valor1:
      bloque
      [break;]
   case valor2:
      bloque
      [break;]
  ...
   case valorN:
      bloque
      [break;]
   [default:
     bloque]
}
```

La sentencia switch es similar a una serie de sentencias if en la misma expresión. En muchas ocasiones, se quiere 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 ello sirve la sentencia switch.

Los siguientes dos ejemplos son dos modos distintos de escribir el mismo algoritmo, uno usa una serie de sentencias if y el otro usa la sentencia switch:

```
...
if ($i == 0) {
    print "i es igual a 0";
}
if ($i == 1) {
    print "i es igual a 1";
}
if ($i == 2) {
    print "i es igual a 2";
}
...
```

```
...
switch ($i) {
    case 0:
        print "i es igual a 0";
        break;
     case 1:
        print "i es igual a 1";
        break;
     case 2:
        print "i es igual a 2";
        break;
}
...
```

La sentencia switch comienza a ejecutar las sentencias cuando se encuentra una cláusula case con un valor que coincide con el valor de la expresión switch**.** PHP ejecutara todas las instrucciones que se encuentre hasta el final del bloque, o hasta encontrar la cláusua break**.** Si no se escribe una sentencia break al final de una lista de sentencias case, PHP seguirá ejecutando las sentencias del siguiente case. Por ejemplo:

```
switch ($i) {
    case 0:
        print "i es igual a 0";
    case 1:
        print "i es igual a 1";
     case 2:
        print "i es igual a 2";
}
```

Ojo!!!! Aquí sí $i es igual a 0, PHP ejecutará todas las sentencias print. Si $i es igual a 1, PHP ejecutará las últimas dos sentencias print y sólo si $i es igual a 2, se obtendría la conducta esperada y solamente se mostraría ’i es igual a 2’.

Es importante no olvidar las sentencias break.

En una sentencia switch, la condición se evalúa sólo una vez y el resultado se compara con el valor de cada cláusula case. En una sentencia elseif, la condición se evalúa otra vez. Si tu condición es más complicada que una comparación simple, un switch puede ser más rápido.

Los bloques de código de las cláusulas case pueden estar vacíos, en cuyo caso el flujo del programa continúa en el bloque de código del siguiente case.

```
...
switch ($i) {
    case 0:
    case 1:
    case 2:
        print 'i es menor que 3, pero no negativo';
        break;
    case 3:
        print 'i es 3';
}
...
```

Cláusula default. Se trata de un caso especial de case cuyo bloque de código se ejecutará cuando el valor de la expresión del switch no coincida con ninguno de los valores que acompañan a las otras cláusulas case.

Ejemplo:

```
...
switch ($i) {
    case 0:
        print "i es igual a 0";
        break;
    case 1:
        print "i es igual a 1"
        break;
    case 2:
        print "i es igual a 2";
        break;
    default:
        print "i no es igual a 0, 1 o 2";
}
...
```

Estandar de codificación:

- Las sentencias de control basadas en if y elseif deben tener un único espacio en blanco antes del paréntesis inicial y un único espacio en blanco después del paréntesis final.
- Los operadores dentro de la condición deben separarse con espacios. Es recomendable usar paréntesis internos para mejorar la agrupación de las expresiones condicionales largas.
- La llave de inicio { debe estar en la misma línea que la condición, mientras que la de cierre } debe estar sola. El código dentro de las llaves debe estar indentado:

```
    ...
    if ($a != 2) {
        $a = 2;
    }
    ...
```

- Si el largo de la condición es mayor al largo máximo de la línea, y la condición tiene varias cláusulas, se puede partir en varias líneas:

```
...
if (($a == $b)
   && ($b == $c)
    || (Foo::CONST == $d)
) {
 $a = $d;
}
...
```

- Toda sentencia if, elseif o else debe usar llaves de apertura { y cierre } sin excepción.
- Se recomienda utilizar else if en vez de elseif.
- El código dentro de un bloque switch y dentro de cada case debe estar indentado:

```
...
switch ($respuestaPregunta) {
    case 1:
        break;
    case 2:
        break;
    default:
        break;
}
...
```

- Nunca se debe omitir la cláusula default en una instrucción switch.
- Si se omite intencionalmente un break dentro de un case, debe escribirse un comentario justificando el motivo.

Trabajo de clase

1.- Cargar en dos números en variables y escribir el mayor de ellos.

2.- Cargar en variables mes y año e indicar el número de días del mes.

3.- Cargar fecha de nacimiento en una variable y calcular la edad.

4.- Cabecera en función de la estación del año.

5.- Lista de enlaces en función del perfil de usuario.

Estructuras de control repetitivas.

while (expLogica)
  bloque

Ejecuta el bloque de código mientras la expresión lógica devuelva como resultado TRUE.
Conviene recordar como trata PHP los tipos de datos cuando aparecen en expresiones lógicas.
En función del valor inicial de la expresión lógica, puede darse el caso de que el bloque de código no se ejecute nunca.
Es necesario crear en el bloque las condiciones necesarias para garantizar que en algún momento la expresión lógica se evalúe como FALSE y evitar bucles infinitos.
También es posible utilizar la sintaxis alternativa:

while (expLogica):
  sentencia1;
  …
  sentenciaN;
endwhile;

Ejemplos:

```
...
   $i = 1;
   while ($i <= 10) {
      print $i++;
   }
...
```

```
...
$i = 1;
while ($i <= 10):
   print $i;
   $i++;
endwhile;
...
```

do
  sentencia1;
  …
  sentenciaN;
while (expLogica);

Los bucles do..while son muy similares a los bucles while, excepto que 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 los bucles do..while garantizan la ejecución de la primera iteración.
Es necesario garantizar que la expLogica llegue a evaluarse como FALSE para evitar bucles infinitos.
El bucle do..while no admite sintaxis alternativa.

```
..
$i = 0;
do {
    print $i;
}
while ($i>0);
...
```

El bucle de arriba se ejecutaría exactamente una sola vez, después de la primera iteración la condición se evalúa como FALSE y la ejecución del bucle finaliza.

for (expr1; expr2; expr3)
bloque

La primera expresión (expr1) se evalúa (ejecuta) incondicionalmente una vez al principio del bucle. Al comienzo de cada iteración se evalúa expr2 . Si se evalúa como TRUE, el bucle continúa y se ejecuta el bloque de código. Si se evalúa como FALSE la ejecución del bucle finaliza. Al final de cada iteración, se evalúa (ejecuta) expr3.
Cada una de las expresiones puede estar vacía. Que expr2 esté vacía significa que el bucle debería repetirse indefinidamente (PHP implicitamente lo considera como TRUE).

Ejemplos:

```
...
for ($i = 1; $i <= 10; $i++) {
    print $i;
}
...
```

```
...
for ($i = 1; $i <= 10; print $i, $i++) ;
...
```

PHP también soporta la "sintaxis de dos puntos" alternativa para bucles for.

```
for (expr1; expr2; expr3):
    sentencia1;
     …
    sentenciaN;
endfor;
```

foreach

Se trata de un bucle especial que estudiaremos con arrays asociativos.

Estandar de codificación:

Utilizamos los mismos criterios que en las estructuras condicionales.

Trabajo de clase.

1.- Escribir los números 1 al 10

2.- Sumar los 3 primeros números pares.

3.- Tablar de multiplicar 1 al 10

4.- Mostrar paleta de colores.

5.- Dado un mes y un año, mostrar el calendario del mes. Marcar el día actual en verde y los festivos en rojo.