Πίνακες

Στο κεφάλαιο 2 (μεταβλητές και τύποι δεδομένων) αναφέραμε ότι ένας από τους τύπους δεδομένων της PHP είναι οι πίνακες (**arrays**). Οι πίνακες είναι μια κατηγορία μεταβλητών οι οποίες μπορούν να αποθηκεύσουν πολλαπλές τιμές υπό ένα κοινό όνομα. Φυσικά αυτό απαιτεί πολλαπλές θέσεις μνήμης και δεν είναι δυνατό να εντοπίσουμε μια τιμή μέσα στο πίνακα απλά με το όνομα του πίνακα. Έτσι θα μπορούσαμε να πούμε ότι ένας πίνακας είναι μια μεταβλητή με πολλές θέσεις για δεδομένα, κάθε μια από τις οποίες χαρακτηρίζεται από ένα μοναδικό κλειδί. Ας πούμε για παράδειγμα ότι έχουμε ένα πίνακα με το όνομα $**table** όπως αυτός που εικονίζεται παρακάτω.

	0	1	2	3	4	5	6
$**table**	25	38	8	45	62	19	45

Ο παραπάνω πίνακας είναι ένας πίνακας 7 στοιχείων. Μέσα στα κελιά έχουμε τις τιμές που είναι αποθηκευμένες στο πίνακα ενώ από πάνω βλέπουμε το κλειδί του κάθε κελιού. Το κλειδί μας βοηθά να έχουμε πρόσβαση στο περιεχόμενο του κελιού. Έτσι αν θέλουμε για παράδειγμα να τυπώσουμε το περιεχόμενο του κελιού 3 θα πρέπει να χρησιμοποιήσουμε το όνομα του πίνακα ($table) συνοδευόμενο από το κλειδί του κελιού που μας ενδιαφέρει (3) ως εξής:

```
echo $table[3];
// θα εμφανίσει 45

//άλλα παραδείγματα
echo $table[1]+$table[2]+$table[5];
// θα εμφανίσει 65

$t = 1;
echo $table[$t]+$table[$t+1]+$table[$t+4];
// ισοδυναμεί με το προηγούμενο και θα εμφανίσει 65
```

Τα χρησιμοποιούμενα κλειδιά πρέπει να είναι αυστηρά μέσα στα όρια των κλειδιών του πίνακα. Για παράδειγμα δεν μπορούμε να ζητήσουμε το $**table**[9]. Επειδή τέτοιος δείκτης δεν υπάρχει στον πίνακα το Script θα τερματιστεί με λάθος για εσφαλμένη αναφορά θέσης μνήμης.

Η ανάθεση τιμών σε ένα κελί γίνεται με τον ίδιο τρόπο

```
$table[3] = 2;
echo $table[3];
// θα εμφανίσει 2

$table[3] = $table[1]+$table[2];
echo $table[3];
// θα εμφανίσει 46
```

Οι πίνακες ως δυναμική δομή

Οι πίνακες στην PHP έχουν σημαντικές διαφορές με άλλες κλασσικές αντικειμενοστραφείς γλώσσες όπως η Pascal και η C++. Οι πίνακες στις περισσότερες γλώσσες είναι στατικές δομές στις οποίες πρέπει να δηλωθεί εξ' αρχής το μέγεθός τους και ο τύπος των δεδομένων που θα αποθηκεύουν. Το μέγεθος δε μπορεί να τροποποιηθεί κατά την εκτέλεση του προγράμματος και όλα τα κελία παίρνουν όλα το ίδιο τύπο δεδομένων. Η υλοποίηση δυναμικών πινάκων στις περισσότερες γλώσσες είναι μια σύνθετη διαδικασία. Αντίθετα στην PHP οι πίνακες είναι εξ' ορισμού δυναμικές δομές, δηλαδή **δεν χρειάζεται να δηλώσουμε το μέγεθός τους**. Η PHP διαχειρίζεται αυτόματα τους πίνακες και την απαιτούμενη μνήμη.

Άλλη μια σημαντική διαφορά στο τρόπο που χειρίζεται η PHP τους πίνακες είναι το γεγονός ότι **σε ένα πίνακα μπορούμε να συνδυάσουμε δεδομένα διαφορετικών τύπων**. Έτσι ένα κελί του πίνακα μπορεί να περιέχει έναν ακέραιο, ένα άλλο έναν πραγματικό και ένα τρίτο να περιέχει έναν δεύτερο πίνακα. Ένας τέτοιος πίνακας θυμίζει πολύ τις εγγραφές (records) που χρησιμοποιούνται σε άλλες γλώσσες.

Τέλος η PHP επιτρέπει μια πολύ ευέλικτη χρήση των κλειδιών των κελιών που θα δούμε αργότερα. Έτσι τα κλειδιά μπορεί να είναι αριθμοί ή κείμενα, σε σειρά ή μπερδεμένα.

Αυτά τα τρία χαρακτηριστικά των πινάκων της PHP μαζί με ένα μεγάλο σύνολο έτοιμων συναρτήσεων, τους κάνουν πολύ ευέλικτους και εύκολους στη χρήση. Απαιτείται βέβαια πολύ καλή γνώση και αντίληψη της δομής που δουλεύουμε κάθε φορά.

Δημιουργία πίνακα

Ας δούμε λοιπόν πως μπορεί να δημιουργηθεί ένας πίνακας στη PHP. Ας υποθέσουμε ότι θέλουμε να φτιάξουμε ένα πίνακα για την αποθήκευση τριών ονομάτων, όπως αυτός που παρουσιάζεται παρακάτω.

	0	1	2
$names	Νίκος	Γεωργία	Βαγγέλης

Ο πιο εύκολος τρόπος είναι να δημιουργήσουμε τον πίνακα σε μια εντολή ως εξής

```
$names = array('Νίκος', 'Γεωργία', 'Βαγγέλης');
```

Σε αυτή την εντολή δηλώνουμε ότι η μεταβλητή $names θέλουμε να είναι πίνακας και στη παρένθεση τοποθετούμε τα περιεχόμενα που θέλουμε να έχει. Θα παραχθεί μια δομή ακριβώς όπως αυτή που αναφέραμε αρχικά, με κλειδιά 0,1 και 2. Τα κλειδιά αυτά τα διαχειρίστηκε η PHP αυτόματα.

Αν για κάποιο λόγο δεν είχαμε διαθέσιμα όλα τα ονόματα και θέλαμε να τα βάλουμε στο πίνακα τμηματικά θα μπορούσαμε να χρησιμοποιήσουμε τον παρακάτω κώδικα.

```
$names = array();
$names[] = 'Νίκος';
```

```
$names[] = 'Γεωργία';
$names[] = 'Βαγγέλης';
```

Από τον παραπάνω κώδικα παίρνουμε το ίδιο αποτέλεσμα. Η πρώτη εντολή απλά δηλώνει ότι η μεταβλητή **$names** είναι πίνακας. Η κάθε μια από τις υπόλοιπες τρεις εντολές προσθέτει ένα στοιχείο στον πίνακα. Και σε αυτό το παράδειγμα αφήνουμε την PHP να διαχειριστεί τους δείκτες (κλειδιά) του πίνακα. Η χρήση της άδειας αγκύλης σημαίνει «**δημιούργησε ένα νέο στοιχείο με κλειδί το επόμενο ελεύθερο**».

Ένας τέτοιος πίνακας με αριθμητικά κλειδιά ονομάζεται **indexed array**.

Αν θέλαμε να δημιουργήσουμε έναν πίνακα όπου θα είχαμε συγκεκριμένα κλειδιά τότε θα έπρεπε να τα ορίσουμε εμείς ώστε να αποτρέψομε τη PHP να χρησιμοποιήσει τα προεπιλεγμένα. Έστω ότι θέλουμε να φτιάξουμε το πίνακα

	15	17	22
$names	Νίκος	Γεωργία	Βαγγέλης

Ο κώδικας που παρουσιάσαμε νωρίτερα τροποποιείται σε

```
$names = array(15=>'Νίκος', 17=>'Γεωργία', 22=>'Βαγγέλης');

//Όταν τα στοιχεία είναι πολλά
//μπορεί να δείτε και τη παρακάτω γραφή
$names = array(
      15=>'Νίκος',
      17=>'Γεωργία',
      22=>'Βαγγέλης'
);

//Αν θέλαμε να δημιουργήσουμε το πίνακα
//με ξεχωριστές και όχι μια εντολή θα είχαμε
$names = array();
$names[15] = 'Νίκος';
$names[17] = 'Γεωργία';
$names[33] = 'Βαγγέλης';
```

Ένα ζευγάρι τιμών που έχουν ανάμεσα το σύμβολο '=>' αντιπροσωπεύει ένα ζευγάρι κλειδιού – τιμής. Αν θέλαμε θα μπορούσαμε να ορίσουμε μη αριθμητικά κλειδιά με τον ίδιο τρόπο. Έστω ο παρακάτω πίνακας για την αποθήκευση των στοιχείων ενός σπουδαστή.

	firstname	lastname	age	gender
$student	Νίκος	Παπαδόπουλος	28	Άρρεν

Ακολουθούν παραδείγματα κώδικα που μπορούν να χρησιμοποιηθούν για τη δημιουργία ενός τέτοιου πίνακα.

```
$student = array('firstname'=>'Νίκος',
=>'lastname'=>'Παπαδόπουλος', 'age'=>28, 'gender'=> 'Άρρεν');
```

```
//Όταν τα στοιχεία είναι πολλά
//μπορεί να δείτε και τη παρακάτω γραφή
$student = array(
      'firstname'=>'Νίκος',
      'lastname'=>'Παπαδόπουλος',
      'age'=>28,
      'gender'=> 'Άρρεν'
);

//Αν θέλαμε να δημιουργήσουμε το πίνακα
//με ξεχωριστές και όχι μια εντολή θα είχαμε
$student = array();
$student['firstname'] = 'Νίκος';
$student['lastname'] = 'Παπαδόπουλος';
$student['age'] = 28;
$student['gender'] = 'Άρρεν';
```

Ένας τέτοιος πίνακας στη PHP που χρησιμοποιεί μη αριθμητικούς δείκτες, ονομάζεται **associative array**. Σε ένα τέτοιο πίνακα η χρήση των δεικτών είναι σκόπιμη άρα τα ζευγάρια κλειδιού-τιμής που προκύπτουν έχουν κάποιο νόημα και δεν πρέπει να χαθεί η σχέση τους, αλλιώς όλος ο πίνακας μπορεί να είναι άχρηστος και η πληροφορία λανθασμένη.

Σάρωση του πίνακα με εντολή επανάληψης

Οι τιμές του πίνακα μπορούν να χρησιμοποιηθούν μεμονωμένα εφόσον το επιθυμούμε αλλά συνήθως χρειάζεται να εφαρμόσουμε κάποια λειτουργία σε όλα τα στοιχεία του πίνακα, όπως για παράδειγμα να εμφανίσουμε όλα τα στοιχεία του. Η εφαρμογή επαναληπτικών διαδικασιών σε μια μεγάλη ποσότητα δεδομένων είναι έτσι κι' αλλιώς η βασική ανάγκη που έχει οδηγήσει στην επινόηση αυτής της δομής.

Ας θεωρήσουμε ότι έχουμε τον παρακάτω **indexed** πίνακα.

	0	1	2
$names	Νίκος	Γεωργία	Βαγγέλης

Ο πίνακας αυτός έχει απλούς αριθμητικούς δείκτες που αρχίζουν από το 0. Σε αυτή τη περίπτωση θα μπορούσαμε να χρησιμοποιήσουμε μια απλή εντολή for για να σαρώσουμε το πίνακα. Παρακάτω παρουσιάζεται ένα παράδειγμα κώδικα με τον οποίο μπορούμε να εμφανίσουμε τα στοιχεία του πίνακα μη επαναληπτικά.

```
echo $names[0];
echo $names[1];
echo $names[2];
```

Κάτι τέτοιο φαίνεται απλό και χρηστικό αλλά θα ήταν αδύνατο αν ο πίνακας είχε μεγάλο αριθμό στοιχείων ή αν το πλήθος των στοιχείων άλλαζε συνεχώς. Θα μπορούσαμε λοιπόν να σαρώσουμε το πίνακα με τη βοήθεια της παρακάτω εντολής **for**.

```
for ($i=0;$i<=3;$i++) {
      echo $names[$i];
}
```

Στο παραπάνω κώδικα χρησιμοποιούμε μια επαναληπτική εντολή **for** όπου ο μετρητής παίρνει τιμές όμοιες με αυτές των δεικτών του πίνακα. Στη συνέχεια οι τιμές του μετρητή χρησιμοποιούνται εντός της επανάληψης για τη σωστή αναφορά στις διάφορες θέσεις του πίνακα. Ο χειρισμός του μετρητή πρέπει να είναι προσεκτικός για να μην αναφερθούμε κατά λάθος σε ανύπαρκτη θέση του πίνακα.

Ας θεωρήσουμε τώρα ότι ο πίνακας έχει κανονική αρίθμηση στους δείκτες ξεκινώντας από το 0 αλλά δεν είμαστε σίγουροι για το πόσα είναι τα στοιχεία του πίνακα. Σε αυτή τη περίπτωση θα έπρεπε να τροποποιήσουμε τον προηγούμενο κώδικα ώστε να ορίσουμε το άνω όριο του μετρητή με τη συνάρτηση **count**.

```
for ($i=0;$i<count($names);$i++) {
      echo $names[$i];
}
```

Η συνάρτηση **count** δέχεται σαν όρισμα έναν πίνακα και επιστρέφει το πλήθος των στοιχείων του. Έτσι μπορούμε να σαρώσουμε έναν indexed πίνακα χωρίς να γνωρίζουμε ακριβώς τον αριθμό των στοιχείων του.

Στη συνέχεια θα θεωρήσουμε έναν πίνακα όπου οι δείκτες είναι αριθμητικοί αλλά δεν είναι κανονικοί (δεν είναι συνεχόμενοι ή δεν ξεκινάνε από 0 ή είναι μπερδεμένοι). Σε αυτή τη περίπτωση δεν μπορούμε να χρησιμοποιήσουμε τη for με τον απλό τρόπο που είδαμε. Μπορούμε με άλλους τρόπους αλλά η πολυπλοκότητα που προκύπτει μπορεί να αποφευχθεί με τη χρήση μιας νέας μορφής της εντολής **for** ειδικά για πίνακες, την **foreach**. Η εντολή αυτή χρησιμοποιείται και στους **associative** πίνακες.

	15	17	22
$names	Νίκος	Γεωργία	Βαγγέλης

Η **foreach** μπορεί να εφαρμοστεί στον παραπάνω πίνακα χωρίς να χρειάζεται να ασχοληθούμε καθόλου με δείκτες, μετρητές και συνθήκες τερματισμού. Ουσιαστικά η **foreach** σαρώνει αυτόματα τον πίνακα και μας δίνει τη δυνατότητα να έχουμε πρόσβαση στο στοιχείο του πίνακα στο οποίο βρίσκεται σε κάθε επανάληψη.

```
foreach ($names as $name) {
      echo $name;
}
```

Η παραπάνω εντολή είναι σαφώς πιο εύκολη στη σάρωση ενός πίνακα από ότι είναι η κλασσική for. Στη δήλωση της εντολής πρέπει να δώσουμε το όνομα του πίνακα που θέλουμε να σαρώσουμε και ένα όνομα μεταβλητής με το οποίο η **foreach** θα μας δίνει πρόσβαση στο στοιχείο που σαρώνει σε κάθε επανάληψη. Το όνομα της δεύτερης

μεταβλητής ορίζεται από εμάς όπως όλα τα ονόματα των μεταβλητών. Αν θέλαμε να περιγράψουμε σε ελεύθερη γλώσσα την παραπάνω εντολή foreach θα μπορούσαμε να πούμε: «**Σάρωσε όλα τα στοιχεία του πίνακα $names και βάλε στην μεταβλητή $name το τρέχον στοιχείο κάθε επανάληψης**».

Στο προηγούμενο παράδειγμα αδιαφορούμε για τους δείκτες και θέλουμε απλά πρόσβαση στα στοιχεία. Αν για κάποιο λόγο θέλαμε να ξέρουμε και τον δείκτη του κάθε κελιού από το οποίο περνάει η **foreach** θα την συμπληρώναμε ως εξής:

```
foreach ($names as $key => $name) {
      echo 'Το όνομα '.$name.' βρίσκεται στο κελί με δείκτη '.$key;
}
```

Σε αυτή τη μορφή της **foreach** συμπληρώνουμε μια ακόμα μεταβλητή $key (ή όποια άλλο όνομα θέλουμε). Σε αυτή τη μεταβλητή η εντολή θα αναθέσει τον δείκτη του στοιχείου που σαρώνει εκείνη τη στιγμή. Ας θυμηθούμε ότι η έκφραση 'δείκτης' => 'τιμή' που χρησιμοποιήσαμε στα παραδείγματα δημιουργίας πίνακα, δηλώνει ένα ζευγάρι δείκτη-τιμής. Άρα η εντολή **foreach ($key => $name)** θα μπορούσε να ερμηνευτεί ως: «**Σάρωσε όλα τα στοιχεία του πίνακα $names και βάλε στην μεταβλητή $key τον τρέχον δείκτη και στη $name το τρέχον στοιχείο κάθε επανάληψης**».

Θα χρησιμοποιήσουμε τη τελευταία μορφή της foreach για να σαρώσουμε τον παρακάτω πίνακα.

	firstname	lastname	age	gender
$student	Νίκος	Παπαδόπουλος	28	Άρρεν

Από τα στοιχεία αυτά θα εμφανίσουμε έναν HTML πίνακα με τα κλειδιά στη πρώτη στήλη και τις τιμές κάθε κελιού στη δεύτερη, όπως παρουσιάζεται παρακάτω:

firstname	Νίκος
Lastname	Παπαδόπουλος
age	28
gender	Άρρεν

```
<?php
echo '<table>';
foreach ($student as $key => $element) {
      echo '<tr>';
      echo '<td>'.$key.'</td>';
      echo '<td>'.$element.'</td>';
      echo '</tr>';
}
echo '</table>';
?>
```

```
<!—εναλλακτικά βάζοντας PHP μέσα στην HTML-->
<table>
<?php foreach ($student as $key => $element) { ?>
      <tr>
      <td><?php echo $key; ?></td>
      <td><?php echo $element; ?></td>
      </tr>
<php } ?>
</table>
```

Παραδείγματα

- Θα κατασκευάσουμε ένα πίνακα με τυχαίους αριθμούς από -100 έως 100 με τη βοήθεια της rand. Το πλήθος των αριθμών μας είναι άγνωστο και θα σταματήσουμε τη παραγωγή τους μόλις η rand επιστρέψει 0. Αφού ολοκληρωθεί η δημιουργία του πίνακα θα υπολογίσουμε το μέσο όρο των αριθμών. Το μηδέν δε θα συμπεριληφθεί στα αποτελέσματα.

```
<?php
//αρχικοποίηση του πίνακα
$numbers = array();

//παραγωγή των αριθμών και αποθήκευση στον πίνακα
do {
      $t = rand(-100,100);
      if ($t != 0)
            $numbers[] = $t;
} while ($t != 0);

//υπολογισμός αθροίσματος
$sum = 0;
foreach ($numbers as $n) {
      $sum += $n;
}

//υπολογισμός μέσου όρου με τη βοήθεια της count
//για την εύρεση του πλήθους των αριθμών
$mo = $sum/count($numbers);
echo $mo;
?>
```

- Θα εμφανίσουμε τον πίνακα που δημιουργήθηκε στο προηγούμενο παράδειγμα σαν έναν HTML πίνακα 10 στηλών. Δεν γνωρίζουμε πόσοι είναι οι αριθμοί άρα δεν ξέρουμε πόσες σειρές θα χρειαστούν. Πρέπει μόνο να αλλάζουμε σειρά κάθε 10 στοιχεία που εμφανίζουμε.

```
<table>
<tr>
<?php
// θεωρούμε ότι ο πίνακας $numbers
// έχει παραχθεί όπως στο παράδειγμα 1
foreach ($numbers as $key => $n) {
      echo '<td>'.$n.'</td>';
      if ($key % 10 == 0)
            echo '</tr><tr>';
}
?>
```

```
</tr>
</table>
```

- Θα υλοποιήσουμε μια ψηφιακή κλήρωση ενός τυχερού παιχνιδιού αριθμών, όπου κληρώνονται 6 νούμερα από ένα σύνολο 49 αριθμών. Τα αποτελέσματα θα αποθηκεύονται σε ένα πίνακα 6 θέσεων και θα εμφανίζονται στο τέλος στη σελίδα. Ένα νούμερο που έχει κληρωθεί δεν πρέπει να ξανα-κληρωθεί.

```
<?php
//αρχικοποίηση του πίνακα
$numbers = array();

//παραγωγή των αριθμών και αποθήκευση στον πίνακα
for ($i=0;$i<=5;$i++) {
	// η παραγωγή ενός αριθμού επαναλαμβάνεται
	// όσο παράγονται αριθμοί που υπάρχουν ήδη στο πίνακα
	do {
		$t = rand(1,49);
	} while (in_array($t,$numbers));

	$numbers[$i] = $t;
}

foreach ($numbers as $n) {
	echo $n.'<br>';
}
?>
```

Πίνακες δυο ή περισσότερων διαστάσεων

Είπαμε ότι τα στοιχεία ενός πίνακα μπορούν να περιέχουν δεδομένα οποιουδήποτε τύπου και μάλιστα διαφορετικοί τύποι να συνδυαστούν στον ίδιο πίνακα. Αυτό σημαίνει ότι μπορεί ένα στοιχείο του πίνακα να είναι και αυτό πίνακας. Τότε λέμε ότι έχουμε ένα πίνακα με δυο επίπεδα ή δυο διαστάσεις.

Παρατηρήστε το παρακάτω κομμάτι κώδικα που δημιουργεί ένα τέτοιο πίνακα.

```
<?php
$students = array();

//στο πρώτο στοιχείο του πίνακα students αναθέτουμε έναν άλλο πίνακα
$students[] = array(
	'firstname'=>'Νίκος',
	'lastname'=>'Παπαδόπουλος',
	'age'=>28,
	'gender'=> 'Άρρεν'
);
//το ίδιο και στα επόμενα στοιχεία
$students[] = array(
	'firstname'=>'Γεωργία',
	'lastname'=>'Νικολάου',
	'age'=>25,
	'gender'=> 'Θήλυ'
);
$students[] = array(
	'firstname'=>'Κώστας',
	'lastname'=>'Παπακώστας',
	'age'=>22,
	'gender'=> 'Άρρεν'
```

```
);
?>
```

Ένας τέτοιος πίνακας λέγεται δισδιάστατος διότι για να αντλήσουμε κάποιο από τα δεδομένα του χρειαζόμαστε 2 δείκτες, έναν για να προσδιορίσουμε για ποιον μαθητή ενδιαφερόμαστε και ένα για να προσδιορίσουμε το στοιχείο του μαθητή που θέλουμε. Ας δούμε πως θα μπορούσαμε να τυπώσουμε τα στοιχεία του πρώτου μαθητή στην έξοδο.

```
<?php
echo $students[0]['firstname'].'<br>';
echo $students[0]['lastname'].'<br>';
echo $students[0]['age'].'<br>';
echo $students[0]['gender'].'<br>';
?>
```

Για να τυπώσουμε τα στοιχεία όλων των μαθητών θα έπρεπε να χρησιμοποιήσουμε μια επαναληπτική εντολή ως εξής:

```
<?php
foreach ($students as $student) {
      echo $student['firstname'].'<br>';
      echo $student ['lastname'].'<br>';
      echo $student['age'].'<br>';
      echo $student ['gender'].'<br>';
}
?>
```

Ας γυρίσουμε ξανά στο παράδειγμα 3 από τα παραδείγματα που προηγήθηκαν και ας υποθέσουμε ότι δεν θέλουμε να έχουμε αποθηκευμένη σε πίνακα μόνο μια κλήρωση αλλά 100. Συνολικά θα χρειαστούμε 100*6=600 θέσεις μνήμης. Πως όμως είναι το σωστό να τακτοποιήσουμε-δομήσουμε αυτές τις θέσεις. Θα μπορούσαμε να χρησιμοποιήσουμε ένα μεγάλο μονοδιάστατο πίνακα 600 θέσεων αλλά σε αυτή τη περίπτωση τα δεδομένα θα ήταν όλα μαζί αποθηκευμένα. Ο χειρισμός του πίνακα θα ήταν δύσκολος και πολύ εύκολα με μια λάθος λειτουργία θα μπερδεύαμε τα νούμερα των κληρώσεων και θα καταστρέφαμε τη πληροφορία.

Η σωστή αντιμετώπιση είναι να οργανώσουμε τη πληροφορία σεβόμενη το ότι αποτελούν 100 εξάδες. Έτσι ο σωστός τρόπος είναι να φτιάξουμε έναν πίνακα 2 διαστάσεων, 100 X 6. Άρα θα έχουμε έναν βασικό πίνακα 100 θέσεων κάθε στοιχείο του οποίου θα είναι ένας πίνακας 6 θέσεων. Ακολουθεί ο κώδικας για τη δημιουργία του πίνακα και την εκτύπωσή του.

```
<?php
//αρχικοποίηση του πίνακα
$kliroseis = array();

//επανάληψη για τη δημιουργία των 100 κληρώσεων
for ($a=0;$a<=99;$a++) {
      //παραγωγή των αριθμών μιας κλήρωσης
      //σε μια βοηθητική μεταβλητή
      $numbers = array();

      for ($i=0;$i<=5;$i++) {
```

```
            // η παραγωγή ενός αριθμού επαναλαμβάνεται
            // όσο παράγονται αριθμοί που υπάρχουν ήδη στο πίνακα
            do {
                  $t = rand(1,49);
            } while (in_array($t,$numbers));
            $numbers[$i] = $t;
      }
      $kliroseis[$a] = $numbers;
}

//εμφάνιση αποτελεσμάτων
foreach ($kliroseis as $klirosi) {
      foreach ($klirosi as $n) {
            echo $n.' ';
      }
      echo '<br>';
}
?>
```

Όπως βλέπουμε για το σωστό χειρισμό αυτού του δισδιάστατου πίνακα χρειάζονται δυο επαναληπτικές εντολές η μια μέσα στην άλλη (εμφωλευμένες).

Συναρτήσεις πινάκων

Η PHP έχει μια μεγάλη συλλογή έτοιμων συναρτήσεων για πίνακες που κάνουν το χειρισμό τους πιο εύκολο. Στους παρακάτω συνδέσμους μπορείτε να βρείτε λίστα με όλες τις διαθέσιμες συναρτήσεις

http://php.net/manual/en/ref.array.php

http://www.w3schools.com/php/php_ref_array.asp

Εμείς θα αναφέρουμε μόνο κάποιες από τις βασικές συναρτήσεις που έχουμε χρησιμοποιήσει σε διάφορα παραδείγματα κατά τη διάρκεια των μαθημάτων

- **count**(array) – Δέχεται έναν πίνακα και επιστρέφει το πλήθος των στοιχείων του
- **in_array**(value, array) – Δέχεται μια τιμή και έναν πίνακα και επιστρέφει μια λογική τιμή ανάλογα με το αν η τιμή υπάρχει ή όχι στον πίνακα.
- **sort**(array) – Ταξινομεί τα στοιχεία του πίνακα σε **αύξουσα** τάξη μεγέθους. Χρησιμοποιείται σε **indexed** και **όχι** σε **associative** πίνακες.
- **rsort**(array) – Ταξινομεί τα στοιχεία του πίνακα σε **φθίνουσα** τάξη μεγέθους. Χρησιμοποιείται σε **indexed** και **όχι** σε **associative** πίνακες.
- **asort**(array) – Ταξινομεί τα στοιχεία του πίνακα σε **αύξουσα** τάξη μεγέθους. Χρησιμοποιείται κυρίως σε **associative** πίνακες.
- **arsort**(array) – Ταξινομεί τα στοιχεία του πίνακα σε **φθίνουσα** τάξη μεγέθους. Χρησιμοποιείται κυρίως σε **associative** πίνακες.
- **ksort**(array) – Ταξινομεί τα στοιχεία του πίνακα σε **αύξουσα** τάξη μεγέθους του κλειδιού και όχι του στοιχείου του ίδιου.
- **krsort**(array) – Ταξινομεί τα στοιχεία του πίνακα σε **φθίνουσα** τάξη μεγέθους του κλειδιού και όχι του στοιχείου του ίδιου.
- **isset**(array element) – Ελέγχει αν ένα στοιχείο του πίνακα υπάρχει. Είναι γενική εντολή και μπορεί να εφαρμοστεί σε όλες τις μεταβλητές και όχι μόνο σε πίνακες
- **array_key_exists** (key, array) – Ελέγχει αν το κλειδί key υπάρχει ή όχι στο πίνακα array