Web Application Development (WAD)

Vth Sem – BBAITM (Unit – 2)

By: Binit Patel

Control Structure:

1) Conditional Statements

The linear, hierarchic, top down, execution of a program, is controlled by 'Control Structures' embedded within program code. PHP provides programmers with conditional statements to achieve this.

There are two types of conditional statements:

If (…else) statement – This statement is used, if a block of code must test a specific condition, if the condition tested returns a 'TRUE' then the code executes. If the condition tested returns 'FALSE' another block of code executes.

Switch statement – This statement is used to choose and execute **one** code block from among multiple code blocks for execution.

The if Statement

If statements are the most common conditional statements and they exist in most programming languages. Use the if-statement to execute different code snippests of PHP code in a program, based on what is returned when condition evaluated.

```
<?php
    $tell_joke = "yes";
     if ($tell_joke=="yes")
{
```

```
    echo "How do you spot a modern spider?<br>";
    echo "She doesn't have a web she has a website!<br>";
  }
?>
```

Output:

How do you spot a modern spider?

She doesn't have a web she has a website!

If-else

```
<?php
  $age = 15;
  $discount_age = 60;
    if ($age>=$discount_age)
      {
        echo "Congratulations! You qualify for our senior discount.";
      }
      Else
       {
        echo "Sorry, but you do not qualify for our senior discount.";
       }
?>
```

Output:

Sorry, but you do not qualify for our senior discount.

Because, the condition returns false, so, else part will get executed.

If-elseif

The **elseif** construct can be used to conduct a series of conditional checks and only execute the code following the first condition that met. The last **elseif** could be substituted with an **else** clause.

```
<?php
$color = "orange";

if ($color=="purple")
{
      echo "Purple conveys royalty, nobility & wisdom.";
 }
elseif ($color=="orange")
{
      echo "Orange conveys energy, warmth & flamboyancy.";
 }
elseif ($color=="blue")
{
      echo "Blue conveys peace, tranquility & security.";
 }
else
{
      echo "I dunno what that color conveys, sorry.";
 }
?>
```

Output:

Orange conveys energy, warmth & flamboyancy.

2) Looping Structure

A loop can be thought of as something that is wrapped around the PHP code spec, to make it happen again and again.

In PHP the following Loop statements are available:

For – loops through a block of code a specified number of times.

While – loops through a block of code as long as a specified condition is TRUE

The for loop

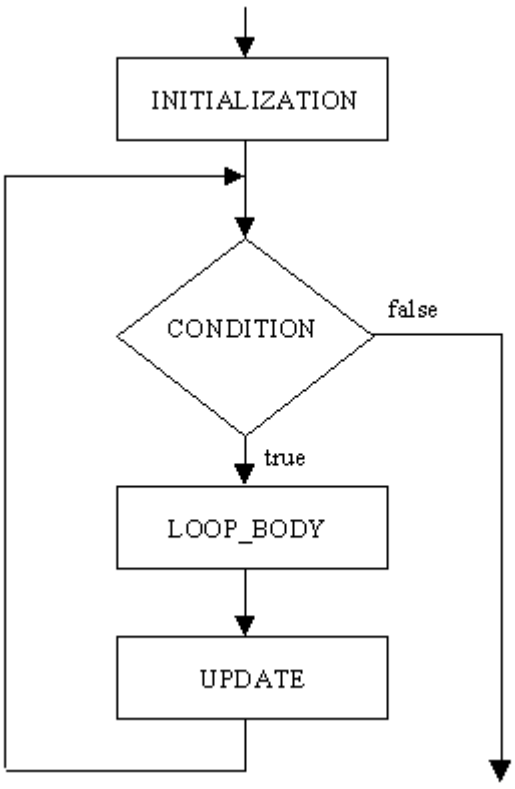
For loops can be used when you know in advance how many times a block of code should be run.

The **for** loop is a special kind of loop covering the following tasks:

- Set a counter variable to some initial value
- Check to see if the conditional statement is true
- Execute the code spec held within the loop
- Increment a counter at the end of each iteration through the loop

Syntax:

for (initialize; condition; increment) { action to take }



```
<?php
    $number = 75;
   for ($count=1; $count<=5; $count++)
{
      echo $number . "<br>";
      $number += 75;
 }
?>
```

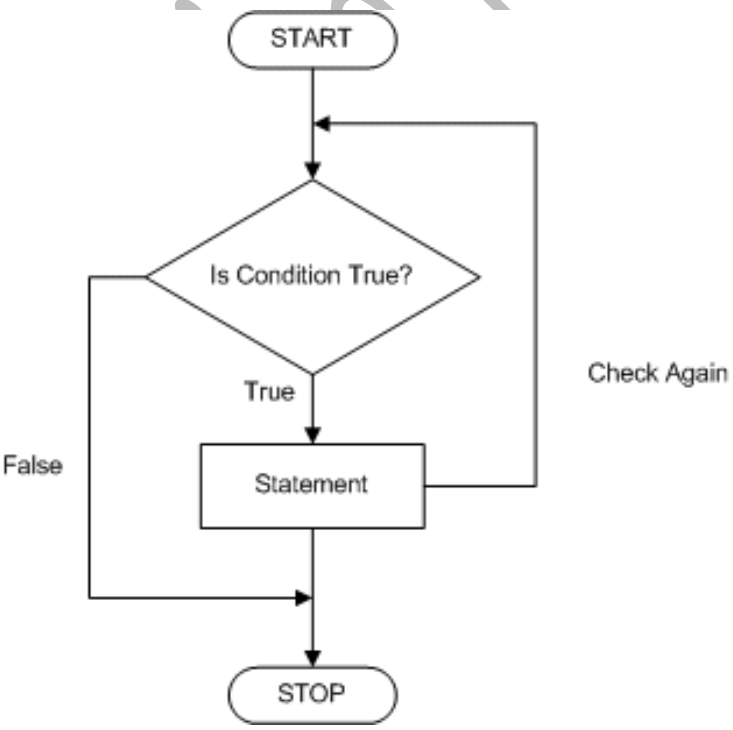
Output

75
150
225
300
375

The While loop

The while loop is quite easy to setup and use. A WHILE loop will, as the name suggests, execute a piece of code until a certain condition is met.

The function of a while loop is to do a task over and over as long as the specified conditional statement is true.



```
<?php
   $count = 1;
    while ($count<=5)
{
      echo $count . "<br>";
      $count++;
 }
?>
```

Output

1
2
3
4
5

ARRAY

An array is a variable that can store multiple values instead of just one. The values in an array can be referenced either collectively or individually.

You can create as many variables as you need to store your data individually, but what if that data is all related and you want to search through it, or sort it in a particular way? Storing related data in an array will allow you to perform these functions and many more.

Numerical arrays use numbers as the "key" that references each value stored in an array. The value of a numerical array begins with 0 instead of 1. This is the default array type.

Associative arrays use a unique ID key, specified by the programmer, to reference each value stored in an array.

Numerical Array / Indexed Array

By default, arrays are numerical, meaning that each value stored in an array is represented by a number. The number index always begins at 0 instead of 1.

Creating a numerical array is very simple. You can assign each value manually, or use the array() function, which accepts multiple values and associates each value with a unique ID number, or numerical key.

```
<?php
   $animals[0] = "Monkey";
   $animals[1] = "Panda";
   $animals[2] = "Spider";
   $animals[3] = "Elephant";
   $animals[4] = "Ferret";

    $animals = array("Monkey", "Panda", "Spider", "Elephant", "Ferret");

   $animals = array(1 => "Monkey", 2 => "Panda", 3 => "Spider", 4 => "Elephant", 5 => "Ferret");
?>
```

Associative Arrays

Associative arrays allow you to us a value as a key, so that a value can be assigned to each value.

Let's say, for example, that you run a zoo, and that you have a list of animals living at your zoo, and you need to keep track of how many of each animal that you have living at your zoo. This is too much information for a numerical array, but not for an associative array.

```
<?php
   $zoo_animals['Monkey'] = 15;
   $zoo_animals['Panda'] = 3;
   $zoo_animals['Spider'] = 167;
   $zoo_animals['Elephant'] = 5;
   $zoo_animals['Ferret'] = 7;
```

```
   $zoo_animals = array("Monkey" => 15, "Panda" => 3, "Spider" => 167,
"Elephant" => 5, "Ferret" => 7);
?>
```

Values can be added to the end of an associative array at any time using the following syntax:
$array_name['key'] = 'value';

User Defined Functions

Writing user-defined functions is as simple as using the function command as:

```
Function myFunction()
{
        // Some PHP Code spec
}
```

While creating a function, a unique function name follows the **function** keyword (in this case **myFunction**). A pair of opening and closing parentheses follows the function name.

```
<?php
   function test() {
       echo "I am in a function!";
   }

   test();
?>
```

Output
I am in a function

Function can be executed by calling it by its name. This is known as Function Call.

Function with arguments

```
<?php
    function mathy_stuff($a, $b) {
      echo $a * $b;
   }

   mathy_stuff(5, 7);
?>
```

Output

35

In above script, we called function by using function call **mathy_stuff(5, 7).** So, here we pass two values 5 and 7. So when we call this function, 5 will be assigned to first variable of the function i.e. $a and 7 will be assigned to second variable of the function i.e. $b.

Function with return values

```
<?php
    function example($a, $b) {
      $total = $a + $b;
      return $total;
   }

   $addition = example(50, 51);
   echo $addition ." dalmations!";
?>
```

Output

101

In above script, we called function by using function call **example(50, 51).** So, here we pass two values 50 and 51. So when we call this function, 50 will be

assigned to first variable of the function i.e. $a and 51 will be assigned to second variable of the function i.e. $b. Then, the script perform addition and answer is in $total variable. And we return the value of $total to calling function variable which is $addition.

DISCLAIMER :

This study material is prepared by **Mr. Binit Patel**. The objective of this material is to supplement teaching and discussion in the class room in the subject. Students are required to go for extra reading in the subject through library book.