Arrays

Arrays

- array_map — Applies the callback to the elements of the given arrays
- array_merge_recursive — Merge one or more arrays recursively
- array_merge — Merge one or more arrays
- array_multisort — Sort multiple or multi-dimensional arrays
- array_pad — Pad array to the specified length with a value
- array_pop — Pop the element off the end of array
- array_product — Calculate the product of values in an array
- array_push — Push one or more elements onto the end of array
- array_rand — Pick one or more random keys out of an array
- array_reduce — Iteratively reduce the array to a single value using a callback function
- array_replace_recursive — Replaces elements from passed arrays into the first array recursively
- array_replace — Replaces elements from passed arrays into the first array
- array_reverse — Return an array with elements in reverse order
- array_search — Searches the array for a given value and returns the first corresponding key if successful
- array_shift — Shift an element off the beginning of array
- array_slice — Extract a slice of the array
- array_splice — Remove a portion of the array and replace it with something else
- array_sum — Calculate the sum of values in an array
- array_udiff_assoc — Computes the difference of arrays with additional index check, compares data by a callback function
- array_udiff_uassoc — Computes the difference of arrays with additional index check, compares data and indexes by a callback function
- array_udiff — Computes the difference of arrays by using a callback function for data comparison
- array_uintersect_assoc — Computes the intersection of arrays with additional index check, compares data by a callback function
- array_uintersect_uassoc — Computes the intersection of arrays with additional index check, compares data and indexes by separate callback functions
- array_uintersect — Computes the intersection of arrays, compares data by a callback function
- array_unique — Removes duplicate values from an array
- array_unshift — Prepend one or more elements to the beginning of an array
- array_values — Return all the values of an array
- array_walk_recursive — Apply a user function recursively to every member of an array
- array_walk — Apply a user supplied function to every member of an array
- array — Create an array
- arsort — Sort an array in descending order and maintain index association
- asort — Sort an array in ascending order and maintain index association
- compact — Create array containing variables and their values
- count — Counts all elements in an array or in a Countable object
- current — Return the current element in an array

- each — Return the current key and value pair from an array and advance the array cursor
- end — Set the internal pointer of an array to its last element
- extract — Import variables into the current symbol table from an array
- in_array — Checks if a value exists in an array
- key_exists — Alias of array_key_exists
- key — Fetch a key from an array
- krsort — Sort an array by key in descending order
- ksort — Sort an array by key in ascending order
- list — Assign variables as if they were an array
- natcasesort — Sort an array using a case insensitive "natural order" algorithm
- natsort — Sort an array using a "natural order" algorithm
- next — Advance the internal pointer of an array
- pos — Alias of current
- prev — Rewind the internal array pointer
- range — Create an array containing a range of elements
- reset — Set the internal pointer of an array to its first element
- rsort — Sort an array in descending order
- shuffle — Shuffle an array
- sizeof — Alias of count
- sort — Sort an array in ascending order
- uasort — Sort an array with a user-defined comparison function and maintain index association
- uksort — Sort an array by keys using a user-defined comparison function
- usort — Sort an array by values using a user-defined comparison function

https://www.php.net/manual/en/book.array.php

ARRAYS MULTIDIMENSIONAIS EM PHP

CURSO DE PHP

por Cláudio Rogério Carvalho Filho

- ÍNDICE
- ARTIGO
- VIDEOAULA

Nesta aula estudaremos as estruturas Arrays que possuem mais de uma dimensão.

ARRAYS MULTIDIMENSIONAIS

Array Multidimensional é uma estrutura que tem vinculado um outro Array. Podemos construir estruturas com quantas dimensões forem necessárias, porém, na maior parte das vezes trabalharemos com estruturas que tenham 2 dimensões, ou seja, estruturas que armazenam informações tabulares e que também são chamadas de Tabelas, ou então, Planilhas.

Estruturas multidimensionais em PHP também são representadas como Matrizes onde temos: Linhas * Colunas.

```
$jogo = array
(
    array(1, "Zé", 11),
    array(2, "José", 4),
    array(3, "Zéca", 22)
);
```

A estrutura do Array acima pode também ser representada seguinte maneira:

ID	NOME	PONTOS
1	Zé	11
2	José	4
3	Zéca	22

```
$jogo = array

(
    array(1, "Zé", 11),
    array(2, "José", 4),
    array(3, "Zéca", 22)
);

for ($linha=0; $linha<3; $linha++) {
    for ($coluna=0; $coluna<3; $coluna++) {
        echo $jogo[$linha][$coluna]."  ";
    }
    echo "<br/> \n";
}
```

ARRAYS MULTIDIMENSIONAIS CONTENDO ESTRUTURAS DE DADOS

É bastante comum encontrarmos código que possuem Arrays associados a chaves e estes, estão representando estruturas de informações que por exemplo, serão enviadas a um componente. A seguir, temos um código demonstrativo de como essas estruturas aparecerão.

```
$x = [
        "array1" => [
            "aaa"=>100,
            "bbb"=>200,
            "ccc"=>300
        ],
        "array2" => [
            "ddd"=>500,
            "eee"=>600,
            "fff"=>700
        ]
    ];

echo $x["array"]["bbb"];
```

EXEMPLO FEITO EM AULA

```
#   | ID | NOME | PONTOS |
#   | 1  | Zé   | 11     |
#   | 2  | José | 4      |
#   | 3  | Zéca | 22     |

//$jogo = array(
//    array("ID"=>1, "NOME"=>"Zé",   "PONTOS"=>11),
//    array("ID"=>2, "NOME"=>"José", "PONTOS"=>4),
//    array("ID"=>3, "NOME"=>"Zéca", "PONTOS"=>22)
//);
$jogo = [
            ["ID"=>1, "NOME"=>"Zé",   "PONTOS"=>11],
            ["ID"=>2, "NOME"=>"José", "PONTOS"=>4],
            ["ID"=>3, "NOME"=>"Zéca", "PONTOS"=>22]
        ];
print_r($jogo);

echo $jogo[1]["NOME"];
```

https://excript.com/php/array-multidimensional-php.html

PHP Multidimensional Arrays

In the previous pages, we have described arrays that are a single list of key/value pairs.

However, sometimes you want to store values with more than one key. For this, we have multidimensional arrays.

PHP - Multidimensional Arrays

A multidimensional array is an array containing one or more arrays.

PHP supports multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.

The dimension of an array indicates the number of indices you need to select an element.

- For a two-dimensional array you need two indices to select an element
- For a three-dimensional array you need three indices to select an element

PHP - Two-dimensional Arrays

A two-dimensional array is an array of arrays (a three-dimensional array is an array of arrays of arrays).

First, take a look at the following table:

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

We can store the data from the table above in a two-dimensional array, like this:

```
$cars = array (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
);
```

Now the two-dimensional $cars array contains four arrays, and it has two indices: row and column.

To get access to the elements of the $cars array we must point to the two indices (row and column):

Example

```
<?php
echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>
```

We can also put a `for` loop inside another `for` loop to get the elements of the $cars array (we still have to point to the two indices):

Example

```
<?php
for ($row = 0; $row < 4; $row++) {
  echo "<p><b>Row number $row</b></p>";
  echo "<ul>";
  for ($col = 0; $col < 3; $col++) {
    echo "<li>".$cars[$row][$col]."</li>";
  }
  echo "</ul>";
}
?>
```

Complete PHP Array Reference

For a complete reference of all array functions, go to our complete PHP Array Reference.

The reference contains a brief description, and examples of use, for each function!

https://www.w3schools.com/php/php_arrays_multidimensional.asp

```
<?php

$Array = array(
    array("Conta"=>"FRANCIELE OLIVEIRA", "CPF"=>"","Telefone Res."=>'(00) 0000-
0000'),
    array("Conta"=>"BEATRIX BEHN", "CPF" => "","Telefone Res."=>'(00) 0000-
0000')
);

foreach ($Array as $row)
{
    foreach($row as $i => $a)
    {
        echo '<div>'. $i." ".$a .'</div>';
    }
}
```

Como varrer array multidimensional com php

janeiro 28, 2014by vinicius7 Comentários

Mais uma dica, para quem está **iniciando no php**, dando continuidade ao tutorial anterior. Vamos entender como varrer um array multidimensional com php

É muito mais simples do que em outras linguagens. Do qual você teria que colocar dois for e ficar contando… No php você tem a função **foreach**, que varre o array, e faz a iteração automaticamente.

Rodando o array

supondo um retorno de banco que possua vários **arrays** um dentro de outro, para acessarmos todos os itens desse array faríamos o seguinte:

```
<?php
$arr_result = array(
                    'maçã',
                    'pêra',
                    'uva',
                    'outros' => array(
                            'jaca',
                            'melão',
                            'melância'
                            )
                   );
foreach($arr_result as $data)
{
     if(is_array($data))
     {
          foreach($data as $other_data)
          {
               echo $other_data, '<br/>';
          }
     }
     else
     {
          echo $data, '<br/>';
     }
}
```

Explicando o código

As primeiras linhas são a definição de um **array**, para podermos fazer as iterações.

A linha do **foreach** os parâmetros são o array, e **$data** é a variável que irá receber o valor da iteração do momento.

A função **is_array** verifica se o parâmetro passado é ou não um **array**.

A função **echo** ele exibe na tela o valor contido na variável.Nos próximos ~~capítulos dessa novela~~ posts, explico o porque de usar a virgula, e não a concatenação. De antemão lhe aconselho a usar dessa forma.

Função recursiva

Esse código seria muito eficiente para um array que sabemos a definição do tamanho, e para uma pequena varredura.

Porem para um array enorme com muitas dimensões seria um código imenso, por isso podemos utilizar da recursividade.

```
function recursive_show_array($arr)
{
     foreach($arr as $value)
     {
          if(is_array($value))
          {
               recursive_show_array($value);
          }
          else
          {
               echo $value;
          }
     }
}
```

Dessa forma se ele for array, ele irá chamar novamente a função e fazer o loop do “sub-array”, dessa forma todos os níveis serão atingidos.

Você pode conferir as funções na documentação do php:

http://php.net/is_array
http://php.net/foreach
http://php.net/echo

https://viniciusmuniz.com/pt/varrer-array-multidimensional-php/

Multidimensional arrays in PHP

- Difficulty Level : Basic
- Last Updated : 31 Jul, 2021

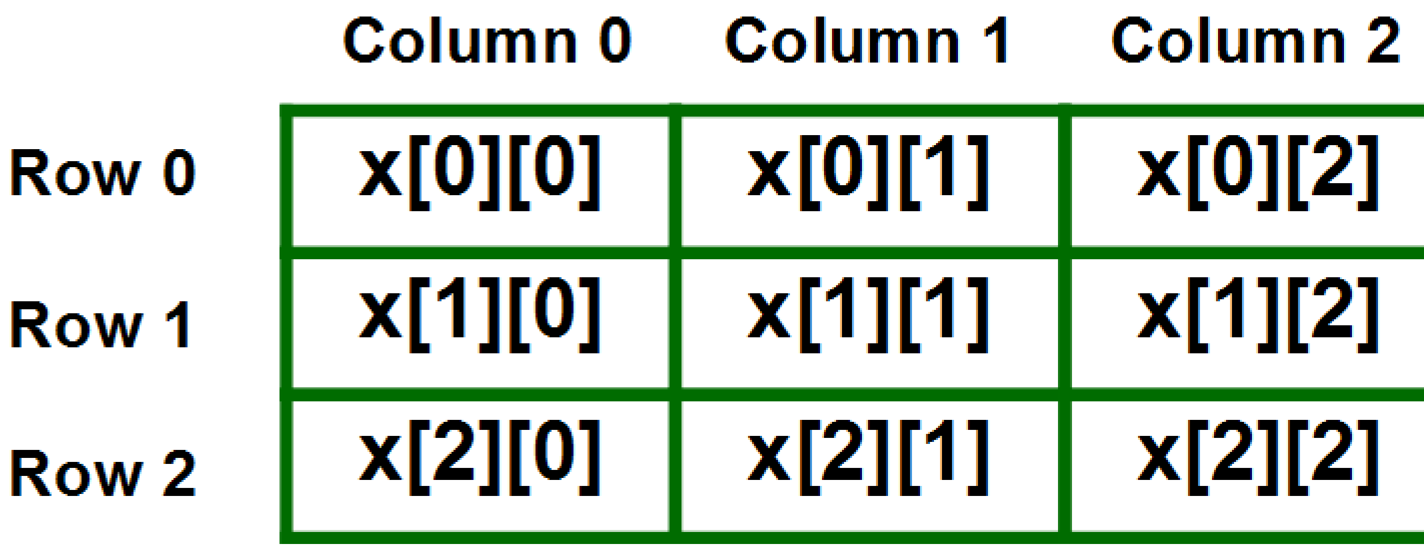
Multi-dimensional arrays are such type of arrays which stores an another array at each index instead of single element. In other words, define multi-dimensional arrays as array of arrays. As the name suggests, every element in this array can be an array and they can also hold other sub-arrays within. Arrays or sub-arrays in multidimensional arrays can be accessed using multiple dimensions.

Dimensions: Dimensions of multidimensional array indicates the number of indices needed to select an element. For a two dimensional array two indices to select an element.

Two dimensional array: It is the simplest form of a multidimensional array. It can be created using nested array. These type of arrays can be used to store any type of elements, but the index is always a number. By default, the index starts with zero.

Syntax:

```
array (
    array (elements...),
    array (elements...),
    ...
)
```



Example:

```
<?php

// PHP program to create
// multidimensional array

// Creating multidimensional
// array
$myarray = array(

    // Default key for each will
    // start from 0
    array("Ankit", "Ram", "Shyam"),
    array("Unnao", "Trichy", "Kanpur")
);

// Display the array information
print_r($myarray);
?>
```

Output:

```
Array
(
    [0] => Array
        (
            [0] => Ankit
            [1] => Ram
            [2] => Shyam
        )

    [1] => Array
        (
            [0] => Unnao
            [1] => Trichy
            [2] => Kanpur
        )

)
```

Two dimensional associative array: Al associative array is similar to indexed array but instead of linear storage (indexed storage), every value can be assigned with a user-defined key of string type.

Example:

```
<?php

// PHP program to creating two
// dimensional associative array
$marks = array(

    // Ankit will act as key
    "Ankit" => array(

        // Subject and marks are
        // the key value pair
        "C" => 95,
        "DCO" => 85,
        "FOL" => 74,
    ),

    // Ram will act as key
    "Ram" => array(

        // Subject and marks are
        // the key value pair
        "C" => 78,
        "DCO" => 98,
        "FOL" => 46,
    ),

    // Anoop will act as key
    "Anoop" => array(

        // Subject and marks are
        // the key value pair
        "C" => 88,
        "DCO" => 46,
        "FOL" => 99,
    ),
);

echo "Display Marks: \n";

print_r($marks);
?>
```

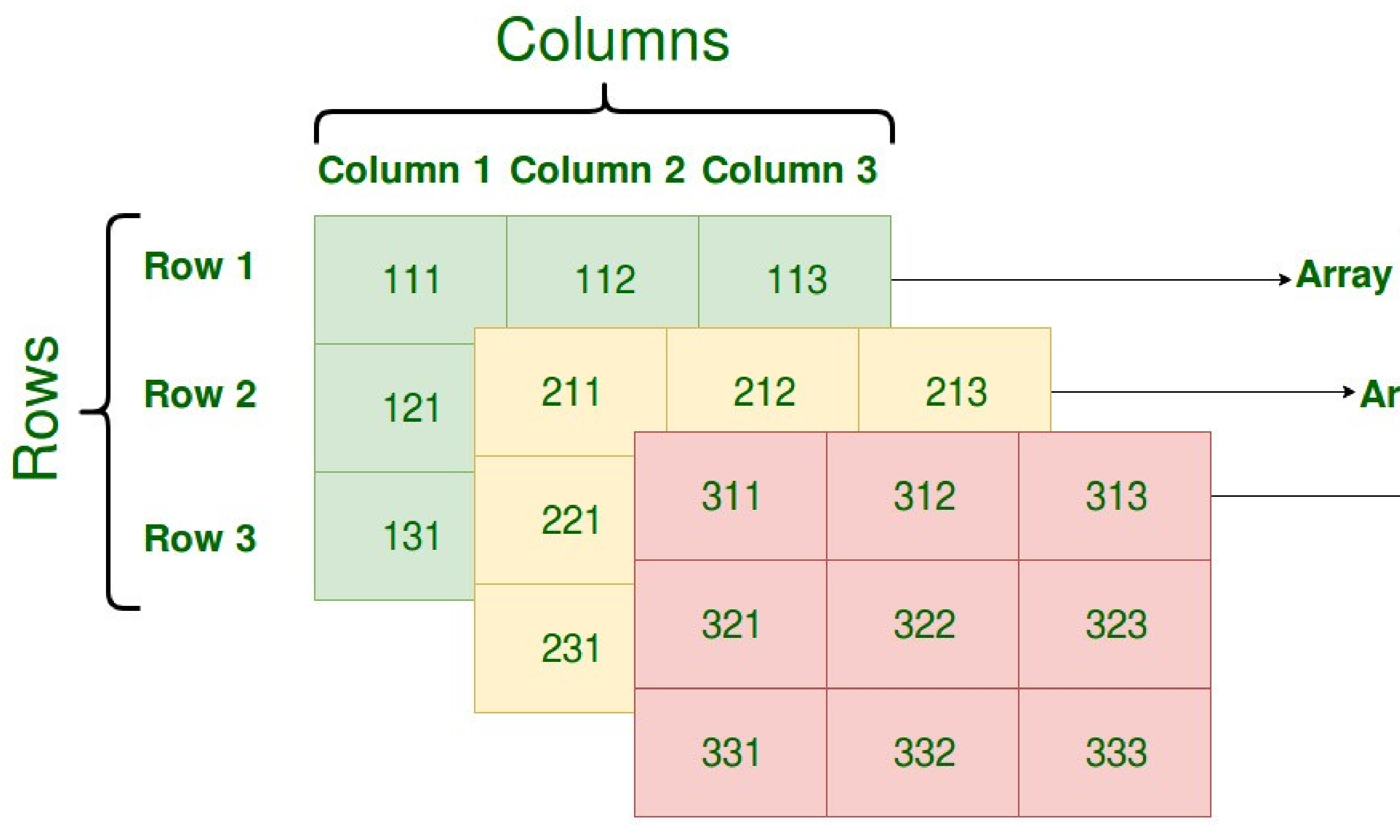
Output:

```
Display Marks:
Array
(
    [Ankit] => Array
        (
            [C] => 95
            [DCO] => 85
            [FOL] => 74
        )

    [Ram] => Array
        (
            [C] => 78
            [DCO] => 98
            [FOL] => 46
        )

    [Anoop] => Array
        (
            [C] => 88
            [DCO] => 46
            [FOL] => 99
        )

)
```

Three Dimensional Array: It is the form of multidimensional array. Initialization in Three-Dimensional array is same as that of Two-dimensional arrays. The difference is as the number of dimension increases so the number of nested braces will also increase.

Syntax:

```
array (
    array (
        array (elements...),
        array (elements...),
        ...
    ),
    array (
        array (elements...),
        array (elements...),
        ...
    ),
    ...
)
```



Example:

```
<?php

// PHP program to creating three
// dimensional array

// Create three nested array
$myarray = array(
    array(
        array(1, 2),
        array(3, 4),
    ),
    array(
        array(5, 6),
        array(7, 8),
    ),
);

// Display the array information
print_r($myarray);
?>
```

Output:

```
Array
(
    [0] => Array
        (
            [0] => Array
                (
                    [0] => 1
                    [1] => 2
                )

            [1] => Array
                (
                    [0] => 3
                    [1] => 4
                )

        )

    [1] => Array
        (
            [0] => Array
                (
                    [0] => 5
                    [1] => 6
                )

            [1] => Array
                (
                    [0] => 7
                    [1] => 8
                )

        )

)
```

Accessing multidimensional array elements: There are mainly two ways to access multidimensional array elements in PHP.

- Elements can be accessed using dimensions as array_name[‘first dimension’][‘second dimension’].
- Elements can be accessed using for loop.
- Elements can be accessed using for each loop.

Example:

```
<?php

// PHP code to create
// multidimensional array

// Creating multidimensional
// associative array
$marks = array(

    // Ankit will act as key
    "Ankit" => array(

        // Subject and marks are
        // the key value pair
        "C" => 95,
        "DCO" => 85,
        "FOL" => 74,
    ),

    // Ram will act as key
    "Ram" => array(

        // Subject and marks are
        // the key value pair
        "C" => 78,
        "DCO" => 98,
        "FOL" => 46,
    ),

    // Anoop will act as key
    "Anoop" => array(

        // Subject and marks are
        // the key value pair
        "C" => 88,
        "DCO" => 46,
        "FOL" => 99,
    ),
);

// Accessing the array element
// using dimensions

// It will display the marks of
// Ankit in C subject
echo $marks['Ankit']['C'] . "\n";

// Accessing array elements using for each loop
foreach($marks as $mark) {
    echo $mark['C']. " ".$mark['DCO']." ".$mark['FOL']."\n";
}

?>
```

Output:

```
95
95 85 74
78 98 46
88 46 99
```

PHP is a server-side scripting language designed specifically for web development. You can learn PHP from the ground up by following this PHP Tutorial and PHP Examples.

https://www.geeksforgeeks.org/multidimensional-arrays-in-php/