3.1 Definició i crides a funcions

Com qualsevol llenguatge de programació, PHP permet l'ús de funcions.

- Definició d'una funció de PHP:

```
function nom_funcio($argument1, $argument2, ...){
        //instruccions;
        [return valor_de_retorn;]
}
```

Exemple 1: Funció que retorna un valor

```
<?php
        function f1(){
                $var=10;
                return $var;
        }
?>
```

Per cridar a aquesta funció farem:

```
<?php
        $a=f1();
        echo $a;
?>
```

Exemple 2: funció que no retorna cap valor

```
<?php
	function f2(){
		$var=10;
		echo $var;
	}
?>
```

Per cridar a aquesta funció farem:

```
<?php
	f2();
?>
```

Exemple 3:

```
<?php
	$var=10;
	function f3(){
		global $var;
		echo $var;
	}
?>
```

Per cridar a aquesta funció farem:

```
<?php
	f3();
?>
```

Observacions:

- La clàusula *return* és opcional.
- Per tant, un procediment en realitat serà una funció que no retorna res.
- Les variables declarades dins de la funció només són accessibles des de dins de la funció.

- Podem accedir a qualsevol variable declarada fora de qualsevol funció, utilitzant la clàusula *global $nom_var*

3.2 Pas de variables per referència

- Per defecte quan passem variables a una funció en PHP es passen per valor.
- Podem passar les variables per referència utilitzant l'operador & en la capçalera de la funció.

Exemple: pas per referència

```
<?php

        function  incrementa(&$var){

                $var++;

        }

        ...

        $a=10;

        echo incrementa($a);

?>
```

3.3 Recursivitat

- PHP suporta la programació recursiva, això vol dir que podem crear funcions que es criden a si mateixes.

Exemple recursivitat

```
<?php
function factorial($numero){
        if($numero==1){
                return $numero;
        }else{
                return $numero * factorial($numero-1)
        }
  }

  echo “El factorial de 7 és: “.factorial(7).”<br>”;
?>
```

3.4 Funcions: include(..) i require(..)

- Podem crear arxius a mode de llibreria o paquet que continguin les funcions que hem definit prèviament per tal d'utilitzar-les sempre que vulguem.
- Aquestes funcions s'han de crear en un arxiu de text i es recomana que sigui en un arxiu amb extensió .php o .inc
- Per tal d'importar des dels nostres scripts PHP aquestes funcions utilitzarem les funcions *include(...)* o *require(...)*

Exemples:

```
include ("llibrerira.php");

o

require ("llibreria.php");
```

- Observacions:
  - El funcionament d'aquestes 2 funcions no és exactament el mateix i es recomana només l'ús d' *include(..)* ja que en cas d'error, aquesta només emet un warning, i l'altre emet un error
  - Cal anar en compte amb l'ús d'aquestes funcions, si tenim fitxers que es criden entre ells i necessiten importar llibreries, podem arribar a un bucle infinit on els fitxers s'estan constantment important l'un a l'altre.
  - Per evitar aquests errors existeixen les funcions *include_once(...)* i *require_once(...)*, amb les quals si s'intenta importar diversos cops un fitxer, PHP ignora les crides.

3.5 Funcions pel tractament d'strings

3.5.1 Introducció

Les principals característiques dels strings:

- Un string és una cadena de caràcters.
- Podem declarar strings utilitzant cometes dobles i cometes simples, però cal saber que les dobles evaluen la expressió i les simples no.
- El símbol per concatenar strings és el punt.
- Un string pot ser tractat com una unitat, o també posició a posició.
- Per accedir a cada posició d'un string utilitzarem els simbols { }.
  Exemple:

```
<?php
      $cadena="Hola";
       $cadena{0}="C";
       echo $cadena;  //mostra per pantalla Cola
?>
```

3.5.2 Mida d'una cadena

a) **Funció strlen($cadena)**
   - Retorna la mida en caràcters de la cadena que li passem com a paràmetre.

   Exemple:

```
<?php
    $cadena="Aquesta cadena té moltes lletres";
    $num_caracters=strlen($cadena);
    echo $num_caracters;
?>
```

3.5.3 Buscar la posició d'un caràcter o d'una subcadena

a. Funció strpos($cadena, $subcadena)

- Retorna la casella on troba la subcadena dins de la cadena passada.
- Sempre retorna la primera ocurrència.
- Si no troba la subcadena retorna un *false*.
- Exemple:

```
<?php

    $cadena="Aquesta cadena té moltes lletres";

    echo "La primera ocurrència de moltes és ".strpos($cadena,"moltes");

?>
```

b. Funció strrpos($cadena, $subcadena)

- És igual que strpos(...) però començant pel final (strpos reverse).

Altres funcions equivalents són: *ereg(..);* i *eregi(..);*

3.5.4 Comparació de cadenes

a) Funció strcmp($cad1, $cad2)

- Retorna un 0 si les 2 cadenes són iguals.
- Retorna <0 si la primera cadena és més petita.
- Retorna >0 si la primera cadena és més gran.

Exemple

```
<?php

    echo strcmp("hola","adeu");

?>
```

- Observació:
  També podem comparar 2 cadenen amb l'operador de comparació ==

3.5.5.- Selecció de subcadenes

a) **Funció substr($cadena, inici, [tamany])**

- Retorna una subcadena de caràcters d'una cadena a partir d'una posició especificada fins al final o del tamany especificat.
- La cadena original no pateix cap modificació.
- Exemple:

```
<?php

    $cadena=”PHP és un llenguatge fàcil”;

    echo substr($cadena,0,3);   //mostra per pantalla “PHP”;

    echo substr($cadena,21); //mostra per pantalla “fàcil”;

?>
```

- Observacions:
  - Si inici és negatiu, significa que comencem per darrera.
  - Si tamany és negatiu, significa que acabem en aquella posició contant pel final (realment no indica el tamany).

    Exemple:

```
<?php

        $cadena=”PHP és un llenguatge fàcil”;

        echo substr($cadena,-5);  //mostra els últims 5 caràcters

        //mostra des de posició 10 fins 3 abans d'arribar al final

        echo substr($cadena,10, -3);

        ?>
```

3.5.6 Eliminació d'espais en blanc dins de cadenes

a) Funció trim($cadena)

- Elimina els espais en blanc, les tabulacions i els salts de línia que hi ha a l'inici i al final de la cadena.
  Exemple:

```
<?php
        echo trim("     espais a l'inici i al final      ");
?>
```

b) Funció ltrim($cadena)

- Elimina els espais en blanc que hi ha a l'inici de la cadena.
  Exemple:

```
<?php
        echo ltrim("        cadena amb espais a l'inici");
?>
```

3.5.7.- Substitució de cadenes

a) Funció str_replace($antiga, $nova, $cadena)

- Aquesta funció substitueix la cadena *$antiga* per la cadena *$nova* dins de *$cadena*.
- Exemple:

```
<?php
        $cadena="PHP és fàcil";
        $antiga="és fàcil";
        $nova="no és dificil";
        echo str_replace($antiga, $nova, $cadena);
?>
```

Altres funcions equivalents són: *ereg_replace(..); eregi_replace(..);*

3.5.8.- Conversió majúscules i minúscules

a) **Funció strtolower($cadena)**
- Passa la cadena a minúscules

b) **Funció strtoupper($cadena)**
- Passa la cadena a majúscules

3.5.9.- Dividir una cadena segons un caràcter o patró

- La funció split(...) permet dividir una cadena segons un caràcter o patró, i retorna el resultat en un array.
- Quan arribem al tema d'arrays veurem com utilitzar-la.

Exercici 1

Creeu una funció que dibuixi una taula en HTML. Aquesta funció ha de rebre 2 paràmetres que siguin la quantitat de files i columnes de la taula. Cal posar text en cada una de les cel·les.

Exercici 2

Busqueu per què serveix la funció *str_word_count(...)* i poseu un exemple on es vegi el seu funcionament.

- Podeu trobar ajuda a http://php.net/manual/en/function.str-word-count.php

Exercici 3

Busqueu per què serveix la funció *levenshtein(...)* i poseu un exemple on es vegi el seu funcionament.

Exercici 4

Busqueu que és l'operador ternari de PHP i poseu un exemple on es vegi el seu ús.

Exercici 5

a. Utilitzeu la funció strcmp(...) per comparar 2 paraules
b. Creeu una funció anomenada *comparaParaules(...)* que rebi 2 paraules i retorni -1 si la 1a paraula és més petita que la 2a, retorni 0 si són iguals i retorni +1 si la 1a paraula és més gran que la 2a.

- Observació: *Recordeu que aquestes comparacions es fan mirant els codis ASCII de cada caràcter i per tant qualsevol lletra en majúscules sempre serà més petita que qualsevol lletra en minúscules.*

Exercici 6

Demostreu el funcionament del pas de variables per referència.

Exercici 7

Tot i que encara no sabem treballar amb arrays, expliqueu què creieu que fa aquest bloc de codi

```
function funcioMultiplesReturns($v1,$v2,$v3){
  $v1="variable1";
  $v2="variable2";
  $v3="variable3";
  return array($v1,$v2,$v3);
}
```

Exercici 8

Creeu la funció *comprova_email(...)* que rep una cadena de caràcters que conté una adreça de correu electrònic, i li fa les següents comprovacions:

- La converteix a minúscules
- Li elimina tots els espais en blanc
- Comprova si té el caràcter @
- Compta el número de caràcters

Finalment, si la mida és menor de 75 caràcters i conté l'@, l'adreça és vàlida i per tant ha de retornar un TRUE, sinó ha de retornar un FALSE.

Exercici 9

A l'exercici anterior afegiu-li un nou control sobre l'adreça de correu mitjançant la funció *checkdnsrr(...)* , que donat el domini d'una adreça de correu, permet veure si aquest domini existeix o no.
Aquí tens un exemple de com fer servir aquesta funció:

```
if (checkdnsrr("nom_de_domini","MX")){
      echo "Aquest domini existeix";
}else{
      echo "Aquest domini inventat No existeix";
}
```

Exercici 10

Poseu la funció anterior en un fitxer anomenat *funcions.php* i proveu el funcionament de la funció *include(..)*

Exercici 11

Existeixen moltes funcions per treballar amb dates, proveu:
a. Què fa la funció *time(...)*?
b. Proveu el funcionament de la funció *date(...)* sabent que li podeu passar el format. Exemple: *echo date("G:i l Y");*
Les opcions més interessants de format són:

Y=any, F=Mes en format text, m=mes en format numèric,
M=Mes en text abreviat, l=dia de la setmana en text,
d=dia numèric del mes, G=hora del dia format 24h, i=minut de l'hora...

- Podeu trobar més informació a la plana http://php.net/manual/es/function.date.php