4...

Arrays in PHP

Learning Objectives...

- To understand the Indexed versus Associative arrays.
- To learn about Identifying Elements of an Array.
- To study the Storing data in arrays and Multidimensional Arrays.
- To learn the concept of Traversing Arrays and Sorting.

4.1 ARRAYS IN PHP

- Arrays are data structures used to store multiple values in a single variable. PHP supports various types of arrays: Indexed and Associative arrays.

4.1.1 Indexed Versus Associative Arrays

- The fundamental difference between indexed arrays and associative arrays lies in how their elements are accessed and organized.

1. Indexed Arrays:

- An indexed array is an array with a numeric key. It is basically an array where in each of the keys is associated with its own specific value.

Key Type:

- Keys are numeric and start from 0 by default. Sequential integers as keys, typically starting from 0.

Access Method:

- Elements are accessed by their numerical position or index within the array (e.g., array[0], array[1]).

For Example,

```
$colors = array("Red", "Green", "Blue");
echo $colors[1];  // Output: Green
```

2. **Associative Arrays:**

- An associative array is stored in the form of a key-value pair. This type of array is where the key is stored in the numeric or string format.

For Example,

```
$student = array("name" => "ali", "rollno" =>225);
echo $student["name"];  // Output: ali
```

4.2 IDENTIFYING ELEMENTS OF AN ARRAY

- Identifying elements within a PHP array involves understanding how to access and inspect their values and structure.

Accessing Individual Elements:

- **Indexed Arrays:** Elements are accessed using their numerical index (starting from 0).

For Example,

```
$students = ["Zaid", "Faizan", "ammar"];
echo $students[0];  // Zaid
```

- **Associative Arrays:** Elements are accessed using their string key.

For Example,

```
$marks = ["Math" => 90, "Science" => 85];
echo $marks["Math"]; // 90
```

- To check if an element exists:

For Example,

```
if (array_key_exists("Math", $marks))
{ echo "Math marks found.";
}
```

4.3 STORING DATA IN ARRAYS

- In PHP, arrays are special variables capable of holding more than one value in a single variable. This eliminates the need to declare multiple individual variables for related data.
- There are three main types of arrays in PHP for storing data:
  1. Indexed Arrays (Numeric Arrays):
  2. Associative Arrays:
  3. Multidimensional Arrays:
- You can add values to an array dynamically:

For Example,

```
$fruits[] = "Apple";
$fruits[] = "Banana";
$profile["email"] = "test@example.com";
$profile["city"] = "Mumbai";
```

Functions to create/store arrays:

- array()
- [](short syntax)
- array_push(), array_unshift()

4.4 MULTIDIMENSIONAL ARRAYS

- A multidimensional array is an array in which each element may also be an array, and each element in the sub-array can also be an array or have another array within it.
- A multidimensional array is an array of arrays. In a PHP array, each element can be another array.

Program 4.1: Program of multidimensional array.

```
<?php
    // Define a multidimensional array
    $friends = array(
    array(
        "name" => "Mohsin",
        "city" => "pune",
        ),
    array(
        "name" => "Ali",
        "city"  => "pune",
        ),
    array(
        "name" => "Umar",
        "city"  => "Islampur",
        )
    );
    // Access nested value
    echo "Ali is from " . $friends[1]["city"] . ".";
?>
```

Output:

Ali is from pune.

4.5 EXTRACTING MULTIPLE VALUES

- In PHP, extracting multiple values can be achieved through various methods, depending on the source and desired outcome.

4.5.1 From Arrays

- **list():** This construct allows assigning array values directly to individual variables. It works best with numerically indexed arrays or when you know the order of elements in an associative array.

For Example,

```
$person = ['Fred', 35, 'Betty'];
list($name, $age, $wife) = $person;
// $name is 'Fred', $age is 35, $wife is 'Betty'
```

- **extract():** This function imports variables from an array into the current symbol table. It's particularly useful for associative arrays, where array keys become variable names. Flags can be used to control behavior in case of variable name collisions.

For Example,

```
$data = ['name' => 'Alice', 'age' => 28];
extract($data);
// $name is 'Alice', $age is 28
```

- **array_slice():** To extract a consecutive subset of values from an array, array_slice() returns a new array containing the specified range.

For Example,

```
$numbers = [1, 2, 3, 4, 5];
$subset = array_slice($numbers, 1, 3); // Extracts 3 elements starting from
index 1
// $subset is [2, 3, 4]
```

- **array_column():** When dealing with an array of arrays or objects, array_column() extracts the values of a specific column (or key) into a new indexed array.

For Example,

```
$records = [
    ['id' => 1, 'name' => 'John'],
    ['id' => 2, 'name' => 'Jane']
];
$names = array_column($records, 'name');
// $names is ['John', 'Jane']
```

4.5.2 From Functions

- **Returning an Array:** The most common and straightforward way to return multiple values from a function is to encapsulate them within an array. This array can then be processed using list(), extract(), or direct array access.

For Example,

```
function getUserDetails() {
    return ['name' => 'Charlie', 'email' => 'charlie@example.com'];
}
$details = getUserDetails();
echo $details['name'];
```

4.5.3 From Strings

- **explode():** To extract values from a delimited string, explode() splits the string into an array based on a specified delimiter.

For example,

```
$csv_string = "apple,banana,orange";
$fruits = explode(',', $csv_string);
// $fruits is ['apple', 'banana', 'orange']
```

4.6 CONVERTING BETWEEN ARRAYS AND VARIABLES

- PHP offers two primary functions for converting between arrays and variables: extract() and compact().

4.6.1 Converting an Array to Variables using extract()

- The extract() function takes an associative array as input and creates individual variables from its key-value pairs. The array keys become the variable names, and their corresponding values are assigned to these new variables.

Program 4.2:

```
<?php
   $student = array(
      'name' => 'Aliza',
      'rollno' => 30,
      'city' => 'pune'
   );
   extract($student);
      echo "Name: " . $name . "\n";
      echo "rollno: " . $rollno . "\n";
      echo "City: " . $city . "\n";
?>
```

Output:

```
Name: Aliza
rollno: 30
City: pune
```

4.6.2 Converting Variables to an Array using compact()

- The compact() function takes a list of variable names (as strings) and creates an associative array where the variable names are the keys and their current values are the array's values.

Program 4.3:

```
<?php
    $name = 'suhel';
    $occupation = 'Asst.professor';
    $company = 'AISC';
    $details = compact('name', 'occupation', 'company');
    print_r($details);
?>
```

Output:

```
Array
(
    [name] => suhel
    [occupation] => Asst.professor
    [company] => AISC
)
```

4.7 TRAVERSING ARRAYS

- Traversing arrays in PHP involves iterating through each element to access or manipulate its value. Several methods are available for this purpose, each suited for different scenarios. You can loop through arrays using: foreach and for loop

1. foreach loop:

Program 4.4:

```
<?php
    $colors = ["Red", "Green", "Blue"];
    foreach ($colors as $color)
    {
        echo $color . "<br>";
    }
    $marks = ["Math" => 90, "English" => 85];
    foreach ($marks as $subject => $mark)
    {
        echo "$subject: $mark <br>";
    }
```

Output:

```
Red
Green
Blue
Math: 90
English: 85
```

2. for loop (for indexed arrays):

For Example,

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

4.8 SORTING

- Sorting is the process of arranging a collection of items or data elements in a particular order, usually based on some predefined criteria. It is a fundamental operation in computer science and is used extensively in various algorithms and applications.
- The purpose of sorting is to bring organization and structure to a set of data so that it can be easily searched, accessed, or presented in a meaningful way. By arranging the data in a specific order, sorting allows for efficient searching, comparison, and retrieval operations.
- PHP provides many array sorting functions which are discussed below:

4.8.1 Indexed Arrays

1. sort($colors); // Ascending
2. rsort($colors); //Descending

1. sort():

- The sort() function sorts an array in ascending order based on the values. It reorders the elements of the array and modifies the original array.

Program 4.5: Program to sorts the elements of the $numbers array in ascending numerical order:

```
<?php
  $numbers = array(4, 2, 1,5, 3);
  sort($numbers);
  print_r($numbers);
?>
```

Output

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
    [3] => 4
    [4] => 5
)
```

2. The rsort() function:

- The rsort() function is similar to sort(), but it sorts the array in descending order.

Program 4.6:

```
<?php
    $numbers = array(4, 2, 1,5, 3);
    rsort($numbers);
    print_r($numbers);
?>
```

Output

```
Array
(
    [0] => 5
    [1] => 4
    [2] => 3
    [3] => 2
    [4] => 1
)
```

4.8.2 Associative Arrays

1. asort($marks); // Sort by value (ascending)
2. ksort($marks); // Sort by key (ascending)
3. arsort($marks); // Sort by value (descending)
4. krsort($marks); // Sort by key (descending)

- These functions help organize data alphabetically, numerically, or by custom logic.

1. The asort() function

- The asort() function sorts an array in ascending order based on the values while maintaining the association between keys and values.

Program 4.7:

```
<?php
    $fruits = array("apple" => 3, "banana" => 2, "cherry" => 1);
    asort($fruits);
    print_r($fruits);
?>
```

Output

```
Array
(
    [cherry] => 1
    [banana] => 2
    [apple] => 3
)
```

2. ksort() function:

- The ksort() function sorts an array in ascending order based on the keys while maintaining the association between keys and values.

Program 4.8:

```
<?php
   $age = array("Peter"=>"60", "Ben"=>"45", "Joe"=>"36");
   ksort($age);
   print_r($age);
?>
```

Output

```
Array
(
    [Ben] => 45
    [Joe] => 36
    [Peter] => 60
)
```

3. arsort()

- The arsort() function is similar to asort(), but it sorts the array in descending order while maintaining the association between keys and values.

Program 4.9:

```
<?php
   $age = array("Peter"=>"60", "Ben"=>"36", "Joe"=>"45");
   arsort($age);
   print_r($age);
?>
```

Output

```
Array
(
    [Peter] => 60
    [Joe] => 45
    [Ben] => 36
)
```

4. The krsort() function:

- The krsort() function is similar to ksort(), but it sorts the array in descending order based on the keys while maintaining the association between keys and values.

Program 4.10:

```
<?php
   $fruits = array("banana" => 2, "apple" => 3, "cherry" => 1);
   krsort($fruits);
```

```
print_r($fruits);  // Output: Array ( [cherry] => 1 [banana] => 2 [apple] =>
3 )
?>
```

Output

```
Array
(
    [cherry] => 1
    [banana] => 2
    [apple] => 3
)
```

Sample programs

Program 4.11: Write a PHP program to store student names with their marks in an associative array and display each student's name with marks.

```
<?php
$marks = [
    "Ammar" => 85,
    "Ali" => 90,
    "Aliza" => 78
];
echo "Student Marks:<br>";
foreach ($marks as $name => $score) {
    echo "$name : $score <br>";
}
?>
```

Output:

```
Student Marks:
Ammar : 85
Ali : 90
Aliza : 78 <br>
```

Program 4.12: Write a PHP program to store 3 students' names with marks in 2 subjects using a multidimensional array and display them in table format.

```
<?php
    $students = [
        ["Name" => "Ali", "Math" => 85, "Science" => 90],
        ["Name" => "aiman", "Math" => 92, "Science" => 88],
        ["Name" => "aiza", "Math" => 78, "Science" => 84]
    ];
    echo "<table border='1' cellpadding='5'>";
```

```
    echo "<tr><th>Name</th><th>Math</th><th>Science</th></tr>";
    foreach ($students as $student)
    {
        echo "<tr>";
        echo "<td>" . $student["Name"] . "</td>";
        echo "<td>" . $student["Math"] . "</td>";
        echo "<td>" . $student["Science"] . "</td>";
        echo "</tr>";
    }
    echo "</table>";
?>
```

Output:

Name	Math	Science
Ali	85	90
aiman	92	88
aiza	78	84

Check Your Understanding

1. Which of the following defines an indexed array in PHP?
   (a) `$arr = array("one" => 1, "two" => 2);`
   (b) `$arr = array(1, 2, 3);`
   (c) `$arr = [];`
   (d) `$arr = "index";`
2. Which function is used to get all the keys of an array?
   (a) `array_values()` (b) `array_keys()`
   (c) `keys()` (d) `get_keys()`
3. Which of the following loops is best suited for traversing an array?
   (a) `For` (b) `foreach`
   (b) `While` (d) `do...while`
4. What does extract() function do?
   (a) Extracts part of a string
   (b) Converts array values to uppercase
   (c) Imports variables from an array into the current symbol table
   (d) Removes duplicates from an array
5. Which function merges two or more arrays in PHP?
   (a) `array_combine()` (b) `array_merge()`
   (b) `array_sum()` (d) `array_join()`

Answers

1. (b)	2. (b)	3. (b)	4. (c)	5. (b)

Practice Questions

Q.I Answer the following questions in short:

1. What is the key difference between an indexed array and an associative array?
2. How can you access the third element of an indexed array named `$colors`?
3. Write the syntax to create an associative array for a student with name, age, and marks.
4. How do you declare a 2D array in PHP?
5. What function is used to convert array keys into individual variables?
6. What is the use of `compact()` in PHP?
7. Which loop is most commonly used for traversing arrays?
8. How do you sort an array by value in ascending order?
9. What is the output of `count($arr) if $arr = [1, 2, 3, 4];`?
10. Write a short code to merge two arrays: `$a = [1, 2];` and `$b = [3, 4];`.

Q.II Answer the following questions in detail:

1. Explain the difference between indexed and associative arrays in PHP with examples.
2. How do you identify and access specific elements of an array? Give examples using both indexed and associative arrays.
3. Write a PHP script to store and display student names and their marks using arrays.
4. What are multidimensional arrays? Explain with an example of a 2D array storing student data (name, age, marks).
5. Describe different methods to extract multiple values from an array. Include functions like `list()`, `array_values()`, and `extract()`.
6. How can arrays be converted into variables and vice versa? Explain extract() and compact() functions with examples.
7. Describe the different ways of traversing arrays using foreach, for, and while loops. Provide examples.
8. What are the various built-in functions used for sorting arrays in PHP? Explain each with examples.
9. Write a PHP program to create an associative array of products (name and price), and sort them by price.
10. Compare and contrast the use of `array_merge()`, `array_combine()`, and `array_slice()` in PHP. Provide examples.

***