|  |  |
| --- | --- |
|  | Department of Information Technology - State Polytechnic of Malang  **Jobsheet-4: PHP**  **Web Design and Programming Courses**  Web Design and Programming Teaching Team  September 2024 |

**Name : Ailsa Sahda Garizah**

**Class : SIB 2G**

**NIM : 244107060006**

**Topic**

* Introduction to PHP

**Purpose**

Students are expected to:

1. Students are able to create static websites using PHP

**Introduction**

**What is PHP?**

PHP (PHP: Hypertext Preprocessor) is an open source server-side scripting programming language. As a scripting language, PHP executes programming instructions during runtime. The results of the instructions will certainly differ depending on the data processed. PHP is a server-side programming language, so scripts from PHP will later be processed on the server. Types of servers that are often used in conjunction with PHP include Apache, Nginx, and LiteSpeed. In addition, PHP is also an open source programming language. Users are free to modify and develop according to their needs.

**Why Use PHP?**

Today, no less than 78% of websites around the world use the programming language created by Rasmus Lerdorf in 1995. Even big platforms like Facebook use it.

So, what makes PHP so popular? Why use PHP that's more than two decades old? Here are some reasons:

* **It tends to be easy to learn** — compared to some other popular programming languages, PHP is easier to learn.
* **Abundant learning materials** — the "quite old" age of PHP has led to a lot of documentation, guidance, and active communities scattered around the virtual space. So, there is no need to be afraid if you are having difficulties.
* **PHP is open-source** — anyone can use PHP without spending a dime.
* **High speed** — PHP has been proven to improve loading speed compared to other languages. For example, it is three times faster than Python in some cases.
* **Lots of database options** — PHP can be used in almost any type of database. Starting from MySQL, to non-relational databases such as Redis.
* **Good compatibility with HTML** — PHP scripts don't interfere with HTML at all. Instead, the two of them complement each other.
* **High flexibility —** PHP can be combined with a lot of other programming languages. So you can use it as needed.
* **Multi-platform** — PHP can be used on a variety of operating systems. Starting from Windows, Linux, to MacOS.
* **It's always up to date** — since it first appeared in 1995, PHP is now at version 8.2.4.
* **Supports cloud services** — who would have thought, even though PHP is almost two decades old, it can support cloud services with good scalability.

**PHP Basic Syntax**

Each programming language has its own coding rules. The same goes for PHP. All PHP code written should be stored with .php extension files. The basic syntax opens with <?php and closes with ?> as seen in the following example:

|  |
| --- |
| **<?php**  echo "Welcome";  **?>** |

Description:

* **<?php** : mandatory code to open PHP programs.
* echo : a command to display text.
* "Welcome": the text to be displayed and written between quotation marks and semicolons.
* **?>** is the code to terminate PHP and is mandatory when combined with other programming languages such as HTML.

PHP syntax is **case sensitive**. So, the use of uppercase or lowercase letters will also affect the output given. Example:

|  |
| --- |
| **<?php**  $alamat = "Yogyakarta";  echo $alamat;  **?>** |

The above code will produce the output: **Yogyakarta** on the website page.

However, if it is written like this:

|  |
| --- |
| **<?php**  $alamat = "Yogyakarta";  echo $Alamat;  **?>** |

Then an error will be generated. The reason is that there is a difference between  **the $alamat** and **$Alamat** variables.

In PHP, comments can be written as an explanation of the code. Comments in PHP use or **\*/** and will not be displayed as output in the browser. Examples are as follows:

|  |
| --- |
| **<?php**  Here's an example of using comments  echo "How are you?";  /\*  Well this is also an example of a comment  written more than one line  \*/  **?>** |

**Writing PHP Codes**

Basically, the coding of this programming language is divided into two:

### 1. PHP Native

Native is writing PHP code from scratch when designing a website. PHP Native is often used by developers who have good coding skills or those who want to create unique pipeline frameworks with high functionality.

### 2. PHP Framework

When using a framework, developers can take advantage of ready-made website management frameworks. This means that there is no need to make it from scratch making the work easier. A framework is a framework that can help developers work more efficiently and complete website development faster. Some of the popular PHP frameworks used include: CodeIgniter, Laravel framework, Yii, Symfony and Zend Framework. If you are already proficient in PHP native, it is highly recommended to try switching to the PHP framework. That's because the code in the framework has been optimized according to standards, in terms of speed and security.

### Variables in PHP

Variables are storage places that are used to store data that can change during the course of a program. Variables can be thought of as containers filled with different types of data. Here are a few things you need to know about variables in PHP:

* **Variable Declaration:** the variable is declared with a dollar sign ($) followed by the variable name. For example: **$namaVariabel**.
* **Variable Naming Rules:**  variable names must begin with a letter or underscore ( \_ ) and may contain letters, numbers, or underscores. However, there is a difference between case and case sensitive. Examples: **$nama**, **$umur**, **$alamat**.
* **Assignment Value:** a value can be assigned to a variable by using the assignment operator (=). Example: **$nama = "John";**.
* **Data Types:** PHP is a programming language whose data types are dynamic, meaning that there is no need to initialize data types when declaring variables. Variables can store different types of data, such as strings, numbers, booleans, arrays, and others.
* **Accessing Variable Values:**  variable values can be accessed by using their variable names. Example: **$nama**.

Example of implementing a variable script:

* In the form of a string

|  |
| --- |
| **<?php**  $name = "John";  $job = "Programmer";  echo "Hello, my name is $name and I am a $job.";  **?>** |

* In the form of numbers

|  |
| --- |
| **<?php**  $age = 25;  $height = 175.5;  echo "I am $age years old and I am $height cm tall.";  **?>** |

### Constants in PHP

A constant is a fixed value that cannot be changed during the course of the program. They are often used to store values that should not change during the execution of a program. Here are a few things about constants in PHP:

* **Constant Declaration:** Constants are declared using **the define() function**  with two parameters: the name of the constant and its value. Example: **define("PI", 3.14);**.
* **Constant Naming Rules:** Constants are generally written in capital letters and use an underscore (\_) as a word separator if the name of the constant consists of several words. Example: **MAX\_VALUE**.
* **Constant Data Types:** Constants can also store different types of data, such as strings, numbers, booleans, or arrays. But, once a constant is defined, its value cannot be changed during the program.
* **Accessing Constants:** You can access constants by using the name of the constant. Example: **echo PI;**.

Example of a constant script implementation:

|  |
| --- |
| **<?php**  define("PI", 3.14159265);  $radius = 5;  $area = PI \* ($radius \* $radius);  echo "The area of a circle with a radius of $radius is $area.";  **?>** |

**Practical Section 1. Variables and Constants**

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Create a new file in the dasarWeb folder and name it variabelkonstanta.php. |
| 2 | Write the following code in the variabelkonstanta.php file |
| 3 |  |
| 4 | Add the contents of the variabelkonstanta.php file with the code below. |
| 5 |  |
| 6 | Add the contents of the variabelkonstanta.php file with the code below. |
| 7 |  |
| 8 | Save the file, then open a browser and run localhost/dasarWeb/variabelkonstanta. |
| 9 | What do you understand from the use of variables on the file? Record your understanding below. (Question No. 1)    Variables are used to store values that can be processed and displayed again by the program. For example, the variables number1 and number2 store the numbers 10 and 5, which are then added together in the variable result to display the calculation result. In addition, the variables true and false are used to store logical (boolean) values, namely true and false, the results of which are displayed when executed. From this, it can be seen that variables in PHP function as data containers whose values can be numbers, text, or logic, which can later be processed and displayed according to the program's needs |

**Data Types in PHP**

A data type is a classification of a type of data or a form of data. There are a wide variety of data types ranging from native data types to artificial data types. PHP supports at least eight scalar data types. The eight types of data are:

|  |  |
| --- | --- |
| **Data Type** | **Description** |
| Integer | Contains whole numbers. |
| Float | Contains decimal numbers. |
| Boolean | Contains only 2 values: true and false. |
| String | Contains text data enclosed by a ''' or " " ' sign. |
| Array | Contains a dataset. |
| Object | Used to create objects from predefined classes. |
| NULL | It is used to state that a variable has no value. |
| Callable | Used to store a reference to a callable function or method. |

**Practicum 2: Use of Data Types**

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Create a new file named tipedata.php. Type the code in step 2 inside the tipedata.php |
| 2 |  |
| 3 | Save the file, then open a browser and run localhost/dasarWeb/tipedata.php |
| 4 | Explain what you understand from the code below. (Question No. 2)    Variables are used to store values ​​and perform arithmetic operations. Each variable is defined with its own contents, some of which are directly defined with numbers and others with the results of calculations from other variables. All variable values ​​are displayed using the echo command with the addition of br to display the results on a new line in the output. Finally, the var\_dump function is used to display the data type and value of the variable $e, which is int(40). |
| 5 | Add the contents of the tipedata.php file with the code below. |
| 6 | Save the file, then open a browser and run localhost/dasarWeb/tipedata.php |
| 7 | Explain what you understand from the code below. (Question No. 3)    This query calculates the average grade for three subjects: Mathematics, Science, and Indonesian. These three variables are each assigned a value of 5.1, 6.7, and 9.3. The average is then calculated by adding the three variables and dividing by 3. The result is stored in the rataRata variable. Echo is then used to display the grades for each subject and the average on the screen. The var\_dump(rataRata) function is used to display the average grade information along with its data type, which in this case is float because it involves decimal numbers. |
| 8 | Add the contents of the tipedata.php file with the code below. |
| 9 | Save the file, then open a browser and run localhost/dasarWeb/tipedata.php |
| 10 | Explain what you understand from the code below. (Question No. 4)    The query contains Boolean data, namely whether the Student Passed, which has a true value, and whether the Student Has Passed an Exam, which has a false value. Boolean variables are used to represent true or false conditions. A true value indicates a statement is true, while a false value indicates a statement is false. To display this information, the var\_dump() function is used, which displays not only the value but also its data type. Therefore, var\_dump(apakahSiswaLulus) will display bool(true), while var\_dump(apakahSiswa SudahUjian) will display bool(false). Thus, the resulting output is two rows displaying the boolean values ​​of each variable. |
| 11 | Add the contents of the tipedata.php file with the code below. |
| 12 | Save the file, then open a browser and run localhost/dasarWeb/tipedata.php |
| 13 | Explain what you understand from the code below. (Question No. 5)    The query is used to store first and last name data into variables, which will then be displayed when the program is run. These variables are of type String, and these variables are then combined into a new variable, fullName, using the string concatenation operator. |
| 14 | Add the contents of the tipedata.php file with the code below. |
| 15 | Save the file, then open a browser and run localhost/dasarWeb/tipedata.php |
| 16 | Explain what you understand from the code below. (Question No. 6)    The query creates an array named listMahasiswa containing three string elements, namely "Wahid Abdullah", "Elmo Bachtiar", and "Lendis Fabri". This array is used to store a list of student names. After that, the command echo listMahasiswa[0]; is used to display the array element with index 0, which is the first element of the array. Because array indexes in PHP start at 0, the output produced in the browser is Wahid Abdullah, according to the displayed image showing the name. |

**Operators in PHP**

An operator in programming is a symbol or sign used to perform operations on data. Operators are used to perform various actions such as mathematical operations, comparisons, logic, and more on the values present in the program. We use operators in the PHP programming language when it is necessary to perform basic operations such as data manipulation, value comparison, condition evaluation, and various other actions on the values present in the program. Operators make it possible to change, manipulate, or make decisions based on those values.

**Types of Operators in PHP**

Here is a table that contains a list of some of the commonly used operators in PHP

| Operator Type | Symbol | Description and Examples of Use |
| --- | --- | --- |
| Aritmatika | + | Sum: **$hasil = $angka1 + $angka2;** |
|  | – | Subtraction: **$hasil = $angka1 – $angka2;** |
|  | \* | Multiply: **$hasil = $angka1 \* $angka2;** |
|  | / | Division: **$hasil = $angka1 / $angka2;** |
|  | % | Remainder Divided: **$sisa = $angka1 % $angka2;** |
| Checklists | == | Equal to: **$hasil = $nilai1 == $nilai2;** |
|  | != | Not equal to: **$hasil = $nilai1 != $nilai2;** |
|  | < | Smaller than: **$hasil = $nilai1 < $nilai2;** |
|  | > | Greater than: **$hasil = $nilai1 > $nilai2;** |
|  | <= | Less than or equal to: **$hasil = $nilai1 <= $nilai2;** |
|  | >= | Greater than or equal to: **$hasil = $nilai1 >= $nilai2;** |
| Logic | && / and | AND (dan): **$hasil = $a && $b;** |
|  | || / or | OR (atau): **$hasil = $a || $b;** |
|  | ! / not | NOT (negasi): **$hasil = !$a;** |
| Assignment | = | Typical assignment: **$a = $b;** |
|  | += | Assignment with summation: **$a += $b;** |
|  | -= | Assignment with subtraction: **$a -= $b;** |
|  | \*= | Assignment by multiplication: **$a \*= $b;** |
|  | /= | Assignment by division: **$a /= $b;** |
|  | %= | Assignment with remainder divided by: **$a %= $b;** |
| Increment/ Decrement | ++ | Increment (menambah 1): **$a++;** |
|  | — | Decrement (mengurangi 1): **$a–;** |
| Bitwise | & | AND bitwise: **$hasil = $a & $b;** |
|  | | | OR bitwise: **$hasil = $a | $b;** |
|  | ^ | XOR bitwise: **$hasil = $a ^ $b;** |
|  | ~ | NOT bitwise: **$hasil = ~$a;** |
| Appointment | \*\* | Rank: **$hasil = $angka \*\* $pangkat;** |
| Identik | === | Identical (values and data types are the same): **$hasil = $a === $b;** |
|  | !== | Not Identical (values or data types are not the same): **$hasil = $a !== $b;** |
| Ternary | ? : | Conditional operator (ternary): **$hasil = ($nilai > 10) ? "Bigger" : "Smaller";** |
| Array | [] | Access array elements: **$nilai = $array[0];** |
|  | + | Array merging: **$array 3 = $array 1 + $array 2;** |

**Practical Section 3: The Use of PHP Operators**

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Create a new file named operator.php. Type the code in step 2 inside the operator.php |
| 2 |  |
| 3 | Complete the program code above so that it can display the results and be neat.  (Question No. 7)    The query above is the result of adding code so that the program can display output when running, this addition is done using the echo command. |
| 4 | Save the file, then open a browser and run localhost/dasarWeb/operator.php . Observe what is happening and explain what you understand. (Question No. 8)    This query combines the use of various arithmetic operators. It includes addition, subtraction, multiplication, division, remainder, and exponentiation. Each result is stored in a separate variable and then displayed when run using the echo command, resulting in a page display that resembles the image showing all the calculation results. |
| 5 | Add the following code to the operator.php |
| 6 | Complete the program code above so that it can display the results and be neat. Save the file, then open a browser and run /refresh localhost/basicWeb/operator.php  Observe what is happening and explain what you understand. (Question No. 9)      The query shows how to use comparison operators to compare two values, for example variables a and b. Each line creates a new variable that stores the result of the comparison, such as a == b to check if the two values ​​are equal, a != b to check if the values ​​are different, a < b to check if a is smaller, a > b to check if a is greater, and a <= b and a >= b to check if a is less than or greater than b. The comparison result is a boolean value that when displayed directly will appear as 1 if the value is true and empty without a number if false. |
| 7 | Add the code in step 8 on operator.php |
| 8 |  |
| 9 | Complete the program code above so that it can display the results and be neat. Save the file, then open a browser and run /refresh localhost/basicWeb/operator.php  Observe what is happening and explain what you understand. (Question No. 10)      Logical operators work on two numeric variables a and b. The AND (&&) operator returns true only if both variables are true or non-zero. Since both a and b contain non-zero numbers, the AND result is true and when printed appears as 1. The OR (||) operator returns true if one or both variables are true, so in this condition it also returns 1. Meanwhile, the NOT (!) operator functions to invert logical values. Since both a and b are considered true (non-zero), the inverse result of both is false, and the printed result will appear blank. |
| 10 | Add the code in step 11 on operator.php |
| 11 |  |
| 12 | Complete the program code above so that it can display the results and be neat. Save the file, then open a browser and run /refresh localhost/basicWeb/operator.php  Observe what is happening and explain what you understand. (Question No. 11)    This operator is used to perform arithmetic operations while updating variable values. For example, in a += b, the value of a is added to b and the result is immediately stored back into a, so that a becomes 15. This operation is the same as other operations; the operator is a shortcut for calculation operations so that queries are not too long and calculation operations appear simple. |
| 13 | Add the code in step 14 on operator.php |
| 14 |  |
| 15 | Complete the program code above so that it can display the results and be neat. Save the file, then open a browser and run /refresh localhost/basicWeb/operator.php  Observe what is happening and explain what you understand. (Question No. 12)      In PHP, the identical operator (===) checks whether two values are the same and have the same data type. Meanwhile, the non-identical operator (!==) checks whether the values or data types are different. In variables a and b, the values are different, so a === b produces false and the output displays nothing, indicating that the numbers are different, while a !== b produces true and displays the number 1, indicating that the variable numbers are not the same. |
| 16 | **Question**: A restaurant has 45 seats in it. On one night, 28 seats were occupied by customers. What percentage of seats are still empty in the restaurant? |
| 17 | Create a file with named latihanoperator.php. Write down the program code for step 16 and display the result below along with the program code (Question No. 13) |

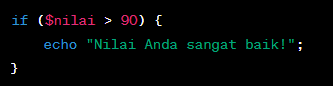
**Control Structure In PHP**

Basically, programming is about controlling the execution flow of a program. A control structure is a tool used to decide how code will be executed based on certain conditions.

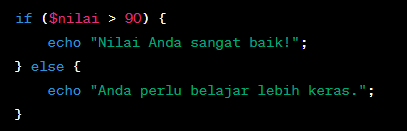
**Conditional Statement**

Conditional statements are used to make decisions based on certain conditions. There are several types of conditional statements in PHP:

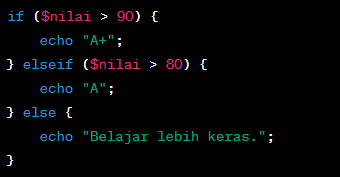
* **if**: Used to execute a block of code if certain conditions are true.



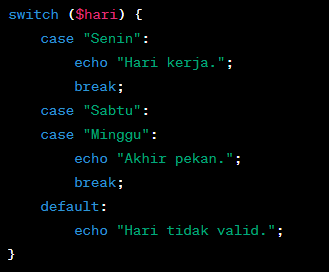
* **else**: Used in conjunction with **if** to execute a block of code if the condition is incorrect.



* **elseif**: Used to add additional conditions after **if**.



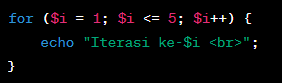
* **switch**: Used to evaluate expressions and execute code based on matching values.



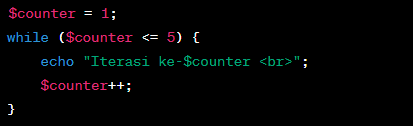
**Looping**

Looping allows you to iterate through the code multiple times based on certain conditions. There are three types of loops that are most commonly used in PHP:

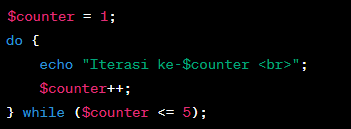
* **for**: Used to repeat the code a certain number of times.



* **while**: Used to repeat code as long as a certain condition is true.



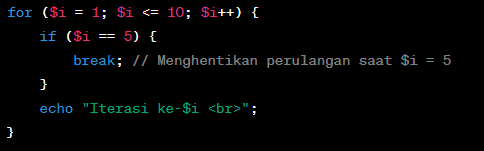
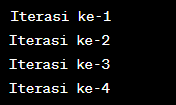
* **do-while**: Similar to **while**, but will execute a block of code at least once before checking for conditions.



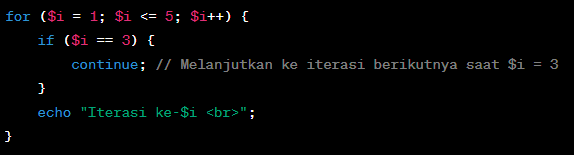
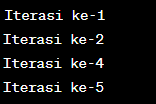
**Flow Control Statement**

This statement allows you to control the flow of program execution in a certain way:

* **break**: Used to stop the current loop or switch.

* **continue**: Used to resume the next iteration in the loop.

**Practical Section 4: The Use of Control Structures in PHP**

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Create a new file named strukturkontrol.php. Type the code in step 2 inside the strukturkontrol.php |
| 2 |  |
| 3 | Save the file, then open a browser and run /refresh localhost/dasarWeb/strukturkontrol.php |
| 4 | Note here what you observe from the addition of the program code above. (Question No. 14)    The variable nilaiNumerik is given a value of 92, then the program checks the conditions sequentially. Since 92 is in the range of 90 to 100, the first condition is met and the program immediately displays the letter grade: A. Other conditions such as B, C, or D are not executed because only one condition is selected. From this added code, it can be understood that the program is able to determine the letter grade category based on the numerical value entered, so that if the value is different, the letter displayed will also adjust to the predetermined value range. |
| 5 | Add the code in step 6 in strukturkontrol.php |
| 6 |  |
| 7 | Complete the program code above so that the result is neat. Save the file, then open a browser and run /refresh localhost/dasarWeb/strukturkontrol.php |
| 8 | Note here what you observe from the addition of the program code above. (Question No. 15)    The query adds logic using a while loop to calculate how long it takes an athlete to cover a distance of 500 kilometers. The initial value of currentDistance is 0, with a target distance of 500, and each day the athlete increases their distance by 30 kilometers. The while loop will continue to add distance until it reaches or exceeds the target. The day variable is used to calculate the number of days required. The result shows that 17 days is the time required to cover a distance of 500 kilometers. |
| 9 | Add the code in step 9 inside the strukturkontrol.php |
| 10 |  |
| 11 | Complete the program code above so that the result is neat. Save the file, then open a browser and run /refresh localhost/dasarWeb/strukturkontrol.php |
| 12 | Note here what you observe from the addition of the program code above. (Question No. 16) |
| 13 | Add the code in step 14 inside the strukturkontrol.php |
| 14 |  |
| 15 | Complete the program code above so that the result is neat. Save the file, then open a browser and run /refresh localhost/dasarWeb/strukturkontrol.php |
| 16 | Note here what you observe from the addition of the program code above. (Question No. 17) |
| 17 | Add the code in step 18 inside the strukturkontrol.php |
| 18 |  |
| 19 | Complete the program code above so that the result is neat. Save the file, then open a browser and run /refresh localhost/dasarWeb/strukturkontrol.php |
| 20 | Note here what you observe from the addition of the program code above. (Question No. 18)    The query uses an array and a foreach loop to evaluate student scores one by one. The nilaiSiswa variable contains a collection of numerical values from several students. Then, using foreach, each value is taken and checked using an if structure. If the value is less than 60, the program displays the message “Failed” and uses the continue command to jump directly to the next iteration without executing the subsequent command. Conversely, if the value is greater than or equal to 60, the program displays the message “Passed.” |
| 21 | **Question**: A teacher wants to calculate the total score of 10 students in a math exam. This teacher wanted to ignore the two highest grades and the two lowest grades. Help this teacher calculate the total grades that will be used to determine the average grade after ignoring the highest and lowest grades. The following is a list of grades from 10 students (85, 92, 78, 64, 90, 75, 88, 79, 70, 96) |
| 22 | Create a file with the name kontrol1.php. Create the program code for step 21 and display the result below along with the program code (Question No. 19) |
| 23 | There is a story: A customer wants to buy a product at a price of Rp 120,000. The store offers a 20% discount for purchases above IDR 100,000. Help these customers to calculate the price to be paid after getting the discount. |
| 24 | Create a file with the name kontrol2.php. Create the program code for step 23 and display the result below along with the program code (Question No. 20) |
| 25 | **Question:** A gamer wants to calculate their total score in the game. They get a score based on the points they collect. If they have more than 500 points, then they will get additional rewards. Create the first line of "Player's total score is: (**points**)". And the second line "Do players get additional rewards? (**YES/NO**)" |
| 26 | Create a file with the name kontrol3.php. Create the program code for step 25 show the result below along with the program code (Question No. 21) |

**Practical Section 5: Using Arrays in PHP**

### One-Dimensional Array

A one-dimensional array is a data structure that is used to store a number of values in a single variable. You can declare a one-dimensional array in PHP using several methods:

#### Method 1: Using array()



#### Method 2: Using square brackets [] (Starting from PHP 5.4+)



To access the values in the array used indexes (starting from 0 for the first index):



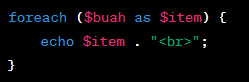
You can add a value to an array using a specific index or by the **[]** method to add a new value to the end of the array:



You can change the values in an array by accessing its index and replacing it:



You can iterate through all the elements in the array using **for**, **foreach**, or **while** loops. Here's an example with **foreach**:

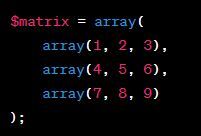


### Two-Dimensional Array

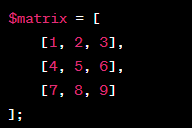
A two-dimensional array is a data structure used to store data in the form of a two-dimensional table, which is similar to rows and columns in a spreadsheet. Here is a brief explanation of two-dimensional arrays and examples of their implementation in PHP:

### 1. Two-Dimensional Array Declaration

You can declare a two-dimensional array in PHP by composing a one-dimensional array in another array:



You can also use the square bracket **syntax [][]**:



### 2. Accessing Values in a Two-Dimensional Array

You can access values in a two-dimensional array by using two indexes, one for rows and one for columns:



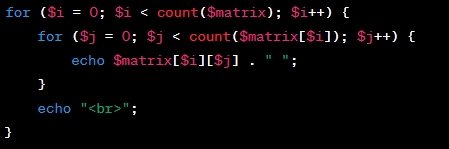
### 3. Adding and Changing Values in a Two-Dimensional Array

You can add or change values in a two-dimensional array by accessing elements using an index:

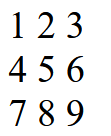


### 4. Iteration Through Two-Dimensional Arrays

You can iterate through all elements in a two-dimensional array using nested loops, such as loop **for**:



This will print the entire element in a two-dimensional array, resulting in a table-like appearance.

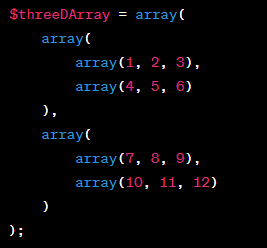


### Multidimensional Array

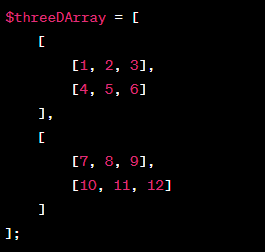
A multidimensional array is a data structure that allows you to store data in more than two dimensions. This is an expansion of the two-dimensional array described earlier.

### 1. Three-Dimensional Multidimensional Array Declaration

You can declare a three-dimensional multidimensional array by adding an array within an array:

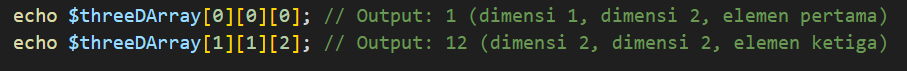


This is an example of a three-dimensional array. You can also use the square bracket syntax **[][][]**:



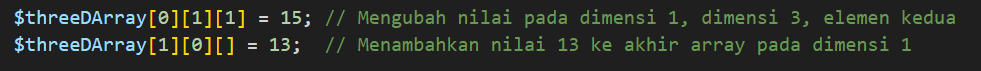
### 2. Accessing Values in a Three-Dimensional Multidimensional Array

You can access values in a three-dimensional multidimensional array by using three indexes, each for the corresponding dimension level:



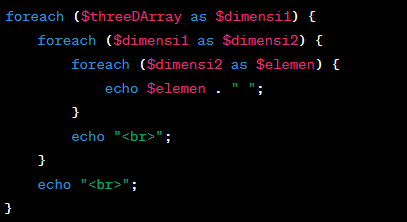
### 3. Adding and Changing Values in a Three-Dimensional Multidimensional Array

You can add or change values in a three-dimensional multidimensional array by accessing elements using an index:

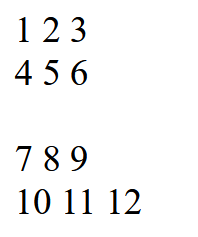


### 4. Iteration Through Three-Dimensional Multidimensional Arrays

To iterate through a three-dimensional multidimensional array, you need to use three levels of nested loops:



In the example above, we use a three-level nested loop to access all the elements in a three-dimensional array and print them.

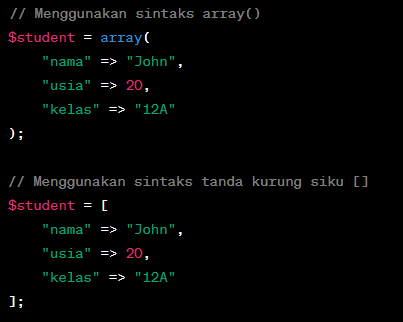


### Associative Array

An associative array is a type of array in PHP that uses a pair of "keys" and "values" to organize data. Here is an explanation and example of implementing associative arrays in PHP:

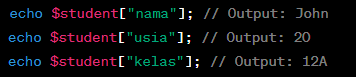
### 1. Associative Array Declaration

You can declare associative arrays by using **the array()** syntax or square **brackets [].** Each element of an associative array has a key that pairs with its value. Here's an example of using associative arrays in PHP:



### 2. Accessing Values in Associative Arrays

You can access the values in an associative array by using the appropriate key:

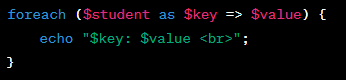


### 3. Adding and Changing Values in Associative Arrays

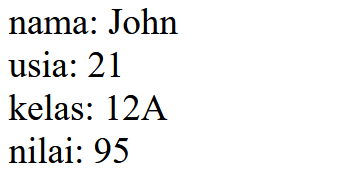
You can add or change values in an associative array by accessing elements using keys:



### 4. Iteration Through Associative Arrays



The output of the loop above will print all key-value pairs in the associative array.



|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Create a new file named array.php. Type the code in step 2 inside the array.php |
| 2 |  |
| 3 | Save the file, then open a browser and run /refresh localhost/dasarWeb/array.php |
| 4 | Note here what you observe from adding the code above. (Question No. 22)    The query uses an array to store a list of student scores. In each iteration, the student's score is checked with the condition if (score >= 70). If the condition is met, the score is entered into a new array named passingScore using the [] operator. |
| 5 | Type the additional code in step 6 inside the array.php |
| 6 |  |
| 7 | Save the file, then open a browser and run /refresh localhost/dasarWeb/array.php |
| 8 | Note here what you observe from the addition of the code above. (Question No. 23)    The query uses a two-dimensional array to store employee data and their years of experience. Each element in the Employee list consists of two parts: the employee's name at index 0 and the number of years of experience at index 1. Using a foreach loop, the program checks each employee. If the employee's work experience is more than 5 years (employee[1] > 5), then the employee's name is added to the fiveYearExperienceEmployees array. After all data has been checked, the final result is displayed, so that the list of employee names with more than 5 years of experience is displayed on the screen. |
| 9 | Type the additional code in step 10 inside the array.php |
| 10 |  |
| 11 | Save the file, then open a browser and run /refresh localhost/dasarWeb/array.php |
| 12 | Note here what you observe from the addition of the program code above. (Question No. 24)    The query stores student grade data in a nested array, where each course, such as Mathematics, Physics, and Chemistry, has a list of students and their respective grades. The mataKuliah variable is set to Physics, so the program will only retrieve and display data from that course. Using foreach, each element in the course array is processed. The execution results display a list of student grades for the Physics course, namely Alice with a grade of 90, Bob with a grade of 88, and Charlie with a grade of 75. |
| 13 | **Question**: A teacher wants to print a list of students' grades in a math exam. The teacher has data on each student consisting of names and grades. Help this teacher print a list of students who achieved grades above the class average. With the provision that the student's name and grade Alice can get 85, Bob can get 92, Charlie can get 78, David can get 64, Eva can get 90 |
| 14 | Create the code for step 13 with a two-dimensional array and display the result below along with the program code (Question No. 25) |