



ТЕМА 3

Керуючі структури та масиви в PHP

Умови · цикли · масиви · ітерація · редагування елементів · сортування та функціональна обробка

Бекенд-розробка, сервіси та мікросервіси

Комп'ютерна інженерія · 3 курс · Лекція 3 з 18

10 питань сьогоднішньої теми

01

if, elseif, else

02

switch та match

03

while та do-while

04

for та foreach

05

break і continue

06

Масиви: види та створення

07

Ітерація масиву

08

Редагування елементів

09

Сортування масивів

10

Функціональна обробка

if, elseif, else

```
<?php
$age = 20;

if ($age < 18) {
    echo "Неповнолітній";
} elseif ($age < 65) {
    echo "Дорослий";
} else {
    echo "Пенсіонер";
}
// Виведе: Дорослий
```

Truthy / Falsy

Вважаються false:

"" (порожній рядок)

0

[] (порожній масив)

null

false

match — сучасна альтернатива switch

switch

```
switch ($day) {  
    case 1:  
        echo "Пн";  
        break;  
    case 2:  
        echo "Вт";  
        break;  
    default:  
        echo "Інше";  
}
```

match (PHP 8.0+)

```
$dayName = match($day) {  
    1 => "Пн",  
    2 => "Вт",  
    default => "Інше",  
};  
  
// строге порівняння  
// без break  
// повертає значення
```

Умова перед циклом чи після нього

while

Перевіряє умову ДО виконання — тіло може не виконатись жодного разу

```
$i = 1;
while ($i <= 5) {
    echo $i . " ";
    $i++;
}
// 1 2 3 4 5
```

do-while

Перевіряє умову ПІСЛЯ виконання — тіло гарантовано виконається хоч раз

```
$n = 0;
do {
    $n++;
    echo "Спроба $n\n";
} while ($n < 3);
// Спроба 1, 2, 3
```

Лічильник vs перебір колекції

for

Коли заздалегідь відома кількість ітерацій

```
for ($i = 0; $i < 5; $i++) {  
    echo "Ітерація $i\n";  
}
```

foreach

Найпоширеніший спосіб перебору масивів

```
$fruits = ["яблуко", "банан"];  
  
foreach ($fruits as $i => $f) {  
    echo "$i: $f\n";  
}
```

break та continue

break

Зупиняє цикл повністю

```
foreach ([1,2,3,4,5] as $n) {  
    if ($n === 3) break;  
    echo $n . " ";  
}  
// 1 2
```

continue

Пропускає лише поточну ітерацію

```
foreach ([1,2,3,4,5] as $n) {  
    if ($n % 2 === 0) continue;  
    echo $n . " ";  
}  
// 1 3 5
```

Індексовані та асоціативні масиви

```
// Індексований масив
$colors = ["червоний", "зелений", "синій"];
echo $colors[0];    // червоний

// Асоціативний масив
$user = [
    "name" => "Оксана",
    "age"  => 25,
    "city" => "Львів",
];
echo $user["name"]; // Оксана
```

У PHP існує лише один тип масиву — упорядкований тар, який може бути індексованим, асоціативним або їх сумішшю.

Багатовимірні масиви та способи створення

```
$students = [  
    ["name" => "Іван", "grade" => 92],  
    ["name" => "Марія", "grade" => 87],  
];  
echo $students[0]["name"];    // Іван
```

Спосіб	Приклад
Короткий синтаксис []	<code>\$arr = [1, 2, 3];</code>
Функція <code>array()</code>	<code>\$arr = array(1, 2, 3);</code>
<code>range()</code>	<code>\$arr = range(1, 10);</code>
<code>array_fill()</code>	<code>\$arr = array_fill(0, 5, 0);</code>

foreach за посиланням

```
$numbers = [1, 2, 3, 4, 5];

foreach ($numbers as &$num) {
    $num = $num * 2;
}
unset($num);    // важливо "відв'язати" посилання!

print_r($numbers);    // [2, 4, 6, 8, 10]
```

unset(\$num) необхідний — інакше змінна далі "вказує" на останній елемент масиву й може непомітно зіпсувати дані.

Додавання елементів

Операція	Приклад
Додати в кінець	<code>\$arr[] = "новий";</code>
Додати з ключем	<code>\$arr["key"] = "значення";</code>
<code>array_push()</code> — один/кілька	<code>array_push(\$arr, "a", "b");</code>
<code>array_unshift()</code> — на початок	<code>array_unshift(\$arr, "перший");</code>

```
$cart = ["хліб", "молоко", "яйця"];  
$cart[] = "сир";           // [хліб, молоко, яйця, сир]  
array_unshift($cart, "вода"); // [вода, хліб, молоко, яйця, сир]
```

Видалення елементів

Операція	Приклад
<code>unset()</code> — за ключем	<code>unset(\$arr["key"]);</code>
<code>array_pop()</code> — останній	<code>\$last = array_pop(\$arr);</code>
<code>array_shift()</code> — перший	<code>\$first = array_shift(\$arr);</code>

```
unset($cart[2]);           // видалити "молоко" за ключем
$cart = array_values($cart); // переіндексувати ключі підряд від 0
```

Після `unset()` за числовим ключем залишається "дірка" в нумерації — `array_values()` її прибирає.

Функції сортування

Функція	Що робить
<code>sort(\$arr)</code>	За зростанням, переіндексовує ключі
<code>rsort(\$arr)</code>	За спаданням, переіндексовує ключі
<code>asort(\$arr)</code>	За значенням, зберігає ключі
<code>ksort(\$arr)</code>	За ключами (зростання)
<code>usort(\$arr, \$fn)</code>	За власною функцією порівняння

```
usort($students, function($a, $b) {  
    return $b["grade"] <=> $a["grade"];    // за спаданням оцінки  
});
```

array_map, array_filter, array_reduce

array_map()

Трансформує кожен елемент

```
array_map(fn($p) => $p*1.2,  
$prices);
```

array_filter()

Залишає елементи за умовою

```
array_filter($prices, fn($p) => $p >  
100);
```

array_reduce()

Згортає масив в одне значення

```
array_reduce($prices, fn($c,$p)=>$c+  
$p, 0);
```

Об'єднання, зрізи, пошук

Функція	Призначення
<code>array_merge(\$a, \$b)</code>	Об'єднує масиви в один
<code>array_slice(\$arr, \$off, \$len)</code>	Вирізає частину (не змінює оригінал)
<code>array_splice(\$arr, \$off, \$len)</code>	Вирізає й замінює (змінює оригінал)
<code>in_array(\$v, \$arr)</code>	Перевіряє наявність значення
<code>array_search(\$v, \$arr)</code>	Повертає ключ знайденого значення
<code>array_unique(\$arr)</code>	Видаляє дублікати

`array_slice()` не змінює оригінал; **`array_splice()`** змінює його на місці — поширена причина плутанини.

Підсумок теми 3

- `match` — сучасніша й безпечніша альтернатива `switch`
- `foreach` — основний спосіб ітерації масивів, & для зміни елементів
- У PHP один тип масиву: індексований, асоціативний або змішаний
- `array_push/pop/shift/unshift` керують кінцем і початком масиву
- `array_map`, `array_filter`, `array_reduce` — функціональна обробка без ручних циклів