- Els noms de paràmetre se separen dels valors amb el caràcter =.
- Se substitueixen els caràcters especials seguint la taula següent:
  - El caràcter ' ' (espai en blanc) es converteix en +.
  - Els caràcters no alfanumèrics i els caràcters especials, com els usats per a codificació (+, etc.), es representen de la forma `%HH`, on `HH` representa el valor hexadecimal del codi ASCII del caràcter.
  - Els salts de línia es representen com a `%0D %0A`.

5.1.4. Redireccions

Podem reexpedir el client a una pàgina diferent des d'un programa CGI. Per a això només cal no tornar el codi HTML estàndard precedit del `Content-type`, sinó que hauríem de tornar un camp de codi d'estat seguit de la localització de la nova pàgina com en l'exemple:

```
#include <stdio.h>
int main()
 {
    printf("Status: 302\r\n");
    printf("Location: nova.html\r\n");
    exit(1);
 }
```

5.2. PHP

PHP (la sigla del qual responen a un acrònim recursiu, *hypertext preprocessor*) és un llenguatge senzill, de sintaxi còmoda i similar a la d'altres llenguatges com Perl, C i C++. És ràpid, interpretat, orientat a objectes i multiplataforma. Per a ell es troba disponible una multitud de llibreries. PHP és un llenguatge ideal tant per a aprendre a desenvolupar aplicacions web com per a desenvolupar aplicacions web complexes. PHP afegeix a tot això l'avantatge que l'intèrpret de

PHP, els diversos mòduls i gran quantitat de llibreries desenvolupades per a PHP són de codi lliure, amb la qual cosa el programador de PHP disposa d'un arsenal impressionant d'eines lliures per tal de desenvolupar aplicacions.

PHP se sol utilitzar conjuntament amb Perl, Apache, MySQL o PostgreSQL en sistemes Linux, formant una combinació barata (tots els components són de codi lliure), potent i versàtil. Així ha estat l'expansió d'aquesta combinació que fins i tot ha merescut conèixer-se amb un nom propi *LAMP* (format per les inicials dels diversos productes).

Apache, i també alguns altres servidors web –entre els quals hi ha Roxen–, pot incorporar PHP com un mòdul propi del servidor, la qual cosa permet que les aplicacions escrites en PHP resultin molt més ràpides que les aplicacions CGI habituals.

5.2.1. Com funciona PHP

Si sol·licitem al nostre servidor una pàgina PHP, l'envia a l'intèrpret de PHP que l'executa (de fet, no es tracta més que d'un programa) i torna el resultat (generalment HTML) al servidor web, que, al seu torn, l'enviarà al client.

Imaginem que tenim una pàgina PHP amb el contingut següent:

```
<?php echo "<h1>Hola món!</h1>";?>
```

Si tenim aquest codi en un fitxer amb extensió `.php` el servidor enviarà la pàgina a l'intèrpret de PHP, el qual executa la pàgina i obté com a resultat:

```
<h1>Hola món!</h1>
```

El servidor l'enviarà al navegador client que ha sol·licitat la pàgina. El missatge apareixerà a la pantalla d'aquest últim. Veurem que PHP permet barrejar a la mateixa pàgina HTML i PHP, la qual cosa facilita notablement la feina, però d'altra banda representa un perill, ja que

complica el treball en cas que els dissenyadors de web i els programadors treballin conjuntament a les pàgines.

Tenim, en els sistemes en què hi hagi PHP instal·lat, un fitxer de configuració global de PHP anomenat `php.ini` que ens permetrà configurar alguns paràmetres globals. Convé revisar aquest fitxer, ja que encara que els valors per defecte solen ser correctes, ens pot interessar fer alguns canvis.

5.2.2. Sintaxi de PHP

Per a començar a comprendre la sintaxi del llenguatge, analitzarem un programa mínim de PHP:

```
<?php
 $MYVAR = "1234";
 $myvar = "4321";
 echo $MYVAR. "<br>\n";
 echo $myvar."<br>\n";
?>
```

L'execució d'aquest programa (la seva visualització des d'un navegador), donarà com a resultat:

```
1234<br>
4321<br>
```

El primer punt que hem de destacar és que els blocs de codi de PHP estan delimitats en HTML amb `<?php` i `?>`. Podem escriure, per tant, una pàgina HTML i incloure-hi diversos blocs d'instruccions PHP:

```
<HTML>
 <HEAD>
    <TITLE>Títol del document</TITLE>
 </HEAD>
<BODY>
 <h1>Capçalera H1</h1>
 <?php echo "Hola" ?>
 <h1>Capçalera H1 segona</h1>
 <?php
```

```
 $MYVAR = 1234;
 $myvar = 4321;
 echo $MYVAR. "<br>";
 echo $myvar."<br>";
 // Aquest programa presenta en pantalla uns números
 ?>
 </BODY>
</HTML>
```

El punt següent que convé destacar és que els noms de variables es distingeixen perquè sempre han de començar amb $, i que igual que en C/C++, són *case sensitive*, és a dir, diferencien majúscules i minúscules. Fixeu-vos també que per a concatenar text (les variables i "
") utilitzem el caràcter punt ". " i, a més, que totes les sentències acaben amb ";".

Així convé observar que les variables, malgrat ser numèriques, es poden concatenar amb un text ("
"). En aquest cas l'intèrpret converteix el valor numèric de la variable en text per poder dur a terme la concatenació.

També podem observar que hi ha un comentari dins del codi. Aquest comentari no afectarà de cap manera el programa ni serà enviat al navegador del client (de fet, el navegador client mai no rebrà codi PHP). Per a introduir comentaris en el nostre codi, tenim dues opcions:

```
// Comentari d'una sola línia

/*  Això és un comentari de diverses línies. Per
    a  això  utilitzem  aquest  altre  marcador
    d'inici i final de comentari */
```

5.2.3. Variables

PHP no necessita que declarem *a priori* la variable que utilitzarem ni el tipus d'aquesta. PHP declararà la variable i li assignarà el tipus de dades correcte en el moment en què la utiiltzem per primera vegada:

```
<?php $cadena = "Hola Món";
 $numero = 100;
 $decimal = 8.5;
?>
```

Com podem observar, les tres variables van ser definides en el moment d'assignar-los valor i no vam haver de definir tipus.

En PHP les variables poden tenir, bàsicament, dos àmbits: un de global, en què seran accessibles des de tot el codi i un altre local, en què només seran accessibles des de la funció en què les creem. Per a assignar a una variable un àmbit global n'hi haurà prou de declarar-la (en aquest cas, sí que fa falta una declaració de variable) i utilitzar la paraula reservada `global` en la declaració:

```
<?php
 global $test;
?>
```

Les variables que no qualifiquem com a globals, però que siguin definides fora de qualsevol funció, tindran com a àmbit el global.

N'hi haurà prou de definir una variable dins d'una funció. En aquest cas, el seu àmbit quedarà restringit a la funció on la declarem.

```
<?php
 global $variable; // Variable global
 $a=1; // Variable global implícita
 function suma()
 {
    $b=1; // b és una variable local
    $res=$a+$b; // res és una variable local
 }
?>
```

Podem veure que tant `a` com `variable` són variables globals, mentre que `b` i `res` són variables locals.

A més, disposem en PHP de variables de vectors o *arrays*. Aquestes són variables que poden contenir llistes d'elements, als quals accedirem per mitjà d'un índex.

```
<?php
 $mars = array(); //amb array() declarem un vector
 $mars[0]= "Mediterrània";
 $mars[1] = "Aral";
 $mars[2] = "Mort";
?>
```

Com podem veure, hem declarat la variable `mares` amb una crida a `array()`. Això indica a PHP que aquesta variable és un vector d'elements.

Per a accedir als elements individuals del vector, hem d'utilitzar el nom del vector i indicar la posició de l'element a què volem accedir entre claudàtors. En PHP els vectors comencen a numerar-se a 0.

A més de vectors amb índexs numèrics, PHP suporta vectors els índexs dels quals siguin cadenes de text:

```
<?php
 $muntanyes = array(); //amb array() declarem un vector
 $muntanyes["Everest"]= "Himàlaia";
 $muntanyes["Fitz Roy"] = "Andes";
 $muntanyes["Montblanc"] = "Alps";

 echo $muntanyes["Everest"]; // Imprimirà Himàlaia
?>
```

5.2.4. Operadors

Els operadors són símbols que s'utilitzen per a fer tant operacions matemàtiques com comparacions o operacions lògiques.

Els més habituals en PHP són:

- Operadors matemàtics:

  a) + suma diversos números: `5 + 4 = 9`.

  b) – resta diversos números: `5 – 4 = 1`.

  c) * fa una multiplicació: `3 * 3 = 9`.

  d) / fa una divisió: `10 / 2 = 5`.

  e) % torna el residu d'una divisió: `10 % 3 = 1`.

  f) ++ incrementa en 1: `$v++ (incrementa $v en 1)`.

  g) – – decreix en 1: `$v– –(decreix $v en 1)`.

- Operadors de comparació:

  a) == avalua a cert si la condició d'igualtat es compleix: `2 == 2 (Vertader)`.

  b) != avalua a cert si la condició d'igualtat no es compleix `2 != 2 (Fals)`.

  c) < avalua a cert si un número és més petit que l'altre `2 < 5 (Vertader)`.

  d) > avalua a cert si un número és més gran que l'altre `6 > 4 (Vertader)`.

  e) < =avalua a cert si un número és més petit o igual que un altre `2 <= 5 (Vertader)`.

  f) >=avalua a cert si un número és més gran o igual que un altre `6 >= 4 (Vertader)`.

- Operadors lògics:

  a) `&&` avalua a cert si els dos operadors són certs.

  b) `||` avalua a cert si algun dels operadors és cert.

  c) `And` avalua a cert si els operadors són certs.

  d) `Or` avalua a cert si algun dels operadors és cert.

  e) `Xor` avalua a cert si o un operador és cert o ho és l'altre.

  f) `!` inverteix el valor de cert de l'operador.

L'exemple següent ens mostrarà els operadors matemàtics més comuns:

```
<?php
 $a = 5;
 $b = 10;
 $c = ($a + $b); //$c val 15
 $d = ($b - $a); //$d val 5
 $e = ($a * $b); //$e val 50
 $f = ($b / $a); //$f val 2
 $g = ($b % $a); //$g val 0
?>
```

5.2.5. Estructures de control

Les estructures de control de PHP ens permeten controlar el flux de l'operació del nostre programa, controlant en tot moment quines porcions de codis s'executen en funció de determinades condicions.

Condicionals

Els condicionals són estructures que permeten dur a terme determinades operacions només en cas que es compleixi una condició. També se solen anomenar *bifurcacions*, ja que permeten dividir el fluix d'execució del programa d'acord amb el valor de veritat d'una sentència o condició.

Disposem en PHP de dos de condicionals principals, el condicional `if/else` i el condicional `switch`.

El condicional `if` ens permet escollir entre dos blocs de codi en funció del compliment o no d'una condició.

```
<?php
 $a = 0;
 $b = 1;
 if($a == $b)
 {
     echo "Resulta que 0 és igual a 1";
 }
 else
 {
     echo "Tot continua igual. 0 no és igual a 1";
 }
?>
```

Si seguim el flux d'execució d'aquest programa, veurem que inicialment creem dues variables `a` i `b`, a les quals assignem dos valors numèrics diferents. Tot seguit arribem a la sentència condicional `if`. Aquesta verifica la veracitat o compliment de la condició especificada. En aquest cas tenim un operador == d'igualtat, que ens torna que la comparació és falsa; per tant, la sentència `if` no executa el

primer bloc de codi, el que hauria executat en cas que es complís la condició, sinó que executa el segon, el precedit per `else`.

Podem definir, doncs, l'estructura d'`if/else` com a:

```
if(condició)
 {
    codi que s'executarà si la condició és certa
 }
else
 {
    codi que s'executarà si la condició és falsa
 }
```

Podem comprovar més d'una condició encadenant diversos `if/else`:

```
if(condició1)
 if(condició2)
    {
      codi que s'executarà si la condició2 és certa
      i la condició 1 és certa
    }
 else
    {
      codi que s'executarà si la condició2 és falsa
      i la condició 1 és certa
    }
else
 {
   codi que s'executarà si la condició1 és falsa
 }
```

Un cas estès de l'encadenament d'`if/else` és el corresponent a aquells supòsits en què hem d'executar codi diferent en funció del valor d'una variable. Malgrat que podem fer un encadenament d'`if/else` comprovant el valor d'aquesta variable, si hem de comprovar diversos valors el codi resulta molest. És per això que PHP proporciona una construcció condicional més adequada per a això que es diu `switch`.

```
<?php
 $a=1;
 switch($a)
```

```
{
   case 1:
   case 2: echo "A és 1 o 2"; break;
   case 3: echo "A és 3"; break;
   case 4: echo "A és 4"; break;
   case 5: echo "A és 5"; break;
   case 6: echo "A és 6"; break;
   default: echo "A és un altre valor";
}
```

L'execució de `switch` és una mica complexa; de fet, és molt semblant a la de C. La sentència `switch` s'executa línia a línia. Inicialment no s'executa cap codi de cap línia. Quan es troba un `case` amb un valor que coincideix amb el de la variable del `switch` PHP comença a executar les sentències. Aquestes sentències es continuen executant fins al final de `switch` o fins que es trobi un `break`. Per això en l'exemple si la variable val 1 o si la variable val 2, s'executa el mateix bloc de codi.

Tenim, a més, un valor especial `default` que sempre coincideix amb el valor de la variable.

Bucles

L'altra estructura de control important és la dels bucles, que ens permeten executar repetidament un bloc de codi en funció d'una condició.

Tenim tres bucles principals en PHP: `for`, `while` i `foreach`.

- El bucle **while**

El bucle `while` és el més simple dels tres, però tot i així segurament és el bucle més utilitzat. El bucle s'executa mentre la condició que li hem passat sigui certa:

```
<?php
 $a = 1;
 while($a < 4)
 {
    echo "a=$a<br>";
    $a++;
 }
?>
```

En aquest cas, el bucle s''executarà quatre vegades. Cada vegada que s'executi, incrementarem el valor de `a` i imprimirem un missatge. Cada vegada que es torna a executar el codi, `while` comprova la condició i, en cas de complir-se, torna a executar el codi. La quarta vegada que s'executi, com que `a` valdrà quatre, no es complirà la condició especificada i el bucle no es tornarà a executar.

- El bucle `for`

Per a bucles del tipus anterior, on la condició de continuació és sobre una variable que augmenta o disminueix amb cada iteracció, tenim un tipus de bucle més apropiat: `for`.

El codi anterior, usant `for` quedaria així:

```
<?php
 for($a=1;$a < 4; $a++)
 {
    echo "a=$a<br>";
 }
?>
```

Com podem veure, en el cas del bucle `for` en la mateixa sentència declarem la variable sobre la qual iterarem, la condició d'acabament i la d'increment o continuació.

- **foreach**

Per a aquells casos en què volem que el nostre bucle faci un recorregut sobre els elements d'un vector disposem d'una sentència que ens ho simplifica: `foreach`.

```
<?php
$a = array (1, 2, 3, 17);
foreach ($a as $v
{
 print "Valor: $v.\n";
}
?>
```

Com podem veure, en la seva forma més simple, `foreach` assigna a una variable v un a un tots els valors d'un vector a.

5.2.6. Funcions

Un altre punt clau de PHP són les funcions. Les funcions en PHP poden rebre o no paràmetres i sempre poden tornar un valor. Les funcions es poden fer servir per a donar més modularitat al codi, cosa que evita la repetició de codi, permet l'aprofitament de codi entre projectes, etc.

Un esquema de funció és el següent:

```
<?php
 function exemp ($arg_1, $arg_2, ..., $arg_n)
 {
 // Codi de la funció
 return $retorn;
 }
?>
```

Podem cridar les funcions des del codi principal o des d'altres funcions:

```
<?php
      function suma ($a1, $a2)
      {
         $retorn=$a1+$a2;
         return $retorn;
      }

      function sumatori ($b1, $b2, $b3)
      {
         for($i=$b1;$i<$b2;$i++)
         {
           $res=suma($res,$b3);
         }
         return $res;
      }
      echo sumatori(1,3,2);
?>
```

El resultat d'executar aquest programa serà que imprimirà un número sis.

Les funcions en PHP reben habitualment els paràmetres per valor, és a dir, la variable que es passa com a paràmetre en el codi que crida no sofreix modificacions si el paràmetre de la funció és modificat. Podem passar, no obstant això, paràmetres per referència (de manera similar als punters d'altres llenguatges de programació):

```
<?php
 function modifi ($&a1, $a2)
 {
      $a1=0;
      $a2=0;
 }
 $b1=1;
 $b2=1;
 modifi($b1,$b2);
 echo $b1." ".$b2;
?>
```

En aquest cas, el resultat del programa serà:

```
1 0
```

5.2.7. Ús de PHP per a aplicacions web

Per a usar PHP com a llenguatge de desenvolupament d'aplicacions web, la primera necessitat que tenim és saber com interactuarà PHP amb el nostre usuari web. Podem dividir aquesta interactuació en dues parts, mostrant informació a l'usuari i recollint-ne.

Mostrant informació

Tenim dos mecanismes perquè PHP mostri informació a l'usuari: d'una banda podem escriure pàgines HTML corrents, inserint només el codi PHP que requerim al mig del codi HTML. Per exemple:

```
<HTML>
 <HEAD>
     <TITLE>Títol del document</TITLE>
 </HEAD>
<BODY>
 <h1>Capçalera H1</h1>
 <?php $a=1; ?>
 <h1>Capçalera H1 segona</h1>
 <?php $b=1; ?>
 </BODY>
</HTML>
```

D'altra banda, podem usar PHP per a generar contingut dinàmic. Per a això hem d'utilitzar les instruccions de PHP de sortida de dades, la més important, `echo`.

```
<HTML>
 <HEAD>
     <TITLE>Títol del document</TITLE>
 </HEAD>
<BODY>
 <h1>Capçalera H1</h1>
 <?php echo "Contingut de la <B>pàgina</B>"; ?>
 <h1>Capçalera H1 segona</h1>
 </BODY>
</HTML>
```

Recollida d'informació de l'usuari

Per tal de recollir informació de l'usuari, podem utilitzar els formularis d'HTML, fent servir els nostres programes PHP com `ACTION`. Com que PHP va ser dissenyat per a crear aplicacions web, l'accés als valors introduïts per l'usuari en els camps del formulari és realment fàcil en PHP, perquè defineix un vector anomenat `REQUEST` al qual accedim amb el nom del camp com a índex i que conté el valor contingut en executar el programa PHP.

Si tenim aquest formulari:

```
<HTML>
   <HEAD>
      <TITLE>Títol del document</TITLE>
   </HEAD>
<BODY>
   <FORM ACTION="programa.php" METHOD=GET>
   Entre el seu nom: <INPUT TYPE=TEXT NAME="nom">
   <INPUT TYPE=submit>
   </FORM>
   </BODY>
</HTML>
```

I definim el programa següent PHP com a `programa.php` perquè respongui al formulari:

```
<HTML>
  <HEAD>
      <TITLE>Títol del document</TITLE>
  </HEAD>
  <BODY>
  <?php
  echo "Hola ".$REQUEST["nom"];
  ?>
  </BODY>
</HTML>
```

Aquest programa recollirà el nom introduït per l'usuari i ens el mostrarà per pantalla.

5.2.8. Funcions de cadena

PHP proveeix un conjunt de funcions molt interessants per al treball amb cadenes de text. Algunes de les més destacades són:

`strlen` Torna la longitud d'una cadena.

`explode` Divideix una cadena en funció d'un caràcter separador, i torna un vector amb cada tros de la cadena.

`implode` Actua al revés que `explode`, unint diverses cadenes d'un vector amb un caràcter d'unió.

`strcmp` Compara dues cadenes a nivell binari.

`strtolower` Converteix una cadena en minúscules.

`strtoupper` Converteix una cadena en majúscules.

`chop` Elimina l'últim caràcter d'una cadena, útil per a eliminar salts de línia o espais finals superflus.

`strpos` Busca dins d'una cadena una altra cadena especificada i torna la seva posició.

`str_replace` Reemplaça en una cadena una aparició d'una subcadena per una altra subcadena.

Podem veure el funcionament d'algunes d'aquestes funcions en l'exemple següent:

```
<?php
 $cadena1 = "hola";
 $cadena2 = "pera,poma,maduixa";

 $longitud = str_len($cadena1); //longitud=4

 $partes = explode(",",$cadena2);
 //genera l'array $parts amb $parts[0]="pera",
 //$parts[1]="poma"; y $parts[2]="maduixa";

 $chop = chop($cadena); // chop elimina la "a"

 $cadena3 = str_replace(",",";",$otracadena);
 //$cadena3 conté: pera-poma-maduixa
 //Canviem les , per -
?>
```

5.2.9. Accés a fitxers

PHP proporciona un repertori ampli de mètodes per a l'accés a fitxers. Mostrarem la més pràctica i simple, molt adequada si els fitxers als quals accedirem són petits.

El codi que comentarem és el següent:

```
<?php
 $fitxer = file("entrada.txt");
 $numlin = count($fitxer);
 for($i=0; $i < $numlin; $i++)
 {
     echo $fitxer[$i];
 }
?>
```

En aquest exemple llegim un arxiu que té per nom `entrada.txt` i el mostrem com a sortida. El primer pas que seguim és declarar la variable `fitxer`, la qual cosa ens genera un vector en què PHP col·locarà totes les línies de l'arxiu. Per a això utilitzarem la funció de llibreria `file`. El pas següent consisteix a esbrinar quants elements conté `fitxer`. Per a això utilitzem la funció `count`, que ens torna la mida d'un vector, en aquest cas el vector que hem generat en llegir el fitxer. Finalment podem escriure un bucle que recorrerà el vector tractant cadascuna de les línies de l'arxiu.

PHP proporciona moltes funcions més de tractament de fitxers. Disposem, per exemple, de la funció `fopen`, que permet obrir fitxers o recursos sense llegir-los íntegrament en memòria i que és capaç d'obrir fitxers de la manera següent:

```
 <?php
 $recurs = fopen ("entrada.txt", "r");
 $recurs = fopen ("sortida.gif", "wb");
 $recurs = fopen ("http://www.uoc.edu/", "r");
 $recurs = fopen (
    "ftp://usuari:password@uoc.edu/sortida.txt", "w");
?>
```

Hi podem veure com obrim un fitxer per a lectura (`"r"`), per a escriptura binària (`"wb"`), una pàgina web per llegir-la com si es tractés d'un fitxer i un fitxer per mitjà d'FTP per a escriure-ho, respectivament.

5.2.10. Accés a bases de dades

PHP proporciona mètodes per accedir a un gran nombre de sistemes de bases de dades (mySQL, PostgreSQL, Oracle, ODBC, etc.).

Aquesta funcionalitat és imprescindible per al desenvolupament d'aplicacions web complexes.

Accés a mySQL des de PHP

mySQL és un dels sistemes de bases de dades més populars en el desenvolupament d'aplicacions web lleugeres pel seu alt rendiment per a treballar amb bases de dades senzilles. Gran quantitat d'aplicacions web de consulta, etc. estan desenvolupades amb el duet PHPmySQL. Per això l'API d'accés a mySQL de PHP està altament desenvolupat.

Veurem un exemple d'accés a la base de dades des de PHP per comprovar així la senzillesa amb què podem utilitzar bases de dades en les nostres aplicacions web:

```
<?php
 $connexio=mysql_connect($servidor,$usuari,$password);
 if(!$connexio).
 {
    exit();
 }
 if(!(mysql_select_db($basedades,$connexio)))
 {
    exit();
 }
 $consulta=mysql_query(
    "select nom, telefon from agenda order by nom",
    $conexió);
 while($fila = mysql_fetch_array($consulta))
 {
    $nom = $fila["nombre"];
    $telèfon = $fila["telefon"];
    echo "$nom: $telefon\n<br>";
 }
 mysql_free_result($consulta);
 mysql_close($connexio);
?>
```

Podem veure que el primer pas per a accedir a la base de dades és obrir-hi una connexió. Per a això necessitarem l'adreça de l'ordinador que contingui la base de dades, l'usuari amb què connectarem i

la paraula d'accés a aquesta base de dades. Una vegada connectats al servidor de mySQL, hem de seleccionar quina base de dades de les múltiples que pot tenir el servidor volem utilitzar per a treballar. Arran d'aquesta seqüència de connexió, tindrem en la variable connexió les dades de la connexió a mySQL. Hem de passar aquesta variable a totes les funcions de PHP que accedeixin a bases de dades. Això ens permet tenir diverses connexions a diferents bases de dades obertes alhora i treballar-hi simultàniament.

El pas següent serà executar una sentència de consulta de bases de dades en el llenguatge de la nostra, en aquest cas SQL. Per a això usarem la funció de PHP `mysql_query`, que ens tornarà el resultat de la consulta i que desarem en la variable `consulta`. La consulta concreta enumera el contingut d'una taula de la base de dades anomenada *agenda*, que conté dues columnes: `nom` i `telefon`.

Després podem executar un bucle que recorrerà tots els registres perquè ens torni la consulta a la base de dades, accedint-hi un a un i podent mostrar així els resultats.

Quan hàgim acabat l'accés a la base de dades, hem d'alliberar la memòria i els recursos consumits amb la consulta. Per a això utilitzarem la funció `mysql_free_result` i després podrem tancar la connexió a mySQL.

Accés a PostgreSQL des de PHP

De manera similar a com accedim a mySQL, podem accedir a bases de dades que tinguem en servidors PostgreSQL. Per a això utilitzarem, igual que en la secció anterior, un codi que enumerarà el contingut de la nostra taula `agenda`.

```
<?php
  //conectem a la base de dades
  //$connexio = pg_connect("dbname=".$basedades);
  // conectem a la base de dades en servidor port "5432"
  //$connexio = pg_connect(
// "host=$servidor port=5432 dbname=$basedades");
```

```
//conectem a la base de dades en servidor port "5432"
//amb usuari i password
$connexio = pg_connect("host=$servidor port=5432 ".
   "dbname=$basedades user=$usuari password=$password")
   or die "No connecta";

$resultat = pg_query($connexio,
   "select nom, telefon from agenda order by nom");

while($fila = pg_fetch_array($result))
{
   $nom = $fila["nom"];
   $telefon = $fila["telefon"];
   echo "$nom: $telefon\n<br>";
}
pg_close($dbconn);
?>
```

Com podem observar si comparem aquest exemple amb l'anterior fet en mySQL, hi ha una gran similitud entre tots dos codis. No obstant això, malgrat les similituds, l'API de PostgreSQL per a PHP presenta algunes diferències respecte al de mySQL, ja que d'una banda ofereix més flexibilitat a l'hora de connectar i, de l'altra, proporciona el suport per treballar amb objectes grans, etc., que mostra la potència més gran de PostgreSQL enfront de mySQL. Un punt controvertit en l'API de PostgreSQL és que ens aïlla totalment del sistema de transaccions de PostgreSQL, cosa que malgrat que en la majoria de situacions és més que correcta, hi ha casos en què seria desitjable tenir-hi un control més gran.

5.2.11. Per a continuar aprofundint

Un dels punts forts i una de les claus de l'èxit de PHP com a llenguatge de programació d'aplicacions web resideix en la gran quantitat de llibreries, mòduls, etc., que se li han desenvolupat a propòsit. PHP posa a la nostra disposició una quantitat ingent d'API, funcions, mòduls, classes (recordeu que, progressivament, PHP s'està convertint en un llenguatge de programació orientat a objectes), que ens permeten operar amb la complexitat creixent de les aplicacions web. Aquesta diversitat de suport inclou, entre altres coses:

- Control de sessions.
- Control d'identitat d'usuaris.