

4. Referència del llenguatge II

Quan comenceu a programar, de seguida us adoneu que amb les sentències no en teniu prou per implementar totes les funcionalitats que voleu que realitzin els vostres scripts, necessiteu quelcom més. Aquest apartat tracta de tot allò que en el llenguatge PHP complementa a les sentències i que ús ajudarà a implementar qualsevol funcionalitat. Ens concret, ens referim a les estructures de control a diversos grups de funcions: integrades de PHP, de cadenes de caràcters, d'arrays i d'usuari.

4.1 Estructures de control

Les estructures de control ens permeten indicar quines accions volem executar en determinades situacions. Aquestes les podem classificar en dos grans grups: les **estructures condicionals** i les **estructures iteratives**; les quals en PHP són comunes a la gran majoria de llenguatges de programació.

4.1.1 Estructures de control condicionals

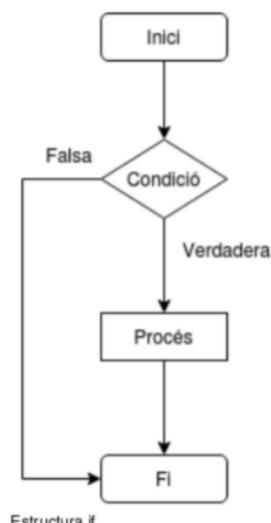
Les estructures de control condicionals ens permeten executar unes accions o altres, depenen si es compleixen determinades condicions; trobem:

- l'estructura if,
- l'estructura else,
- l'estructura elseif i
- l'estructura switch.

Estructura if

Mitjançant l'estructura **if** les accions s'executarán si la condició analitzada és certa, si no, no s'executarán.

En el cas següent la condició a analitzar és el valor que pren la variable **\$departament**. Si **\$departament** és igual a “**RR.HH**”, s'executarà el codi que hi ha entre les claus, en cas contrari, continuarem amb el procés de l'script continuant per la línia de codi que trobarem just després de l'estructura if.



```

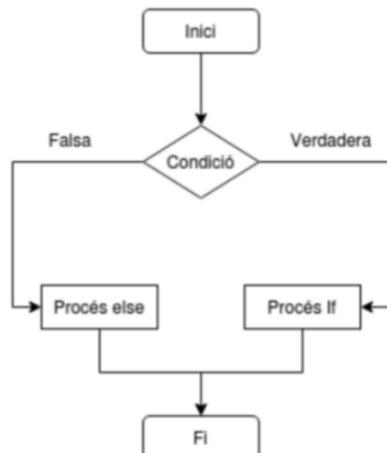
1 <?php
2
3 echo "<h3>ESTRUCTURA IF</h3>"; //Títol secció
4
5 //Declaració variables
6 $departament = "RR.HH";
7
8 //Estructura if
9 if ($departament == "RR.HH") { //Si es compleix la condició aleshores...
10
11   echo "El departament de recursos humans es troba en la 1a planta.<br/>"; //
... es realitza aquesta acció
12
13 }
14
15 //quan sortim de l'estructura if o bé no entrem....
16 echo "Final de l'script.<br/>"; //... es realitza aquesta acció

```

Estructura else

Mitjançant l'estructura **else** les accions s'executaran si la condició analitzada en l'estructura **if** és falsa, si no, com en el cas anterior, s'executaran les accions de l'estructura **if**. En la figura 4.1 podeu veure el diagrama d'aquesta estructura.

FIGURA 4.1. Estructura if - else



En el cas següent, com en l'anterior, la condició a analitzar és el valor que pren la variable **\$departament**. Si **\$departament** és igual a “**RR.HH**”, s'executarà el codi que hi ha entre les claus de l'estructura **if**, en cas contrari, s'executarà el codi que hi ha entre les claus de l'estructura **else**.

```

1 <?php
2
3 echo "<h3>ESTRUCTURA IF ELSE</h3>"; //Títol secció
4
5 //Declaració variables
6 $departament = "RR.HH";
7
8 //Estructura if
9 if ($departament == "RR.HH") { //Si es compleix la condició aleshores...
10
11   echo "El departament de recursos humans es troba en la 1a planta.<br/>"; //
... es realitza aquesta acció
12

```

```

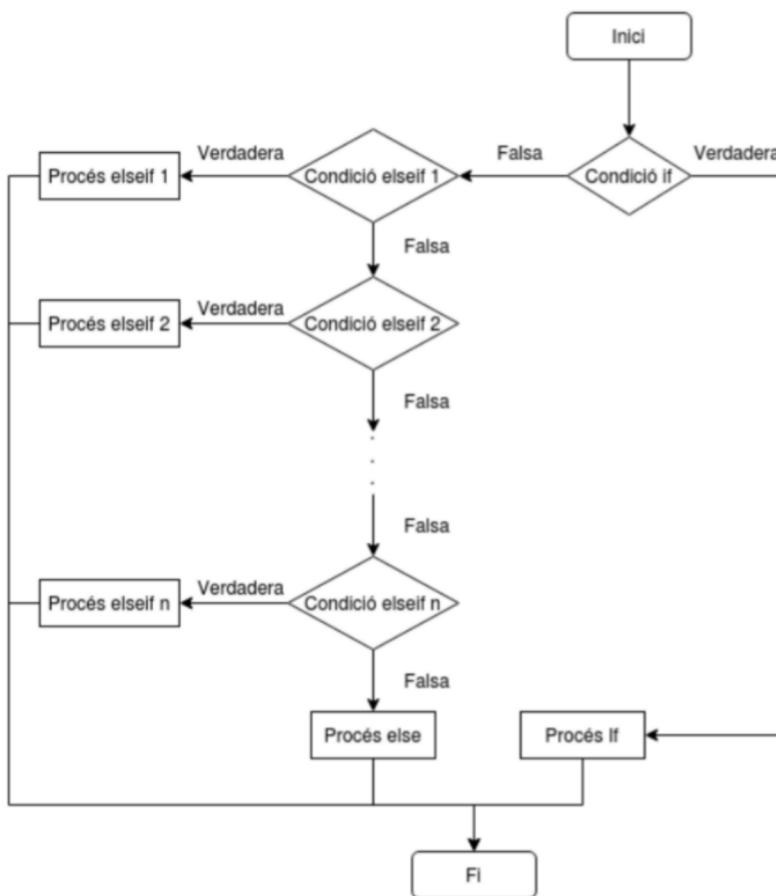
13 }else{ //Si no es compleix la condició aleshores...
14
15     echo "El departament no és el de RR.HH.<br/>"; //... es realitza aquesta acció
16
17 }
18
19 //quan sortim de l'estructura if else...
20 echo "Final de l'script.<br/>"; //... es realitza aquesta acció

```

Estructura elseif

Juntament amb la condició de l'estructura **if**, les estructures **elseif** ens permeten afegir més condicions. Cada estructura **elseif** té un conjunt d'accions associades que s'executaran si es compleix la condició de l'estructura **elseif** corresponent. Si no es compleix la condició d'una estructura **elseif**, es passarà a analitzar la condició de la següent estructura **elseif** i així fins que es trobi que es compleix la condició d'alguna estructura **elseif** o no es compleix cap, i en aquest últim cas, s'arribarà al final on podrem trobar un **else** o no. En la figura 4.2 podeu veure el diagrama d'aquesta estructura.

FIGURA 4.2. Estructura if - elseif - else



En el cas següent, com en els anteriors, la condició a analitzar és el valor que pren la variable **\$departament**, però en aquest cas **\$departament** pot ser igual a “**Comercial**”, “**RR.HH**” i “**Administració**”. Si el valor és “**Comercial**”, s'executarà el codi que hi ha entre les claus de l'estructura **if**, si és “**RR.HH**”

s'executarà el codi que hi ha entre les claus de l'estruatura del primer elseif i si és “**Administració**” s'executarà el codi que hi ha entre les claus de l'estruatura del segon elseif. Si no és cap d'aquests tres departaments, en el nostre cas com hi ha un else, s'executarà el codi que hi ha entre les claus de l'estruatura else. Si no hi hagués un else, no s'executaria cap acció.

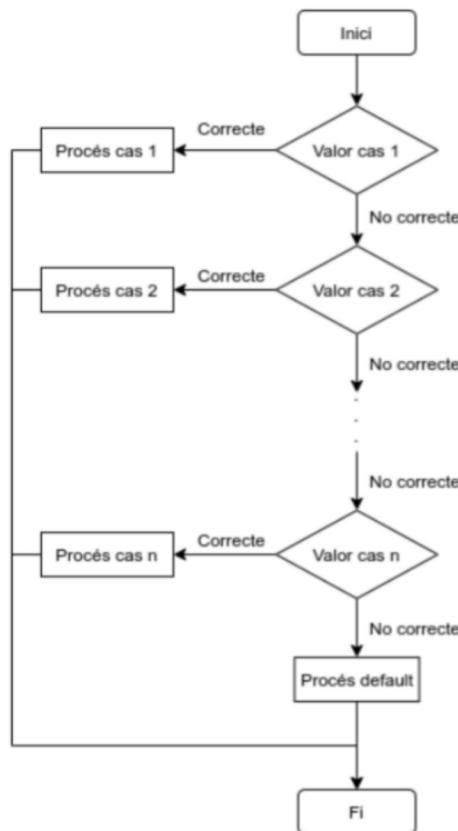
```

1 <?php
2
3 echo "<h3>ESTRUCTURA IF ELSEIF</h3>"; //Títol secció
4
5 //Declaració variables
6 $departament = "RR.HH";
7
8 //Estructura if elseif
9 if ( $departament == "Comercial" ) { //Si es compleix la condició aleshores...
10
11     echo "El departament comercial es troba en la 2a planta.<br/>"; //... es
12         realitza aquesta acció
13
14 } elseif ( $departament == "RR.HH" ) { //Si es compleix la condició aleshores
15     ...
16
17     echo "El departament de RR.HH es troba en la 1a planta.<br/>"; //... es
18         realitza aquesta acció
19
20 } elseif ( $departament == "Administració" ) { //Si es compleix la condició
21     aleshores...
22
23     echo "El departament d'administració es troba en la planta 0.<br/>"; //...
24         ... es realitza aquesta acció
25
26 } else { //Si no es compleix cap de les condicions anteriors aleshores...
27
28     echo "Aquest departament no existeix.<br/>"; //... es realitza aquesta acció
29
30 }
31
32 //quan sortim de l'estruatura if elseif
33 echo "Final de l'script.<br/>"; //... es realitza aquesta acció

```

Estructura switch

Mitjançant l'estruatura **switch** s'executarà un conjunt d'accions o altre depenent del valor que prengui una variable o retorna una expressió. En la figura 4.3 podeu veure el diagrama d'aquesta estruatura.

FIGURA 4.3. Estructura switch

En el cas següent, com en els anteriors, el valora a analitzar és el que pren la variable **\$departament**. En aquest cas **\$departament** pot ser igual a “**Comercial**”, “**RR.HH**” i “**Administració**”. Si el valor és “**Comercial**”, s’executarà el codi que hi ha després dels dos punts del primer cas, si és “**RR.HH**” s’executarà el codi que hi ha després dels dos punts del segon cas i si és “**Administració**” s’executarà el codi que hi ha després dels dos punts del tercer cas. Si no és cap d’aquests tres departaments, en el nostre cas, com hi ha un *default*, s’executarà el codi que hi ha després dels dos punts del *default*. Si no hi hagués un *default*, no s’executaria cap acció.

```

1 <?php
2
3 echo "<h3>ESTRUCTURA SWITCH 1</h3>"; //Títol secció
4
5 //Declaració variables
6 $departament = "RR.HH";
7
8 //Estructura switch
9
10 switch ( $departament ) { //variable a avaluar
11
12   case "Comercial": //Si el departament és el comercial....
13
14     echo "El departament comercial es troba en la 2a planta.<br/>"; //... es
15     //realitza aquesta acció
16
17     break; //sortim de l'estructura switch
18
19   case "RR.HH": //Si el departament és el de RR.HH....
20
21     echo "El departament de RR.HH es troba en la 1a planta.<br/>"; //... es
  
```

```

realitza aquesta acció
22
23     break;//sortim de l'estructura switch
24
25
26     case "Administració": //Si el departament és el d'administració....
27
28         echo "El departament d'administració es troba en la planta 0.<br/>"; //
... es realitza aquesta acció
29
30     break;//sortim de l'estructura switch
31
32
33     default: //Si el valor de $departament no és cap dels anteriors...
34
35         echo "Aquest departament no existeix.<br/>"; //... es realitza aquesta
acció
36
37     break; //sortim de l'estructura switch
38
39 }
40
41 //quan sortim de l'estructura switch
42
43 echo "Final de l'script.<br/>"; //... es realitza aquesta acció

```

Com podem veure, aquest cas és equivalent al cas de l'estructura if - elseif - else vist en l'apartat anterior.

Si les accions que s'han d'executar en un cas d'un switch, són les mateixes que en altres casos del mateix switch, aleshores aquests casos es poden agrupar per només escriure un sol cop les accions. En el següent script veurem un exemple on comprovarem si el valor d'un enter entre 1 i 8 és parell o imparell. En els casos que el valor sigui parell, les accions a executar són les mateixes i el mateix passa en els casos que el valor sigui imparell, per tant, agrupem els casos parells per un costat, i els imparells per un altre.

```

1 <?php
2
3 echo "<h3>ESTRUCTURA SWITCH 2</h3>"; //Títol secció
4
5 /*Comprovarem si un nombre entre 1 i 8 és parell o imparell.*/
6
7 $enter = 1; //Nombre enter a avaluar
8
9 switch ($enter) {
10
11     //Nombres enters imparells
12     case 1:
13     case 3:
14     case 5:
15     case 7:
16
17         echo "$enter és imparell.<br/>"; //Acció a executar en cas que el valor
sigui imparell
18
19         break;
20
21     //Nombres enters parells
22     case 2:
23     case 4:
24     case 6:
25     case 8:
26
27         echo "$enter és parell.<br/>"; //Acció a executar en cas que el valor sigui

```

```

1 parell
2
3 break;
4
5 //No és un nombre enter o bé és un nombre enter més petit que 1 o bé és un
6 nombre enter més gran que 8
7 default:
8
9 echo "El valor no és un nombre enter o no està entre 1 i 8.<br/>";
10
11 break;
12 }

```

Els valors dels casos no tenen per què ser un valor literal, poden ser expressions que ens retornen un valor. En el següent script veurem un exemple. L'exemple es basa en l'script anterior, però en lloc d'escriure valors literals pels casos (1, 3, 5 i 7 pels valors imparells, i 2, 4, 6 i 8 pels valors parells), escriurem una expressió que ens retornarà verdader si el residu de dividir l'enter entre 2 és 0 (valor parell) i fals en cas contrari (valor imparell).

```

1 <?php
2
3 echo "<h3>ESTRUCTURA SWITCH 3</h3>"; //Títol secció
4
5 /*Comprovarem si un nombre enter entre 1 i 8 és parell o imparell amb
6   expressions.*/
7
8 $enter = 2; //Nombre enter a avaluar
9
10 switch ($enter) {
11
12     //Nombres enters imparells, és a dir, els que compleixen que $enter % 2 != 0
13     case $enter % 2 != 0:
14
15         echo "$enter és imparell.<br/>";
16
17         break;
18
19     //Nombres enters parells, és a dir, els que compleixen que $enter % 2 == 0
20     case $enter % 2 == 0:
21
22         echo "$enter és parell.<br/>";
23
24         break;
25
26     //No és un nombre enter o bé és un nombre enter més petit que 1 o bé és un
27     //nombre enter més gran que 8
28     default:
29
30         echo "El valor no és un nombre enter o no està entre 1 i 8.<br/>";
31
32         break;
33 }

```

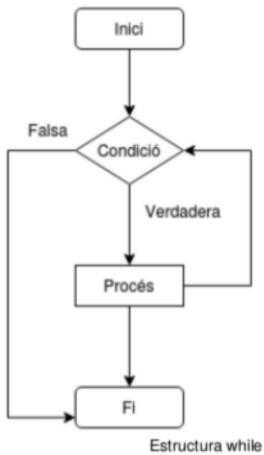
4.1.2 Estructures de control iteratives

Les estructures de control iteratives ens permeten repetir un conjunt d'accions, depenen si es compleixen determinades condicions; aquestes són:

- l'estructura while,
- l'estructura do - while,
- l'estructura for

Estructura while

Mitjançant l'estructura **while** les accions s'executaran tants cops com es compleixi una condició, si no, no s'executaran.



En l'exemple següent imprimirem per pantalla la taula de multiplicar del 2 fent servir l'estructura iterativa while. La condició a analitzar és l'expressió que ens retornarà verdader si es compleix que **\$comptador** és igual o menor que 10, si és així, s'executarà el codi que hi ha entre les claus (imprimir per pantalla els 10 productes que formen la taula de multiplicar del 2), en cas contrari, continuarem amb el procés de l'script continuant per la línia de codi que trobarem just després de l'estructura while.

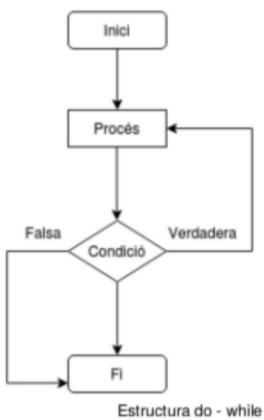
```

1  <?php
2
3  echo "<h3>ESTRUCTURA WHILE</h3>"; //Títol secció
4
5  //Declaració variables
6  $comptador = 1;
7
8  //Estructura while
9
10 while ($comptador <= 10) { //mentre es compleixi la condició...
11
12     echo "$comptador x 2 = ".($comptador * 2)."<br/>"; //... es realitza aquesta
13     //acció...
14
15     $comptador++; //i finalment augmentem en 1 el comptador
16 }
17
18 //Si no es compleix la condició, no entrem dins de l'estruutura i sortim
19 echo "<p>Final de l'script amb comptador = $comptador.</p>"; //Acció final de l'script un cop sortim de l'estruutura

```

Estructura do - while

Mitjançant l'estructura **do - while**, com en el cas de l'estructura while, les accions s'executaran tants cops com es compleixi una condició, si no, no s'executaran, però en aquest cas, com a mínim les accions sempre s'executaran un primer cop, ja que la condició s'avalua un cop executades les accions.



En l'exemple següent imprimirem per pantalla els nombres majors a 200 i menors a 400. La condició a analitzar és l'expressió que ens retornarà verdader si es compleix que **\$nombre** és major que 200 i menor que 400, si és així, s'executarà el codi que hi ha entre les claus, en cas contrari, continuarem amb el procés de l'script continuant per la línia de codi que trobarem just després de l'estructura do - while, però en aquest cas, com a mínim s'imprimirà **\$nombre** un primer cop, es compleixi o no la condició que evaluarem un cop executades les accions.

```

1 <?php
2
3 echo "<h3>ESTRUCTURA DO...WHILE</h3>"; //Títol secció
4
5 //Declaració variables
6 $nombre = 1;
7
8 //Estructura do...while
9
10 do{ //realitzem...
11
12 echo "El nombre és: $nombre<br/>"; //...aquesta acció...
13
14 $nombre++; //...augmentem en 1 el comptador...
15
16 }while (($nombre > 200) && ($nombre < 400)); //...mentre es compleixi aquesta
17 condició
18 /*En aquest cas sempre es realitzaran les accions de dins de l'estructura com a
19 mínim un cop (la primera), ja que la
*condició s'avalua un cop executades les accions de dins l'estructura*/
20
21 //sortim de l'estructura while quan la condició ja no es compleix i ..
22
23 echo "<p>Final de l'script amb nombre = $nombre.</p>"; //... es realitza
aquesta acció

```

Estructura for

L'estructura **for**, fa el mateix que l'estructura while, les accions s'executaràn tants cops com es compleixi una condició, si no, no s'executaràn. La diferència amb el while, és que el comptador i l'expressió per aplicar un increment o decrement al comptador, poden ser arguments de l'estructura **for**, igual que ho és la condició. El diagrama de flux de l'estructura for, serà el mateix que el de l'estructura while.

En l'exemple següent imprimirem per pantalla la taula de multiplicar del 2 de manera creixent (de l'1 al 10) i de manera decreixent (del 10 a l'1) fent servir l'estructura iterativa for, és a dir:

- En el cas de la taula creixent, la condició a analitzar és l'expressió que ens retornarà verdader si es compleix que **\$comptador** és igual o menor que 10, inicialitzant **\$comptador** a 0, si és així, s'executarà el codi que hi ha entre les claus (imprimir per pantalla els 10 productes que formen la taula de multiplicar del 2 creixent) de l'estructura for. En aquest cas, sumarem a **\$comptador** una unitat cada cop que finalitzi l'execució de l'acció d'imprimir per pantalla el producte corresponent.
- En el cas de la taula creixent, la condició a analitzar és l'expressió que ens retornarà verdader si es compleix que **\$comptador** és major que 10, inicialitzant **\$comptador** a 10, si és així, s'executarà el codi que hi ha entre les claus (imprimir per pantalla els 10 productes que formen la taula de multiplicar del 2 decreixent) de l'estructura for. En aquest cas, restarem a **\$comptador** una unitat cada cop que finalitzi l'execució de l'acció d'imprimir per pantalla el producte corresponent.

```

1  <?php
2
3  echo "<h3>ESTRUCTURA FOR</h3>"; //Títol secció
4
5  /*L'estruutura for està formada per tres arguments: inicialització d'una
   variable (comptador), condició que
6   *s'ha de complir perquè s'executin les accions de dins l'estruutura for i
   sumatori o decrement de la variable
7   *inicialitzada.
8   *Dels arguments només és obligatori el segon, però si no utilitzem un dels
   altres dos, han d'estar presents
9   *en l'script com fem amb el while. */
10
11 echo "<h4>TAULA DE MULTIPLICAR DEL 2 CREIXENT:</h4>";
12
13 //Inicialitzem la variable (argument 1: $comptador=1)
14 for ($comptador=1; $comptador<=10; $comptador++) { //Mentre es compleixi la
   condició (argument 2: $comptador<=10)
15
16 echo $comptador." x 2 = ".($comptador * 2)."<br/>";
17
18 //augmentem el comptador en 1 (argument 3: $comptador++)
19
20 }
21
22 echo "<p>Final de l'script amb comptador = $comptador.</p>"; //... es realitza
   aquesta acció
23
24 echo "<h4>TAULA DE MULTIPLICAR DEL 2 DECREIXENT:</h4>";
25
26 //Inicialitzem la variable (argument 1: $comptador=10)
27 for ($comptador=10; $comptador>0; $comptador--) { //Mentre es compleixi la
   condició (argument 2: $comptador>0)
28
29 echo $comptador." x 2 = ".($comptador * 2)."<br/>";
30 //disminuïm el comptador en una unitat (argument 3: $comptador--)
31
32 }
33
34 //sortim de l'estruutura for quan la condició ja no es compleix i ...
35
36 echo "<p>Final de l'script amb comptador = $comptador.</p>"; //... es realitza
   aquesta acció

```

4.1.3 Estructures de control niuades

Quan parlem d'**estructures de control niuades**, ens referim al fet que dins d'una estructura de control trobem altres estructures de control.

Dins d'una estructura de control, sigui iterativa o condicional, podem trobar altres estructures de control del mateix tipus o diferents, és a dir, dins d'una estructura iterativa puc trobar altres estructures iteratives o condicionals, i dins d'una estructura condicional altres condicionals o iteratives. Això és al que anomenem estructures de control niuades.

En el següent exemple veurem una de les utilitats que té niuar estructures de control. En aquest cas el que farem és imprimir per pantalla una taula HTML de 12 x 12 de manera dinàmica, és a dir, sense haver d'escriure una per una totes les seves files i cel·les.

```

1 <?php
2
3 echo "<h3>ESTRUCTURES ITERATIVES NIUADES</h3>"; //Títol secció
4
5 //Creació d'una taula HTML de 12 x 12 de manera dinàmica
6
7 echo "<table style=\"border: 1px solid #000;\">"; //Obrim taula amb estil
8
9 //Construïm les files
10 for ($fila=1; $fila<=12; $fila++) {
11
12 echo "<tr>"; //Obrim fila
13
14 //Construïm les cel·les de la fil·la oberta
15 for ($cella=1; $cella<=12; $cella++) {
16
17 //Obrim cel·la amb estil
18 echo "<td style=\"border: 1px solid #000; width: 25px; text-align:center
19 ;\\">";
20
21 //Mostrem contingut de la cel·la
22 echo ($fila * $cella);
23
24 //Tanquem cel·la
25 echo "</td>";
26 }
27
28 echo "</tr>"; //Tanquem fila
29 }
30
31 echo "</table>"; //Tanquem taula

```

Les estructures de control es poden niuar fins a tants nivells com necessitem, és a dir, dins d'una estructura de control, podem trobar una altra estructura de control, que a l'hora també conté una altra estructura de control, i així fins a tants nivells com necessitem.

4.1.4 Recorregut d'arrays

Per recórrer un array, és a dir, accedir a tots els seus elements, ho podem fer mitjançant estructures de control iteratives. En el cas d'un array numèric ho podem fer mitjançant qualsevol estructura de les que ja coneixem, i en el cas dels arrays associatius, ho podem fer mitjançant una nova estructura, l'estructura **foreach**.

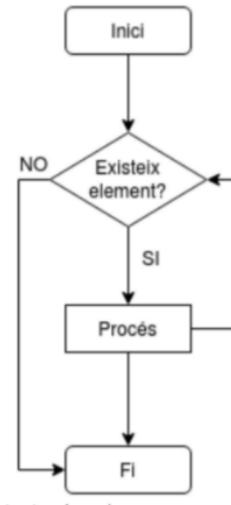
L'estructura **foreach**, fa un recorregut per tots els elements d'un array, sigui numèric o associatiu, des del primer element fins a l'últim, retornant el contingut dels elements i opcionalment els dels índexs. Cada cop que accedeix a un element, s'executarà el conjunt d'accions que trobem entre les claus de l'estructura **foreach**.

En el següent exemple veurem un recorregut d'un array numèric mitjançant l'estructura for i d'un array associatiu mitjançant l'estructura foreach.

```

1 <?php
2

```



Estructura foreach

```

3  /*Per recórrer un array, accedir a tots els seus elements, ho podem fer mitjanç
   ant estructures iteratives. En el cas d'un array
4   *numèric ho podem fer mitjançant qualsevol estructura de les vistes fins ara,
   i en el cas dels arrays associatius, ho podem
5   *fer mitjançant una nova estructura, l'estructura foreach*/
6
7 //RECORREGUT D'UN ARRAY NUMÈRIC
8
9 echo "<h3>RECORREGUT D'UN ARRAY ÚNIMERIC</h3>"; //Títol secció
10
11 $arrayNumeric = array("Pep", "RR.HH", 48, "00000000T"); //Definim un array numèric
12
13 for ($i=0; $i<count($arrayNumeric); $i++) { //Mentre quedin elements...
14     echo " L'element $i és $arrayNumeric[$i]<br/>"; //mostrem l'índex de l'
   element actual i el seu contingut
15 }
16
17 //RECORREGUT D'UN ARRAY ASSOCIATIU
18
19 echo "<h3>RECORREGUT D'UN ARRAY ASSOCIATIU</h3>"; //Títol secció
20
21 $arrayAssociatiu = array("Nom"=>"Pep", "Departament"=>"RR.HH", "Edat"=>48, "Nif"=>
   "00000000T"); //Definim un array associatiu
22
23 foreach ($arrayAssociatiu as $index => $contingut) { //Mentre quedin elements
   ...
24     echo " $index : $contingut<br/>"; //mostrem l'índex de l'element actual i el
   seu contingut
25 }
```

En el cas de l'estructura `foreach`, el primer argument **\$arrayAssociatiu** és l'array que volem recórrer, **\$index** la variable on es guardarà l'índex de l'element actual (al que en aquest moment està accedint l'estructura `foreach`) i **\$contingut** la variable on es guardarà el contingut de l'element actual. La variable **\$index** és opcional, per tant, si no volem obtenir l'índex, com arguments de l'estructura `foreach` simplement escriurem:

```
1  foreach ($arrayAssociatiu as $contingut){....}
```

4.2 Funcions integrades de PHP

Una **funció** és un conjunt d'instruccions independents agrupades en un mateix bloc i que poden cridar-se per ser executades en qualsevol lloc d'un script.

Les funcions en PHP es poden dividir en dos grans grup, les funcions **integrades** o **predefinides**, i les funcions **d'usuari**.

Les funcions **integrades** o **predefinides**, són les que defineix el llenguatge, és a dir, en el nostre cas PHP, i les funcions **d'usuari**, són les que defineix l'usuari, és a dir, el programador.

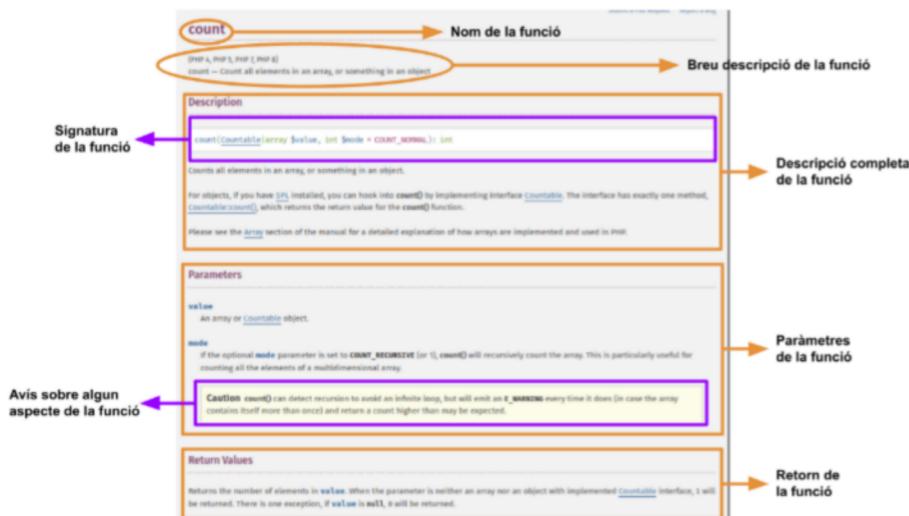
En la figura 4.4 podem veure un exemple de com s'especifiquen les signatures i descriuen les funcions en el lloc web [php.net](http://www.php.net). En aquest cas la funció que es

Funcions predefinides

La signatura i descripció de les funcions predefinides de PHP, les podem trobar al lloc web oficial de PHP www.php.net.

mostra és la funció **count**.

FIGURA 4.4. Estructura per especificar la signatura i descriure una funció en el lloc web php.net



La signatura i descripció de totes les funcions **integrades** que trobem en php.net, segueixen la mateixa estructura que podem veure en la figura 4.4.

Quan nosaltres volem executar una funció dins d'un script, sigui integrada o d'usuari, el que fem és **cridar** a la funció. Cridar una funció, no és més que escriure en el lloc de l'script on volem executar la funció, el seu nom i els paràmetres corresponents segons la seva signatura.

4.3 Funcions de cadenes de caràcters

Les funcions de cadenes de caràcters, són aquelles funcions integrades de PHP per treballar amb cadenes de caràcters. Nosaltres veurem algunes d'aquestes funcions, en concret, les que més es fan servir:

- Indexació de cadenes de caràcters
- Funcions per treballar amb subcadenes
- Funcions per netejar cadenes de caràcters
- Funcions per modificar cadenes de caràcters

4.3.1 Indexació de cadenes de caràcters

La indexació de cadenes de caràcters, no fa referència a cap funció, sinó que és la manera que tenim d'accendir als caràcters d'una cadena.

Per accedir als caràcters d'una cadena de caràcters, ho podem fer com ho fem per accedir als elements d'un array, és a dir, a cada caràcter de la cadena li corresindrà un índex que començarà per l'índex 0 assignat al primer caràcter de la cadena i anirà augmentat en una unitat fins a arribar a l'últim caràcter de la cadena.

En el següent exemple podeu veure com indexar una cadena de caràcters:

```

1 <?php
2
3 /* Podem accedir als caràcters d'una cadena mitjançant un índex. Primer carà
   cter corresindrà a l'índex 0.
4 * Els índexs augmentaran de manera consecutiva d'esquerra a dreta fins arribar
   a l'últim caràcter. */
5
6 $frase = "Sóc una frase";
7 echo "El primer caràcter és $frase[0]<br/>"; //Mostrarà el caràcter "S"
8 echo "El cinquè caràcter és $frase[5]<br/>"; //Mostrarà el caràcter "u"
9
10 /* La funció strlen("cadena") ens retorna la longitud d'una cadena, és a dir,
    la quantitat de caràcters
11 * que té la cadena tenint en compte que també comptabilitza el final de línia
    com un caràcter i què els
12 * caràcters amb titlla (accents, ñ, etc..) els compte com a dos. */
13
14 $longitud = strlen($frase); //Longitud $frase
15 echo "La longitud de la frase és $longitud<br/>";
16 $ultimoCaracter = $frase[$longitud - 1]; //Mostrarà el caràcter "e"
17 echo "L'últim caràcter és $ultimoCaracter<br/>";

```

Hem de tenir en compte què als caràcters amb titlla (accents, ñ, etc..) els hi corresponen dos índexs. En l'exemple al caràcter “ó” de \$frase li corresponen els índexs 1 i 2, així doncs, \$frase[3], ens retornarà el caràcter “c”.

Finalment, en l'exemple podeu veure la funció **strlen(\$frase)** integrada de PHP, què el que fa és retornar la longitud de la cadena passada per paràmetre, és a dir, en l'exemple la longitud de la cadena \$frase, la qual és 14, que correspon al nombre de caràcters de \$frase, tenint en compte que el caràcter “ó” es comptabilitza com a dos caràcters, ja que porta titlla.

4.3.2 Funcions per treballar amb subcadenes

Dins de les funcions integrades de PHP, hi ha un conjunt de funcions que serveixen per treballar amb les subcadenes que formen part d'una cadena.

Una subcadena és un conjunt de caràcters seqüencials dins d'una cadena de caràcters.

D'aquestes, nosaltres veurem les funcions **strstr**, **strpos**, **substr** i **strtok**.

Funció strstr

Veiem el següent exemple:

```

1 <?php
2
3 /* strstr("cadena", "subcadena").
4  * Retorna la subcadena cercada si la troba, si no retorna fals.
5  * És case sensitive. */
6
7 $cadena = "Sóc una cadena";
8
9 echo "<h3> FUNCIÓ STRSTR </h3>";
10
11 $subCadena = strstr($cadena, "una cadena");
12 if ($subCadena) { //Subcadena trobada
13     echo "\"$subCadena\" trobada.<br/>";
14 } else { //Subcadena no trobada
15     echo "No s'ha trobat la subcadena.<br/>";
16 }
```

En aquest cas, la funció **strstr(\$cadena, “una cadena”)** ens retorna la subcadena “una cadena” que hem passat com a segon paràmetre, ja que es troba dins de la cadena \$cadena que hem passat com a primer paràmetre. Si la subcadena no es trobés dins la cadena, la funció ens retornaria fals.

Funció strpos

Veiem el següent exemple:

```

1 <?php
2
3 /* strpos("cadena", "subcadena", [índex por on començar la cerca]).
4  * Retorna la posició del primer caràcter de la subcadena cercada. Si no troba
5  * la subcadena retorna fals.
6  * És case sensitive. */
7
8 $cadena = "Sóc una cadena";
9
10 echo "<h3> FUNCIÓ STRPOS</h3>";
11
12 $posicio = strpos($cadena, "una cadena");
13 if ($posicio) { //Subcadena trobada
14     echo "El primer caràcter de \"\$subCadena\" ocupa la posició $posicio.<br/>";
15 }
16
17 //Cerquem la subcadena a partir de la posició 7
18 $posicio = strpos($cadena, "una cadena", 7);
19 if ($posicio) { //Subcadena trobada
20     echo "El primer caràcter de \"\$subCadena\" ocupa la posició $posicio.<br/>";
21 }
22
23 } else { //Subcadena no trobada
24     echo "No s'ha trobat la subcadena.<br/>";
}
```

En la primera part de l'script de l'exemple, la funció **strpos(\$cadena, “una cadena”)** ens retorna l'índex 5 de la cadena \$cadena que hem passat com a primer paràmetre, ja que correspon al primer caràcter de la subcadena “una cadena” que

hem passat com a segon paràmetre. Si la subcadena no es trobés dins la cadena, la funció ens retornaria fals.

En la segona part de l'script de l'exemple, a la funció li passem un tercer paràmetre opcional, per tant, la funció **strpos(\$cadena, "una cadena", 7)** ens retorna fals perquè la subcadena “una cadena” que hem passat com a segon paràmetre no es troba en la cadena \$cadena que hem passat com a primer paràmetre a partir de l'índex 7, què és l'enter que hem passat com a tercer paràmetre i indica l'índex de la cadena passada com a primer paràmetre a partir del qual hem de cercar la subcadena passada com a segon paràmetre.

Funció substr

Veiem el següent exemple:

```

1 <?php
2
3 /* substr("cadena", índice per on començar la cerca, [nombre de caràcters a
   mostrar]). */
4 * Retorna la cadena de caràcters que es troba des de la posició passada com a
   paràmetre fins al final
5 * de la cadena o bé tants caràcters com el número passat com a tercer parà
   metre. */
6
7 $cadena = "Sóc una cadena";
8
9 echo "<h3> FUNCIÓ SUBSTR</h3>";
10
11 echo substr($cadena, 3) . "<br/>"; //Retornarà "c una cadena"
12 echo substr($cadena, 3, 4) . "<br/>"; //Retornarà "c un"
```

En l'exemple, la funció **substr(\$cadena,3)** ens retorna la subcadena “c una cadena”, ja que és la subcadena que trobem dins de la cadena \$cadena que hem passat com a primer paràmetre, a partir de l'índex 3 passat com a segon paràmetre fins al final de la cadena.

La funció **substr(\$cadena,3,4)** de l'exemple, ens retorna la subcadena “c un”, ja que és la subcadena que trobem dins de la cadena \$cadena que hem passat com a primer paràmetre, a partir de l'índex 3 passat com a segon paràmetre fins al final de la cadena, menys els 4 últims caràcters. Per tant, el tercer paràmetre opcional de la funció, indica el nombre de caràcters que li hem de treure a la subcadena retornada començant pel final d'aquesta, que correspondrà amb el final de la cadena passada com a primer paràmetre.

Funció strtok

Veiem el següent exemple:

```

1 <?php
2
3 /* strtok("cadena", "delimitador o delimitadors").
   * Retorna la subcadena fins que troba un dels delimitadors passats com a parà
   * metres. Si no troba el delimitador
5 * passat com a paràmetre retorna fals. Només és necessari passar-li la cadena
   * en la primera crida al mètode,
```

```

6 * ja que realitza un seguiment de la cadena a partir del lloc en què es troba
7   en la cadena actual. */
8
9 $cadena = "Sóc una cadena";
10
11 echo "<h3> FUNCIO STRTOK</h3>";
12
13 $delimitador = " ";
14 $resultat = strtok($cadena, $delimitador);
15 echo $resultat . "<br/>"; //Retorna Sóc
16
17 echo "<h3> FUNCIO STRTOK DINS UNA ESTRUCTURA ITERATIVA</h3>";
18
19 //Mostrem totes les subcadenes sense els espais en blanc
20 $resultat = strtok($cadena, $delimitador); //Incialitzem de nous $resultat
21 while ($resultat) { //Mentre quedí cadena
22     echo $resultat . "<br/>"; //Mostrem resultat
23     $resultat = strtok($delimitador); //Llegim següent subcadena
24 }
echo "<p/>";

```

En la primera part de l'script de l'exemple, la funció **strtok(\$cadena, \$delimitador)** ens retorna “Sóc”, ja que és la subcadena que trobem en la cadena \$cadena que hem passat com a primer paràmetre des de l'inici d'aquesta fins que trobem “ ” què és el delimitador que hem passat com a segon paràmetre. Si el delimitador no es trobés dins la cadena, la funció ens retornaria fals.

En la segona part de l'script de l'exemple, la funció **strtok(\$cadena, \$delimitador)** ens retorna totes les subcadenes que trobem en la cadena \$cadena que hem passat com a primer paràmetre des de l'inici d'aquesta fins que trobem el primer “ ”, què és el delimitador que hem passat com a segon paràmetre, és a dir la subcadena “Sóc”, les subcadenes entre dos “ ”, és a dir, en l'exemple només la subcadena “una”, i la subcadena des de l'últim “ ” fins al final de la cadena, és a dir “cadena”.

Això és possible perquè cridem la funció dins una estructura iterativa while i aquesta funció inclou un punter que sempre apunta al següent índex després del delimitador passat com a segon paràmetre de la cadena passada com a primer paràmetre.

4.3.3 Funcions per netejar cadenes de caràcters

Dins de les funcions integrades de PHP, hi ha un conjunt de funcions que serveixen per netejar les cadenes de caràcters, és a dir, per eliminar caràcters d'aquestes que ens poden fer nosa a l'hora de treballar amb elles, com per exemple, quan un usuari ha d'introduir el seu correu electrònic en un formulari i, sense adonar-se, afegeix algun espai en blanc a l'inici de l'adreça. Aquests espais en blanc sobren, si els deixem, el sistema no reconeixerà l'adreça de correu electrònic. Una solució és eliminar-los mitjançant una d'aquestes funcions; com ara: **trim**, **ltrim** i **strip_tags**.

Funció trim

Veiem el següent exemple:

```

1 <?php
2
3 /* trim("cadena", [llista de caràcter a eliminar]). */
4 * Retorna la cadena sense els espais en blanc de l'inici i final de la cadena.
5 * Per defecte elimina tots els caràcters que insereixen un espai en blanc de
6 * la cadena:
7 * " " (ASCII 32), espai simple.
8 * "\t" (ASCII 9), tabulació.
9 * "\n" (ASCII 10), salt de línia.
10 * "\r" (ASCII 13), retorn de carro.
11 * "\0" (ASCII 0 ), NUL.
12 * "\x0B" (ASCII 11), tabulació vertical.
13 * Si no es volen eliminar tots, s'especificarà mitjançant el segon paràmetre
14 * aquells que es volen
15 * eliminar. */
16
17 $cadena = " Sóc una cadena ";
18 $cadenaNeta = trim($cadena);
19 echo "<p>$cadenaNeta.<br/>";

```

En l'exemple, la funció **trim(\$cadena)** ens retorna “Sóc una cadena”, és a dir, la cadena passada per paràmetre sense els espais en blanc de l'inici i del final.

Per defecte elimina tota mena de caràcter en blanc, però si només volem eliminar alguns d'aquests tipus, els escriurem en format de llista com segon paràmetre opcional de la funció.

Funció ltrim

Veiem el següent exemple:

```

1 <?php
2
3 /* ltrim("cadena", [llista de caràcter a eliminar]). */
4 * Retorna la cadena sense els caràcters en blanc de l'inici. Si no es volen
5 * eliminar tots els tipus de caràcters
6 * d'espai en blanc, s'especificarà mitjançant el segon paràmetre aquells que
7 * es volen eliminar.*/
8
9 $cadena = " Sóc una cadena ";
10 $cadenaNeta = ltrim($cadena);
11 echo "$cadenaNeta.<br/>";

```

En l'exemple, la funció **ltrim(\$cadena)** ens retorna “Sóc una cadena”, és a dir, la cadena passada per paràmetre sense els espais en blanc de l'inici.

Com passa amb la funció **trim**, per defecte elimina tota mena de caràcter en blanc, però si només volem eliminar alguns d'aquests tipus, els escriurem en format de llista com segon paràmetre opcional de la funció.

Funció strip_tags

Veiem el següent exemple:

```

1 <?php
2
3 /* strip_tags("cadena", [llista d'etiquetes a eliminar]).
4  * Retorna la cadena sense les etiquetes html. Si no es volen eliminar totes
5  * les etiquetes HTML,
6  * s'especificarà mitjançant el segon paràmetre aquelles que no es volen
7  * eliminar.*/
8
9 $cadena = "<p><b>Sóc una cadena</b></p>";
10 $cadenaNeta = strip_tags($cadena, "<b></b>");
11 echo "$cadenaNeta.<br/>";
12 echo"<p/>";

```

En l'exemple, la funció **strip_tags(\$cadena, “”)** ens retorna “**<p>Sóc una cadena</p>**”, és a dir, la cadena passada per paràmetre sense les etiquetes HTML “****” passades com a segon paràmetre.

Si volem que s'eliminin totes les etiquetes HTML de la cadena, no passarem el segon paràmetre optional.

4.3.4 Funcions per modificar cadenes de caràcters

Dins de les funcions integrades de PHP, hi ha un conjunt de funcions que serveixen per modificar les cadenes de caràcters. D'aquestes nosaltres veurem les funcions **substr_replace**, **str_replace**, **strtoupper** i **strtolower**.

Funció substr_replace

Veiem el següent exemple:

```

1 <?php
2
3 /* substr_replace("cadena", "substitut", inici, [longitud]).
4  * Substitueix una subcadena de "cadena" delimitada pels paràmetres inici i
5  * opcionalment longitud amb la cadena "substitut".
6  * El paràmetre inici és un nombre enter. Si és positiu, el reemplaçament s'
7  * iniciarà en la posició de la cadena que coincideixi
8  * amb aquest número. Si no indiquem longitud es realitzarà la substitució fins
9  * a arribar al final de la cadena, si s'indica
10 * longitud es realitzarà la substitució tantes posicions com indiqui longitud.
11 */
12
13 $cadena = "Sóc una cadena";
14 $substitut = "frase";
15 $resultat = substr_replace($cadena, $substitut, 9);
16 echo "$resultat.<br/>"; //Sóc una frase.
17 $resultat = substr_replace($cadena, $substitut, 9, 5);
18 echo "$resultat.<br/>"; //Sóc una frasea

```

En l'exemple, la funció **substr_replace(\$cadena, \$substitut, 9)** ens retorna la cadena “Sóc una frase” què és el resultat de substituir el contingut de la cadena \$cadena, passada com a primer paràmetre, des de l'índex 9, passat com a tercer paràmetre, fins al final de la cadena per la cadena \$substitut, passada com a segon paràmetre.

La funció **substr_replace(\$cadena, \$substitut, 9, 5)** de l'exemple, ens retorna la cadena “Sóc una frasea” què és el resultat de substituir el contingut de la cadena \$cadena, passada com a primer paràmetre, des de l'índex 9, passat com a tercer paràmetre, fins a 5 caràcters més, què és el valor passat com a quart paràmetre opcional, per la cadena \$substitut, passada com a segon paràmetre.

En l'exemple la substitució comença a partir de l'índex 9, a la que li hem d'afegir 5 caràcters més, per tant, en total la cadena que farà de substituta ha d'estar formada per $1 + 5 = 6$ caràcters, el primer corresponent a l'índex passat com a tercer paràmetre i els 5 restants al valor passat com a quart paràmetre. Així doncs a "frase" que és la cadena que farà de substituta passada com a segon paràmetre, li faltarà 1 caràcter per arribar als 6, ja que només té 5.

En aquest cas el que fa la funció, és afegir-li els caràcters necessaris de la cadena original que es troben a partir de la longitud de la cadena substituïda, és a dir, en l'exemple el caràcter “a” què és el que hi ha després de la longitud 5, corresponent a la longitud de “frase”. En la figura 4.5 podeu veure una representació gràfica del que s’ha explicat.

FIGURA 4.5. Funcionament del segon substr_replace de l'script d'exemple



Funció str_replace

Vejem el següent exemple:

```
1 <?php
2
3 /* str_replace("subcadena", "substitut", "cadena", [nombre substitucions]). */
4 * Retorna la "cadena" amb les seves subcadenes iguals a "subcadena" reemplaç-
5   ades per "substitut". */
6
7 $cadena = "Sóc una cadena";
8 $subcadena = "a";
9 $substitut = "j";
```

```

9 $resultat = str_replace($subcadena, $substitut, $cadena);
10 echo "$resultat.<br/>"; //Sóc uni cideni.
```

En l'exemple, la funció **str_replace(\$subcadena, \$substitut, \$cadena)** ens retorna la cadena “Sóc uni cideni” què és el resultat de substituir totes les subcadenes “a”, iguals a la subcadena passada com a primer paràmetre, de la cadena \$cadena, passada com a tercer paràmetre, per la cadena “i”, passada com a segon paràmetre.

Si no volem substituir totes les subcadenes iguals a la cadena passada com a primer paràmetre, podeu afegir un quart paràmetre opcional per indicar quantes substitucions s'han de fer. En aquest cas es faran tantes substitucions com indiqui aquest paràmetre començant per l'esquerra de la cadena.

Funció strtoupper

Veiem el següent exemple:

```

1 <?php
2
3 /* strtoupper("cadena"). Converteix les minúscules de la cadena en majúscules.
   */
4
5 $cadena = "Sóc una cadena";
6 $resultat = strtoupper($cadena);
7 echo "$resultat.<br/>"; //SÓC UNA CADENA.
```

En l'exemple, la funció **strtoupper(\$cadena)** ens retorna “SÓC UNA CADENA”, és a dir, la cadena passada per paràmetre amb totes les seves lletres en majúscula.

Quan apliquem la funció strtoupper, segons el tipus de família de fonts amb el que vulguem mostrar les lletres en el navegador, les lletres amb titlla se seguiran mostrant en minúscula.

Funció strtolower

Veiem el següent exemple:

```

1 <?php
2
3 /* strtolower("cadena"). Converteix les majúscules de la cadena en minúscules.
   */
4
5 $resultat = strtolower($cadena);
6 echo "$resultat.<br/>"; //sóc una cadena.
```

En l'exemple, la funció **strtolower(\$cadena)** ens retorna “sóc una cadena”, és a dir, la cadena passada per paràmetre amb totes les seves lletres en minúscula.

4.4 Funcions d'arrays

Les funcions d'arrays, són aquelles funcions integrades de PHP per treballar amb arrays. Nosaltres veurem algunes d'aquestes funcions, en concret, les que més es fan servir:

- Funcions per inserir i eliminar elements dels arrays
- Funcions per consultar els elements dels arrays
- Funcions per crear nous arrays a partir d'altres arrays

4.4.1 Funcions per inserir i eliminar elements dels arrays

Dins de les funcions integrades de PHP, hi ha un conjunt de funcions que serveixen per inserir i eliminar elements dels arrays. D'aquestes nosaltres veurem les funcions **array_push**, **strpos**, **substr** i **strtok**.

Veiem el següent exemple:

```

1 <?php
2
3 $treballador = array("Nom"=>"Pep", "Departament"=>"RR.HH", "Edat"=>48, "NIF"=>
4     "00000000T");
5
6 /* Inserim un o més elements al final d'un vector que ja existeix */
7 echo "<h3>FUNCIO ARRAY_PUSH</h3>";
8
9 array_push($treballador, "C/Gran, 13", "Barcelona");
10
11 //Mostrem contingut
12 foreach ($treballador as $index => $valor) {
13     echo "$index...$valor<br/>";
14 }
15 echo "<hr/>"; //Inserim línia html
16
17 /* Eliminem els dos últims elements que hem inserit anteriorment */
18 echo "<h3>FUNCIO ARRAY_POP</h3>";
19
20 for ($i = 1; $i <= 2; $i++) {
21     array_pop($treballador); //Eliminem element
22 }
23
24 //Mostrem contingut
25 foreach ($treballador as $index => $valor) {
26     echo "$index...$valor<br/>";
27 }
28 echo "<hr/>"; //Inserim línia html
29
30 /* Inserim un o més elements a l'inici d'un vector que ja existeix */
31 echo "<h3>FUNCIO ARRAY_UNSHIFT</h3>";
32
33 array_unshift($treballador, "C/Gran, 13", "Barcelona");
34
35 //Mostrem contingut
36 foreach ($treballador as $index => $valor) {
37     echo "$index...$valor<br/>";
38 }
39 echo "<hr/>"; //Inserim línia html
40
41 /* Eliminem els dos primers elements que hem inserit anteriorment */
42 echo "<h3>FUNCIO ARRAY_SHIFT</h3>";
43
44 for ($i = 1; $i <= 2; $i++) {
45     array_shift($treballador); //Eliminem element
46 }
47
48 //Mostrem contingut
49 foreach ($treballador as $index => $valor) {
```

```
49     echo "$index...$valor<br/>";
50 }
```

En aquest exemple, la funció **array_push(\$treballador, “C/Gran, 13”, “Barcelona”)** crearà dos nous elements que afegirà al final de l’array \$treballador passat com a primer paràmetre. Aquests dos nous elements tindran índexs numèrics començant pel 0 i continuant per l’1, i el seu contingut serà el dels dos últims paràmetres de la funció, és a dir, índex 0 => “C/Gran, 13” i índex 1 => “Barcelona”.

Si l’array fos numèric, continuaria amb la numeració de l’últim element de l’array, és a dir, si per exemple l’índex de l’últim element de l’array fos 4, es crearia un nou element amb índex 5, però com en aquest cas és un array associatiu, comença per l’índex 0 augmentat en una unitat de manera seqüencial la resta d’índex a afegir.

Mitjançant aquesta funció podem afegir tants elements com vulguem simplement afegint cada contingut dels elements com un nou paràmetre de la funció.

Aquesta funció retornarà l’array \$treballador passat com a primer paràmetre amb els nous elements afegits al final.

En el cas de la funció **array_unshift(\$treballador, “C/Gran, 13”, “Barcelona”)**, farà i retornarà exactament el mateix que la funció array_push, però afegint els nous elements a l’inici de l’array.

La funció **array_pop(\$treballador)** eliminarà l’últim element de l’array \$treballador passat com a paràmetre.

Aquesta funció retornarà l’array \$treballador passat com a primer paràmetre sense l’element eliminat.

En el cas de la funció **array_shift(\$treballador)**, farà i retornarà exactament el mateix que la funció array_pop, però eliminant el primer element de l’array.

4.4.2 Funcions per consultar els elements dels arrays

Dins de les funcions integrades de PHP, hi ha un conjunt de funcions que serveixen per consultar els elements dels arrays. D’aquestes nosaltres veurem les funcions **in_array**, **array_key_exists**, **array_keys** i **array_values**.

Funció in_array

Veiem el següent exemple:

```
1 <?php
2
3 $treballador = array("Nom"=>"Pep", "Departament"=>"RR.HH", "Edat"=>48, "Nif"=>
4   00000000T");
5
6 echo "<h3>FUNCIO IN_ARRAY</h3>";
/* La funció in_array(valor, array), retorna verdader si valor es troba en l'
   array*/
```

```

7
8 $valor = "Pep";
9 if (in_array($valor, $treballador)) {
10     echo "Pep existeix!!!";
11 } else {
12     echo "Pep no existeix!!!";
13 }
14 echo "<hr/>"; //Inserim línia html

```

En aquest exemple, la funció **in_array(\$valor, \$treballador)** retornarà verdader si “Pep”, el valor passat com a primer paràmetre, és contingut d'un dels elements de l'array \$treballador passat com a segon paràmetre. En aquest cas retornarà verdader, ja que “Pep” és el contingut de l'element amb índex “Nom”. En cas contrari retornaria fals.

Funció array_key_exists

Veiem el següent exemple:

```

1 <?php
2
3 $treballador = array("Nom"=>"Pep", "Departament"=>"RR.HH", "Edat"=>48, "Nif"=>
4     00000000T);
5
6 echo "<h3>FUNCió ARRAY_KEY_EXISTS</h3>";
7 /* La funció array_key_exists(clau, array), retorna verdader si la clau es
8    troba en l'array*/
9
10 $valor = "Nom";
11 if (array_key_exists($valor, $treballador)) {
12     echo "La clau nom existeix!!!";
13 } else {
14     echo "La clau nom no existeix!!!";
15 }
16 echo "<hr/>"; //Inserim línia html

```

En aquest exemple, la funció **array_key_exists(\$valor, \$treballador)** retornarà verdader si “Nom”, el valor passat com a primer paràmetre, és un dels índexs de l'array \$treballador passat com a segon paràmetre. En aquest cas retornarà verdader, ja que “Nom” és índex de l'array. En cas contrari retornaria fals.

Funció array_keys

Veiem el següent exemple:

```

1 <?php
2
3 $treballador = array("Nom"=>"Pep", "Departament"=>"RR.HH", "Edat"=>48, "Nif"=>
4     00000000T);
5
6 echo "<h3>FUNCió ARRAY_KEYS</h3>";
7 /* La funció array_keys ( vector [, valor_clau] ) retorna un array amb tots els
8    índexs de l'array passat
9    * com a paràmetre. Si especifiquem el paràmetre opcional, només retornarà els
10       índexs que continguin
11       * aquest valor */
12
13 $indexes = array_keys($treballador,"RR.HH"); //Accedim als índexs
14

```

```

12 //Mostrem contingut
13 foreach ($indexs as $index => $valor) {
14     echo "$index...$valor<br/>";
15 }
16 echo "<hr/>"; //Inserim línia html

```

En aquest exemple, la funció **array_keys(\$treballador,“RR.HH”)** retornarà un array numèric amb els índexs dels elements de l'array \$treballador passat com a primer paràmetre, el contingut dels quals sigui igual a “RR.HH”, el valor passat com a segon paràmetre opcional.

Si el que volem que ens retorne un array amb tots els índexs de l'array passat com a primer paràmetre, el que farem és no afegir el segon paràmetre opcional.

Funció array_values

Veiem el següent exemple:

```

1 <?php
2
3 $treballador = array("Nom"=>"Pep", "Departament"=>"RR.HH", "Edat"=>48, "Nif"=>
4     "00000000T");
5
6 echo "<h3>FUNCIÓ ARRAY_VALUES</h3>";
7 /* La funció array_values(vector) retorna un array amb tots els valors de l'
8    array passat com a paràmetre*/
9
10 $valors = array_values($treballador); //Accedim als valors
11
12 //Mostrem contingut
13 foreach ($valors as $index => $valor) {
14     echo "$index...$valor<br/>";
15 }
16 echo "<hr/>"; //Inserim línia html

```

En aquest exemple, la funció **array_values(\$treballador)** retornarà un array numèric amb els continguts dels elements de l'array \$treballador passat com a paràmetre.

4.4.3 Funcions per crear nous arrays a partir d'altres arrays

Dins de les funcions integrades de PHP, hi ha un conjunt de funcions que serveixen per crear nous arrays a partir d'altres arrays. D'aquestes nosaltres veurem les funcions **array_merge** i **shuffle**.

Funció array_merge

El que fa la funció **array_merge** és combinar dos arrays o més ja existents i retornar un nou vector amb aquesta combinació.

Veiem el següent exemple:

```

1  <?php
2
3  /* Combinem dos arrays o més que ja existeixen. Si es tracta d'arrays
4   * associatius amb el mateix índex, els valors del nou array seran els de
5   * l'últim array passat com a paràmetre. Si no són associatius, referà la
6   * numeració dels índexs donant cabuda als dos arrays. Si són de diferents
7   * tipus o associatius amb diferent índex combinàrà els diferents índexs
8   * tal i com s'ha explicat. */
9
10 //Creem tres vectors
11 $treballador1 = array("Nom"=>"Pep", "Departament"=>"RR.HH", "Edat"=>48, "Nif"=>
12     "00000000T");
13 $treballador2 = array("Nom"=>"Pere", "Departament"=>"Comercial", "Edat"=>50, "Nif"=>
14     "11111111A");
15 $treballador3 = array("Bob", "Comercial", 30, "2222222B");
16
17 echo "<h3>FUNCIÓ MERGE EXEMPLE 1</h3>";
18 //Combinem $treballador2 amb $treballador1
19 $nouArray = array_merge($treballador2, $treballador1);
20
21 //Mostrem contingut
22 foreach ($nouArray as $index => $valor) {
23     echo "$index...$valor<br/>";
24 }
25 echo "<hr/>"; //Inserim línia html
26
27 echo "<h3>FUNCIÓ MERGE EXEMPLE 2</h3>";
28 //Combinem $treballador3 amb $treballador3
29 $nouArrayNum = array_merge($treballador3, $treballador3);
30
31 //Mostrem contingut
32 foreach ($nouArrayNum as $index => $valor) {
33     echo "$index...$valor<br/>";
34 }
35 echo "<hr/>"; //Inserim línia html
36
37 echo "<h3>FUNCIÓ MERGE EXEMPLE 3</h3>";
38 //Combinem $nouVector amb $treballador3
39 $nouArray = array_merge($nouArray, $treballador3);
40
41 //Mostrem contingut
42 foreach ($nouArray as $index => $valor) {
43     echo "$index...$valor<br/>";
44 }
45 echo "<hr/>"; //Inserim línia html

```

En aquest script, podem veure tres exemples amb les diferents maneres de combinar arrays mitjançant la funció **array_merge**:

- En el primer exemple la funció **array_merge(\$treballador2, \$treballador1)** retorna un nou array resultat de combinar els dos arrays passats per paràmetre \$treballador2 i \$treballador1. Com aquest dos arrays són associatius amb els mateixos índex, l'array que retornarà la funció, tindrà els mateixos índexs que qualsevol dels dos arrays passats per paràmetre, i els continguts associats a aquests índexs, els continguts dels elements de l'últim array passat per paràmetre, en el nostre cas l'array \$treballador1.
- En el segon exemple la funció **array_merge(\$treballador3, \$treballador3)** retorna un nou array resultat de combinar els dos arrays passats per paràmetre \$treballador3 i \$treballador3, que com podem veure són el mateix array. Com aquests dos arrays són numèrics, la longitud del nou array que retornarà la funció, serà la suma de les longituds dels dos arrays passats per

paràmetre, en el nostre cas la longitud serà $4 + 4 = 8$, i pel que fa als índexs, seran nous, començant per 0 i augmentant en una unitat la resta d'índexs de manera seqüencial, per tant en el nostre cas l'últim índex serà 7. Els continguts associats a aquests índexs, seran els continguts dels dos arrays passat per paràmetre en l'ordre amb què s'han afegit, és a dir, primer els elements del primer paràmetre i després els del segon, en l'ordre que es troben dins dels arrays.

- En l'últim exemple la funció **array_merge(\$nouArray, \$treballador3)** retorna un nou array resultat de combinar els dos arrays passats per paràmetre \$nouArray, què és l'array que ha retornat la funció en el primer exemple, i \$treballador3. En aquest cas es vol combinar un array associatiu \$nouArray amb un array numèric \$treballador3, per tant, l'array que retornarà la funció, tindrà en les primeres posicions els elements de l'array passat com a primer paràmetre, amb els seus índexs i continguts, i en les següents posicions els elements de l'array passat com a segon paràmetre, també amb els seus índexs i continguts.

Funció shuffle

Veiem el següent exemple:

```

1 <?php
2
3 echo "<h3>FUNCIÓ SHUFFLE</h3>";
4 /* La funció shuffle retorna l'array passat com a paràmetre, però amb un
5 * nou ordre dels elements. Barreja els elements de l'array passat com a
6 * paràmetre de manera aleatòria. Retorna true si tot ha sortit bé i false
7 * en cas contrari */
8
9 $treballador1 = array("Nom"=>"Pep", "Departament"=>"RR.HH", "Edat"=>48, "Nif"=>
10   "00000000T");
11
12 if (shuffle($treballador1)) { //Barregem valors i si èxit...
13   echo "La barreja ha estat un èxit!!!<br/><br/>";
14   //Mostrem contingut
15   foreach ($treballador1 as $index => $valor) {
16     echo "$index...$valor<br/>";
17   }
18 } else { //No èxit...
19   echo "La barreja ha fracassat!!!";
}
```

En aquest exemple, la funció **shuffle(\$treballador1)** retornarà l'array \$treballador1 passat com a primer paràmetre però amb els seus elements ordenats aleatoriament de manera diferent. Si el procés ha estat exitós, a més de l'array retornarà verdader, si no, retornarà només fals.

4.5 Funcions d'usuari

Com hem vist en l'apartat 4.2 “Funcions integrades de PHP”, les **funcions d'usuari** són aquelles que crea l'usuari (programador).

En aquest apartat veurem com es crea una funció d'usuari i com es crida per ser executada, quin és l'àmbit de les variables en aquest tipus de funcions i perquè serveix i com niuar funcions d'usuari.

4.5.1 Definició de les funcions d'usuari

Veiem el següent exemple:

```

1  <?php
2
3  //Creació d'una funció sense paràmetres
4  function socFuncio() {
5      echo "<h3>Sóc una funció.</h3>";
6  }
7
8  //Crida a la funció
9  socFuncio(); //Imprimirà per pantalla "Sóc una funció"
10
11 echo "<hr/>";
12
13 //Creació d'una funció amb paràmetres
14 function mostrarMissatge($missatge) {
15     echo "$missatge <br/>";
16 }
17
18 //Crida a la funció. En aquest cas la cridem més d'un cop
19 mostrarMissatge("Això és un missatge.");
20 mostrarMissatge("Això és un nou missatge.");
21 mostrarMissatge("Aquest és l'últim missatge");
22
23 echo "<hr/>";
24
25 //Creació d'una funció amb retorn
26 function sumar($primerOperand, $segonOperand) {
27     $suma = $primerOperand + $segonOperand;
28     return $suma; //No cal crear una variable per retornar el valor, o podem
29         fer directament com return $primerOperand + $segonOperand
30 }
31
32 //Crida a la funció. En aquest cas la funció no mostra res per pantalla, només
33     guardem en la variable $resultat, el valor que ens retorna.
34 $resultat = sumar(3, 4);
35
36 //Mostrem resultat
37 echo $resultat; //També ho podem mostrar com a echo sumar(3,4);
38
39 echo "<hr/>";

```

Seguint l'script d'exemple, per crear una funció d'usuari el que hem de fer és escriure la paraula reservada **function** seguida del nom que vulguem donar a la funció com la funció **socFuncio** de l'exemple. Després escrivim els parèntesis () on afegirem els paràmetres, si és el cas, si no els deixarem buits, com en el cas de la funció **socFuncio()**. Finalment entre les claus escriurem totes les accions que volem que es realitzin quan cridem a la funció. En el cas de la funció **socFuncio()** l'acció és **echo “<h3>Sóc una funció.</h3>“**.

Per cridar a la funció quan vulguem executar-la, el que fem és escriure el nom de la funció, passant-li els paràmetres escaients segons la signatura de la funció. En

Seguint les bones pràctiques de programació, els noms de les funcions haurien d'escriure's en minúscula i, si estant formats per més d'una paraula, en notació camell o separant les paraules amb guíó baix (_). Aquesta segona és la notació que més es fa servir en PHP.

el cas de l'script d'exemple, cridem a la funció **socFuncio()**, on com podeu veure, no li hem de passar cap paràmetre per això els parèntesis estan buits.

En el cas de la funció de l'script **mostrarMissatge(\$missatge)**, sí que li hem de passar un paràmetre, el paràmetre **\$missatge**. El fet de passar-li un paràmetre a una funció, li dona dinamisme a l'script; ja que cada cop que la cridem li podem passar un paràmetre diferent i per tant el resultat de la seva execució també serà diferent, tal com es pot veure en l'script d'exemple.

Finalment, en l'script d'exemple tenim la funció **sumar(\$primerOperand, \$segonOperand)**. Com es pot veure, a aquesta funció li passem dos paràmetres **\$primerOperand** i **\$segonOperand**, i el que fa és sumar aquests dos paràmetres i **retornar** el valor de la suma. Si volem que una funció retorna un valor, el que farem serà escriure la paraula **return** seguida del valor que volem que es retorna. Així doncs quan cridem una funció que retorna un valor, el resultat de la seva execució serà el valor retornat. Això ho podeu veure en l'script d'exemple, amb la crida a la funció **sumar(3, 4)**, on el valor que retorna se li assigna a la variable **\$resultat**.

4.5.2 Àmbit de les variables en les funcions d'usuari

Les variables declarades dins d'una funció són només per a l'àmbit de la funció. Fora de la funció no es reconeixen. De la mateixa manera una variable declarada fora d'una funció és només per a l'àmbit extern a la funció. Dins de la funció no es reconeix.

Veiem el següent exemple:

```

1 <?php
2
3 /* Les variables declarades dins d'una funció són només per a l'àmbit
4 * de la funció. Fora de la funció no es poden utilitzar com si fossin
5 * les mateixes variables. */
6
7 function mostrarNom() {
8     $nom = "Pere Bernat";
9     return $nom;
10 }
11
12 //Intentem mostrar el resultat de la variable $nom. No es mostra res perquè
13 //no està declarada fora de l'àmbit de la funció.
14 echo "El contingut de la variable \$nom és $nom <br/>";
15
16 echo "<hr/>";
17
18 /* De la mateixa manera una variable declarada fora d'una funció és només
19 * per a l'àmbit extern a la funció. Dins de la funció no es pot utilitzar
20 * com si fos la mateixa variable. */
21 $cognom = "Sastre";
22
23 //Intentem utilitzar la variable $cognom dins de la funció. No mostra res
24 //perquè està declarada fora de l'àmbit de la funció.
25 function mostrarCognom1() {
26     echo "El contingut de la variable \$cognom és $cognom";
27 }
28

```

```

29  mostrarCognom1(); //No mostra res
30
31  echo "<hr/>";
32
33  /* Per utilitzar una variable dins d'una funció declarada fora del seu
34   * àmbit, el que fem és declarar-la com a global dins de la funció. D'aquesta
35   * manera la funció recordarà el valor que té la variable fora d'ella. */
36
37  //Intentem utilitzar la variable $cognom dins de la funció. Mostra el seu valor
38  //perquè l'hem declarat com a global dins la mateixa funció.
39  function mostrarCognom2() {
40      global $cognom; //Declarem com a global
41      echo "El contingut de la variable \$cognom és $cognom";
42  }
43
44  mostrarCognom2(); //Mostrem Sastre.
45
46  echo "<hr/>";

```

La variable \$nom de dins la funció **mostrarNom()**, no és la mateixa que la variable \$nom externa a ella, encara que tinguin el mateix nom, són dos variables diferents, la primera pertany a l'àmbit de la funció i la segona a l'àmbit extern a la funció. El mateix passa amb la variable \$cognom de dins la funció **mostrarCognom1()**.

Si volem que una variable de l'àmbit d'una funció sigui reconeguda fora de la funció, la declarem com a **global** dins de la funció com podeu veure en el cas de la variable \$cognom de la funció **mostrarCognom2()**.

Per últim tenim el cas de les variables de l'àmbit d'una funció declarades com **static**. Aquestes variables només existeixen dins l'àmbit de la funció on les declarem i no perden el seu valor quan finalitza l'execució de la funció on s'han declarat. Veiem el següent exemple:

```

1  <?php
2
3  /* Les variables declarades com estàtiques només existeixen dins l'àmbit de la
4   * funció on les declarem, però no perden el seu valor quan l'execució del
5   * programa abandona l'àmbit de la funció. */
6
7  //Funció que ens ha de permetre crear un llistat numerat
8
9  function crearLlistatNumerat($missatge) {
10     static $numeracio = 0; //Inicialitzem
11     // $numeracio=0; //Inicialitzem
12     $numeracio++; //Augmentem numeració
13     echo "<h3>$numeracio $missatge</h3>"; //Mostrem punt de la llista+missatge
14 }
15
16 //Cridem funció per crear llista de dos punts
17 crearLlistatNumerat("Primer punt.");
18 crearLlistatNumerat("Segon punt.");

```

Cada cop que cridem a la funció **crearLlistatNumerat(\$missatge)**, aquesta crea un punt d'una llista numerada formada pel número corresponent de la llista i el missatge passat per paràmetre. Per tant, perquè es pugui crear la llista numerada afegint els números corresponents, és necessari guardar el valor de l'últim número afegit perquè la funció sàpiga quin nou número ha d'afegir quan la tornem a cridar. En aquest cas, això ho fem declarant com estàtica la variable **\$numeracio**, què és on guardarem l'últim número afegit a la llista i gràcies al fet que és estàtica mantindrà l'últim valor que se li hagi assignat.

4.5.3 Funcions d'usuari niuades

Les funcions es poden niuar, això vol dir que una funció pot cridar a altres funcions com a part de les accions que s'executaran quan la cridem. Això ho fem per simplificar el codi, és a dir, si jo tinc un bloc de codi que es repeteix dins d'una o diferents funcions, puc crear una altra funció amb aquest bloc de codi i simplement cridar-la dins de les funcions que em faci falta, tants cops com sigui necessari, per no tornar a escriure el mateix codi.

Veiem el següent exemple:

```

1 <?php
2
3 /* Les funcions poden cridar a altres funcions */
4
5 //Funció que ens retorna l'increment en una unitat d'un número passat com a
6 //paràmetre.
7 function sumemU($num) {
8     $num += 1;
9     echo "$num<br/>"; //Mostra increment
10    return $num;
11 }
12
13 //Funció que ens mostra per pantalla l'increment en 2 unitats d'un número
14 //passat com a paràmetre. Mitjançant aquesta segona funció accedim a la primera
15
16 function sumemDos($num) {
17     $resultat = sumemU($num); //Increment en 1
18     sumemU($resultat); //Increment en 2
19 }
20
21 sumemDos(0);

```

En aquest script d'exemple, tenim la funció **sumemU(\$num)**, que suma una unitat al valor \$num passat com a paràmetre i mostra el resultat per pantalla. La funció **sumemDos(\$num)** crida a la funció **sumemU(\$num)** dos cops per incrementar en dues unitats el valor \$num passat com a paràmetre i mostrar per pantalla els resultats dels increments. En lloc de repetir el codi dos cops, crida a la funció **sumemU(\$num)** dos cops.

4.6 Pas d'arguments en les funcions

El pas d'arguments en les funcions es pot fer de dues maneres: **per valor** i **per referència**. Anem a veure aquestes dues modalitats amb dos exemples:

```

1 <?php
2
3 /* Pas de paràmetres per valor. */
4
5 //Funció que suma 5 a un número passat com a paràmetre i retorna el resultat
6 function afegirCincV($num) {
7     $num += 5;
8     return $num;
9 }

```

```

10
11 $elMeuNumero = 10;
12 echo afegirCincV($elMeuNumero); //Suma 5 a elMeuNumero i mostra 15
13 echo "<br/>";
14 echo $elMeuNumero; //Mostra 10
15
16 echo "<hr/>";

```

En aquest script d'exemple, per un costat tenim la funció **afegirCincV(\$num)**, que retorna el resultat de sumar 5 al valor \$num passat com argument. Per un altre costat creem la variable **\$elMeuNumero** a la qual li assignem un valor inicial 10 què és el que passarem com a argument en la funció afegirCincV quan la cridem a continuació.

Com podeu veure, mostrem per pantalla el retorn de la funció i es mostra 15 ($10 + 5$), però en mostrar el valor de la variable **\$elMeuNumero**, aquest continua sent el seu valor inicial, és a dir, 10. Això és així perquè hem passat l'argument per valor, i per tant, per molt que modifiquem el valor inicial de l'argument en l'execució de la funció, en finalitzar aquesta execució, el valor de la variable passada com argument tornarà a ser l'inicial.

```

1 <?php
2
3 /* Pas de paràmetres per referència */
4
5 function afegirCincR(&$num) {
6     $num += 5;
7     return $num;
8 }
9
10 $elMeuNumero = 10;
11 echo afegirCincR($elMeuNumero); //Suma 5 a elMeuNumero i mostra 15
12 echo "<br/>";
13 echo $elMeuNumero; //Mostra 15
14
15 echo "<hr/>";

```

Igual que en l'altre script d'exemple, per un costat tenim la funció **afegirCincR(&\$num)**, que retorna el resultat de sumar 5 al valor \$num passat com argument. Per un altre costat de nou creem la variable **\$elMeuNumero** a la qual li assignem un valor inicial 10 què és el que passarem com a argument en la funció afegirCincR quan la cridem a continuació.

També mostrem per pantalla el retorn de la funció i es mostra 15 ($10 + 5$), com en l'altre script, però en mostrar el valor de la variable **\$elMeuNumero**, a diferència de l'altre script d'exemple, el valor que es mostra, és el nou valor, és a dir, 15. Això és així perquè hem passat l'argument per referència, i per tant, en modificar el valor inicial de l'argument en l'execució de la funció, en finalitzar aquesta execució, el valor de la variable passada com argument passa a ser el nou valor que se li ha assignat.

Per indicar que un argument el passem per referència, fem servir el símbol **&** precedint al nom de l'argument.

Finalment, també hem de tenir en compte que als arguments els hi podem assignar un valor per defecte. Veiem com fer-ho en el següent script d'exemple:

```

1 <?php
2
3 /* Pas de paràmetres opcionals. */
4
5 //Funció que ens permet canviar el format del títol HTML d'un text passat com a
6 //paràmetre o bé deixar el format del //títol HTML per defecte (en el
7 //nostre cas H1) segons el valor corresponent al format del títol.
8 function formatTitolHTMLH1($text, $titol = "1") { //Dos paràmetres: text i
9     format del ittol HTML H1
10    echo "<h$titol>$text</h$titol>";
11 }
12
13 formatTitolHTMLH1("Títol H1"); //Format per defecte del títol
14 formatTitolHTMLH1("Títol H2", "2");
15 formatTitolHTMLH1("Títol H3", "3");
16 formatTitolHTMLH1("Títol H4", "4");
17
18 echo "<hr/>";

```

En aquest script d'exemple, li assignem el valor “1” al segon argument **\$titol** de la funció **formatTitolHTMLH1**. Com podeu veure, per assignar un valor per defecte a un argument, simplement li hem d'assignar aquest valor mitjançant l'operador d'assignació “=”.

Que aconseguim assignant un valor per defecte a un argument? Que cada cop que cridem a la funció, si no li passem aquest argument, el sistema interpreta que el valor passat és el valor per defecte assignat a l'argument. En el cas de l'script d'exemple, quan cridem a la funció per primer cop, no li passem el segon argument, per tant, el sistema ja sap que aquest segon argument és 1 i imprimirà “Títol H1” en format de títol H1 d'HTML.

Podem modificar aquest valor per defecte? Sí, simplement passant l'argument al qual li hem assignat el valor per defecte, amb un valor diferent de l'assignat per defecte. En l'script d'exemple, això ho podem veure en les següents crides a la funció **formatTitolHTMLH1**. En totes elles passem el segon argument amb un valor diferent del valor per defecte.

4.7 Gestió d'errors

Per comprovar quins errors es produeixen en els nostres scripts, el que farem és interpretar les notificacions que se'ns mostren en el navegador quan es produeix algun d'aquests errors.

Interpretem l'error de la notificació de la figura 4.6:

FIGURA 4.6. Notificació d'error Warning

Warning: Undefined variable \$nom in /var/www/html/M09ASIX/_10Funcions/_3VisibilitatVariables.php on line 14
El contingut de la variable \$nom és

El primer que veiem és una paraula amb negreta que ens indica quin tipus d'error s'ha produït, en el nostre cas **Warning**. Aquesta paraula fa referència a una