Tema 2. ESTRUCTURES DE CONTROL

Les estructures de control que podem trobar a PHP són comuns en quant al concepte a la majoria de llenguatges de programació d'alt nivell i quasi idèntiques a les que presenten llenguatges com ara C, C++, Java, Perl o Bash.

2.1. SENTÈNCIES CONDICIONALS

Són les estructures de control més senzilles, es basen en l'ús de la sentència **if...else** i en les diferents formes que pot presentar. Utilitzant aquestes sentències podrem fer que programa escolleixi entre dos camins d'execució diferent en funció de l'avaluació d'una expressió lògica.

2.1.1. if

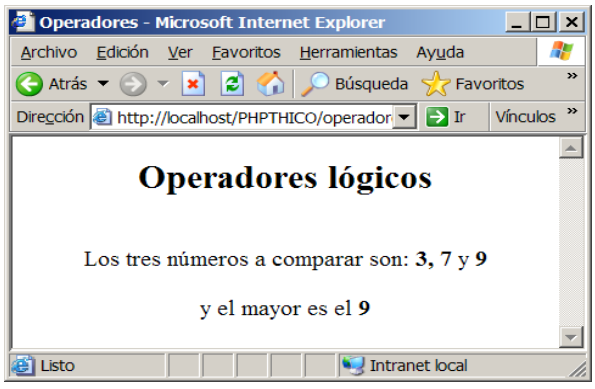
La sintaxis és:

```
if (condició){
        sentències;
}
```

L'intèrpret de PHP avalua la *condició*, que ha de ser una expressió lògica i, si resulta certa s'executaran les *sentències* que es trobin entre { } i, si és falsa, PHP ignorarà les sentències i continuarà amb l'execució normal del programa.

Exercici 1

Donats 3 números, indicar quin és el més gran utilitzant la estructura d'aquest apartat.



Les sentències if es poden nidar, és a dir, podem posar dins d'un bloc if altres sentències if.

2.1.2. if...else

Sovint interessa executar un codi diferent si l'avaluació de la expressió que acompanya a la instrucció if no és certa. Llavors s'utilitza la sentència **if...else**; consta d'un bloc **if** que s'executa quan l'expressió s'avalua a cert i d'un bloc **else** que executa les seves instruccions quan s'avalua a fals.

```
if (expressió){
        sentències;
} else {
        sentències;
}
```

Exercici 2

A partir de l'exercici 1 fes les modificacions pertinents utilitzant l'estructura d'aquest apartat.

2.1.2. if...elseif

Hi ha moltes ocasions en que es vol avaluar una nova comprovació utilitzant una sentència **if** dins del cos d'una sentència **else**; per a aquests casos es pot utilitzar la sentència **elseif** que permet combinar ambdues sentències en una sola:

```
if (expressió){
        sentències;
} elseif (expressió) {
        sentències;
} else {
        sentències;
}
```

Exercici 3

A partir de l'exercici 2 fes les modificacions pertinents utilitzant l'estructura d'aquest apartat.

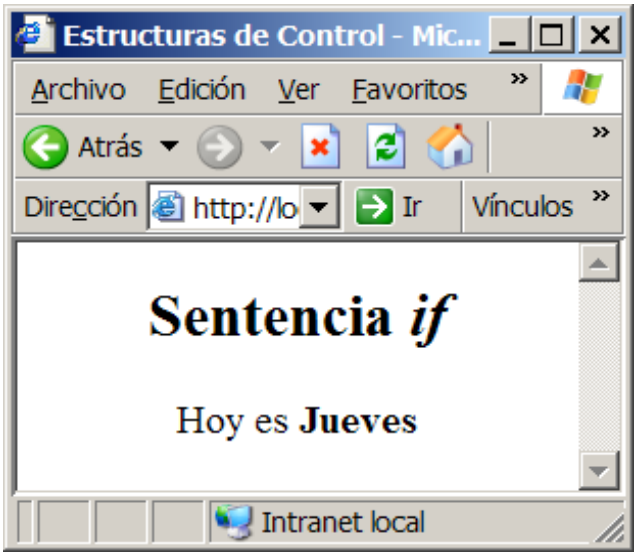
PHP ofereix una sintaxis alternativa per algunes de les seves estructures de control. La forma bàsica de la sintaxis alternativa per a la instrucció **if** és canviar el caràcter **{** pel caràcter **:** i el caràcter **}** per la paraula **endif**, quedant de la següent forma:

```
if (expressió):
        sentències;
elseif (expressió):
        sentències;
else
        sentències;
endif
```

La sintaxis de la sentència **if-elseif-else** pot tenir tants components **elseif** com siguin necessaris. Aquest tipus de composició permet fer processos de selecció múltiple en una sola estructura.

```
<HTML>
 <HEAD>
  <TITLE>Estructuras de Control</TITLE>
 </HEAD>
 <BODY>
  <CENTER>
   <H2>Sentencia <I>if</I></H2>
   <?php
     echo "Hoy es <B>";
     $dia = date("D");

       if ($dia == "Mon") {
            echo "Lunes";
          } elseif ($dia == "Tue") {
              echo "Martes";
            } elseif ($dia == "Wed") {
                echo "Miercoles";
              } elseif ($dia == "Thu") {
                  echo "Jueves";
                } elseif ($dia == "Fri") {
                    echo "Viernes";
                  } elseif ($dia == "Sat") {
                      echo "Sabado";
                    } else {
                        echo "Domingo";
                    }
     echo "</B>";
   ?>
  </CENTER>
 </BODY>
</HTML>
```



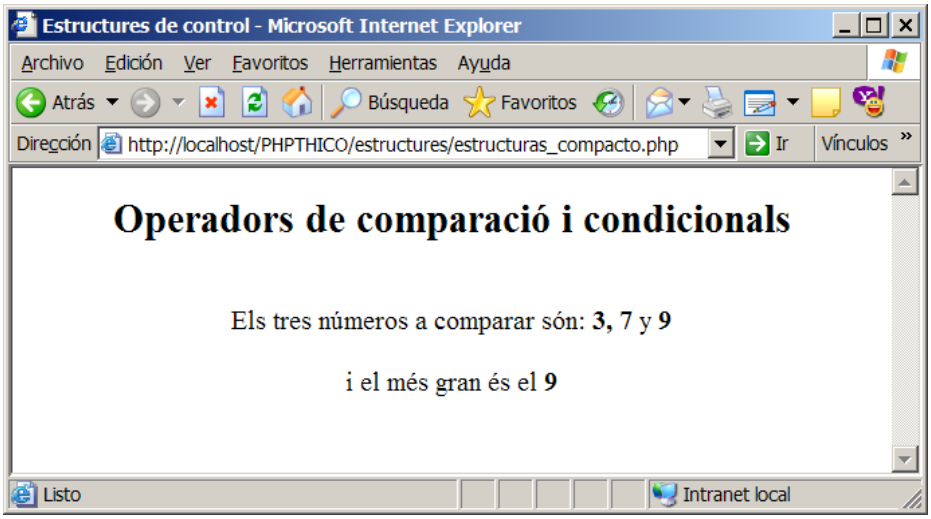
2.1.4. Expressió condicional (if compacte)

PHP utilitza dos operadors (**?** i **:**) per a formar expressions condicionals que retornaran un de dos possibles valors bassant-se en el valor lògic obtingut en avaluar l'expressió. La sintaxis d'aquests operadors és:

<expressió1> ? <expressió2> : <expressió3>;

D'aquesta forma, si **expressió1** s'avalua a *true*, llavors es retorna el valor d'**expressió2**, en cas contrari, es retorna el resultat d'**expressió3**. Podem veure un exemple en el següent codi:

```
<HTML>
 <HEAD>
  <TITLE>Estructures de control</TITLE>
 </HEAD>
 <BODY>
  <CENTER>
   <H2>Operadors de comparació i condicionals</H2>
   <?php
     $a=3;
     $b=7;
     $c=9;
     echo "<BR>Els tres números a comparar són: ";
     echo "<B>$a, $b </B>y<B> $c</B><BR><BR>";
     echo " i el més gran és el <B>";
     //Obtenim el més gran de $a i $b
     $mesgran = ($a>$b)?$a:$b;
     echo ($mesgran<$c)?$c:$mesgran;
     echo "</B>";
   ?>
  </CENTER>
 </BODY>
</HTML>
```



2.1.5. switch

La sentència **switch** s'utilitza per a comparar una dada amb un conjunt de possibles valors. Aquesta tasca es pot realitzar utilitzant múltiples sentències **if** o amb una sentència **if...elseif** múltiple, però la sentència **switch** és molt més llegible i permet especificar un conjunt de sentències per defecte, en el cas que la dada no tingui un valor amb el qual comparar-lo (equivalent a la clàusula **else** de la sentència **if**).

```
Switch ($variable) {
   case valor1:     sentències;
                    break;

   case valor2:     sentències;
                    break;

   case valorN:     sentències;
                    break;

  [default:         sentències;]
}
```

La sentència **break** dins de cadascuna de les clàusules **case** permet que una vegada trobat el valor cercat, no es continuin realitzant les comparacions; força la finalització de la sentència **switch**. Si no s'utilitza aquesta sentència l'execució d'una clàusula continua per la següent.

Exercici 4

Implementa l'exemple dels dies de la setmana de l'apartat 11.1.2. mitjançant la clàusula **switch**.

Existeixen ocasions on és interesant prescindir de la utilització de la sentència **break** dins d'una clàusula **case**. També és habitual que alguna de les clàusules **case** no tingui associada cap sentència, ocasionant que el control passi a la clàusula **case** immediatament posterior.

Exercici 5

Modifica l'exercici 4 per a que només indiqui si el dia en el que estem és festiu o de diari.

2.2. SENTÈNCIES DE BUCLES

La utilització de bulces dins d'un script serveix per a molts propòsits, com ara comptar. També s'utilitzen per a recòrrer objectes o estructures compostes per més d'un element, com ara les estructures de tipus *array*.

2.2.1. for

Aquesta sentència permet realitzar un conjunt d'instruccions un determinat nombre de vegades. La seva sintaxis és:

```
for ([exp_inicialització] ; [exp_condició] ; [exp_bucle]){
        sentències;
}
```

Les 3 expressions tancades entre parèntesis són opcionals, però és necessari escriure els **;** que les separen encara que les expressions no s'indiquin, per a que cada expressió es trobi al seu lloc.

Les opcions que permet aquesta sentència són:

- **exp_inicialització:** normalment s'utilitza per a inicialitzar i declarar la variable o variables que s'han d'utilitzar com a comptadors del bucle; només s'executa una vegada a l'inici del bucle. Si hi ha més d'una variable, es separen per comes. Sol ser de la forma: **$variable = valor_inicial**

- **exp_condició:** defineix una o més condicions que han de complir-se (avaluar-se a *true*) per a poder executar les sentències incloses al for. Mentre aquestes condicions siguin certes s'aniran executant les sentències. L'expressió s'avalua abans de cada iteració i, si no es compleix la condició, ja no continuarà l'execució. Sol ser de la forma: **$variable <= límit**

- **exp_bucle:** modifica el valor de la variable o variables (separades per comes) utilitzades com a comptadors del bucle. S'executa cada vegada que finalitza una iteració. No segueix un patró fixe, algunes de les formes que presenta són:

  $variable++
  $variable--
  $variable+=valor ...

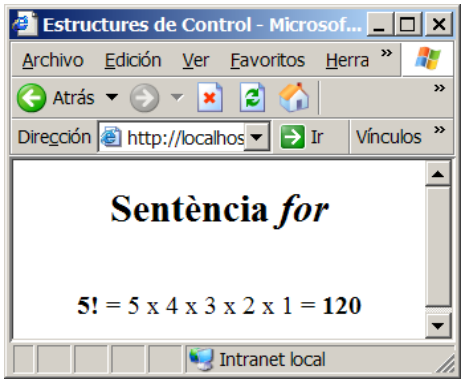
Exercici 6

Realitza un bucle per sumar els 10 primers números.

D'igual forma que el comptador pot anar incrementant-se en cada interacció del bucle, també pot anar decrementant-se.

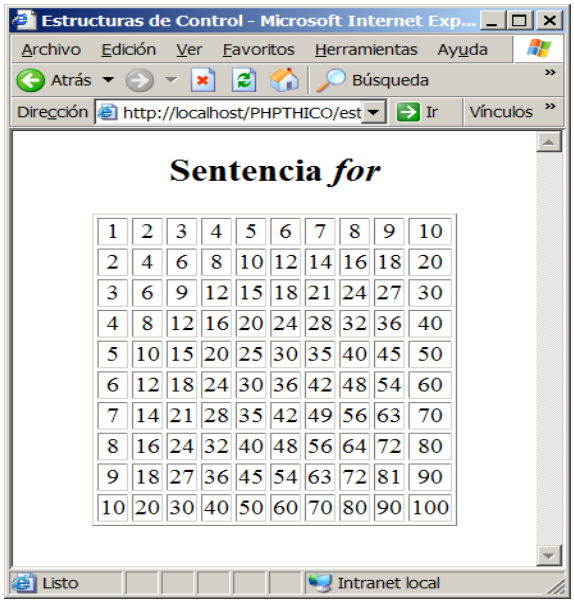
Exercici 7

Realitzar un bucle per calcular el factorial d'un nombre donat.



Exercici 8

Realitzar un programa que generi una taula de multiplicar de 10x10 elements:



2.2.2. foreach

Aquesta sentència ens permet recòrrer les estructures de tipus *array* d'una forma senzilla, obtenint en cada pas de la iteració un dels seus components. Té dos sintaxis. La primera és:

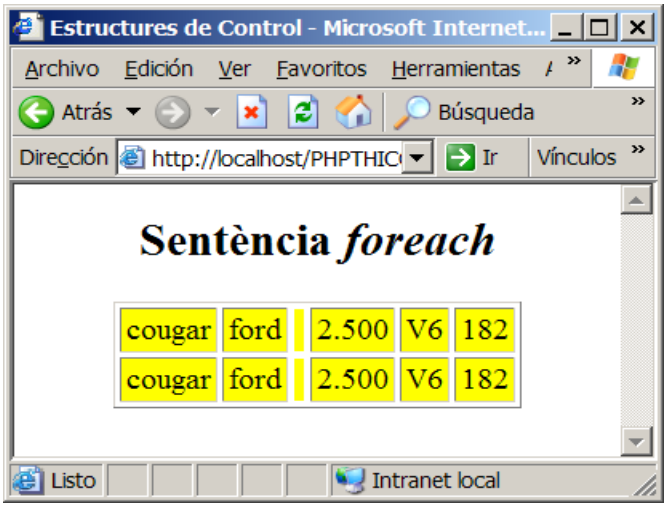
```
foreach (nom_array as $variable){
        sentències;
}
```

El que fa aquest bucle és recòrrer cadascun dels elements de l'array que té per nom **nom_array**, assignant a cada valor de l'element de l'array el contingut de la variable **$variable**. El bucle fa ús d'un punter intern que apunta a la posició actual de l'array (començant per la primera i continuant en ordre ascendent) i que va actualitzant de forma automàtica amb cadascuna de les iteracions.

A l'exemple següent podem veure dos formes de recòrrer un array. A la primera es fa mitjançant un bucle **for** i a la segona mitjançant un bucle **foreach**. En el segon cas no cal conèixer la quantitat d'elements que conformen l'array ja que la sentència té un punter intern per a recòrre'l.

```
<HTML>
 <HEAD>
  <TITLE>Estructures de Control</TITLE>
 </HEAD>
 <BODY>
  <CENTER>
   <H2>Sentència <I>foreach</I></H2>
   <?php
    // creem l'array i omplim de dades
    $matriz[0]="cougar";
    $matriz[1]="ford";
    // posició sense contingut
    $matriz[2]=null;
    $matriz[3]="2.500";
    $matriz[4]="V6";
    $matriz[5]=182;
   ?>
   <TABLE BORDER="1" CELLPADDING="2" CELLSPACING="2">
     <TR ALIGN="center" BGCOLOR="yellow">
       <?php
        for($i=0;$i<6;$i++){
           echo "<TD> $matriz[$i] </TD>";
        }
       ?>
     </TR>
     <TR ALIGN="center" BGCOLOR="yellow">
       <?php
        foreach($matriz as $valor){
           echo "<TD> $valor </TD>";
        }
       ?>
     </TR>
```

```
    </TABLE>
  </CENTER>
 </BODY>
</HTML>
```



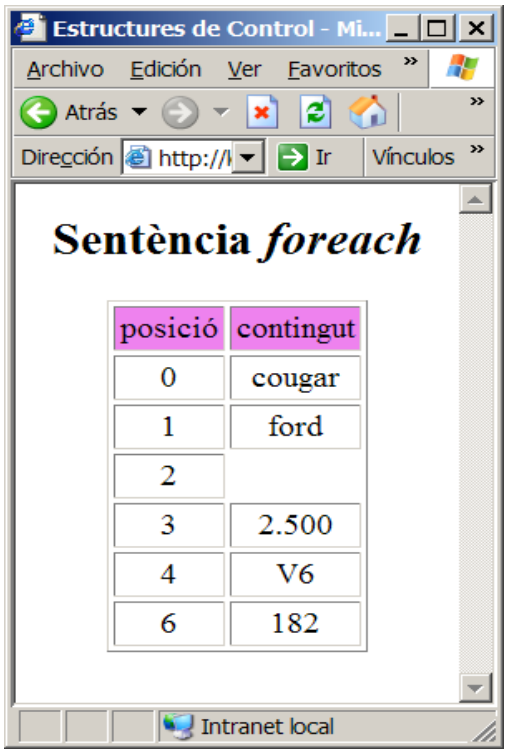
Ambdues sentències mostres d'igual forma el contingut de l'array.

La segona sintaxis que ofereix la sentència **foreach** és:

```
foreach (nom_array as $clau => $valor){
        sentències;
}
```

Amb aquesta sintaxis podem conèixer en tot moment l'índex de la posició exacta a on es troba l'array, mitjançant la variable **$clau**. No cal inicialitzar el punter intern a la primera posició ja que es fa automàticament.

Un exemple d'ús d'aquesta sintaxis seria:



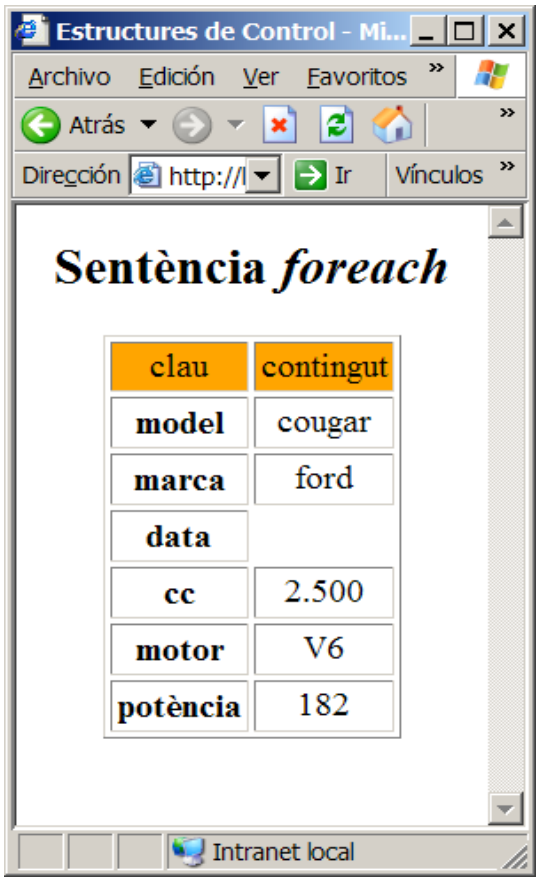

```
<HTML>
 <HEAD>
  <TITLE>Estructures de Control</TITLE>
 </HEAD>
 <BODY>
  <CENTER>
   <H2>Sentència <I>foreach</I></H2>
   <?php
    // creem l'array i omplim de dades
    $matriz[0]="cougar";
    $matriz[1]="ford";
    // posició sense contingut
    $matriz[2]=null;
    $matriz[3]="2.500";
    $matriz[4]="V6";
    // saltem una posició
    $matriz[6]=182;
   ?>
   <TABLE BORDER="1" CELLPADDING="2" CELLSPACING="2">
     <TR ALIGN="center" BGCOLOR="violet">
       <TD>posició</TD>
       <TD>contingut</TD>
     </TR>
        <?php
          foreach($matriz as $key => $valor){
             echo "<TR ALIGN='center'>";
             echo "<TD> $key </TD>";
             echo "<TD> $valor </TD>";
             echo "</TR>";
          }
        ?>
     </TR>
   </TABLE>
  </CENTER>
 </BODY>
</HTML>
```

S'ha d'observar que una posició que estigui buida (la 2) el recorregut no la salta, en canvi, si la posició no s'utilitza (la 5) no es té en compte en aquest tipus de recorregut.

S'ha de dir que aquesta sentència es pot aplicar també a un array de tipus **associatiu**, on l'índex de cada element no és de tipus numèric. Podem veure un exemple a continuació:

```
<HTML>
 <HEAD>
  <TITLE>Estructures de Control</TITLE>
 </HEAD>
 <BODY>
  <CENTER>
   <H2>Sentència <I>foreach</I></H2>
   <?php
    // creem la matriu associativa
    $matriz['model']="cougar";
    $matriz['marca']="ford";
    // posició sense contingut
    $matriz['data']=null;
    $matriz['cc']="2.500";
    $matriz['motor']="V6";
    $matriz['potència']=182;
   ?>
   <TABLE BORDER="1" CELLPADDING="2" CELLSPACING="2">
    <TR ALIGN="center" BGCOLOR="orange">
     <TD>clau</TD>
     <TD>contingut</TD>
    </TR>
      <?php
       foreach($matriz as $key => $valor){
          echo "<TR ALIGN='center'>";
          echo "<TD><B> $key </B></TD>";
          echo "<TD> $valor </TD>";
          echo "</TR>";
       }
      ?>

    </TR>
   </TABLE>
  </CENTER>
 </BODY>
</HTML>
```



2.2.3. while

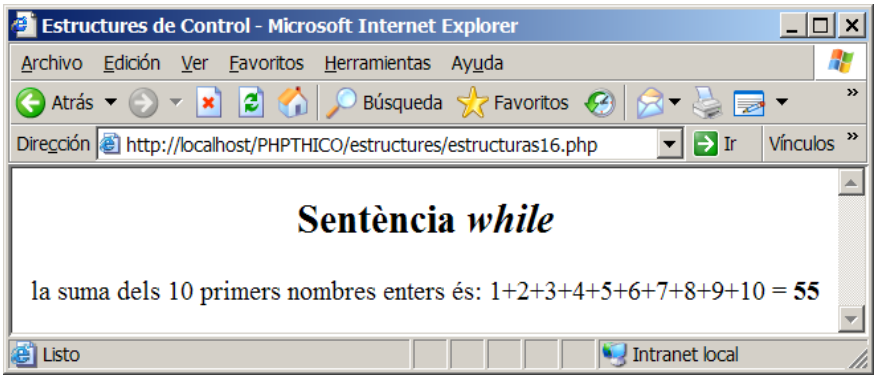
La sentència **while** funciona de forma molt semblant a la sentència **for**, però la diferència és que no inclou en la seva declaració ni la inicialització de la variable de control del bucle ni el seu decrement o increment. Per tant, aquesta variable s'haurà de declarar abans del bucle **while** i el seu increment/decrement s'haurà de realitzar dins del cos del bucle. La seva sintaxis és:

```
while (condició){
        sentències;
}
```

Amb aquesta instrucció es podran executar un conjunt d'instruccions un determinat nombre de vegades sempre i quan el resultat de comprovar la *condició* sigui certa. Si la *condició* s'avalua a *true*, s'executen les sentències del cos del bucle; després d'executar-les, es tornarà a avaluar la *condició*, de forma que, si es continua complint es tornaran a executar les sentències. Això es repeteix fins que la *condició* que s'avalua és *false* i llavors l'execució continua per la instrucció següent a la sentència **while**.

Exercici 9

Realitza la suma dels 10 primers nombres amb la sentència **while**.



2.2.4. do...while

Aquesta sentència funciona exactament igual que el bucle **while** excepte que la condició no es comprova fins que s'ha realitzat una iteració (la *condició* es comprova al final de cada iteració). Això garanteix que, com a mínim, el cos del bucle es realitza una vegada, encara que l'expressió *condició* s'avalui a *false*. La seva sintaxis és:

```
do {
        sentències;
} while (condició);
```

En el següent exemple podem veure el seu ús bàsic per a calcular el factorial d'un nombre donat:

```
<HTML>
 <HEAD>
  <TITLE>Estructures de Control</TITLE>
 </HEAD>
 <BODY>
  <CENTER>
   <H2>Sentència <I>do...while</I></H2>
   <?php
     $numero=6;
     echo "<BR><B>$numero!</B> = ";
     $factorial=1;
     do{
       echo $numero." x ";
       $factorial*=$numero;
         $numero--;
     }while($numero>1);
     echo "1 = <B>$factorial</B>";
   ?>
  </CENTER>
 </BODY>
</HTML>
```