APPLICATION DEVELOPMENT AND EMERGING TECHNOLOGIES

MODULE 2: PHP OPERATORS AND CONTROL STRUCTURES

Objectives

- To understand the different types of operators that are available on PHP.
- To know what is operator precedence and operator associativity in PHP.
- To ensure that programs expression are logically correct by using the correct referencing values of its precedence.
- To use escape sequence properly in the program.
- To know the different approach of control structures.
- To know the fundamentals syntax for conditional and looping structures.
- To properly use the compound expression using the logical operators.
- To know the rules of break, continue, and goto statements.
- To see some alternative enclosure for control structures.

OPERATORS

An operator is a symbol that specifies a particular action in an expression

OPERATOR PRECEDENCE, ASSOCIATIVITY, AND PUPOSE

Operator	Associativity	Purpose
new	NA	Object instantiation
( )	NA	Expression subgrouping
[ ]	Right	Index enclosure
! ~ ++ --	Right	Boolean NOT, bitwise NOT, increment, decrement
@	Right	Error suppression
/ * %	Left	Division, multiplication, modulus
+ - .	Left	Addition, subtraction, concatenation
<< >>	Left	Shift left, shift right (bitwise)
< <= > >=	NA	Less than, less than or equal to, greater than, greater than equal to
& ^ |	Left	Bitwise AND, bitwise XOR, bitwise OR
&& ||	Left	Boolean AND, boolean OR
?:	Right	Ternary operator
= += *= /= .= %= &= |= ^= <<= >>=	Left	Assignment operators
AND XOR OR	Left	Boolean AND, boolean XOR, boolean OR
,	Left	Expression separation; example: $days = array(1=>"Monday", 2=>"Tuesday")

OPERATOR PRECEDENCE

is a characteristic of operators that determines the order in which they evaluate the operands surrounding them.

Example:

```php
<?php
    $price = 450;
    $discount = 12;
    $value = $price - $price * $discount / 100;
    echo "<p>$value pesos</p>";
    //is the same as
    $value = ($price - (($price * $discount) / 100));
    echo "<p>$value pesos</p>";
?>
```

Output:

localhost/php/s

396

396

OPERATOR ASSOCIATIVITY

is a characteristic of an operator specifies how operations of the same precedence are evaluated as they are executed.

Example:

```php
<?php
    $a = 600;
    $b = 30;
    $c = 5;
    $x = $a / $b * $c;
    echo "<p>$x</p>";
    //is also same as
    $x = (($a / $b) * $c);  //left to right
    echo "<p>$x</p>";

    $x = $y = $z = $c;
    echo "<p>$x</p>";
    //is also same as
    ($x = ($y = ($z = $c))); //right to lefts
    echo "<p>$x</p>";
?>
```

Output:

100

100

5

5

ARITHMETIC OPERATOR

Example	Label	Outcome
$a + $b	Addition	Sum of $a and $b
$a - $b	Subtraction	Difference of $a and $b
$a * $b	Multiplication	Product of $a and $b
$a / $b	Division	Quotient of $a and $b
$a % $b	Modulus	Remainder of $a divided by $b

ASSIGNMENT OPERATOR

Example	Label	Outcome
$a = 5	Assignment	$a equals 5
$a += 5	Addition-assignment	$a equals $a plus 5
$a *= 5	Multiplication-assignment	$a equals $a multiplied by 5
$a /= 5	Division-assignment	$a equals $a divided by 5
$a .= 5	Concatenation-assignment	$a equals $a concatenated with 5

INCREMENT AND DECREMENT OPERATOR

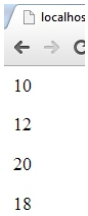
Example	Label	Outcome
++$a, $a++	Increment	Increment $a by 1
--$a, $a--	Decrement	Decrement $a by 1

Example:

```php
<?php
    $x = 10;
    //post increment
    echo "<p>".$x++."</p>"; //10
    //pre increment
    echo "<p>".++$x."</p>"; //12

    $y = 20;
    //post decrement
    echo "<p>".$y--."</p>"; //20
    //pre decrement
    echo "<p>".--$y."</p>"; //18
?>
```

Output:



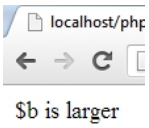
COMPARISON OPERATOR

Example	Label	Outcome
$a < $b	Less than	True if $a is less than $b
$a > $b	Greater than	True if $a is greater than $b
$a <= $b	Less than or equal to	True if $a is less than or equal to $b
$a >= $b	Greater than or equal to	True if $a is greater than or equal to $b
($a == 12) ? 5 : -1	Ternary	If $a equals 12, return value is 5; otherwise, return value is -1

Example:

```php
<?php
    $a = 10;
    $b = 20;
    $x = $a > $b ? "\$a is larger": "\$b is larger";
    echo $x;
?>
```

Output:



LOGICAL OPERATOR

Example	Label	Outcome
$a && $b	AND	True if both $a and $b are true
$a AND $b	AND	True if both $a and $b are true
$a || $b	OR	True if either $a or $b is true
$a OR $b	OR	True if either $a or $b is true
!$a	NOT	True if $a is not true
NOT $a	NOT	True if $a is not true
$a XOR $b	Exclusive OR	True if only $a or only $b is true

EQUALITY OPERATOR

Example	Label	Outcome
$a == $b	Is equal to	True if $a and $b are equivalent
$a != $b	Is not equal to	True if $a is not equal to $b
$a === $b	Is identical to	True if $a and $b are equivalent and $a and $b have the same type

BITWISE OPERATOR

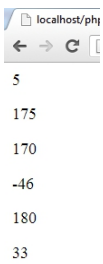
Example	Label	Outcome
$a & $b	AND	And together each bit contained in $a and $b
$a | $b	OR	Or together each bit contained in $a and $b
$a ^ $b	XOR	Exclusive-or together each bit

		contained in $a and $b
~ $b	NOT	Negate each bit in $b
$a << $b	Shift left	$a will receive the value of $b shifted left two bits
$a >> $b	Shift right	$a will receive the value of $b shifted right two bits

Example:

```php
<?php
    $a = 135; $b = 45;
    $c = $a & $b;
    echo "<p>$c</p>";
    $c = $a | $b;
    echo "<p>$c</p>";
    $c = $a ^ $b;
    echo "<p>$c</p>";
    $c = ~$b;
    echo "<p>$c</p>";
    $c = $b << 2;
    echo "<p>$c</p>";
    $c = $a >> 2;
    echo "<p>$c</p>";
?>
```

Output:



ESCAPE SEQUENCES

Sequence	Description
\n	Newline character
\r	Carriage return
\t	Horizontal tab
\\	Backslash
\$	Dollar sign
\”	Double quote
\[0-7]{1,3}	Octal notation
\x[0-9A-Fa-f]{1,2}	Hexadecimal notation

PHP CONTROL STRUCTURES: CONDITIONAL STATEMENTS

if statement

syntax:

```
if(expression) {
    statement…
}
```

else statement

syntax:

```
if(expression) {
    statement…
} else {
    statement
}
```

elseif statement

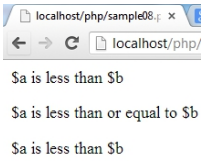
syntax:

```
if(expression) {
    statement…
}
elseif(expression) {
    statement…
} else {
    statement…
}
```

Example:

```php
<?php
    $a = 50; $b = 80;
    //if statement
    if($a < $b){
        echo "<p>\$a is less than \$b</p>";
    }
    //if...else statement
    if($a > $b){
        echo "<p>\$a is greater than \$b</p>";
    } else {
        echo "<p>\$a is less than or equal to \$b</p>";
    }
    //if...elseif...else
    if($a > $b){
        echo "<p>\$a is greater than \$b</p>";
    } elseif($a < $b) {
        echo "<p>\$a is less than \$b</p>";
    } else {
        echo "<p>\$a is equal to \$b</p>";
    }
?>
```

Output:



PHP CONTROL STRUCTURES: CONDITIONAL STATEMENTS
Nested if…else

Example:

```php
<?php
    $score = 86;
    if($score > 80){
        if($score < 91){
            echo "Your score is between 80 to 91";
        }else{
            echo "Your score is higher than 90";
        }
    }else{
        echo "Your score is lower than 81";
    }
?>
```

Output:

Your score is between 80 to 91

PHP CONTROL STRUCTURES: CONDITIONAL STATEMENTS COMPOUND EXPRESSION USING LOGICAL OPERATORS

Example:

```php
<?php
    $score = 86;
    if($score > 80 && $score<91){
        echo "Your score is between 80 to 91";
    } else if($score > 90){
        echo "Your score is higher than 90";
    } else{
        echo "Your score is lower than 81";
    }
?>
```

Output:

Your score is between 80 to 91

PHP CONTROL STRUCTURES: CONDITIONAL STATEMENTS

switch statement

syntax:

```
switch($category){
    case opt1:
        statement…
        break;
    case opt2:
        statement…
        break;
    case opt3:
        statement…
        break;
        …
    default:
        statement…
}
```

Example:

```php
<?php
    $month = 2;
    switch($month){
        case 1:
            echo "January"; break;
        case 2:
            echo "February"; break;
        case 3:
            echo "March"; break;
        case 4:
            echo "and so on up to 12"; break;
        default:
            echo "Invalid"; break;
    }
?>
```

Output:

February

PHP CONTROL STRUCTURES: LOOPING STATEMENTS

while statement

syntax:

```
while(expression){
    statement…
}
```

do … while statement

syntax:

```
do{
    statement…
}while(expression);
```

for statement

syntax:

```
for(expr1;expr2;expr3)
{
    statement…
}
```

Example:

```php
<?php
    $x = 10;
    while($x>0){
        echo $x--." ";
    }
    echo "<br/>";

    $y = 20;
    do{
        echo $y++." ";
    }while($y<21);
    echo "<br/>";

    $z=5;
    for($i=$z;$i>0;$i--){
        echo $i." ";
    }
?>
```

Output:

10 9 8 7 6 5 4 3 2 1
20
5 4 3 2 1

PHP CONTROL STRUCTURES: BREAK, CONTINUE, AND GOTO STATEMENT

break

break statement is placed within the code of a loop to cause the program to break out of the loop statement.

continue

continue statement causes execution of the current loop iteration to end and commence at the beginning of the next iteration.

goto … label:

goto statement is used to jump to other section of the program to support labels.

Example:

```php
<?php
    $i=1;
    while(true){
        $i++;
        if($i<20){
            continue;
        }elseif($i>=20 && $i<30){
            echo $i."<br/>";
        }else{
            break;
        }
    }
    goto display;
    echo "This statement were skipped by goto";
    display:
    echo "statement from display label";
?>
```

Output:

20
21
22
23
24
25
26
27
28
29
statement from display label

PHP CONTROL STRUCTURES: ALTERNATIVE ENCLOSURE SYNTAX

Involves replacing the opening bracket with a colon(:) and replacing the closing bracket with endif;, endwhile;,endfor;, endswitch

Example:

```php
<?php
    $a = 10; $b = 20; $c = 30;
    if($a<$b):
        echo "<p>\$a is less than \$b</p>";
    endif;
    for($i=1;$i<11;$i++):
        echo $i." ";
    endfor;
?>
```

Output:

$a is less than $b

1 2 3 4 5 6 7 8 9 10

SUMMARY

- Simplifying expression needs to follow a general precedence set by PHP scripts.
- An operator specifies a particular action in a given expression.
- Available PHP operators are arithmetic, conditional, assignment, logical and bitwise operator.
- Operator precedence characteristics determines the order in which they evaluate the operands surrounding them.
- Operator associativity characteristic specifies how operations of the same precedence are evaluated as they are executed.
- Post increment or decrement always execute the statement first before incrementing or decrementing the value of a given variable.
- Pre increment or decrement always increment or decrement its variable value before executing the statement.
- Escape sequence has its own special task in executing the string as an output.
- Conditional statements are statement the will execute statement inside a block if a given condition is true
- Use compound expression to minimize the process of coding using nested if statement.
- break statement caused the program to break if used.
- continue statement end the current loop and continue to next iteration.
- goto statement used to jump to other section of the program.
- Alternative enclosure syntax is available for if, while, for, foreach, and switch control.