Chapter 8

How to code control statements

Objectives

Applied

1. Use any of the statements that are presented in this chapter with any of the types of conditional expressions that are presented in this chapter as you develop your own applications.

Objectives (continued)

Knowledge

1. Describe type coercion and distinguish between the equality and identity operators.
2. Describe the order of precedence for the relational and logical operators and explain how parentheses can be used to control that order.
3. Describe the use of the if, switch, while, do-while, and for statements.
4. Describe the use of the break and continue statements.

The equality operators

```
== (double equal)
!=
<>
```

PHP Type Coercion Rules for comparisons

Operand 1	Operand 2	Action
NULL	String	Convert NULL to an empty string.
Boolean or NULL	Not a string	Convert both to Boolean.
String	Number	Convert string to a number.
Numeric string	Numeric string	Convert strings to numbers.
Text string	Text string	Compare strings as if using the strcmp function.

Unusual results with the equality operator

Expression	Result
`null == ''`	True; NULL is converted to the empty string.
`null == false`	True; NULL is converted to FALSE.
`null == 0`	True; NULL is equal to any value that evaluates to FALSE.
`false == '0'`	True; empty strings and “0” are converted to FALSE.
`true == 'false'`	True; all other strings are converted to true.
`3.5 == "\t3.5 mi"`	True; the string is converted to a number first.
`INF == 'INF'`	False; the string “INF” is converted to 0.
`0 == ''`	True; the empty string is converted to 0.

The identity operators: no type conversion.

Comparing different types using identity operators is always FALSE.

```
=== (triple equal)
!== (! double equal)
```

The relational operators

```
<
<=
>
>=
```

Comparing strings to numbers with the relational operators

Expression	Result
`1 < '3'`	true
`'10' < 3`	false

Comparing strings with the relational operators

Expression	Result
`'apple' < 'orange'`	true
`'apple' < 'appletree'`	true
`'Orange' < 'apple'`	true
`'@' < '$'`	false

Unusual results with the relational operators

Expression	Result
`0 <= 'test'`	True; the string "test" is converted to 0.
`'' < 5`	True; the empty string is converted to 0.
`false < true`	True; FALSE is considered less than TRUE.
`null < true`	True; NULL is converted to FALSE.

The logical operators

`!`	`NOT`
`&&`	`AND`
`||`	`OR`

Compound conditional expressions

The NOT operator

```
!is_numeric($number)
```

The AND operator

```
$age >= 18 && $score >= 680
```

The OR operator

```
$state == 'CA' || $state == 'NC'
```

Order of precedence for conditional expressions

Order	Operators
1	`!`
2	`<, <=, >, >=, <>`
3	`==, !=, ===, !==`
4	`&&`
5	`||`

The logical operators in conditional expressions

AND and OR operators

```
$age >= 18 && $score >= 680 || $state == 'NC'
```

AND, OR, and NOT operators

```
!$old_customer ||
    $loan_amount >= 10000 && $score < $min_score + 200
```

How parentheses can change the evaluation

```
(!$old_customer || $loan_amount >= 10000)
    && $score < $min_score + 200
```

An if clause with one statement and no braces

```
if (!isset($rate)) $rate = 0.075;
```

An if clause with one statement and braces

```
if ($qualified) {
    echo 'You qualify for enrollment.';
}
```

If and else clauses with one statement each and no braces

```
if ($age >= 18)
    echo 'You may vote.';
else
    echo 'You may not vote.';
```

Why you should use braces with else clauses

```
if ($age >= 18)
    echo 'You may vote.';
else
    echo 'You may not vote.';
    $may_vote = false;
```

Braces make your code easier to modify or enhance

```
if ($score >= 680) {
    echo 'Your loan is approved.';
} else {
    echo 'Your loan is not approved.';
}
```

A nested if statement to determine if a year is a leap year

```
$is_leap_year = false;
if ($year % 4 == 0) {
    if ($year % 100 == 0) {
        if ($year % 400 == 0) {
            $is_leap_year = true;
        } else {
            $is_leap_year = false;
        }
    } else {
        $is_leap_year = true;
    }
} else {
    $is_leap_year = false;
}
```

An if statement with one else if clause

```
if ($age < 18) {
    echo "You're too young for a loan.";
} else if ($score < 680) {
    echo "Your credit score is too low for a loan.";
}
```

An if statement with an else if and an else clause

```
if ($age < 18) {
    echo "You're too young for a loan.";
} elseif ($score < 680) {
    echo "Your credit score is too low for a loan.";
} else {
    echo "You're approved for your loan.";
}
```

An if statement with two else if clauses and an else clause

```
$rate_is_valid = false;
if (!is_numeric($rate)) {
    echo 'Rate is not a number.';
} else if ($rate < 0) {
    echo 'Rate cannot be less than zero.';
} else if ($rate > 0.2) {
    echo 'Rate cannot be greater than 20%.';
} else {
    $rate_is_valid = true;
}
```

An if statement to determine a student's letter grade

```
if ($average >= 89.5) {
    $grade = 'A';
} else if ($average >= 79.5) {
    $grade = 'B';
} else if ($average >= 69.5) {
    $grade = 'C';
} else if ($average >= 64.5) {
    $grade = 'D';
} else {
    $grade = 'F';
}
```

Syntax of the conditional operator

```
(conditional_expression) ? value_if_true : value_if_false
```

Set a string based on a comparison

With the conditional operator

```
$message = ($age >= 18) ? 'Can vote' : 'Cannot vote';
```

The same example with an if statement

```
if ($age >= 18) {
    $message = 'Can vote';
} else {
    $message = 'Cannot vote';
}
```

Examples of using the conditional operator

Set a string based on a comparison

```
$message = ($age >= 18) ? 'Can vote' : 'Cannot vote';
```

Calculate overtime pay

```
$overtime =
    ($hours > 40) ? ($hours - 40) * $rate * 1.5 : 0;
```

Select a singular or plural ending based on a value

```
$ending = ($error_count == 1) ? '' : 's'.
$message =
    'Found ' . $error_count . ' error' . $ending . '.';
```

Set a value to 1 if it's at a maximum value when adding 1

```
$value = ($value >= $max_value) ? 1 : $value + 1;
```

Return one of two values based on a comparison

```
return ($number > $highest) ? $highest : $number;
```

Bound a value within a fixed range

```
$value = ($value > $max) ? $max :
   (($value < $min) ? $min : $value);
```

Bound a value within a fixed range

With the conditional operator

```
$value = ($value > $max) ? $max :
    (($value < $min) ? $min : $value);
```

With an if statement

```
if ($value > $max) {
    $value = $max;
} else if ($value < $min) {
    $value = $min;
}
```

A switch statement with a default case

```
switch ($letter_grade) {
    case 'A':
        $message = 'well above average';
        break;
    case 'B':
        $message = 'above average';
        break;
    case 'C':
        $message = 'average';
        break;
    case 'D':
        $message = 'below average';
        break;
    case 'F':
        $message = 'failing';
        break;
    default:
        $message = 'invalid grade';
        break;
}
```

A switch statement with fall through

```
switch ($letter_grade) {
    case 'A':
    case 'B':
        $message = 'Scholarship approved.';
        break;
    case 'C':
        $message = 'Application requires review.';
        break;
    case 'D':
    case 'F':
        $message = 'Scholarship not approved.';
        break;
}
```

A while loop averages 100 random numbers

```
$total = 0;
$count = 0;
while ($count < 100) {
    $number = mt_rand(0, 100);
    $total += $number;
    $count++;
}
$average = $total / $count;
echo 'The average is: ' . $average;
```

A while loop that counts dice rolls until a six is rolled

```
$rolls = 1;
while (mt_rand(1,6) != 6) {
    $rolls++;
}
echo 'Number of times to roll a six: ' . $rolls;
```

Nested while loops that get the average and maximum rolls for a six

```
$total = 0;
$count = 0;
$max = -INF;

while ($count < 10000) {
    $rolls = 1;
    while (mt_rand(1, 6) != 6) {
        $rolls++;
    }
    $total += $rolls;
    $count++;
    $max = max($rolls, $max);
}
$average = $total / $count;
echo 'Average: ' . $average . ' Max: ' . $max;
```

A do-while loop that counts dice rolls until a six is rolled

```
$rolls = 0;
do {
    $rolls++;
} while (mt_rand(1,6) != 6);

echo 'Number of times to roll a six: ' . $rolls;
```

A do-while loop to find the max and min of 10 random values

```
$max = -INF;
$min = INF;
$count = 0;
do {
    $number = mt_rand(0, 100);
    $max = max($max, $number);
    $min = min($min, $number);
    $count++;
} while ($count < 10);

echo 'Max: ' . $max . ' Min: ' . $min;
```

Note

The mt_rand, min, and max functions are presented in detail in chapter 9.

The for statement compared to the while statement

The for statement

```
for ($count = 1; $count <= 10; $count++) {
    echo $count . '<br />';
}
```

The while statement

```
$count = 1;
while ($count <= 10) {
    echo $count . '<br />';
    $count++;
}
```

A for loop to display even numbers from 2 to 10

```
for ($number = 2; $number <= 10; $number += 2) {
    echo $number . '<br />';
}
```

A for loop to display all the factors of a number

```
$number = 18;
for ($i = 1; $i < $number; $i++) {
    if ($number % $i == 0) {
        echo $i . ' is a factor of ' . $number .
             '<br />';
    }
}
```

A for loop that uses the alternate syntax to display a drop-down list

```
<label>Interest Rate:</label>
<select name="rate">
<?php for ($v = 5; $v <= 12; $v++) : ?>
   <option value="<?php echo $v; ?>">
        <?php echo $v; ?>
    </option>
<?php endfor; ?>
</select><br />
```

The break statement in a while loop

```
while (true) {
    $number = mt_rand(1,10);
    if ($number % 2 == 0) {
        break;
    }
}
echo $number; // $number is between 1 and 10 and even
```

The break statement in a for loop

```
$number = 13;
$prime = true;
for ($i = 2; $i < $number; $i++) {
    if ($number % $i == 0) {
        $prime = false;
        break;
    }
}
$result = ($prime) ? ' is ' : ' is not ';
echo $number . $result . 'prime.';
```

The continue statement in a for loop

```
for ($number = 1; $number <= 10; $number++) {
    if ($number % 3 == 0) {
        continue;
    }
    echo $number . '<br />';
}
// Only displays 1, 2, 4, 5, 7, 8, and 10
```

The continue statement in a while loop

```
$number = 1;
while ($number <= 10) {
    if ($number % 3 == 0) {
        $number++;
        continue;
    }
    echo $number . '<br />';
    $number++;
}
// Only displays 1, 2, 4, 5, 7, 8, and 10
```