Дисципліна

«Бекенд-розробка, сервіси та мікросервіси»

КОНСПЕКТ ЛЕКЦІЇ. ТЕМА 3

Керуючі структури: умови та цикли в PHP. Масиви: створення, ітерація, редагування та маніпуляція елементами

Спеціальність: Комп'ютерна інженерія, 3 курс
Обсяг: 1 год 20 хв (лекція 3 з 18)

План лекції

1. Умовні конструкції: if, elseif, else
2. Умовна конструкція switch та вираз match
3. Цикли: while та do-while
4. Цикли: for та foreach
5. Керування циклами: break і continue
6. Масиви в PHP: види та створення
7. Ітерація масиву
8. Редагування елементів масиву
9. Маніпуляція масивом: сортування
10. Маніпуляція масивом: функціональна обробка та корисні функції

> *Мета лекції: опанувати керуючі конструкції PHP (умови й цикли) та основний робочий інструмент серверного програміста — масиви, включно з їх ітерацією, редагуванням і трансформацією.*

1. Умовні конструкції: if, elseif, else

Умовні конструкції дозволяють виконувати різні блоки коду залежно від істинності виразу. Базова конструкція if перевіряє умову: якщо вона істинна (true після приведення до bool), виконується блок коду в фігурних дужках.

```
<?php
$age = 20;

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

Альтернативний синтаксис (з двокрапкою замість фігурних дужок) часто використовують у шаблонах, де PHP-код перемежається з HTML — він читається природніше при вставках у розмітку.

```
<?php if ($isLoggedIn): ?>
    <p>Вітаємо, <?= $username ?>!</p>
<?php else: ?>
    <p>Будь ласка, увійдіть.</p>
<?php endif; ?>
```

> *Важливо пам'ятати про «правдивість» значень (truthy/falsy): порожній рядок "", число 0, порожній масив [], null і false вважаються false у логічному контексті; усе інше — true. Це варто перевіряти явно (===), щоб уникнути неочевидних помилок.*

2. Умовна конструкція switch та вираз match

2.1. switch

Конструкція switch порівнює одне значення з кількома варіантами — зручна альтернатива довгому ланцюжку elseif, коли всі перевірки стосуються однієї змінної. Важливо не забувати break: без нього виконання «провалюється» (fall-through) у наступний case.

```
<?php
$day = 3;

switch ($day) {
    case 1:
        echo "Понеділок";
        break;
    case 2:
        echo "Вівторок";
        break;
    case 3:
        echo "Середа";
        break;
    default:
        echo "Інший день";
}
```

2.2. match (з PHP 8.0)

Вираз match — сучасніша альтернатива switch: використовує строге порівняння (===) за замовчуванням, не потребує break, і, головне, є виразом — тобто повертає значення, яке можна одразу присвоїти змінній.

```
<?php
$day = 3;

$dayName = match($day) {
    1 => "Понеділок",
    2 => "Вівторок",
    3 => "Середа",
    4, 5 => "Четвер або п'ятниця",   // кілька варіантів через кому
    default => "Інший день",
};
echo $dayName;   // Середа
```

> *На відміну від switch, match кидає помилку UnhandledMatchError, якщо жоден варіант не збігся і немає default — це допомагає раніше виявляти забуті випадки, а не мовчки нічого не виконувати.*

3. Цикли: while та do-while

3.1. while

Цикл while виконує блок коду, доки задана умова залишається істинною. Умова перевіряється перед кожною ітерацією — якщо вона хибна із самого початку, тіло циклу не виконається жодного разу.

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

3.2. do-while

Цикл do-while перевіряє умову ПІСЛЯ виконання тіла циклу — тому тіло гарантовано виконається хоча б один раз, навіть якщо умова одразу хибна. Це корисно, наприклад, для меню, яке має показатись хоча б раз.

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

4. Цикли: for та foreach

4.1. for

Цикл for найзручніший, коли заздалегідь відома кількість ітерацій. Він складається з трьох частин, розділених крапкою з комою: ініціалізація лічильника, умова продовження, дія після кожної ітерації.

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

// Зворотний відлік
for ($i = 10; $i > 0; $i--) {
    echo $i . " ";
}
```

4.2. foreach

Цикл foreach створений спеціально для перебору масивів (і об'єктів, що реалізують Iterator) — він автоматично проходить по всіх елементах без потреби вручну керувати індексом і лічильником. Це найпоширеніший спосіб ітерації масивів у PHP.

```
<?php
$fruits = ["яблуко", "банан", "вишня"];

foreach ($fruits as $fruit) {
    echo $fruit . "\n";
}

// З ключем (індексом)
foreach ($fruits as $index => $fruit) {
    echo "$index: $fruit\n";
}
```

> *for підходить, коли важлива саме числова послідовність (наприклад, обхід із кроком 2 або у зворотному напрямку), а foreach — коли потрібно просто обробити кожен елемент колекції незалежно від індексації.*

5. Керування циклами: break і continue

break негайно завершує виконання циклу (або switch) і передає керування коду після нього. continue пропускає решту поточної ітерації й переходить одразу до перевірки умови наступної ітерації, не завершуючи цикл повністю.

```
<?php
// break — зупиняє цикл повністю
foreach ([1, 2, 3, 4, 5] as $num) {
    if ($num === 3) {
        break;
    }
    echo $num . " ";
}
// Виведе: 1 2

// continue — пропускає лише цю ітерацію
foreach ([1, 2, 3, 4, 5] as $num) {
    if ($num % 2 === 0) {
        continue;   // пропустити парні числа
    }
    echo $num . " ";
}
// Виведе: 1 3 5
```

> *У вкладених циклах break 2 (або continue 2) дозволяє перервати одразу два рівні вкладеності — число вказує, скільки циклів «нагору» потрібно розірвати. Цим варто користуватись обережно: часто зрозуміліше винести логіку в окрему функцію з return.*

6. Масиви в PHP: види та створення

6.1. Індексовані та асоціативні масиви

У PHP існує лише один тип масиву — упорядкований map (словник), який може використовуватись і як індексований (числові ключі за замовчуванням від 0), і як асоціативний (довільні ключі-рядки), і навіть як їх суміш.

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

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

6.2. Багатовимірні масиви

Елементом масиву може бути інший масив — так утворюються багатовимірні структури, зручні для представлення таблиць, дерев і вкладених об'єктів (наприклад, результату запиту до бази даних із кількома рядками).

```
<?php
$students = [
    ["name" => "Іван", "grade" => 92],
    ["name" => "Марія", "grade" => 87],
];

echo $students[0]["name"];   // Іван
echo $students[1]["grade"];  // 87
```

6.3. Способи створення масивів

Спосіб	Приклад
Короткий синтаксис []	$arr = [1, 2, 3];
Функція array()	$arr = array(1, 2, 3);
range()	$arr = range(1, 10); // [1,2,...,10]
array_fill()	$arr = array_fill(0, 5, 0); // п'ять нулів

7. Ітерація масиву

7.1. foreach — основний спосіб

Як уже показано в розділі 4, foreach — найзручніший і найбезпечніший спосіб пройтись по всіх елементах масиву. Для асоціативних масивів він природно повертає і ключ, і значення.

```
<?php
$user = ["name" => "Оксана", "age" => 25, "city" => "Львів"];

foreach ($user as $key => $value) {
    echo "$key: $value\n";
}
// name: Оксана
// age: 25
// city: Львів
```

7.2. Ітерація за посиланням

Якщо потрібно змінити елементи масиву безпосередньо під час ітерації, у foreach можна передати значення за посиланням (&$value) — тоді зміни всередині циклу одразу відображаються в оригінальному масиві.

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

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

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

> *unset($num) після foreach за посиланням — не примха, а необхідність: якщо цього не зробити, змінна $num продовжує "вказувати" на останній елемент масиву, і випадкове повторне використання імені $num в наступному коді може непомітно зіпсувати дані масиву.*

7.3. Цикл for з масивом

Для індексованих масивів іноді зручніше використовувати for разом із count() — особливо якщо потрібен доступ до сусідніх елементів (наприклад, порівняння поточного з наступним).

```
<?php
$scores = [85, 92, 78, 95];
```

```
for ($i = 0; $i < count($scores); $i++) {
    echo "Оцінка " . ($i + 1) . ": " . $scores[$i] . "\n";
}
```

8. Редагування елементів масиву

8.1. Додавання елементів

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

8.2. Видалення елементів

Операція	Приклад
unset() — видалити за ключем	unset($arr["key"]);
array_pop() — видалити останній	$last = array_pop($arr);
array_shift() — видалити перший	$first = array_shift($arr);

```
<?php
$cart = ["хліб", "молоко", "яйця"];

$cart[] = "сир";              // додати в кінець → [хліб, молоко, яйця,
сир]
array_unshift($cart, "вода"); // додати на початок → [вода, хліб, молоко,
яйця, сир]

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

$lastItem = array_pop($cart);  // видалити й повернути останній елемент
```

> *Після unset() за числовим ключем у масиві залишається "дірка" в нумерації (ключі не переіндексовуються автоматично) — якщо важлива безперервна послідовність індексів 0,1,2..., після видалення варто викликати array_values($arr).*

8.3. Зміна існуючих елементів

Зміна значення за відомим ключем — найпростіша операція: достатньо звернутись до елемента за індексом і присвоїти нове значення, як до звичайної змінної.

```
<?php
$prices = ["хліб" => 25, "молоко" => 30];
$prices["хліб"] = 27;          // змінити ціну хліба
```

```
$prices["молоко"] += 5;        // підвищити ціну молока на 5
```

9. Маніпуляція масивом: сортування

Функція	Що робить
sort($arr)	Сортує за зростанням, переіндексовує ключі
rsort($arr)	Сортує за спаданням, переіндексовує ключі
asort($arr)	Сортує за значенням, зберігає ключі
ksort($arr)	Сортує за ключами (за зростанням)
usort($arr, $fn)	Сортує за власною функцією порівняння

```
<?php
$numbers = [5, 2, 8, 1, 9];
sort($numbers);
print_r($numbers);   // [1, 2, 5, 8, 9]

// Сортування асоціативного масиву за власним правилом
$students = [
    ["name" => "Іван", "grade" => 78],
    ["name" => "Марія", "grade" => 92],
];
usort($students, function($a, $b) {
    return $b["grade"] <=> $a["grade"];   // за спаданням оцінки
});
```

10. Маніпуляція масивом: функціональна обробка та корисні функції

10.1. array_map, array_filter, array_reduce

Ці три функції — основа функціонального стилю роботи з масивами: замість ручних циклів вони застосовують передану функцію (callback) до кожного елемента, дозволяючи трансформувати, відфільтрувати або згорнути масив в одне значення.

Функція	Призначення
array_map($fn, $arr)	Застосовує функцію до кожного елемента, повертає новий масив тієї ж довжини
array_filter($arr, $fn)	Залишає лише елементи, для яких функція повернула true
array_reduce($arr, $fn, $init)	Згортає масив в одне підсумкове значення

```php
<?php
$prices = [100, 250, 75, 300];

// array_map: додати ПДВ до кожної ціни
$withTax = array_map(fn($p) => $p * 1.2, $prices);
// [120, 300, 90, 360]

// array_filter: залишити тільки ціни понад 100
$expensive = array_filter($prices, fn($p) => $p > 100);
// [250, 300] (ключі 1 і 3 збережено!)

// array_reduce: підсумувати всі ціни
$total = array_reduce($prices, fn($carry, $p) => $carry + $p, 0);
// 725
```

> *array_filter() зберігає оригінальні ключі елементів, що залишились — якщо потрібна послідовна нумерація 0,1,2..., після фільтрації варто викликати array_values(). Стрілкові функції fn() => ... (з PHP 7.4) — короткий синтаксис анонімних функцій, зручний саме для таких одноразових колбеків.*

10.2. Об'єднання, зрізи, пошук

Функція	Призначення
array_merge($a, $b)	Об'єднує масиви в один
array_slice($arr, $offset, $len)	Вирізає частину масиву (не змінює оригінал)
array_splice($arr, $offset, $len)	Вирізає й замінює частину масиву (змінює оригінал)

Функція	Призначення
in_array($value, $arr)	Перевіряє наявність значення в масиві
array_search($value, $arr)	Повертає ключ знайденого значення
array_keys($arr) / array_values($arr)	Повертає всі ключі / усі значення масиву
array_unique($arr)	Видаляє дублікати значень
array_reverse($arr)	Обертає порядок елементів
count($arr)	Кількість елементів у масиві

```php
<?php
$a = [1, 2, 3];
$b = [4, 5, 6];
$merged = array_merge($a, $b);        // [1,2,3,4,5,6]

$fruits = ["яблуко", "банан", "вишня", "груша"];
$slice = array_slice($fruits, 1, 2);  // ["банан", "вишня"]

if (in_array("банан", $fruits)) {
    echo "Банан є в списку";
}

$key = array_search("вишня", $fruits); // 2
```

> *array_slice() повертає новий масив і не змінює оригінал; array_splice() навпаки — модифікує оригінальний масив на місці й може одночасно видаляти та вставляти елементи. Плутанина між ними — поширена помилка на практиці, тому варто завжди звіряти з документацією, якщо є сумнів.*

Питання для самоперевірки

- Чим match відрізняється від switch? Наведіть перевагу match у конкретній ситуації.
- Поясніть різницю між while і do-while. Наведіть приклад, де важливо, щоб тіло циклу виконалось хоча б раз.
- Коли доцільніше використовувати for, а коли foreach для перебору масиву?
- Що робить break 2 у вкладених циклах? Чому варто застосовувати його обережно?
- Чим індексований масив відрізняється від асоціативного? Наведіть приклад багатовимірного масиву.
- Для чого потрібен unset($value) після foreach з ітерацією за посиланням?
- Чим array_pop() відрізняється від array_shift()? Наведіть приклад, де вони дають різний результат.
- Опишіть, чим відрізняються array_map(), array_filter() і array_reduce() — яке завдання вирішує кожна?
- Чим array_slice() відрізняється від array_splice()?
- Як переіндексувати масив після видалення елементів через unset()?

Короткий глосарій

Термін	Означення
Fall-through	Провалювання виконання switch у наступний case через відсутність break
Callback	Функція, що передається як аргумент іншій функції для виклику всередині неї
Асоціативний масив	Масив, елементи якого адресуються довільними ключами-рядками
Ітерація	Послідовний прохід по всіх елементах колекції (масиву)
Стрілкова функція (arrow function)	Короткий синтаксис анонімної функції fn($x) => ...
Переіндексація	Оновлення числових ключів масиву в послідовний ряд 0,1,2...