PHP - Multidimensional Array

A multidimensional array is an array of arrays. In a PHP array, each element can be another array. If the array consists of values or key-value pairs with values being of singular scalar types, it is a one-dimensional array. If each element in an array is an array of one or more scalar values, it is a two-dimensional array.

A PHP array may be a two-dimensional associative array also, where each element of the outer array is key-value pair, the value being another associative array.

```
# one dimensional indexed array
$arr = [10, 20, 30, 40];

# one dimensional associative array
$arr = ["key1"=> "val1", "key2" => "val2", "key3" => "val3"];

# two dimensional indexed array
$arr = [
   [1,2,3,4],
   [10, 20, 30, 40],
   [100, 200, 300, 400]
];

# two dimensional associative array
$arr = [
   "row1" => ["key11" => "val11", "key12" => "val12", "key13" => "val13"],
   "row2" => ["key21" => "val21", "key22" => "val22", "key23" => "val23"],
   "row3" => ["key31" => "val31", "key32" => "val32", "key33" => "val33"]
];
```

Iterating over a 2D Array

Two nested loops will be needed to traverse all the elements in a 2D array. The **foreach** loop is more suitable for array traversal. A 2D array is like a tabular representation of data in rows and columns.

Example

The following example shows how you can reproduce a 2D array in a tabular form −

</> Open Compiler

```
<?php
   $tbl = [
      [1,2,3,4],
      [10, 20, 30, 40],
      [100, 200, 300, 400]
   ];
   echo ("\n");
   foreach ($tbl as $row){
      foreach ($row as $elem){
         $val = sprintf("%5d", $elem);
         echo $val;
      }
      echo "\n";
   }
?>
```

It will produce the following **output** −

```
  1   2   3   4
 10  20  30  40
100 200 300 400
```

Example

We can also employ two nested **foreach** loops to traverse a 2D associative array. Unpack each row of the outer array in row-key and row-value variables and traverse each row elements with the inner **foreach** loop.

</> Open Compiler

```
<?php
   $tbl = [
      "row1" => ["key11" => "val11", "key12" => "val12", "key13" => "val13"],
      "row2" => ["key21" => "val21", "key22" => "val22", "key23" => "val23"],
      "row3" => ["key31" => "val31", "key32" => "val32", "key33" => "val33"]
   ];

   echo ("\n");
   foreach ($tbl as $rk=>$rv){
      echo "$rk\n";
      foreach ($rv as $k=>$v){
         echo "$k => $v  ";
      }
      echo "\n";
   }
?>
```

It will produce the following **output** –

```
row1
key11 => val11  key12 => val12  key13 => val13
row2
key21 => val21  key22 => val22  key23 => val23
row3
key31 => val31  key32 => val32  key33 => val33
```

Accessing the Elements in a 2D Array

The $arr[$key] syntax of accessing and modifying an element in the array can be extended to a 2D array too. For a 2D indexed array, the jth element in the ith row can be fetched and assigned by using the expression "**$arr[$i][$j]**".

Example

</> Open Compiler

```
<?php
   $tbl = [[1,2,3,4], [10, 20, 30, 40], [100, 200, 300, 400]];

   # prints number in index 2 of the row 2
   print ("Value at [2], [2] :" . $tbl[2][2]);
?>
```

It will produce the following **output** –

```
Value at [2], [2] :300
```

Similarly, the value at ith row and jth column may be set to another value.

```
$tbl[2][2] = 250;
```

Example

If it is a 2D associative array, we need to use the row key and key-value variables of the desired column to access or modify its value.

Open Compiler

```
<?php
   $tbl = [
   "row1" => ["key11" => "val11", "key12" => "val12", "key13" => "val13"],
   "row2" => ["key21" => "val21", "key22" => "val22", "key23" => "val23"],
   "row3" => ["key31" => "val31", "key32" => "val32", "key33" => "val33"]
   ];

   print "value at row2 - key22 is " . $tbl["row2"]["key22"];
?>
```

It will produce the following **output** −

```
value at row2 - key22 is val22
```

Multi-dimensional Array

In the above example, we had an array in which the associated value of each key was another collection of key-value pairs, and we call it as a 2D array. The concept can be extended to any number of levels. For example, if each element in the inner array associates its key to another array, it becomes a three-dimensional array.

Here is an **example** of a three-dimensional array −

```
$arr3D = [
   [
      [1, 0, 9],
      [0, 5, 6],
      [1, 0, 3]
```

```
    ],
    [
      [0, 4, 6],
      [0, 0, 1],
      [1, 2, 7]
    ],
];
```

Example

To traverse such a 3D array, we need three nested **foreach** loops, as shown below −

Open Compiler

```php
<?php
   $arr3D = [
      [[1, 0, 9],[0, 5, 6],[1, 0, 3]],
      [[0, 4, 6],[0, 0, 1],[1, 2, 7]],
   ];

   foreach ($arr3D as $arr) {
      foreach ($arr as $row) {
         foreach ($row as $element) {
            echo "$element ";
         }
         echo "\n";
      }
      echo "\n";
   }
?>
```

It will produce the following **output** −

```
1 0 9
0 5 6
1 0 3

0 4 6
0 0 1
1 2 7
```

However, it is entirely possible to declare an array extending upto any number of dimensions. For that we need to have a generalized solution to traverse an array of any dimensions.

Recurve Traversal of Multidimensional Array

The following code shows a recursive function that calls itself if the value of a certain key is another array. If we pass any array as an argument to this function, it will be traversed, showing all the k-v pairs in it.

```
function showarray($arr) {
   foreach ($arr as $k=>$v) {
      if (is_array($v)) {
         showarray($v);
      } else {
         echo "$k => $v  ";
      }
   }
   echo "\n";
}
```

Example

Let us pass the above 3D array **$arr3D** to it and see the result –

Open Compiler

```
<?php
   $arr3D = [
      [[1, 0, 9],[0, 5, 6],[1, 0, 3]],
      [[0, 4, 6],[0, 0, 1],[1, 2, 7]],
   ];

   function showarray($arr){
      foreach ($arr as $k=>$v){
         if (is_array($v)){
            showarray($v);
         } else {
            echo "$k => $v  ";
         }
      }
      echo "\n";
   }
   showarray($arr3D);
?>
```

It will produce the following **output** –

```
0 => 1  1 => 0  2 => 9
0 => 0  1 => 5  2 => 6
0 => 1  1 => 0  2 => 3
0 => 0  1 => 4  2 => 6
0 => 0  1 => 0  2 => 1
0 => 1  1 => 2  2 => 7
```

This recursive function can be used with any type of array, whether indexed or associative, and of any dimension.

Example

Let us use a 2D associative array as argument to showarray() function –

</> Open Compiler

```
<?php
   $tbl = [
      "row1" => ["key11" => "val11", "key12" => "val12", "key13" => "val13"],
      "row2" => ["key21" => "val21", "key22" => "val22", "key23" => "val23"],
      "row3" => ["key31" => "val31", "key32" => "val32", "key33" => "val33"]
   ];

   function showarray($arr){
      foreach ($arr as $k=>$v){
         if (is_array($v)){
            showarray($v);
         } else {
            echo "$k => $v  ";
         }
      }
      echo "\n";
   }
   showarray($tbl);
?>
```

It will produce the following **output** −

```
key11 => val11  key12 => val12  key13 => val13
key21 => val21  key22 => val22  key23 => val23
key31 => val31  key32 => val32  key33 => val33
```