UNIT – 4
ARRAYS

UNIVERSITY
OF PETROLEUM
& ENERGY STUDIES

Dr. P S V S Sridhar

Assistant Professor (SS)

Centre for Information Technology

Introduction to Array

-An array can store one or more values in a single variable name.

-Each element in the array is assigned its own ID so that it can be easily accessed.

-$array[*key*] = *value*;

$arrayname[]=value

array() language construct

Array in PHP

1) **Numeric Array**
2) **Associative Array**
3) **Multidimensional Array**

Numeric Array

- A numeric array stores each element with a numeric ID key.
- 2 ways to write a numeric array.

1. Automatically

Example: $names = array("Peter","Quagmire","Joe");

2. Manually

Example1: $names[0] = "Peter"; $names[1] = "Quagmire";

$names[2] = "Joe";

Example 2: $name[] =”Peter”; $name[] =“Quagmire”; $name[] = “joe”;

In the above case we are adding sequential elements that should have numerical counting upward from **zero.**

Associative Arrays

-An associative array, each ID key is associated with a value.

-When storing data about specific named values, a numerical array is not always the best way to do it.

-With associative arrays we can use the values as keys and assign values to them.

Example: Using an array to assign an age to a person.

```
$ages = array(”Brent"=>42, ”Andrew"=>25, "Joshua”=>16);

$ages[’Brent'] = 42;

$ages[’Andrew'] = 25;

$ages['Joshua'] = 16;
```

Eamample:

```
$f = array(0=>”apple”, 1=>”orange”, 2=>”banana”, 3=>”pear”);
```

In the above example index values are 0,1,2,3

Array creation:

- **following creates an array with numeric index**
- $icecream_menu=array(275,160,290,250);

which is equivalent to the following array creation

$icecream_menu=array(0=>275,1=>160,2=>290,3=>250);

- **Arrays can be created with other scalar values as well as mix of them**
- $icecream_menu1[-1]=275;
  $icecream_menu1[‘item 2’]=160;
  $icecream_menu1[3.4]=290;
  $icecream_menu1[false]=250;
- **An array with a mix of all values can also be created**

  $icecream_menu=array(-1=>275,'item 2'=>160,3.4=>290,false=>250);

Creating an Empty Array

- $marks=array();

Finding the Size of an Array

- **The count() function is used to find the number of elements in the array**

- $icecream_menu=array('Almond Punch'=>275,'Nutty Crunch'=>160,'Choco Dip'=>290,'Oreo Cherry'=>250);
  print "There are ".count($icecream_menu) ." flavours of icecreams available!!"

- prints the following
- There are 4 flavours of icecreams available!!

Printing an Array in a readable way

- **The print_r() ,when passed an array, prints its contents in a readable way**.
- $icecream_menu=array('Almond Punch'=>275,'Nutty Crunch'=>160,'Choco Dip'=>290,'Oreo Cherry'=>250);
- print_r($icecream_menu);

- Output of the above code will be
  Array ([Almond Punch] => 275 [Nutty Crunch] => 160 [Choco Dip] => 290 [Oreo Cherry] => 250)

Iterating Array Elements

- The easiest way to iterate through each element of an array is with foreach(). The foreach()construct executes the block of code once for each element in an array.
- Syntax
- foreach($array as [$key =>] [&] $value)

  { //block of code to be executed for each iteration }
- $array – represents the array to be iterated
- $key is optional, and when specified, it contains the currently iterated array value‟s key, which can be any scalar value, depending on the key‟s type.
- $value holds the array value, for the given $key value.
- Using the & for the $value, indicates that any change made in the $value variable while iteration will be reflected in the original array($array) also.

Modifying Array while iteration

- Arrays can be modified during iteration in 2 ways.
- Direct Array Modification
- Indirect Array Modification

Direct Array Modification

- Say for example we need to increase the price of the all icecreams by 50 .The following code helps us to update the price and display the revised price along with the old ones
- <html>
<body>
<h3 align="center">Chillers...</h3>
<table align="center">
<tr bgcolor="#D7D7D7" ><td >Icecream Flavours</td><td>Price</td></tr>
<?php
$icecream_menu=array('Almond Punch'=>275,'Nutty Crunch'=>160,'Choco Dip'=>290,'Oreo Cherry'=>250);

foreach($icecream_menu as $key=>$value)

{

print "<tr bgcolor='#F3F3F3'>";

print "<td>".$key."</td><td>".$value."</td>";

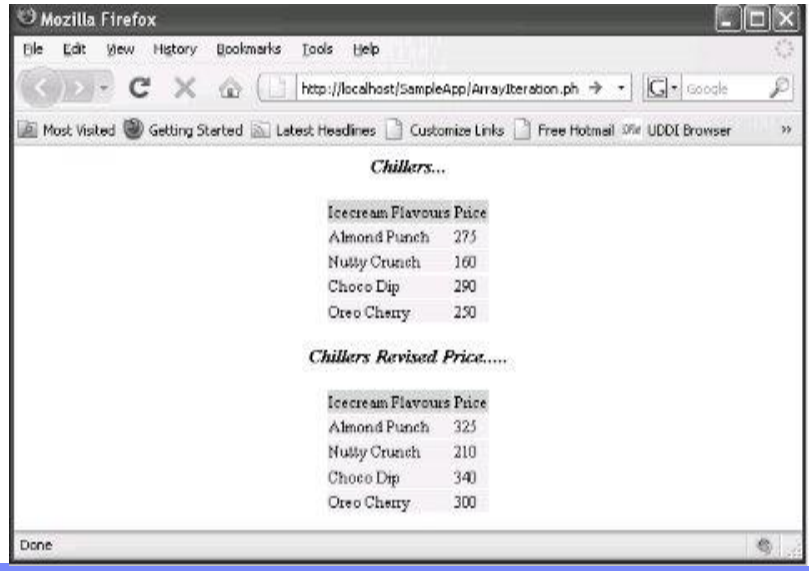
print "</tr>";

}

?>
</table>
<h3 align="center">Chillers Revised Price.....</h3>
<table align="center">
<tr bgcolor="#D7D7D7" ><td >Icecream Flavours</td><td>Price</td></tr>

Contd…

```
<?php
foreach($icecream_menu as $key=>&$value)
{
//increase the price of all icecreams by 50

$icecream_menu[$key]=$value+50;

$value=$icecream_menu[$key];

print "<tr bgcolor='#F3F3F3'>";

print "<td>".$key."</td><td>".$value."</td>";

print "</tr>";

}
?>
</table>
</body>
</html>
```



Indirect Array Modification

- foreach($array as [$key =>] [&] $value) {

  //block of code to be executed for each iteration

  }
- where “&“ can be used with $value.Any change in $value,indirectly modifies the value of the array for the current key during the iteration.
- foreach($icecream_menu as $key=>&$value)

  {

  //increase the price of all icecreams by 50

  $value=$value+50;

  print "<tr bgcolor='#F3F3F3'>";

  print "<td>".$key."</td><td>".$value."</td>";

  print "</tr>";

  }

Iterating Array with Numeric index

- Arrays with numeric index can be accessed using
- foreach() **and** for()

Ex1:

- $hobbies=array("Reading Books","Playing Golf","Watching Tennis","Dancing");
  foreach($hobbies as $hobby)
  {
  print "
 $hobby ";
  }

Ex 2:

- **for($i=0,$arraysize=count($hobbies);$i<$arraysize;$i++)**

  **print "
 $hobbies[$i]";**

Multi dimensional Arrays

- In a multidimensional array, each element in the main array can also be an array.
- And each element in the sub-array can be an array, and so on.

Ex1:

```
$families = array
("Griffin"=>array ( "Peter","Lois", "Megan"  ),
   "Quagmire"=>array  (  "Glenn"  ),
   "Brown"=>array ("Cleveland",  "Loretta",  "Junior"  )
 );
```

Ex2:

```
$icecream_menu=array('Almond
   Punch'=>array("single"=>275,"couple"=>425,"family"=>500),
   'Nutty Crunch'=>array("couple"=>375,"family"=>450),
   'Choco Dip'=>array("single"=>275,"Treat"=>425,"family"=>600),
   'Oreo Cherry'=>array("single"=>450,"family"=>525));
```

Handling Multidimensional arrays

```
$scores[“sam”][1] = 79;
$scores[“sam”][2] = 74;
$scores[“ellen”][1] = 69;
$scores[“ellen”][2] = 84;
Print_r($scores);
```

Output:

```
[sam] => array([1] =>79
               [2] =>74
              )
[ellen] => array([1] => 69
                 [2] => 84
                )
```

Handling Multidimensional arrays

Ex:

$scores[“sam”][] = 79;

$scores[“sam”][] = 74;

$scores[“ellen”][] = 69;

$scores[“ellen”][] = 84;

Here index start from 0.

Ex:

$scores = array(“sam” => array(79,74), “ellen” => array(69,84));

Multi dimensional arrays in loops

```
$scores[0][] = 79;
$scores[0][] = 74;
$scores[1][] = 69;
$scores[1][] = 84;
for($outer = 0; $outer < count($scores); $outer++)
    for($inner = 0; $inner < count($scores[$outer]); $inner++)
    {
            echo $scores[$outer][$inner];
     }
```

Splitting and Merging Arrays

array_slice function will split array

Ex1:

```
$ice_cream[“good”] = “orange”;
$ice_cream[“better”] = “vanilla”;
$ice_cream[“best”] = “rum raisin”;
$ice_cream[“bestest”] = “lime”;
$subarray = array_slice($ice_cream, 1,2);
```

- **The index value of better and best will be stored in $subarray as vanilla rum raisin**

Ex2:

```
<?php
$input = array("a", "b", "c", "d", "e");

$output = array_slice($input, 2); // returns "c", "d", and "e"
$output = array_slice($input, -2, 1); // returns "d"
$output = array_slice($input, 0, 3); // returns "a", "b", and "c“

?>
```

Splitting and Merging Arrays

array_merge will merge the arrays

Ex1:

$pudding = array(“vanilla”, “run raisin”, “orange”);

$ice_cream = array(“chocolate”,”pecan”,”strawberry”);

$desserts = array_merge($pudding, $ice_cream);

The elements of $pudding and $ice_cream will be stored in $desserts array.

Ex2:

```
<?php
$array1 = array("color" => "red", 2, 4);
$array2 = array("a", "b", "color" => "green", "shape" => "trapezoid", 4);
$result = array_merge($array1, $array2);
print_r($result);
?>
```

Output:

Array ([color] => green [0] => 2 [1] => 4 [2] => a [3] => b [shape] => trapezoid [4] => 4)

Sorting of Arrays

- **Sort - Sorts the array values (loses key)**

```
<?php

$fruits = array("lemon", "orange", "banana", "apple");
sort($fruits);
foreach ($fruits as $key => $val) {
echo "fruits[" . $key . "] = " . $val . "\n";
}

?>
```

Output:

fruits[0] = apple fruits[1] = banana fruits[2] = lemon fruits[3] = orange

Sorting of Arrays

- **Ksort - Sorts the array by key along with value**

```
<?php
$fruits = array("d"=>"lemon", "a"=>"orange", "b"=>"banana", "c"=>"apple");
ksort($fruits);
foreach ($fruits as $key => $val) {
echo "$key = $val\n";
}
?>
```

Output:

a = orange b = banana c = apple d = lemon

Sorting of Arrays

- **asort - Sorts array by value, keeping key association**

```
<?php
$fruits = array("d" => "lemon", "a" => "orange", "b" => "banana", "c" =>
"apple");
asort($fruits);
foreach ($fruits as $key => $val) {
echo "$key = $val\n";
}
?>
```

Output:

c = apple b = banana d = lemon a = orange

Array functions

- **count($ar)** - How many elements in an array
- **array_sum($ar) –** sum of array values.
- **sizeof($ar)** – Identical to count()
- **is_array($ar)** - Returns TRUE if a variable is an array
- **sort($ar)** - Sorts the array values (loses key)
- **ksort($ar)** - Sorts the array by key along with value
- **asort($ar)** - Sorts array by value, keeping key association
- **shuffle($ar)** - Shuffles the array into random order
- $a = array() – Creates an array $a with no elements
- $a = range(1,5) – Create an array between 1 and 5. i.e., $a = array(1,2,3,4,5)
- **in_array()** – Takes two arguments: an element and an array. Returns true if the element contained as a value in the array.
- **rsort(), arsort(), krsort()** - Works similar to sort(), asort() and ksort() except that they sort the array in descending order

Array functions

- **list():**
- Assign array values to variable.
- Ex: $f = array('apple','orange','banana');
- list($r,$o) = $f;
- The string apple is assigned to $r and orange is assigned to $o and no assignment of string banana.
- **reset():**
- Gives us a way to "rewind" that pointer to the beginning – it sets the pointer to the first key/value pair and then returned to stored value.
- **next():**
- which moves the current pointer ahead by one,
- **prev():**
- Which moves the pointer back by one.
- **current():**
- Returns the current element in an array
- **end()**
- **Which moves to end of the array**

Next, current, Prev

- <?php

```
$array = array('step one', 'step two', 'step three', 'step four');

// by default, the pointer is on the first element
echo current($array) . "<br />\n"; // "step one"

// skip two steps
next($array);
next($array);
echo current($array) . "<br />\n"; // "step three"

// reset pointer, start again on step one
reset($array);
echo current($array) . "<br />\n"; // "step one"

?>
```

Array functions

- **Deleting from arrays:** Just call **unset()**
- $a[0] = “wanted”;
- $a[1]=“unwanted”;
- $a[2]=“wanted again”;
- unset($a[1]);
- Deletes 1 key value. Now the above array contains two key values (0,2)

Extracting Data from Arrays

- Extract function to extract data from arrays and store it in variable.

```
$ice_cream[“good”] = “orange”;
$ice_cream[“better”] = “vanilla”;
$ice_cream[“best”] = “rum raisin”;
extract($ice_cream);
echo “\$good = $good <br>”;
echo “\$better = $better <br>”;
echo “\$best = $best <br>”;
```

Output:

$good = orange

$better = vanilla

$best = rum raisin

Arrays and Strings

explode(): The function explode converts string into array format .

We can use explode() to split a string into an array of strings

$inp = "This is a sentence with seven words";

$temp = explode(‘ ', $inp);

print_r($temp);

Output:

Array([0] => This [1] => is [2] => a [3] => sentence [4] => with [5] => seven [6] => words)

implode(): To combine array of elements in to a string

- $p = array("Hello", "World,", "I", "am", "Here!");
- $g = ***implode(" ", $p)***;
- **Output:** Hello World I am Here

Arrays and strings

- **str_replace():**

$rawstring = "Welcome Birmingham parent!

Your offspring is a pleasure to have! We believe pronoun is learning a lot.
The faculty simple adores pronoun2 and you can often hear them say \"Attah sex!\"
";

$placeholders = array('offspring', 'pronoun', 'pronoun2', 'sex');

$malevals = array('son', 'he', 'him', 'boy');

$femalevals = array('daughter', 'she', 'her', 'girl');

$malestr = str_replace(***$placeholders, $malevals***, $rawstring);

$femalestr = str_replace(***$placeholders, $femalevals***, $rawstring);

echo "Son: ". $malestr . "
";

echo "Daughter: ". $femalestr;

Output for last but statement:

Son: Welcome Birmingham parent!

Your **son** is a pleasure to have! We believe he is learning a lot.

The faculty simple adores **he**2 and you can often hear them say "Attah **boy**!"

Comparing Arrays to Each other

- You can find difference between two arrays using the **array_diff** function.

```
$ice_cream1 = array(“vanilla”, “chocolate”, “strawberry”);

$ice_cream2 = aray(“vanilla”, “chocolate”, “papaya”);

$diff = array_diff($ice_cream1, $ice_cream2);

foreach($diff as $key => $value)

{

        echo “key : $key; value: $value \n”;

}
```

Output:

Key : 2; value : strawberry

Thanks