使用者自訂函數

雖然程式語言軟體，都有針對常見工作項目提供一些函數，讓程式設計者直接使用以節省這一部份的設計工作，但是針對個別工作項目中，也會有一些個別常見的作業項目，但系統卻没有提供相對應的函數，對這些「遺珠之憾」，系統通常會另外提供「副程式」和「自訂函數」二項功能，來免除反覆編寫程式碼的情形。

※ 兩者之差異在於【副程式：直接處理而不需傳回處理值】、【自訂函數：會傳回處理過後所產生的結果值供程式進行後續的工作】。而PHP語言比較特殊的是，它只提供「自訂函數」的功能，而没有副程式，下面就針對「自訂函數」來說明。

**自訂函數基本運用通則，舉例如下：**

<?php

Function test($x,$y) //定義一個test函數，要執行本函數時，需提供$x,$y  
 { 兩筆資料

//test函數的程式

Return $z; //函數執行完後，回傳$z的資料

}

//主程式區

$acp=test($a,$b); //呼叫執行test函數，同時提供 $a→$x，$b→$y，

//test函數執行完後所回傳的資料 $z→$acp

//主程式區

?>

上面所列，只是自訂函數系列的其中一種型態，實際上，針對各種不同的工作型態，系統也提供各項相對應的自訂函數指令型態，分述如下：

**基本型態**

自訂函數指令格式基本型態分二種如下：

A.直接執行函數區塊指令，不需回傳資料：

function 函數名稱($參數1[,參數2,…,參數n])

{

程式執行區;

}

說明：1.參數指的就是變數，是提供給自訂函數做各項目相關運算的資料

2.函數名稱不可用中文，不可以用關鍵字（其他指令用字），不可以含有”.”符號，第一個字元不可用數字。

上例格式較類似副程式功能，因為它並没有傳回任何新變數值。

B. 執行函數區塊指令後，要傳回某些新變數的運算結果值：

function 函數名稱($參數1[,參數2,…,參數n])

{

程式執行區;

return $變數;

}

說明：return後面的變數是指要傳回主程式區的資料。

**呼叫自訂函數指令**

在PHP程式中，如果有包含「自訂函數」時，程式並不會主動執行「自訂函數」區塊，而是碰到「呼叫自訂函數指令」時，才會去執行該對應的「自訂函數」區塊。

「呼叫函數指令」格式如下：

A. 不需回傳資料之「呼叫函數指令」格式：

自訂函數名稱(參數值1,參數值2,….,參數值n);

B. 需要回傳資料之「呼叫函數指令」格式：

$全域變數=自訂函數名稱(參數值1,參數值2,….,參數值n);

說明：全域變數 是指在主程式區塊適用的變數。

區域變數 是指在「自訂函數」中所宣告的變數，只有在「自訂函數」區塊中才適用。

下面是應用第一種格式設計的自訂函數範例，在函數中直接印出運算結果，而不用傳回主程式區再進行後續運算。

ch13-01a.php

<?php

function swap($a,$b) //一般函數(没有提供資料，以程式中預設值代入運算

{ //兩個資料交換

$temp=$a;

$a=$b;

$b=$temp;

echo “\$a=”.$a.<br/>”;

echo “\$b=”.$b.”<br/>”;

}

function swap1($a=20,$b=60) //直接設定參數值代入運算

{ //兩個資料交換

$temp=$a;

$a=$b;

$b=$temp;

echo “\$a=”.$a.”<br/>”;

echo “\$b=”.$b.”<br/>”;

}

Swap(80,60);

Swap1(80,60);

Swap1();

?>

例：設計一個函數可以計算平均值，此函數會先以 is\_numeric( )函數判斷所輸入的資料是否為數值，如果參數是數值，會傳回平均值；如果參數非數值，會傳回：”所輸入的資料非數值!”的字串。

ch13-01b.php

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>定義函數</title>

</head>

<body>

<h2>定義函數</h2>

<?php

function average($a,$b)

{

//先判斷輸入的資料是否是數值

if(is\_numeric($a) && is\_numeric($b)){

$avg=($a+$b)/2;

//傳回平均值

return $avg;

}else{

return "<font color='red'>所輸入的資料非數值 !</font>";

}

}

//呼叫自訂函數

$info=average(20,64);

//顯示傳回值

echo "參數 20 和 64 的平均值是 : ".$info;

//再呼叫自訂函數,但給定的參數不同

$info=average(aa,23);

echo "<p>參數 aa 和 23 的平均值是 : ".$info;

?>

</body>

</html>

上面談到，區域變數只適用於函數區塊中；而全域變數則適用於函數區塊外；若希望在函數區塊中也能使用到全域變數的資料，則必須在引用的變數前面加上「global」關鍵字，範例如下

例：設計一含有函數的php程式，主程式中宣告一個全域變數，在函數中有一個區域變數，並利用global關鍵字引用該全域變數。

ch13-02.php

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>區域變數與全域變數</title>

</head>

<body>

<h2>區域變數與全域變數</h2>

<?php

$a="Hello!"; //全域變數

//自訂沒有傳回值的函數

function sayhello()

{

$b="Holiday"; //區域變數

global $a; //使用全域變數 (如果要用全域變數就一定要有global才可以喔)

echo $b."<br/>";

echo "--- ".$a."!!!";

}

echo $a."<p>";

//呼叫自訂函數

sayhello();

?>

</body>

</html>

傳遞整個陣列

在函數中，所傳遞的參數也可以是一個陣列變數，當然，所傳回的值，也可以是一個陣列變數值。

例:設計一函數，接收十筆資料的陣列變數，在函數中針對此陣列做遞減排序，再傳回最大的五筆資料，並以foreach指令顯示回傳的資料值。

ch13-03.php

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>傳遞整個陣列</title>

</head>

<body>

<?php

//原始陣列資料

$info=array(12,34,56,76,78,55,66,90,40,88);

function sorting($arr)

{//陣列遞減排序

arsort($arr);

//由陣列最後刪除5元素

for($x=0;$x<5;$x++){

array\_pop($arr);

}

//傳回最大的5筆資料

return $arr;

}

echo "原始陣列資料:<br/>";

print\_r($info);

echo "<p>";

//呼叫函數

$result=sorting($info);

//顯示回傳的陣列資料

echo "顯示回傳的陣列資料 : <br/>";

foreach($result as $value)

{

echo $value." -> ";

}

?>

</body>

</html>

單元作業1：以函數型態，簡化樂透電腦選號程式(lertofunc.htm，lertofunc.php)

說明：在下面的lerto1.php中，我們可看到紅色字的部份是相同的操作步驟，我們可將這設成函數，再以函數呼叫的方式來簡化程式篇幅。

lerto1.php

<html>

<head>

<title>

財神服務中心

</title>

</head>

<body>

<table width="100%"><tr align="center"><td>

<form name="lerto" method="post" action="lerto1.php">

<table align="center">

<tr align="center"><td><img src="th.jpg"><h1>財神電腦選號系統</h1></td></tr>

<tr align="center"><td><input type="submit" name="sen" value="威力彩"></td></tr>

<tr align="center"><td><input type="submit" name="sen" value="大樂透"></td></tr>

<tr align="center"><td><input type="submit" name="sen" value="今彩539"></td></tr>

</table>

</form>

</td></tr>

<tr align="center"><td><br><br>

<?php

if(isset($\_POST["sen"]))

{

//設定彩券資料陣列

for($a=1;$a<=49;$a++)

{

$lertonum1[$a]=$a; //第一區號碼

}

//接收前台傳送資料

$sen=$\_POST["sen"];

//

//判斷樂彩種類及產生對應號碼

//

if($sen=="威力彩")

{

//

//第一區38選6

//

for($a=38;$a>=33;$a--) //應用廻圈產生6個號碼

{

$k=rand(1,$a); //從1~$a間隨機產生一個數字

$ans[]=$lertonum1[$k]; //將第$k位置的號碼資料,放到選取資料陣列

$lertonum1[$k]=$lertonum1[$a]; //$k位置的號碼己被取走，可將最後一個位置的號碼移至此位置

}

sort($ans);

//

//第二區8選1

//

$ans2=rand(1,8); //從1~8間隨機產生一個數字

$p=6; //列印資料筆數用

}

elseif($sen=="大樂透")

{

//

//第一區49選6

//

for($a=49;$a>=44;$a--) //應用廻圈產生6個號碼

{

$k=rand(1,$a); //從1~$a間隨機產生一個數字

$ans[]=$lertonum1[$k]; //將第$k位置的號碼資料,放到選取資料陣列

$lertonum1[$k]=$lertonum1[$a]; //$k位置的號碼己被取走，可將最後一個位置的號碼移至此位置

}

sort($ans);

$p=6; //列印資料筆數用

}

else

{

//

//第一區39選5

//

for($a=39;$a>=35;$a--) //應用廻圈產生5個號碼

{

$k=rand(1,$a); //從1~$a間隨機產生一個數字

$ans[]=$lertonum1[$k]; //將第$k位置的號碼資料,放到選取資料陣列

$lertonum1[$k]=$lertonum1[$a]; //$k位置的號碼己被取走，可將最後一個位置的號碼移至此位置

}

sort($ans);

$p=5; //列印資料筆數用

}

//

//列印報表

//

echo "<h1>".$sen."</h1>";

echo "電腦為你產生的號碼如下:<br>第一區:";

for($a=0;$a<$p;$a++)

{

echo $ans[$a].",";

}

echo "<br>";

if($sen=="威力彩")

{

echo "第二區號碼:".$ans2."<br>";

}

}

else

{

echo "祝您好運亨通!請點樂彩項目";

}

?>

</body>

</html>

參數的傳遞

在函數中所使用到的參數資料，他的傳遞方式有「傳值」與「傳址」兩種。

「傳值方式」：是指主程式中要傳遞的變數(值)，與函數中所用來處理的變數，在記憶體中分別佔用不同的位置，因此，不管函數中的變數資料如何變化，都不會影響到主程式中變數的資料內容。

「傳址呼叫」：將主程式中變數或值在記憶體中所使用的位址傳送給函數，因此所傳送的資料值雖然相同，但在函數中，變數的資料值如果被改變，意謂，主程式中的變數值也相對被改變了。

Function test($a,$b) 以左邊程式為例，$a,$b及$x,$y 在記憶體中所使用的

{　　　　　　　　　　　位置如下圖示：

}

Test($x,$y);

　　　　　　　　　　　記憶體(傳值)　　　 記憶體(傳址)

$a

$b $a/$x

$x $b/$y

$y

各用各的位置 對應的資料使用

相同的位置

下面分別以兩個例子，來驗證兩種資料傳遞方式的結果。

傳值呼叫的指令格式如下：

函數名稱($變數1,$變數2,….,$變數n);

或

函數名稱(資料1, 資料2,…., 資料n);

例：設計一個會互換資料的函數，函數中以echo顯示互換之後的兩個變數值，另外在瀏覽器中顯示原始的兩個變數值，及執行傳值呼叫後的變數值。

ch13-04.php

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>傳值呼叫</title>

</head>

<body>

<h2>傳值呼叫</h2>

<?php

$x=33;

$y=66;

echo "原始\$x 和\$y 變數: <br/>";

echo "\$x=".$x."<br/>";

echo "\$y=".$y."<br/>";

function swap($a,$b)

{//兩個資料交換

$temp=$a;

$a=$b;

$b=$temp;

echo "\$a=".$a."<br/>";

echo "\$b=".$b."<br/>";

}

//以Call By Value 呼叫swap函數

echo "<p>以Call By Value 呼叫 swap函數<br/>";

swap($x,$y);

echo "<p>呼叫函數後的 \$x 和 \$y 變數: <br/>";

echo "\$x=".$x."<br/>";

echo "\$y=".$y."<br/>";

?>

</body>

</html>

傳址呼叫的指令格式如下：

function函數名稱(&$變數1,&$變數2,….,&$變數n);

說明：在參數前面加上”&”符號，就表示要以傳址方式傳送資料 這也就表示要用的資料和主程式是一樣的喔。

例：同上例，但以「傳址呼叫」方式設定函數，比較兩種方式的結果差異。

ch13-05.php

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>傳址呼叫</title>

</head>

<body>

<h2>傳址呼叫</h2>

<?php

$x=33;

$y=66;

echo "原始\$x 和\$y 變數: <br/>";

echo "\$x=".$x."<br/>";

echo "\$y=".$y."<br/>";

function swap(&$a,&$b)

{//兩個資料交換

$temp=$a;

$a=$b;

$b=$temp;

echo "\$a=".$a."<br/>";

echo "\$b=".$b."<br/>";

}

//以Call By Reference 呼叫swap函數

echo "<p>以Call By Reference 呼叫 swap函數<br/>";

swap($x,$y);

echo "<p>呼叫函數後的 \$x 和 \$y 變數: <br/>";

echo "\$x=".$x."<br/>";

echo "\$y=".$y."<br/>";

?>

</body>

</html>

特殊函數

遞迴函數

遞迴函數是指在一個函數中可以再呼叫函數本身，如此可以讓函數反覆運作。在某些問題中，如果可以用遞迴方式呼叫函數，可以精簡函數的篇幅，但要注意的是，使用遞迴函數，一定要有一個終結遞迴的點，否則程式的執行將無法結束。

例：設計一個遞迴函數，輸入參數值N，可以計算出N階乘的值。

ch13-06.php

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>遞迴函數</title>

</head>

<body>

<h2>遞迴函數</h2>

<?php

function factory($n)

{//判定是否為整數且大於零

if(is\_integer($n) && $n>0){

if($n>=2){

$temp=$n ;

$n=$n-1;

//遞迴呼叫本身

$total=$temp \* factory($n);

return $total;

}else{

return 1;

}

}else{

echo "請輸入大於零的整數";

}

}

//呼叫factory 函數

echo "5! = ".factory(5)."<br/>";

echo "6! = ".factory(6)."<br/>";

echo "7! = ".factory(7)."<br/>";

?>

</body>

</html>

巢狀函數

巢狀函數是PHP所提供比較特殊的地方，在其他程式語言中，通常是不提供(允許)巢狀函數的架構。

巢狀函數指令格式如下：

function 父函數名稱($參數)

{

Function子函數名稱($參數)

{

程式執行區(子函數);

}

程式執行區(父函數);

}

例：設計一個nest\_transform函數，內含bin( )、oct( )、hex( )三個子函數，這些函數分別可以將十進位數轉換成二、八、十六進位數。

ch13-07.php

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>巢狀函數</title>

</head>

<body>

<h2>巢狀函數</h2>

<?php

function nest\_transform($n)

{

function bin($n)

{//轉換成為二進位

return decbin($n);

}

function oct($n)

{//轉為八進位

return decoct($n);

}

function hex($n)

{//轉為十六進位

return dechex($n);

}

//以表格顯示轉換後的資料

echo "<table border=2 width=80%>";

//呼叫子函數

echo "<tr><th>二進位</th><td>".bin($n)."</td></tr>";

echo "<tr><th>八進位</th><td>".oct($n)."</td></tr>";

echo "<tr><th>十六進位</th><td>".hex($n)."</td></tr>";

}//結束nest\_transform函數

echo "十進位 100 轉換成其他進位表示 : <br/>";

//呼叫自訂函數

nest\_transform(100);

?>

</body>

</html>

不確定參數數量函數（有點搞不懂）

不定參數函數就是在設定函數時，先不指定參數資料，而是在函數中再以系統提供的特殊函數來取得所傳遞的參數個數或參數資料，在不定參數函數中會使用到的相關函數指令格式如下表：

|  |  |
| --- | --- |
| 函數指令 | 功能說明 |
| func\_num\_arg( ) | 取得傳入參數的個數。 |
| func\_get\_arg(n) | 取得第n個參數的值，n由0開始計數。 |
| func\_get\_args( ) | 傳回參數的整個陣列。 |

說明：上表指令只能在函數區塊內使用，若使用在區塊外，會出現警告訊息

例：設計一個不定不定參數函數，會依所傳入參數的多少，每一個以獨立的一行，列印相關資料到瀏覽器。

ch13-08.php

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>不定參數函數</title>

</head>

<body>

<h2>不定參數函數</h2>

<?php

//不定參數函數

function print\_me()

{

$nm=func\_num\_args();

echo "此次共傳 ".$nm." 個參數。<br/>";

for($x=0;$x<$nm;$x++){

echo func\_get\_arg($x)."<br/>";

}

}

//呼叫函數，傳遞2參數

print\_me("Linda","Peter");

echo "<p>";

//呼叫函數，傳遞3參數

print\_me("Happy","Holiday","Elvin");

?>

</body>

</html>

不定名稱函數

這也是PHP提供的特殊功能，他是先將函數名稱存在一個變數之中，再用函數呼叫變數，這樣，系統就會依照變數所儲存的函數名稱，去執行該對應函數，這種方式，就稱為「不定名稱函數」

例：設計一個可以輸入數值和選取不同進位方式的網頁，後端處理程式會顯示輸入的數值和選取的進位模式，並利用不定名稱函數，顯示對應的轉換值。

ch13-09.htm

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>選許所要模式</title>

</head>

<body>

<h2>請輸入數值 : </h2>

<form name="form1" method="post" action="13-09.php">

<p>

<input type="text" name="number">

</p>

<p>

<label>

<input type="radio" name="transform" value="bin">

二進位</label>

<label>

<input type="radio" name="transform" value="oct">

八進位</label>

<label>

<input type="radio" name="transform" value="hex">

十六進位</label>

</p>

<p>

<input type="submit" name="Submit" value="送出">

<input name="reset" type="reset" id="reset" value="重設">

<br>

</p>

</form>

<p>&nbsp; </p>

</body>

</html>

ch13-09.php

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>不定名稱函數</title>

</head>

<body>

<h2>不定名稱函數</h2>

<?php

//轉換成為二進位

function bin($n)

{

echo "二進位表示 : ".decbin($n);

}

//轉為八進位

function oct($n)

{

echo "八進位表示 : ".decoct($n);

}

//轉為十六進位

function hex($n)

{

echo "十六進位表示 : ".dechex($n);

}

//取得表單傳遞過來的資料

$nm=$\_POST[number];

$choose=$\_POST[transform];

echo "輸入的數值是 : ".$nm."<br/>";

echo "選取的模式是 : ".$choose."<p>";

//呼叫不定名稱函數

//依不同的變數值去呼叫不同的函數

$choose($nm);

?>

</body>

</html>

外部函數檔案

外部檔案是指要使用的函數是設定在別的PHP程式中，換一個角度來說，就是在程式中所設定的函數也希望能提供給其他的PHP程式使用。

若希望程式中所定義的函數能夠提供給其他的PHP程式使用，則在儲存檔案時，檔案的副檔名就必須設定為「\***.**inc」。

.inc基本上就是php的函數庫

外部檔案函數指令有兩個，說明如下：

include指令格式如下

include”路徑和檔案名稱”;

或

Include(”路徑和檔案名稱”);

說明：1. include所引用的外部檔案可以是「\*.inc、\*.txt或\*.php」

2. 如果所要加入的外部檔案不存在，系統會出現警告訊息，但不會中斷程式的執行

例：先設計一個含有average( )和sum( )兩函數的【funs.inc】，再設計一個PHP網頁，以include指令將funs.inc函式庫載入，利用函數所提供的功能去計算平均和總計。

funs.inc

<?php

function average($a,$b)

{

//先判斷輸入的資料是否是數值

if(is\_numeric($a) && is\_numeric($b)){

$avg=($a+$b)/2;

//傳回平均值

return $avg;

}else{

return "<font color='red'>所輸入的資料非數值 !</font>";

}

}

function sum($a,$b)

{

//先判斷輸入的資料是否是數值

if(is\_numeric($a) && is\_numeric($b)){

$total=$a+$b;

//傳回總計

return $total;

}else{

return "<font color='red'>所輸入的資料非數值 !</font>";

}

}

?>

ch13-10.php

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>include外部檔案</title>

</head>

<body>

<h2>include外部檔案</h2>

<?php

//載入外部檔案

include "funs.inc";

$x=566;

$y=128;

echo "輸入的資料是 : ".$x." , ".$y."<p>";

//呼叫函式庫中的函數

echo "平均值 => ".average($x,$y)."<p>";

echo "總計是 => ".sum($x,$y);

?>

</body>

</html>

require指令

require的作用和include相同，差異在於當外部檔案不存在時，會產生執行錯誤，並中斷程式的執行。

基本上程式要上線的時，所有資料都要存在，這算是一個最後確認的概念

例：先設計一個判斷日期資料是否符合規定格式的函數【ddfunc.inc】，再設計一個PHP網頁，以require指令將ddfunc.inc函式庫載入，利用函數判斷，如果是合格的日期格式，則依格式印出日期資料。

ddfunc.inc

<?php

function ddcheck($m,$d,$y)

{

//判斷所輸入的資料是否是合法的日期資料

if(checkdate($m,$d,$y)){

//將輸入的資料轉成UNIX時間戳記

$dd=mktime(0,0,0,$m,$d,$y);

//以date()函數顯示對應格式

echo date("M-d-Y",$dd);

}else{

echo "所輸入的資料不是合法的日期格式!";

}

}

?>

ch13-11.php

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>require外部檔案</title>

</head>

<body>

<h2>require 外部檔案</h2>

<?php

//載入外部檔案

require "ddfunc.inc";

$mm=8;

$dd=20;

$yy=2005;

echo "輸入的資料是 : <br/>";

echo $mm." , ".$dd." , ".$yy."<br/>";

//呼叫載入檔案中的函數

ddcheck($mm,$dd,$yy);

//改變$dd值

echo "<p>輸入的資料是 : <br/>";

$dd=34;

echo $mm." , ".$dd." , ".$yy."<br/>";

//呼叫載入檔案中的函數

ddcheck($mm,$dd,$yy);

?>

</body>

</html>

　善用自訂函數的功能，也可讓程式流程的解讀更加清楚流暢。我們可以將程式中各階段較為煩瑣的步驟，設成自訂函數，而在主程式中，每一主要動作都用　呼叫函數　的方式去完成。下面我們將guess2.php中主要程式的工作片段改成　函數呼叫方式　的結構如下：

藍色字為　主要程式的工作片段函數

紅色字為 主要程式的

Guess2fun.php

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=big5">

<title>猜數字遊戲(一)</title>

</head>

<body>

<?php

/////////////////////////////////////////////////////////////////////////////

function acep($coti,$ank) //定義接收猜測及比對結果資料函數

{

if($coti>=2) //如果猜測次超過1次，接收前面已猜測次數的資料

{

for($a=1;$a<$coti;$a++)

{

$alb="anb".$a;

$tta1[$a]=$\_POST[$alb];

$ala="ans".$a; //ans+n 為己答次數在表單中的name

$bn[$a]=$\_POST[$ala]; //$bn[n]為php已答次數答案

}

$bn[$coti]=$ank; //最近一次的答案

}

else

{ //第一次要將答案放到 $bn[1]

$bn[$coti]=$ank;

}

return $bn;

}

///////////////////////////////////////////////////////////////////////////

function decrypt($tu) //定義題目資料解密函數

{

/////////// 1.資料倒置

for($a=0;$a<4;$a++)

{

$tu[$a]=9-$tu[$a];

}

////////// 2.往右移一位

$k=$tu[3];

$tu[3]=$tu[2];

$tu[2]=$tu[1];

$tu[1]=$tu[0];

$tu[0]=$k;

return $tu;

}

////////////////////////////////////////////////////////////////////////////

function A($tu,$ta) //定義A比對工作函數

{

$at=0; //計算 xA

for($k=0;$k<4;$k++)

{

if($tu[$k]==$ta[$k]){$at++;} //只比較相同位置

}

return $at;

}

////////////////////////////////////////////////////////////////////////

function B($tu,$ta) //定義B比對工作函數

{

$tb=0; //計算 xB

for($j=0;$j<4;$j++)

{

for($j1=0;$j1<4;$j1++)

{

if($j<>$j1) //相同位置不比較

{

if($tu[$j]==$ta[$j1]){$tb++;}

}

}

}

return $tb;

}

//主程式

////////////// 登錄遊戲開始時間 //////////////////////////////

$fisc=$\_POST["fisc"];

$info=localtime($fisc,true);

echo "猜答開始時間:".($info["tm\_year"]+1900)."年".($info["tm\_mon"]+1)."月".$info["tm\_mday"]."日".($info["tm\_hour"]+7)."點".$info["tm\_min"]."分".$info["tm\_sec"]."秒<br>";

/////////////////////////////////////////////////////////////////

//接收資料

$tp=$\_POST["tp"]; //$tp 為題目數字

$coti=$\_POST["coti"]; //$coti為猜測次數

$ank=$\_POST["ans"]; //$ans為最新一次答案

$bn=acep($coti,$ank); //接收前面已猜測次數的資料

//比較本次猜答結果

$tu=str\_split($tp); //拆解題目數字

$tu=decrypt($tu); // 題目資料解密

$ta=str\_split($ank); //拆解猜答數字

//開始比較(數字對，位置也對

$at=A($tu,$ta); //計算 xA

$tb=B($tu,$ta); //計算 xB

//將比較結果放入$tta

$tta=$at."A".$tb."B";

$tta1[$coti]=$tta;

//列印結果表單

//猜測結果成功

if($at==4)

{

echo "<font color='#0000ff'><h1>猜數字遊戲<br>";

echo "題目數字：".$tp."</h1><br>";

/////////////////////////////////////// 猜答時間 ///////////////////////////////////

$clsc=time();

$info=localtime($fisc,true);

echo "猜答開始時間:".($info["tm\_year"]+1900)."年".($info["tm\_mon"]+1)."月".$info["tm\_mday"]."日".($info["tm\_hour"]+7)."點".$info["tm\_min"]."分".$info["tm\_sec"]."秒<br>";

$info=localtime($clsc,true);

echo "猜答結束時間:".($info["tm\_year"]+1900)."年".($info["tm\_mon"]+1)."月".$info["tm\_mday"]."日".($info["tm\_hour"]+7)."點".$info["tm\_min"]."分".$info["tm\_sec"]."秒<br>";

echo "猜答花費時間為:".(int)(($clsc-$fisc)/60)."分".(($clsc-$fisc)%60)."秒<br>";

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////

echo "<h2>恭喜您答對了! 您總共猜了".$coti."次! 您的答案依序為:</h2></font><br>";

for($i=1;$i<=$coti;$i++)

{

echo "第".$i." 次: ".$bn[$i]."　　為：　　".$tta1[$i]."<br>";

}

}

else

{

//再次顯示猜測表單

echo "<font color='#0000ff'><h1>猜數字遊戲</h1></font><br>";

echo "<form method='post' action='guess2.php'>";

//題目數字在 tan

echo "題目數字:<input type='password' name='tan' value=".$tp." size='4' >　　猜測結果<br>"; //測試完成要改成password

echo "<input type='hidden' name='tp' value=".$tp." size='4' ><br>"; //tp為題目數字

$coti++; //已答次數累加

for($b=1;$b<$coti;$b++)

{

$ala="ans".$b; //ans+n 為己答次數在表單中的name

$alb="anb".$b; //anb+n　為猜測判斷結果

echo "第".$b."次猜:<input type='text' name=".$ala." value=".$bn[$b]." size='4' readonly >";

echo "　　　　<input type='text' name=".$alb." value=".$tta1[$b]." size='4' readonly ><br>";

}

echo "<input type='hidden' name='coti' value=".$coti." size='4'><br>";

echo "第".$coti."次猜:<input type='text' name='ans' size='4' ><br>";

echo "<input type='submit' name='submit' value='送出'>";

//////////////////////////////////////////////////////////////////////////////////////////////

echo "<input type='hidden' name='fisc' value='".$fisc."' <br>"; //遊戲開始時間

//////////////////////////////////////////////////////////////////////////////////////////////

echo "</form>";

}

?>

</body>

</html>

單元作業２：用php設計2048遊戲，以按鈕替代↑←↓→按鍵，遊戲規則請自  
　　　　　　行查詢(2048-1.php、2048-2.php)

