# الفصل الرابع
# التكرارت والمصفوفات

لقد أخذنا في البحث السابق شيئاً من أساسيات البرمجة وهو الدوال الشرطية وصناعة القرارات والآن نحن نتجه إلى شيء يحب جهاز الكمبيوتر عمله وهو التكرارت والمصفوفات.

في الواقع قد يكون لديك يومياً شي تفعله بشكل مستمر مثل الإفطار في الصباح الباكر والنوم مساء، انك تستمر على هذا الروتين دائماً.... نحن نسمي هذا الشي في لغة البرمجه التكرار.

هناك شي آخر يسمي المصفوفات... في الواقع قد يحتوي درج مكتبك الخاصة بالكتب على عدة أدراج الدرج الاول منها يحتوي على الكتب الإسلامية والدرج الثاني منها يحتوي على الكتب الرياضية والدرج الثالث يحتوي على كتب الرياضيات... أو لنفرض أنك مدرس في إحدي المدارس ولديك جدول للحصص ففي الحصة الأولى لديك مثلاً تدريس مادة الرياضيات.... والحصة الثانية لديك تدريس مادة العلوم والثالثة لديك تدريس مادة الكيمياء.... إن حصصك مرتبة بشكل معين مع أنها كلها تسمي حصص إلا أن كل حصة تختلف عن الأخري في المادة ! وهي مرتبه بشكل تصاعدي (الحصة الاولى، الثانية، الثالثة.....).

نسمي هذه التقنية بالمصفوفات.... المصفوفات عبارة عن متغير اسمه ثابت ولها اكثر من قيمة وكل قيمة لها رقم معين ولكي تحصل على القيمة فانك تكتب المتغير ثم رقم القيمة التي فيه، لايشترط أن تكون هذه القيم متسلسلة فقد يكون هناك قيمتين ولكل قيمة رقم يختلف تماماً ويبعد كل البعد عن القيمه الثانية مثال رقم ١ و ٢٥٨ كلاهما مختلف تماماً ويبتعد كل البعد عن الاخر.

إن دمج ميزة التكرارات مع المصفوفات يساعدك على توفير عدد الأسطر للكود ويساعدك على صنع أشياء عجيبة في أقل عدد ممكن من الأسطر.

## التكرارات

التكرارات عبارة عن تكرار أمر معين بعدد معين من المرات ولقد اخذنا سابقاً الدوال الشرطية أو العبارات الشرطية بالأصح فوجدنا أن الكود الذي نكتبه في العبارات الشرطية لا تتنفذ إلا عندما يكون الشرط صحيحاً

أيضا التكرارت فهي تختبر الشرط فإذا كانت قيمته صحيحة فإنها تقوم بعمل الكود المطلوب ثم تقوم بإعادة اختبار القيمة فإذا كان صحيحاً فإنها تقوم بإعادة تنفيذ الكود وهكذا ، أما عندما لا يكون الشرط صحيحاً فإنها تتوقف عن تنفيذ الكود ويتم اكمال البرنامج بشكل عادي... هناك ثلاثة أنواع من التكرارات.

إن أول دالة نقوم بأخذها في البداية هي الدالة while

### التكرار while

لقد قمنا بأخذ التكرار while لأنه بسيط جداً وصيغه هذا التكرار هي:

```
While (condition شرط )
{
code
}
```

**مثال:**

```
<?
$d =10 ;
while ($d<15)
{
echo “$d <br>”;
$d++;
}
?>
```

سيقوم الـPHP أولاً بإعطاء المتغير $d القيمة ١٠ ثم يقوم بعد ببدء التكرار while فإذا كان الشرط صحيحاً (وهو أن المتغير أصغر من الرقم ١٥) فإنه يقوم بتنفيذ الكود الذي بين الأقواس وعمل هذا الكود أن يقوم بطباعة المتغير ثم يقوم بإضافة واحد على القيمة الموجودة في المتغير $d ثم بعد ذلك سيتم اختبار الشرط مرة ثانية فإذا كان صحيحاً فستتم نفس العملية حتي يكون الشرط غير صحيح فيتوقف عندها التكرار ويتم إكمال الكود الذي يقع بعد الأقواس.

إذا لم تقم بوضع حد للتكرار فلن يتوقف التكرار وقد يكون لانهائي....

**مثال:**

```
<?
$d =10 ;
while ($d<15)
{
echo “$d <br>”;
}
?>
```

ستتم طباعة الرقم ١٠ ولن يتوقف التكرار لأن الشرط صحيح دائماً وليس هناك ما يوقفه بينما في الكود السابق استطعنا إيقاف الكود بسبب أننا كنا نضيف واحد على القيمة الموجودة في المتغير وكلما تمت إعادة اختبار الكود كلما تغيرت القيمة حتى يصبح الشرط غير صحيح بسبب أن $d أكبر من ١٥.

## التكرار do - while

هذا التكرار يعمل بنفس طريقة التكرار الأول إلا أنه يوجد بعض الاختلافات البسيطة وصيغته كالتالي:

```
do
code
;(شرط)while (condition
```

**مثال:**

```
<?
$f=15 ;
do
{
echo “$f”;
$f ++
}
while () ;
```

سيقوم التكرار بتنفيذ السطر الموجود بين القوسين أولاً ثم يقوم بتنفيذ باختبار الشرط فإذا كان الشرط صحيحاً قام بإعادة العملية الموجودة بين القوسين وهي إضافة واحد على المتغير $f وهكذا حتي يكون الشرط غير صحيح فيتم التوقف.. لاحظ أننا في التكرار الأول قمنا باختبار الشرط قبل صناعة أي عمل بينما في التكرار الثاني قمنا بتنفيذ الكود أولاً ثم قمنا بإجراء الاختبار.

## التكرار FOR

يختلف هذا التكرار عن سابقيه لكن وظيفته هي نفس وظيفتهما وهي تكرار الأوامر عند حصول شيء معين

الصيغة:

```
For (counter عداد ; test value اختبار القيمة ; set counter أداء عمليه على العداد)
{
code شيفرة
}
```

**مثال:**

```
<?
For ($u = 18 ; $u>10 ; $u--)
{
echo $u;
}
?>
```

يتكون هذا التكرار من ثلاثة أقسام.... القسم الأول نضع فيه متغير يحتوي على قيمة حيث سيبدأ التكرار العمل من عند هذه القيمة والقسم الثاني نكتب فيه الشرط الذي سيقوم التكرار بفحصه (والذي هو كالمعتاد اختبار لقيمة المتغير في القسم الأول) والقسم الثالث نضع فيه العمل الذي سيجري على المتغير عند كل تكرار ثم نقوم بكتابة الكود الذي سيقوم بتنفيذ التكرار بين القوسين.

كأننا نقول للـphp بشكل عام أن يقوم في البداية بإعطاء المتغير $u القيمة ١٨ وقبل أن يقوم بتنفيذ الكود عليه أن يقوم بتحليل الشرط فإذا كان الشرط صحيحاً فإنه يقوم بإنقاص واحد من المتغير $u ويتم تنفيذ الكود حتى يصبح المتغير$u قيمته ٩ فيقوم الـPHP آنذاك بالخروج من التكرار والذهاب إلى الكود الذي يلي القوسين.

## المصفوفات

لقد قمنا بتعريف المصفوفات سابقاً بشكل بسيط وحان الوقت الآن لنعرفها ونعرف كيفية عملها.

المصفوفات عبارة عن متغير وهذا المتغير يحتوي على أكثر من قيمة أو عنصر (element) وكل عنصر له فهرسة (Index) تبدأ هذه الفهرسة من الصفر إذا لم تقم بتحديدها.

**مثال:**

```
<?
$A[ ] = “alfareees”;
$A[ ] = 13;
?>
```

في هذا المثال سيقوم الـPHP بإعطاء الفهرسة تلقائياً فسيقوم بوضع الرقم فتصبح المتغير فهرسته كالتالي:

```
$A[0] = “alfareees”;
$A[1] = 13;
```

إننا لم نقم بإدخال هذه الأرقام من تلقاء أنفسنا ولكن الـPHP قام بوضعها مع أنه يمكننا أن ندخلها بشكل عادي فمثلاً لو كتبنا:

```
<?
$A[0]= “alfareees”;
$A[1] = 13;
?>
```

سيقوم الـPHP بأخذ الفهرسة المعتمدة ولن يضع أي فهرسة أخرى يمكننا أيضاً أن نكتب أي فهرسة ولا نعتمد على الترتيب في الأرقام.

**مثال:**

```
<?
$A[10 ] = “alfareees”;
$A[ 25] = 13;
?>
```

هل لاحظت أيضاً أننا لم نقم بتعريف نوع متغيرات المصفوفة وقام الـPHP بتعريفها تلقائياً بدلاً منا فمرة استخدمنا قيمة حرفية ومرة استخدمنا رقماً ورغم ذلك فلم يقم الـPHP بعمل أي اعتراض إضافة إلى ذلك فإن الـPHP يقوم بتحديد عدد عناصر المصفوفة تلقائياً فهو يعرف مثلاً من المثال السابق أن عدد عناصر المصفوفة الكلي هو عنصرين.

يمنحنا الـPHP ميزة أخري وهي عدم التقيد بالأرقام في الفهرسة فمثلاً يمكننا استخدام حروف عادية.

**مثال:**

```
<?
$A[“a” ] = “alfareees”;
$A[“b” ] = 13;
?>
```

لاحظ أننا استخدمنا القيم الحرفية ولم يعترض الPHPJ بتاتاً ويمكننا طباعة أي عنصر من عناصر المصفوفة بكل بساطة.

مثال:

```
<?
$r [“aa”] = “ahmed ali”;
$r [1] = 13273;
$r [20] = 13273;
echo $r[aa];
echo $r[20];
];"aa"echo $r[
?>
```

لا فرق بين أن نكتب النص الحرفي (aa) بين علامتي تنصيص عند الطباعه وعند كتابته بدون علامات تنصيص... سيقوم الـPHP بمعرفة ذلك تلقائياً.

يمكننا تعريف المصفوفات أيضاً بطريقة أخرى

```
$variable = array (elements) ;
```

**مثال:**

```
<?
 “alfarsi”); ، “salem”، “ali”،$t =array (“ahmed”
echo $t [0];
?>
```

يقوم الـPHP بإعطاء كل عنصر من عناصر المصفوفة رقم فهرسة فتصبح كالتالي:

| Index الفهرسة | Element العنصر |
|---|---|
| ٠ | Ahmed |
| ١ | Ali |
| ٢ | Salem |
| ٣ | alfarsi |

إذن القيمة التي سيطبعها الـPHP في النهاية هي ahmed، لاحظ أن الـPHP قام بإعطاء رقم الفهرسة وقام بالبدء من الصفر ولكن يمكننا جعل الـPHP يبدأ الفهرسة من الرقم واحد كالتالي:

```
<?
 “alfarsi”);،”salem”، “ali”،$r = array (1=>”ahmed”
?>
```

عند تعريفك لرقم الفهرسة للقيمة الأولى سيقوم الـPHP بإعطاء أرقام فهرسة بشكل تسلسلي، عندئذ ستصبح الفهرسة كالتالي:

| Index الفهرسة | Element العنصر |
|---|---|
| ١ | ahmed |
| ٢ | Ali |
| ٣ | salem |
| ٤ | alfarsi |

هناك طريقة لتكون أيضاً الفهرسة هي عبارة عن حروف:

```
<?
 “bv”=> “alfarsi”);، “da”=>”salem”، “sf”=> “ali”،$r = array (“ss”=>”ahmed”
?>
```

عندئذ ستصبح الفهرسة كالتالي:

| Index الفهرسة | Element العنصر |
|---|---|
| Ss | Ahmed |
| Sf | Ali |

| | |
|---|---|
| Da | Salem |
| Bv | Alfarsi |

عندما نريد تغيير أي عنصر في المصفوفة فيمكننا عمل ذلك ببساطة.

**مثال:**

```
$r [ss]= “لمياء”;
```

لاحظ أننا قمنا بتغيير القيمة من (ahmed) إلى (لمياء)....طريقة بسيطة أليس كذلك:

## قراءه المصفوفات واستخراج القيم

تكلمنا سابقا عن التكرار For

يمكننا استخراج عناصر مصفوفة وطباعتها ببساطة وتوفير وقت عن طريق التكرارات

لنفرض أن لديك هذه المصفوفة:

```
<?
$people =array (“ahmed”، “ali”، “salem”، “alfarsi”);
?>
```

وأردت أن تطبع أسماء جميع الأشخاص المتواجدين فيها

أولاً نحن نعرف أن المصفوفة إذا لم نقم بتعريف رقم فهرسة لها فإن الـPHP يقوم ببداية فهرستها من الصفر وعلى ذلك فإن رقم العنصر الأول ٠ ورقم العنصر الرابع ٣... على ذلك يمكننا بكل بساطة كتابة الكود التالي الذي يقوم بطباعة المصفوفة كالتالي:

```
<?
$people =array (“ahmed”، “ali”، “salem”، “alfarsi”);
echo “$people[0]. <br>”;
```

```
echo “$people[1]. <br>”;
echo “$people[2]. <br>”;
echo “$people[3]. <br>”;
?>
```

لنفرض أن لديك ثلاثين أو ثلاثة آلاف اسم في مصفوفة ألن تبدو هذه الطريقة متعبة قليلا !!!

هناك طريقة أخري وهي عن طريق التكرارات.

لنفرض أننا أردنا كتابة تكرار يقوم بطباعة الأرقام من واحد إلى عشرة فإننا نستطيع كتابة التكرار بالشكل التالي:

```
<?
For ($I=1;$I<11;$I++)
{
Echo "$I <br>";
}
?>
```

والآن لنقل أننا نريد طباعة الأربعة عناصر في المصفوفة كل ما علينا هو إجراء عملية بسيطة على الكود لكي يتم ذلك:

```
<?
 "alfarsi"); ، "salem"، "ali"،$people =array ("ahmed"


For ($I=0;$I<4;$I++)
{
Echo "$people[$I] <br>";
}
?>
```

لاحظ أننا بدأنا العداد بالقيمة صفر ثم اشترطنا أن يكون أقل من ٤ لأن آخر عنصر في المصفوفة رقم فهرسته ٣ ثم قمنا بجعله يزداد بقيمة ١ لأننا نريد طباعة جميع

عناصر المصفوفة وقمنا بوضع رقم العداد في خانة الفهرسة وعلى ذلك سيتم في كل تكرار طباعة عنصر المصفوفة الذي فهرسته تساوي رقم العداد.

لقد تكلمنا سابقاً في فصل النماذج عن إخراج القيم من قائمة على شكل مصفوفة.

**مثال:**

```
<form action = “array.php” method = post>
ما هو مشروبك المفضل ؟
<br>
<select name = “a[]” multiple>
</option>شاي<option>
</option>قهوة<option>
</option>كابتشينو<option>
</option>مته<option>
</option>برتقال<option>
</select>
<br>
” >لذيذ<input type=submit value = “
</form>
```

في ملف الـ array.php اكتب:

```
<html>
لقد قمت باختيار التالي:
<?
For ($I=0;$I<4;$I++)
{
Echo "$a[$I] <br>";
}
?>
</html>
```

لقد عرضنا في القائمة خمسة عناصر... لاحظ أننا وضعنا في اسم المتغير للقائمة قوسين [ ] لكي يتعرف الـhtml على أنه سيتم تخزين البيانات تلقائياً بعد ذلك قام الـPHP بفهرسة العناصر التي تم إرسالها من قبل العميل سواء كانت ثلاثة أو أربعة ولكنها بالطبع لن تزيد على خمسة.... على ذلك سيكون آخر رقم تنتهي به المصفوفة هو ٤.

أتوقع أنك الآن بدأت تحب المصفوفات.... يمكننا صناعة القائمة عن طريق المصفوفة أيضاً....

**مثال:**

```
<form action = “list.php” method = post>
من هو صديقك المفضل ؟
<br>
<select name = “s” >
<?
”);شاي”،”قهوة”،”كابتشينو”،”مته”،”برتقال$shrab =array(“
For ($k=0;$k<4;$k++)
{
echo “<option>”.$shrab[$k].”</option>”;
}
?>
</select>
</form>
```

عند اختيار المستخدم للقيمة سيتم وضعها في المتغير $s يمكنك مراجعة درس النماذج لكي تفعل ذلك، هذا المثال يقوم بصناعة مصفوفة للمشروبات ثم يقوم بإخراجها في قائمة مما يوفر علينا الوقت في كتابة الكود فلو كان لديك مثلا حوالى مئة دولة فيمكنك مثلا وضعها في مصفوفة وبعد ذلك بناء القائمة التي سوف تقوم ببناء القائمة التي ستحتوي على هذه الدول عن طريق المصفوفات والتكرارات.
قم بحفظ التغييرات في ملف إمتداده php وقم بكتابة الملف list.php اعتمادا على معلوماتك السابقة في درس النماذج.

## دوال المصفوفات

### الدالة key

لنفرض أن لدينا مصفوفة مكونة من عنصرين:

مثال:

```
$s= array ("على"، "محمد");")
```

الآن لنضف إليها هذه السطور

```
<?
$s= array ("على"، "محمد");")
$t=key ($s);
echo $t;
?>
```

يقوم الأمر key بإيجاد رقم الفهرسه (index) العنصر النشط حالياً.... وهو الرقم صفر حيث أننا لم نضع فهرسة وهذه هي الفهرسة التي وضعها الـPHP تلقائياً عندما لم نضع فهرسة... قد تحيرك كلمة النشط لكن ستعرف أننا نستطيع التجول بين عناصر المصفوفة لاحقاً.

قد يكون رقم الفهرسة حروف أو كلمات

**مثال:**

```
<?
$s= array ("ع"=>"علي"، "م"=>"محمد");")
$t=key ($s);
echo $t;
?>
```

### الدالة current()

تقوم الدالة current بإيجاد القيمة لعنصر المصفوفة الحالي (index value).

**مثال:**

```
<?
");محمد"=>"م"،"علي"=>"ع$s= array ("
$p=current ($s);
echo $p;
?>
```

في المثال السابق قمنا بإيجاد القيمة الحالية للعنصر النشط.... لاحظ أننا أوجدنا بالأمر key رقم الفهرسة بينما أوجدنا بالأمر current القيمة للعنصر المفهرس.

*كيف يمكننا تنشيط العناصر الأخرى للمصفوفة ؟!*

يمكننا ذلك عن طريق الدالتين next() و prev اللتان تقومان بالتجول بين عناصر المصفوفة..... لنفرض أن لدينا مصفوفة تتكون من ثلاثة عناصر

**مثال:**

```
<?
");أحمد"=>"ا"،"محمد"=>"م"،"علي"=>"ع$s= array ("
echo key($s)."<br>";
echo current($s)."<br>";
?>
```

لقد قمنا في هذا المثال بطباعة قيمة رقم الفهرسة للعنصر الحالي وقيمته (اقصد برقم الفهرسة الحرف(ع) وأقصد بالقيمة (على))..... لنقم الآن بالتجول بين عناصر المصفوفة ولنر نتيجة الطباعة.

**مثال:**

```
<?
");أحمد"=>"ا"،"محمد"=>"م"،"علي"=>"ع$s= array ("
next($s);
echo key($s)."<br>";
echo current($s)."<br>";
```

```
?>

<?
");أحمد"=>"ا"،"محمد"=>"م"،"علي"=>"ع$s= array ("
next($s);
next($s);
echo key($s)."<br>";
echo current($s)."<br>";
?>
```

لاحظ أننا كتبنا الدالة next() قبل أن نقوم بالانتقال لكي يتم تنشيط العنصر الثاني في أول مثال ولتنشيط العنصر الثالث في ثالث مثال (ولاحظ أننا كتبنا next() مرتين).

يمكننا الرجوع لتنشيط العنصر السابق بوضع الدالة prev() فمثلاً يمكننا تعديل المثال التالي:

```
<?
");أحمد"=>"ا"،"محمد"=>"م"،"علي"=>"ع$s= array ("
next($s);
next($s);
prev($s);
echo key($s)."<br>";
echo current($s)."<br>";
?>
```

فسيقوم الـPHP في هذه الحالة بطباعة العنصر الثاني وليس الثالث لأنه تم التراجع خطوة عن طريق prev()

ماذا سيحصل إذا قمنا بإضافة عنصر على مصفوفة غير محدودة الفهرسة ؟!

لنفرض أن لدينا مصفوفة وأضفنا إليها عنصراً غير محدد الفهرسة. مثل:

```
<?
");أحمد44=>"،"محمد5=>"،"علي$s= array (12=>"
";هشام$s[ ]= "
Next($s);
```

```
Next($s);
Next($s);
Echo key ($s).”<br>”;
Echo current($s).”<br>”;
?>
```

سيقوم الـPHP ببساطة بالبحث عن أكبر رقم فهرسة وبعد ذلك يبدأ بإعطاء الفهرسة تسلسلاً بعده فإذا كانت أرقام الفهرسة حروفاً بدأ من الصفر في إعطاء الرقم.. ولاحظ في هذا المثال بأنه قام بإعطاء العنصر الرقم ٤٥ لأن أكبر عنصر في المصفوفة هو ٤٤ وعلى ذلك قام بإعطاء الأرقام تسلسلاً بعد هذا الرقم.

## الدالة List و Each

لنفرض أنك قد قمت بصنع مصفوفة غير مفهرسة بالترتيب

**مثال:**

```
<?
”);أحمد44=>”،”محمد5=>”،”علي$s= array (12=>“
?>
```

على ذلك دعنا نخبرك بخبر سار وهو أنك تستطيع أن تجعل حياتك مع PHP أسهل من حياتك مع نفسك !

```
=each (array) (قيمة العنصر Element value، Index ارقام الفهرسة While (list(
```

تستطيع بواسطة هاتين الدالتين وعن طريق التكرار while استخراج جميع العناصر الموجودة في المصفوفة

```
$r) = each ($s))،While (list($e
{
echo “<br> $e<br> $r”;
}
```

أولاً أنت تقوم بتسمية متغيرين واحد منهما لرقم الفهرسة ($e) والثاني للعنصر ($r) ويمكننا تسميتهما بأي اسم وفي حالة ما إذا أردنا عرض العنصر فقط أو معرفة العنصر فقط فيمكننا حذف ($e) ولكننا لا نحذف الفاصلة

```
$r) = each ($s)،While (list(
{
echo “<br> $e<br> $r”;
}
```

لنعد إلى المثال الذي فيه رقم الفهرسة والعنصر... سيقوم التكرار بوضع رقم الفهرسة (الذي قد يكون نصياً) في المتغير $e وسيضع قيمة العنصر الذي رقم الفهرسة له هو $e في المتغير $r ثم سيقوم بطباعة العناصر حتى ينتهي منها جميعها...

**ملاحظة مهمة**: إذا لم تقم بتعريف فهرسة للمصفوفة (حروف أو أرقام أيا كان) فسيتم استخدام العناصر عندما يطلب التكرار الفهارس.

**مثال:**

```
<?
"tewr")؛"terhfgfd"،$e=array("fsda"
$V)=each($e))،While (list ($I
{
echo "<br>$e[$I]";
}
?>
```

لاحظ مع أننا طلبنا طباعة الفهرسة (index) إلا أنه تم أخذ العناصر (elements) بدلاً من الفهرسة.

يمكننا بواسطة هذه الدالة صناعة أشياء مفيدة وكمثال لذلك لنفرض أن لدينا مصفوفة أرقام هواتف ونريد أن نخرج هذه المصفوفة على جدول html فسنستطيع صناعة هذا الجدول عن طريق التكرار السابق بكل سهولة.

**مثال:**

```
<table align='center' dir = "rtl" border="1" width="100%"
cellspacing="0" bordercolorlight="#000000" bordercolordark="#000000"
bordercolor="#000000">
<tr>
</td>الاسم    <td align='center'>
```

```
</td>رقم التلفون    <td align='center'>
</tr>
<?
);٤٦٥٨٧٣ ،"سالم"=>٤٥٦٥٤٦ ،"عادل"=>"$s = array (658
$r) = each ($s)،While (list($e
{
echo "<tr><td align='center'>". $r. "</td><td align='center'>".$e.
"</td></tr>";
}
?>
</table>
```

أرايت كيف استخرجنا جميع أرقام التلفونات في جدول بواسطة تكرار بسيط، يمكنك صناعة الأكثر واختصار الكثير من الوقت، على ذلك إذا كانت المصفوفة تحتوي على المئات من الأرقام بواسطة هذا الكود بدلاً من أن تكتب الكود على شكل html وتكتب البيانات وتتعب نفسك.

يمكنك أيضاً معرفة عدد العناصر في مصفوفة معينة إذا كنت تريد معرفة عددها وذلك بالطريقة التالية:

```
<?
");أحمد44=>"، "محمد5=>"، "علي$s= array (12=>"
$S=0;
$r) = each ($s))،While (list($E
{
$S++;
}
". $S++; عدد عناصر المصفوفةECHO "
?>
```

## فرز المصفوفات

هناك العديد من الدوال التي يوفرها لنا الـPHP لفرز المصفوفات. نحن سنأخذ نظرة عن الخمسة دوال الأكثر استخداماً:

### الدالة Sort()

هذه الدالة من أساسيات فرز المصفوفات وهي جداً أساسية وهي تقوم بأخذ محتويات المصفوفة ومن ثم تقوم بفرزها هجائياً اعتماداً على الأحرف الكبيرة أولا ثم الصغيرة.. تتطلب هذه الدالة اسم المصفوفة التي سيتم عليها الفرز

```
Sort (ArrayName);
```

إذا قمنا بإنشاء مصفوفة بالشكل التالي:

```
;)"Hesham"،"Ammar"،"Khaled"،"hythem"،"salem"،$NaNo=array ("ali"
```

فإذا أردنا فرزها عن طريق الدالة sort() فإننا نقوم باستخدامها كالتالي:

```
<?
;)"Hesham"،"Ammar"،"Khaled"،"hythem"،"salem"،$NaNo=array ("ali"
sort($NaNo);
$r) = each ($NaNo))،While (list($e
{
echo "<br> $e<br> $r";
}
?>
```

لاحظ أنه عند تنفيذك للمثال ستجد أن الـPHP قام بالفرز اعتماداً على الأحرف الكبيرة أولاً ثم قام بالفرز بعدها اعتماداً على الأحرف الصغيرة.

### الدالة Arsort()

هذه الدالة تعمل نفس عملية الدالة sort() ولكن هناك اختلاف بسيط فمثلاً لو كتبنا المصفوفة كالتالي:

```
 “kh”=> “khaled”); ،$NaNo=array (“ad”=>”ahmed”
```

وأردنا فرزها وطباعة الفهارس والقيم كما في المثال التالي:

```
<?
 “kh”=> “khaled”); ،$NaNo=array (“ad”=>”ahmed”
sort($NaNo);
$r) = each ($NaNo))،While (list($e
{
echo “<br> $e<br> $r”;
}
?>
```

قارن ناتج المثال السابق مع هذا المثال:

```
<?
 “kh”=> “khaled”); ،$NaNo=array (“ad”=>”ahmed”
asort($NaNo);
$r) = each ($NaNo))،While (list($e
{
echo “<br> $e<br> $r”;
}
?>
```

اعتقد انك قد عرفت الفرق ففي المثال الأول قامت الدالة sort باستبدال الحروف بأرقام في الفهرسة أما في المثال الثاني فقد تم وضع الحروف كما هي وتم فرزها كما تفعل الدالة sort في الفرز.

باختصار لايوجد فرق بين sort و asort إلا في أن الدالة sort تستبدل فهرسة الحروف بأرقام.

### الدالة Rsort() و arsort

تقوم بنفس عمل sort و asort ولكن بشكل عكسي جرب الأمثلة التالية:

**مثال:**

```
<?
 “kh”=> “khaled”); ،$NaNo=array (“ad”=>”ahmed”
rsort($NaNo);
```

```
$r) = each ($NaNo)،While (list($e
{
echo “<br> $e<br> $r”;
}
?>
```

**مثال:**

```
<?
 “kh”=> “khaled”); ،$NaNo=array (“ad”=>”ahmed”
arsort($NaNo);
$r) = each ($NaNo)،While (list($e
{
echo “<br> $e<br> $r”;
}
?>
```

ستجد أن الدالة rsort تقوم بنفس عملية الدالة sort ولكن بشكل عكسي.

أيضاً الدالة arsort تقوم بنفس عملية asort ولكن بشكل عكسي.

يمكنك استعمال كل هذه الدوال في الفرز مع الحروف العربية (إذا كان السيرفر يدعم اللغة العربية)

قم بتطبيق المثال التالي:

```
RSORT()
<?
"); أحمد "kh"=> "،"سالم$NaNo=array ("ad"=>"
rsort($NaNo);
$r) = each ($NaNo)،While (list($e
{
echo "<br> $e<br> $r";
}
?>
<br>--------<br>
ARSORT()
<?
"); أحمد "kh"=> "،"أحمد$NaNo=array ("ad"=>"
arsort($NaNo);
$r) = each ($NaNo)،While (list($e
```

```
{
echo "<br> $e<br> $r";
}
?>
<br>--------<br>
ASORT()
<?
"); جمال "kh"=> "،"هاشم$NaNo=array ("ad"=>"
asort($NaNo);
$r) = each ($NaNo))،While (list($e
{
echo "<br> $e<br> $r";
}
?>
<br>--------<br>
SORT()
<?
"); جمال "kh"=> "،"هاشم$NaNo=array ("ad"=>"
sort($NaNo);
$r) = each ($NaNo))،While (list($e
{
echo "<br> $e<br> $r";
}
?>
```

### الدالة ksort

تكلمنا سابقاً عن طريقة فرز المصفوفات ولكن نريد أن نلفت نظرك أننا كنا نعتمد على العنصر في الفرز (element) ولكن هذه الدالة تقوم بالاعتماد على رقم الفهرسة في الفرز (index)

**مثال:**

```
<br>--------<br>
asort()
<?
"); جمال "kh"=> "،"هاشم$NaNo=array ("ad"=>"
asort($NaNo);
$r) = each ($NaNo))،While (list($e
{
```

```
echo "<br> $e<br> $r";
}
?>
<br>--------<br>
ksort()
<?
"); جمال "kh"=> "،"هاشم$NaNo=array ("ad"=>"
ksort($NaNo);
$r) = each ($NaNo))،While (list($e
{
echo "<br> $e<br> $r";
}
?>
```

لقد اعتمد الـphp على index ولم يعتمد على الـelement في الفرز.

## دوال المصفوفات الإضافية

هناك الكثير من الدوال التي يمنحنا إياها الـPHP للتعامل مع المصفوفات والتي لا يكفي الوقت لذكرها الآن وسنقوم بشرح أهم دالتين والمستخدمتين بكثرة وهي **array_push()** و **array_pop()**

لنفرض أننا قمنا بإنشاء مصفوفة بالشكل التالي:

```
<?
$saher[ 5]=”salem”;
$saher[ 85]=”khaled”;
$saher[ 35]=”mohmed”;
$saher[ 19]=”hajeer”;
?>
```

وأردنا أن نضيف عنصراً جديد لها فقمنا بالتالي:

```
<?
$saher[ 5]=”salem”;
$saher[ 85]=”khaled”;
$saher[ 35]=”mohmed”;
$saher[ 19]=”hajeer”;
$saher[ ]=”Alfarees”;
?>
```

انظر إلى العنصر الأخير الذي سيعطيه الـPHP رقم الفهرسة (index) وسيكون رقم فهرسته هو 86.

نريد أن نلفت نظرك بأننا نستطيع عمل إضافة لعنصر على المصفوفة بطريقة أخرى وهي عن طريق الدالة array_push() كالتالي:

array_push (ArrayName اسم المصفوفه، Elemnt1، Elemnt2، Elemnt3، .....)

نضع في القسم الأول من الدالة اسم المصفوفة التي نريد إضافة العنصر لها ونضع في القسم الثاني عنصر واحد أو أكثر وهي التي سيتم إضافتها للمصفوفة.

**مثال:**

```
<?
$saher=array("salem"=>5);
$saher=array("khaled"=>85);
$saher=array("mohmed"=>35);
$saher=array("hajeer"=>19);
array_push($saher, "Alfarees");
?>
```

**مثال:**

```
<?
$saher=array("salem"=>5);
$saher=array("khaled"=>85);
$saher=array("mohmed"=>35);
$saher[ 19]="hajeer";
array_push ($saher, "Alfarees", "salem", "sameer", "thamer");
?>
```

ولو أردنا حذف مثلاً عنصر من المصفوفة فإننا نقوم بتعريف المصفوفة من جديد أو يمكننا استخدام الدالة array_pop التي تقوم بحذف آخر عنصر من المصفوفة والتي تتطلب فقط اسم المصفوفة

Array_pop(ArrayName اسم المصفوفة)

**مثال:**

```
<?
$saher[ 5]=”salem”;
$saher[ 85]=”khaled”;
$saher[ 35]=”mohmed”;
$saher[ 19]=”hajeer”;
array_pop($saher)
?>
```

سيتم حذف العنصر hajeer من المصفوفة ولن يكون في المصفوفة غير ثلاثة عناصر.

## Implode و Explode

تقوم هاتين الدالتين باقتصاص قيمة معينة من مصفوفة أو نصوص وتقوم بإضافة قيمة معينة على مصفوفة أو نصوص.

### الدالة Implode

تقوم بإضافة قيمة على عنصر بين عناصر المصفوفة.

**مثال:**

```
<?
 "alfarsi"); ، "ali"، "salem"،$stng =array ("ahmed"
$stng);،$r =implode ("H"
echo $r;
?>
```

### الدالة explode

تقوم بحذف قيمة من مصفوفة  وذلك لا يعني حذف عناصر من  المصفوفة.

**مثال:**

```
<?
 "alfarsi"); ، "ali"، "salem"،$stng =array ("ahmed"
```

```
$stng);،$r =implode ("-"
echo $r;
$stng);،$r= explode ("-"
echo $r;
?>
```

## HTTP_GET_VARS و HTTP_POST_VARS

هذه ليست متغيرات بل مصفوفات، نعم هذه مصفوفات ولكن في ماذا نستخدمها ولماذا؟

في الواقع تحدثنا في الفصل السابق عن طريقة التعامل مع النماذج والحصول على البيانات من المستخدم وتكلمنا عن أسلوبين لنقل البيانات وهما GET و POST عندما تصل البيانات محفوظة في متغيرات إلى صفحة الـPHP فإنه يقوم بتعريفها تلقائياً ويمكنك طباعة المتغيرات وقيمها مباشرة من غير تعريف.... ولكن هذه الميزة في الـPHP يمكن إلغاؤها عن طريق الملف PHP.INI وذلك بإغلاق ميزة register_globals
وذلك بوضع off بدلاً من on

الوضع الافتراضي لها هو on ولكن تستطيع إغلاقها وقد تكون مستأجراً عند مزود خدمة ويب وسيط فيقوم بإغلاق هذه الميزة من باب الحماية ليس إلا.... لا تقلق يمكنك الحصول على البيانات فهي ما زالت موجودة ولكن يجب عليك أن تقوم باستخدام هاتين المصفوفتين لكي تستخرج البيانات.

لنفرض أنك اشتركت عند مزود ويب وكان قد أغلق ميزة (register_globals) حسناً لنفرض أنك قد صنعت نموذجاً يستخدم مربع نص ويحفظ قيمته في متغير اسمه Dorrah ثم بعد ذلك يقوم بإرسال هذه القيمة باستخدام الأسلوب GET إذا سيكون جزء من الكود في الصفحة الأولى والتي تحتوي على النموذج كالتالي:

```
<form method =get action = “try.php”>
ماهو اسم الطفل الذي استيقظ به العالم الاسلامي من غفلته قبل عدة شهور !!
<br>
<input type=text name = “Dorrah”>
<br>
```

في الملف الثاني(try.php) سنقوم بكتابة الجزء الذي سيقوم بطباعة القيمة كالتالي:

```
<?
Echo HTTP_GET_VARS[“Dorrah”];
?>
```

لاحظ أننا لم نستخدم $ ولكن إذا أردنا الاحتفاظ بقيمة المتغير في متغير آخر فيمكننا ذلك بشكل عادي كالتالي:

```
<?
$Dorrah= HTTP_GET_VARS[“Dorrah”];
?>
```

طريقه بسيطة.... أليس كذلك ولكن.... لنفترض أن مزود خدمة الويب لديك حريص جداً ولذلك فقد ألغى أيضاً ميزة استقبال هذه القيم في المصفوفات.... يمكنه ذلك في ملف الـphp.ini في إعدادات الـtrack_vars الذي يقوم بمنع السيرفر من استخدام هذه المصفوفات (هذه الميزة يمكن إلغاؤها في php4).... على ذلك انصحك بإرسال رسال تذمر وشكوى إلى مزود الخدمة لديك.. تعلن فيها أن الأمر أصبح لا يحتمل.

## مصفوفة متعددة الأبعاد

يمكنك صناعة مصفوفات بداخل مصفوفات على حسب ما تحتاجه في معلوماتك الرياضية فقد تحتاج مثلاً إلى إنشاء أشياء معقدة (ومقلقة نفسياً) نريد أن نخبرك على أية حال أنه يمكنك صناعة المصفوفات المتعددة الأبعاد ويمكنك استخدام حتى مئة مصفوفة متداخلة ولكن يجب أن تراعي حجم الذاكرة المستخدمة في السيرفر

لديك (وعلى كل حال إن استطعت أن تقوم بالتركيز في صناعة عشر مصفوفات متداخلة بدون أي مشاكل أو مرض نفسي أو.... فأنت تستحق جائزة).

يمكننا كتابة مصفوفة متداخلة كالتالي:

```
<?
2 => array ("salem ، 154786)،$mon= array (1=>array ("sharkeh al-jafali"
1257)); ، almazen"
while (list($personnum) =each ($mon))
{
echo ("<br>$personnum<br>");

$phone)=each ($mon[$personnum]))،while (list(
{
echo ("$phone");
}
}
?>
```

**الشرح**

هذا المثال قد يكون غامضاً جداً لكن فكرته بسيطة أولاً افترض أنك تعلم عن list..each جيداً وتعرف صيغة التكرار الذي يستخدمهما.

الآن لدينا مصفوفة تتكون من رقمين للفهرسة هذين الرقمين كل واحد منهما عنصره عبارة عن مصفوفة هذه المصفوفة تحتوي على عنصرين (ولنتناسى أنهما يحتويان على أرقام فهرسة) وهما اسم شخص ورقم هاتفه.

**في أول خطوة:**

```
while (list($personnum) =each ($mon))
{
echo (“<br>$personnum”);
```

قمنا بإخراج رقم الفهرسة الأساسي للمصفوفة والذي يعتبر هو الرقم التسلسلي للأشخاص أصحاب الهواتف ومن بعد ذلك نقوم بطباعة هذا الرقم التسلسلي ويبدأ من سطر جديد.

**في الخطوة الثانية:**

```
$phone)=each ($mon[$personnum]))،while (list(
{
echo ("$phone");
}
```

نقوم بإخبار الـPHP بطباعة العناصر الذي تحتويها المصفوفة التي تمت طباعة رقم فهرستها، ولاحظ )،(**$phone** أنها تشير إلى عناصر مصفوفة وليس فهارسها لأننا تجاهلنا فهارس المصفوفة الداخلية.

لا تقلق الأمر سهل ولكنه يحتاج إلى تدرب فقط، وعليك أن تتدرب وصدقني أنني حاولت أن أبسط المثال من أجلك... أتمنى أن تكون قد فهمت.

**تطبيق عملي**

افتح محرر النصوص لديك واكتب الكود التالي:

```
<?
Echo "<form method =post action = 'exam2.php' " ;
");حسن"،"**سعد**"،" خالد"،"أحمد$boy=array ("
$Name) = each ($boy))،while (list(
{
";؟$Name ماهي السنة الدراسية لـ echo "
Echo "<select name = 'school[]'>
</option>أول ثانوي<option>
</option>ثاني ثانوي<option>
</option>ثالث ثانوي<option>
</select>";
echo "<br><br>";
```

```
echo "<input type =hidden name =boy[] value ='$Name'>";
}
echo "<input type =submit ></form>";
?>
```

احفظ الكود باسم exam.php

افتح محرر النصوص واكتب الكود التالي واحفظه في ملف باسم exam2.php

```
<html dir = "rtl">
<?
$V)=each($school)،While (list($I
{
$friendschool[] = $school[$I].$boy[$I];
}
asort ($friendschool);
$V)=each($friendschool)،While (list ($I
{
echo "<br>$boy[$I]"." ".$school[$I];
}
?>
```

قم بتشغيله بعد نقله لمجلد السيرفر.

**الشرح**

الذي قمنا به في المثال السابق هو أننا قمنا بإنشاء مصفوفة لعدة أشخاص ($boy) ونريد أن نعرف مرحلهم الدراسية في الثانوية فأنشأنا لكل طالب قائمة منسدلة بواسطة التكرار (list-each) بصناعة قوائم منسدلة وحقول مخفية يتم تخزين قيم الحقول (التي تحتوي على أسماء الأشخاص) في المصفوفة ($boy) وسيتم تخزين نتائج كل القوائم في مصفوفة ($school) وبعد أن يختار المستخدم الإجابات التي تناسبه ويرسل البيانات سيتم استقبال المصفوفة التي فيها نتائج القوائم المنسدلة ($school) واستقبال المصفوفة التي فيها أسماء الأشخاص ($boy) ومن ثم يتم إنشاء مصفوفة جديدة باسم $friendschool[] ويؤخذ

منها معلومات المصفوفتين ويتم دمجها فيها ومن ثم يتم بتكرار آخر طباعة عناصر المصفوفتين $boy و$school.

## تكرار foreach

هذا التكرار هو من الأشياء الجديدة في الـphp4 وهو يساعدك على معرفة عناصر مصفوفة معينة أو طباعة محتوياتها.

```
Foreach ($ArrayName As $ArrayItem)
{
code شيفره
}
```

**مثال:**

```
<?
 c=>"car")، b => "basem"، $T= array (a=>"ahmed "

Foreach ($T As $A => $r)
{
echo $A."-----". $r;
}
?>
```

## الدالة count

تقوم بحساب عدد العناصر الموجودة في المصفوفة

**مثال:**

```
<?
"c")،"b"،$c=array("a"
$v=count($c);
echo $v;
?>
```