University of Anbar

College of Computer Sciences and

Information Technology

Web Development

Dr. Alaa Abdalqahar Jihad

Fourth Stage – Second Course

Lecture 3: Control Structures (Conditional Statements & Loops)

Objectives:

- Understand how conditional statements work in PHP.
- Use if, if-else, elseif, and switch statements for decision-making.
- Learn about different types of loops (for, while, do-while, foreach).
- Apply control structures to solve real-world programming problems.

Introduction to Control Structures

What are Control Structures?

- Control structures allow us to **control the flow of execution** in a program.
- Two main types in PHP:
    1. **Conditional Statements** – Execute different code based on conditions.
    2. **Loops** – Execute a block of code multiple times.

Conditional Statements (Decision Making)

If-Else Statement

Basic if Statement:

- Executes a block of code **only if** the condition is true.

```
<?php
$age = 20;
if ($age >= 18) {
   echo "You are eligible to vote.";
}
?>
```

if-else Statement:

- Provides an alternative block of code **if the condition is false**.

```
<?php
$age = 16;
if ($age >= 18) {
   echo "You can vote.";
} else {
   echo "You are too young to vote.";
}
?>
```

elseif Statement:

- Used when multiple conditions need to be checked.

```
<?php
$score = 85;
if ($score >= 90) {
   echo "Grade: A";
} elseif ($score >= 80) {
   echo "Grade: B";
} elseif ($score >= 70) {
   echo "Grade: C";
} else {
   echo "Grade: F";
}
?>
```

Switch Case Statement

Used as an alternative to multiple if-elseif conditions.

```
<?php
$day = "Monday";
switch ($day) {
   case "Sunday":
      echo "Start of the workweek!";
```

```
        break;
    case "Thursday":
        echo "Weekend is near!";
        break;
    case "Friday":
        echo "Relax, it's the weekend!";
        break;
    default:
        echo "Just another regular day.";
}
?>
```

Key Points:
Each case must end with break to prevent fall-through.
default is optional but recommended for handling unexpected values.

Loops (Repetitive Execution)

While Loop

Repeats a block of code while a condition is true.

```
<?php
$counter = 1;
while ($counter <= 5) {
    echo "Counter: $counter <br>";
    $counter++;
}
?>
```

Do-While Loop

Executes the loop at least once, then checks the condition.

```
<?php
$counter = 1;
do {
   echo "Number: $counter <br>";
   $counter++;
} while ($counter <= 5);
?>
```

Difference Between while and do-while:

Loop Type	Condition Checked	Executes at Least Once?
while	Before execution	No
do-while	After execution	Yes

For Loop

Used when the number of iterations is known in advance.

Loop Structure:

```
for (initialization; condition; increment/decrement) {
   // Code to be executed
}
```

```
<?php
for ($i = 1; $i <= 5; $i++) {
   echo "Iteration $i <br>";
}
?>
```

Example: Display Even Numbers from 1 to 10

```
<?php
for ($i = 2; $i <= 10; $i += 2) {
   echo "$i ";
}
?>
```

Foreach Loop

Used to iterate over arrays easily.

```
<?php
$colors = ["Red", "Green", "Blue"];
foreach ($colors as $color) {
   echo "$color <br>";
}
?>
```

Example: Associative Array Iteration

```
<?php
$person = ["name" => "Ali", "age" => 25, "city" => "Anbar"];
foreach ($person as $key => $value) {
   echo "$key: $value <br>";
}
?>
```

Summary

Conditional statements allow decision-making using if, elseif, else, and switch.

Loops help execute repetitive tasks:

- while – Runs **while** a condition is true.
- do-while – Runs **at least once**, then checks the condition.
- for – Used for **fixed number of iterations**.
- foreach – Iterates over **arrays** easily.

Homework Assignment

Create a PHP script that:

1. Checks if a number is even or odd using if-else.
2. Uses a for loop to display **odd numbers** from 1 to 20.
3. Uses a foreach loop to display the elements of an array containing **5 favorite fruits**.
4. Uses a for loop to display the multiplication table of 5.

Example Output:

1. 15 → odd.
2. 1 3 5 7 9 11 13 15 17 19
3. Apple, Banana, Mango, Orange, Grapes
4. 5 * 1 = 5
   5 * 2 = 10 …

Lecture 4: Arrays and Data Handling

Objectives:

- Understand what arrays are and their types in PHP.
- Learn how to create and manipulate arrays.
- Perform common array operations such as adding, removing, and searching for elements.
- Learn how to iterate through arrays using loops.
- Understand associative arrays and multidimensional arrays.

Introduction to Arrays

What is an Array?

- An **array** is a collection of elements stored under a single variable.
- Arrays help organize and manipulate large sets of data efficiently.
- PHP supports different types of arrays:
    1. **Indexed Arrays** (Numerical Index)
    2. **Associative Arrays** (Key-Value Pairs)
    3. **Multidimensional Arrays** (Arrays Inside Arrays)

Indexed Arrays (Numerical Arrays)

Creating and Accessing Indexed Arrays

Declaring an Indexed Array:

```
<?php
$colors = ["Red", "Green", "Blue"];
echo $colors[0]; // Outputs: Red
?>
```

Alternative Way (Using array() Function):

```
<?php
$colors = array("Red", "Green", "Blue");
?>
```

Modifying Array Elements:

```
<?php
$colors[1] = "Yellow"; // Changes "Green" to "Yellow"
?>
```

Adding Elements to an Array:

```
<?php
$colors[] = "Purple"; // Adds a new element
array_push($colors, "Orange"); // Another way to add an element
?>
```

Associative Arrays (Key-Value Pairs)

Creating and Accessing Associative Arrays

What is an Associative Array?

- Uses named **keys** instead of numeric indexes.

Declaring an Associative Array:

```
<?php
$person = [
   "name" => "Ali",
   "age" => 25,
   "city" => "Anbar"
];
echo $person["name"]; // Outputs: Ali
?>
```

Modifying Associative Array Elements:

```
<?php
$person["city"] = "Baghdad"; // Change the city
?>
```

Adding New Key-Value Pairs:

```
<?php
$person["country"] = "Iraq"; // Adds a new key-value pair
?>
```

Multidimensional Arrays (Arrays Inside Arrays)

Understanding Multidimensional Arrays

What is a Multidimensional Array?

- An array containing one or more arrays inside it.

Example of a 2D Array:

```
<?php
$students = [
   ["Ali", 25, "Anbar"],
   ["Ahmed", 22, "Baghdad"],
   ["Sara", 23, " Anbar"]
];
echo $students[0][0]; // Outputs: Ali
?>
```

Example of an Associative Multidimensional Array:

```
<?php
$people = [
   "Ali" => ["age" => 25, "city" => "Anbar"],
   "Ahmed" => ["age" => 22, "city" => "Baghdad"]
];
echo $people["Ahmed"]["city"]; // Outputs: Baghdad
?>
```

Looping Through Arrays

Looping Through Indexed Arrays

Using for Loop:

```
<?php
$colors = ["Red", "Green", "Blue"];
for ($i = 0; $i < count($colors); $i++) {
   echo $colors[$i] . "<br>";
}
?>
```

Using foreach Loop (Best for Arrays):

```
<?php
foreach ($colors as $color) {
   echo $color . "<br>";
}
?>
```

Looping Through Associative Arrays

Using foreach Loop with Key-Value Pairs:

```
<?php
$person = ["name" => "Ali", "age" => 25, "city" => "Anbar"];
foreach ($person as $key => $value) {
   echo "$key: $value <br>";
}
?>
```

Common Array Functions

Useful PHP Array Functions

Function	Description	Example
count($array)	Returns the number of elements in an array	count($colors);
array_push($array, $value)	Adds an element to the end of the array	array_push($colors, "Black");
array_pop($array)	Removes the last element	array_pop($colors);
array_shift($array)	Removes the first element	array_shift($colors);
array_unshift($array, $value)	Adds an element to the beginning	array_unshift($colors, "White");
array_key_exists($key, $array)	Checks if a key exists in an associative array	array_key_exists("age", $person);
in_array($value, $array)	Checks if a value exists in an array	in_array("Red", $colors);
array_search($value, $array)	Returns the index of a value	array_search("Blue", $colors);

Example: Using count() and in_array()

```
<?php
$fruits = ["Apple", "Banana", "Cherry"];
if (in_array("Banana", $fruits)) {
    echo "Banana is in the list.";
}
echo "Total Fruits: " . count($fruits);
?>
```

Summary

Arrays store multiple values in a single variable.
Indexed arrays use numerical indexes, while associative arrays use named keys.
Multidimensional arrays store arrays inside arrays.
Loops (for, foreach) are useful for iterating through arrays.
PHP provides many built-in functions for working with arrays.

Homework Assignment

Create a PHP script that:

1. Declares an indexed array of 5 favorite colors. And, adds two more colors dynamically.
2. Loops through and finds the **largest number** in an array from 10 elements.
3. Uses an **associative array** to store student names and their scores, then prints those who scored above 80.

Example Output:

1. My favorite colors are: Red, Green, Blue, Yellow, Purple, Orange, Black.
2. Largest number in the array: 99
3. Students who scored above 80:
   - Ali: 90
   - Sara: 85